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

Document Version	4.2.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Change Description
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'
18.12.2009	4.0.0	AUTOSAR Administration	• Requirements splitted for conformance test purposes. • Debugging Concept introduced. • Wake-up and Sleep Concept finalized. • Edge detection concept reworked. • Legal disclaimer revised
23.06.2008	3.0.1	AUTOSAR Administration	• Legal disclaimer revised
07.12.2007	3.0.0	AUTOSAR Administration	• The code file structure of the module was completely reworked. • The following requirements were added: ICU088, ICU220, ICU221, ICU228 and ICU229. • The flow charts related to the ECU Wake-Up moved to the • SWS document of the ECU State Manager. • Document meta information extended • Small layout adaptations made

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

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

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)

[ICU243] ⌈In development mode the DET will be called.

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

[ICU244] ⌈The ICU driver will do the reporting of wakeup interrupts to the EcuM.⌋ ()

5.6 File structure

5.6.1 Code file structure

The code file structure shall not be defined within this specification.
At this point it shall be pointed out that the code-file structure shall include the following files named

- Icu_Lcfg.c – for link time configurable parameters and
- Icu_PBcfg.c – for post build time configurable parameters.

These files shall contain all link time and post-build time configurable parameters

5.6.2 Header file structure

The code file structure shall be as follows:

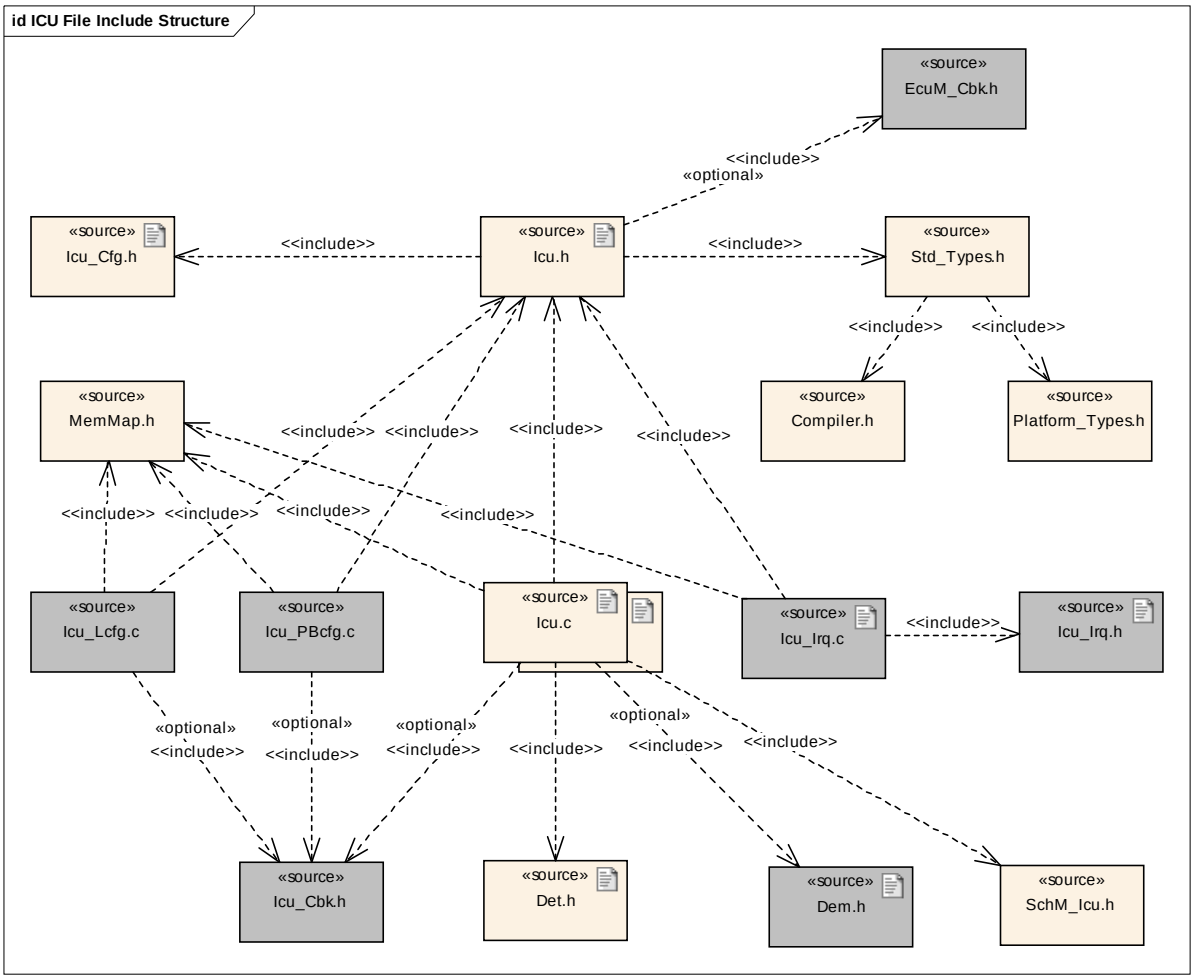


Figure 5.1: Header file structure

[ICU219] ⌈`Icu.c` shall include `Icu.h`⌋ ()

[ICU245] ⌈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.⌋ ()

[ICU246] ⌈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.⌋ ()

