Document Title	Specification of ICU Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	023
Document Classification	Standard
Document Version	4.4.0
Document Status	Final
Part of Release	4.1
Revision	2

Document Change History

Date	Version	Changed by	Change Description
31.10.2013	4.4.0	AUTOSAR Release Management	• ICU00354 - Check for a valid notification interval rephrased • ICU078 - Removed the sentence "This is done by the hardware." from the note. • ICU295 - Removed ICU_ACTIVE_TIME from the range of enumeration Icu_SignalMeasurementPropertyType • Editorial changes • Removed chapter(s) on change documentation
14.02.2013	4.3.0	AUTOSAR Administration	• Modified the scope of the parameters from ECU/Module to local • Reworked according to the new SWS_BSWGeneral • Changed MemMap.h to Icu_MemMap.h
02.11.2011	4.2.0	AUTOSAR Administration	• Corrected Type errors • Updated description of Icu_IndexType
21.10.2010	4.1.0	AUTOSAR Administration	• Services 'Icu_DisableEdgeDetection' and 'Icu_EnableEdgeDetection' were added. • Configuration parameters 'IcuEdgeDetectApi'and 'IcuWakeupFunctionalityApi' has been added. • Definition of 'duty cycle' has been corrected. • Corrected values of the parameter 'Icu_SignalMeasurementPropertyType'
23.06.2008	3.0.1	AUTOSAR Administration	• Legal disclaimer revised

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
07.12.2007	3.0.0	AUTOSAR Administration	• The code file structure of the module was completely reworked. • The following requirements were added: SWS_Icu_00088, SWS_Icu_00220, SWS_Icu_00221, SWS_Icu_00228 and SWS_Icu_00229. • The flow charts related to the ECU Wake-Up moved to the • SWS document of the ECU State Manager. • Document meta information extended • Small layout adaptations made
31.01.2007	2.1.0	AUTOSAR Administration	▪ Default start edge is now used for edge configuration ▪ Enable and Disable Notification can now be used for Timestamp functionality. ▪ Edge detection functionality is now pre compile time configurable On/Off ▪ Legal disclaimer revised ▪ Release Notes added ▪ “Advice for users” revised ▪ “Revision Information” added
30.01.2006	2.0.0	AUTOSAR Administration	Added the following services - Icu_SetActivationCondition - Icu_StartTimeStamp - Icu_StopTimeStamp - Icu_GetTimestampIndex - Icu_ResetEdgeCount - Icu_EnableEdgeCount - Icu_DisableEdgeCount - Icu_GetEdgeNumbers - Icu_GetTimeElapsed - Icu_GetDutyCycleValues - Icu_GetVersionInfo
30.06.2005	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and functional overview

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

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

The ICU driver provides services for

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

2 Acronyms and abbreviations

Abbreviation / Acronym:	Description:
Active Time	This depends on the starting edge of the signal to be captured. ▪ Start edge = falling edge => Active Time = Low Time ▪ Start edge = rising edge => Active Time = High Time ▪ Start edge = both edges => Active Time = High Time (if rising edge occurs initially) ▪ Start edge = both edges => Active Time = Low Time (if falling edge occurs initially)
DEM	Diagnostic Event Manager
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_BSWGeneral.pdf

[2] General Requirements on SPAL,
AUTOSAR_SRS_SPALGeneral.pdf

[3] Specification of Standard Types,
AUTOSAR_SWS_StandardTypes.pdf

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

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

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

[7] Requirements on ICU Driver,
AUTOSAR_SRS_ICUDriver.pdf

[8] Specification of ECU Configuration,
AUTOSAR_TPS_ECUConfiguration.pdf

[9] Layered Software Architecture,
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

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

[11] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[12] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

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

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [12] (SWS BSW General), which is also valid for ICU Driver.

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

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

5.1 Module DET (Development Error Tracer)

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

5.2 Module MCU

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

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

5.3 OS (Operating System)

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

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

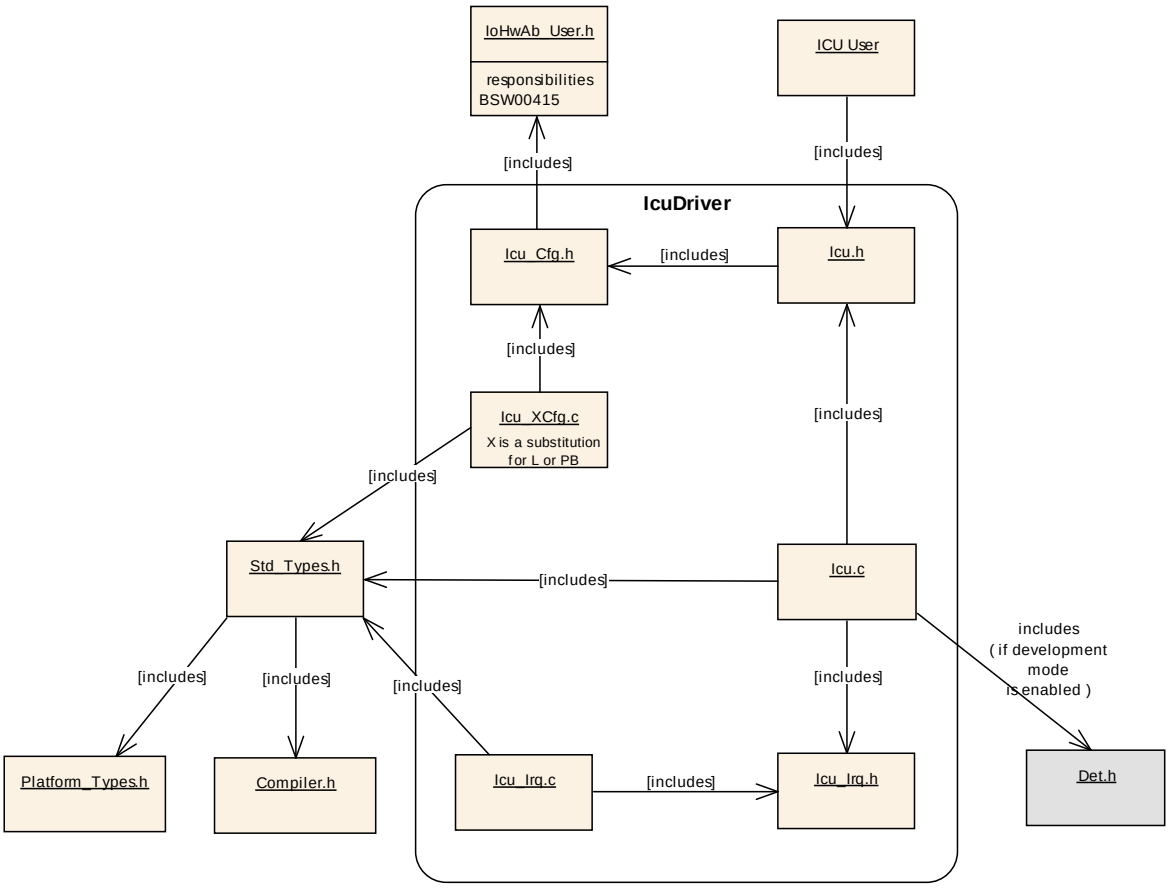
5.4 Module PORT

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

5.5 Module EcuM

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

5.6 File structure



5.6.1 Header file structure

The code file structure shall be as follows:

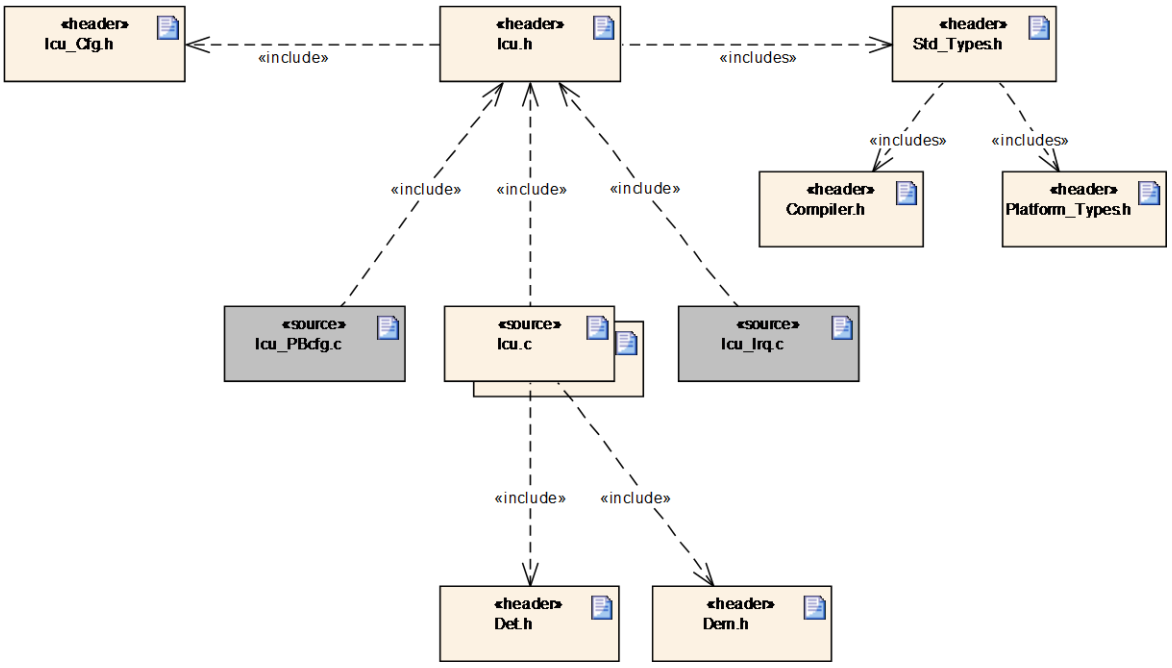


Figure 5.1: Header file structure

[SWS_Icu_00245] ⌈`Icu.h` shall include `Icu_Cfg.h` for the API pre-compiler switches.

Icu.c has access to the `Icu_Cfg.h` via the implicitly included `Icu.h` file.⌋ ()

[SWS_Icu_00246] ⌈`Icu_Irq.c` shall include `Icu.h` for the function which shall be called in the interrupt function.and `Icu_Irq.h` for the declaration of interrupt functions.⌋ ()

[SWS_Icu_00247] ⌈The Type definitions for `Icu_Lcfg.c` and `Icu_PBcfg.c` are located in the file `Icu_Cfg.h` or `Icu.h`.

The implicit include of `Icu_Cfg.h` via `Icu.h` in the files `Icu_Lcfg.c` and `Icu_PBcfg.c` is necessary and can be solved like in the following construct:

