Document Title	AUTOSAR Specification of ICU Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	023
Document Classification	Standard

Document Version	3.3.0
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change Description
28.02.2014	3.3.0	AUTOSAR Release Management	• Changed Icu Edge Counting Requirement • Editorial changes • Removed chapter(s) on change documentation
05.06.2012	3.2.0	AUTOSAR Administration	• Updated description of Icu_IndexType
27.04.2011	3.1.0	AUTOSAR Administration	• Added requirements ICU251 and ICU252 • Update Chapter 8 and 10 • Legal disclaimer revised
23.06.2008	3.0.1	AUTOSAR Administration	Legal disclaimer revised
07.12.2007	3.0.0	AUTOSAR Administration	• The code file structure of the module was completely reworked. • The following requirements were added: ICU088, ICU220, ICU221, ICU228 and ICU229. • The flow charts related to the ECU Wake-Up moved to the • SWS document of the ECU State Manager. • Document meta information extended • Small layout adaptations made
28.04.2006	2.0.0	AUTOSAR Administration	Document structure adapted to common Release 2.0 SWS Template. • Major changes in chapter 10 • Structure of document changed partly • Other changes see chapter 11

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
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

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 starting edge of the signal to 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 a handle with a name. With this handle each channel and its related properties can be found within the configuration structure. In “C” programming language this can be realized e.g. by #defines and enums.
Wakeup event	A wakeup event is understood as a pattern of edges, which will lead to the wake up of this driver. Nevertheless the decision whether a pattern is valid or not isn’t done by this driver. This shall be done by an upper layer.

3 Related documentation

3.1 Input documents

[1] General Requirements on Basic Software Modules,
AUTOSAR_SRS_General.pdf

[2] General Requirements on SPAL,
AUTOSAR_SRS_SPAL_General.pdf

[3] Specification of Standard Types,
AUTOSAR_SWS_StandardTypes.pdf

[4] List of Basic Software Modules,
AUTOSAR_BasicSoftwareModules.pdf

[5] Specification of Diagnostics Event Manager (DEM),
AUTOSAR_SWS_DEM.pdf

[6] Specification of Development Error Tracer,
AUTOSAR_SWS_Development_Error_Tracer.pdf

[7] Requirements on ICU Driver,
AUTOSAR_SRS_ICU_Driver.pdf

[8] Specification of ECU Configuration,
AUTOSAR_ECU_Configuration.pdf

[9] Layered Software Architecture,
AUTOSAR_LayeredSoftwareArchitecture.pdf

[10] Specification of ECU State Manager,
AUTOSAR_SWS_EcuStateManager.pdf

[11] AUTOSAR Basic Software Module Description Template,
AUTOSAR_BSW_Module_Description.pdf

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

ICU219: The code file structure shall be as follows:

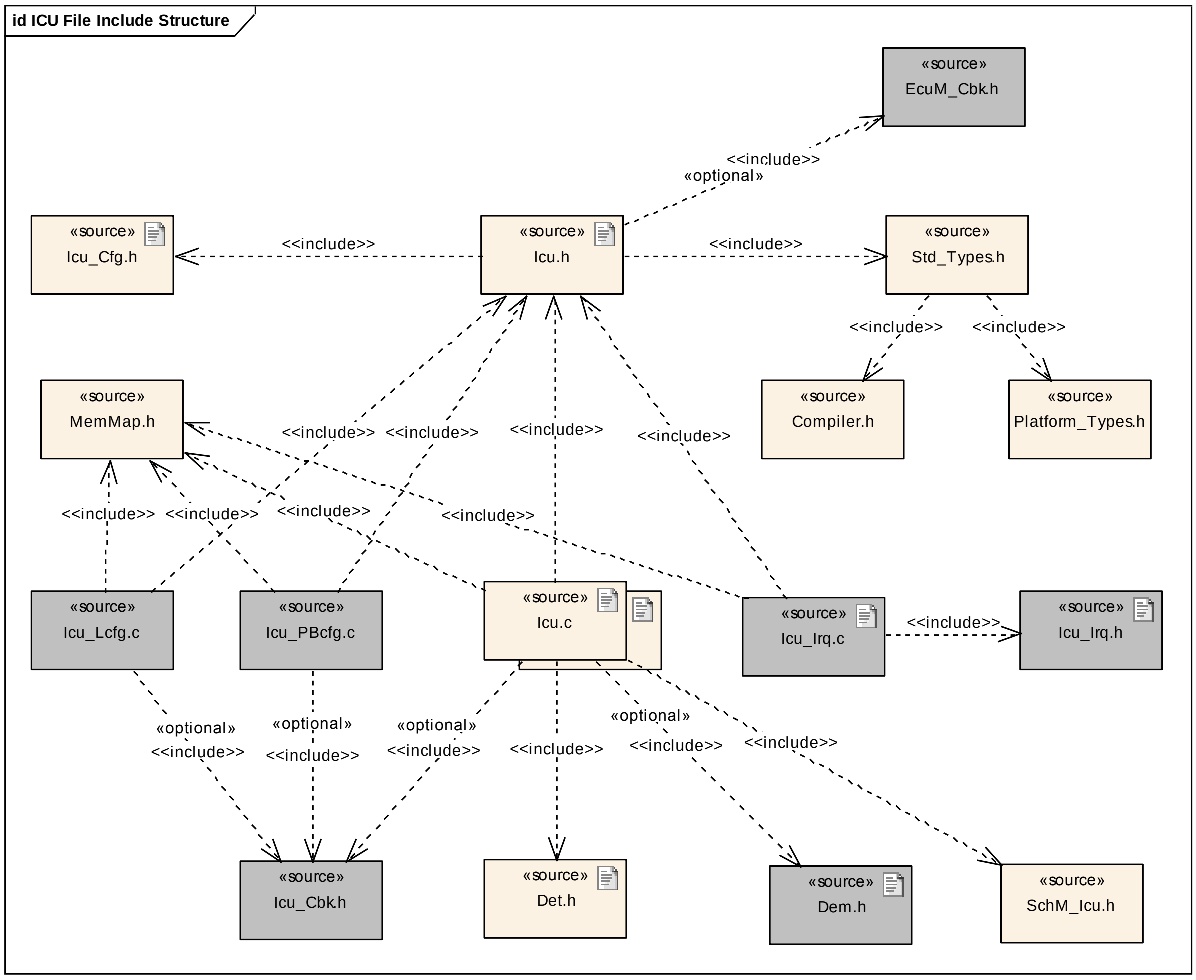


Figure 5.1: Header file structure

- Icu.c shall include Icu.h
- Icu.h shall include Icu_Cfg.h for the API pre-compiler switches.
- Icu_Xcfg.h where X is a placeholder for 'L' or 'PB'

Icu.c has access to the Icu_Cfg.h via the implicitly included Icu.h file.

Icu_Irq.c shall include Icu.h for the function which shall be called in the interrupt function.and Icu_Irq.h for the declaration of interrupt functions.

The Type definitions for Icu_Lcfg.c and Icu_PBcfg.c are located in the file Icu_Cfg.h. or Icu.h.

Rather the implicit include of Icu_Cfg.h via Icu.h in the files Icu_Lcfg.c and Icu_PBcfg.c is necessary to solve the following construct:

`Icu.h` shall include `EcuM_Cbk.h`, if wakeup functionality is configured.

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

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

`Icu_Lcfg.c` shall include I`cu_Cbk.h` for a link time configuration if the call back function is linked to the module via the ROM structure.

Icu_PBcfg.c shall include I`cu_Cbk.h` for post build time configuration if the call back function is linked to the module via the ROM structure.

`Icu.c` shall include I`cu_Cbk.h` for pre-compile time configuration

ICU116: The module shall optionally include the Dem.h file if any production error will be issued by the implementation. 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)

Requirement	***Satisfied by***
[**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)
[**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	ICU187
[**BSW00360**] Parameters of callback functions	Not applicable (non-functional requirement)
[**BSW00361**] Compiler specific language extension header	Not applicable (non-functional requirement)

Requirement	***Satisfied by***
[**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 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

Requirement	***Satisfied by***
[**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	ICU182, ICU183
[**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)
[**BSW005]** No hard coded horizontal interfaces within MCAL	Not applicable (non-functional requirement)
[**BSW006]** Platform independency	Not applicable (non-functional requirement)

Requirement	***Satisfied by***
[**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 9.3
[**BSW00415**] User dependent include files	Not applicable (this is a basic software module)

Requirement	***Satisfied by***
[**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 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 status	Not applicable (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 9.3, ICU055, ICU056, ICU057
[**BSW12075**] Use of application buffers	ICU063ICU110
[**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	**6.1 Chapter 10.3 “Published Information** ”

Requirement	***Satisfied by***
[**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	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: The Icu module functions for different channel numbers shall be re-entrant, except for:

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

ICU149: The Icu module's environment shall check the integrity if several calls for the same ICU channel are used during runtime in different tasks or ISRs.

ICU150: The Icu module shall not check the integrity if several calls for the same ICU channel are used during runtime in different tasks or ISRs.

The Icu module has 2 modes: `ICU_MODE_NORMAL` and `ICU_MODE_SLEEP`.

ICU011: In `ICU_MODE_NORMAL` mode all notifications are available as

- configured by service `Icu_SetActivationCondition() or IcuDefaultStartEdge.`
- 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.

ICU088: The module Icu shall allow the configuration per channel of the definition on which edge the period starts.

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 Icu module shall avoid the integration of incompatible files by implementating the following 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 are 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.

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
API Icu_Init service called when the ICU driver and the Hardware are already initialized	Development	ICU_E_ALREADY_INITIALIZED	0x17
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 `IcuDevErrorDetect` shall activate or deactivate the detection of all development errors.

ICU112: If the switch `IcuDevErrorDetect` 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.

ICU048: If development error detection for the Icu module is enabled: All Icu module functions shall skip functionality and return without any action (except for raising the development error) if a development error is detected.

ICU022: If development error detection for the Icu module is enabled: All Icu module functions, except for Icu_Init and Icu_GetVersionInfo, shall raise development error ICU_E_UNINIT when the function Icu_Init has not been called.

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 `IcuDevErrorDetect` is set (see ICU026)

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

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

ICU190:

Module	***Imported Type***
Dem	Dem_EventIdType
EcuM	EcuM_WakeupSourceType
Std_Types	Std_VersionInfoType

8.2 Type definitions

8.2.1 Icu_ModeType

Name:	`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

Name:	`Icu_ChannelType`		
Type:	`Unsigned Integer`		
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

Name:	`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

Name:	`Icu_ConfigType`
Type:	`Structure`

Range:	--	Hardware and implementation dependent structure. The contents of the initialization data structure are microcontroller specific.
Description:	This type contains initialization data.	

ICU038: The Icu_ConfigType shall contain the following initialization data.
- 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 within the Icu_ConfigType 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

Name:	`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

Name:	`Icu_ValueType`		
Type:	`Unsigned Integer`		
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

Name:	`Icu_DutyCycleType`		
Type:	`Structure`		
Element:	`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

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

8.2.9 Icu_EdgeNumberType

Name:	`Icu_EdgeNumberType`		
Type:	`Unsigned Integer`		
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

Name:	`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

Name:	`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

Name:	`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

ICU191:

Service name:	Icu_Init	

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

Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfigPtr	Pointer to a selected configuration structure
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function initializes the driver.	

ICU006: The function Icu_Init shall initialize all relevant registers of the configured hardware with the values of the structure referenced by the parameter `ConfigPtr`.

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

- **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 start-up code.
- **ICU129**: All other registers shall be initialized by the startup code.

ICU061: The function Icu_Init shall disable all notifications.

ICU121: The function Icu_Init shall disable the wakeup-capability of all channels .

ICU040: The function Icu_Init shall set all used ICU channels to status ICU_IDLE.

ICU060: The function Icu_Init shall set the module mode to ICU_MODE_NORMAL.

ICU054: The function Icu_Init shall only set the resources that are configured in the configuration file (including clearing of pending interrupt flags).

ICU151: The Icu module's environment shall not call Icu_Init during a running operation (e. g. timestamp measurement or edge counting).

ICU023: If development error detection for the Icu module is enabled: The function Icu_Init shall check the parameter `ConfigPtr` for not being NULL and shall raise the development error code `ICU_E_PARAM_CONFIG` if the check fails.

ICU220: If development error detection for the ICU module is enabled and the function Icu_Init is called when the ICU driver and hardware are already initialized, the function Icu_Init shall raise development error ICU_E_ALREADY_INITIALIZED and return without any action.

ICU048 applies to the function Icu_Init.

8.3.2 Icu_DeInit

ICU193:

Service name:	Icu_DeInit

Syntax:	`void Icu_DeInit(`
	
	`)`

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

ICU035: The function Icu_DeInit shall de-initialize the ICU module.

ICU036: The function Icu_DeInit shall set the state of the peripherals used by configuration as the same after power on reset. Values of registers which are not writeable are excluded.

ICU091: The function Icu_DeInit shall influence only the peripherals which are allocated by static configuration and/or the runtime configuration set passed by the previous call of Icu_Init().

ICU037: The function Icu_DeInit shall disable all used interrupts and notifications.

ICU152: The Icu module's environment shall not call Icu_DeInit during a running operation (e. g. timestamp measurement or edge counting)

ICU092: The function Icu_DeInit shall be pre compile time configurable `On/Off` by configuration parameter IcuDeInitApi.

ICU221: A re-initialization of the ICU module by executing the Icu_Init() function requires a de-initialization before by executing the Icu_DeInit() function.

ICU022 and ICU048 apply to the function Icu_DeInit.

8.3.3 Icu_SetMode

ICU194:

Service name:	Icu_SetMode	

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

Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Mode	ICU_MODE_NORMAL: Normal operation, all used interrupts are enabled according to the notification requests. ICU_MODE_SLEEP: Reduced power mode. In sleep mode only those notifications are available which are configured as wakeup capable.
Parameters (inout):	None	

Parameters (out):	None
Return value:	None
Description:	This function sets the ICU mode.

ICU008: The function Icu_SetMode shall set the operation mode to the given mode parameter.

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

ICU095: The function Icu_SetMode shall be pre-compile time configurable `On/Off` by the configuration parameter IcuSetModeApi.

ICU125:If development error detection is enabled for the module Icu: the function Icu_SetMode shall check the parameter `Mode` and shall raise the error `ICU_E_PARAM_MODE` if the parameter `Mode` is not within the allowed range set in the configuration.
ICU133: This service can be called during running operations. If so, an ongoing operation that generates interrupts on a wakeup capable channel like e.g. time stamping or edge counting might lead to the ICU module not being able to properly enter sleep mode. This is then a system or ECU configuration issue not a problem of this specification.

ICU022 and ICU048 apply to the function Icu_SetMode.

8.3.4 Icu_DisableWakeup

ICU195:

Service name:	Icu_DisableWakeup	

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

Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (limited according to ICU050)	
Parameters (in):	Channel	Numeric identifier of the ICU channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function disables the wakeup capability of a single ICU channel.	

ICU013: The function Icu_DisableWakeup shall disable the wakeup capability of a single ICU channel but only for ICU channels configured statically as wakeup capable true.

ICU096: The function Icu_DisableWakeup shall be pre compile time configurable `On/Off` by the configuration parameter IcuDisableWakeupApi.

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

ICU024: If development error detection is enabled: The function Icu_DisableWakeup shall check the parameter `Channel` and shall raise development error `ICU_E_PARAM_CHANNEL` if `Channel` is not within the allowed range set in the configuration..

ICU059: If development error detection is enabled: The function Icu_DisableWakeup shall check the parameter `Channel`. The function Icu_DisableWakeup shall raise development error `ICU_E_PARAM_CHANNEL` if `Channel` is indexing an ICU channel statically not configured as wakeup capable.

ICU022 and ICU048 apply to the function Icu_DisableWakeup.

8.3.5 Icu_EnableWakeup

ICU196:

Service name:	Icu_EnableWakeup	

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

Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (limited according to ICU050)	
Parameters (in):	Channel	Numeric identifier of the ICU channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function (re-)enables the wakeup capability of the given ICU channel.	

ICU014: The function Icu_EnableWakeup shall re-enable the wakeup capability of a single ICU channel for the following ICU mode selection(s). This service is only feasible for ICU channels configured as wakeup capable true.

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

ICU097: The function Icu_EnableWakeup shall be pre compile time configurable `On/Off` by configuration parameter IcuEnableWakeupApi.

ICU155: If development error detection is enabled: The function Icu_EnableWakeup shall check the parameter `Channel` and shall raise the error `ICU_E_PARAM_CHANNEL` if `Channel` is invalid.

ICU156: If development error detection is enabled: The function Icu_EnableWakeup shall check the parameter `Channel`. The function Icu_EnableWakeup shall raise the error ICU_E_PARAM_CHANNEL if Channel is indexing an ICU channel statically not configured as wakeup capable.

ICU022 and ICU048 apply to the function Icu_EnableWakeup.

8.3.6 Icu_SetActivationCondition

ICU197:

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

ICU090: The function Icu_SetActivationCondition shall set the activation-edge according to Activation parameter for the given channel. This service shall support channels which are configured for the following 8.2.10
:

- ICU_MODE_SIGNAL_EDGE_DETECT
- ICU_MODE_TIMESTAMP
- ICU_MODE_EDGE_COUNTER

ICU139: The function Icu_SetActivationCondition shall reset the state for the given channel to ICU_IDLE.

ICU159: If development error detection is enabled: The function Icu_SetActivationCondition shall check the parameter Channel and shall raise the error ICU_E_PARAM_CHANNEL if Channel is not within the range set in the configuration..

ICU043: If development error detection is enabled: The function Icu_SetActivationCondition shall check the parameter Activation. The function Icu_SetActivationCondition shall raise the error ICU_E_PARAM_ACTIVATION if Activation is invalid but only for the requested ICU channel.

ICU022 and ICU048 apply to the function Icu_SetActivationCondition.

8.3.7 Icu_DisableNotification

ICU198:

Service name:	Icu_DisableNotification	
Syntax:	`void Icu_DisableNotification( Icu_ChannelType Channel )`	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (limited according to ICU050)	
Parameters (in):	Channel	Numeric identifier of the ICU channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function disables the notification of a channel.	

ICU009: The function Icu_DisableNotification shall disable the notification on the given channel.

ICU251: If IcuWakeupCapability is TRUE and Icu_DisableNotification is called, ICU driver shall call the API EcuM_CheckWakeup().
This is done to avoid possible loss of wakeup events during a shutdown sequence.