[ICU247] ⌈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⌋ ()
```

[ICU248] ⌈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.⌋ ()

[ICU249] ⌈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.⌋ (BSW00435)

[ICU250] ⌈Icu.c shall include Icu_Cbk.h for pre-compile time configuration⌋ ()

[ICU251] ⌈Icu.c shall include Det.h, SchM_Icu.h and MemMap.h.⌋ (BSW00436)

[ICU252] ⌈Icu_Irq.c shall include MemMap.h.⌋ ()

[ICU253] ⌈Icu_Lcfg.c shall include [ICU.h and MemMap.h.⌋ ()

[ICU254] ⌈Icu_PBcfg.c shall include MemMap.h and Icu.h.⌋ ()

[ICU256] ⌈Icu.h shall include EcuM_Cbk.h.⌋ ()

[ICU116] ⌈The module shall optionally include the `Dem.h` file if any production error will be issued by the implementation. By this inclusion, the API's to report errors as well as the required Event Id symbols are included.
This specification defines the name of the Event Id symbols, which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in `Dem_IntErrId.h`.⌋ ()

6 Requirements traceability

Requirement	Satisfied by
-	ICU199
-	ICU287
-	ICU321
-	ICU143
-	ICU203
-	ICU201
-	ICU208
-	ICU245
-	ICU171
-	ICU211
-	ICU181
-	ICU229
-	ICU228
-	ICU372
-	ICU281
-	ICU314
-	ICU220
-	ICU367
-	ICU294
-	ICU164
-	ICU353
-	ICU293
-	ICU288
-	ICU334
-	ICU361
-	ICU276
-	ICU248
-	ICU146
-	ICU091
-	ICU368
-	ICU317
-	ICU316
-	ICU144
-	ICU260
-	ICU341
-	ICU376
-	ICU141
-	ICU250

-	ICU324
-	ICU207
-	ICU194
-	ICU291
-	ICU161
-	ICU326
-	ICU373
-	ICU364
-	ICU330
-	ICU354
-	ICU358
-	ICU159
-	ICU333
-	ICU318
-	ICU325
-	ICU319
-	ICU152
-	ICU200
-	ICU196
-	ICU371
-	ICU338
-	ICU180
-	ICU191
-	ICU205
-	ICU286
-	ICU342
-	ICU252
-	ICU343
-	ICU306
-	ICU305
-	ICU259
-	ICU366
-	ICU290
-	ICU292
-	ICU313
-	ICU216
-	ICU178
-	ICU212
-	ICU254
-	ICU289
-	ICU377
-	ICU218

-	ICU280
-	ICU329
-	ICU352
-	ICU300
-	ICU156
-	ICU369
-	ICU206
-	ICU331
-	ICU059
-	ICU118
-	ICU309
-	ICU256
-	ICU197
-	ICU140
-	ICU298
-	ICU349
-	ICU179
-	ICU275
-	ICU263
-	ICU172
-	ICU217
-	ICU258
-	ICU198
-	ICU163
-	ICU323
-	ICU148
-	ICU121
-	ICU213
-	ICU301
-	ICU214
-	ICU279
-	ICU175
-	ICU337
-	ICU335
-	ICU274
-	ICU346
-	ICU365
-	ICU246
-	ICU219
-	ICU378
-	ICU202
-	ICU299

-	ICU137
-	ICU332
-	ICU190
-	ICU339
-	ICU277
-	ICU360
-	ICU204
-	ICU142
-	ICU195
-	ICU362
-	ICU359
-	ICU308
-	ICU170
-	ICU113
-	ICU112
-	ICU221
-	ICU173
-	ICU320
-	ICU296
-	ICU176
-	ICU356
-	ICU174
-	ICU134
-	ICU284
-	ICU145
-	ICU210
-	ICU345
-	ICU064
-	ICU193
-	ICU350
-	ICU315
-	ICU283
-	ICU348
-	ICU169
-	ICU116
-	ICU166
-	ICU278
-	ICU310
-	ICU322
-	ICU150
-	ICU297
-	ICU139

-	ICU135
-	ICU247
-	ICU370
-	ICU344
-	ICU340
-	ICU347
-	ICU336
-	ICU050
-	ICU162
-	ICU375
-	ICU363
-	ICU253
-	ICU149
-	ICU374
-	ICU160
-	ICU312
-	ICU327
-	ICU351
-	ICU273
-	ICU303
-	ICU285
-	ICU209
-	ICU117
-	ICU295
-	ICU307
-	ICU243
-	ICU304
-	ICU155
-	ICU244
-	ICU177
-	ICU261
-	ICU136
-	ICU311
-	ICU302
-	ICU138
-	ICU165
-	ICU328
0x0A	ICU001
0x0B	ICU272
0x0C	ICU264
0x0D	ICU265
0x0E	ICU266

0x0F	ICU267
0x14	ICU268
0x15	ICU269
0x16	ICU270
0x17	ICU271
0x18	ICU355
0x19	ICU357
BSW003	ICU005
BSW00300	ICU380
BSW00301	ICU380
BSW00302	ICU380
BSW00304	ICU380
BSW00305	ICU380
BSW00306	ICU380
BSW00307	ICU380
BSW00308	ICU380
BSW00309	ICU380
BSW00310	ICU380
BSW00312	ICU380
BSW00314	ICU380
BSW00318	ICU380
BSW00321	ICU380
BSW00323	ICU043, ICU048, ICU024, ICU023, ICU022, ICU120, ICU125
BSW00324	ICU380
BSW00325	ICU380
BSW00326	ICU380
BSW00327	ICU380
BSW00328	ICU380
BSW00329	ICU380
BSW00330	ICU380
BSW00331	ICU380
BSW00333	ICU380
BSW00334	ICU380
BSW00335	ICU380
BSW00336	ICU035, ICU037
BSW00337	ICU004
BSW00338	ICU111, ICU002
BSW00339	ICU003
BSW00341	ICU380
BSW00342	ICU380
BSW00344	ICU006
BSW00347	ICU380

BSW00348	ICU380
BSW00350	ICU380
BSW00353	ICU380
BSW00355	ICU380
BSW00357	ICU380
BSW00358	ICU380
BSW00359	ICU187
BSW00360	ICU380
BSW00361	ICU380
BSW00369	ICU049, ICU002
BSW00370	ICU380
BSW00371	ICU380
BSW00373	ICU380
BSW00374	ICU005
BSW00376	ICU380
BSW00377	ICU380
BSW00378	ICU380
BSW00379	ICU380
BSW00383	ICU380
BSW00387	ICU380
BSW00395	ICU380
BSW00397	ICU380
BSW00398	ICU380
BSW00399	ICU380
BSW004	ICU005
BSW00400	ICU380
BSW00404	ICU006
BSW00405	ICU006
BSW00406	ICU022
BSW00407	ICU182, ICU183
BSW00408	ICU380
BSW00409	ICU380
BSW00410	ICU055, ICU099, ICU094, ICU095, ICU096, ICU097, ICU090, ICU092, ICU063, ICU111, ICU100, ICU101, ICU106, ICU102, ICU103, ICU104, ICU105, ICU122
BSW00411	ICU094
BSW00413	ICU380
BSW00414	ICU380
BSW00415	ICU380
BSW00416	ICU380
BSW00417	ICU380
BSW00420	ICU380
BSW00421	ICU380

BSW00422	ICU380
BSW00423	ICU380
BSW00424	ICU380
BSW00425	ICU380
BSW00426	ICU380
BSW00427	ICU380
BSW00428	ICU380
BSW00429	ICU380
BSW00431	ICU380
BSW00432	ICU380
BSW00433	ICU380
BSW00434	ICU380
BSW00435	ICU249
BSW00436	ICU251
BSW00437	ICU380
BSW00439	ICU380
BSW00440	ICU380
BSW00441	ICU380
BSW005	ICU380
BSW006	ICU380
BSW007	ICU380
BSW009	ICU380
BSW010	ICU380
BSW101	ICU006
BSW12056	ICU018
BSW12057	ICU040, ICU061, ICU060, ICU006
BSW12063	ICU083, ICU082, ICU081, ICU063
BSW12064	ICU133
BSW12067	ICU012, ICU011, ICU008
BSW12068	ICU380
BSW12069	ICU055, ICU057, ICU056
BSW12075	ICU063
BSW12077	ICU380
BSW12092	ICU380
BSW12125	ICU054
BSW12129	ICU119
BSW12163	ICU035, ICU037, ICU036
BSW12169	ICU008
BSW12265	ICU380
BSW12305	ICU044, ICU042, ICU010, ICU009
BSW12368	ICU039
BSW12369	ICU021

BSW12370	ICU008
BSW12371	ICU031, ICU030, ICU032
BSW12407	ICU040, ICU061
BSW12408	ICU014, ICU013
BSW12425	ICU039, ICU088
BSW12429	ICU036
BSW12430	ICU063, ICU066
BSW12431	ICU067
BSW12432	ICU078
BSW12433	ICU079
BSW12434	ICU080
BSW12435	ICU082
BSW12436	ICU084
BSW12438	ICU063
BSW12439	ICU072, ICU073, ICU074
BSW12442	ICU081
BSW12443	ICU083
BSW12444	ICU215
BSW12448	ICU049, ICU048, ICU107, ICU108
BSW12453	ICU071
BSW12455	ICU039
BSW12456	ICU039, ICU065
BSW12461	ICU051, ICU053, ICU052, ICU006, ICU128, ICU129
BSW12463	ICU380
BSW13100	ICU072
BSW157	ICU021, ICU030, ICU003, ICU002
BSW160	ICU380
BSW161	ICU380
BSW162	ICU380
BSW164	ICU380
BSW167	ICU380
BSW168	ICU380
BSW170	ICU380
BSW171	ICU098, ICU099, ICU094, ICU095, ICU096, ICU097, ICU092, ICU100, ICU101, ICU106, ICU102, ICU103, ICU104, ICU105, ICU122
BSW172	ICU380

7 Functional specification

7.1 General behavior

7.1.1 Background & Rationale

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

7.1.2 Requirements

[ICU050] ⌈The Icu module functions for different channel numbers shall be re-entrant, except for:
- `Icu_Init()`
- `Icu_DeInit()`
- `Icu_SetMode()`
- `Icu_GetVersionInfo()`⌋ ()

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