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

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

Icu_Cfg.h
---------------
#include "Icu.h"
#define ICU_VERSION_INFO_API⌋ ()
```

[SWS_Icu_00248] ⌈`Icu_Lcfg.c` shall include `Icu_Cbk.h` for a link time configuration if the call back function is linked to the module via the ROM structure.⌋ ()

[SWS_Icu_00249] ⌈`Icu_PBcfg.c` shall include `Icu_Cbk.h` for post build time configuration if the call back function is linked to the module via the ROM structure.⌋ (SRS_BSW_00435)

[SWS_Icu_00250] ⌈`Icu.c` shall include `Icu_Cbk.h` for pre-compile time configuration⌋ ()

[SWS_Icu_00251] ⌈`Icu.c` shall include `Det.h, SchM_Icu.h` and `Icu_MemMap.h.`⌋ (SRS_BSW_00436)

[SWS_Icu_00252] ⌈`Icu_Irq.c` shall include `Icu_MemMap.h.`⌋ ()

[SWS_Icu_00253] ⌈`Icu_Lcfg.c` shall include `[ICU.h` and `Icu_MemMap.h.`⌋ ()

[SWS_Icu_00254] ⌈`Icu_PBcfg.c` shall include `Icu_MemMap.h` and `Icu.h.`⌋ ()

[SWS_Icu_00256] ⌈`Icu.h` shall include `EcuM_Cbk.h.`⌋ ()

[SWS_Icu_00116] ⌈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

Requirement	Description	Satisfied by
-	-	SWS_Icu_00050
-	-	SWS_Icu_00059
-	-	SWS_Icu_00064
-	-	SWS_Icu_00091
-	-	SWS_Icu_00116
-	-	SWS_Icu_00121
-	-	SWS_Icu_00134
-	-	SWS_Icu_00135
-	-	SWS_Icu_00136
-	-	SWS_Icu_00137
-	-	SWS_Icu_00138
-	-	SWS_Icu_00139
-	-	SWS_Icu_00140
-	-	SWS_Icu_00141
-	-	SWS_Icu_00142
-	-	SWS_Icu_00143
-	-	SWS_Icu_00144
-	-	SWS_Icu_00145
-	-	SWS_Icu_00146
-	-	SWS_Icu_00148
-	-	SWS_Icu_00149
-	-	SWS_Icu_00150
-	-	SWS_Icu_00152
-	-	SWS_Icu_00155
-	-	SWS_Icu_00156
-	-	SWS_Icu_00159
-	-	SWS_Icu_00160
-	-	SWS_Icu_00161
-	-	SWS_Icu_00162
-	-	SWS_Icu_00163
-	-	SWS_Icu_00164
-	-	SWS_Icu_00165
-	-	SWS_Icu_00166
-	-	SWS_Icu_00169
-	-	SWS_Icu_00170
-	-	SWS_Icu_00171

-	-	SWS_Icu_00172
-	-	SWS_Icu_00173
-	-	SWS_Icu_00174
-	-	SWS_Icu_00175
-	-	SWS_Icu_00176
-	-	SWS_Icu_00177
-	-	SWS_Icu_00178
-	-	SWS_Icu_00179
-	-	SWS_Icu_00180
-	-	SWS_Icu_00181
-	-	SWS_Icu_00188
-	-	SWS_Icu_00189
-	-	SWS_Icu_00190
-	-	SWS_Icu_00191
-	-	SWS_Icu_00193
-	-	SWS_Icu_00194
-	-	SWS_Icu_00195
-	-	SWS_Icu_00196
-	-	SWS_Icu_00197
-	-	SWS_Icu_00198
-	-	SWS_Icu_00199
-	-	SWS_Icu_00200
-	-	SWS_Icu_00201
-	-	SWS_Icu_00202
-	-	SWS_Icu_00203
-	-	SWS_Icu_00204
-	-	SWS_Icu_00205
-	-	SWS_Icu_00206
-	-	SWS_Icu_00207
-	-	SWS_Icu_00208
-	-	SWS_Icu_00209
-	-	SWS_Icu_00210
-	-	SWS_Icu_00211
-	-	SWS_Icu_00212
-	-	SWS_Icu_00213
-	-	SWS_Icu_00214
-	-	SWS_Icu_00216
-	-	SWS_Icu_00217
-	-	SWS_Icu_00218

-	-	SWS_Icu_00220
-	-	SWS_Icu_00221
-	-	SWS_Icu_00228
-	-	SWS_Icu_00229
-	-	SWS_Icu_00244
-	-	SWS_Icu_00245
-	-	SWS_Icu_00246
-	-	SWS_Icu_00247
-	-	SWS_Icu_00248
-	-	SWS_Icu_00250
-	-	SWS_Icu_00252
-	-	SWS_Icu_00253
-	-	SWS_Icu_00254
-	-	SWS_Icu_00256
-	-	SWS_Icu_00258
-	-	SWS_Icu_00259
-	-	SWS_Icu_00260
-	-	SWS_Icu_00261
-	-	SWS_Icu_00276
-	-	SWS_Icu_00277
-	-	SWS_Icu_00278
-	-	SWS_Icu_00279
-	-	SWS_Icu_00280
-	-	SWS_Icu_00281
-	-	SWS_Icu_00283
-	-	SWS_Icu_00284
-	-	SWS_Icu_00285
-	-	SWS_Icu_00286
-	-	SWS_Icu_00287
-	-	SWS_Icu_00288
-	-	SWS_Icu_00289
-	-	SWS_Icu_00290
-	-	SWS_Icu_00291
-	-	SWS_Icu_00292
-	-	SWS_Icu_00293
-	-	SWS_Icu_00294
-	-	SWS_Icu_00295
-	-	SWS_Icu_00296
-	-	SWS_Icu_00297

-	-	SWS_Icu_00298
-	-	SWS_Icu_00299
-	-	SWS_Icu_00300
-	-	SWS_Icu_00301
-	-	SWS_Icu_00302
-	-	SWS_Icu_00303
-	-	SWS_Icu_00304
-	-	SWS_Icu_00305
-	-	SWS_Icu_00306
-	-	SWS_Icu_00307
-	-	SWS_Icu_00308
-	-	SWS_Icu_00309
-	-	SWS_Icu_00310
-	-	SWS_Icu_00311
-	-	SWS_Icu_00312
-	-	SWS_Icu_00313
-	-	SWS_Icu_00314
-	-	SWS_Icu_00315
-	-	SWS_Icu_00316
-	-	SWS_Icu_00317
-	-	SWS_Icu_00318
-	-	SWS_Icu_00319
-	-	SWS_Icu_00320
-	-	SWS_Icu_00321
-	-	SWS_Icu_00322
-	-	SWS_Icu_00323
-	-	SWS_Icu_00324
-	-	SWS_Icu_00325
-	-	SWS_Icu_00326
-	-	SWS_Icu_00327
-	-	SWS_Icu_00328
-	-	SWS_Icu_00329
-	-	SWS_Icu_00330
-	-	SWS_Icu_00331
-	-	SWS_Icu_00332
-	-	SWS_Icu_00333
-	-	SWS_Icu_00334
-	-	SWS_Icu_00335
-	-	SWS_Icu_00336

-	-	SWS_Icu_00337
-	-	SWS_Icu_00338
-	-	SWS_Icu_00339
-	-	SWS_Icu_00340
-	-	SWS_Icu_00341
-	-	SWS_Icu_00342
-	-	SWS_Icu_00343
-	-	SWS_Icu_00344
-	-	SWS_Icu_00345
-	-	SWS_Icu_00346
-	-	SWS_Icu_00348
-	-	SWS_Icu_00349
-	-	SWS_Icu_00354
-	-	SWS_Icu_00356
-	-	SWS_Icu_00358
-	-	SWS_Icu_00359
-	-	SWS_Icu_00360
-	-	SWS_Icu_00361
-	-	SWS_Icu_00362
-	-	SWS_Icu_00363
-	-	SWS_Icu_00364
-	-	SWS_Icu_00365
-	-	SWS_Icu_00366
-	-	SWS_Icu_00367
-	-	SWS_Icu_00368
-	-	SWS_Icu_00369
-	-	SWS_Icu_00370
-	-	SWS_Icu_00371
-	-	SWS_Icu_00372
-	-	SWS_Icu_00373
-	-	SWS_Icu_00374
-	-	SWS_Icu_00375
-	-	SWS_Icu_00376
-	-	SWS_Icu_00377
-	-	SWS_Icu_00378
BSW00324	-	SWS_Icu_00380
BSW00420	-	SWS_Icu_00380
BSW00421	-	SWS_Icu_00380
BSW00431	-	SWS_Icu_00380

BSW00434	-	SWS_Icu_00380
SRS_BSW_00005	Modules of the æC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_Icu_00380
SRS_BSW_00006	The source code of software modules above the æC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_Icu_00380
SRS_BSW_00007	All Basic SW Modules written in C language shall conform to the MISRA C 2004 Standard.	SWS_Icu_00380
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_Icu_00380
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_Icu_00380
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_Icu_00006
SRS_BSW_00160	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	SWS_Icu_00380
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_Icu_00380
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_Icu_00380
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_Icu_00380
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_Icu_00380
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_Icu_00380
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_Icu_00380

SRS_BSW_00171	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	SWS_Icu_00092, SWS_Icu_00095, SWS_Icu_00096, SWS_Icu_00097, SWS_Icu_00098, SWS_Icu_00099, SWS_Icu_00100, SWS_Icu_00101, SWS_Icu_00102, SWS_Icu_00103, SWS_Icu_00104, SWS_Icu_00105, SWS_Icu_00106, SWS_Icu_00122
SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	SWS_Icu_00380
SRS_BSW_00300	All AUTOSAR Basic Software Modules shall be identified by an unambiguous name	SWS_Icu_00380
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_Icu_00380
SRS_BSW_00302	All AUTOSAR Basic Software Modules shall only export information needed by other modules	SWS_Icu_00380
SRS_BSW_00304	All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types	SWS_Icu_00380
SRS_BSW_00305	Data types naming convention	SWS_Icu_00380
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_Icu_00380
SRS_BSW_00307	Global variables naming convention	SWS_Icu_00380
SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_Icu_00380
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_Icu_00380
SRS_BSW_00310	API naming convention	SWS_Icu_00380
SRS_BSW_00312	Shared code shall be reentrant	SWS_Icu_00380
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_Icu_00380
SRS_BSW_00318	Each AUTOSAR Basic Software Module file shall provide version numbers in the header file	SWS_Icu_00380
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software	SWS_Icu_00380

	Modules shall be enumerated according specific rules	
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_Icu_00022, SWS_Icu_00023, SWS_Icu_00024, SWS_Icu_00043, SWS_Icu_00120, SWS_Icu_00125
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_Icu_00380
SRS_BSW_00326	-	SWS_Icu_00380
SRS_BSW_00327	Error values naming convention	SWS_Icu_00380
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_Icu_00380
SRS_BSW_00329	-	SWS_Icu_00380
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_Icu_00380
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_Icu_00380
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_Icu_00380
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_Icu_00380
SRS_BSW_00335	Status values naming convention	SWS_Icu_00380
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_Icu_00037
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_Icu_00380
SRS_BSW_00342	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	SWS_Icu_00380
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_Icu_00006
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_Icu_00380
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_Icu_00380
SRS_BSW_00350	All AUTOSAR Basic Software Modules shall apply a specific	SWS_Icu_00380

	naming rule for enabling/disabling the detection and reporting of development errors	
SRS_BSW_00353	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	SWS_Icu_00380
SRS_BSW_00355	-	SWS_Icu_00380
SRS_BSW_00357	For success/failure of an API call a standard return type shall be defined	SWS_Icu_00380
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_Icu_00380
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_Icu_00187
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_Icu_00380
SRS_BSW_00361	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header	SWS_Icu_00380
SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	SWS_Icu_00049
SRS_BSW_00370	All AUTOSAR Basic Software Modules shall group and out-source callback declarations in a separate header file	SWS_Icu_00380
SRS_BSW_00371	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	SWS_Icu_00380
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_Icu_00380
SRS_BSW_00376	-	SWS_Icu_00380
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_Icu_00380
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_Icu_00380

SRS_BSW_00379	All software modules shall provide a module identifier in the header file and in the module XML description file.	SWS_Icu_00380