ICU160: If development error detection is enabled: The function Icu_DisableNotification shall check the parameter `Channel` and shall raise the error `ICU_E_PARAM_CHANNEL` if `Channel` is invalid (invalid identifier).

ICU022 and ICU048 apply to the function Icu_DisableNotification.

8.3.8 Icu_EnableNotification

ICU199:

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

ICU010: The function Icu_EnableNotification shall enable the notification on the given channel.

ICU127: The function Icu_EnableNotification shall cancel pending interrupts in case the channel is configured for edge detection functionality.

ICU252: If IcuWakeupCapability is TRUE and Icu_EnableNotification is called, ICU driver shall call the API EcuM_CheckWakeup().
This is done to avoid possible loss of wakeup events during a shutdown sequence.

ICU161: If development error detection is enabled: The function Icu_EnableNotification shall check the parameter `Channel` and shall raise the error `ICU_E_PARAM_CHANNEL` if `Channel` is invalid (invalid identifier).

ICU022 and ICU048 apply to the function Icu_EnableNotification.

8.3.9 Icu_GetInputState

ICU200:

Service name:	Icu_GetInputState	

Syntax:	`Icu_InputStateType Icu_GetInputState(`
	`    Icu_ChannelType Channel`
	`)`	

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

ICU030: The function Icu_GetInputState shall return the status of the ICU input. Only channels which are configured for the following IcuMeasurementMode shall be supported:
- `ICU_MODE_SIGNAL_EDGE_DETECT`
- `ICU_MODE_SIGNAL_MEASUREMENT`

ICU031: If an activation edge has been detected the function Icu_GetInputState shall return ICU_ACTIVE for Edge Detection channels. For Signal Measurement a channel should be set to ICU_ACTIVE not until this measurement has completed and the driver is able to provide useful information on the input signal.

ICU032: Once the function Icu_GetInputState has returned the status ICU_ACTIVE, the function Icu_GetInputState shall set the stored status to ICU_IDLE until the next edge is detected.

ICU033: If no activation edge has been detected the function Icu_GetInputState shall return ICU_IDLE.

ICU122: The function Icu_GetInputState shall be pre compile time configurable On/Off by the configuration parameter IcuGetInputStateApi.

ICU162: If development error detection is enabled: The function Icu_GetInputState shall check the parameter `Channel` and shall raise the error `ICU_E_PARAM_CHANNEL` if `Channel` is invalid (invalid identifier or channel not configured for modes `ICU_MODE_SIGNAL_EDGE_DETECT or ICU_MODE_SIGNAL_MEASUREMENT)`

ICU049: If development error detection is enabled: The function Icu_GetInputState shall return ICU_IDLE if an error is detected.

ICU022 and ICU048 apply to the function Icu_GetInputState.

8.3.10 Icu_StartTimestamp

ICU201:

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

ICU063: The function Icu_StartTimestamp shall start 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 Icu module shall start at the beginning of the buffer.

ICU065: If linear buffer handling is configured, when the capture functionality reaches the end of the buffer, the Icu module shall stop capturing timer values.

ICU134: The Icu module shall only call a notification function if the following facts are given:

- A notification function is configured
- The notification has been enabled by the call of Icu_EnableNotification()
- NotifyInterval > 0
- The number of events specified by NotifyInterval has been captured

ICU066: The function Icu_StartTimeStamp shall only be available in Measurement Mode "ICU_MODE_TIMESTAMP".

ICU098: The function Icu_StartTimestamp shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_TIMESTAMP_API`

ICU163: If development error detection is enabled: The function Icu_StartTimestamp shall check the parameter Channel and shall raise the error ICU_E_PARAM_CHANNEL if Channel is invalid (invalid identifier or channel not configured for mode ICU_MODE_TIMESTAMP).

ICU120: If development error detection is enabled: The function Icu_StartTimestamp shall check the parameter `BufferPtr`. The function Icu_StartTimestamp shall raise the error `ICU_E_PARAM_BUFFER_PTR` if `BufferPtr` is invalid (e.g. "0").

ICU108: If development error detection is enabled: The function Icu_StartTimestamp shall check the parameter `BufferSize` (check that size > 0). The function Icu_StartTimestamp shall raise the error `ICU_E_PARAM_BUFFER_SIZE` if `BufferSize` is invalid (e.g. "0").

ICU022 and ICU048 apply to the function Icu_StartTimestamp.

8.3.11 Icu_StopTimestamp

ICU202:

Service name:	Icu_StopTimestamp
Syntax:	`void Icu_StopTimestamp(`

	`Icu_ChannelType Channel`
	`)`	

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

ICU067: The function Icu_StopTimestamp shall stop the timestamp measurement of the given channel.

ICU168: In production mode the function Icu_StopTimestamp shall not return an error when the Channel is not active (has not started or has already stopped).

ICU165: The function Icu_StopTimestamp shall only be available in Measurement Mode: ICU_MODE_TIMESTAMP.

ICU099: The function Icu_StopTimestamp shall be pre compile time configurable `On/Off` by the configuration parameter: `IcuTimestampApi` (see also chapter 10.2.8. Configuration of optional API services)

ICU164: If development error detection is enabled: The function Icu_StopTimestamp shall check the parameter Channel and shall raise development error ICU_E_PARAM_CHANNEL if Channel is invalid (invalid identifier or channel not configured for mode ICU_MODE_TIMESTAMP)

ICU166: If development error detection is enabled: The function Icu_StopTimestamp shall raise development error `ICU_E_NOT_STARTED` if Channel is not active (has not started or is already stopped).

ICU022 and ICU048 apply to the function Icu_StopTimestamp.

8.3.12 Icu_GetTimestampIndex

ICU203:

Service name:	Icu_GetTimestampIndex	

Syntax:	`Icu_IndexType Icu_GetTimestampIndex(`
	`    Icu_ChannelType Channel`
	`)`	

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (limited according to ICU050)	
Parameters (in):	Channel	Numeric identifier of the ICU channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Icu_IndexType	Abstract return type to cover different microcontrollers.

Description:	This function reads the timestamp index of the given channel.

ICU071: The function Icu_GetTimestampIndex shall read the timestamp index of the given channel, which is the next to be written.

ICU135: The function Icu_GetTimestampIndex shall return “0” in case the service is called before Icu_StartTimestamp() (no buffer is defined in this case).

ICU170: The function Icu_GetTimestamp shall only be available in Measurement Mode ICU_MODE_TIMESTAMP.

ICU100: The function Icu_GetTimestamp shall be pre compile time configurable `On/Off` by the configuration parameter: `IcuTimestampApi`

ICU169: If development error detection is enabled: the function Icu_GetTimestamp shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode Icu_GetTimestamp), the function Icu_GetTimeStamp shall raise development error ICU_E_PARAM_CHANNEL.

ICU107: If development error detection is enabled: The function Icu_GetTimestampIndex shall return “0” if an error is detected.

ICU022 and ICU048 apply to the function Icu_GetTimestampIndex.

8.3.13 Icu_ResetEdgeCount

ICU204:

Service name:	Icu_ResetEdgeCount	

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

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

ICU072: The function Icu_ResetEdgeCount shall reset the value of the counted edges to zero.

ICU101: The function Icu_ResetEdgeCount shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_EDGE_COUNT_API`

ICU171: If development error detection is enabled: the function Icu_ResetEdgeCount shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode `ICU_MODE_EDGE_COUNTER`), then Icu_ResetEdgeCount shall raise development error ICU_E_PARAM_CHANNEL.

ICU022 and ICU048 apply to the function Icu_ResetEdgeCount.

8.3.14 Icu_EnableEdgeCount

ICU205:

Service name:	Icu_EnableEdgeCount	

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

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

ICU078:The function Icu_EnableEdgeCount shall enable the counting of edges of the given channel.

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

ICU073: The function Icu_EnableEdgeCount shall only count the configured[1] edges (rising edge / falling edge / both edges).

ICU074: The function IcuEnableEdgeCount shall be available for each ICU channel in Measurement Mode "Edge Counter".

ICU102: The function Icu_EnableEdgeCount shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_EDGE_COUNT_API`

ICU172: If development error detection is enabled, the function Icu_EnableEdgeCount shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode `ICU_MODE_EDGE_COUNTER`), then the function Icu_EnableEdgeCount shall raise development error ICU_E_PARAM_CHANNEL

ICU022 and ICU048 apply to the function Icu_EnableEdgeCount.

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

8.3.15 Icu_DisableEdgeCount

ICU206:

Service name:	Icu_DisableEdgeCount	

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

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

ICU079: The function Icu_DisableEdgeCount shall disable the counting of edges of the given channel.

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

ICU103: The function Icu_DisableEdgeCount shall be pre compile time configurable `On/Off` by the configuration parameter IcuEdgeCountApi.

ICU173: If development error detection is enabled: the function Icu_DisableEdgeCount shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode `ICU_MODE_EDGE_COUNTER`), the function Icu_DisableEdgeCount shall raise development error ICU_E_PARAM_CHANNEL.

ICU022 and ICU048 apply to the function Icu_DisableEdgeCount.

8.3.16 Icu_GetEdgeNumbers

ICU207:

Service name:	Icu_GetEdgeNumbers	

Syntax:	`Icu_EdgeNumberType Icu_GetEdgeNumbers(`
	`    Icu_ChannelType Channel`
	`)`	

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

ICU080: The function Icu_GetEdgeNumbers shall read the number of counted edges after the last call of Icu_ResetEdgeCount().

ICU104: The function Icu_GetEdgeNumbers shall be pre compile time configurable `On/Off` by the configuration parameter: `ICU_EDGE_COUNT_API`

ICU174: If development error detection is enabled, the function Icu_GetEdgeNumbers shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode `ICU_MODE_EDGE_COUNTER`), the function Icu_GetEdgeNumbers shall raise development error ICU_E_PARAM_CHANNEL.

ICU175: If development error detection is enabled: The function Icu_GetEdgeNumbers shall return "0" if an error is detected.

ICU022 and ICU048 apply to the function Icu_GetEdgeNumbers.

8.3.17 Icu_StartSignalMeasurement

ICU208:

Service name:	Icu_StartSignalMeasurement	

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

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

ICU140: The function Icu_StartSignalMeasurement shall start the measurement of signals beginning with the configured default start edge which occurs first after the call of this service.

ICU141: The function Icu_StartSignalMeasurement shall only be available in Measurement Mode "ICU_MODE_SIGNAL_MEASUREMENT".

ICU146: The function Icu_StartSignalMeasurment shall reset the state for the given channel to ICU_IDLE.

ICU142: The function Icu_StartSignalMeasurement shall be pre compile time configurable `On/Off` by the configuration parameter IcuSignalMeasurementApi.

ICU176: If development error detection is enabled, the function Icu_StartSignalMeasurement shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode

ICU_MODE_SIGNAL_MEASUREMENT), the function Icu_StartSignalMeasurement shall raise development error `ICU_E_PARAM_CHANNEL`.

ICU022 and ICU048 apply to the function Icu_StartSignalMeasurement.

8.3.18 Icu_StopSignalMeasurement

ICU209:

Service name:	Icu_StopSignalMeasurement	

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

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

ICU143: The function Icu_StopSignalMeasurement shall stop the measurement of signals of the given channel.

ICU144: The function Icu_StopSignalMeasurement shall only be available in Measurement Mode"ICU_MODE_SIGNAL_MEASUREMENT"

ICU145: The function Icu_StopSignalMeasurement shall be pre compile time configurable `On/Off` by the configuration parameter `IcuSignalMeasurementApi`

ICU177: If development error detection is enabled: the function Icu_StopSignalMeasurement shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode ICU_MODE_SIGNAL_MEASUREMENT), the function Icu_StopSignalMeasurement shall raise development error `ICU_E_PARAM_CHANNEL`.

ICU022 and ICU048 apply to the function Icu_StopSignalMeasurement.

8.3.19 Icu_GetTimeElapsed

ICU210:

Service name:	Icu_GetTimeElapsed

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

Service ID[hex]:	0x10
Sync/Async:	Synchronous

Reentrancy:	Reentrant (limited according to ICU050)	
Parameters (in):	Channel	Numeric identifier of the ICU channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Icu_ValueType	see Description
Description:	This function reads the elapsed Signal Low Time for the given channel.	

ICU081: The function Icu_GetTimeElapsed shall read the elapsed Signal Low Time for the given channel that is configured in Measurement Mode “Signal Measurement, Signal Low Time”. The elapsed time is measured between a falling edge and the consecutive rising edge of the channel.

ICU082: The function Icu_GetTimeElapsed shall read the elapsed Signal High Time for the given channel that is configured in Measurement Mode “Signal Measurement, Signal High Time”. The elapsed time is measured between a rising edge and the consecutive falling edge of the channel.

ICU083: The function Icu_GetTimeElapsed shall read the elapsed Signal Period Time for the given channel that is configured in Measurement Mode “Signal Measurement, Signal Period Time”. The elapsed time is measured between consecutive rising (or falling) edges of the channel. The period start edge is configurable.

ICU136: The function Icu_GetTimeElapsed shall return “0” in case
- no requested time has been captured (see Figure 9.19, letter “A”)
- the capturing of a requested time is ongoing and not finished (see Figure 9.19, letter “B”)
- a captured time was already returned once by this service and this service is called again (see Figure 9.19, letter “D”)

ICU105: The function Icu_GetTimeElapsed shall be pre compile time configurable `On/Off` by the configuration parameter IcuGetTimeElapsedApi.

ICU178: 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.

ICU179: If development error detection is enabled and an error is detected this service shall return `“0”`.

ICU022 and ICU048 apply to the function Icu_GetTimeElapsed.

8.3.20 Icu_GetDutyCycleValues

ICU211:

Service name:	Icu_GetDutyCycleValues

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

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

ICU084: The function Icu_GetDutyCycleValues shall read the coherent active time and period time for the given ICU Channel, if it is configured in Measurement Mode “Signal Measurement, Duty Cycle Values”.
ICU137: The function Icu_GetDutyCycleValues shall return “0” in case

- no coherent active- and period time has been captured (similar to Figure 9.19, letter “A”)
- the capturing of a requested high- and period time is ongoing and not finished (meant: the function shall return “0” until the first valid value has been captured and the captured value shall be stored until a new value is captured) (similar to Figure 9.19, letter “B”)
- captured duty cycle values were already returned once by this service and this service is called again (similar to Figure 9.19, letter “D”)

ICU106: The function Icu_GetDutyCycleValues shall be pre compile time configurable `On/Off` by the configuration parameter IcuGetDutyCycleValuesApi.

ICU180: If development error detection is enabled: the function Icu_GetDutyCycleValues shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode `ICU_MODE_SIGNAL_MEASUREMENT`, Duty Cycle Values), the function Icu_GetDutyCycleValues shall raise development error `ICU_E_PARAM_CHANNEL`.

ICU181: If development error detection is enabled, the function Icu_GetDutyCycleValues shall check the parameter `DutyCycleValues`. If `DutyCycleValues` is invalid, the function Icu_GetDutyCycleValues shall raise development error `ICU_E_PARAM_BUFFER_PTR`.

ICU022 and ICU048 apply to the function Icu_GetDutyCycleValues.

8.3.21 Icu_GetVersionInfo

ICU212:

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 (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	This function returns the version information of this module.	

ICU182: The function Icu_GetVersionInfo shall return the version information of this module. The version information includes:

- Module Id (See Literature [4])
- Vendor Id
- Vendor specific version numbers.

ICU183: The Icu module's environment may call the function Icu_GetVersionInfo at any time.

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.

ICU094: The function Icu_GetVersionInfo shall be pre compile time configurable On/Off by the configuration parameter IcuGetVersionInfoApi.

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.

ICU213:

API function	***Description***
Dem_ReportErrorStatus	Reports errors to the DEM.
Det_ReportError	Service to report development errors.
EcuM_ValidateWakeupEvent	After wakeup, the ECU State Manager will stop the process during the WAKEUP VALIDATION state to wait for validation of the wakeup event. The validation is carried out with a call to this API service.

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

- **ICU055**: the static configuration parameter IcuReportWakeupSource is set to "ON"
- **ICU056**: the module is in mode ICU_MODE_SLEEP
- **ICU057**: a wakeup event occurs on a wakeup capable ICU channel.

ICU228: EcuM_CheckWakeup shall be called within the Interrupt Service Routine servicing the ICU channel wakeup event on wakeup-capable channel.

ICU229: The ISR's, providing the wakeup events, shall be responsible for resetting the interrupt flags if required by hardware.

8.6.3 Configurable interfaces

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

ICU119: The ISRs shall reset the interrupt flags (if needed by hardware) and call the corresponding notification functions.

ICU018: The Icu 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.

ICU187: The Icu module's notification functions shall have no parameters and no return value.

ICU214:

Service name:	Icu_SignalNotification_<Channel>

Syntax:	`void Icu_SignalNotification_<Channel>(`
	
	`)`

Sync/Async:	Synchronous
Reentrancy:	Reentrancy of interface not relevant for this module. (in general it is in this case

	not reentrant).
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	According to the last call of Icu_EnableNotification, this notification function to be called if the requested signal edge (rising / falling / both edges) occurs (once per edge).

ICU021: According to the last call of Icu_EnableNotification(), the Icu module shall call the notification function Icu_SignalNotification_<Channel> if the requested signal edge (rising / falling / both edges) occurs (once per edge).

ICU044: Only those edge notifications shall be provided, which are supported by hardware.

ICU042: After a call of Icu_DisableNotification ,the Icu module shall not call the the notification function Icu_SignalNotification_<Channel>.

ICU215:

Service name:	Icu_TimestampNotification_<Channel>

Syntax:	`void Icu_TimestampNotification_<Channel>(`
	
	`)`

Sync/Async:	Synchronous
Reentrancy:	Reentrancy of interface not relevant for this module. (in general it is in this case not reentrant).
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This notification to be called if the number of requested timestamps (Notification interval > 0) are acquired and if the notification has been enabled by the call of Icu_EnableNotification().

ICU216: The Icu module shall call the notification Icu_TimestampNotification_<Channel> if the number of requested timestamps (Notification interval > 0) are acquired and if the notification has been enabled by the call of Icu_EnableNotification().

ICU217: After a call of Icu_DisableNotification the Icu module shall NOT call the notification Icu_TimestampNotification_<Channel>.