[ICU150] ⌈The Icu module shall not check the integrity if several calls for the same ICU channel are used during runtime in different tasks or ISRs. ⌋ ()

[ICU258] ⌈The Icu module has 2 modes:
- `ICU_MODE_NORMAL`
- `ICU_MODE_SLEEP`

⌋ ()

In `ICU_MODE_NORMAL` mode all notifications are available as

- **[ICU011]** ⌈configured by service `Icu_SetActivationCondition()` or `IcuDefaultStartEdge.` ⌋ (BSW12067)
- **[ICU259]** ⌈selected by the `Icu_DisableNotification()` and `Icu_EnableNotification()` services before or after the call of `Icu_SetMode().`⌋ ()

In `ICU_MODE_SLEEP` mode

- **[ICU012]** ⌈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()`⌋ (BSW12067)

- **[ICU260]** ⌈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. ⌋ ()
- **[ICU261]** ⌈All channels are stopped except those channels
  - which have been configured as wakeup capable and
  - which were explicitly enabled by the call of `Icu_EnableWakeup`.⌋ ()

[ICU088] ⌈The module Icu shall allow the configuration per channel of the definition on which edge the period starts.⌋ (BSW12425)

7.1.3 Version check

[ICU005] ⌈The ICU module shall perform Inter-Module checks to avoid integration of incompatible files.

The imported include files shall be checked by pre-processor directives.

The following version numbers shall be verified:
- `<MODULENAME>_AR_RELEASE_MAJOR_VERSION`
- `<MODULENAME>_AR_RELEASE_MINOR_VERSION`

Where `<MODULENAME>` is the module short name of the other (external) modules which provide header files included by the ICU module.

If the values are not identical to the expected values, an error shall be reported.⌋ (BSW003, BSW00374, BSW004)

7.1.4 Time Unit Ticks

7.1.4.1 Background & Rationale

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

7.1.4.2 Requirements

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

7.2 Error classification

7.2.1 Background & Rationale

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

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

- Production / series
  Those errors are hardware errors and software exceptions that cannot be avoided.

7.2.2 Requirements

[ICU117] ⌈Values for production code event ID's are assigned externally by the configuration of Diagnostic Event Manager.⌋()

[ICU263] ⌈Values for production code are published in the file `Dem_IntErrId.h` and included via `Dem.h`.⌋()

[ICU118] ⌈Development error values are of type uint8.⌋()

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⌋(BSW00337, BSW00385)	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	Development	`ICU_E_PARAM_CHANNEL`	`0x0B`

Type or error	*Relevance*	*Related error code*	*Value [hex]*
configured for the functionality of the calling API.⌋()			
[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 `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⌋()	Development	`ICU_E_NOT_STARTED`	`0x15`
[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	Development	`ICU_E_ALREADY_INITIALIZED`	`0x17`

Type or error	*Relevance*	*Related error code*	*Value [hex]*
Hardware are already initialized.」()			
[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

[ICU111] ⌈The detection of development errors is configurable at pre-compile time.⌋ (BSW00338, BSW00410)

[ICU274] ⌈The detection of development errors is configurable (`ON/OFF`). ⌋()

[ICU273] ⌈The switch `IcuDevErrorDetect` shall activate or deactivate the detection of all development errors.⌋()

[ICU112] ⌈If the switch `IcuDevErrorDetect` is enabled, API parameter checking is enabled.

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

[ICU113] ⌈The detection of production code errors cannot be switched off. ⌋()

[ICU048] ⌈If development error detection for the Icu module is enabled: All Icu module functions shall skip functionality and return without any action (except for raising the development error) if a development error is detected. ⌋(BSW00323, BSW12448)

[ICU022] ⌈If development error detection for the Icu module is enabled: All Icu module functions, except for `Icu_Init` and `Icu_GetVersionInfo`, shall raise

development error `ICU_E_UNINIT` when the function `Icu_Init` has not been called.」(BSW00323, BSW00406)

7.4 Error notification

[ICU002] ⌈Detected development errors shall be reported to the `Det_ReportError` service of the Development Error Tracer (DET) if the pre-processor switch `IcuDevErrorDetect` is set (see ICU026_Conf).」(BSW00338, BSW00369, BSW157)

[ICU003] ⌈Production errors shall be reported to the Diagnostic Event Manager.」(BSW00339, BSW157)

[ICU004] ⌈Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the ICU device specific implementation specification. The classification and enumeration shall be compatible with the errors listed above.」(BSW00337)

7.5 Debugging Concept

7.5.1 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.

7.5.2 Requirements

[ICU350] ⌈Each variable that shall be accessible by AUTOSAR debugging, shall be defined as global variable.」()

[ICU351] ⌈All type definitions of variables which shall be debugged shall be accessible by the header file Icu.h.」()

[ICU352] ⌈The declaration of variables in the header file shall be such, that it is possible to calculate the size of the variables by C-"sizeof".」()

[ICU353] ⌈Variables available for debugging shall be described in the respective Basic Software Module description.」()

8 API specification

8.1 Imported types

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

[ICU190] ⌈`Dem_EventIdType` shall be imported from `Dem_Types.h`.⌋ ()

[ICU275] ⌈`Std_VersionInfoType` shall be imported from `Std_Types.h`. ⌋ ()

[ICU276] ⌈`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

[ICU277] ⌈

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

[ICU278] ⌈

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

[ICU279] ⌈

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

[ICU280] ⌈

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.	

⌋()

[ICU281] ⌈The `Icu_ConfigType` shall contain:
Optional parameters
- MCU dependent properties for used HW units.
- Clock source with optional prescaler (if provided by HW).⌋()

[ICU039] ⌈The definition for each Channel within the `Icu_ConfigType` shall contain:
Common parameters
- Default Start Edge
- Hardware Specific Settings per channel
- Measurement Mode
    - Signal Edge Detection / Notification
    - Signal Measurement
    - Timestamp
    - Edge Counter

Specific parameters