SRS_BSW_00383	The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description	SWS_Icu_00380
SRS_BSW_00384	The Basic Software Module specifications shall specify at least in the description which other modules they require	SWS_Icu_00131
SRS_BSW_00387	The Basic Software Module specifications shall specify how the callback function is to be implemented	SWS_Icu_00380
SRS_BSW_00395	The Basic Software Module specifications shall list all configuration parameter dependencies	SWS_Icu_00380
SRS_BSW_00397	The configuration parameters in pre-compile time are fixed before compilation starts	SWS_Icu_00380
SRS_BSW_00398	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	SWS_Icu_00380
SRS_BSW_00399	Parameter-sets shall be located in a separate segment and shall be loaded after the code	SWS_Icu_00380
SRS_BSW_00400	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	SWS_Icu_00380
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_Icu_00006
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_Icu_00006
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_Icu_00022
SRS_BSW_00408	All AUTOSAR Basic Software Modules configuration parameters shall be named according to a specific naming rule	SWS_Icu_00380
SRS_BSW_00409	All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from	SWS_Icu_00380

	Dem configuration	
SRS_BSW_00410	Compiler switches shall have defined values	SWS_Icu_00055, SWS_Icu_00063, SWS_Icu_00090, SWS_Icu_00092, SWS_Icu_00095, SWS_Icu_00096, SWS_Icu_00097, SWS_Icu_00099, SWS_Icu_00100, SWS_Icu_00101, SWS_Icu_00102, SWS_Icu_00103, SWS_Icu_00104, SWS_Icu_00105, SWS_Icu_00106, SWS_Icu_00122
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_Icu_00380
SRS_BSW_00414	The init function may have parameters	SWS_Icu_00380
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_Icu_00380
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_Icu_00380
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_Icu_00380
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_Icu_00380
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_Icu_00380
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_Icu_00380
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_Icu_00380
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_Icu_00380
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_Icu_00380
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_Icu_00380
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_Icu_00380

SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_Icu_00380
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_Icu_00380
SRS_BSW_00435	Each AUTOSAR Basic Software Module implementation .c shall include its respective header file SchM_.h	SWS_Icu_00249
SRS_BSW_00436	Each AUTOSAR Basic Software Module implementation *.c shall include the support memory mapping.	SWS_Icu_00251
SRS_BSW_00437	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	SWS_Icu_00380
SRS_BSW_00439	Enable BSW modules to handle interrupts	SWS_Icu_00380
SRS_BSW_00440	The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via Rte_Call API	SWS_Icu_00380
SRS_BSW_00441	Naming convention for type, macro and function	SWS_Icu_00380
SRS_Icu_12305	The ICU driver shall allow to enable/disable the notification for an ICU channel at runtime	SWS_Icu_00009, SWS_Icu_00010, SWS_Icu_00042, SWS_Icu_00044
SRS_Icu_12368	The ICU driver shall support basic static configurations per channel	SWS_Icu_00039
SRS_Icu_12369	The ICU driver shall provide notification for an ICU Channel at the configured signal edge	SWS_Icu_00021
SRS_Icu_12370	The ICU driver shall provide a service for selecting the sleep mode	SWS_Icu_00008
SRS_Icu_12371	The ICU driver shall provide a synchronous service that returns the status of the ICU input	SWS_Icu_00030, SWS_Icu_00031, SWS_Icu_00032
SRS_Icu_12407	After initialization of the ICU driver all notifications shall be disabled	SWS_Icu_00040, SWS_Icu_00061
SRS_Icu_12408	-	SWS_Icu_00013, SWS_Icu_00014
SRS_Icu_12425	For each ICU Channel the 'property' that could be measured shall be configurable	SWS_Icu_00039, SWS_Icu_00088

SRS_Icu_12429	The ICU Driver shall provide the functionality to deinitialize ICU channels to their power on reset state	SWS_Icu_00036
SRS_Icu_12430	The ICU driver shall provide an asynchronous service for starting the timestamp measurement on an ICU channel	SWS_Icu_00063, SWS_Icu_00066
SRS_Icu_12431	The ICU driver shall provide a synchronous service for canceling the timestamp measurement on an ICU channel	SWS_Icu_00067
SRS_Icu_12432	Edge counting service shall be available on an ICU channel	SWS_Icu_00078
SRS_Icu_12433	Edge counting service on a ICU channel shall be disabled	SWS_Icu_00079
SRS_Icu_12434	Edge counting read service shall be available	SWS_Icu_00080
SRS_Icu_12435	The elapsed Signal High Time for each ICU Channel shall be provided	SWS_Icu_00082
SRS_Icu_12436	The High time and Period Time of an ICU Channel shall be provided	SWS_Icu_00084
SRS_Icu_12438	The ICU driver shall provide the functionality to capture timer values on configurable edges to an external buffer	SWS_Icu_00063
SRS_Icu_12439	Edges of a signal shall be counted by the ICU	SWS_Icu_00072, SWS_Icu_00073, SWS_Icu_00074
SRS_Icu_12442	The elapsed Signal Low Time for each ICU Channel shall be provided	SWS_Icu_00081
SRS_Icu_12443	The elapsed Period Time for an ICU Channel shall be provided	SWS_Icu_00083
SRS_Icu_12444	The ICU driver shall provide a notification if the number of requested timestamps are acquired	SWS_Icu_00215
SRS_Icu_12453	The Timestamp index service shall be provided by ICU	SWS_Icu_00071
SRS_Icu_12455	If circular buffer handling is configured, the driver shall restart at the beginning of the external buffer, when the end of the buffer is reached	SWS_Icu_00039
SRS_Icu_12456	If linear buffer handling is configured, the driver shall stop capturing timer values, when the end of the buffer is reached	SWS_Icu_00039, SWS_Icu_00065

SRS_Icu_13100	Reseting the value of counted edges of an ICU channel shall be available	SWS_Icu_00072
SRS_SPAL_00157	All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers	SWS_Icu_00021, SWS_Icu_00030
SRS_SPAL_12056	All driver modules shall allow the static configuration of notification mechanism	SWS_Icu_00018
SRS_SPAL_12057	All driver modules shall implement an interface for initialization	SWS_Icu_00006, SWS_Icu_00040, SWS_Icu_00060, SWS_Icu_00061
SRS_SPAL_12063	All driver modules shall only support raw value mode	SWS_Icu_00063, SWS_Icu_00081, SWS_Icu_00082, SWS_Icu_00083
SRS_SPAL_12064	All driver modules shall raise an error if the change of the operation mode leads to degradation of running operations	SWS_Icu_00133
SRS_SPAL_12067	All driver modules shall set their wake-up conditions depending on the selected operation mode	SWS_Icu_00008, SWS_Icu_00011, SWS_Icu_00012
SRS_SPAL_12068	The modules of the MCAL shall be initialized in a defined sequence	SWS_Icu_00380
SRS_SPAL_12069	All drivers of the SPAL that wake up from a wake-up interrupt shall report the wake-up reason	SWS_Icu_00055, SWS_Icu_00056, SWS_Icu_00057
SRS_SPAL_12075	All drivers with random streaming capabilities shall use application buffers	SWS_Icu_00063
SRS_SPAL_12077	All drivers shall provide a non blocking implementation	SWS_Icu_00380
SRS_SPAL_12092	The driver's API shall be accessed by its handler or manager	SWS_Icu_00380
SRS_SPAL_12125	All driver modules shall only initialize the configured resources	SWS_Icu_00054
SRS_SPAL_12129	The ISRs shall be responsible for resetting the interrupt flags and calling the according notification function	SWS_Icu_00119
SRS_SPAL_12163	All driver modules shall implement an interface for de-initialization	SWS_Icu_00036, SWS_Icu_00037
SRS_SPAL_12169	All driver modules that provide different operation modes shall provide a service for mode	SWS_Icu_00008

	selection	
SRS_SPAL_12265	Configuration data shall be kept constant	SWS_Icu_00380
SRS_SPAL_12448	All driver modules shall have a specific behavior after a development error detection	SWS_Icu_00049, SWS_Icu_00107, SWS_Icu_00108
SRS_SPAL_12461	Specific rules regarding initialization of controller registers shall apply to all driver implementations	SWS_Icu_00006, SWS_Icu_00051, SWS_Icu_00052, SWS_Icu_00053, SWS_Icu_00128, SWS_Icu_00129
SRS_SPAL_12463	The register initialization settings shall be combined and forwarded	SWS_Icu_00380

7 Functional specification

7.1 General behavior

7.1.1 Background & Rationale

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

7.1.2 Requirements

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

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

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

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

[SWS_Icu_00258] ⌈The Icu module has 2 modes:

- `ICU_MODE_NORMAL`
- `ICU_MODE_SLEEP`

⌋ ()

In `ICU_MODE_NORMAL` mode all notifications are available as

- **[SWS_Icu_00011]** ⌈configured by service `Icu_SetActivationCondition() or IcuDefaultStartEdge.` ⌋ (SRS_SPAL_12067)
- **[SWS_Icu_00259]** ⌈selected by the `Icu_DisableNotification()` and `Icu_EnableNotification()` services before or after the call of `Icu_SetMode().`⌋ ()

In `ICU_MODE_SLEEP` mode

- **[SWS_Icu_00012]** ⌈only those wakeup events are available which are configured as wakeup capable, enabled via `Icu_EnableWakeup()` after `Icu_Init()` and which are not disabled via service `Icu_DisableWakeup()`⌋ (SRS_SPAL_12067)

- **[SWS_Icu_00260]** ⌈all other interrupts handled by this module are disabled and must not lead to an exit from the reduced power mode state (e.g. idle, halt) of the MCU if the event occurs. ⌋ ()