ICU218: The Icu module’s notification Icu_TimestampNotification_<Channel> depends on pre-processor switch `IcuTimestampApi`

9 Sequence diagrams

9.1 Icu_Init

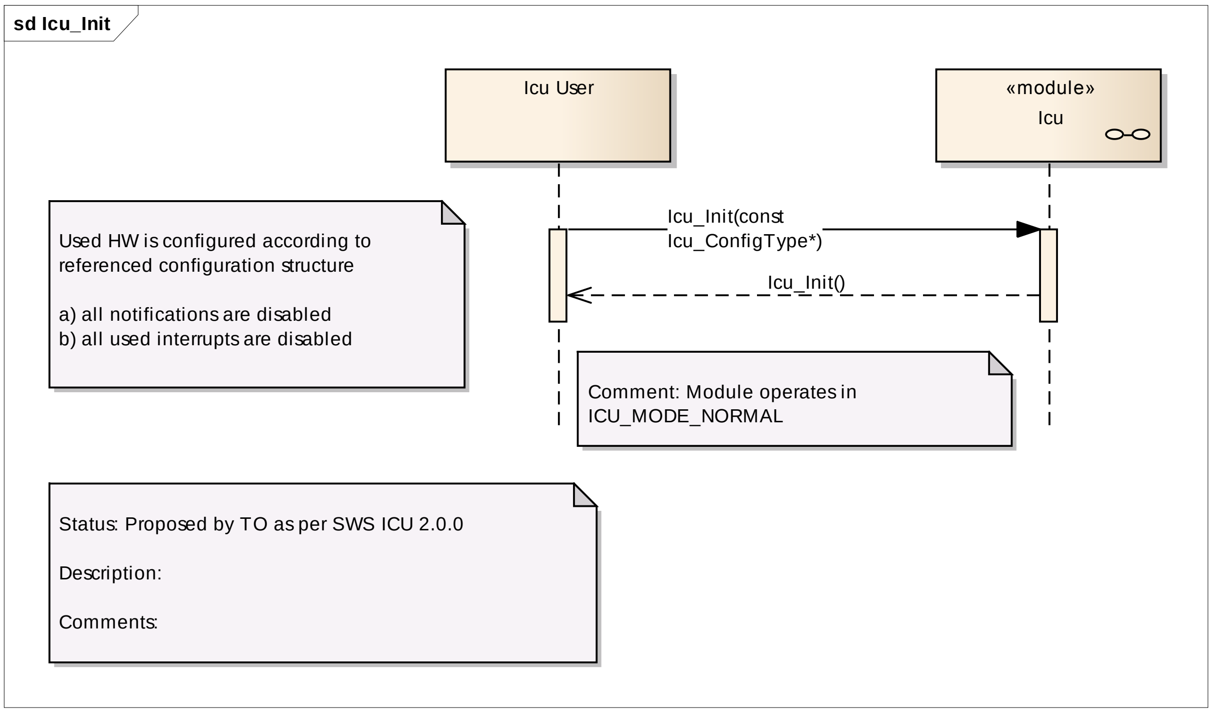

sd Icu_Init
Icu User
«module»
Icu
Icu_Init(const Icu_ConfigType*)
Icu_Init()
Used HW is configured according to referenced configuration structure
a) all notifications are disabled
b) all used interrupts are disabled
Comment: Module operates in ICU_MODE_NORMAL
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

Figure 9.1: Initialization of the ICU driver

9.2 Icu_DeInit

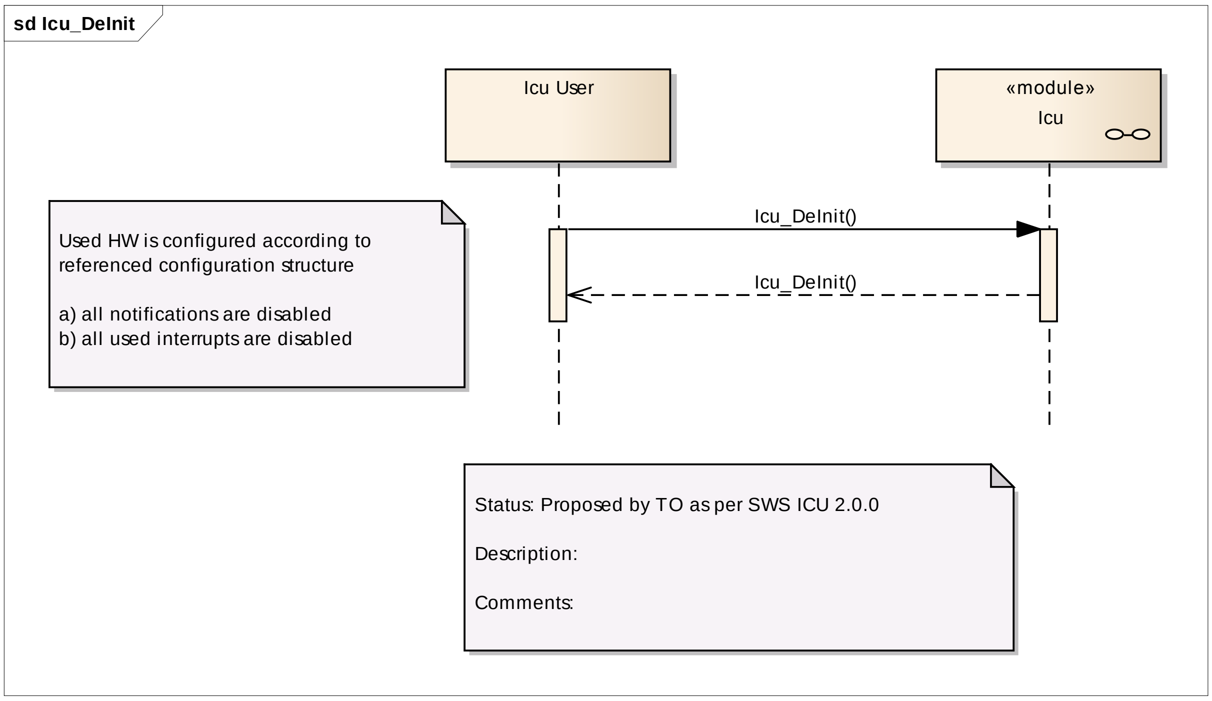

sd Icu_DeInit
Icu User
«module»
Icu
Icu_DeInit()
Icu_DeInit()
Used HW is configured according to referenced configuration structure
a) all notifications are disabled
b) all used interrupts are disabled
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

Figure 9.2: De-Initialization of the ICU driver

9.3 Validate Wakeup Events

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

9.4 Icu_SetMode

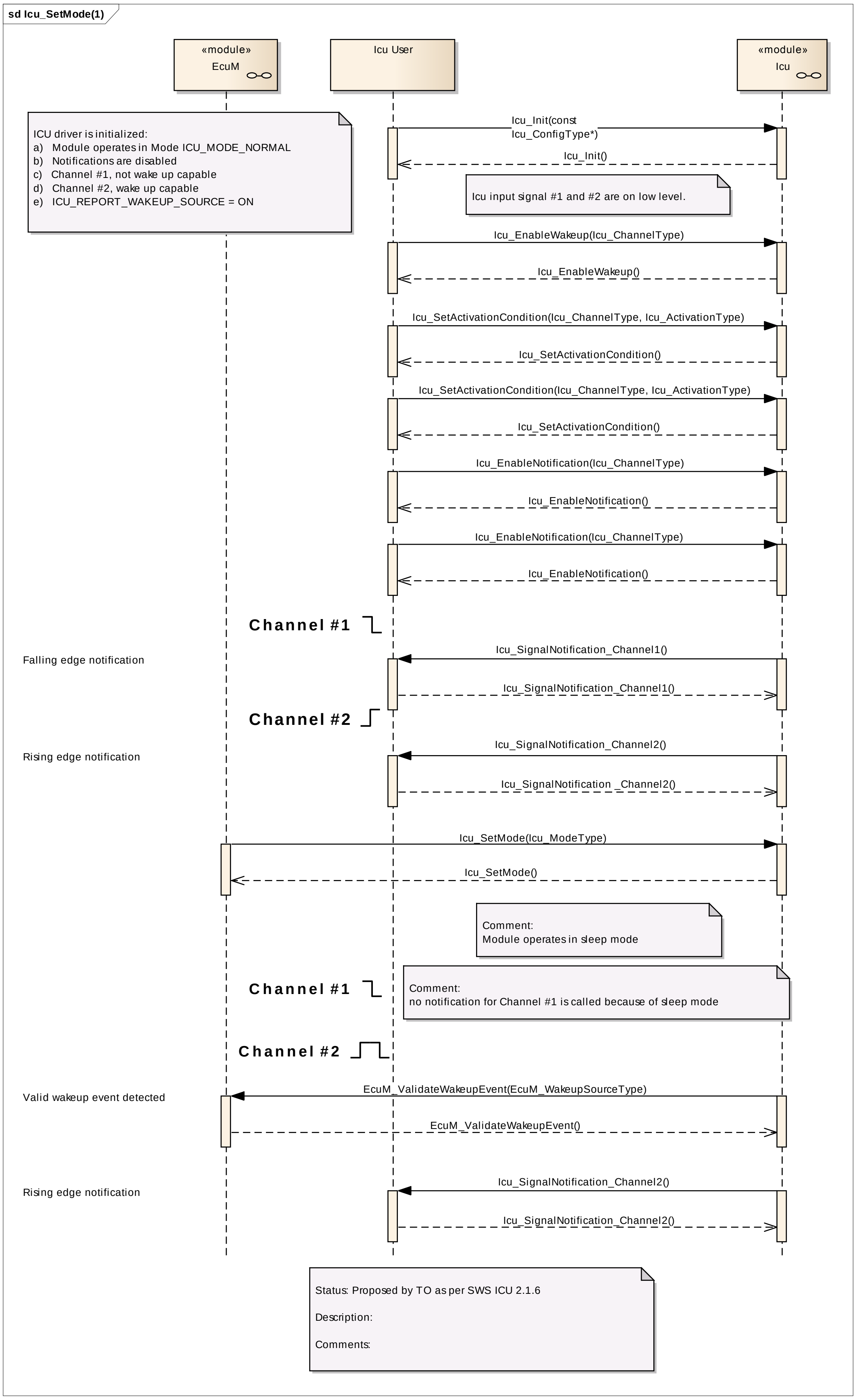

sd Icu_SetMode(1)
«module» EcuM
Icu User
«module» Icu
ICU driver is initialized:
a) Module operates in Mode ICU_MODE_NORMAL
b) Notifications are disabled
c) Channel #1, not wake up capable
d) Channel #2, wake up capable
e) ICU_REPORT_WAKEUP_SOURCE = ON
Icu_Init(const Icu_ConfigType*)
Icu_Init()
Icu input signal #1 and #2 are on low level.
Icu_EnableWakeup(Icu_ChannelType)
Icu_EnableWakeup()
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Channel #1
Falling edge notification
Icu_SignalNotification_Channel1()
Icu_SignalNotification_Channel1()
Channel #2
Rising edge notification
Icu_SignalNotification_Channel2()
Icu_SignalNotification _Channel2()
Icu_SetMode(Icu_ModeType)
Icu_SetMode()
Comment:
Module operates in sleep mode
Channel #1
Comment:
no notification for Channel #1 is called because of sleep mode
Channel #2
Valid wakeup event detected
EcuM_ValidateWakeupEvent(EcuM_WakeupSourceType)
EcuM_ValidateWakeupEvent()
Rising edge notification
Icu_SignalNotification_Channel2()
Icu_SignalNotification_Channel2()
Status: Proposed by TO as per SWS ICU 2.1.6
Description:
Comments:

Figure 9.3: Enabled notifications in SLEEP mode

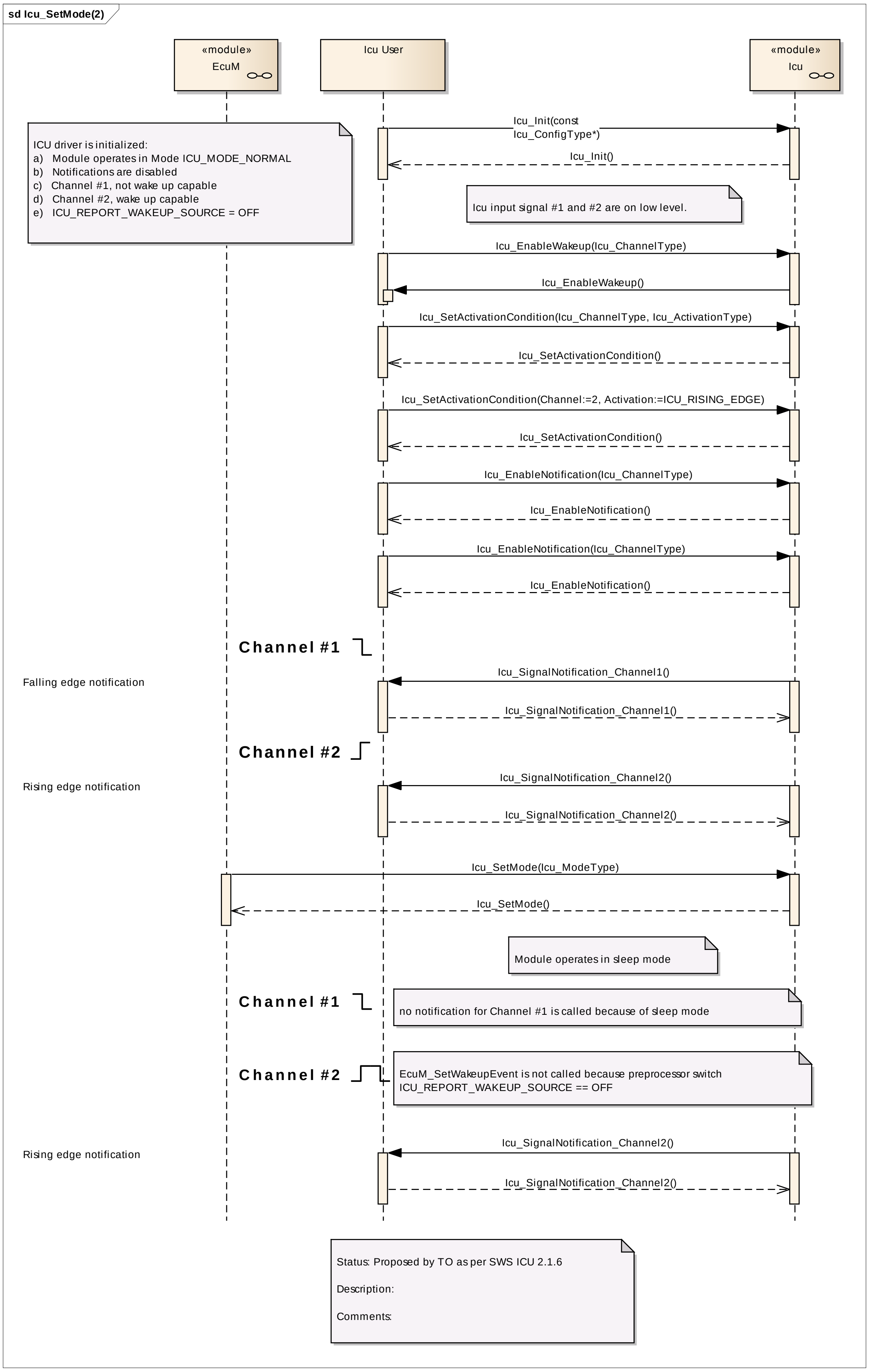

sd Icu_SetMode(2)
«module» EcuM
Icu User
«module» Icu
ICU driver is initialized:
a) Module operates in Mode ICU_MODE_NORMAL
b) Notifications are disabled
c) Channel #1, not wake up capable
d) Channel #2, wake up capable
e) ICU_REPORT_WAKEUP_SOURCE = OFF
Icu_Init(const Icu_ConfigType*)
Icu_Init()
Icu input signal #1 and #2 are on low level.
Icu_EnableWakeup(Icu_ChannelType)
Icu_EnableWakeup()
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Icu_SetActivationCondition(Channel:=2, Activation:=ICU_RISING_EDGE)
Icu_SetActivationCondition()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Channel #1
Falling edge notification
Icu_SignalNotification_Channel1()
Icu_SignalNotification_Channel1()
Channel #2
Rising edge notification
Icu_SignalNotification_Channel2()
Icu_SignalNotification_Channel2()
Icu_SetMode(Icu_ModeType)
Icu_SetMode()
Module operates in sleep mode
Channel #1
no notification for Channel #1 is called because of sleep mode
Channel #2
EcuM_SetWakeupEvent is not called because preprocessor switch ICU_REPORT_WAKEUP_SOURCE == OFF
Rising edge notification
Icu_SignalNotification_Channel2()
Icu_SignalNotification_Channel2()
Status: Proposed by TO as per SWS ICU 2.1.6
Description:
Comments:

Figure 9.4: Disabled reporting of wakeup sources in SLEEP mode

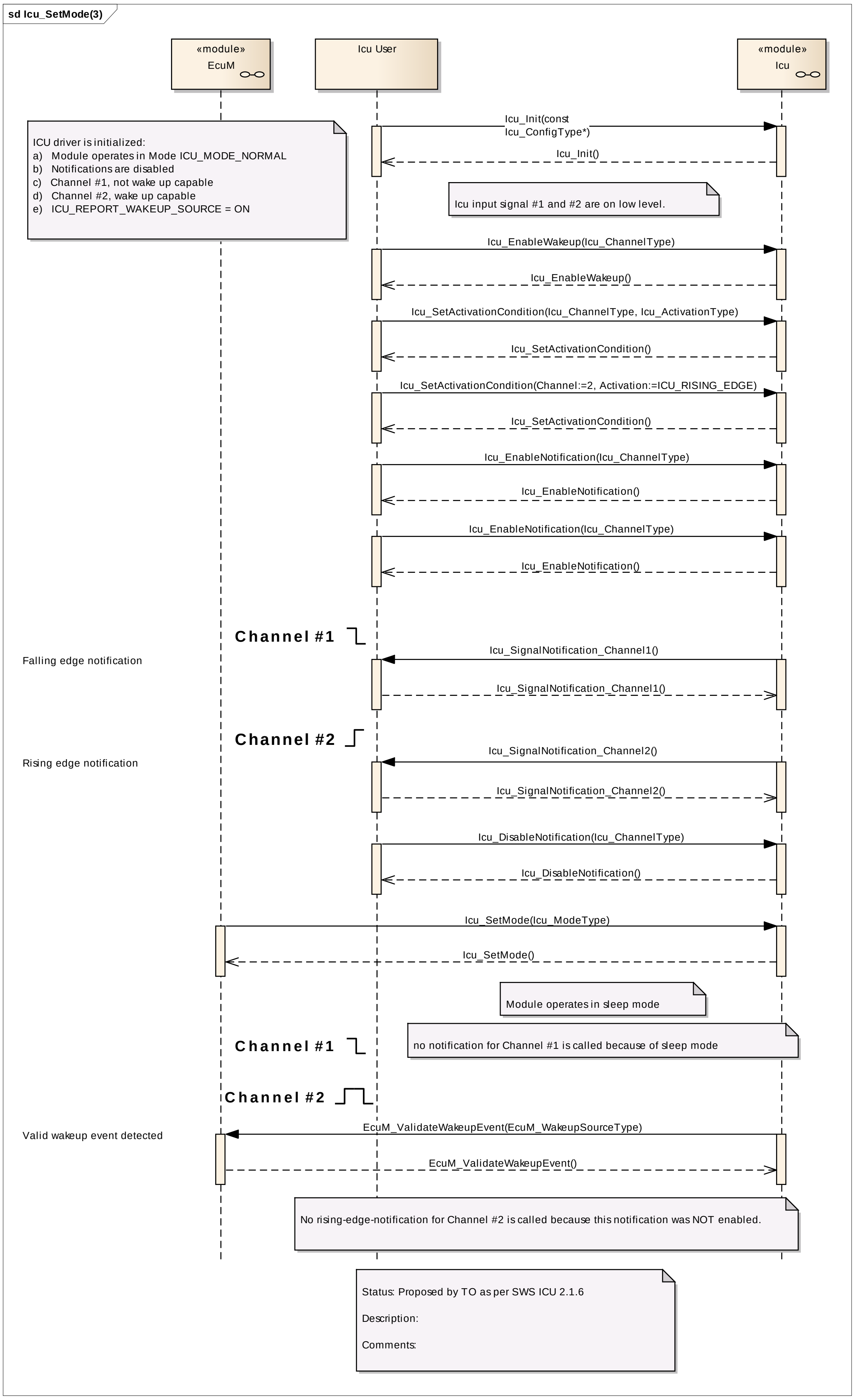

sd Icu_SetMode(3)
«module» EcuM
Icu User
«module» Icu
ICU driver is initialized:
a) Module operates in Mode ICU_MODE_NORMAL
b) Notifications are disabled
c) Channel #1, not wake up capable
d) Channel #2, wake up capable
e) ICU_REPORT_WAKEUP_SOURCE = ON
Icu_Init(const Icu_ConfigType*)
Icu_Init()
Icu input signal #1 and #2 are on low level.
Icu_EnableWakeup(Icu_ChannelType)
Icu_EnableWakeup()
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Icu_SetActivationCondition(Channel:=2, Activation:=ICU_RISING_EDGE)
Icu_SetActivationCondition()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Channel #1
Falling edge notification
Icu_SignalNotification_Channel1()
Icu_SignalNotification_Channel1()
Channel #2
Rising edge notification
Icu_SignalNotification_Channel2()
Icu_SignalNotification_Channel2()
Icu_DisableNotification(Icu_ChannelType)
Icu_DisableNotification()
Icu_SetMode(Icu_ModeType)
Icu_SetMode()
Module operates in sleep mode
Channel #1
no notification for Channel #1 is called because of sleep mode
Channel #2
Valid wakeup event detected
EcuM_ValidateWakeupEvent(EcuM_WakeupSourceType)
EcuM_ValidateWakeupEvent()
No rising-edge-notification for Channel #2 is called because this notification was NOT enabled.
Status: Proposed by TO as per SWS ICU 2.1.6
Description:
Comments:

Figure 9.5: Disabled edge notification in SLEEP mode

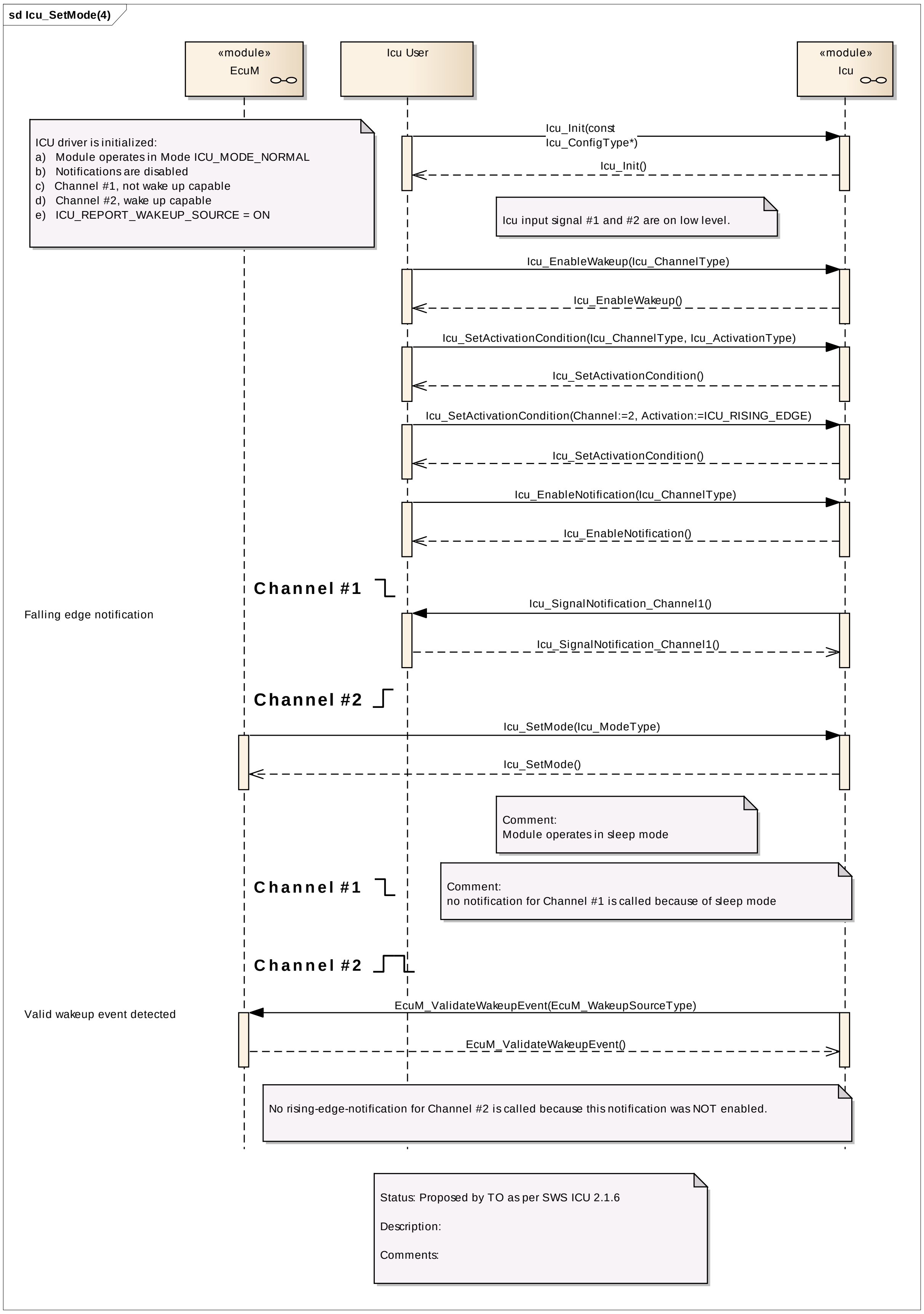

sd Icu_SetMode(4)
«module» EcuM
Icu User
«module» Icu
ICU driver is initialized:
a) Module operates in Mode ICU_MODE_NORMAL
b) Notifications are disabled
c) Channel #1, not wake up capable
d) Channel #2, wake up capable
e) ICU_REPORT_WAKEUP_SOURCE = ON
Icu_Init(const Icu_ConfigType*)
Icu_Init()
Icu input signal #1 and #2 are on low level.
Icu_EnableWakeup(Icu_ChannelType)
Icu_EnableWakeup()
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Icu_SetActivationCondition(Channel:=2, Activation:=ICU_RISING_EDGE)
Icu_SetActivationCondition()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Channel #1
Falling edge notification
Icu_SignalNotification_Channel1()
Icu_SignalNotification_Channel1()
Channel #2
Icu_SetMode(Icu_ModeType)
Icu_SetMode()
Comment:
Module operates in sleep mode
Channel #1
Comment:
no notification for Channel #1 is called because of sleep mode
Channel #2
Valid wakeup event detected
EcuM_ValidateWakeupEvent(EcuM_WakeupSourceType)
EcuM_ValidateWakeupEvent()
No rising-edge-notification for Channel #2 is called because this notification was NOT enabled.
Status: Proposed by TO as per SWS ICU 2.1.6
Description:
Comments:

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

9.5 Icu_DisableWakeup

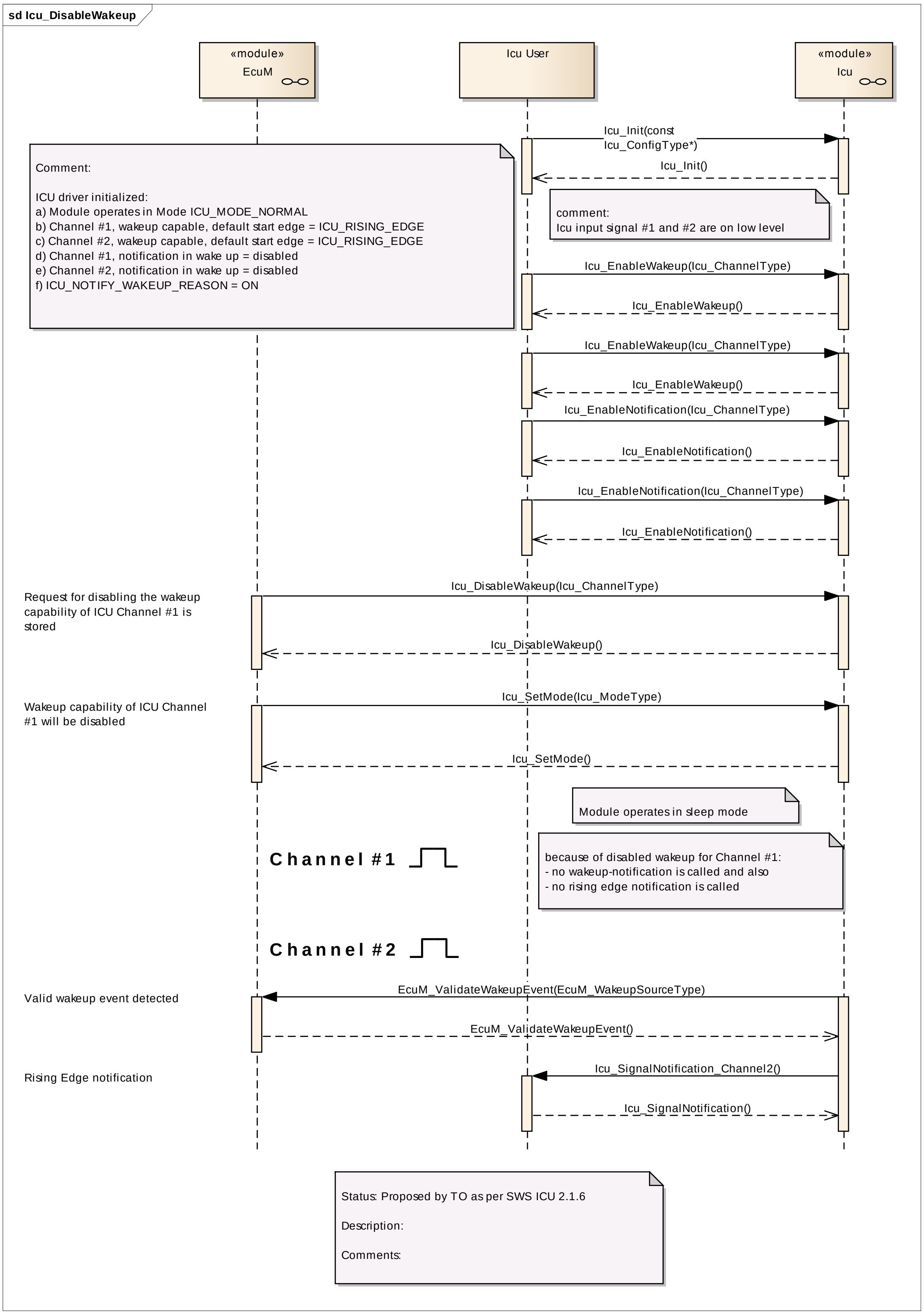

sd Icu_DisableWakeup
«module»
EcuM
Icu User
«module»
Icu
Icu_Init(const Icu_ConfigType*)
Icu_Init()
Comment:
ICU driver initialized:
a) Module operates in Mode ICU_MODE_NORMAL
b) Channel #1, wakeup capable, default start edge = ICU_RISING_EDGE
c) Channel #2, wakeup capable, default start edge = ICU_RISING_EDGE
d) Channel #1, notification in wake up = disabled
e) Channel #2, notification in wake up = disabled
f) ICU_NOTIFY_WAKEUP_REASON = ON
comment:
Icu input signal #1 and #2 are on low level
Icu_EnableWakeup(Icu_ChannelType)
Icu_EnableWakeup()
Icu_EnableWakeup(Icu_ChannelType)
Icu_EnableWakeup()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Icu_DisableWakeup(Icu_ChannelType)
Request for disabling the wakeup capability of ICU Channel #1 is stored
Icu_DisableWakeup()
Icu_SetMode(Icu_ModeType)
Wakeup capability of ICU Channel #1 will be disabled
Icu_SetMode()
Module operates in sleep mode
Channel #1
because of disabled wakeup for Channel #1:
- no wakeup-notification is called and also
- no rising edge notification is called
Channel #2
EcuM_ValidateWakeupEvent(EcuM_WakeupSourceType)
Valid wakeup event detected
EcuM_ValidateWakeupEvent()
Icu_SignalNotification_Channel2()
Rising Edge notification
Icu_SignalNotification()
Status: Proposed by TO as per SWS ICU 2.1.6
Description:
Comments:

Figure 9.7: Disabling of wakeup-capabilities

9.6 Icu_EnableWakeup

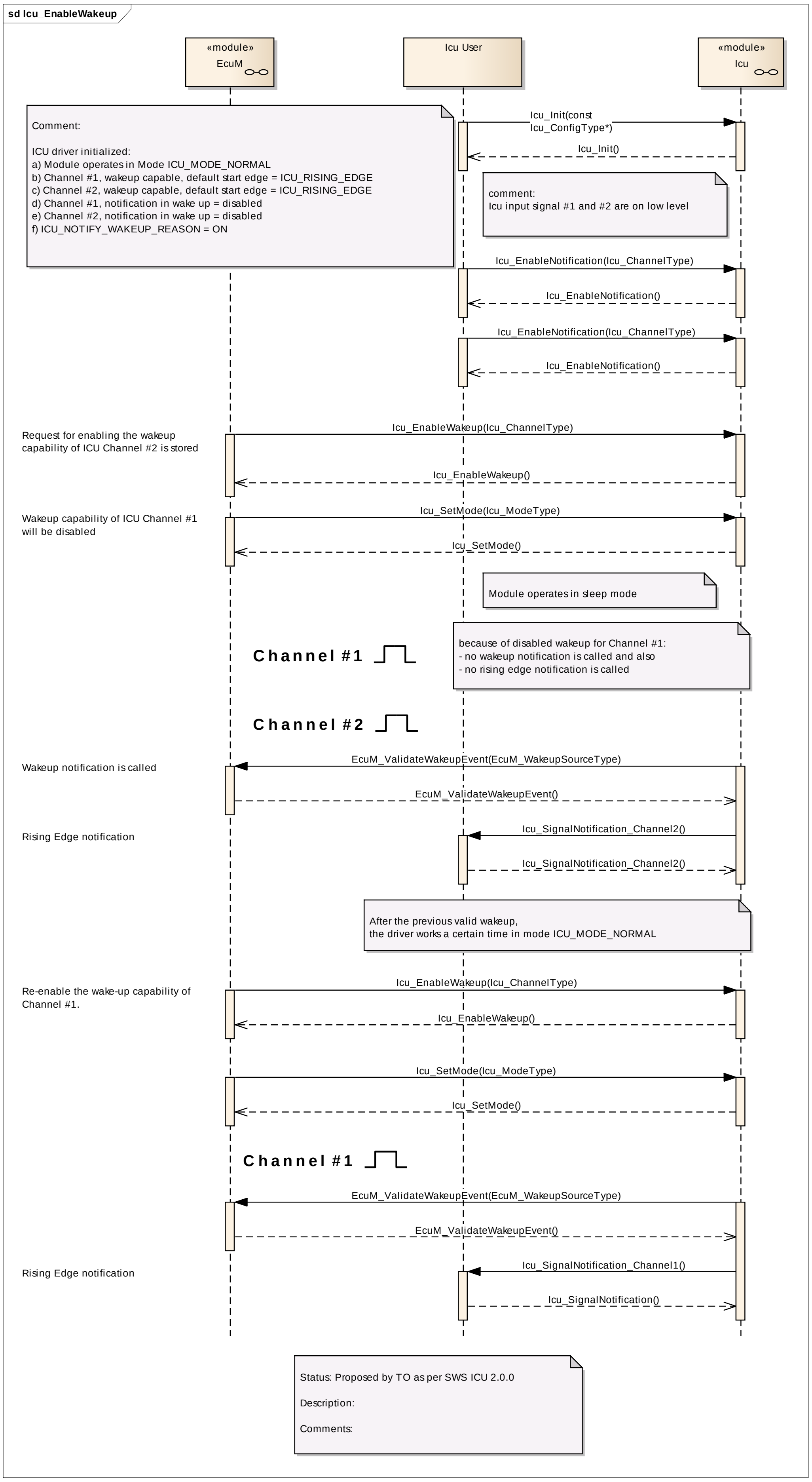

sd Icu_EnableWakeup
«module» EcuM
Icu User
«module» Icu
Comment:
ICU driver initialized:
a) Module operates in Mode ICU_MODE_NORMAL
b) Channel #1, wakeup capable, default start edge = ICU_RISING_EDGE
c) Channel #2, wakeup capable, default start edge = ICU_RISING_EDGE
d) Channel #1, notification in wake up = disabled
e) Channel #2, notification in wake up = disabled
f) ICU_NOTIFY_WAKEUP_REASON = ON
Icu_Init(const Icu_ConfigType*)
Icu_Init()
comment:
Icu input signal #1 and #2 are on low level
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Request for enabling the wakeup capability of ICU Channel #2 is stored
Icu_EnableWakeup(Icu_ChannelType)
Icu_EnableWakeup()
Wakeup capability of ICU Channel #1 will be disabled
Icu_SetMode(Icu_ModeType)
Icu_SetMode()
Module operates in sleep mode
Channel #1
because of disabled wakeup for Channel #1:
- no wakeup notification is called and also
- no rising edge notification is called
Channel #2
Wakeup notification is called
EcuM_ValidateWakeupEvent(EcuM_WakeupSourceType)
EcuM_ValidateWakeupEvent()
Rising Edge notification
Icu_SignalNotification_Channel2()
Icu_SignalNotification_Channel2()
After the previous valid wakeup,
the driver works a certain time in mode ICU_MODE_NORMAL
Re-enable the wake-up capability of Channel #1.
Icu_EnableWakeup(Icu_ChannelType)
Icu_EnableWakeup()
Icu_SetMode(Icu_ModeType)
Icu_SetMode()
Channel #1
EcuM_ValidateWakeupEvent(EcuM_WakeupSourceType)
EcuM_ValidateWakeupEvent()
Rising Edge notification
Icu_SignalNotification_Channel1()
Icu_SignalNotification()
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