⌋(BSW12368, BSW12425, BSW12455, BSW12456)

[ICU283] ⌈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.⌋ ()

[ICU284] ⌈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 ICU295.⌋ ()

[ICU285] ⌈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 ICU296.⌋ ()

[ICU378] ⌈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. ⌋ ()

[ICU286] ⌈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 ICU289.⌋ ()

[ICU287] ⌈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`.⌋ ()

[ICU288] ⌈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

[ICU289] ⌈

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

[ICU290] ⌈

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

[ICU291] ⌈

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

[ICU292] ⌈

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

[ICU293] ⌈

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

[ICU294] ⌈

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

[ICU295] ⌈

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_ACTIVE_TIME`	The channel is configured for reading the elapsed Signal Active 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

[ICU296] ⌈

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

⌋ ()

8.3 Function definitions

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

8.3.1 Icu_Init

[ICU191] ⌈

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

⌋ ()

[ICU297] ⌈The function `Icu_Init` shall be non re-entrant.⌋ ()

[ICU298] ⌈The function `Icu_Init` initializes the driver.⌋ ()

[ICU006] ⌈The function `Icu_Init` shall initialize all relevant registers of the configured hardware with the values of the structure referenced by the parameter `ConfigPtr.` ⌋(BSW00344, BSW00404, BSW00405, BSW101, BSW12057, BSW12461)

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

- **[ICU051]** ⌈If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register.⌋ (BSW12461)
- **[ICU052]** ⌈If the register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver.⌋ (BSW12461)
- **[ICU053]** ⌈If the register can affect several hardware modules and if it is not an I/O register it shall be initialized by the MCU driver.⌋ (BSW12461)
- **[ICU128]** ⌈One-time writable registers that require initialization directly after reset shall be initialized by the start-up code.⌋ (BSW12461)
- **[ICU129]** ⌈All other registers shall be initialized by the startup code.⌋ (BSW12461)

[ICU061] ⌈The function `Icu_Init` shall disable all notifications.⌋ (BSW12057, BSW12407)

[ICU121] ⌈The function `Icu_Init` shall disable the wakeup-capability of all channels.⌋ ()

[ICU040] ⌈The function `Icu_Init` shall set all used ICU channels to status `ICU_IDLE`.⌋ (BSW12057, BSW12407)

[ICU060] ⌈The function `Icu_Init` shall set the module mode to `ICU_MODE_NORMAL`.⌋ (BSW12057)

[ICU054] ⌈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).⌋ (BSW12125)

[ICU023] ⌈If development error detection for the Icu module is enabled: The function `Icu_Init` shall check the parameter `ConfigPtr` for not being NULL and shall raise

the development error code `ICU_E_PARAM_CONFIG` if the check fails.⌋ (BSW00323)

[ICU220] ⌈If development error detection for the ICU module is enabled and the function `Icu_Init` is called when the ICU driver and hardware are already initialized, the function `Icu_Init` shall raise development error `ICU_E_ALREADY_INITIALIZED` and return without any action.⌋ ()

[ICU138] ⌈The initialization function of this module shall always have a pointer as a parameter, even though for Variant PC no configuration set shall be given. Instead a NULL pointer shall be passed to the initialization function.⌋ ()

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

ICU048 applies to the function `Icu_Init`.⌋ ()

8.3.2 Icu_DeInit

[ICU193] ⌈

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.

⌋ ()

[ICU035] ⌈The function `Icu_DeInit` shall de-initialize the ICU module.⌋ (BSW00336**,** BSW12163)

[ICU036] ⌈The function `Icu_DeInit` shall set the state of the peripherals used by configuration as the same after power on reset.⌋ (BSW12163, BSW12429)

[ICU300] ⌈Values of registers which are not writeable are excluded from setting the state by the function `Icu_DeInit`.⌋ ()

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

[ICU037] ⌈The function `Icu_DeInit` shall disable all used interrupts and notifications.⌋ (BSW00336, BSW12163)

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

[ICU092] ⌈The function `Icu_DeInit` shall be pre compile time configurable by configuration parameter IcuDeInitApi.⌋ (BSW00410, BSW171)

[ICU301] ⌈The function `Icu_DeInit` shall be configurable `ON/OFF` by configuration parameter `IcuDeInitApi`.⌋ ()

[ICU221] ⌈A re-initialization of the ICU module by executing the `Icu_Init()` function requires a de-initialization before by executing the `Icu_DeInit()` function.⌋ ()

[ICU299] ⌈`Icu_DeInit` operation is Non re-entrant.

ICU022 and ICU048 apply to the function `Icu_DeInit`.⌋ ()

8.3.3 Icu_SetMode

[ICU194] ⌈

Service name:	Icu_SetMode	

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

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

⌋()

[ICU008] ⌈The function `Icu_SetMode` shall set the operation mode to the given mode parameter. 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.⌋(BSW12067, BSW12169, BSW12370)

[ICU302] ⌈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.⌋()

[ICU095] ⌈The function `Icu_SetMode` shall be pre-compile time configurable by the configuration parameter `IcuSetModeApi`.⌋(BSW00410, BSW171)

[ICU303] ⌈The function `Icu_SetMode` shall be configurable `ON/OFF` by the configuration parameter `IcuSetModeApi`.⌋()

[ICU125] ⌈If development error detection is enabled for the module Icu the function `Icu_SetMode` shall check the parameter `Mode` and shall raise the error `ICU_E_PARAM_MODE` if the parameter `Mode` is not within the allowed range set in the configuration.⌋(BSW00323)

[ICU133] ⌈This service can be called during running operations. If so, an ongoing operation that generates interrupts on a wakeup capable channel like e.g. time stamping or edge counting might lead to the ICU module not being able to properly enter sleep mode. This is then a system or ECU configuration issue not a problem of this specification.

ICU022 and ICU048 apply to the function `Icu_SetMode`.⌋(BSW12064)

8.3.4 Icu_DisableWakeup

[ICU195] ⌈