- **[SWS_Icu_00261]** ⌈All channels are stopped except those channels
  - which have been configured as wakeup capable and
  - which were explicitly enabled by the call of `Icu_EnableWakeup`.⌋ ()

[SWS_Icu_00088] ⌈The module Icu shall allow the configuration per channel of the definition on which edge the period starts.⌋ (SRS_Icu_12425)

7.1.3 Time Unit Ticks

7.1.3.1 Background & Rationale

To get times out of register values it is necessary to know the oscillator frequency, prescalers and so on. Since these settings are made in the MCU module and/or in other modules it is not possible to calculate such times.
Hence the conversions between time and ticks shall be part of an upper layer.

7.1.3.2 Requirements

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

7.2 Error classification

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

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]*
[ICU001] ⌈The Development error `ICU_E_PARAM_CONFIG (0x0A)` shall be detectable by the ICU driver depending on its build version when API `Icu_Init` service called with wrong parameter⌋ (SRS_BSW_00337, SRS_BSW_00385)	Development	`ICU_E_PARAM_CONFIG`	`0x0A`
[ICU272] ⌈The Development error `ICU_E_PARAM_CHANNEL (0x0B)` shall be detectable by the ICU driver depending on its build version when 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`
[ICU264] ⌈The Development error `ICU_E_PARAM_ACTIVATION (0x0C)` shall be detectable by the ICU driver depending on its build version when API service used with an invalid or not feasible activation.⌋ ()	Development	`ICU_E_PARAM_ACTIVATION`	`0x0C`
[ICU265] ⌈The Development error `ICU_E_PARAM_BUFFER_PTR (0x0D)` shall be detectable by the ICU driver depending on its build version when API service used with an invalid application-buffer pointer.⌋ ()	Development	`ICU_E_PARAM_BUFFER_PTR`	`0x0D`
[ICU266] ⌈The Development error `ICU_E_PARAM_BUFFER_SIZE (0x0E)` shall be detectable by the ICU driver depending on its build version when API service used with an invalid buffer size.⌋ ()	Development	`ICU_E_PARAM_BUFFER_SIZE`	`0x0E`
[ICU267] ⌈The Development error `ICU_E_PARAM_MODE (0x0F)` shall be detectable by the ICU driver depending on its build version when API service `Icu_SetMode` used with an invalid mode.⌋ ()	Development	`ICU_E_PARAM_MODE`	`0x0F`
[ICU268] ⌈The Development error `ICU_E_UNINIT (0x14)` shall be detectable by the ICU driver depending on its build version when API service used without module initialization.⌋ ()	Development	`ICU_E_UNINIT`	`0x14`
[ICU269] ⌈The Development error	Development	`ICU_E_NOT_STARTED`	`0x15`

Type or error	*Relevance*	*Related error code*	*Value [hex]*
ICU_E_NOT_STARTED (0x15) shall be detectable by the ICU driver depending on its build version when API service Icu_StopTimestamp called on a channel which was not started or already stopped⌋ ()			
[ICU270] ⌈The Development error ICU_E_BUSY_OPERATION (0x16) shall be detectable by the ICU driver depending on its build version when API service Icu_SetMode is called while a running operation.⌋ ()	Development	ICU_E_BUSY_OPERATION	0x16
[ICU271] ⌈The Development error ICU_E_ALREADY_INITIALIZED (0x17) shall be detectable by the ICU driver depending on its build version when API Icu_Init service is called and when the ICU driver and the Hardware are already initialized.⌋ ()	Development	ICU_E_ALREADY_INITIALIZED	0x17
[ICU355] ⌈The Development error ICU_E_PARAM_NOTIFY_INTERVAL(0x18) shall be detectable by the ICU driver depending on its build version when API Icu_StartTimeStamp is called and the parameter NotifyInterval is invalid (e.g."0", NotifyInterval < 1) ⌋ ()	Development	ICU_E_PARAM_NOTIFY_INTERVAL	0x18
[ICU357] ⌈The development error ICU_E_PARAM_VINFO (0x19) shall be detectable by the ICU driver depending on its build version when API Icu_GetVersionInfo is called and the parameter versioninfo is is invalid (e.g. NULL) ⌋ ()	Development	ICU_E_PARAM_VINFO	0x19
None	Production	None	Assigned by DEM

7.3 Error detection

[SWS_Icu_00022] ⌈If development error detection for the Icu module is enabled: All Icu module functions, except for Icu_Init and Icu_GetVersionInfo, shall raise development error ICU_E_UNINIT when the function Icu_Init has not been called. ⌋ (SRS_BSW_00323, SRS_BSW_00406)

7.4 Debugging Concept

7.4.1 Background & Rationale

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

8 API specification

8.1 Imported types

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

[SWS_Icu_00190] ⌈`Dem_EventIdType` shall be imported from `Dem_Types.h`.⌋ ()

[SWS_Icu_00276] ⌈`EcuM_WakeupSourceType` shall be imported from `EcuM_Types.h`.

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

⌋ ()

8.2 Type definitions

8.2.1 Icu_ModeType

[SWS_Icu_00277] ⌈

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

[SWS_Icu_00278] ⌈

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

⌋ ()

8.2.3 Icu_InputStateType

[SWS_Icu_00279] ⌈

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

[SWS_Icu_00280] ⌈

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.	

⌋ ()

[SWS_Icu_00281] ⌈The `Icu_ConfigType` shall contain:
Optional parameters
- MCU dependent properties for used HW units.
- Clock source with optional prescaler (if provided by HW).⌋ ()

[SWS_Icu_00039] ⌈The definition for each Channel within the `Icu_ConfigType` shall contain:
Common parameters
- Default Start Edge
- Hardware Specific Settings per channel
- Measurement Mode
    - Signal Edge Detection / Notification
    - Signal Measurement
    - Timestamp
    - Edge Counter

Specific parameters

⌋ (SRS_Icu_12368, SRS_Icu_12425, SRS_Icu_12455, SRS_Icu_12456)

[SWS_Icu_00283] ⌈If the measurement mode for each Channel within the `Icu_ConfigType` is configured as “signal edge detection” the notification function for signal notification shall be configurable.⌋ ()

[SWS_Icu_00284] ⌈If the measurement mode for each Channel within the `Icu_ConfigType` is configured as “signal measurement”, the property that could be measured shall be configurable. The values shall be as specified in SWS_Icu_00295.⌋ ()

[SWS_Icu_00285] ⌈If the measurement mode for each Channel within the `Icu_ConfigType` is configured as “timestamp measurement”, buffer handling shall be configurable. The values shall be as specified in SWS_Icu_00296.⌋ ()

[SWS_Icu_00378] ⌈If the measurement mode for each Channel within the `Icu_ConfigType` is configured as “timestamp measurement”, the notification function for notifying the number of requested timestamps shall be configurable. ⌋ ()

[SWS_Icu_00286] ⌈If the measurement mode for each Channel within the `Icu_ConfigType` is configured as “edge counter”, the counting mode (activation edge) shall be configurable. The values shall be as specified in SWS_Icu_00289.⌋ ()

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

[SWS_Icu_00288] ⌈If, in the definition for each Channel within the `Icu_ConfigType`, the channel is configured as wakeup capable then the value transmitted to the EcuM shall be configurable.⌋ ()

8.2.5 Icu_ActivationType

[SWS_Icu_00289] ⌈

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

[SWS_Icu_00290] ⌈

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

⌋ ()

8.2.7 Icu_DutyCycleType

[SWS_Icu_00291] ⌈

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

[SWS_Icu_00292] ⌈

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

⌋ ()

8.2.9 Icu_EdgeNumberType

[SWS_Icu_00293] ⌈

Name:	Icu_EdgeNumberType		
Type:	uint		
Range:	--	--	Implementation specific. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.

Description:	Type, to abstract the return value of the service Icu_GetEdgeNumbers().

⌋ ()

8.2.10 Icu_MeasurementModeType

[SWS_Icu_00294] ⌈

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

[SWS_Icu_00295] ⌈

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

[SWS_Icu_00296] ⌈

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

[SWS_Icu_00191] ⌈

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.	

⌋ ()

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

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

[SWS_Icu_00006] ⌈The function `Icu_Init` shall initialize all relevant registers of the configured hardware with the values of the structure referenced by the parameter `ConfigPtr`.⌋ (SRS_BSW_00344, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00101, SRS_SPAL_12057, SRS_SPAL_12461)

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

- **[SWS_Icu_00051]** ⌈If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register.⌋ (SRS_SPAL_12461)
- **[SWS_Icu_00052]** ⌈If the register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver.⌋ (SRS_SPAL_12461)
- **[SWS_Icu_00053]** ⌈If the register can affect several hardware modules and if it is not an I/O register it shall be initialized by the MCU driver.⌋ (SRS_SPAL_12461)
- **[SWS_Icu_00128]** ⌈One-time writable registers that require initialization directly after reset shall be initialized by the start-up code.⌋ (SRS_SPAL_12461)
- **[SWS_Icu_00129]** ⌈All other registers shall be initialized by the startup code.⌋ (SRS_SPAL_12461)

[SWS_Icu_00061] ⌈The function `Icu_Init` shall disable all notifications.⌋ (SRS_SPAL_12057, SRS_Icu_12407)

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

[SWS_Icu_00040] ⌈The function `Icu_Init` shall set all used ICU channels to status `ICU_IDLE`.⌋ (SRS_SPAL_12057, SRS_Icu_12407)

[SWS_Icu_00060] ⌈The function `Icu_Init` shall set the module mode to `ICU_MODE_NORMAL`.⌋ (SRS_SPAL_12057)