Figure 9.8: Enabling of wakeup-capabilities

9.7 Icu_SetActivationCondition

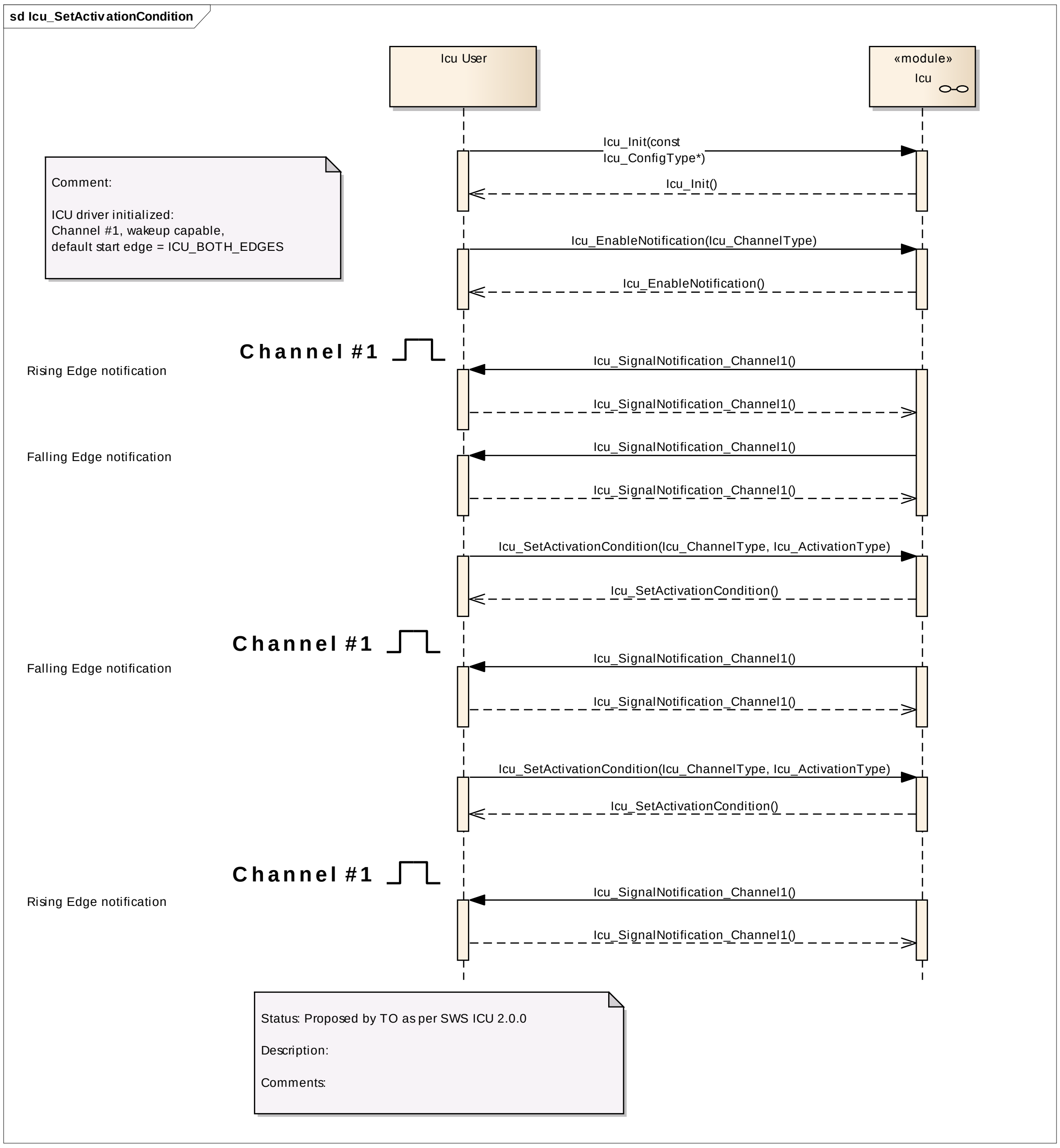

sd Icu_SetActivationCondition
Icu User
«module» Icu
Comment:
ICU driver initialized:
Channel #1, wakeup capable,
default start edge = ICU_BOTH_EDGES
Icu_Init(const Icu_ConfigType*)
Icu_Init()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Channel #1
Rising Edge notification
Icu_SignalNotification_Channel1()
Falling Edge notification
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

Figure 9.9: Setting up the activation condition for a channel

9.8 Icu_DisableNotification

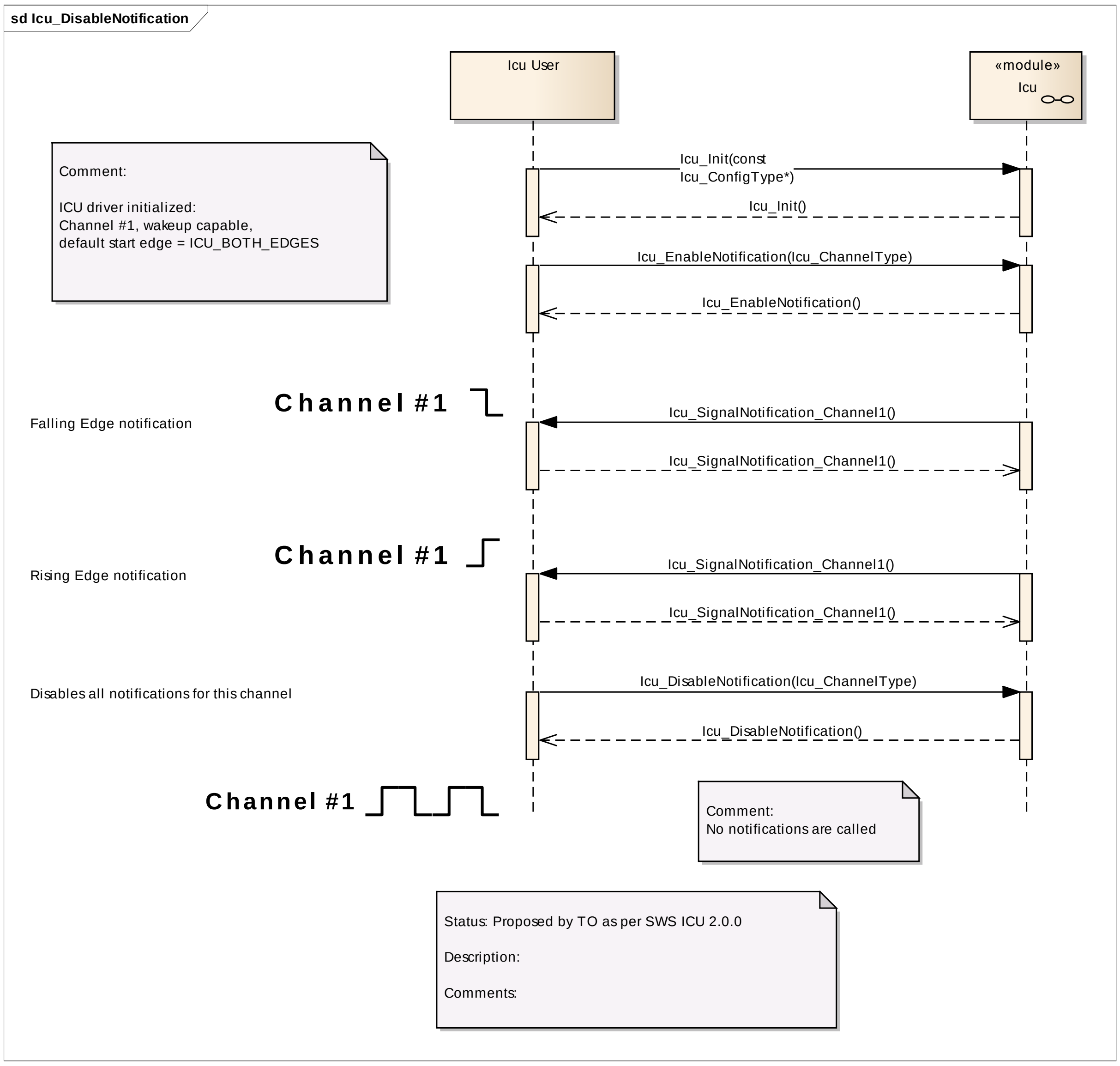

sd Icu_DisableNotification
Icu User
«module» Icu
Comment:
ICU driver initialized:
Channel #1, wakeup capable,
default start edge = ICU_BOTH_EDGES
Icu_Init(const Icu_ConfigType*)
Icu_Init()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Channel #1
Falling Edge notification
Icu_SignalNotification_Channel1()
Icu_SignalNotification_Channel1()
Channel #1
Rising Edge notification
Icu_SignalNotification_Channel1()
Icu_SignalNotification_Channel1()
Disables all notifications for this channel
Icu_DisableNotification(Icu_ChannelType)
Icu_DisableNotification()
Channel #1
Comment:
No notifications are called
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

Figure 9.10: Disabling of the notification for a channel

9.9 Icu_EnableNotification

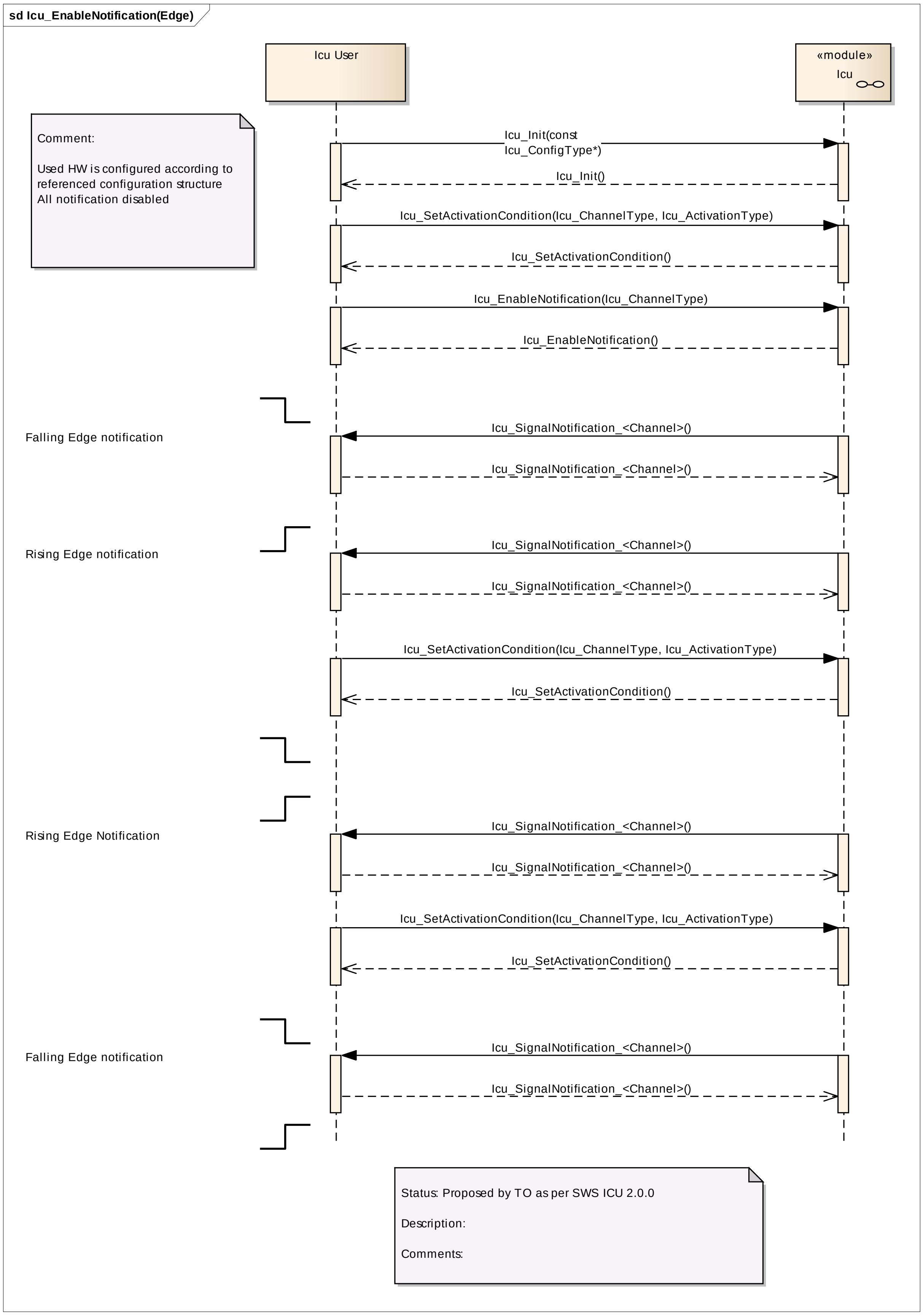

sd Icu_EnableNotification(Edge)
Icu User
«module»
Icu
Comment:
Used HW is configured according to referenced configuration structure
All notification disabled
Icu_Init(const Icu_ConfigType*)
Icu_Init()
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Falling Edge notification
Icu_SignalNotification_<Channel>()
Icu_SignalNotification_<Channel>()
Rising Edge notification
Icu_SignalNotification_<Channel>()
Icu_SignalNotification_<Channel>()
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Rising Edge Notification
Icu_SignalNotification_<Channel>()
Icu_SignalNotification_<Channel>()
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Falling Edge notification
Icu_SignalNotification_<Channel>()
Icu_SignalNotification_<Channel>()
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

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

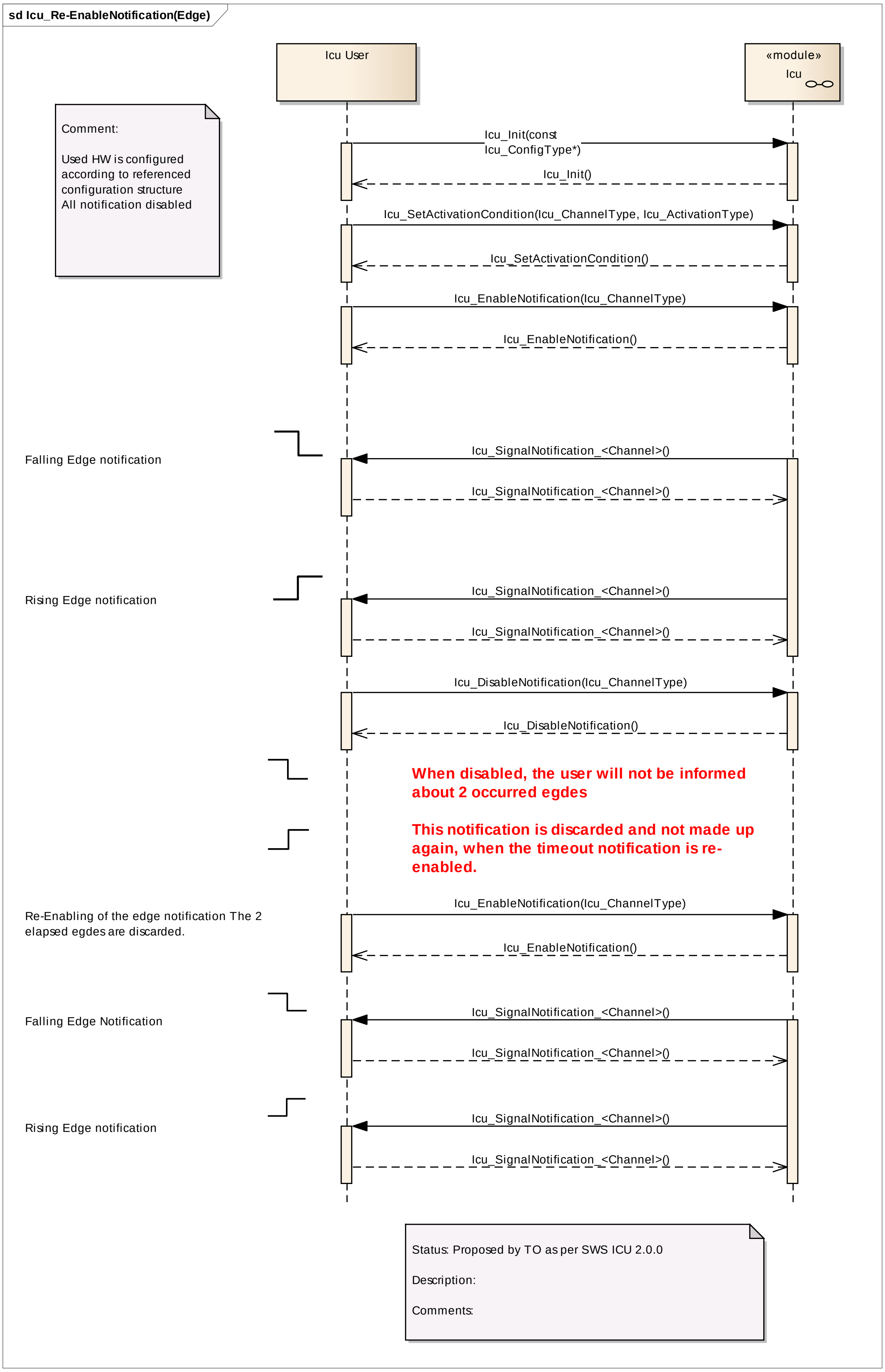

sd Icu_Re-EnableNotification(Edge)
Icu User
«module» Icu
Comment:
Used HW is configured according to referenced configuration structure
All notification disabled
Icu_Init(const Icu_ConfigType*)
Icu_Init()
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Falling Edge notification
Icu_SignalNotification_<Channel>()
Icu_SignalNotification_<Channel>()
Rising Edge notification
Icu_SignalNotification_<Channel>()
Icu_SignalNotification_<Channel>()
Icu_DisableNotification(Icu_ChannelType)
Icu_DisableNotification()
When disabled, the user will not be informed about 2 occurred egdes
This notification is discarded and not made up again, when the timeout notification is re-enabled.
Re-Enabling of the edge notification The 2 elapsed egdes are discarded.
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Falling Edge Notification
Icu_SignalNotification_<Channel>()
Icu_SignalNotification_<Channel>()
Rising Edge notification
Icu_SignalNotification_<Channel>()
Icu_SignalNotification_<Channel>()
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