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

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

⌋()

[ICU013] ⌈The function `Icu_DisableWakeup` shall disable the wakeup capability of a single ICU channel.⌋(BSW12408)

[ICU305] ⌈The function `Icu_DisableWakeup` shall disable the wakeup capability of a single ICU channel only for ICU channels configured statically as wakeup capable true.⌋()

[ICU304] ⌈The function `Icu_DisableWakeup` shall be re-entrant.⌋()

[ICU096] ⌈The function `Icu_DisableWakeup` shall be pre compile time configurable by the configuration parameter `IcuDisableWakeupApi`.⌋(BSW00410, BSW171)

[ICU306] ⌈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.⌋()

[ICU024] ⌈If development error detection is enabled: The function `Icu_DisableWakeup` shall check the parameter `Channel` and shall raise development error `ICU_E_PARAM_CHANNEL` if `Channel` is not within the allowed range set in the configuration.⌋(BSW00323)

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

ICU022 and ICU048 apply to the function Icu_DisableWakeup.⌋()

8.3.5 Icu_EnableWakeup

[ICU196] ⌈

Service name:	Icu_EnableWakeup

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

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

⌋()

[ICU307] ⌈The function `Icu_EnableWakeup` shall be re-entrant.⌋()

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

To make the selection effective a call of the function `Icu_SetMode`, requesting the mode `ICU_MODE_SLEEP` is required.⌋(BSW12408)

[ICU097] ⌈The function `Icu_EnableWakeup` shall be pre compile time configurable by configuration parameter `IcuEnableWakeupApi`.⌋(BSW00410, BSW171)

[ICU308] ⌈The function `Icu_EnableWakeup` shall be configurable `ON/OFF` by configuration parameter `IcuEnableWakeupApi`.⌋()

[ICU155] ⌈If development error detection is enabled: The function `Icu_EnableWakeup` shall check the parameter `Channel` and shall raise the error `ICU_E_PARAM_CHANNEL` if `Channel` is invalid.⌋()

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

ICU022 and ICU048 apply to the function `Icu_EnableWakeup`.⌋()

8.3.6 Icu_CheckWakeup

[ICU358] ⌈

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.	

⌋()

[ICU359] ⌈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.⌋()

[ICU360] ⌈The function `Icu_CheckWakeup` is only feasible, if `IcuReportWakeupSource` is statically configured available.⌋()

[ICU361] ⌈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.⌋()

[ICU362] ⌈The function `Icu_CheckWakeup` shall be pre compile time configurable `On/Off` by the configuration parameter: `IcuWakeupFunctionalityApi`⌋()

[ICU363] ⌈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

[ICU197] ⌈

Service name:	Icu_SetActivationCondition	

Syntax:	`void Icu_SetActivationCondition(`
	`  Icu_ChannelType Channel,`
	`  Icu_ActivationType Activation`
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (limited according to ICU050)	
Parameters (in):	Channel	Numeric identifier of the ICU channel
	Activation	Type of activation (if supported by hardware) - ICU_RISING_EDGE

		- ICU_FALLING_EDGE - ICU_BOTH_EDGES
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function sets the activation-edge for the given channel.	

⌋ ()

[ICU090] ⌈The function `Icu_SetActivationCondition` shall set the activation-edge according to Activation parameter for the given channel. This service shall support channels which are configured for the following `IcuMeasurementMode` (for details refer to 8.2.10)

- `ICU_MODE_SIGNAL_EDGE_DETECT`
- `ICU_MODE_TIMESTAMP`
- `ICU_MODE_EDGE_COUNTER`⌋ (BSW00410)

[ICU139] ⌈The function `Icu_SetActivationCondition` shall reset the state for the given channel to `ICU_IDLE`.⌋ ()

[ICU309] ⌈The function `Icu_SetActivationCondition` shall be re-entrant.⌋ ()

[ICU159] ⌈If development error detection is enabled the function `Icu_SetActivationCondition` shall check the parameter `Channel` and shall raise the error `ICU_E_PARAM_CHANNEL` if `Channel` is not within the range set in the configuration.⌋ ()

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

ICU022 and ICU048 apply to the function `Icu_SetActivationCondition`.⌋ (BSW00323)

8.3.8 Icu_DisableNotification

[ICU198] ⌈

Service name:	Icu_DisableNotification

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

Service ID[hex]:	0x06

Sync/Async:	Synchronous	
Reentrancy:	Reentrant (limited according to ICU050)	
Parameters (in):	Channel	Numeric identifier of the ICU channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function disables the notification of a channel.	

⌋ ()

[ICU009] ⌈The function `Icu_DisableNotification` shall disable the notification on the given channel.⌋ (BSW12305)

[ICU310] ⌈The function `Icu_DisableNotification` shall be re-entrant.⌋ ()

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

ICU022 and ICU048 apply to the function Icu_DisableNotification.⌋ ()

8.3.9 Icu_EnableNotification

[ICU199] ⌈

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

⌋ ()

[ICU010] ⌈The function `Icu_EnableNotification` shall enable the notification on the given channel.⌋ (BSW12305)

[ICU311] ⌈The function `Icu_EnableNotification` shall be re-entrant.⌋ ()

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

ICU022 and ICU048 apply to the function `Icu_EnableNotification`.⌋ ()

8.3.10 Icu_GetInputState

[ICU200] ⌈

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

⌋ ()

[ICU313] ⌈`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()`.⌋ ()

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

- `ICU_MODE_SIGNAL_EDGE_DETECT`
- `ICU_MODE_SIGNAL_MEASUREMENT`⌋ (BSW157, BSW12371)

[ICU312] ⌈The function `Icu_GetInputState` shall be re-entrant.⌋ ()

[ICU031] ⌈If an activation edge has been detected the function `Icu_GetInputState` shall return `ICU_ACTIVE` for Edge Detection channels.⌋ (BSW12371)

[ICU314] ⌈For Signal Measurement a channel should be set to `ICU_ACTIVE` not until this measurement has completed and the driver is able to provide useful information on the input signal.⌋ ()

[ICU032] ⌈Once the function `Icu_GetInputState` has returned the status `ICU_ACTIVE`, the function `Icu_GetInputState` shall set the stored status to `ICU_IDLE` until the next edge is detected.⌋ (BSW12371)

[ICU122] ⌈The function `Icu_GetInputState` shall be pre compile time configurable by the configuration parameter `IcuGetInputStateApi`.⌋ (BSW00410, BSW171)