[SWS_Icu_00054] ⌈The function `Icu_Init` shall only set the resources that are configured in the configuration file (including clearing of pending interrupt flags).

The Icu module's environment shall not call `Icu_Init` during a running operation (e. g. timestamp measurement or edge counting).⌋ (SRS_SPAL_12125)

[SWS_Icu_00023] ⌈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.⌋ (SRS_BSW_00323)

[SWS_Icu_00220] ⌈If development error detection for the ICU module is enabled and the function `Icu_Init` is called when the ICU driver and hardware are already initialized, the function `Icu_Init` shall raise development error `ICU_E_ALREADY_INITIALIZED` and return without any action.⌋ ()

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

[SWS_Icu_00148] ⌈If not applicable, a NULL pointer shall be passed to the initialization routine. In this case the check for this NULL pointer (SWS_Icu_00023) has to be omitted

SWS_Icu_00048 applies to the function `Icu_Init`.⌋ ()

8.3.2 Icu_DeInit

[SWS_Icu_00193] ⌈

Service name:	Icu_DeInit

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

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

⌋ ()

[SWS_Icu_00036] ⌈The function `Icu_DeInit` shall set the state of the peripherals used by configuration as the same after power on reset.⌋ (SRS_SPAL_12163, SRS_Icu_12429)

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

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

[SWS_Icu_00037] ⌈The function `Icu_DeInit` shall disable all used interrupts and notifications.⌋ (SRS_BSW_00336, SRS_SPAL_12163)

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

[SWS_Icu_00092] ⌈The function `Icu_DeInit` shall be pre compile time configurable by configuration parameter IcuDeInitApi.⌋ (SRS_BSW_00410, SRS_BSW_00171)

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

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_DeInit`.⌋ ()

8.3.3 Icu_SetMode

[SWS_Icu_00194] ⌈

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.

⌋()

[SWS_Icu_00008] ⌈The function `Icu_SetMode` shall set the operation mode to the given mode parameter. The function `Icu_SetMode` shall set the operation mode to the given mode parameter. This function influences the functionality of the ICU channels. Therefore the mode switching of the module shall be compatible to the overall state of the ECU.⌋(SRS_SPAL_12067, SRS_SPAL_12169, SRS_Icu_12370)

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

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

[SWS_Icu_00095] ⌈The function `Icu_SetMode` shall be pre-compile time configurable by the configuration parameter `IcuSetModeApi`.⌋(SRS_BSW_00410, SRS_BSW_00171)

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

[SWS_Icu_00125] ⌈If development error detection is enabled for the module Icu the function `Icu_SetMode` shall check the parameter `Mode` and shall raise the error `ICU_E_PARAM_MODE` if the parameter `Mode` is not within the allowed range set in the configuration.⌋(SRS_BSW_00323)

[SWS_Icu_00133] ⌈This service can be called during running operations. If so, an ongoing operation that generates interrupts on a wakeup capable channel like e.g. time stamping or edge counting might lead to the ICU module not being able to properly enter sleep mode. This is then a system or ECU configuration issue not a problem of this specification.

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_SetMode`.⌋(SRS_SPAL_12064)

8.3.4 Icu_DisableWakeup

[SWS_Icu_00195] ⌈

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.	

⌋()

[SWS_Icu_00013] ⌈The function `Icu_DisableWakeup` shall disable the wakeup capability of a single ICU channel.⌋(SRS_Icu_12408)

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

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

[SWS_Icu_00096] ⌈The function `Icu_DisableWakeup` shall be pre compile time configurable by the configuration parameter `IcuDisableWakeupApi`.⌋ (SRS_BSW_00410, SRS_BSW_00171)

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

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

[SWS_Icu_00024] ⌈If development error detection is enabled: The function `Icu_DisableWakeup` shall check the parameter `Channel` and shall raise development error `ICU_E_PARAM_CHANNEL` if `Channel` is not within the allowed range set in the configuration.⌋(SRS_BSW_00323)

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function Icu_DisableWakeup.⌋()

8.3.5 Icu_EnableWakeup

[SWS_Icu_00196] ⌈

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.

⌋ ()

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

[SWS_Icu_00014] ⌈The function `Icu_EnableWakeup` shall re-enable the wakeup capability of a single ICU channel for the following ICU mode selection(s). This service is only feasible for ICU channels configured as wakeup capable true.

To make the selection effective a call of the function `Icu_SetMode`, requesting the mode `ICU_MODE_SLEEP` is required.⌋ (SRS_Icu_12408)

[SWS_Icu_00097] ⌈The function `Icu_EnableWakeup` shall be pre compile time configurable by configuration parameter `IcuEnableWakeupApi`.⌋ (SRS_BSW_00410, SRS_BSW_00171)

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

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_EnableWakeup`.⌋ ()

8.3.6 Icu_CheckWakeup

[SWS_Icu_00358] ⌈

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

⌋ ()

[SWS_Icu_00359] ⌈The function `Icu_CheckWakeup` shall check if a wakeup capable ICU channel is the source for a wakeup event and call `EcuM_SetWakeupEvent` to indicate a valid timer wakeup event to the ECU State Manager.⌋ ()

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

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

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

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

8.3.7 Icu_SetActivationCondition

[SWS_Icu_00197] ⌈

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.	

⌋ ()

[SWS_Icu_00090] ⌈The function `Icu_SetActivationCondition` shall set the activation-edge according to Activation parameter for the given channel. This service shall support channels which are configured for the following `IcuMeasurementMode` (for details refer to 8.2.10)

- `ICU_MODE_SIGNAL_EDGE_DETECT`
- `ICU_MODE_TIMESTAMP`
- `ICU_MODE_EDGE_COUNTER`⌋ (SRS_BSW_00410)

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

[SWS_Icu_00309] ⌈The function `Icu_SetActivationCondition` shall be re-entrant.⌋ ()

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

[SWS_Icu_00043] ⌈If development error detection is enabled the function `Icu_SetActivationCondition` shall check the parameter `Activation`. The function `Icu_SetActivationCondition` shall raise the error

ICU_E_PARAM_ACTIVATION if Activation is invalid but only for the requested ICU channel.

SWS_Icu_00022 and SWS_Icu_00048 apply to the function Icu_SetActivationCondition.⌋ (SRS_BSW_00323)

8.3.8 Icu_DisableNotification

[SWS_Icu_00198] ⌈

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.

⌋ ()

[SWS_Icu_00009] ⌈The function Icu_DisableNotification shall disable the notification on the given channel.⌋ (SRS_Icu_12305)

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function Icu_DisableNotification.⌋ ()

8.3.9 Icu_EnableNotification

[SWS_Icu_00199] ⌈

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.

⌋ ()

[SWS_Icu_00010] ⌈The function `Icu_EnableNotification` shall enable the notification on the given channel.⌋ (SRS_Icu_12305)

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_EnableNotification`.⌋ ()

8.3.10 Icu_GetInputState

[SWS_Icu_00200] ⌈

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.	

⌋ ()

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

[SWS_Icu_00030] ⌈The function `Icu_GetInputState` shall return the status of the ICU input. Only channels which are configured for the following `IcuMeasurementMode` shall be supported:

- `ICU_MODE_SIGNAL_EDGE_DETECT`
- `ICU_MODE_SIGNAL_MEASUREMENT`⌋ (SRS_SPAL_00157, SRS_Icu_12371)

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

[SWS_Icu_00031] ⌈If an activation edge has been detected the function `Icu_GetInputState` shall return `ICU_ACTIVE` for Edge Detection channels.⌋ (SRS_Icu_12371)

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

[SWS_Icu_00032] ⌈Once the function `Icu_GetInputState` has returned the status `ICU_ACTIVE`, the function `Icu_GetInputState` shall set the stored status to `ICU_IDLE` until the next edge is detected.⌋ (SRS_Icu_12371)

[SWS_Icu_00122] ⌈The function `Icu_GetInputState` shall be pre compile time configurable by the configuration parameter `IcuGetInputStateApi`.⌋ (SRS_BSW_00410, SRS_BSW_00171)

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

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

[SWS_Icu_00049] ⌈If development error detection is enabled the function `Icu_GetInputState` shall return `ICU_IDLE` if an error is detected.

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_GetInputState`.⌋ (SRS_SPAL_12448, SRS_BSW_00369)

8.3.11 Icu_StartTimestamp

[SWS_Icu_00201] ⌈

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.	

⌋ ()

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

[SWS_Icu_00063] ⌈The function `Icu_StartTimestamp` shall start the capturing of timer values on the edges

- activated by the service Icu_SetActivationCondition ( )

  (rising / falling / both edges)⌋ (SRS_BSW_00410, SRS_SPAL_12063, SRS_SPAL_12075, SRS_Icu_12430, SRS_Icu_12438)