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

9.10 Icu_GetInputState

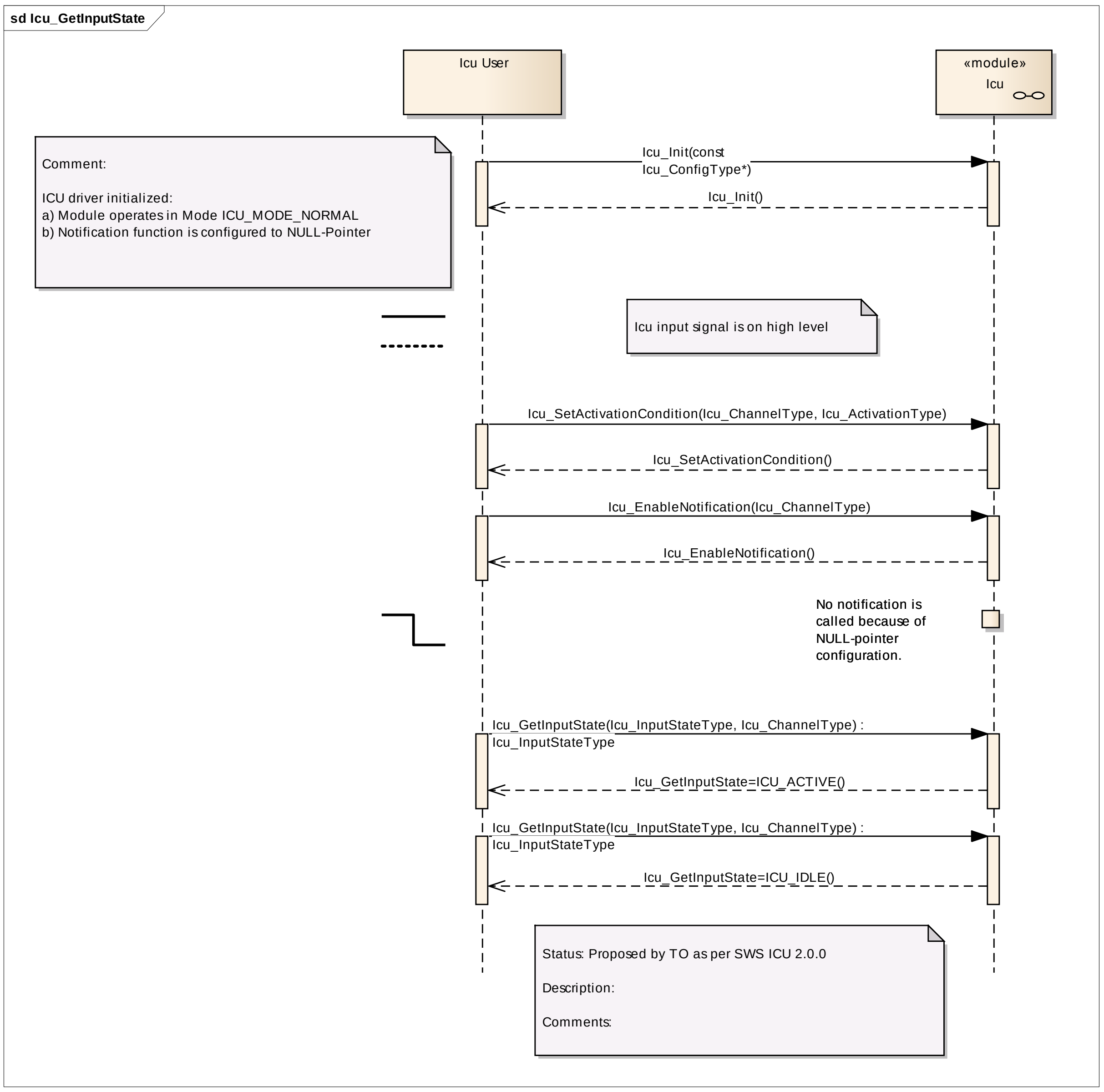

sd Icu_GetInputState
Icu User
«module»
Icu
Comment:
ICU driver initialized:
a) Module operates in Mode ICU_MODE_NORMAL
b) Notification function is configured to NULL-Pointer
Icu_Init(const
Icu_ConfigType*)
Icu_Init()
Icu input signal is on high level
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
No notification is called because of NULL-pointer configuration.
Icu_GetInputState(Icu_InputStateType, Icu_ChannelType) : Icu_InputStateType
Icu_GetInputState=ICU_ACTIVE()
Icu_GetInputState(Icu_InputStateType, Icu_ChannelType) : Icu_InputStateType
Icu_GetInputState=ICU_IDLE()
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

Figure 9.13: Polling of the channel status

9.11 Icu Timestamping

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

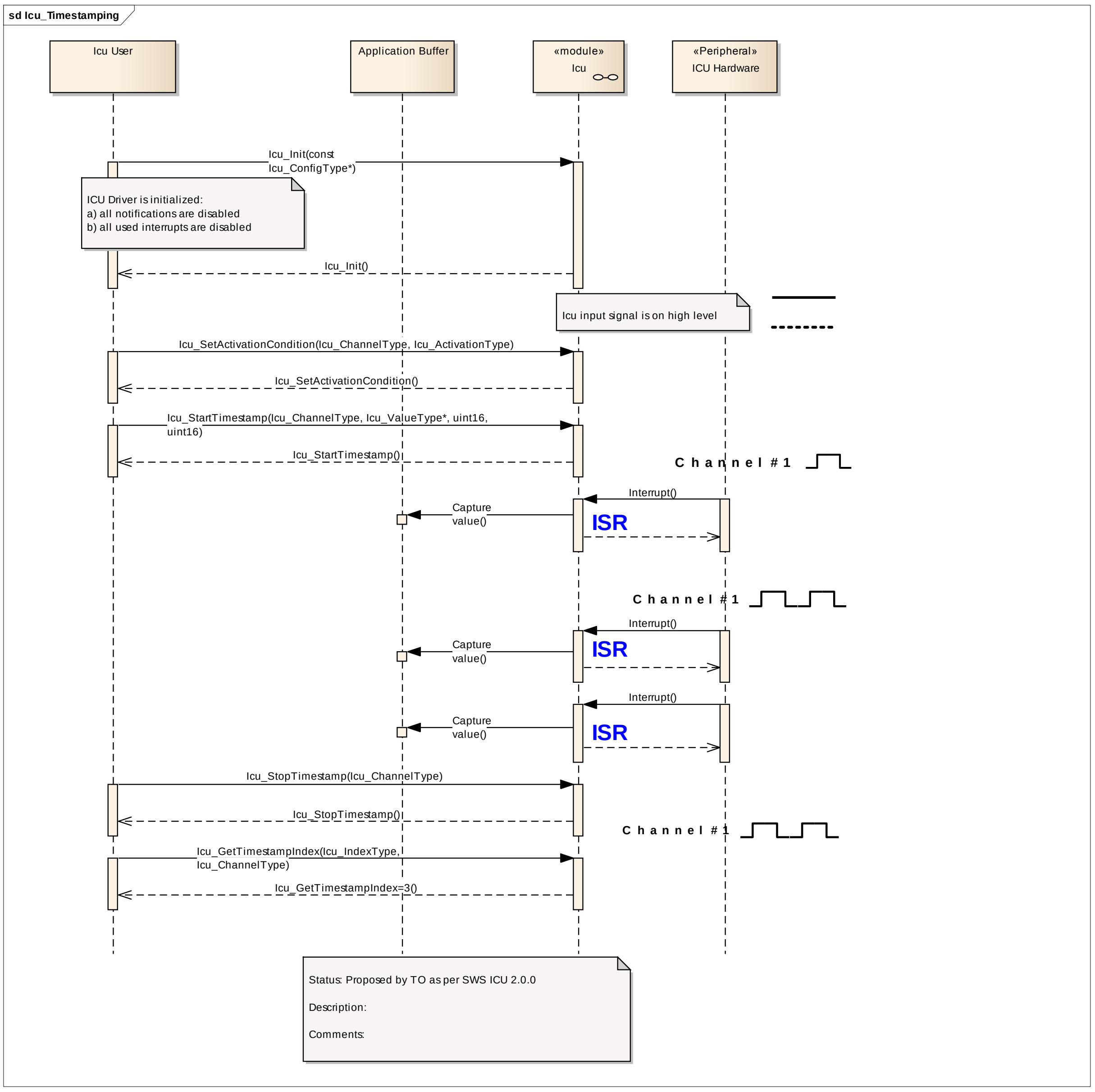
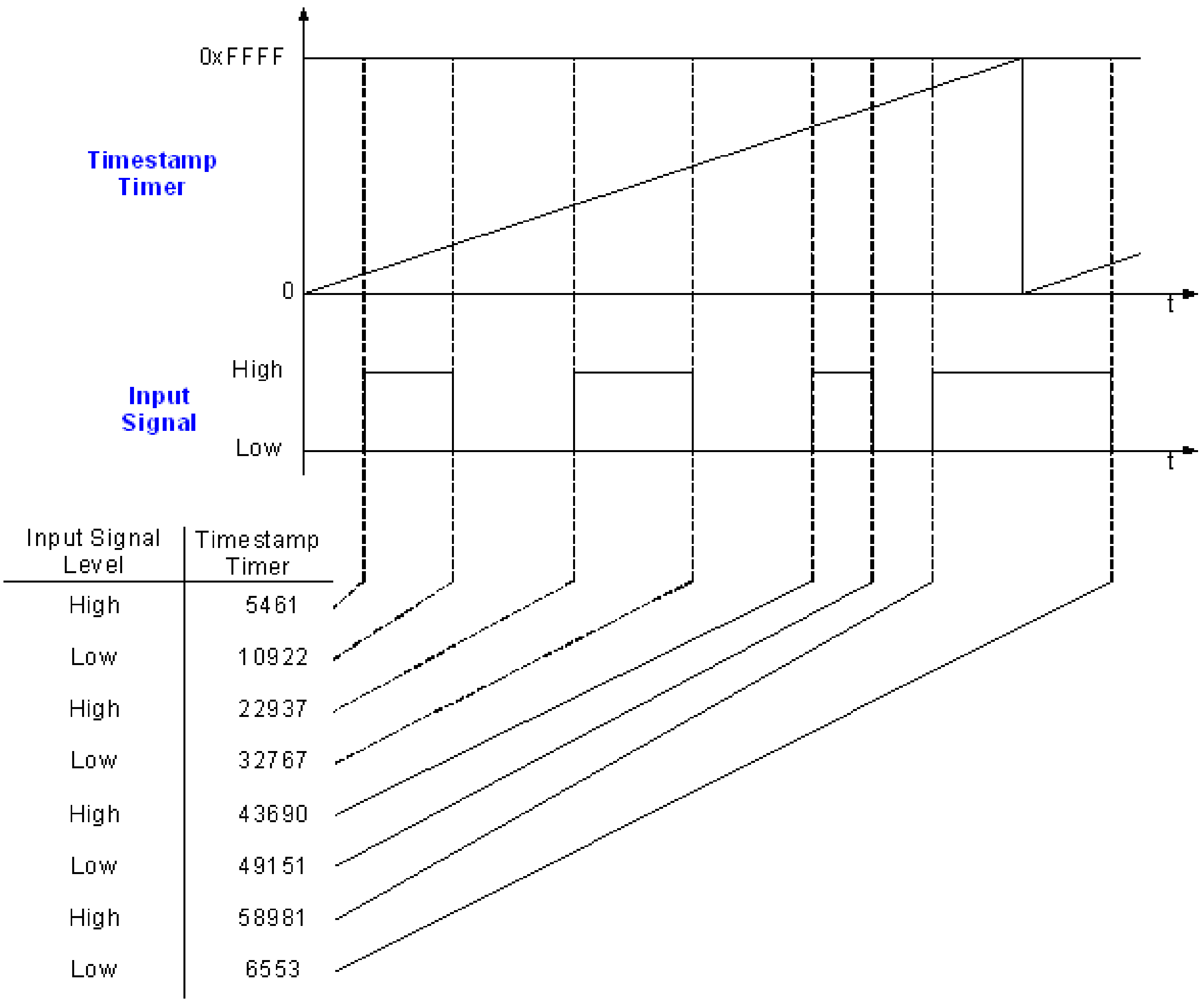

sd Icu_Timestamping
Icu User
Application Buffer
«module» Icu
«Peripheral» ICU Hardware
Icu_Init(const Icu_ConfigType*)
ICU Driver is initialized:
a) all notifications are disabled
b) all used interrupts are disabled
Icu_Init()
Icu input signal is on high level
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Icu_StartTimestamp(Icu_ChannelType, Icu_ValueType*, uint16, uint16)
Icu_StartTimestamp()
Channel #1
Interrupt()
Capture value()
ISR
Channel #1
Interrupt()
Capture value()
ISR
Interrupt()
Capture value()
ISR
Icu_StopTimestamp(Icu_ChannelType)
Icu_StopTimestamp()
Channel #1
Icu_GetTimestampIndex(Icu_IndexType, Icu_ChannelType)
Icu_GetTimestampIndex=3()
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

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

The Timestamping in general is shown in the following figure:



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

Figure 9.15: Timestamping overview

9.12 Icu Edge Counting

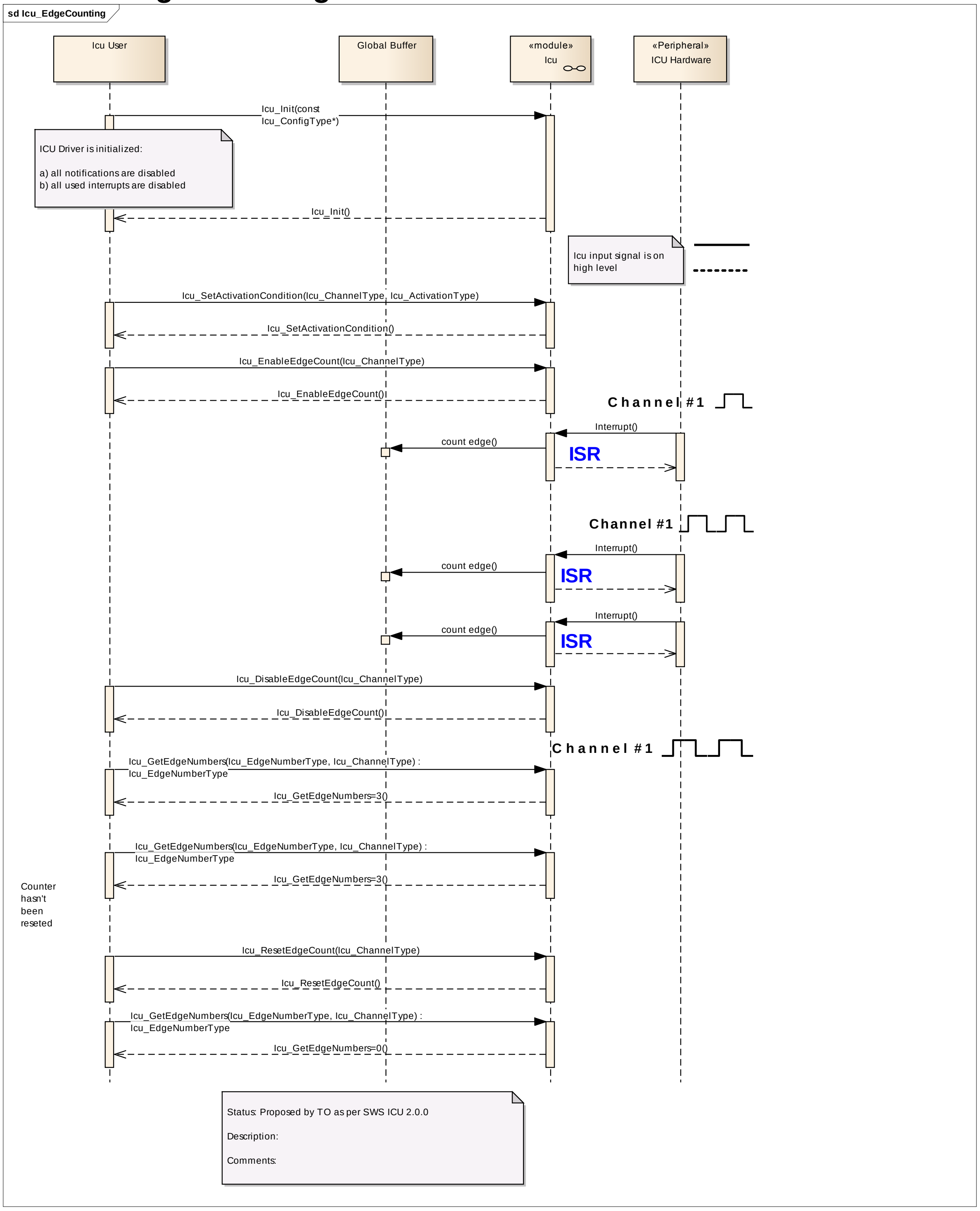

sd Icu_EdgeCounting
Icu User
Global Buffer
«module» Icu
«Peripheral» ICU Hardware
Icu_Init(const Icu_ConfigType*)
ICU Driver is initialized:
a) all notifications are disabled
b) all used interrupts are disabled
Icu_Init()
Icu input signal is on high level
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Icu_EnableEdgeCount(Icu_ChannelType)
Icu_EnableEdgeCount()
Channel #1
Interrupt()
count edge()
ISR
Icu_DisableEdgeCount(Icu_ChannelType)
Icu_DisableEdgeCount()
Icu_GetEdgeNumbers(Icu_EdgeNumberType, Icu_ChannelType) : Icu_EdgeNumberType
Icu_GetEdgeNumbers=3()
Counter hasn't been reseted
Icu_ResetEdgeCount(Icu_ChannelType)
Icu_ResetEdgeCount()
Icu_GetEdgeNumbers=0()
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