[ICU315] ⌈The function `Icu_GetInputState` shall be configurable `ON/OFF` by the configuration parameter `IcuGetInputStateApi`.⌋ ()

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

[ICU049] ⌈If development error detection is enabled the function `Icu_GetInputState` shall return `ICU_IDLE` if an error is detected.

ICU022 and ICU048 apply to the function `Icu_GetInputState`.⌋ (BSW12448, BSW00369)

8.3.11 Icu_StartTimestamp

[ICU201] ⌈

Service name:	Icu_StartTimestamp	
Syntax:	`void Icu_StartTimestamp( Icu_ChannelType Channel, Icu_ValueType* BufferPtr, uint16 BufferSize, uint16 NotifyInterval )`	
Service ID[hex]:	0x09	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant (limited according to ICU050)	
Parameters (in):	Channel	Numeric identifier of the ICU channel
	BufferPtr	Pointer to the buffer-array where the timestamp values shall be placed.
	BufferSize	Size of the external buffer (number of entries)
	NotifyInterval	Notification interval (number of events). This parameter can not be checked in a reasonable way.

Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This function starts the capturing of timer values on the edges.

⌋ ()

[ICU317] ⌈The function `Icu_StartTimestamp` shall start the capturing of timer values on the edges to an external buffer, at the beginning of the buffer. ⌋ ()

[ICU063] ⌈The function `Icu_StartTimestamp` shall start the capturing of timer values on the edges

- activated by the service `Icu_SetActivationCondition()`
  (rising / falling / both edges)⌋ (BSW00410, BSW12063, BSW12075, BSW12430, BSW12438)

[ICU316] ⌈The function `Icu_StartTimestamp` shall be re-entrant.⌋ ()

[ICU064] ⌈If circular buffer handling is configured (for the given channel), when the capture functionality reaches the end of the buffer, the Icu module shall start at the beginning of the buffer.⌋ ()

[ICU065] ⌈If linear buffer handling is configured, when the capture functionality reaches the end of the buffer, the Icu module shall stop capturing timer values.⌋ (BSW12456)

[ICU134] ⌈The Icu module shall only call a notification function if a notification function is configured.⌋ ()

[ICU318] ⌈The Icu module shall only call a notification function if the notification has been enabled by the call of `Icu_EnableNotification()`.⌋ ()

[ICU319] ⌈The Icu module shall only call a notification function if `NotifyInterval` is greater than “0”.⌋ ()

[ICU320] ⌈The Icu module shall only call a notification function if the number of events specified by `NotifyInterval` has been captured. ⌋ ()

[ICU066] ⌈The function `Icu_StartTimeStamp` shall only be available in Measurement Mode “`ICU_MODE_TIMESTAMP`”.⌋ (BSW12430)

[ICU098] ⌈The function `Icu_StartTimestamp` shall be pre-compile time configurable by the configuration parameter: `ICU_TIMESTAMP_API`.⌋ (BSW171)

[ICU321] ⌈The function `Icu_StartTimestamp` shall be configurable `ON/OFF` by the configuration parameter: `ICU_TIMESTAMP_API`.⌋ ()

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

[ICU120] ⌈If development error detection is enabled: The function `Icu_StartTimestamp` shall check the parameter `BufferPtr`. The function `Icu_StartTimestamp` shall raise the error `ICU_E_PARAM_BUFFER_PTR` if `BufferPtr` is invalid (e.g. “0” means NULL pointer).⌋ (BSW00323)

[ICU354] ⌈If development error detection is enabled the function `Icu_StartTimestamp` shall check the parameter `NotifyInterval` (check that Interval > 0). The function Icu_StartTimestamp shall raise the error `ICU_E_PARAM_NOTIFY_INTERVAL` if `NotifyInterval` is invalid (e.g. “0”).⌋ ()

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

ICU022 and ICU048 apply to the function `Icu_StartTimestamp`.⌋ (BSW12448)

8.3.12 Icu_StopTimestamp

[ICU202] ⌈

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

⌋()

[ICU067] ⌈The function `Icu_StopTimestamp` shall stop the timestamp measurement of the given channel.⌋ (BSW12431)

[ICU322] ⌈`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).⌋ ()

[ICU165] ⌈The function `Icu_StopTimestamp` shall only be available in Measurement Mode: `ICU_MODE_TIMESTAMP`.⌋ ()

[ICU099] ⌈The function `Icu_StopTimestamp` shall be pre-compile time configurable by the configuration parameter: `IcuTimestampApi` (see also chapter 10.2.4. Configuration of optional API services)⌋ (BSW00410, BSW171)

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

[ICU323] ⌈The function `Icu_StopTimestamp` shall be configurable `ON/OFF` by the configuration parameter: `IcuTimestampApi`.⌋ ()

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

ICU022 and ICU048 apply to the function `Icu_StopTimestamp`.⌋ ()

8.3.13 Icu_GetTimestampIndex

[ICU203] ⌈

Service name:	Icu_GetTimestampIndex	

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

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (limited according to ICU050)	
Parameters (in):	Channel	Numeric identifier of the ICU channel

Parameters (inout):	None	
Parameters (out):	None	
Return value:	Icu_IndexType	Abstract return type to cover different microcontrollers.
Description:	This function reads the timestamp index of the given channel.	

⌋ ()

[ICU071] ⌈The function `Icu_GetTimestampIndex` shall read the timestamp index of the given channel, which is the next to be written.⌋ (BSW12453)

[ICU324] ⌈The function `Icu_GetTimestampIndex` shall be re-entrant.⌋ ()

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

[ICU170] ⌈The function `Icu_GetTimestampIndex` shall only be available in Measurement Mode `ICU_MODE_TIMESTAMP`.⌋ ()

[ICU100] ⌈The function `Icu_GetTimestampIndex` shall be pre compile time configurable by the configuration parameter: `IcuTimestampApi`⌋ (BSW00410, BSW171)

[ICU325] ⌈The function `Icu_GetTimestampIndex` shall be configurable `ON/OFF` by the configuration parameter: `IcuTimestampApi`.⌋ ()

[ICU169] ⌈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`.⌋ ()

[ICU107] ⌈If development error detection is enabled the function `Icu_GetTimestampIndex` shall return “0” if an error is detected.

ICU022 and ICU048 apply to the function `Icu_GetTimestampIndex`. ⌋ (BSW12448)

8.3.14 Icu_ResetEdgeCount

[ICU204] ⌈

Service name:	Icu_ResetEdgeCount

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

⌋ ()

[ICU072] ⌈The function `Icu_ResetEdgeCount` shall reset the value of the counted edges to zero.⌋ (BSW12439, BSW13100)

[ICU326] ⌈The function `Icu_ResetEdgeCount` shall be re-entrant.⌋ ()