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

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

[SWS_Icu_00065] ⌈If linear buffer handling is configured, when the capture functionality reaches the end of the buffer, the Icu module shall stop capturing timer values.⌋ (SRS_Icu_12456)

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

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

[SWS_Icu_00319] ⌈The Icu module shall only call a notification function if `NotifyInterval` is greater than “0”.⌋ ()

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

[SWS_Icu_00066] ⌈The function `Icu_StartTimeStamp` shall only be available in Measurement Mode “`ICU_MODE_TIMESTAMP`”.⌋ (SRS_Icu_12430)

[SWS_Icu_00098] ⌈The function `Icu_StartTimestamp` shall be pre-compile time configurable by the configuration parameter: `ICU_TIMESTAMP_API`.⌋ (SRS_BSW_00171)

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

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

[SWS_Icu_00120] ⌈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” means NULL pointer).⌋ (SRS_BSW_00323)

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

[SWS_Icu_00108] ⌈If development error detection is enabled the function `Icu_StartTimestamp` shall check the parameter `BufferSize` (check that size > 0). The function `Icu_StartTimestamp` shall raise the error `ICU_E_PARAM_BUFFER_SIZE` if `BufferSize` is invalid (e.g. “0”).

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_StartTimestamp`.⌋ (SRS_SPAL_12448)

8.3.12 Icu_StopTimestamp

[SWS_Icu_00202] ⌈

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.	

⌋ ()

[SWS_Icu_00067] ⌈The function `Icu_StopTimestamp` shall stop the timestamp measurement of the given channel.⌋ (SRS_Icu_12431)

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

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

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

[SWS_Icu_00099] ⌈The function `Icu_StopTimestamp` shall be pre-compile time configurable by the configuration parameter: `IcuTimestampApi` (see also chapter 10.2.4. Configuration of optional API services)⌋ (SRS_BSW_00410, SRS_BSW_00171)

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

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

[SWS_Icu_00166] ⌈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).

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_StopTimestamp`.⌋ ()

8.3.13 Icu_GetTimestampIndex

[SWS_Icu_00203] ⌈

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.	

⌋ ()

[SWS_Icu_00071] ⌈The function `Icu_GetTimestampIndex` shall read the timestamp index of the given channel, which is the next to be written.⌋ (SRS_Icu_12453)

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

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

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

[SWS_Icu_00100] ⌈The function `Icu_GetTimestampIndex` shall be pre compile time configurable by the configuration parameter: `IcuTimestampApi`⌋ (SRS_BSW_00410, SRS_BSW_00171)

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

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

[SWS_Icu_00107] ⌈If development error detection is enabled the function `Icu_GetTimestampIndex` shall return "0" if an error is detected.

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_GetTimestampIndex`. ⌋ (SRS_SPAL_12448)

8.3.14 Icu_ResetEdgeCount

[SWS_Icu_00204] ⌈

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.	

⌋ ()

[SWS_Icu_00072] ⌈The function `Icu_ResetEdgeCount` shall reset the value of the counted edges to zero.⌋ (SRS_Icu_12439, SRS_Icu_13100)

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

[SWS_Icu_00101] ⌈The function `Icu_ResetEdgeCount` shall be pre-compile time configurable by the configuration parameter `ICU_EDGE_COUNT_API`.⌋ (SRS_BSW_00410, SRS_BSW_00171)

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_ResetEdgeCount`.⌋ ()

8.3.15 Icu_EnableEdgeCount

[SWS_Icu_00205] ⌈

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.

⌋ ()

[SWS_Icu_00078] ⌈The function `Icu_EnableEdgeCount` shall enable the counting of edges of the given channel.⌋ (SRS_Icu_12432)

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

[SWS_Icu_00073] ⌈The function `Icu_EnableEdgeCount` shall only count the configured[1] edges (rising edge / falling edge / both edges).⌋ (SRS_Icu_12439)

[SWS_Icu_00074] ⌈The function `Icu_EnableEdgeCount` shall be available for each ICU channel in Measurement Mode "Edge Counter".⌋ (SRS_Icu_12439)

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

[SWS_Icu_00102] ⌈The function `Icu_EnableEdgeCount` shall be pre-compile time configurable by the configuration parameter `ICU_EDGE_COUNT_API`⌋ (SRS_BSW_00410, SRS_BSW_00171)

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_EnableEdgeCount`.⌋ ()

8.3.16 Icu_EnableEdgeDetection

[SWS_Icu_00364] ⌈

Service name:	Icu_EnableEdgeDetection

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

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

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

⌋ ()

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

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

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

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

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

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_EnableEdgeDetection`⌋ ()

8.3.17 Icu_DisableEdgeDetection

[SWS_Icu_00377] ⌈

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

⌋ ()

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

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

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

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_DisableEdgeDetection`.⌋ ()

8.3.18 Icu_DisableEdgeCount

[SWS_Icu_00206] ⌈

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.	

⌋ ()

[SWS_Icu_00079] ⌈The function `Icu_DisableEdgeCount` shall disable the counting of edges of the given channel.⌋ (SRS_Icu_12433)

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

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

[SWS_Icu_00103] ⌈The function `Icu_DisableEdgeCount` shall be pre-compile time configurable by the configuration parameter `IcuEdgeCountApi`.⌋ (SRS_BSW_00410, SRS_BSW_00171)

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_DisableEdgeCount`.⌋ ()

8.3.19 Icu_GetEdgeNumbers

[SWS_Icu_00207] ⌈

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.	

⌋ ()

[SWS_Icu_00080] ⌈The function `Icu_GetEdgeNumbers` shall read the number of counted edges after the last call of Icu_ResetEdgeCount `()`.⌋ (SRS_Icu_12434)

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

[SWS_Icu_00104] ⌈The function `Icu_GetEdgeNumbers` shall be pre compile time configurable by the configuration parameter: `ICU_EDGE_COUNT_API`⌋ (SRS_BSW_00410, SRS_BSW_00171)

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

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_GetEdgeNumbers`.⌋ ()

8.3.20 Icu_StartSignalMeasurement

[SWS_Icu_00208] ⌈

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	None	

(inout):	
Parameters (out):	None
Return value:	None
Description:	This function starts the measurement of signals.

⌋ ()

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

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

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

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

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

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_StartSignalMeasurement`.⌋ ()

8.3.21 Icu_StopSignalMeasurement

[SWS_Icu_00209] ⌈

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.	

⌋ ()

[SWS_Icu_00336] ⌈The function `Icu_StopSignalMeasurement` shall be Re-entrant.⌋ ()

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

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

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

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function Icu_StopSignalMeasurement.⌋ ()

8.3.22 Icu_GetTimeElapsed

[SWS_Icu_00210] ⌈

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.	

⌋()

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

[SWS_Icu_00081] ⌈The function `Icu_GetTimeElapsed` shall read the elapsed Signal Low Time for the given channel that is configured in Measurement Mode “Signal Measurement, Signal Low Time”. The elapsed time is measured between a falling edge and the consecutive rising edge of the channel.⌋(SRS_SPAL_12063, SRS_Icu_12442)

[SWS_Icu_00082] ⌈The function `Icu_GetTimeElapsed` shall read the elapsed Signal High Time for the given channel that is configured in Measurement Mode “Signal Measurement, Signal High Time”. The elapsed time is measured between a rising edge and the consecutive falling edge of the channel.⌋(SRS_SPAL_12063, SRS_Icu_12435)

[SWS_Icu_00083] ⌈The function `Icu_GetTimeElapsed` shall read the elapsed Signal Period Time for the given channel that is configured in Measurement Mode “Signal Measurement, Signal Period Time”. The elapsed time is measured between consecutive rising (or falling) edges of the channel. The period start edge is configurable.⌋(SRS_SPAL_12063, SRS_Icu_12443)

[SWS_Icu_00136] ⌈The function `Icu_GetTimeElapsed` shall return “0” in case no requested time has been captured (see

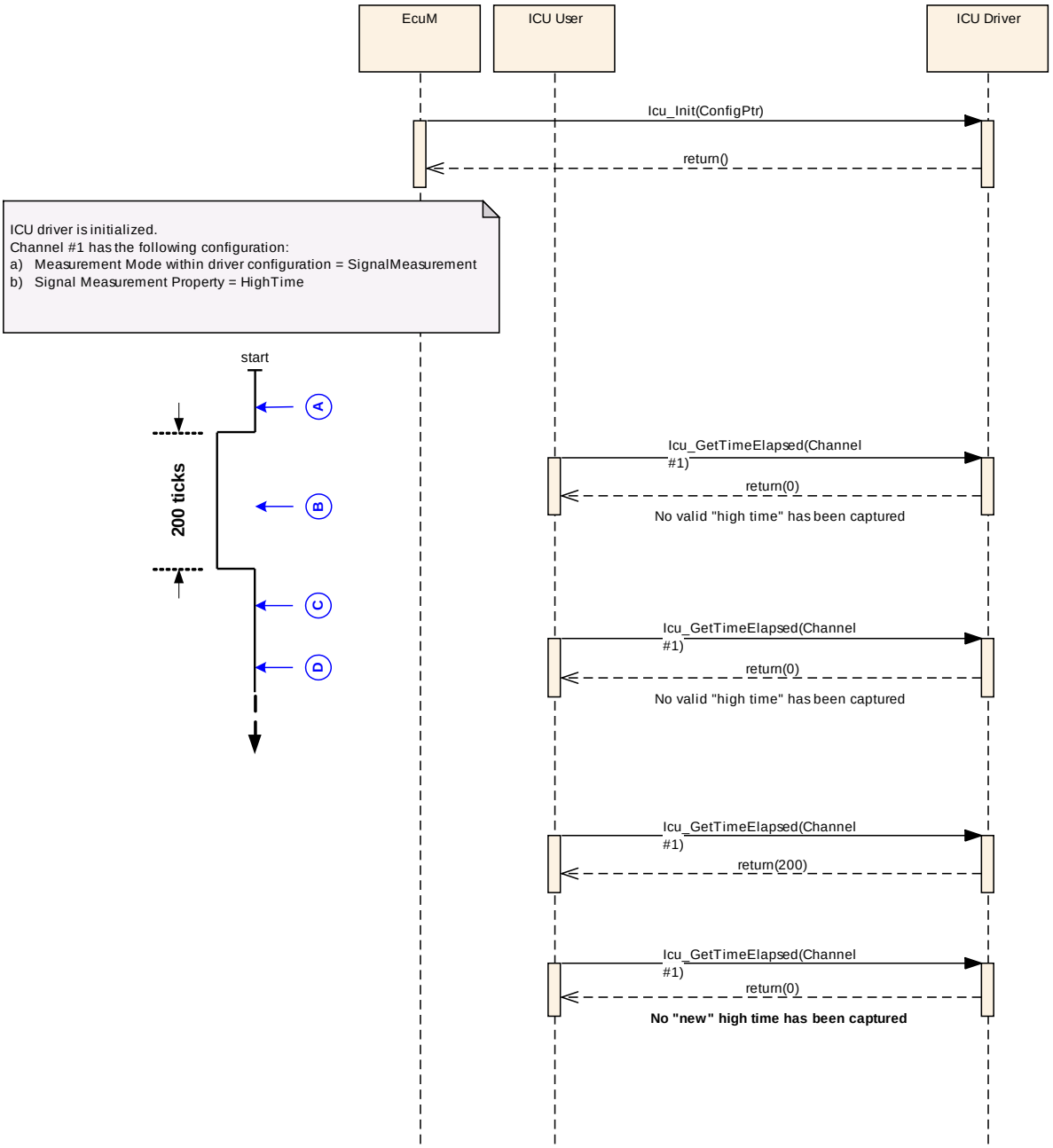


Figure 9.19, letter “A”).⌋ ()

[SWS_Icu_00339] ⌈The function `Icu_GetTimeElapsed` shall return “0” in case the capturing of a requested time is ongoing and not finished (see