Figure 9.16: Inquire the number of counted edges

9.13 Icu_GetTimeElapsed

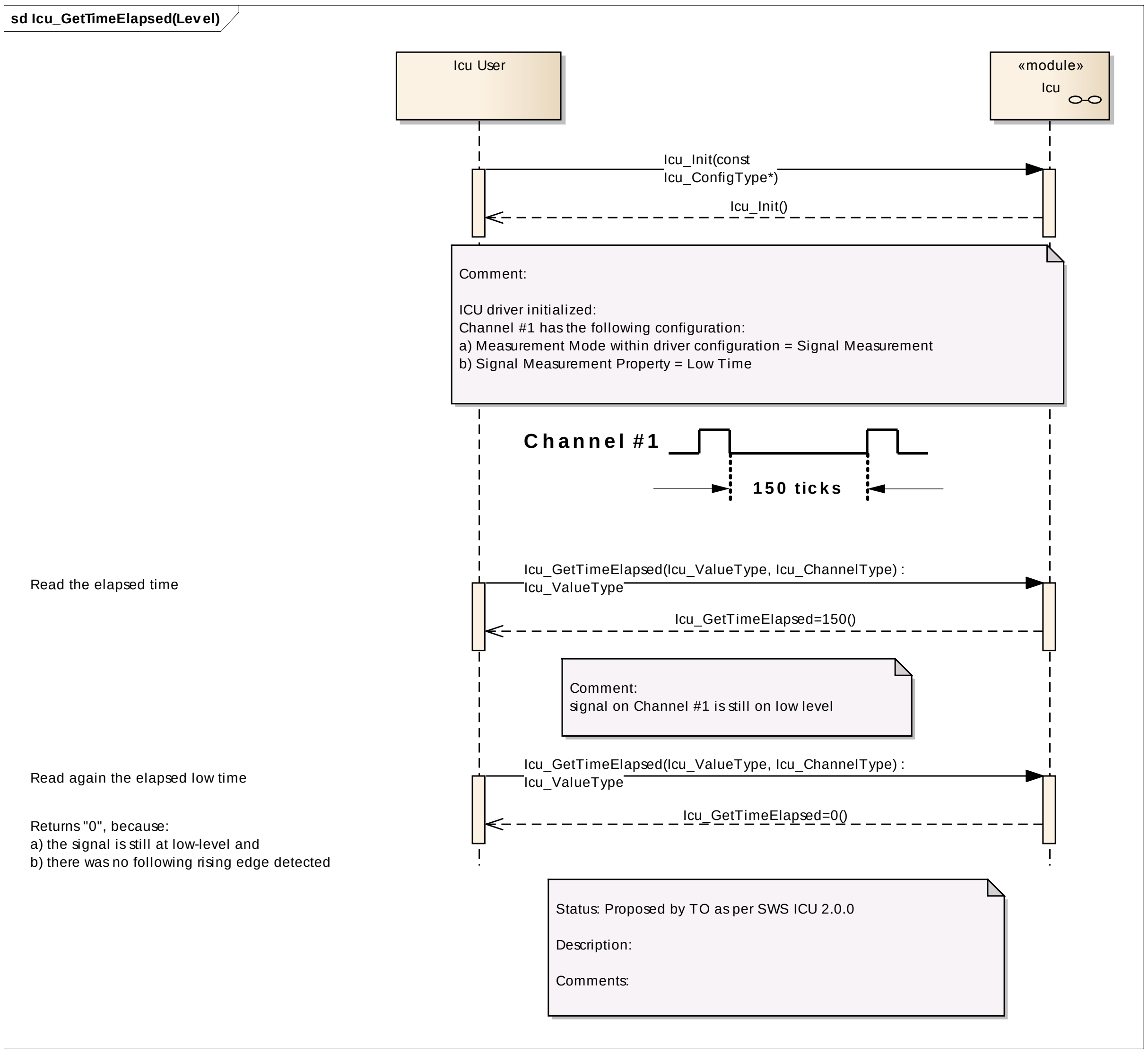

sd Icu_GetTimeElapsed(Level)
Icu User
«module» Icu
Icu_Init(const Icu_ConfigType*)
Icu_Init()
Comment:
ICU driver initialized:
Channel #1 has the following configuration:
a) Measurement Mode within driver configuration = Signal Measurement
b) Signal Measurement Property = Low Time
Channel #1
150 ticks
Read the elapsed time
Icu_GetTimeElapsed(Icu_ValueType, Icu_ChannelType) : Icu_ValueType
Icu_GetTimeElapsed=150()
Comment:
signal on Channel #1 is still on low level
Read again the elapsed low time
Icu_GetTimeElapsed(Icu_ValueType, Icu_ChannelType) : Icu_ValueType
Icu_GetTimeElapsed=0()
Returns "0", because:
a) the signal is still at low-level and
b) there was no following rising edge detected
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

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

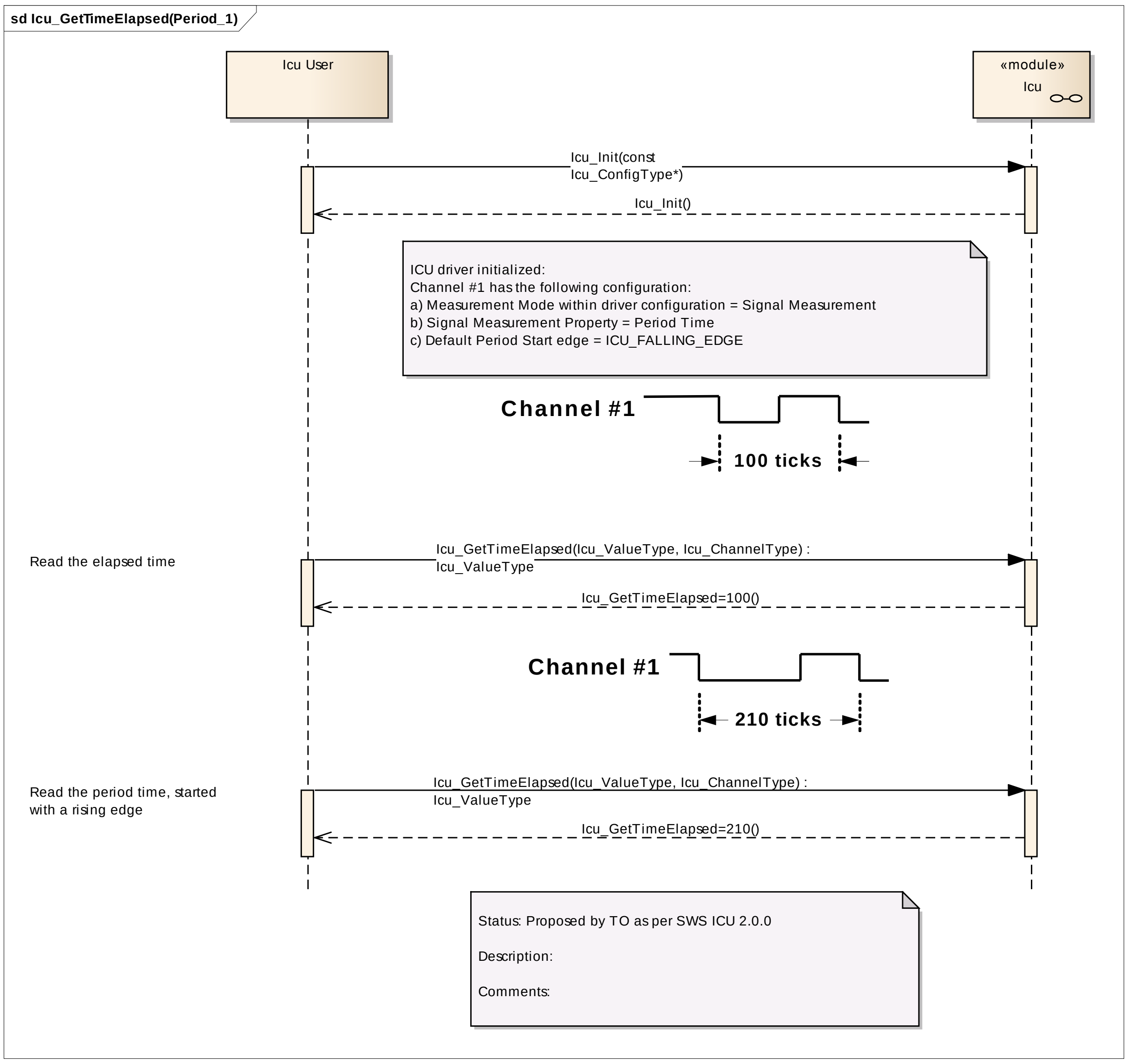

sd Icu_GetTimeElapsed(Period_1)
Icu User
«module»
Icu
Icu_Init(const Icu_ConfigType*)
Icu_Init()
ICU driver initialized:
Channel #1 has the following configuration:
a) Measurement Mode within driver configuration = Signal Measurement
b) Signal Measurement Property = Period Time
c) Default Period Start edge = ICU_FALLING_EDGE
Channel #1
100 ticks
Read the elapsed time
Icu_GetTimeElapsed(Icu_ValueType, Icu_ChannelType) : Icu_ValueType
Icu_GetTimeElapsed=100()
Channel #1
210 ticks
Read the period time, started with a rising edge
Icu_GetTimeElapsed(Icu_ValueType, Icu_ChannelType) : Icu_ValueType
Icu_GetTimeElapsed=210()
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

Figure 9.18: Inquire the elapsed period time of a channel

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

The shown behaviour is also appropriate for the service Icu_GetDutyCycleValues()!

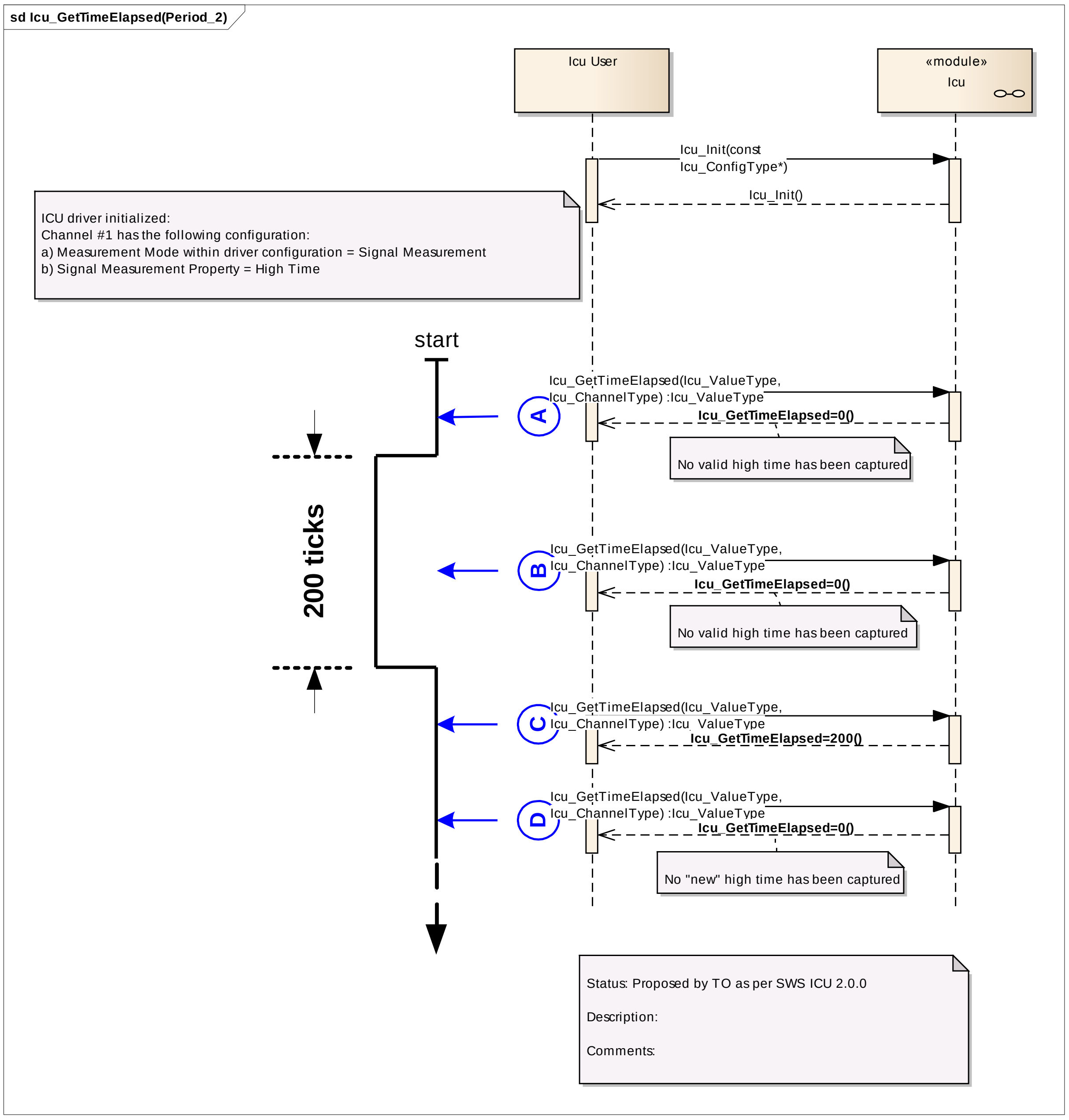

sd Icu_GetTimeElapsed(Period_2)
Icu User
«module» Icu
Icu_Init(const Icu_ConfigType*)
Icu_Init()
ICU driver initialized:
Channel #1 has the following configuration:
a) Measurement Mode within driver configuration = Signal Measurement
b) Signal Measurement Property = High Time
start
200 ticks
A
Icu_GetTimeElapsed(Icu_ValueType, Icu_ChannelType) :Icu_ValueType
Icu_GetTimeElapsed=0()
No valid high time has been captured
B
Icu_GetTimeElapsed(Icu_ValueType, Icu_ChannelType) :Icu_ValueType
Icu_GetTimeElapsed=0()
No valid high time has been captured
C
Icu_GetTimeElapsed(Icu_ValueType, Icu_ChannelType) :Icu_ValueType
Icu_GetTimeElapsed=200()
D
Icu_GetTimeElapsed(Icu_ValueType, Icu_ChannelType) :Icu_ValueType
Icu_GetTimeElapsed=0()
No "new" high time has been captured
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

Figure 9.19: Inquire the elapsed high time of a channel

9.14 Icu_GetDutyCycleValues

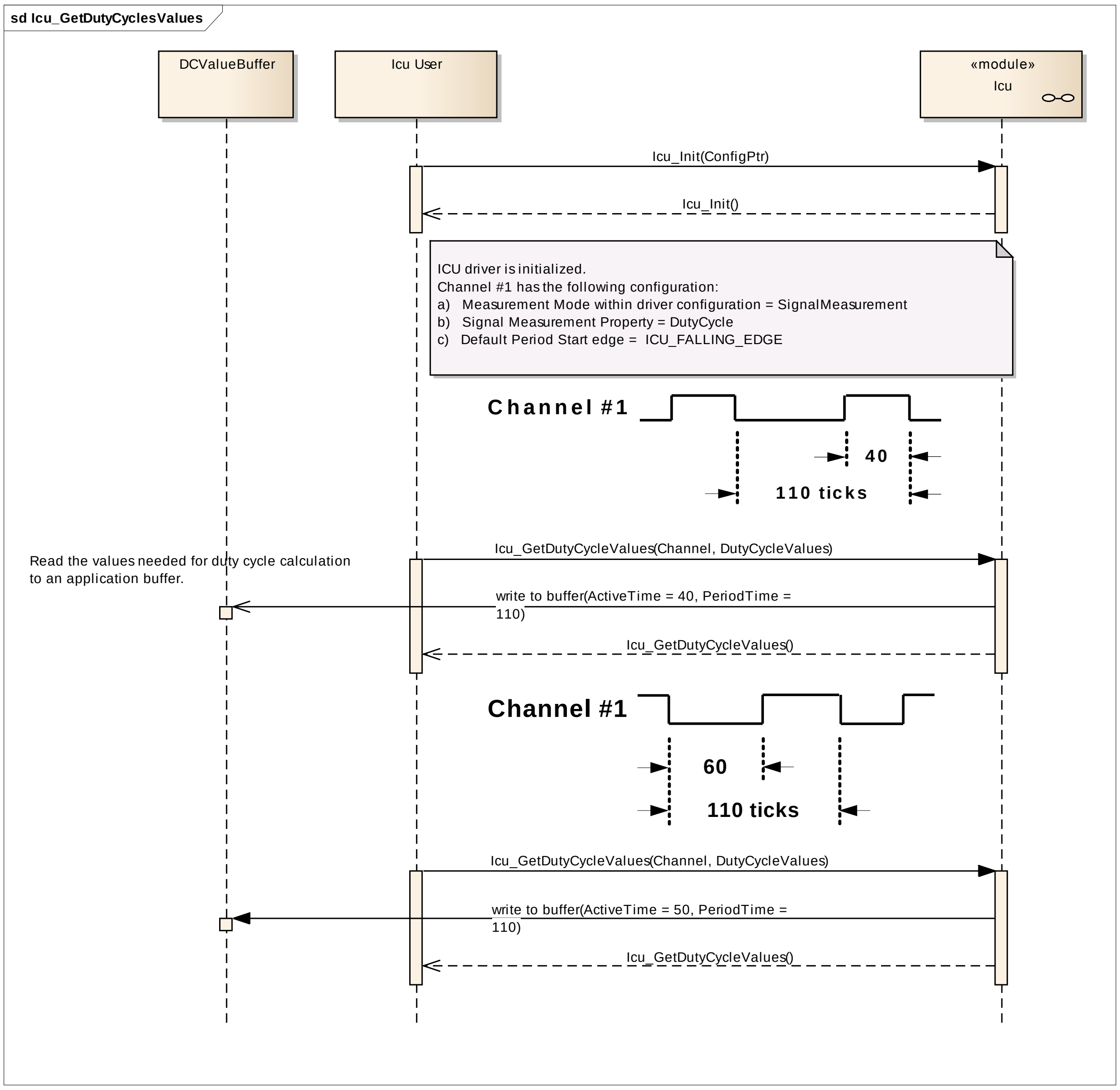

sd Icu_GetDutyCyclesValues
DCValueBuffer
Icu User
«module»
Icu
Icu_Init(ConfigPtr)
Icu_Init()
ICU driver is initialized.
Channel #1 has the following configuration:
a) Measurement Mode within driver configuration = SignalMeasurement
b) Signal Measurement Property = DutyCycle
c) Default Period Start edge = ICU_FALLING_EDGE
Channel #1
40
110 ticks
Icu_GetDutyCycleValues(Channel, DutyCycleValues)
Read the values needed for duty cycle calculation to an application buffer.
write to buffer(ActiveTime = 40, PeriodTime = 110)
Icu_GetDutyCycleValues()
Channel #1
60
110 ticks
Icu_GetDutyCycleValues(Channel, DutyCycleValues)
write to buffer(ActiveTime = 50, PeriodTime = 110)
Icu_GetDutyCycleValues()