[ICU101] ⌈The function `Icu_ResetEdgeCount` shall be pre-compile time configurable by the configuration parameter `ICU_EDGE_COUNT_API.`⌋ (BSW00410, BSW171)

[ICU327] ⌈The function `Icu_ResetEdgeCount` shall be configurable `ON/OFF` by the configuration parameter: `ICU_EDGE_COUNT_API`.⌋ ()

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

ICU022 and ICU048 apply to the function `Icu_ResetEdgeCount`.⌋ ()

8.3.15 Icu_EnableEdgeCount

[ICU205] ⌈

Service name:	Icu_EnableEdgeCount	
Syntax:	`void Icu_EnableEdgeCount( Icu_ChannelType Channel )`	
Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (limited according to ICU050)	
Parameters (in):	Channel	Numeric identifier of the ICU channel
Parameters	None	

(inout):	
Parameters (out):	None
Return value:	None
Description:	This function enables the counting of edges of the given channel.

⌋()

[ICU078] ⌈The function `Icu_EnableEdgeCount` shall enable the counting of edges of the given channel.⌋ (BSW12432)

Note: This service does not do the real counting itself. This is done by the hardware.

[ICU073] ⌈The function `Icu_EnableEdgeCount` shall only count the configured[1] edges (rising edge / falling edge / both edges).⌋ (BSW12439)

[ICU074] ⌈The function `Icu_EnableEdgeCount` shall be available for each ICU channel in Measurement Mode "Edge Counter".⌋ (BSW12439)

[ICU328] ⌈The function `Icu_EnableEdgeCount` shall be re-entrant.⌋ ()

[ICU102] ⌈The function `Icu_EnableEdgeCount` shall be pre-compile time configurable by the configuration parameter `ICU_EDGE_COUNT_API`⌋ (BSW00410, BSW171)

[ICU329] ⌈The function `Icu_EnableEdgeCount` shall be configurable `On/Off` by the configuration parameter: `ICU_EDGE_COUNT_API`.⌋ ()

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

ICU022 and ICU048 apply to the function `Icu_EnableEdgeCount`.⌋ ()

8.3.16 Icu_EnableEdgeDetection

[ICU364] ⌈

Service name:	Icu_EnableEdgeDetection

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

Service ID[hex]:	0x16

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

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.	

⌋ ()

[ICU365] ⌈The function `Icu_EnableEdgeDetection` shall enable the detection of edges for the given channel.⌋ ()

[ICU366] ⌈The function `Icu_EnableEdgeDetection` shall only detect the configured edges (rising edge / falling edge / both edges).⌋ ()

[ICU367] ⌈The function `Icu_EnableEdgeDetection` shall be available for each ICU Channel in Measurement Mode “Edge Detection”.⌋ ()

[ICU368] ⌈The function `Icu_EnableEdgeDetection` shall be re-entrant.⌋ ()

[ICU369] ⌈The function `Icu_EnableEdgeDetection` shall be pre-compile time configurable by the configuration parameter `IcuEdgeDetectApi.`⌋ ()

[ICU370] ⌈The function `Icu_EnableEdgeDetection` shall be configurable `ON/OFF` by the configuration parameter: `IcuEdgeDetectApi.`⌋ ()

[ICU371] ⌈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.`

ICU022 and ICU048 apply to the function `Icu_EnableEdgeDetection`⌋ ()

8.3.17 Icu_DisableEdgeDetection

[ICU377] ⌈

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.	

⌋()

[ICU372] ⌈The function `Icu_DisableEdgeDetection` shall disable the detection of edges of the given channel⌋()

[ICU373] ⌈The function `Icu_DisableEdgeDetection` shall be re-entrant.⌋()

[ICU374] ⌈The function `Icu_DisableEdgeDetection` shall be pre-compile time configurable by the configuration parameter `IcuEdgeDetectApi`⌋()

[ICU375] ⌈The function `Icu_DisableEdgeDetection` shall be configurable `ON/OFF` by the configuration parameter `IcuEdgeDetectApi`⌋()

[ICU376] ⌈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.`

ICU022 and ICU048 apply to the function `Icu_DisableEdgeDetection`.⌋()

8.3.18 Icu_DisableEdgeCount

[ICU206] ⌈

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

⌋()

[ICU079] ⌈The function `Icu_DisableEdgeCount` shall disable the counting of edges of the given channel.⌋ (BSW12433)

[ICU330] ⌈The function `Icu_DisableEdgeCount` shall be re-entrant.

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

[ICU103] ⌈The function `Icu_DisableEdgeCount` shall be pre-compile time configurable by the configuration parameter `IcuEdgeCountApi`.⌋ (BSW00410, BSW171)

[ICU331] ⌈The function `Icu_DisableEdgeCount` shall be configurable `ON/OFF` by the configuration parameter `IcuEdgeCountApi`.⌋ ()

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

ICU022 and ICU048 apply to the function `Icu_DisableEdgeCount`.⌋ ()

8.3.19 Icu_GetEdgeNumbers

[ICU207] ⌈

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

⌋ ()

[ICU080] ⌈The function `Icu_GetEdgeNumbers` shall read the number of counted edges after the last call of `Icu_ResetEdgeCount()`.⌋ (BSW12434)

[ICU332] ⌈The function `Icu_GetEdgeNumbers` shall be re-entrant.⌋ ()

[ICU104] ⌈The function `Icu_GetEdgeNumbers` shall be pre compile time configurable by the configuration parameter: `ICU_EDGE_COUNT_API`⌋ (BSW00410, BSW171)

[ICU333] ⌈The function `Icu_GetEdgeNumbers` shall be configurable `ON/OFF` by the configuration parameter: `ICU_EDGE_COUNT_API`.⌋ ()

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

[ICU175] ⌈If development error detection is enabled the function `Icu_GetEdgeNumbers` shall return “0” if an error is detected.

ICU022 and ICU048 apply to the function `Icu_GetEdgeNumbers`.⌋ ()

8.3.20 Icu_StartSignalMeasurement

[ICU208] ⌈

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

⌋ ()

[ICU334] ⌈The function `Icu_StartSignalMeasurement` shall be re-entrant.⌋ ()

[ICU140] ⌈The function `Icu_StartSignalMeasurement` shall start the measurement of signals beginning with the configured default start edge which occurs first after the call of this service.⌋ ()

[ICU141] ⌈The function `Icu_StartSignalMeasurement` shall only be available in Measurement Mode "`ICU_MODE_SIGNAL_MEASUREMENT`".⌋ ()

[ICU146] ⌈The function `Icu_StartSignalMeasurement` shall reset the state for the given channel to `ICU_IDLE`.⌋ ()

[ICU142] ⌈The function `Icu_StartSignalMeasurement` shall be pre-compile time configurable by the configuration parameter `IcuSignalMeasurementApi`⌋ ()

[ICU335] ⌈The function `Icu_StartSignalMeasurement` shall be configurable `ON/OFF` by the configuration parameter `IcuSignalMeasurementApi`.⌋ ()

[ICU176] ⌈If development error detection is enabled, the function `Icu_StartSignalMeasurement` shall check the parameter `Channel`. If `Channel` is invalid (invalid identifier or channel not configured for mode `ICU_MODE_SIGNAL_MEASUREMENT`), the function `Icu_StartSignalMeasurement` shall raise development error `ICU_E_PARAM_CHANNEL`.