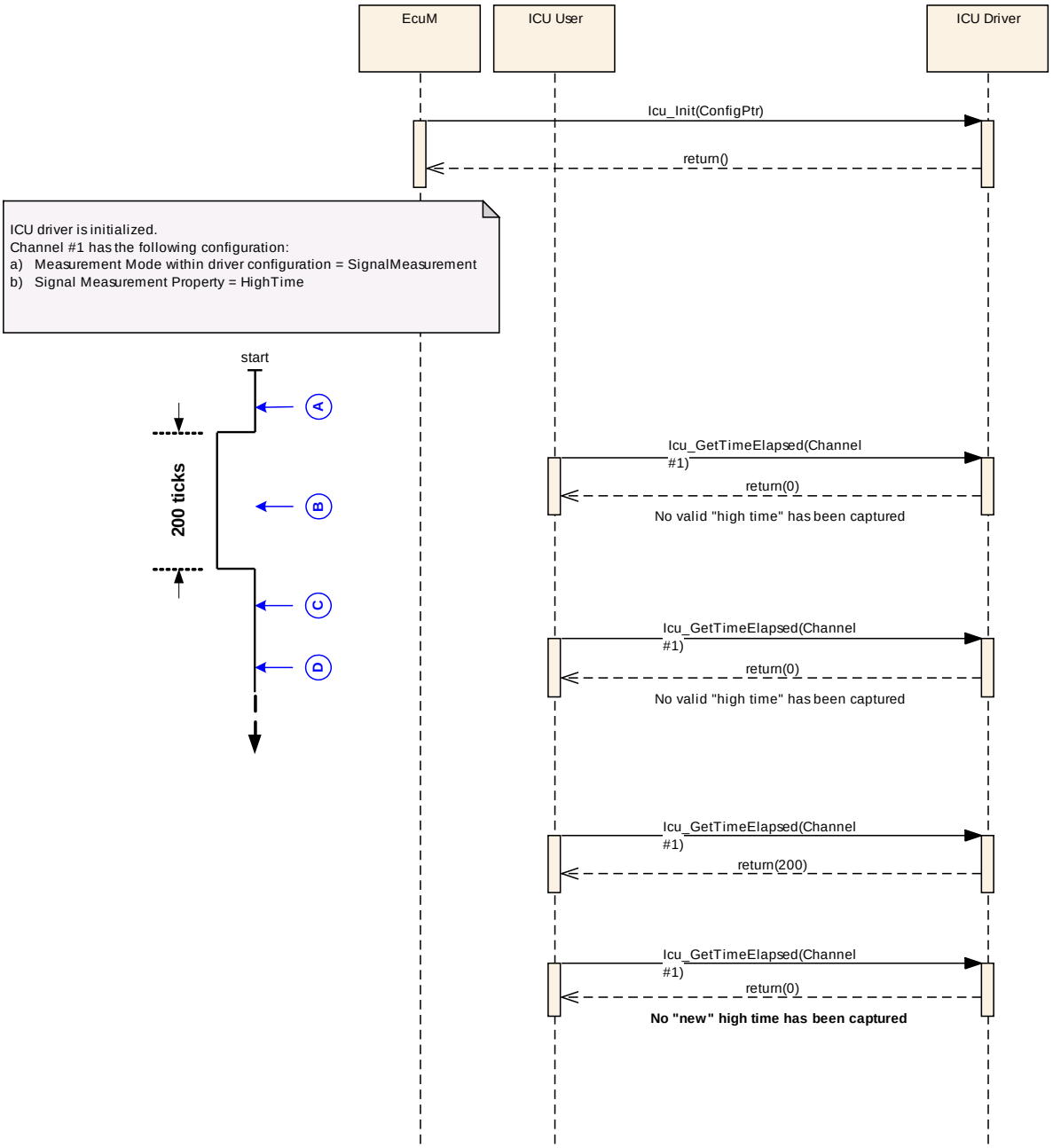


Figure 9.19, letter “B”)⌋ ()

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

again (see

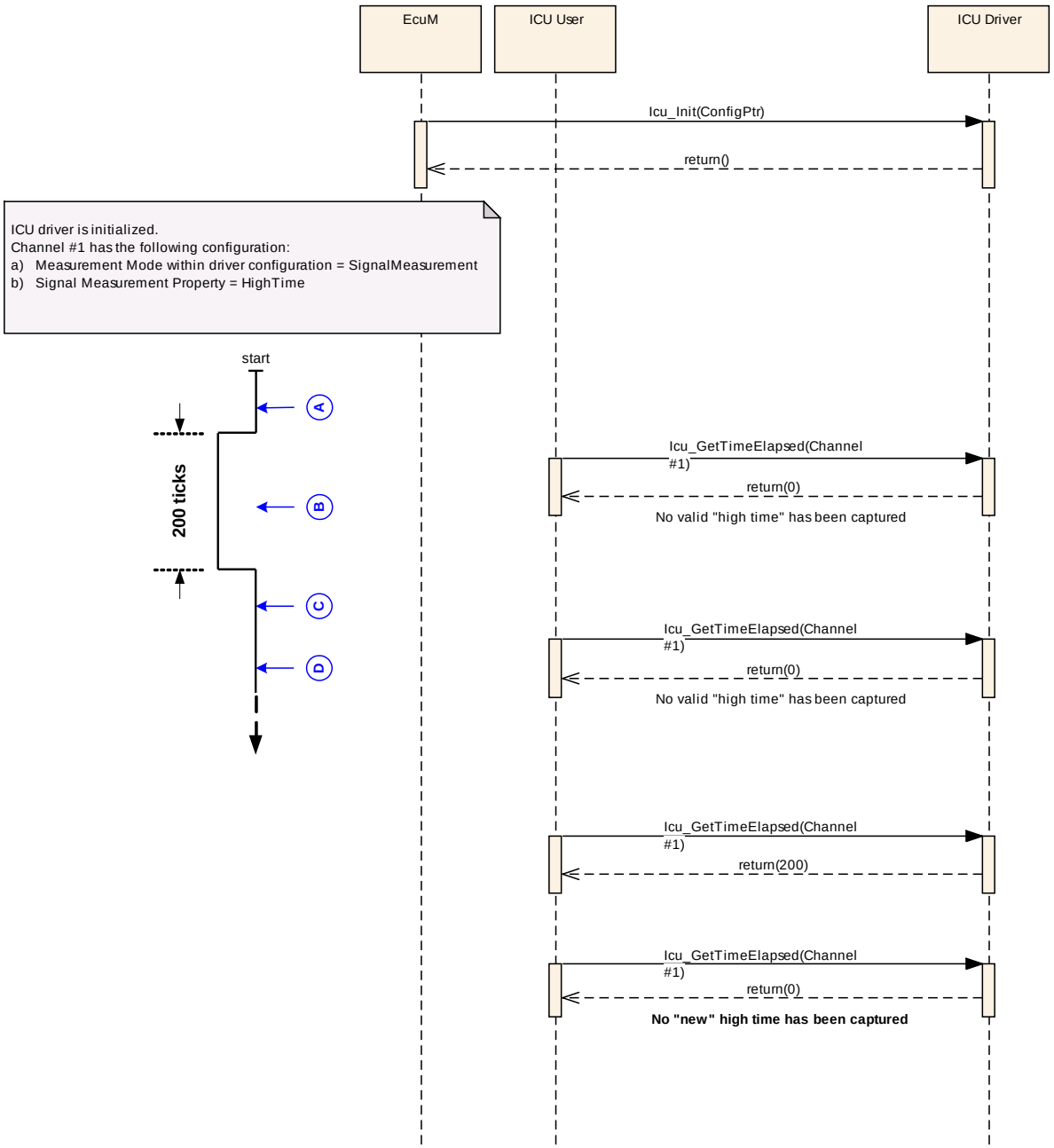


Figure 9.19, letter “D”)⌋ ()

[SWS_Icu_00105] ⌈The function `Icu_GetTimeElapsed` shall be pre compile time configurable by the configuration parameter `IcuGetTimeElapsedApi`.⌋ (SRS_BSW_00410, SRS_BSW_00171)

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

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_GetTimeElapsed`.⌋ ()

8.3.23 Icu_GetDutyCycleValues

[SWS_Icu_00211] ⌈

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.	

⌋ ()

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

[SWS_Icu_00084] ⌈The function `Icu_GetDutyCycleValues` shall read the coherent active time and period time for the given ICU Channel, if it is configured in Measurement Mode “Signal Measurement, Duty Cycle Values”.⌋ (SRS_Icu_12436)

[SWS_Icu_00137] ⌈The function `Icu_GetDutyCycleValues` shall return “0” in case no coherent active- and period time has been captured (similar to

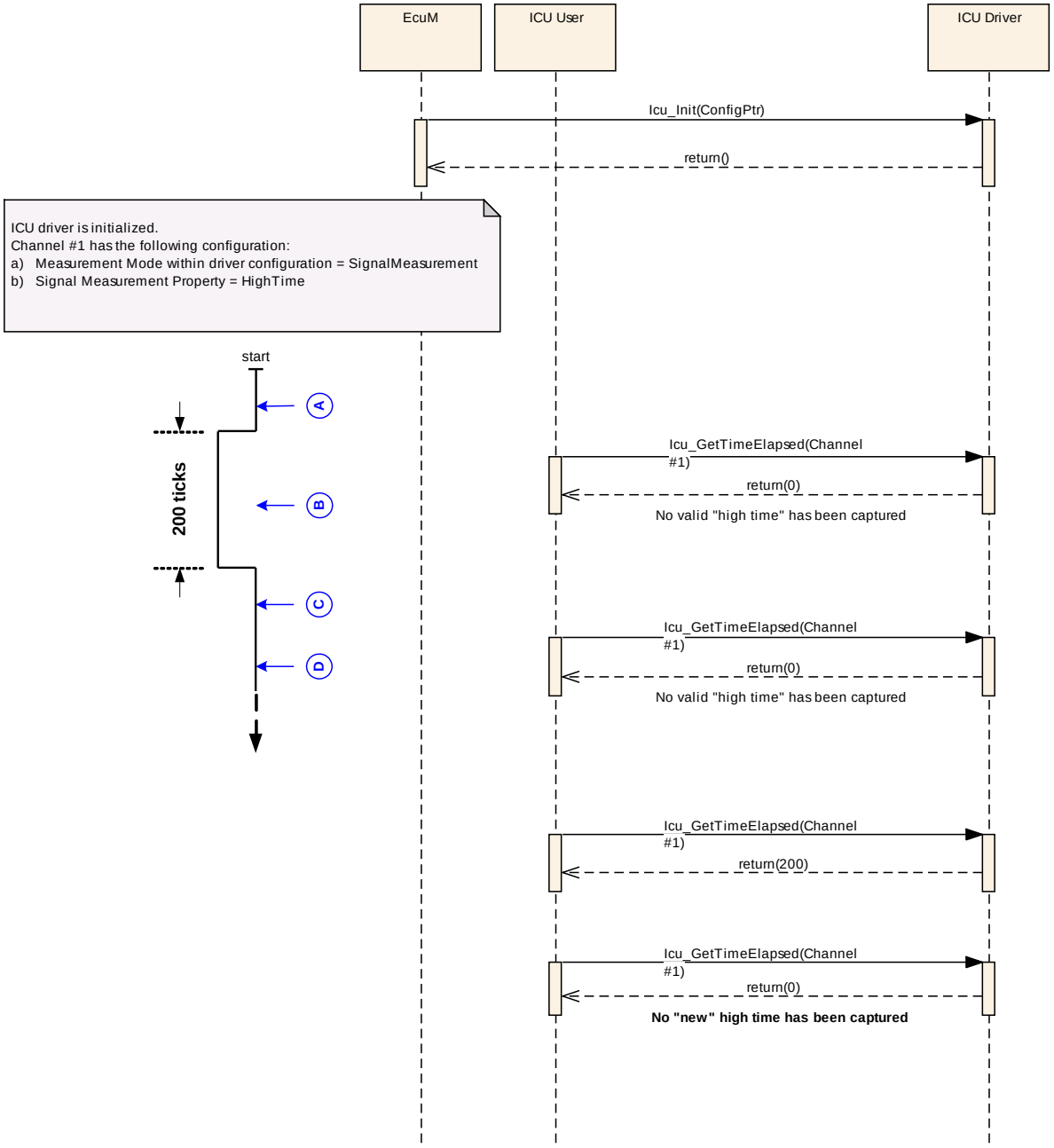


Figure 9.19, letter “A”).⌋ ()

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

the captured value shall be stored until a new value is captured) (similar to

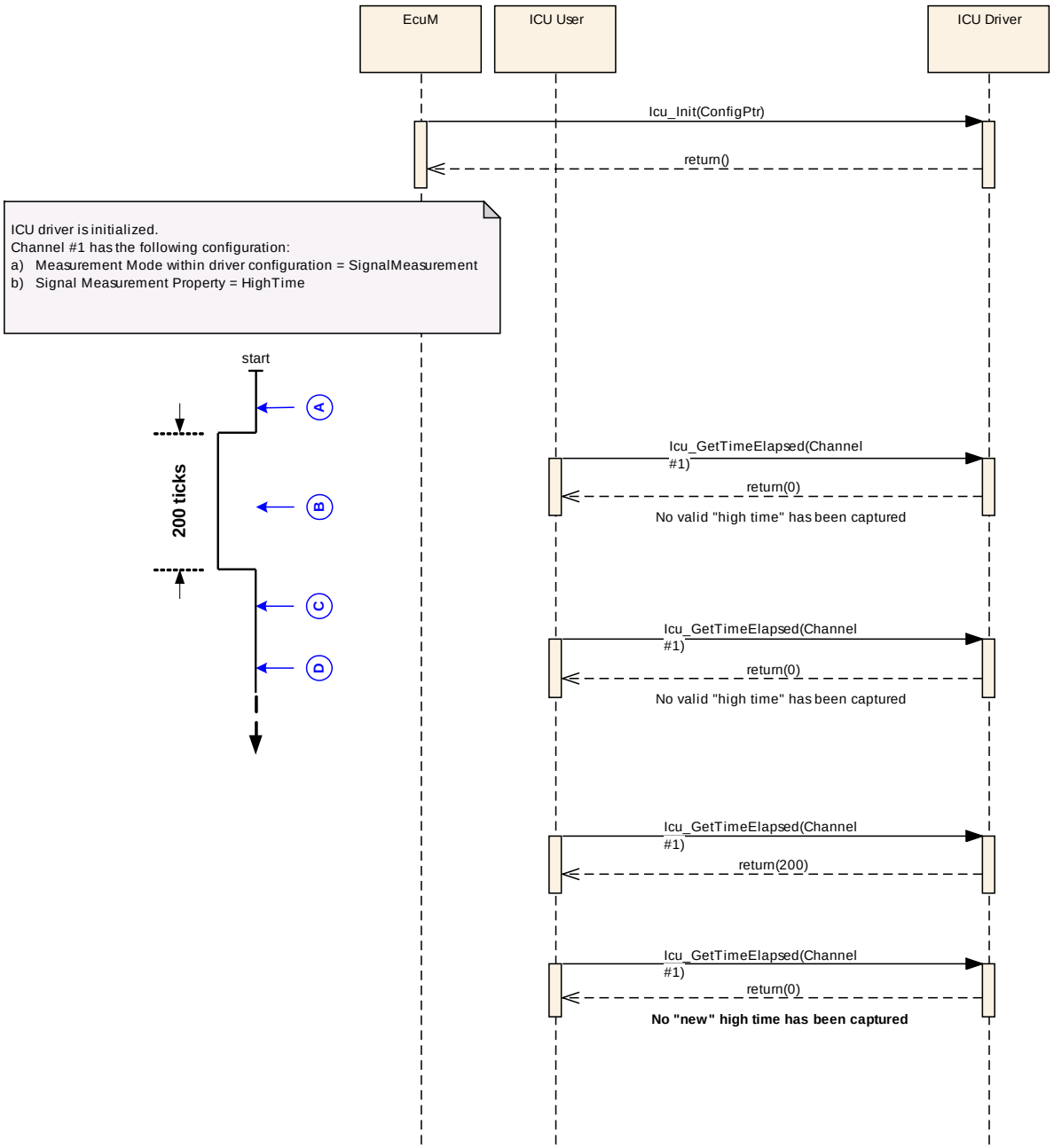


Figure 9.19, letter “B”).⌋ ()

[SWS_Icu_00344] ⌈The function `Icu_GetDutyCycleValues` shall return “0” in case captured duty cycle values were already returned once by this service and this

service is called again (similar to

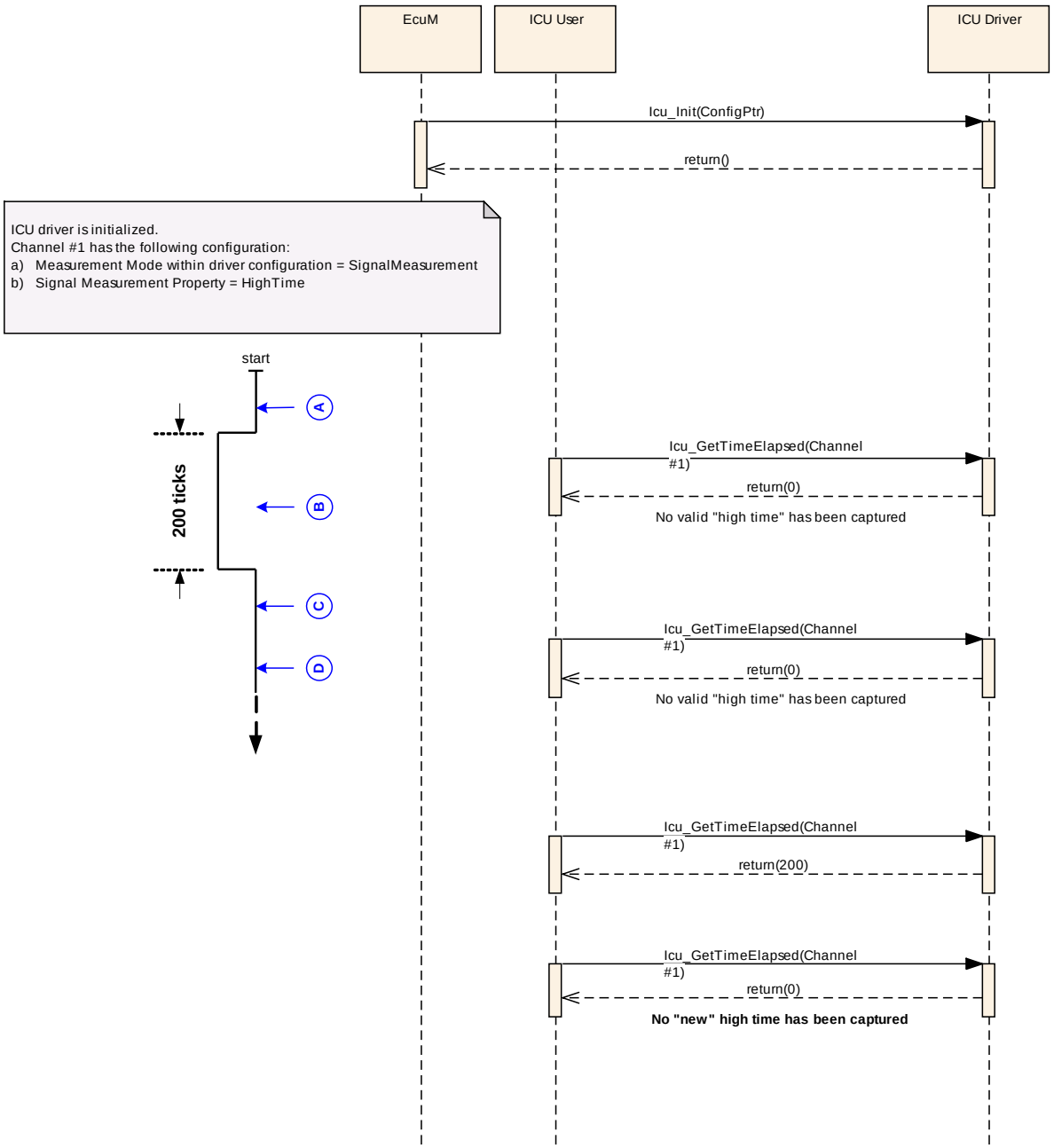


Figure 9.19, letter “D”)⌋ ()

[SWS_Icu_00106] ⌈The function `Icu_GetDutyCycleValues` shall be pre compile time configurable by the configuration parameter `IcuGetDutyCycleValuesApi`.⌋ (SRS_BSW_00410, SRS_BSW_00171)

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

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

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

SWS_Icu_00022 and SWS_Icu_00048 apply to the function `Icu_GetDutyCycleValues`.⌋ ()

8.3.24 Icu_GetVersionInfo

[SWS_Icu_00212] ⌈

Service name:	Icu_GetVersionInfo

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

Service ID[hex]:	0x12
Sync/Async:	Synchronous
Reentrancy:	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.

⌋ ()

[SWS_Icu_00346] ⌈`Icu_GetVersionInfo` operation is Non-Re-entrant.⌋ ()

[

[SWS_Icu_00356] ⌈If development error detection for the Icu module is enabled: The function `Icu_GetVersionInfo` shall check the parameter `versioninfo` for not being `NULL` and shall raise the development error code `ICU_E_PARAM_VINFO` if the check fails.⌋ ()

8.4 Callback notifications

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

8.5 Scheduled functions

None.

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

None.

8.6.2 Optional Interfaces

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

[SWS_Icu_00213] ⌈

API function	*Description*
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. OBD Events Suppression shall be ignored for this computation.
Det_ReportError	Service to report development errors.
EcuM_CheckWakeup	This callout is called by the EcuM to poll a wakeup source. It shall also be called by the ISR of a wakeup source to set up the PLL and check other wakeup sources that may be connected to the same interrupt.
EcuM_SetWakeupEvent	Sets the wakeup event.

⌋ ()

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

- **[SWS_Icu_00055]** ⌈The static configuration parameter `IcuReportWakeupSource` is set to "ON" ⌋ (SRS_SPAL_12069, SRS_BSW_00410)
- **[SWS_Icu_00056]** ⌈The module is in mode `ICU_MODE_SLEEP`⌋ (SRS_SPAL_12069)

- **[SWS_Icu_00057]** ⌈A wakeup event occurs on a wakeup capable ICU channel.⌋ (SRS_SPAL_12069)

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

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

8.6.3 Configurable interfaces

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

[SWS_Icu_00119] ⌈The ISRs shall reset the interrupt flags (if needed by hardware) and call the corresponding notification functions.⌋ (SRS_SPAL_12129)

[SWS_Icu_00018] ⌈The Icu notification functions shall be configurable as function pointers within the initialization data structure (`Icu_ConfigType`).⌋ (SRS_SPAL_12056)

[SWS_Icu_00187] ⌈The Icu module's notification functions shall have no parameters and no return value.⌋ (SRS_BSW_00359)

[SWS_Icu_00214] ⌈

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

⌋ ()

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

[SWS_Icu_00021] ⌈According to the last call of `Icu_EnableNotification()`, the Icu module shall call the notification function Icu_SignalNotification_<Channel> if the requested signal edge (rising / falling / both edges) occurs (once per edge).⌋ (SRS_SPAL_00157, SRS_Icu_12369)

[SWS_Icu_00044] ⌈Only those edge notifications shall be provided, which are supported by hardware.⌋ (SRS_Icu_12305)

[SWS_Icu_00042] ⌈After a call of Icu_DisableNotification , the Icu module shall not call the notification function Icu_SignalNotification_<Channel>.⌋ (SRS_Icu_12305)

[SWS_Icu_00215] ⌈

Service name:	Icu_TimestampNotification_<Channel>

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

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

⌋ (SRS_Icu_12444)

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

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

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

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

9 Sequence diagrams

9.1 Icu_Init

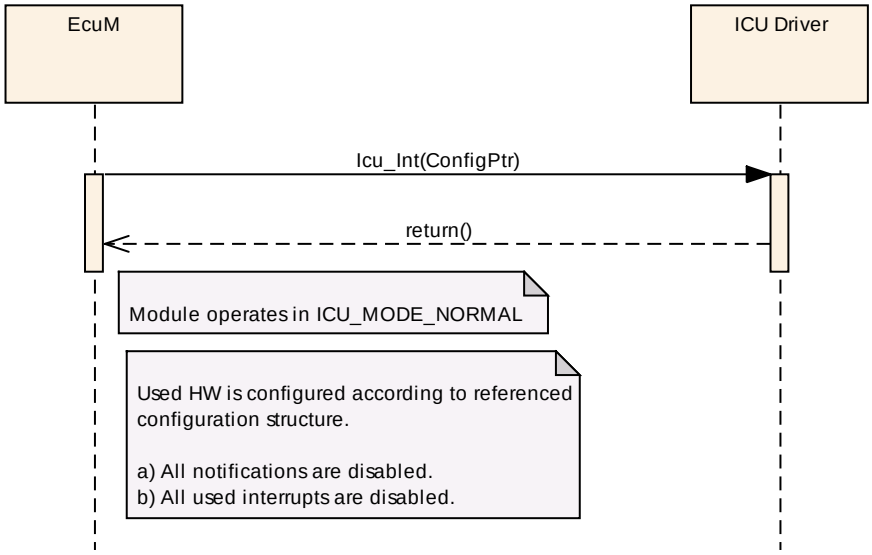


Figure 9.1: Initialization of the ICU driver

9.2 Icu_DeInit

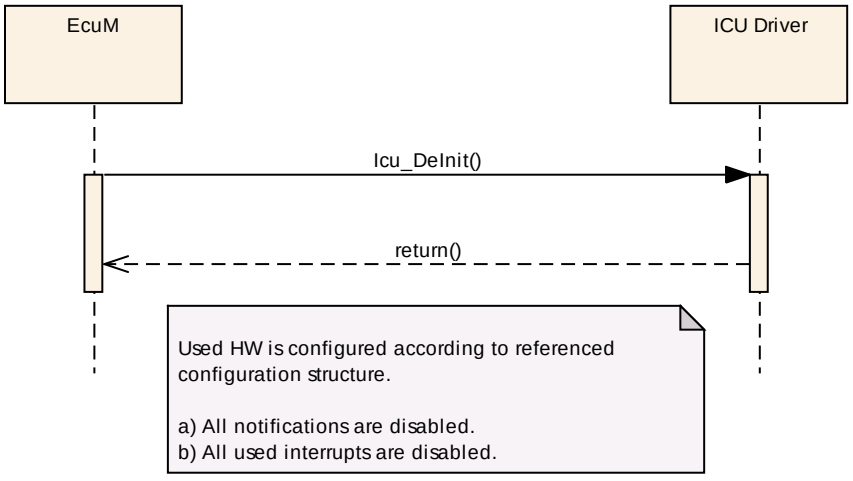


Figure 9.2: De-Initialization of the ICU driver

9.3 Check Wakeup Events

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

9.4 Icu_SetMode

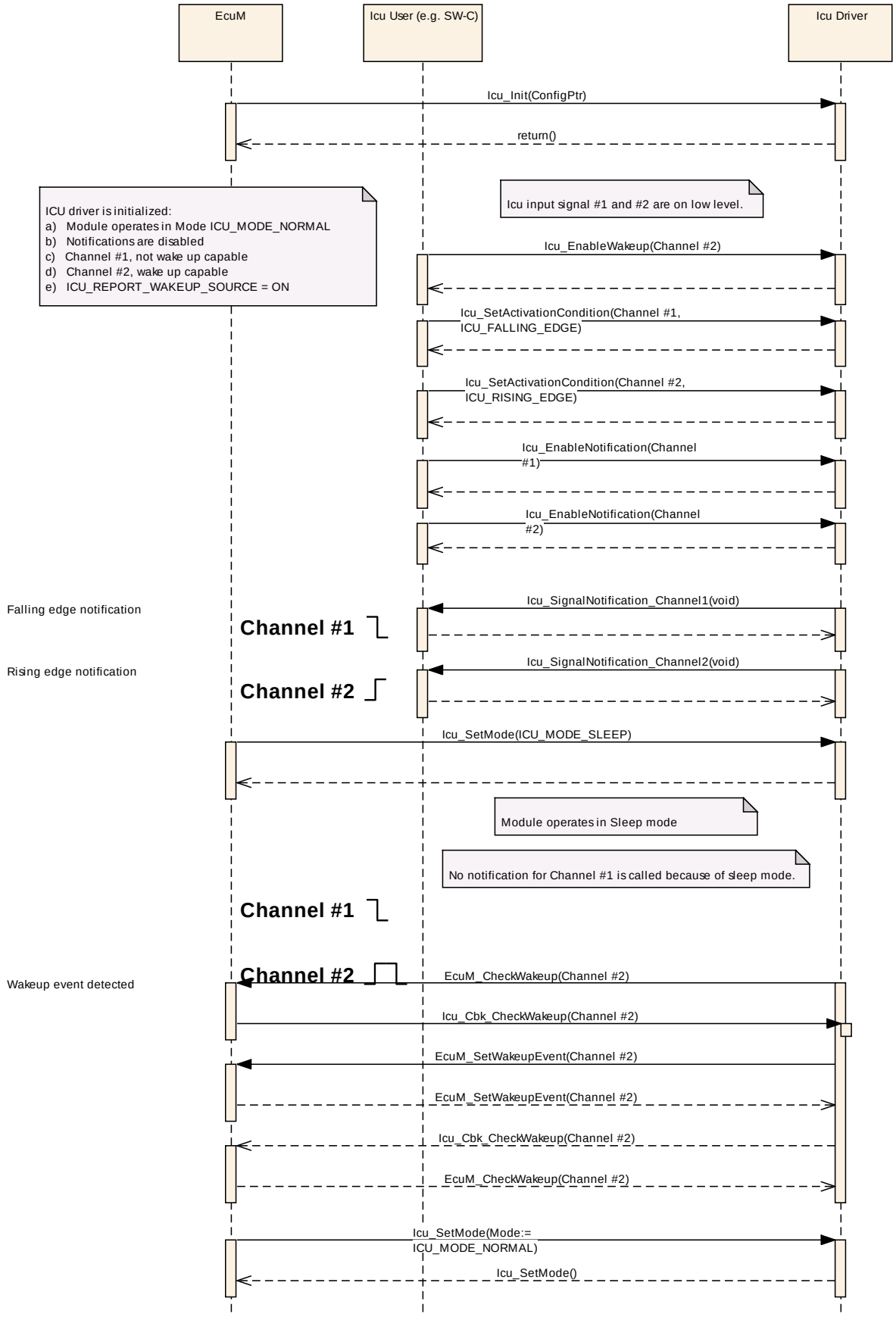


Figure 9.3: Enabled notifications in SLEEP mode

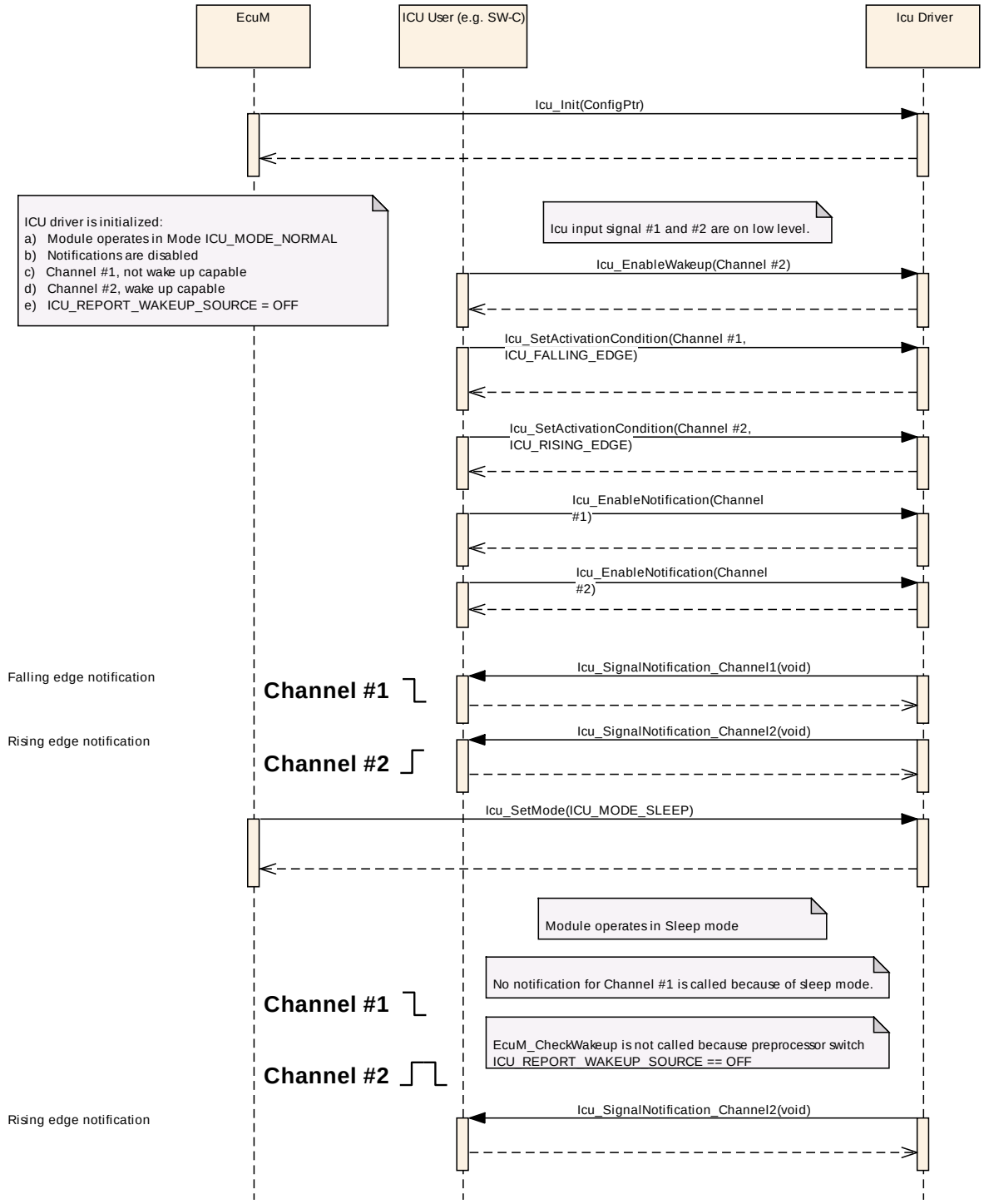


Figure 9.4: Disabled reporting of wakeup sources in SLEEP mode

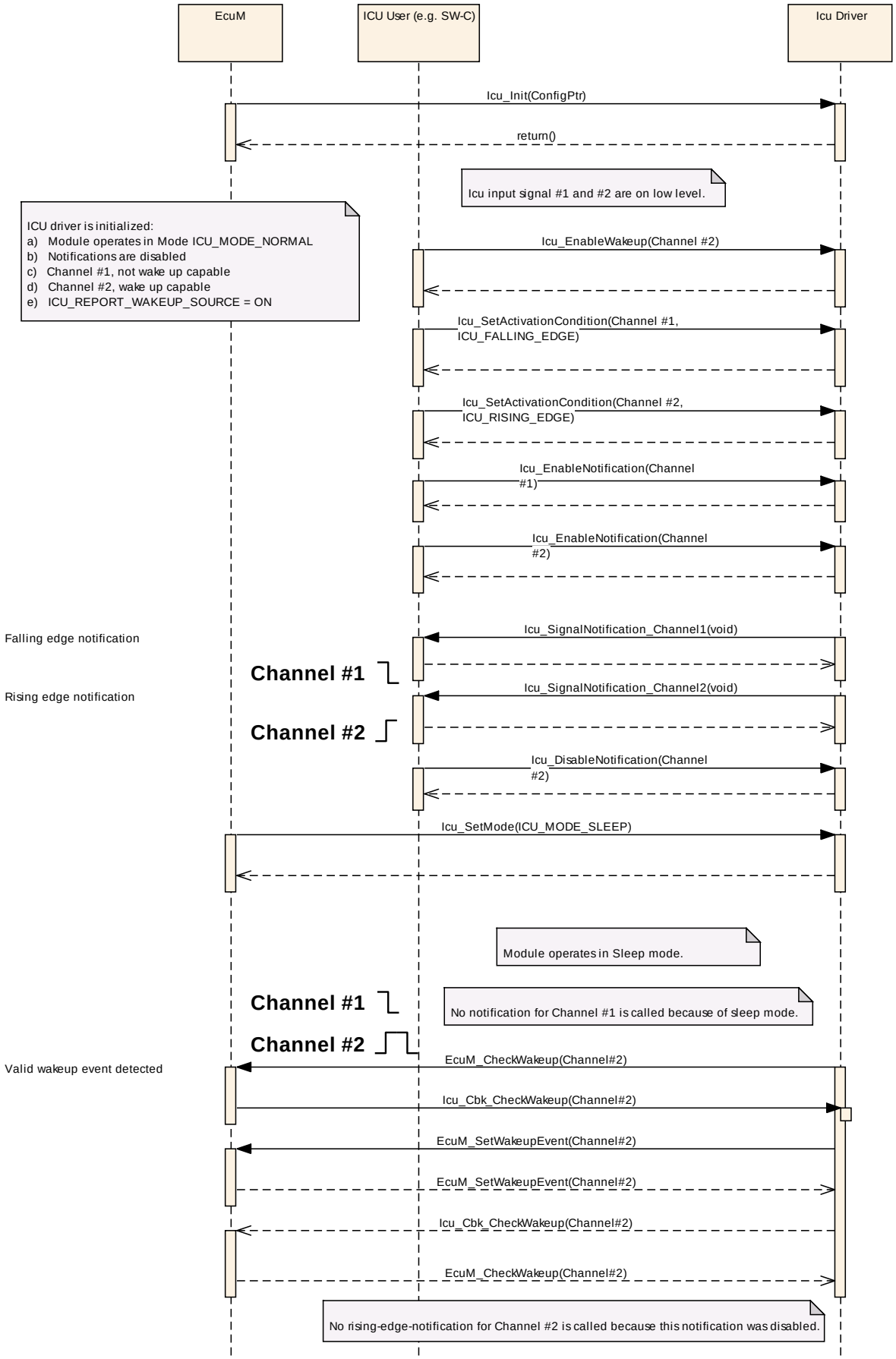


Figure 9.5: Disabled edge notification in SLEEP mode

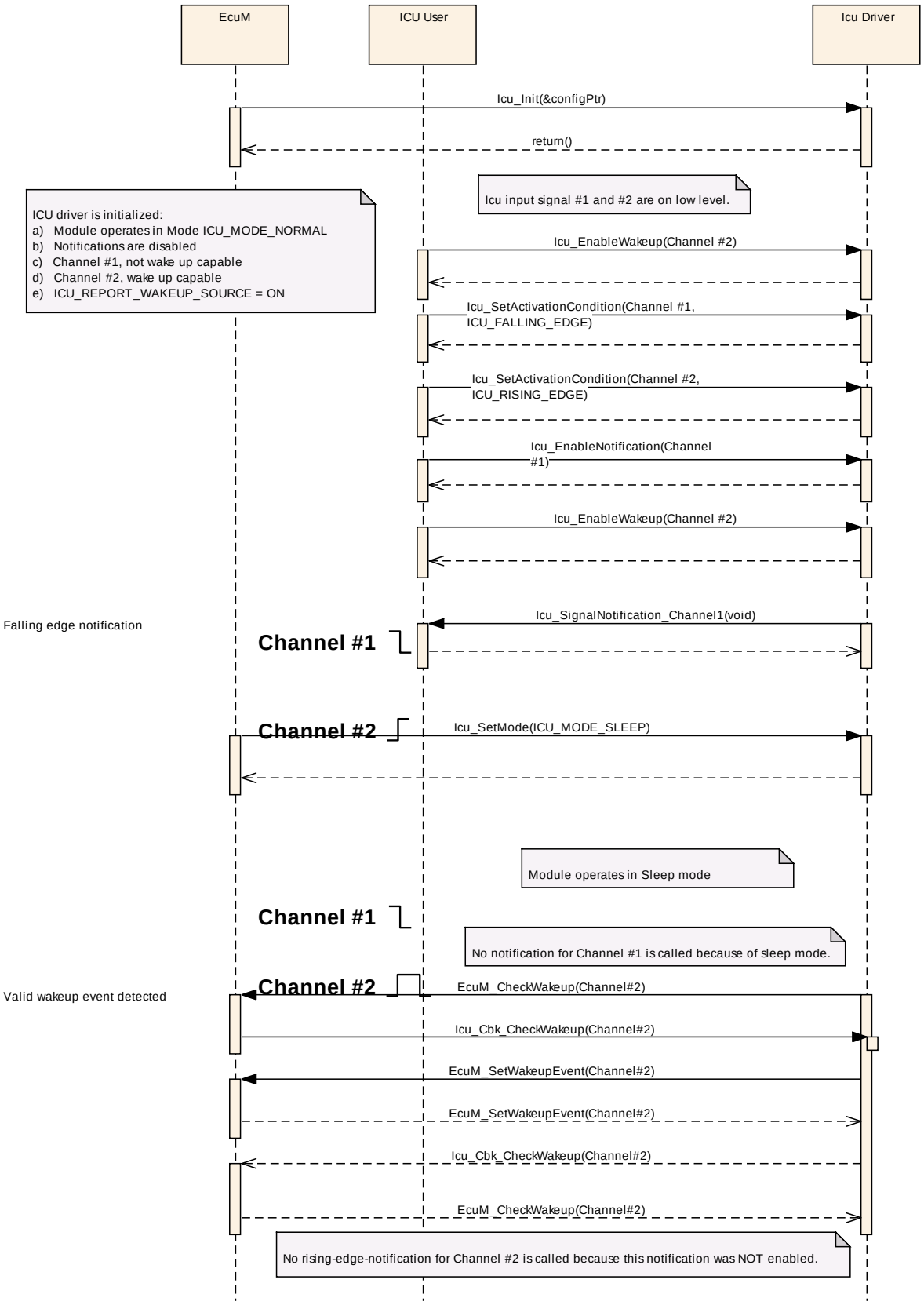


Figure 9.6: Un-Enabled reporting of notifications in SLEEP mode

9.5 Icu_DisableWakeup

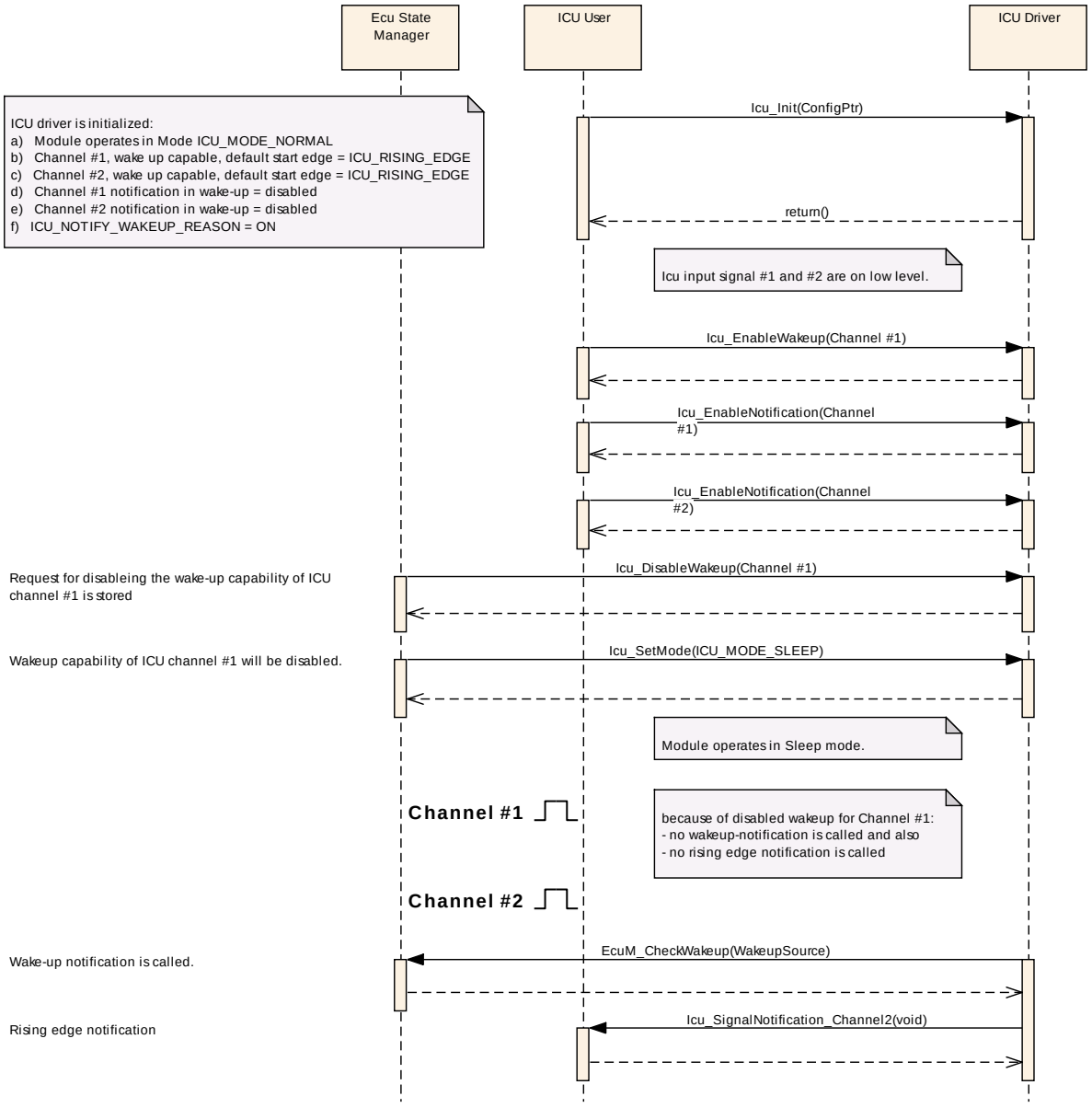


Figure 9.7: Disabling of wakeup-capabilities

9.6 Icu_EnableWakeup

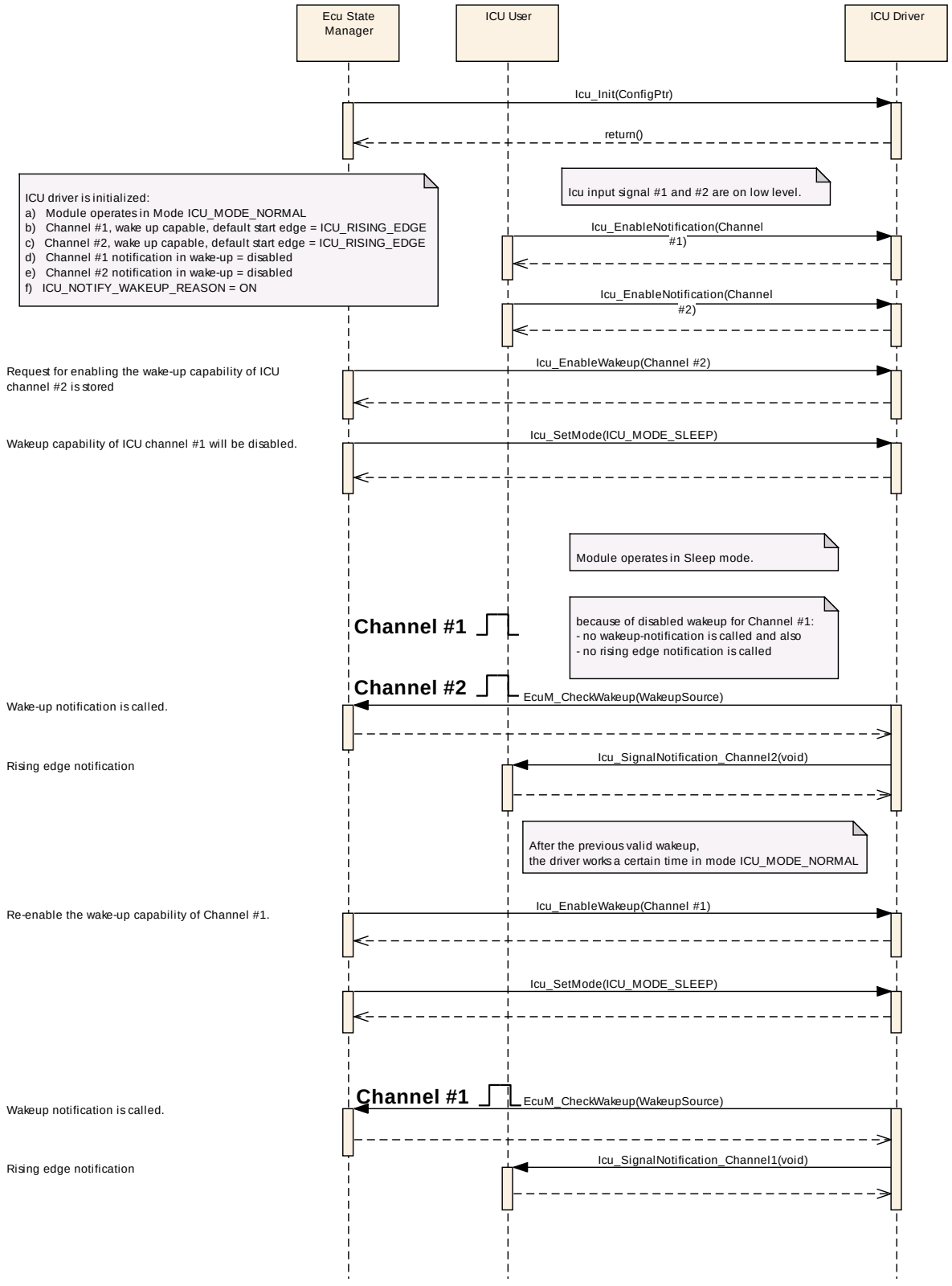


Figure 9.8: Enabling of wakeup-capabilities

9.7 Icu_SetActivationCondition

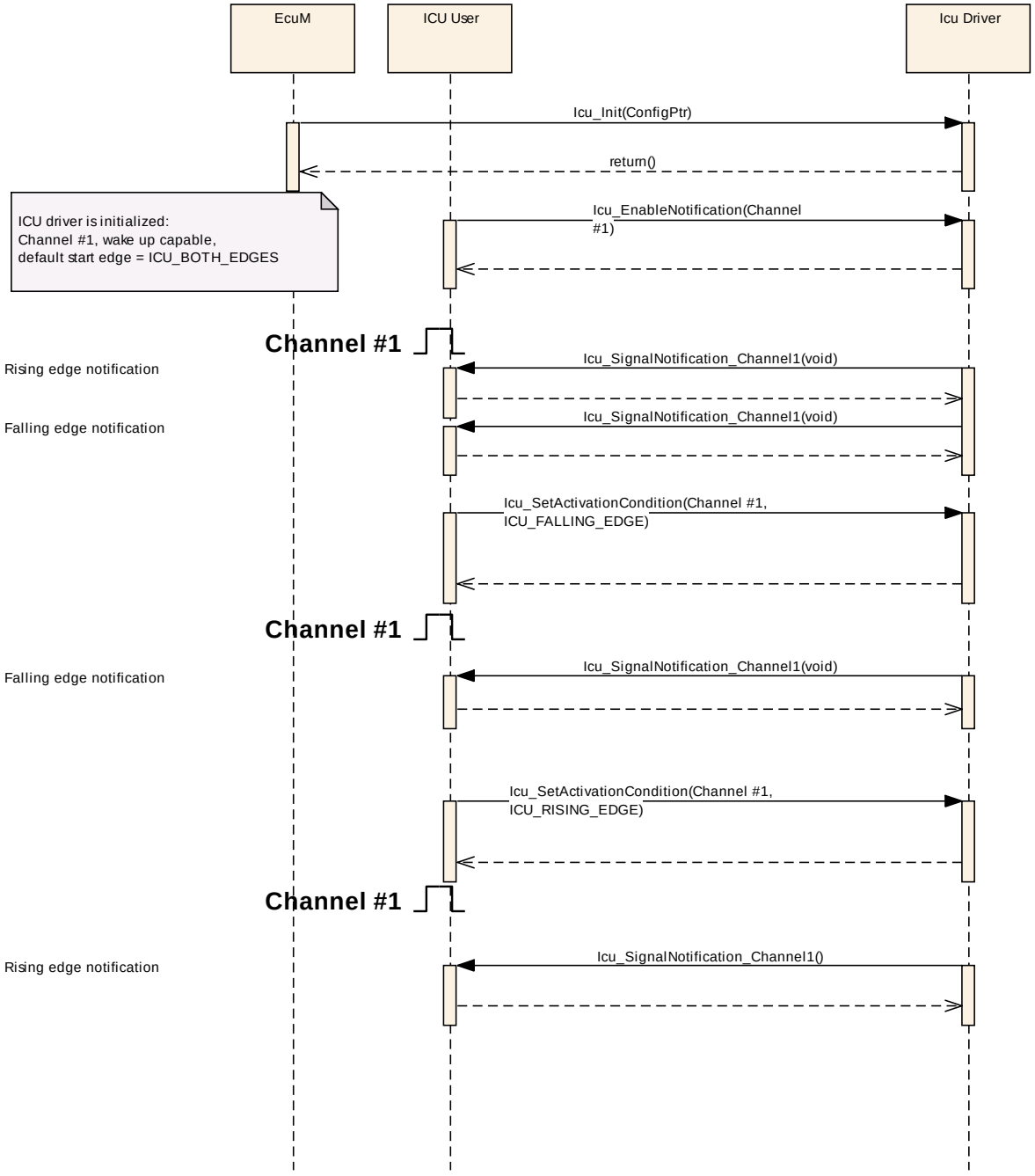


Figure 9.9: Setting up the activation condition for a channel

9.8 Icu_DisableNotification

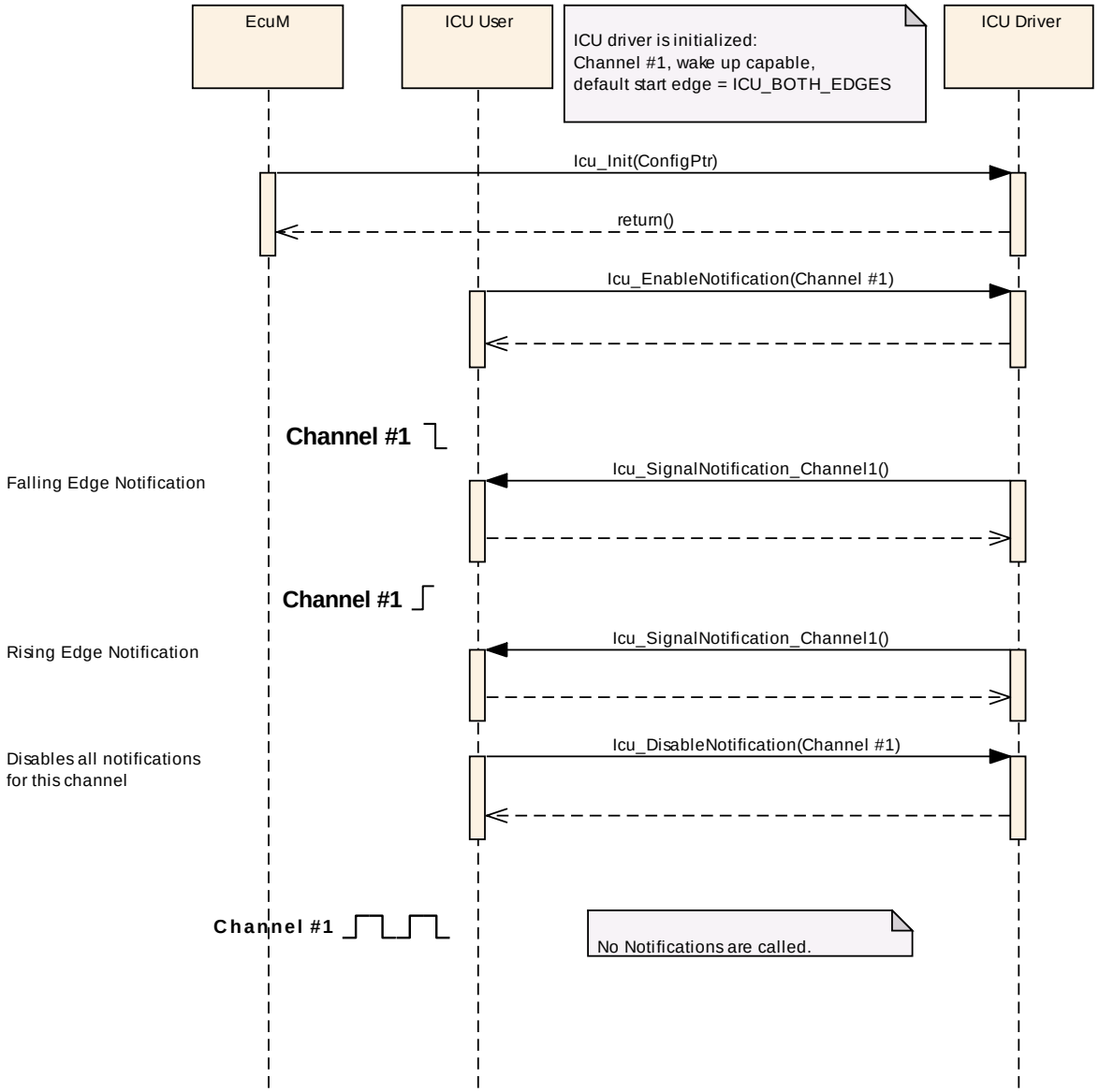


Figure 9.10: Disabling of the notification for a channel

9.9 Icu_EnableNotification

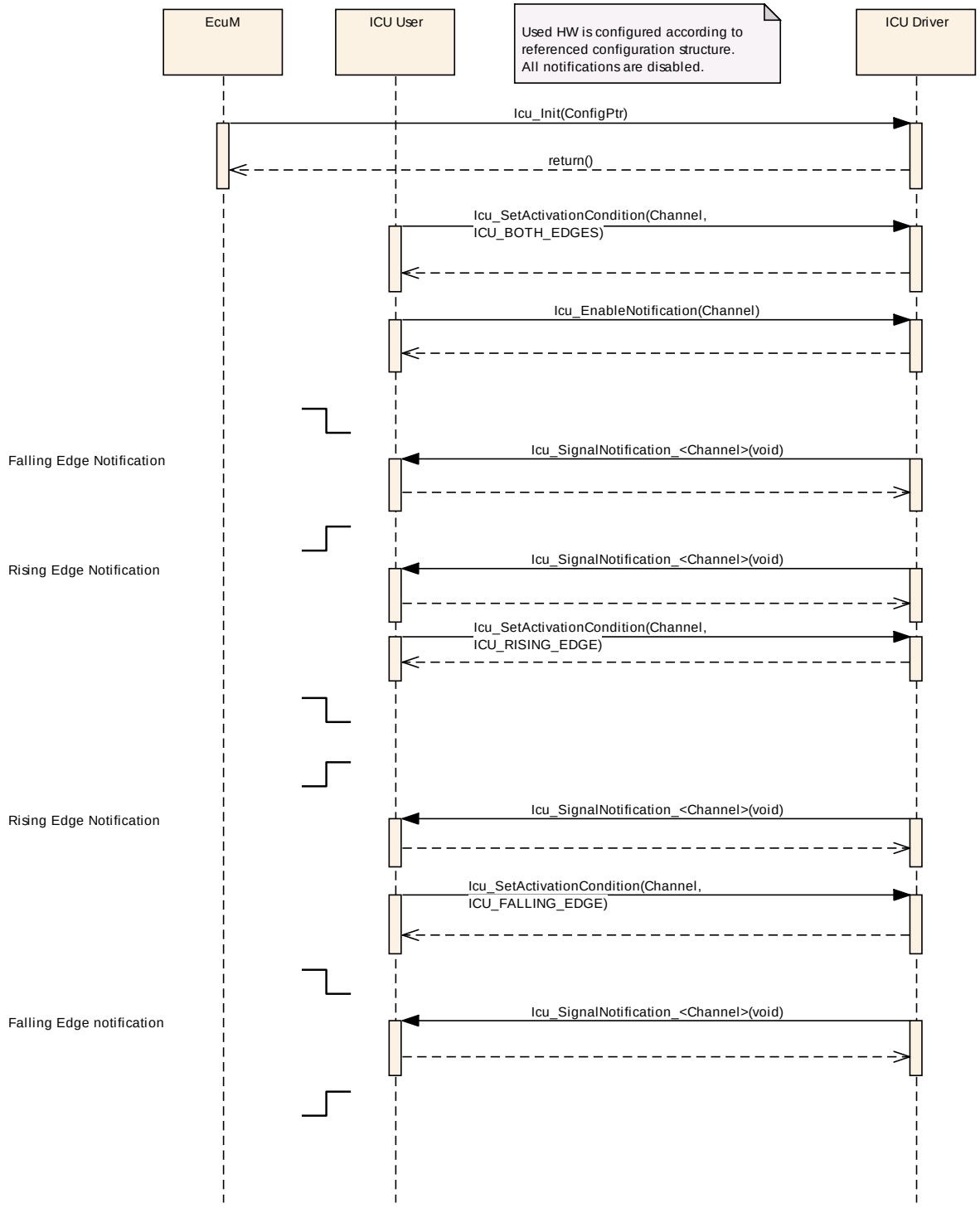


Figure 9.11: Enabling of the edge-notification for a channel

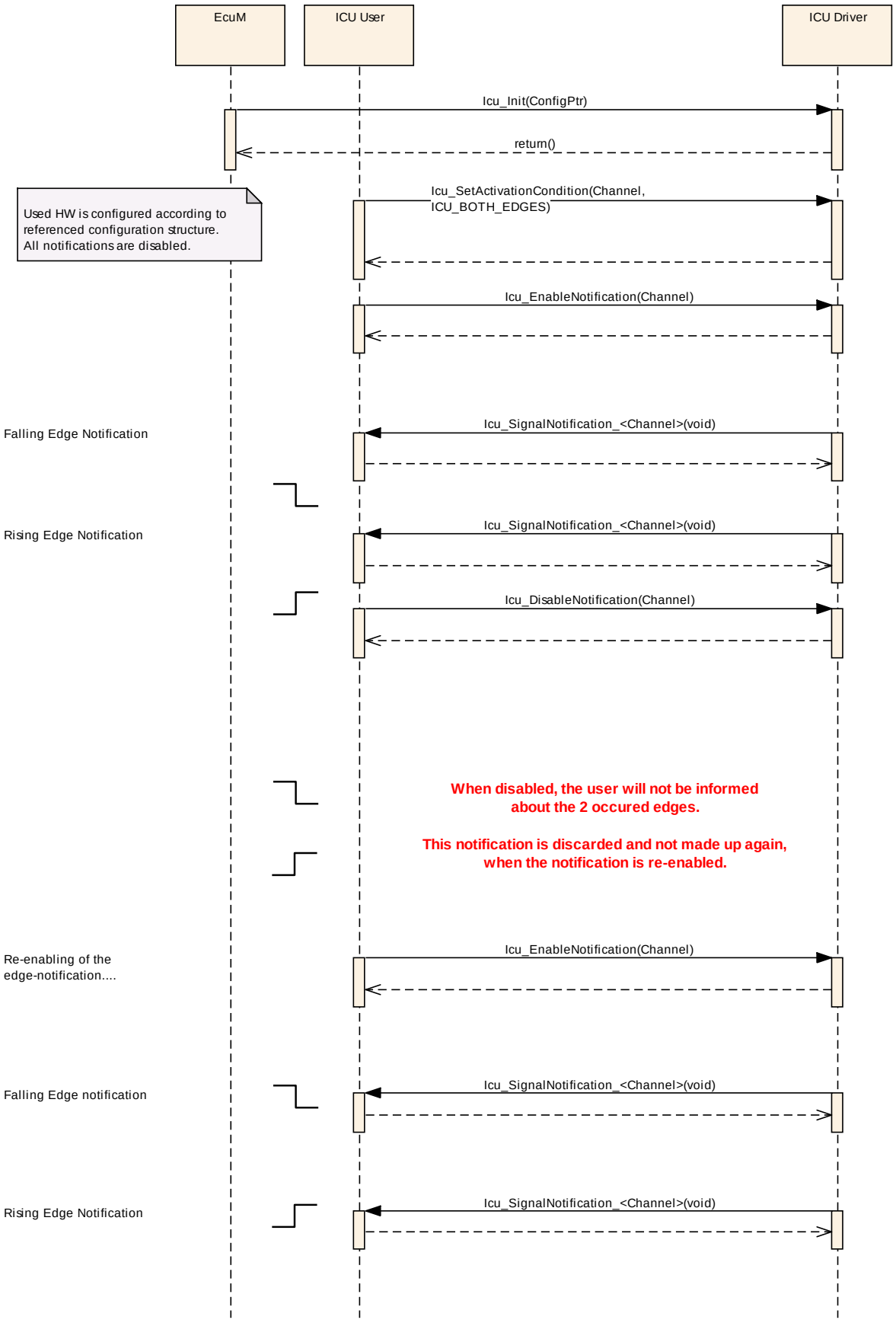

EcuM
ICU User
ICU Driver
Icu_Init(ConfigPtr)
return()
Used HW is configured according to referenced configuration structure. All notifications are disabled.
Icu_SetActivationCondition(Channel, ICU_BOTH_EDGES)
Icu_EnableNotification(Channel)
Falling Edge Notification
Icu_SignalNotification_<Channel>(void)
Rising Edge Notification
Icu_SignalNotification_<Channel>(void)
Icu_DisableNotification(Channel)
When disabled, the user will not be informed about the 2 occured edges.
This notification is discarded and not made up again, when the notification is re-enabled.
Re-enabling of the edge-notification....
Icu_EnableNotification(Channel)
Falling Edge notification
Icu_SignalNotification_<Channel>(void)
Rising Edge Notification
Icu_SignalNotification_<Channel>(void)

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

9.10 Icu_GetInputState

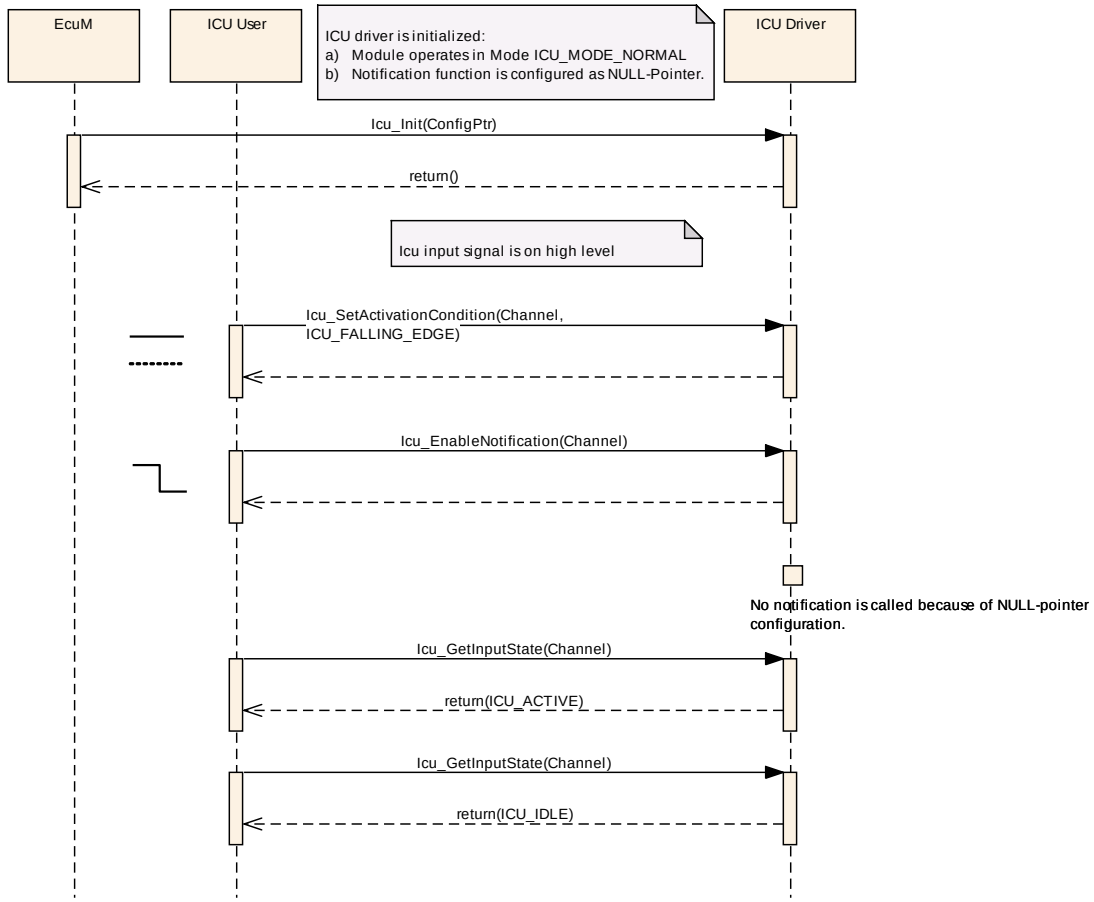


Figure 9.13: Polling of the channel status

9.11 Icu Timestamping

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

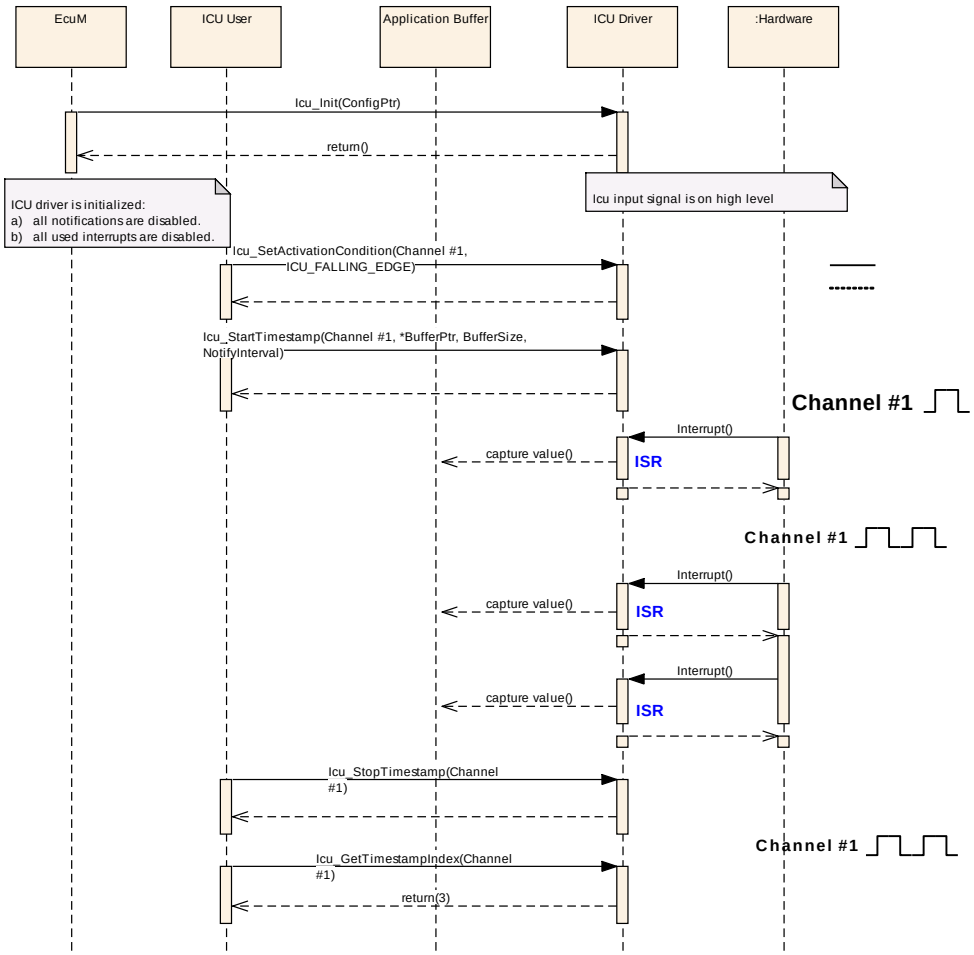


Figure 9.14: Overview of the timestamping functionality of the ICU driver

The Timestamping in general is shown in the following figure:

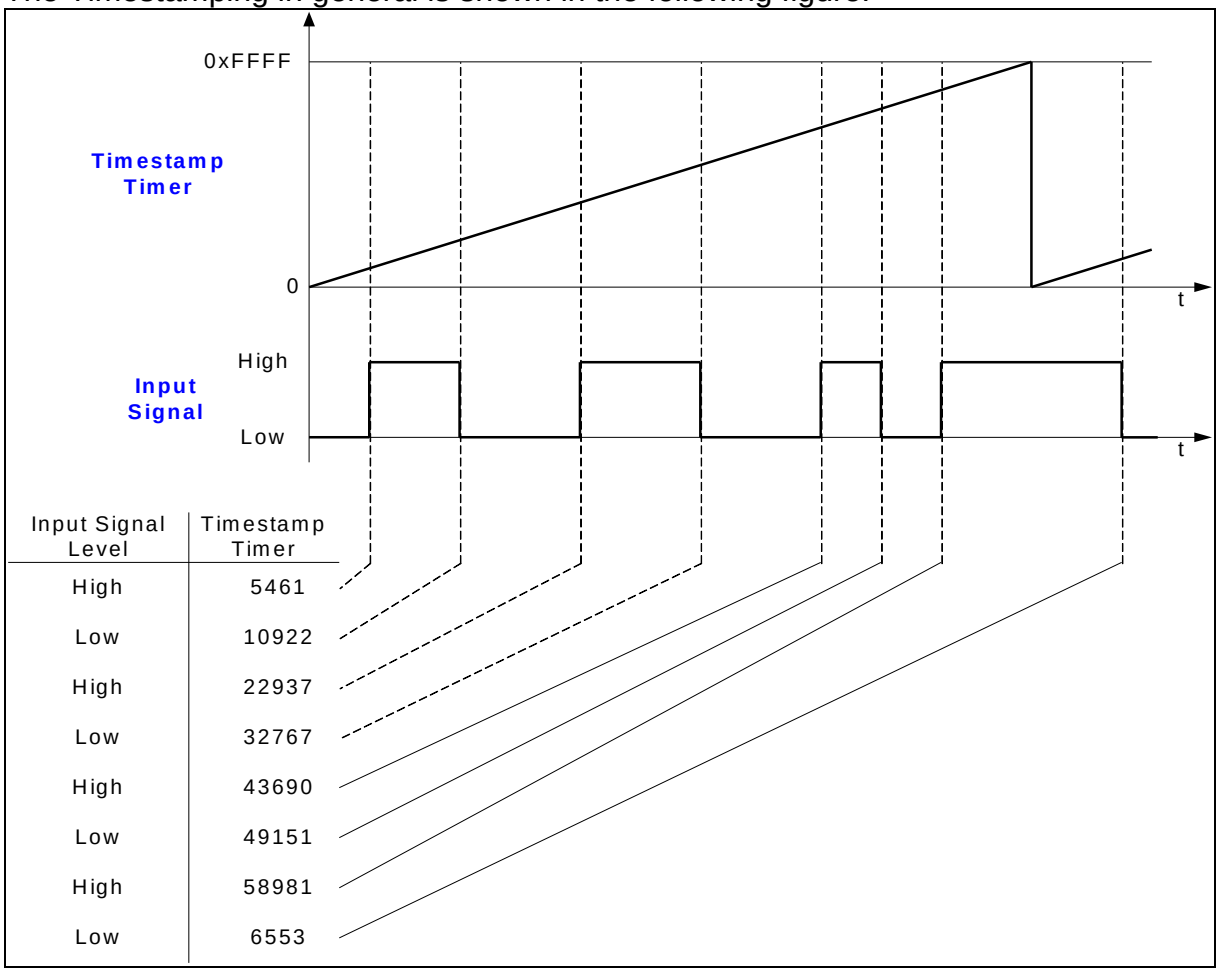


Figure 9.15: Timestamping overview

9.12 Icu Edge Counting

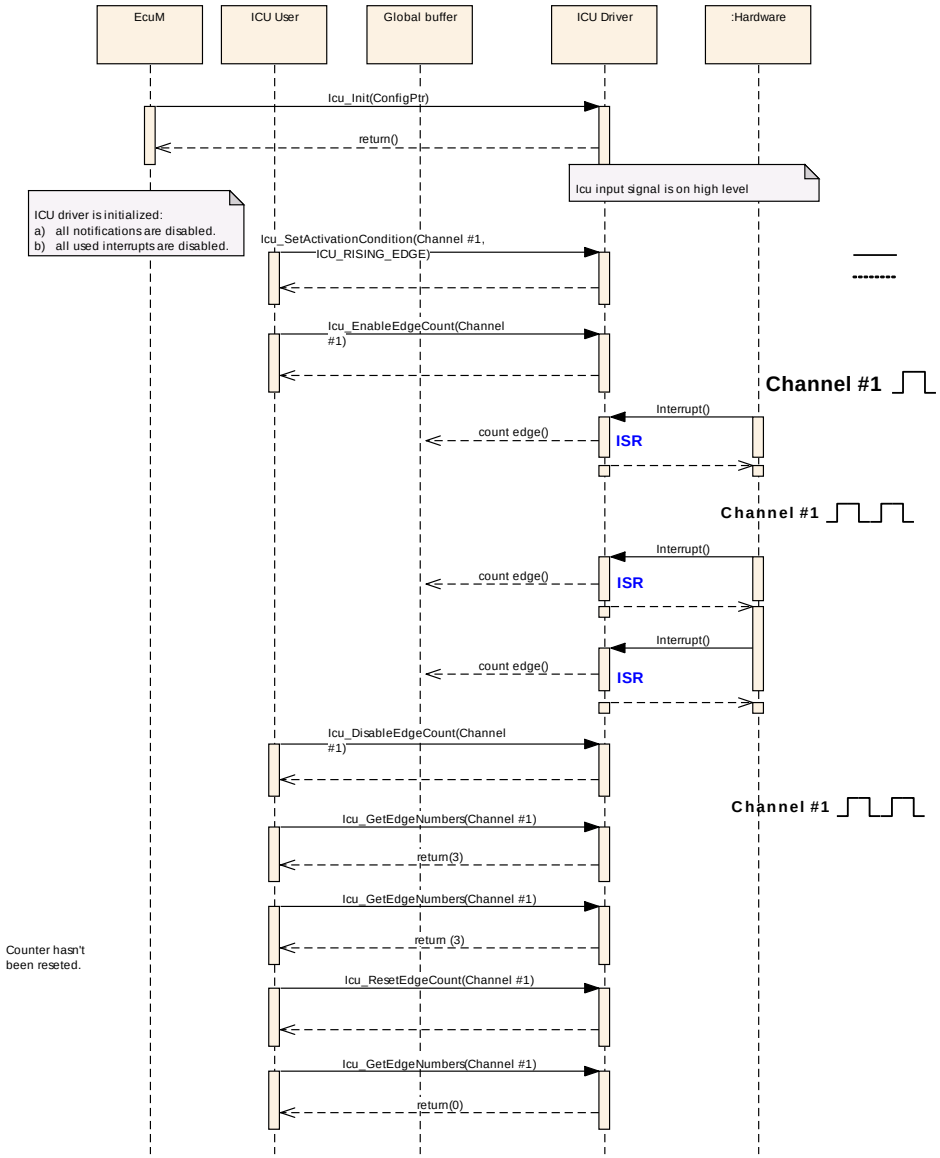


Figure 9.16: Inquire the number of counted edges

9.13 Icu_GetTimeElapsed

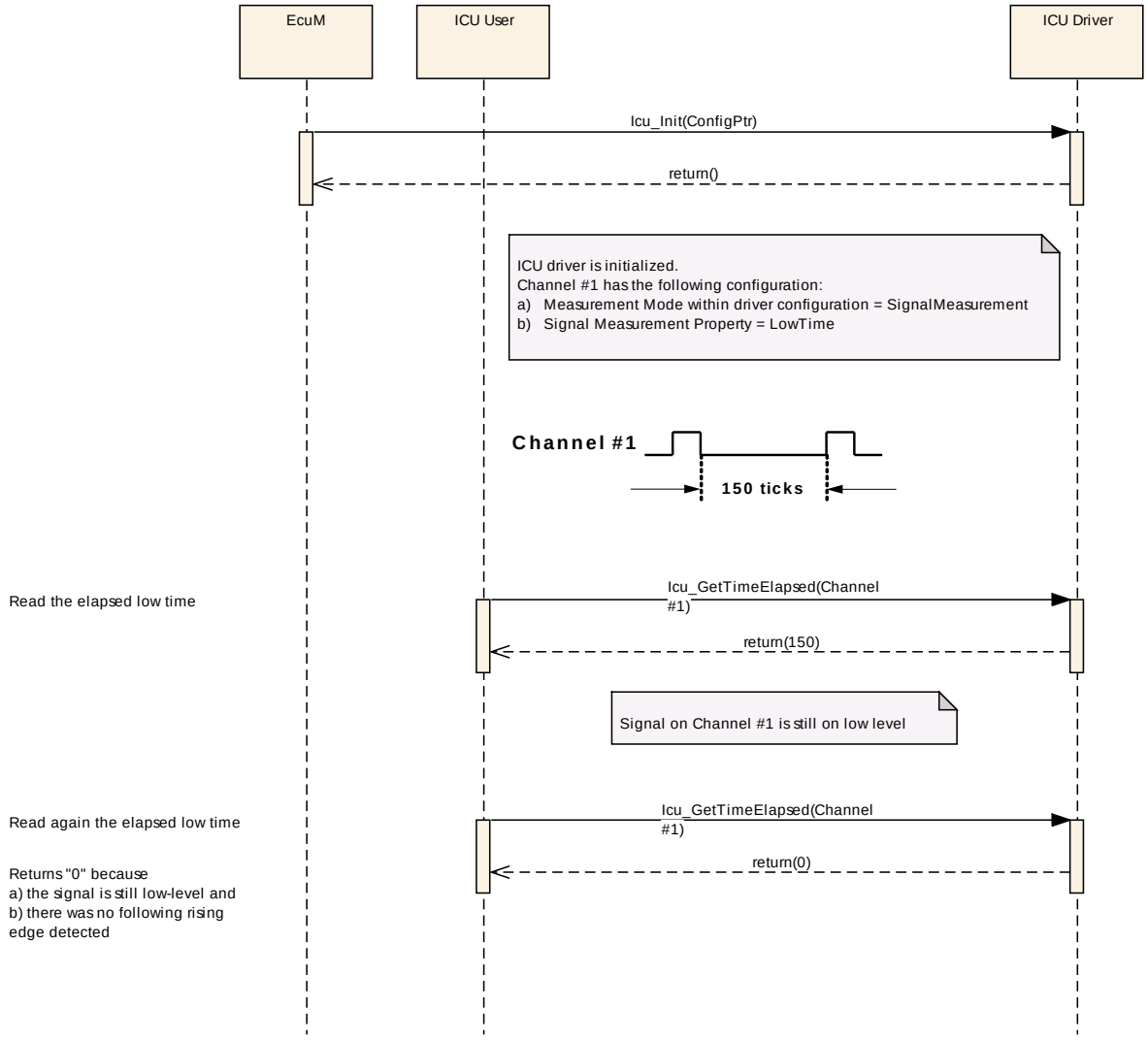


Figure 9.17: Inquire the elapsed level-time of a channel

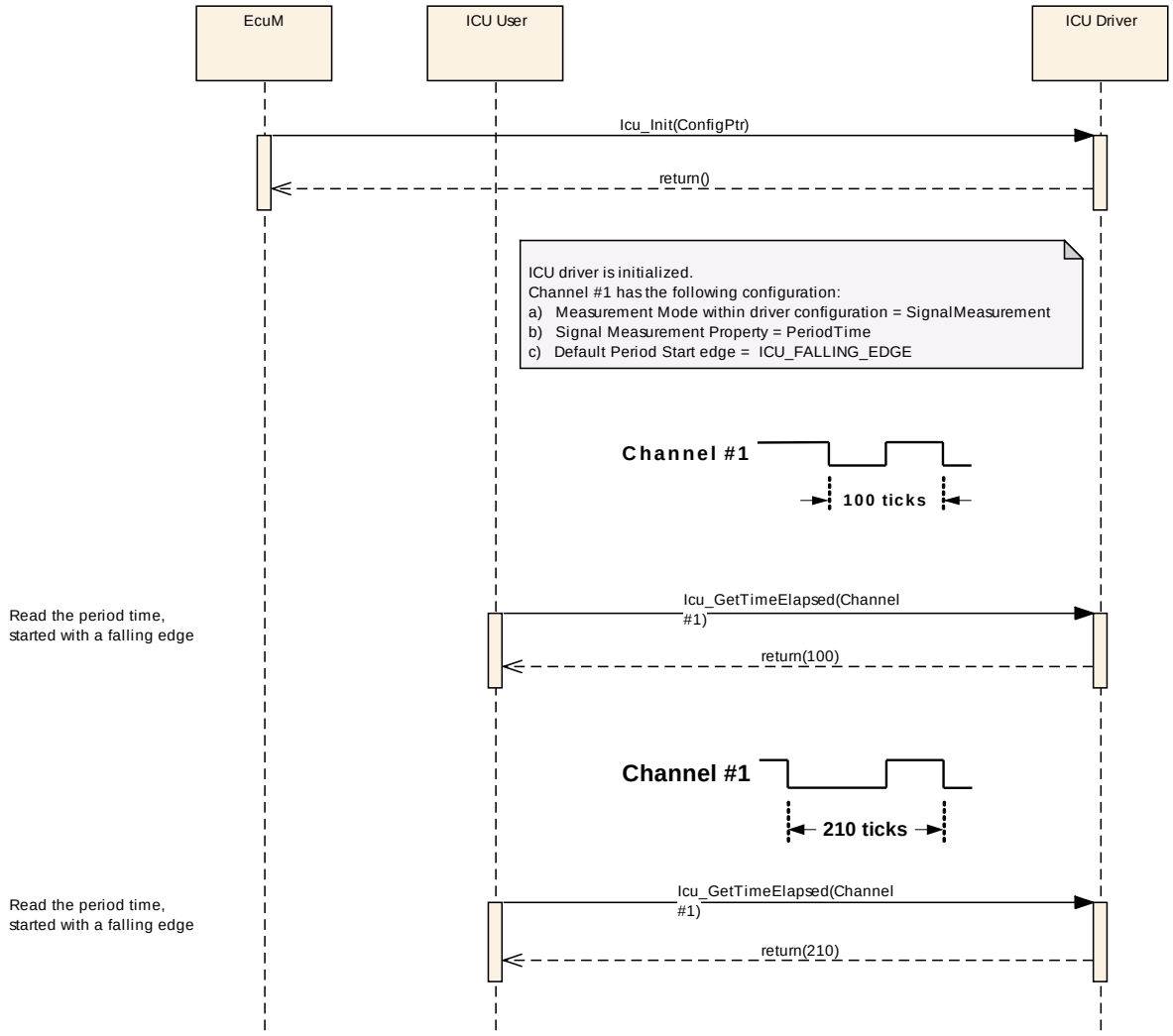


Figure 9.18: Inquire the elapsed period time of a channel

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

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

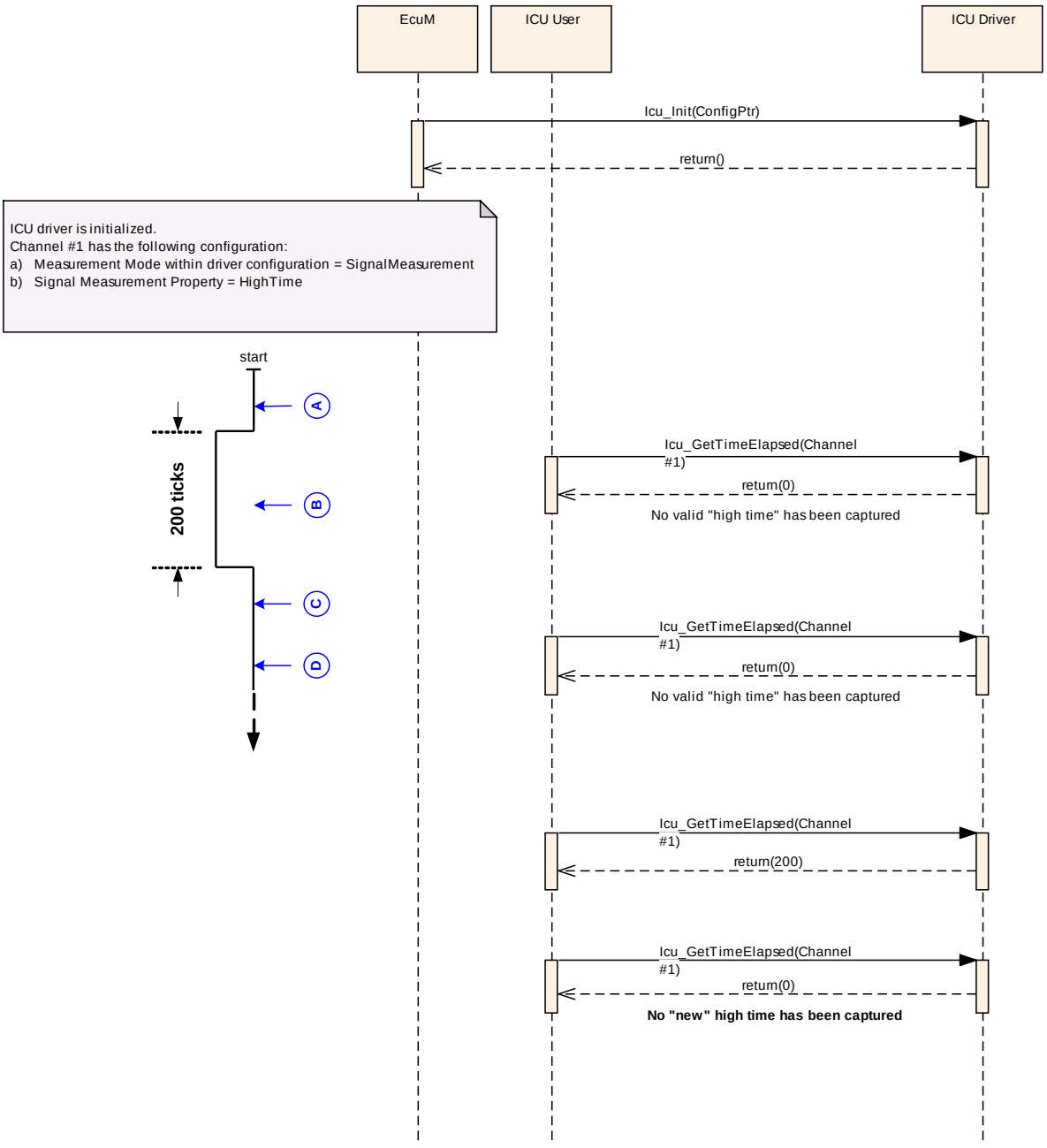


Figure 9.19: Inquire the elapsed high time of a channel

9.14 Icu_GetDutyCycleValues

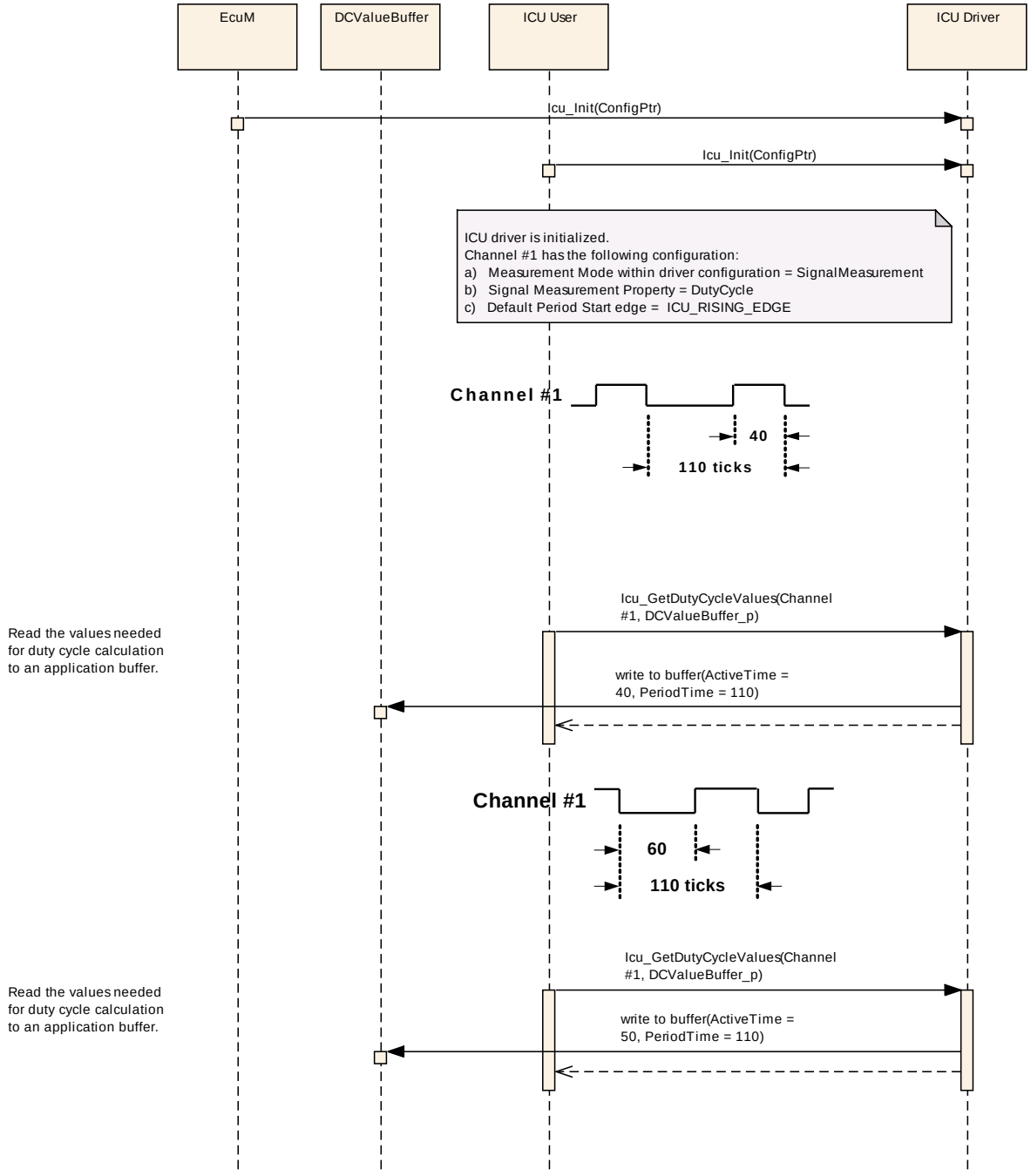


Figure 9.20: Measure the values needed for calculation of duty cycles

9.15 Icu_SignalNotification and Icu_GetInputState

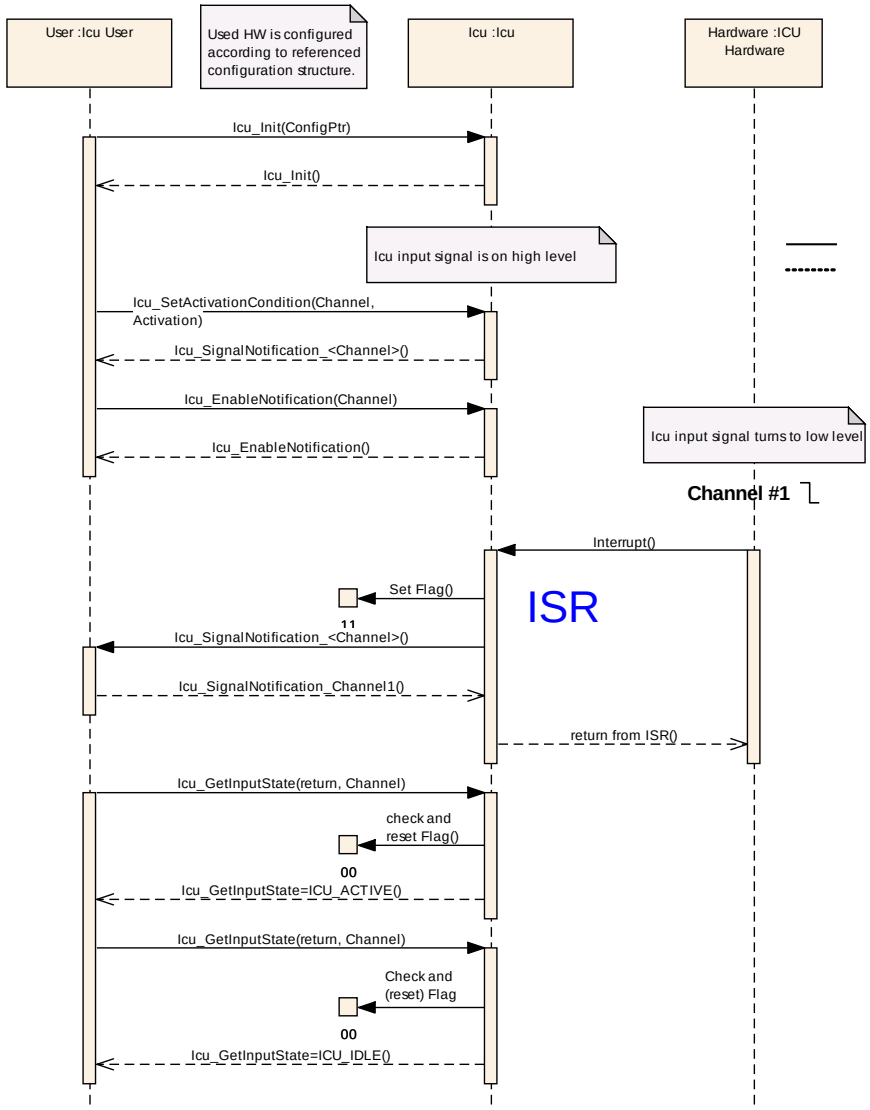


Figure 9.21: Cooperative usage of notification and polling mechanism

10 Configuration specification

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

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

Chapter 10.3 specifies published information of the module ICU.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS_BSWGeneral.*

10.2 Containers and configuration parameters

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

10.2.1 Variants

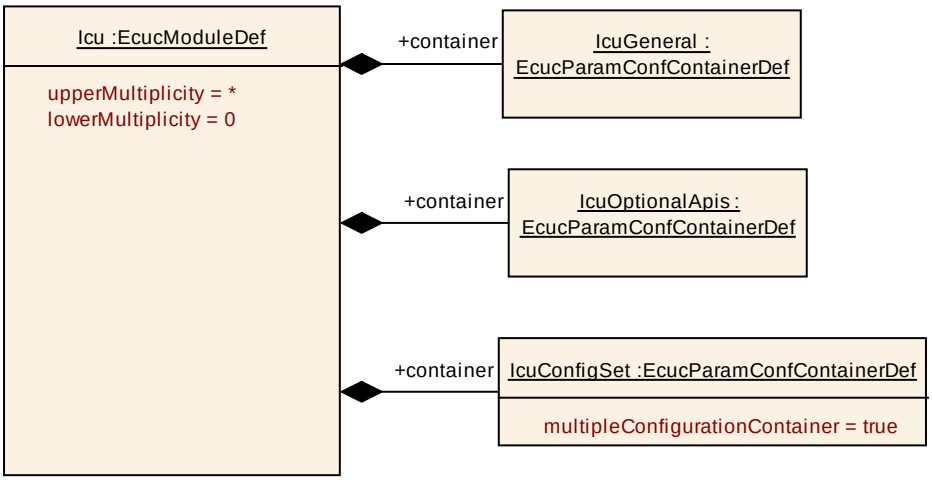