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

9.15 Icu_SignalNotification and Icu_GetInputState

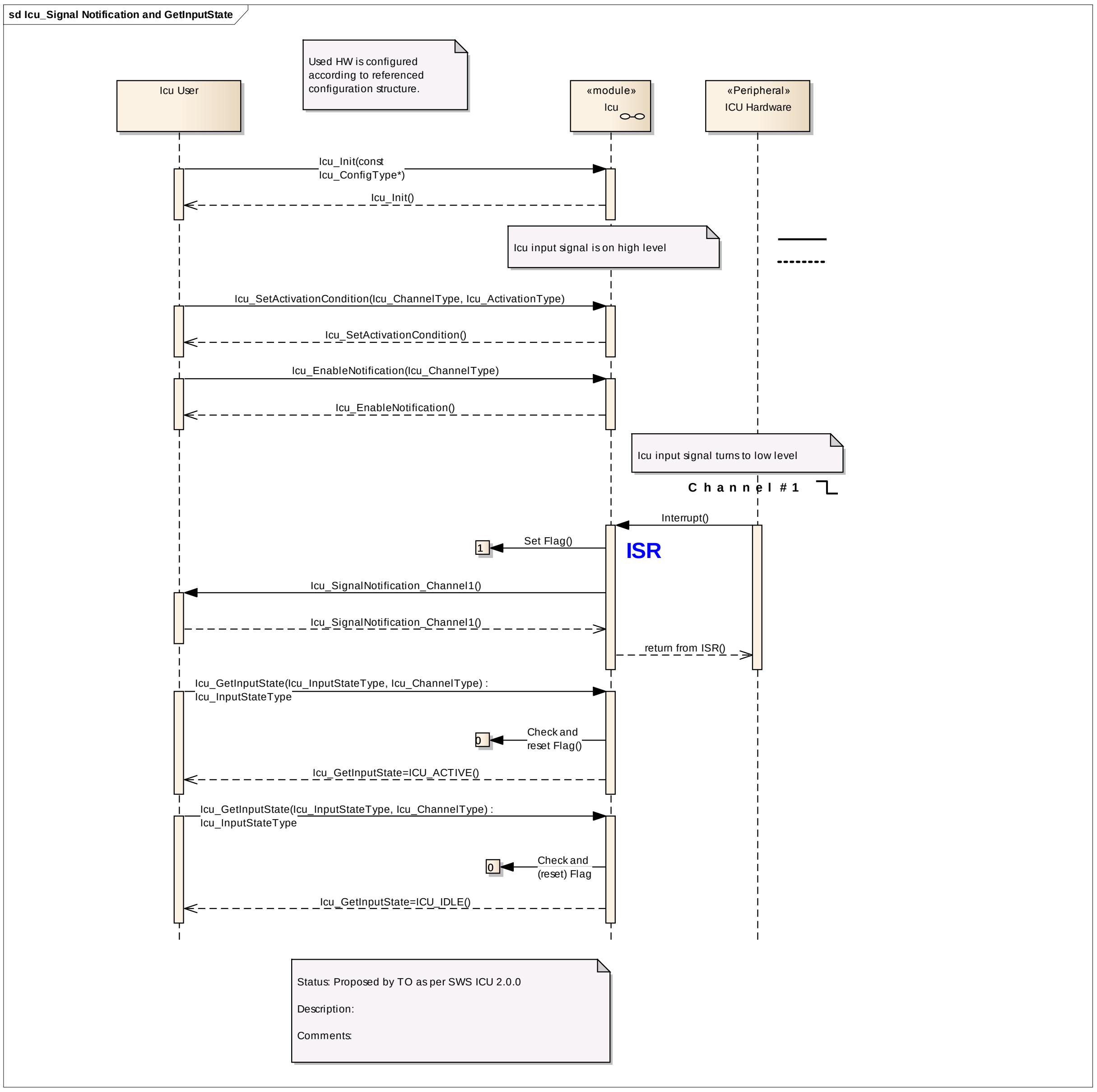

sd Icu_Signal Notification and GetInputState
Used HW is configured according to referenced configuration structure.
Icu User
«module» Icu
«Peripheral» ICU Hardware
Icu_Init(const Icu_ConfigType*)
Icu_Init()
Icu input signal is on high level
Icu_SetActivationCondition(Icu_ChannelType, Icu_ActivationType)
Icu_SetActivationCondition()
Icu_EnableNotification(Icu_ChannelType)
Icu_EnableNotification()
Icu input signal turns to low level
Channel #1
Interrupt()
Set Flag()
1
ISR
Icu_SignalNotification_Channel1()
Icu_SignalNotification_Channel1()
return from ISR()
Icu_GetInputState(Icu_InputStateType, Icu_ChannelType) : Icu_InputStateType
Check and reset Flag()
0
Icu_GetInputState=ICU_ACTIVE()
Icu_GetInputState(Icu_InputStateType, Icu_ChannelType) : Icu_InputStateType
Check and (reset) Flag
0
Icu_GetInputState=ICU_IDLE()
Status: Proposed by TO as per SWS ICU 2.0.0
Description:
Comments:

Figure 9.21: Cooperative usage of notification and polling mechanism

10 Configuration specification

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

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

Chapter 10.3 specifies published information of the module ICU.

10.1 How to read this chapter

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

ICU188: Variant PC (**P**re **C**ompile): Only pre-compile time parameters

ICU189: 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.
ICU148: If not applicable, a NULL pointer shall be passed to the initialization routine. In this case the check for this NULL pointer (**ICU023**) has to be omitted

10.2.2 Icu

Module Name	*Icu*
Module Description	Configuration of the Icu (Input Capture Unit) module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
IcuConfigSet	1	This container is the base for a multiple configuration set
IcuGeneral	1	Configuration of general ICU parameters.
IcuOptionalApis	1	This container contains all configuration switches for configuring optional API services of the ICU driver.

10.2.3 IcuGeneral

SWS Item	**ICU026 :**
Container Name	IcuGeneral{General Configuration}
Description	Configuration of general ICU parameters.
Configuration Parameters	

SWS Item	**ICU232 :**		
Name	IcuDevErrorDetect {ICU_DEV_ERROR_DETECT}		
Description	Switches the Development Error Detection and Notification on or off. true: Enabled. false: Disabled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ICU221 :**

Name	IcuIndex		
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.		
Multiplicity	1		
Type	IntegerParamDef		
Range	..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**ICU233 :**		
Name	IcuReportWakeupSource {ICU_REPORT_WAKEUP_SOURCE}		
Description	Switch for enabling Wakeup source reporting. true: Report Wakeup source. false: Do not report Wakeup source.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.2.4 IcuOptionalApis

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

SWS Item	**ICU234 :**		
Name	IcuDeInitApi {ICU_DE_INIT_API}		
Description	Adds / removes the service Icu_DeInit() from the code. true: Icu_DeInit() can be used. false: Icu_DeInit() can not be used.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ICU235 :**
Name	IcuDisableWakeupApi {ICU_DISABLE_WAKEUP_API}
Description	Adds / removes the service Icu_DisableWakeup() from the code. true: Icu_DisableWakeup() can be used. false: Icu_DisableWakeup() can not be used.
Multiplicity	1
Type	BooleanParamDef
Default value	--

ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ICU124 :**		
Name	IcuEdgeCountApi {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(). true: The services listed above can be used. false: The services listed above can not be used.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ICU236 :**		
Name	IcuEnableWakeupApi {ICU_ENABLE_WAKEUP_API}		
Description	Adds / removes the service Icu_EnableWakeup() from the code. true: Icu_EnableWakeup() can be used. false: Icu_EnableWakeup() can not be used.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ICU237 :**		
Name	IcuGetDutyCycleValuesApi {ICU_GET_DUTY_CYCLE_VALUES_API}		
Description	Adds / removes the service Icu_GetDutyCycleValues() from the code. true: Icu_GetDutyCycleValues() can be used. false: Icu_GetDutyCycleValues() can not be used.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	**Pre-compile time**	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: If IcuSignalMeasurementApi==false this switch shall also be set to false.		

SWS Item	**ICU238 :**
Name	IcuGetInputStateApi {ICU_GET_INPUT_STATE_API}
Description	Adds / removes the service Icu_GetInputState() from the code. true:

	Icu_GetInputState() can be used. false: Icu_GetInputState() can not be used.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ICU239 :**		
Name	IcuGetTimeElapsedApi {ICU_GET_TIME_ELAPSED_API}		
Description	Adds / removes the service Icu_GetTimeElapsed() from the code. true: Icu_GetTimeElapsed() can be used. false: Icu_GetTimeElapsed() can not be used.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: If IcuSignalMeasurementApi==false this switch shall also be set to false.		

SWS Item	**ICU240 :**		
Name	IcuGetVersionInfoApi {ICU_GET_VERSION_INFO_API}		
Description	Adds / removes the service Icu_GetVersionInfo() from the code. true: Icu_GetVersionInfo() can be used. false: Icu_GetVersionInfo() can not be used.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ICU241 :**		
Name	IcuSetModeApi {ICU_SET_MODE_API}		
Description	Adds / removes the service Icu_SetMode() from the code. true: Icu_SetMode() can be used. false: Icu_SetMode() can not be used.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ICU242 :**		
Name	IcuSignalMeasurementApi {ICU_SIGNAL_MEASUREMENT_API}		
Description	Adds / removes the services Icu_StartSignalMeasurement() and Icu_StopSignalMeasurement() from the code. true: Icu_StartSignalMeasurement() and Icu_StopSignalMeasurement() can be used. false: Icu_StartSignalMeasurement() and Icu_StopSignalMeasurement() can not be used.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ICU123 :**		
Name	IcuTimestampApi {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(). true: The services listed above can be used. false: The services listed above can not be used.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.2.5 IcuChannel

SWS Item	**ICU027 :**
Container Name	IcuChannel
Description	Configuration of an individual ICU channel.
Configuration Parameters	

SWS Item	**ECUC_Icu_00354 :**		
Name	IcuChannelId		
Description	Channel Id of the ICU channel. This value will be assigned to the symbolic name derived of the IcuChannel container short name.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 ..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ICU222 :**
Name	IcuDefaultStartEdge {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 "IcuSignalMeasurement" and the properties "DutyCycle" or "Period" are set, the edge configured here is used as Default Period Start Edge. Implementation Type: Icu_ActivationType		
Multiplicity	1		
Type	EnumerationParamDef		
Range	ICU_BOTH_EDGES	As default, both edges are used.	
	ICU_FALLING_EDGE	As default, falling edge is the used.	
	ICU_RISING_EDGE	As default, rising edge is the used.	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**ICU223 :**	
Name	IcuMeasurementMode {Icu_MeasurementMode}	
Description	Configures the measurement mode of this channel. Implementation Type: Icu_MeasurementModeType	
Multiplicity	1	
Type	EnumerationParamDef	
Range	ICU_MODE_EDGE_COUNTER	The channnel 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 IcuEdgeVountApi is switched on.
	ICU_MODE_SIGNAL_EDGE_DETECT	The channel is used for detecting the edges which are configured by the call of the service Icu_SetActivationCondition(). The following API services support this mode: - Icu_EnableNotification() - Icu_DisableNotification() - Icu_GetInputState()
	ICU_MODE_SIGNAL_MEASUREMENT	The channel is used to measure different times between various configurable edges. The configuration of the period-start edges are done by configuration and cannot be changed during runtime. The following API services support this mode: - Icu_GetTimeElapsed() - Icu_GetDutyCycleValues() - Icu_GetInputState() This mode can only be configured if at least one of the following switches are set to "true": - IcuGetDutyCycleValuesApi - IcuGetTimeElapsedApi
	ICU_MODE_TIMESTAMP	The channel is used to capture timer values on the edges which are configured by the call of the service

		Icu_SetActivationCondition(). The following API services support this mode: - Icu_StartTimestamp() - Icu_StopTimestamp() - Icu_GetTimestampIndex() This mode can only be configured if IcuTimeStampApi is switched on.	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: The possible measurement modes are depending on the pre-processor switches, which enable/disable optional API services.		

SWS Item	**ICU224 :**		
Name	IcuWakeupCapability {Icu_WakeupCapability}		
Description	Information about the wakeup-capability of this channel. true: Channel is wakeup capable. false: Channel is not wakeup capable.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

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

10.2.6 IcuSignalEdgeDetection

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

SWS Item	**ICU225 :**
Name	IcuSignalNotification {Icu_SignalNotification_<Channel>}
Description	Notification function for signal notification.
Multiplicity	1
Type	FunctionNameDef
Default value	--

regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: IcuMeasurementMode		

No Included Containers

10.2.7 IcuSignalMeasurement

SWS Item	**ICU226 :**
Container Name	IcuSignalMeasurement{Configuration of Signal Measurement}
Description	This container contains the configuration (parameters) in case the measurement mode is "IcuSignalMeasurement"
Configuration Parameters	

SWS Item	**ICU227 :**		
Name	IcuSignalMeasurementProperty {Icu_SignalMeasurementProperty}		
Description	Configures the property that could be measured in case the mode is "IcuSignalMeasurement". This property can not be changed during runtime. Implementation Type: Icu_SignalMeasurementPropertyType		
Multiplicity	1		
Type	EnumerationParamDef		
Range	ICU_DUTY_CYCLE	The channel is configured to read values which are needed for calculating the duty cycle (coherent Active and Period Time).	
	ICU_HIGH_TIME	The channel is configured for reading the elapsed Signal High Time	
	ICU_LOW_TIME	The channel is configured for reading the elapsed Signal Low Time	
	ICU_PERIOD_TIME	The channel is configured for reading the elapsed Signal Period Time	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: IcuMeasurementMode, IcuGetDutyCycleValuesApi, IcuGetTimeElapsedApi		

No Included Containers

10.2.8 IcuTimestampMeasurement

SWS Item	**ICU228 :**
Container Name	IcuTimestampMeasurement{Configuration of Timestamp Measurement}
Description	This container contains the configuration (parameters) in case the measurement mode is "IcuTimestamp"
Configuration Parameters	

SWS Item	**ICU229 :**
Name	IcuTimestampMeasurementProperty {Icu_TimestampMeasurementProperty}
Description	Configures the handling of the buffer in case the mode is "Timestamp" Implementation Type: Icu_TimestampBufferType
Multiplicity	1
Type	EnumerationParamDef

Range	ICU_CIRCULAR_BUFFER	After reaching the end of the buffer, the driver restarts at the beginning of the buffer	
	ICU_LINEAR_BUFFER	The buffer will just be filled once	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: IcuMeasurementMode		

SWS Item	**ICU230 :**		
Name	IcuTimestampNotification {Icu_TimestampNotification_<Channel>}		
Description	Notification function if the number of requested timestamps (Notification interval > 0) are acquired.		
Multiplicity	1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: IcuTimestampApi		

No Included Containers

10.2.9 IcuEdgeCounterMeasurement

SWS Item	**ICU076 :**
Container Name	IcuEdgeCounterMeasurement{Configuration of Edge Counter Measurement}
Description	This container contains the configuration (parameters) in case the measurement mode is "IcuEdgeCounter"
Configuration Parameters	

No Included Containers

10.2.10 IcuWakeup

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

SWS Item	**ICU231 :**		
Name	IcuChannelWakeupInfo {Icu_ChannelWakeupInfo}		
Description	If the wakeup-capability is true the wakeup source referenced is transmitted to the ECU State Manager (EcuM) . Implementation Type: reference to EcuM_WakeupSourceType		
Multiplicity	1		
Type	Reference to [EcuMWakeupSource]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build	M	VARIANT-POST-BUILD

	time		
Scope / Dependency	scope: ECU dependency: IcuWakeupCapability and IcuReportWakeupSource		

No Included Containers

10.2.11 IcuConfigSet

SWS Item	**ICU219 :**
Container Name	IcuConfigSet [Multi Config Container]
Description	This container is the base for a multiple configuration set
Configuration Parameters	

SWS Item	**ICU220 :**		
Name	IcuMaxChannel {ICU_MAX_CHANNEL}		
Description	This parameter contains the number of Channels configured. It will be gathered by tools during the configuration stage. calculationFormula = Number of configured Icu Channels Implementation Type: Icu_ChannelType		
Multiplicity	1		
Type	DerivedIntegerParamDef		
Range	0 ..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

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

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.

The standard common published information like

vendorId (<Module>_VENDOR_ID),
moduleId (<Module>_MODULE_ID),
arMajorVersion (<Module>_AR_MAJOR_VERSION),
arMinorVersion (<Module>_ AR_MINOR_VERSION),
arPatchVersion (<Module>_ AR_PATCH_VERSION),
swMajorVersion (<Module>_SW_MAJOR_VERSION),
swMinorVersion (<Module>_ SW_MINOR_VERSION),
swPatchVersion (<Module>_ SW_PATCH_VERSION),
vendorApiInfix (<Module>_VENDOR_API_INFIX)

is provided in the BSW Module Description Template (see [11] Figure 4.1 and Figure 7.1).

Additional published parameters are listed below if applicable for this module.

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