[SWS_Icu_00188] ⌈VARIANT-PRE-COMPILE (**P**re **C**ompile): The module ICU shall support a configuration variant pre-compile required for pre-compile time parameters⌋ ()

[SWS_Icu_00189] ⌈VARIANT-POST-BUILD (**P**ost **B**uild): The module ICU shall support a configuration variant post-build. This variant allows a mix of pre-compile time- and post build time-configuration parameters (multiple-selectable configurable configuration parameter sets).y⌋ ()

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

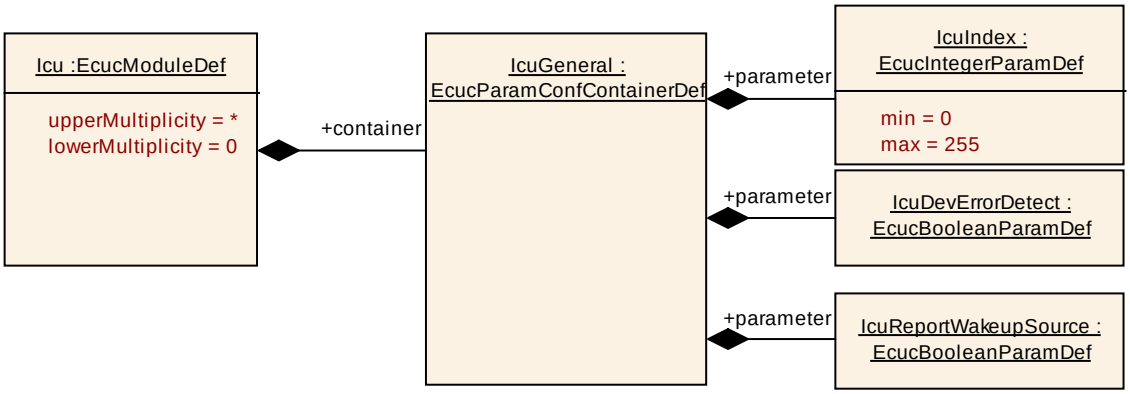
SWS Item	**ECUC_Icu_00026 :**
Container Name	IcuGeneral{General Configuration}
Description	Configuration of general ICU parameters.
Configuration Parameters	

SWS Item	**ECUC_Icu_00232 :**		
Name	IcuDevErrorDetect {ICU_DEV_ERROR_DETECT}		
Description	Switches the Development Error Detection and Notification on or off. true: Enabled. false: Disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Icu_00221 :**		
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	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Icu_00233 :**		
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers



10.2.4 IcuOptionalApis

SWS Item	**ECUC_Icu_00114 :**		
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	**ECUC_Icu_00234 :**		
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Icu_00235 :**		
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Icu_00124 :**
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Icu_00356 :**		
Name	IcuEdgeDetectApi {ICU_EDGE_DETECT_API}		
Description	Adds / removes the services related to the edge detection functionality, from the code: Icu_EnableEdgeDetection() and Icu_DisableEdgeDetection(). true: These services can be used. false: These services can not be used.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Icu_00236 :**		
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Icu_00237 :**		
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: If IcuSignalMeasurementApi==false this switch shall also be set to false.		

SWS Item	**ECUC_Icu_00238 :**		
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Icu_00239 :**		
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: If IcuSignalMeasurementApi==false this switch shall also be set to false.		

SWS Item	**ECUC_Icu_00240 :**		
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Icu_00241 :**		
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

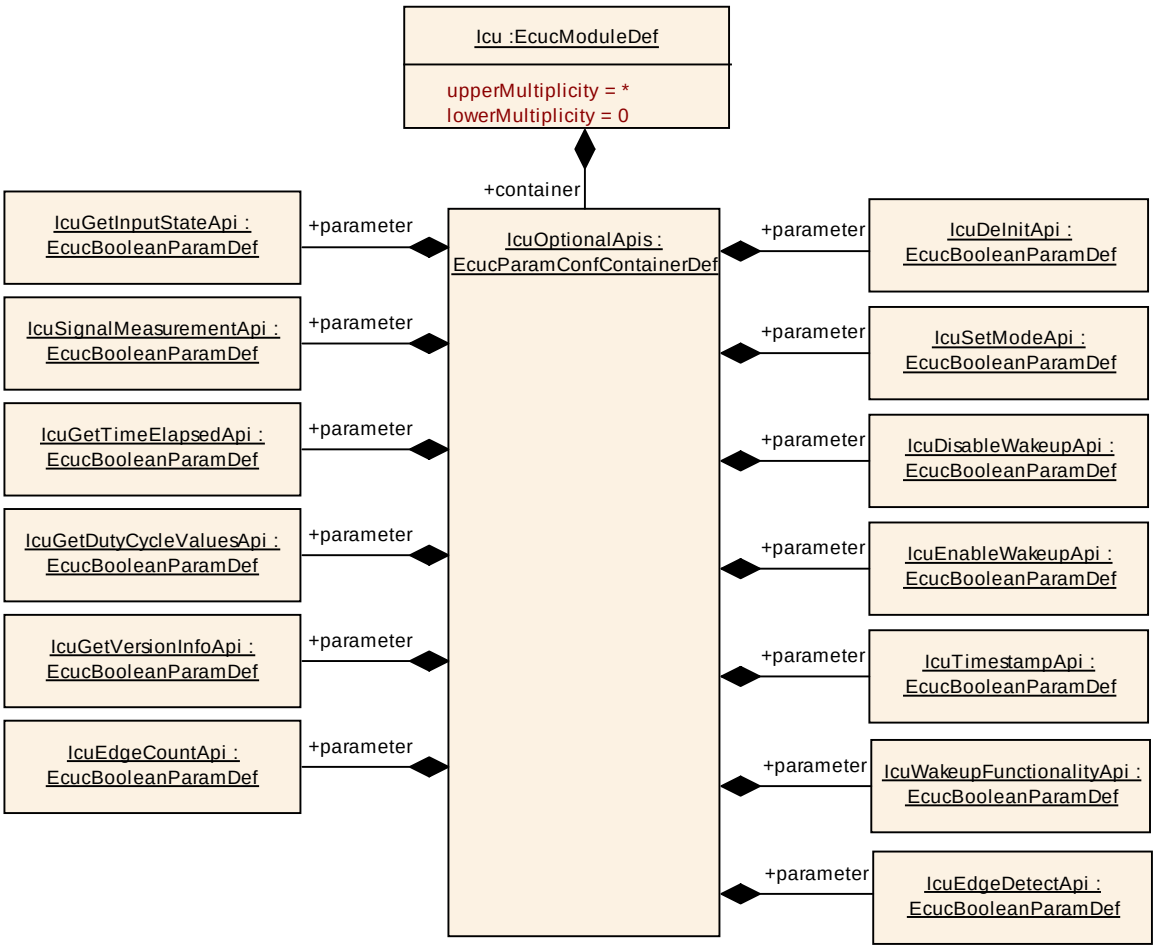
SWS Item	**ECUC_Icu_00242 :**		
Name	IcuSignalMeasurementApi {ICU_SIGNAL_MEASUREMENT_API}		
Description	Adds / removes the services Icu_StartSignalMeasurement() and Icu_StopSignalMeasurement() from the code. true: Icu_StartSignalMeasurement() and Icu_StopSignalMeasurement() can be used. false: Icu_StartSignalMeasurement() and Icu_StopSignalMeasurement() can not be used.		
Multiplicity	1		
Type	EcucBooleanParamDef		

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

SWS Item	**ECUC_Icu_00123 :**		
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Icu_00355 :**		
Name	IcuWakeupFunctionalityApi {ICU_WAKEUP_FUNCTIONALITY_API}		
Description	Adds / removes the service Icu_CheckWakeup() from the code. true: Icu_CheckWakeup() can be used. false: Icu_CheckWakeup() cannot be used.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers



10.2.5 IcuChannel

SWS Item	**ECUC_Icu_00027 :**
Container Name	IcuChannel{Channel configuration}
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	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Icu_00222 :**
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

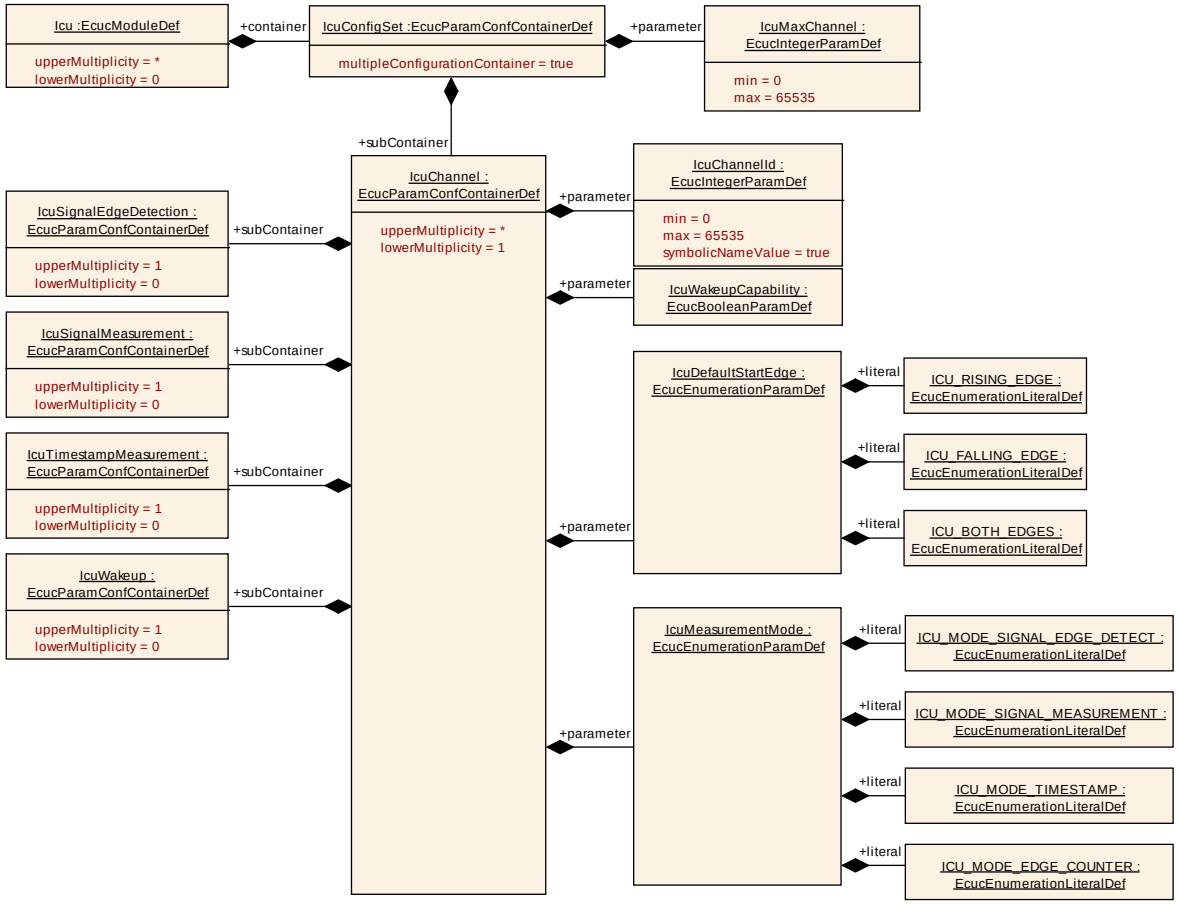
	Icu_SetActivationCondition(). In case the Measurement Mode is "IcuSignalMeasurement" and the properties "DutyCycle" or "Period" are set, the edge configured here is used as Default Period Start Edge. Implementation Type: Icu_ActivationType		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	ICU_BOTH_EDGES	As default, both edges are used.	
	ICU_FALLING_EDGE	As default, falling edge is the used.	
	ICU_RISING_EDGE	As default, rising edge is the used.	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_Icu_00223 :**	
Name	IcuMeasurementMode {Icu_MeasurementMode}	
Description	Configures the measurement mode of this channel. Implementation Type: Icu_MeasurementModeType	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	ICU_MODE_EDGE_COUNTER	The channnel is used to count the edges which are configured by the call of the service Icu_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	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: The possible measurement modes are depending on the pre-processor switches, which enable/disable optional API services.		

SWS Item	**ECUC_Icu_00224 :**		
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
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	**ECUC_Icu_00021 :**
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	**ECUC_Icu_00225 :**		
Name	IcuSignalNotification {Icu_SignalNotification_<Channel>}		
Description	Notification function for signal notification.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: IcuMeasurementMode		

No Included Containers

10.2.7 IcuSignalMeasurement

SWS Item	**ECUC_Icu_00226 :**
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	**ECUC_Icu_00227 :**		
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	EcucEnumerationParamDef		
Range	ICU_DUTY_CYCLE	The channel is configured to read values which are needed for calculating the duty cycle (coherent Active and Period Time).	
	ICU_HIGH_TIME	The channel is configured for reading the elapsed Signal High Time	
	ICU_LOW_TIME	The channel is configured for reading the elapsed Signal Low Time	
	ICU_PERIOD_TIME	The channel is configured for reading the elapsed Signal Period Time	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: IcuMeasurementMode, IcuGetDutyCycleValuesApi, IcuGetTimeElapsedApi		

No Included Containers

10.2.8 IcuTimestampMeasurement

SWS Item	**ECUC_Icu_00228 :**
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	**ECUC_Icu_00229 :**		
Name	IcuTimestampMeasurementProperty {Icu_TimestampMeasurementProperty}		
Description	Configures the handling of the buffer in case the mode is "Timestamp" Implementation Type: Icu_TimestampBufferType		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	ICU_CIRCULAR_BUFFER	After reaching the end of the buffer, the driver restarts at the beginning of the buffer	
	ICU_LINEAR_BUFFER	The buffer will just be filled once	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	

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

SWS Item	**ECUC_Icu_00230 :**		
Name	IcuTimestampNotification {Icu_TimestampNotification_<Channel>}		
Description	Notification function if the number of requested timestamps (Notification interval > 0) are acquired.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: IcuTimestampApi		

No Included Containers

10.2.9 IcuWakeup

SWS Item	**ECUC_Icu_00126 :**
Container Name	IcuWakeup{Wakeup Configuration}
Description	This container contains the configuration (parameters) needed to configure a wakeup capable channel
Configuration Parameters	

SWS Item	**ECUC_Icu_00231 :**		
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	0..1		
Type	Symbolic name reference to [EcuMWakeupSource]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: IcuWakeupCapability and IcuReportWakeupSource		

No Included Containers

10.2.10 IcuConfigSet

SWS Item	**ECUC_Icu_00219 :**
Container Name	IcuConfigSet [Multi Config Container]
Description	This container is the base for a multiple configuration set
Configuration Parameters	

SWS Item	**ECUC_Icu_00220 :**		
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	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

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

10.3 Published Information

[SWS_Icu_00131] ⌈The ICU driver shall describe which other modules (in which versions) are required. This description shall be done by the implementer.⌋ (SRS_BSW_00384)

11 Not applicable requirements

[SWS_Icu_00380] ⌈ These requirements are not applicable to this specification.⌋ (SRS_BSW_00300, SRS_BSW_00301, SRS_BSW_00302, SRS_BSW_00304, SRS_BSW_00305, SRS_BSW_00306, SRS_BSW_00307, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00310, SRS_BSW_00312, SRS_BSW_00314, SRS_BSW_00318, SRS_BSW_00321, BSW00324, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00327, SRS_BSW_00328, SRS_BSW_00329, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00333, SRS_BSW_00334, SRS_BSW_00335, SRS_BSW_00341, SRS_BSW_00342, SRS_BSW_00347, SRS_BSW_00348, SRS_BSW_00350, SRS_BSW_00353, SRS_BSW_00355, SRS_BSW_00357, SRS_BSW_00358, SRS_BSW_00360, SRS_BSW_00361, SRS_BSW_00370, SRS_BSW_00371, SRS_BSW_00373, SRS_BSW_00376, SRS_BSW_00377, SRS_BSW_00378, SRS_BSW_00379, SRS_BSW_00383, SRS_BSW_00387, SRS_BSW_00395, SRS_BSW_00397, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00408, SRS_BSW_00409, SRS_BSW_00413, SRS_BSW_00414, SRS_BSW_00005, SRS_BSW_00006, SRS_BSW_00007, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00160, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00164, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00172, SRS_BSW_00415, SRS_BSW_00416, SRS_BSW_00417, BSW00420, BSW00421, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, BSW00431, SRS_BSW_00432, SRS_BSW_00433, BSW00434, SRS_BSW_00437, SRS_BSW_00439, SRS_BSW_00440, SRS_BSW_00441, SRS_SPAL_12068, SRS_SPAL_12077, SRS_SPAL_12092, SRS_SPAL_12265, SRS_SPAL_12463)