ICU022 and ICU048 apply to the function `Icu_StartSignalMeasurement`.⌋ ()

8.3.21 Icu_StopSignalMeasurement

[ICU209] ⌈

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

⌋ ()

[ICU336] ⌈The function `Icu_StopSignalMeasurement` shall be Re-entrant.⌋ ()

[ICU143] ⌈The function `Icu_StopSignalMeasurement` shall stop the measurement of signals of the given channel.⌋ ()

[ICU144] ⌈The function `Icu_StopSignalMeasurement` shall only be available in Measurement Mode"`ICU_MODE_SIGNAL_MEASUREMENT`"⌋ ()

[ICU145] ⌈The function `Icu_StopSignalMeasurement` shall be pre compile time configurable by the configuration parameter `IcuSignalMeasurementApi`⌋ ()

[ICU337] ⌈The function `Icu_StopSignalMeasurement` shall be configurable `ON/OFF` by the configuration parameter `IcuSignalMeasurementApi`.⌋ ()

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

ICU022 and ICU048 apply to the function Icu_StopSignalMeasurement.⌋ ()

8.3.22 Icu_GetTimeElapsed

[ICU210] ⌈

Service name:	Icu_GetTimeElapsed	

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

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

⌋()

[ICU338] ⌈The function `Icu_GetTimeElapsed` shall be re-entrant.⌋()

[ICU081] ⌈The function `Icu_GetTimeElapsed` shall read the elapsed Signal Low Time for the given channel that is configured in Measurement Mode “Signal Measurement, Signal Low Time”. The elapsed time is measured between a falling edge and the consecutive rising edge of the channel.⌋(BSW12063, BSW12442)

[ICU082] ⌈The function `Icu_GetTimeElapsed` shall read the elapsed Signal High Time for the given channel that is configured in Measurement Mode “Signal Measurement, Signal High Time”. The elapsed time is measured between a rising edge and the consecutive falling edge of the channel.⌋(BSW12063, BSW12435)

[ICU083] ⌈The function `Icu_GetTimeElapsed` shall read the elapsed Signal Period Time for the given channel that is configured in Measurement Mode “Signal Measurement, Signal Period Time”. The elapsed time is measured between consecutive rising (or falling) edges of the channel. The period start edge is configurable.⌋(BSW12063, BSW12443)

[ICU136] ⌈The function `Icu_GetTimeElapsed` shall return “0” in case no requested time has been captured (see

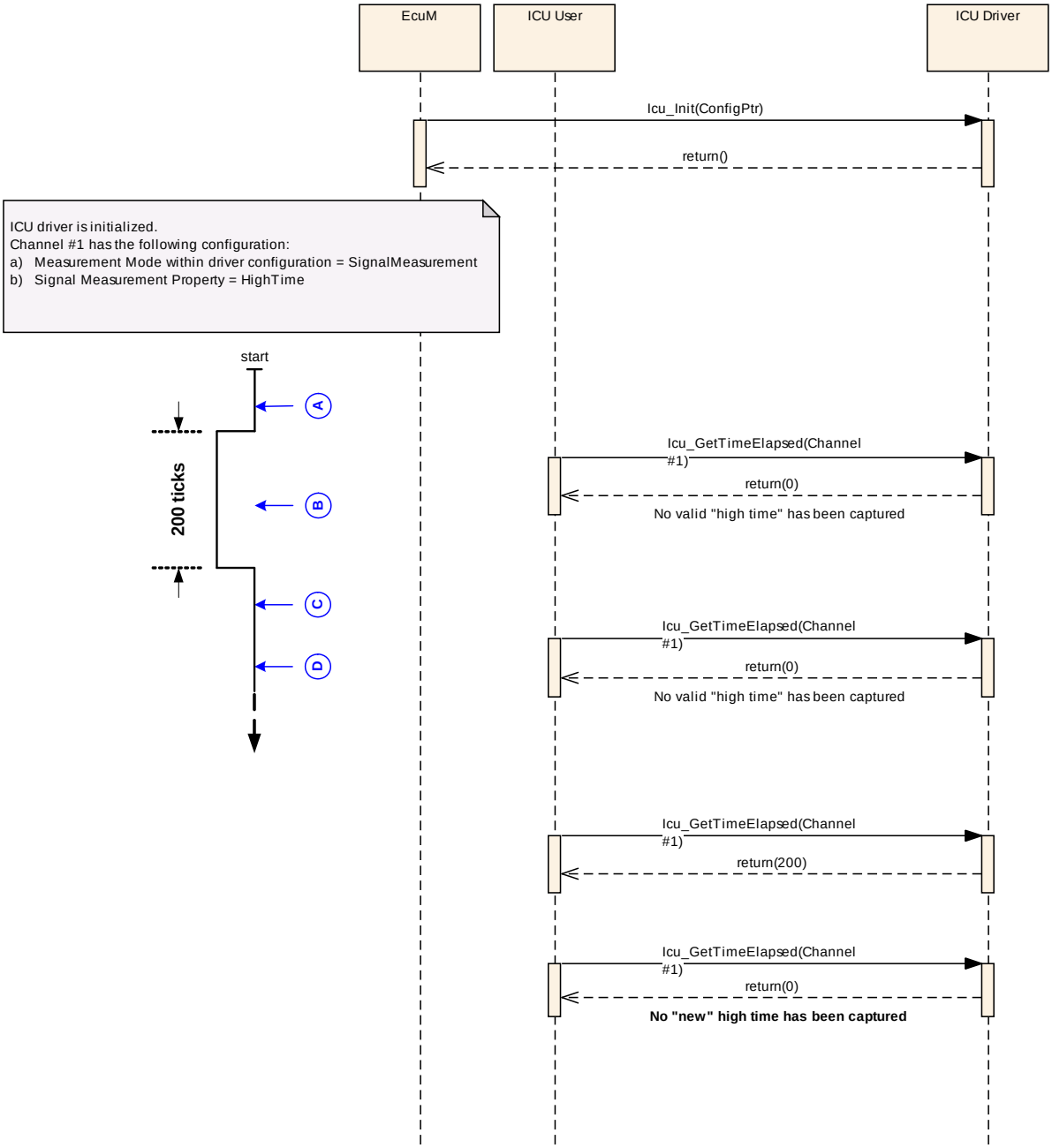


Figure 9.19, letter “A”).」()

[ICU339] ⌈The function `Icu_GetTimeElapsed` shall return “0” in case the capturing of a requested time is ongoing and not finished (see

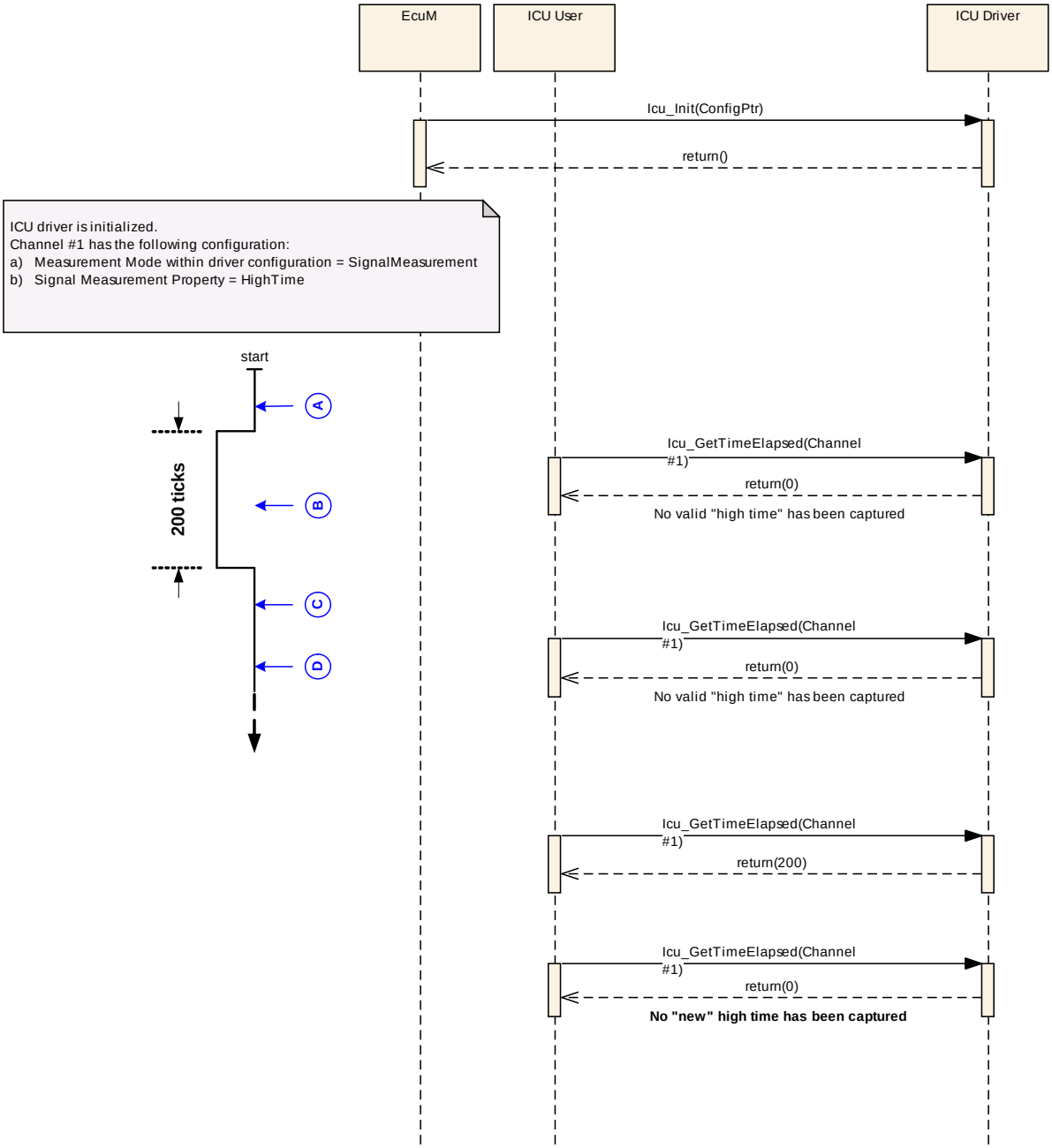


Figure 9.19, letter “B”)⌋ ()

[ICU340] ⌈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

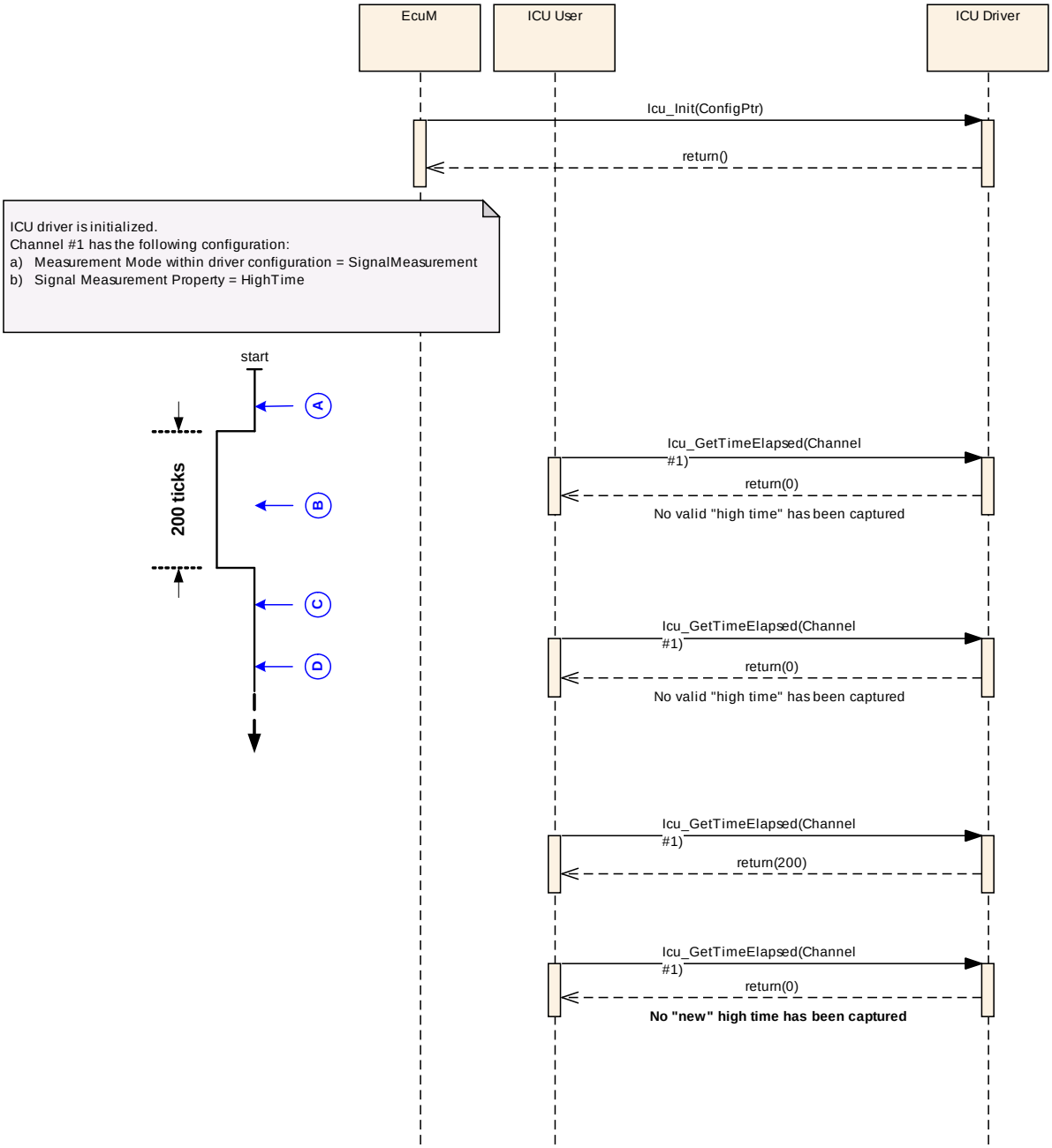


Figure 9.19, letter “D”)⌋ ()

[ICU105] ⌈The function `Icu_GetTimeElapsed` shall be pre compile time configurable by the configuration parameter `IcuGetTimeElapsedApi`.⌋ (BSW00410, BSW171)

[ICU341] ⌈The function `Icu_GetTimeElapsed` shall be configurable `ON/OFF` by the configuration parameter `IcuGetTimeElapsedApi`.⌋ ()

[ICU178] ⌈If development error detection is enabled, the parameter `Channel` shall be checked by this service. If `Channel` is invalid (invalid identifier or channel not

configured for mode ICU_MODE_SIGNAL_MEASUREMENT), then the error ICU_E_PARAM_CHANNEL shall be reported to the Development Error Tracer.⌋ ()

[ICU179] ⌈If development error detection is enabled and an error is detected this service shall return "0".

ICU022 and ICU048 apply to the function Icu_GetTimeElapsed.⌋ ()

8.3.23 Icu_GetDutyCycleValues

[ICU211] ⌈

Service name:	Icu_GetDutyCycleValues	

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

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

⌋ ()

[ICU342] ⌈The function Icu_GetDutyCycleValues shall be re-entrant. ⌋ ()

[ICU084] ⌈The function Icu_GetDutyCycleValues shall read the coherent active time and period time for the given ICU Channel, if it is configured in Measurement Mode "Signal Measurement, Duty Cycle Values".⌋ (BSW12436)

[ICU137] ⌈The function Icu_GetDutyCycleValues shall return "0" in case no coherent active- and period time has been captured (similar to

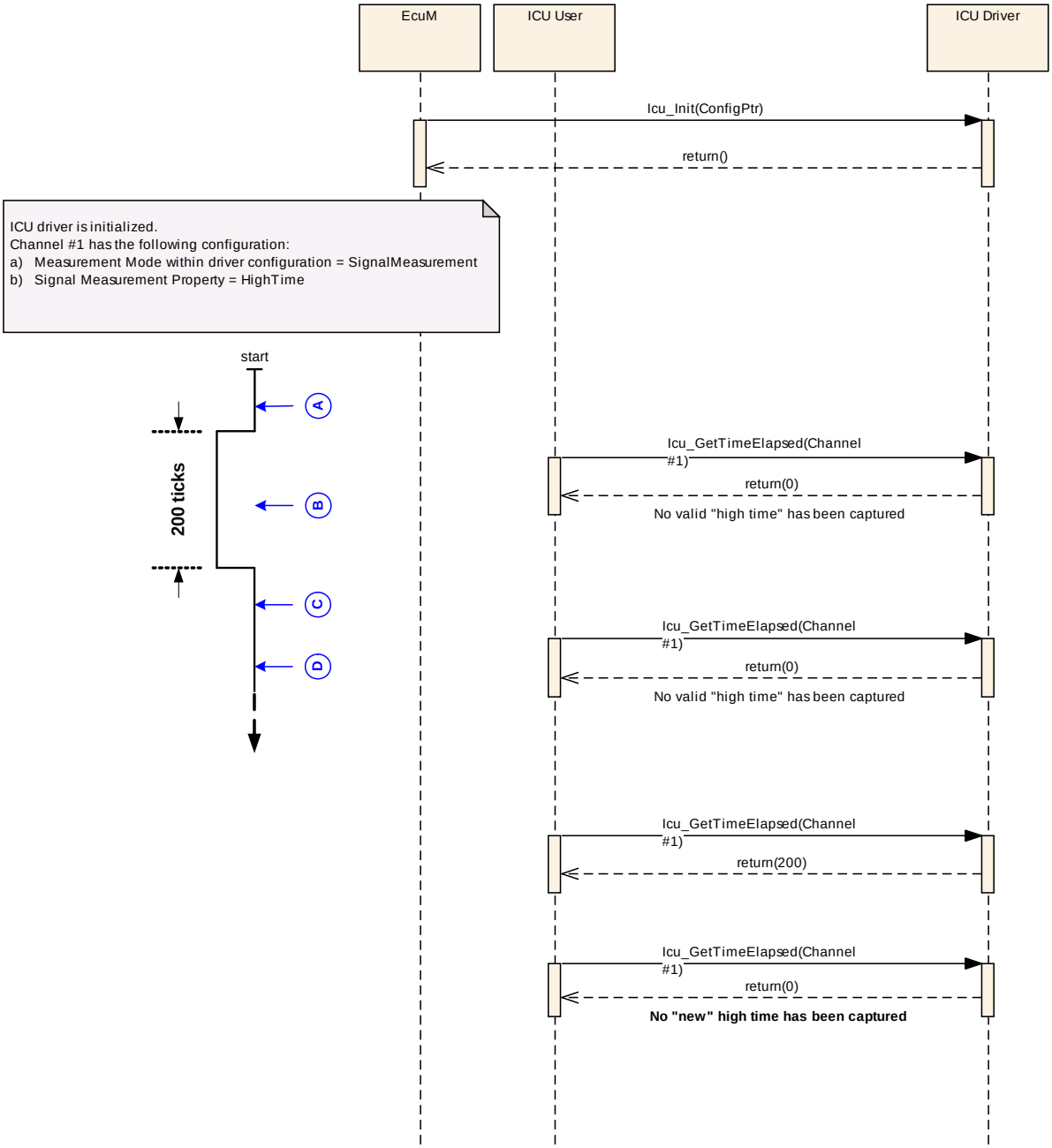


Figure 9.19, letter “A”).⌋ ()

[ICU343] ⌈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

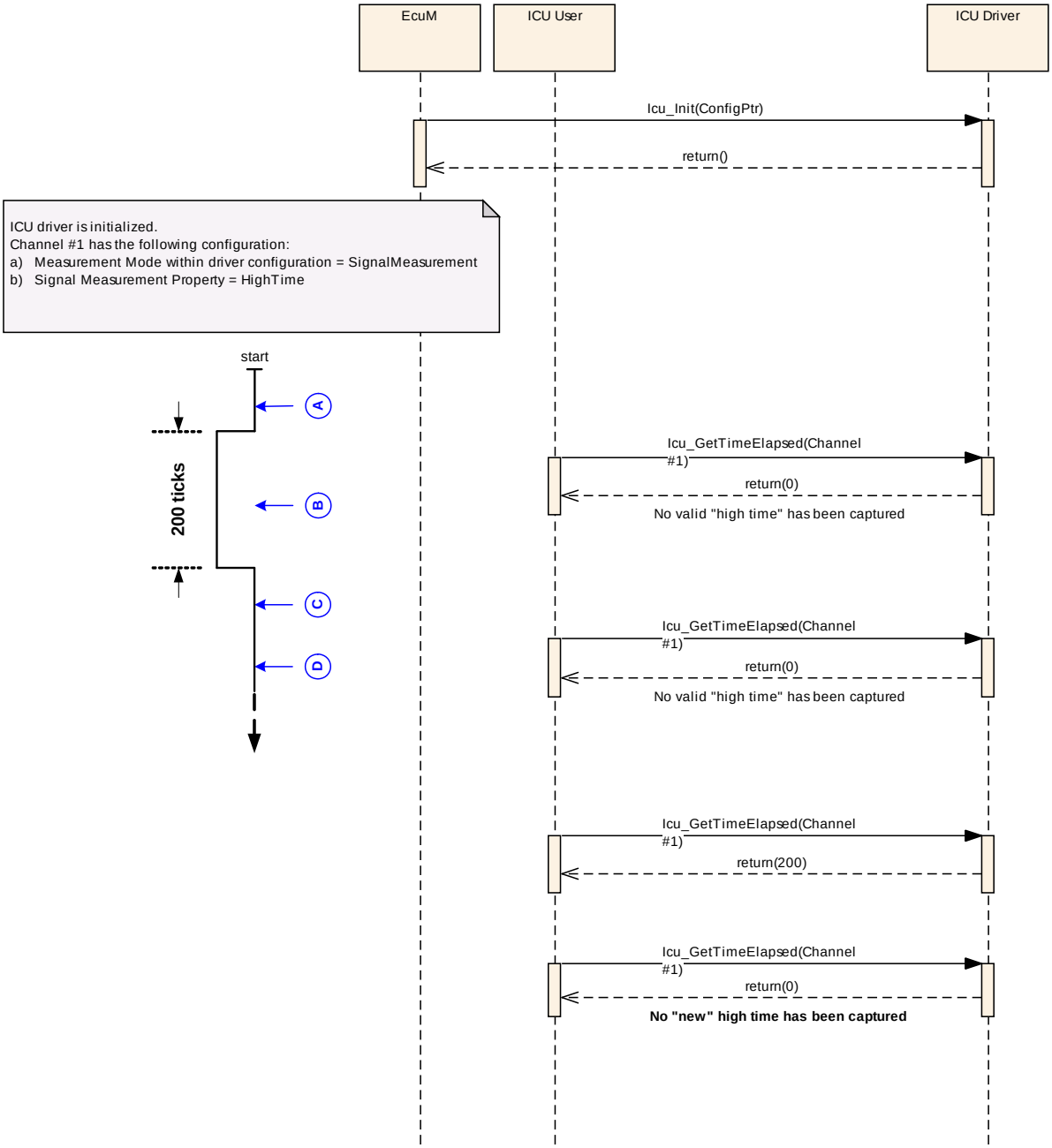


Figure 9.19, letter “B”).」()

[ICU344] ⌈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

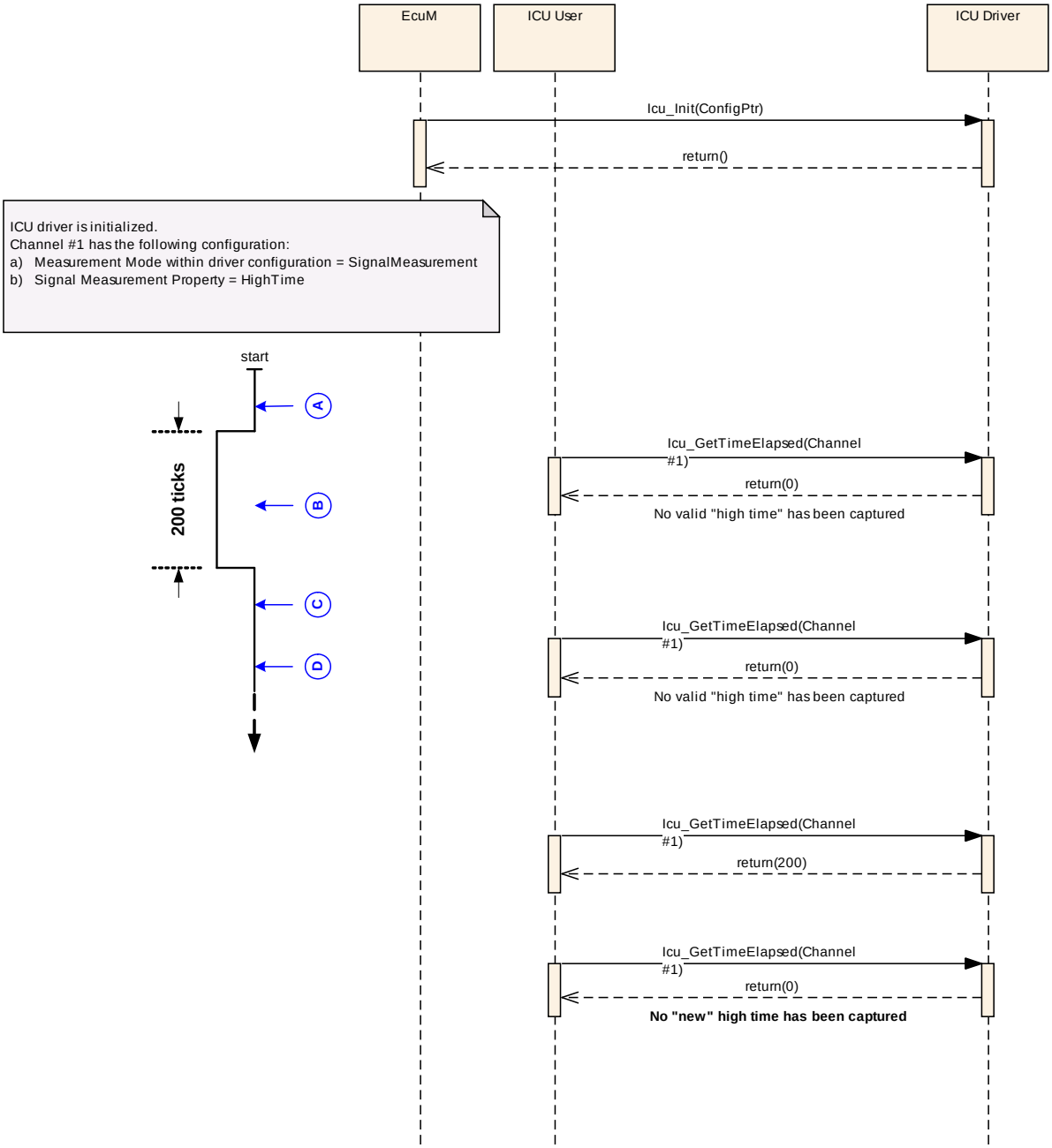


Figure 9.19, letter “D”)⌋ ()

[ICU106] ⌈The function `Icu_GetDutyCycleValues` shall be pre compile time configurable by the configuration parameter `IcuGetDutyCycleValuesApi`.⌋ (BSW00410, BSW171)

[ICU345] ⌈The function `Icu_GetDutyCycleValues` shall be configurable `ON/OFF` by the configuration parameter `IcuGetDutyCycleValuesApi`.⌋ ()

[ICU180] ⌈If development error detection is enabled: the function `Icu_GetDutyCycleValues` shall check the parameter `Channel`. If `Channel` is

invalid (invalid identifier or channel not configured for mode ICU_MODE_SIGNAL_MEASUREMENT, Duty Cycle Values), the function Icu_GetDutyCycleValues shall raise development error ICU_E_PARAM_CHANNEL.⌋ ()

[ICU181] ⌈If development error detection is enabled, the function Icu_GetDutyCycleValues shall check the parameter DutyCycleValues. If DutyCycleValues is invalid, the function Icu_GetDutyCycleValues shall raise development error ICU_E_PARAM_BUFFER_PTR.

ICU022 and ICU048 apply to the function Icu_GetDutyCycleValues.⌋ ()

8.3.24 Icu_GetVersionInfo

[ICU212] ⌈

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.	

⌋ ()

[ICU346] ⌈Icu_GetVersionInfo operation is Non-Re-entrant.⌋ ()

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

- Module Id (See Literature [4])
- Vendor Id
- Vendor specific version numbers.⌋ (BSW00407)

[ICU183] ⌈The Icu module's environment may call the function Icu_GetVersionInfo at any time. ⌋ (BSW00407)

Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.

[ICU094] ⌈The function `Icu_GetVersionInfo` shall be pre compile time configurable by the configuration parameter `IcuGetVersionInfoApi`.⌋ (BSW00410, BSW00411, BSW171)

[ICU356] ⌈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.⌋ ()

[ICU347] ⌈The function `Icu_GetVersionInfo` shall be configurable `ON/OFF` by the configuration parameter `IcuGetVersionInfoApi`.⌋ ()

8.4 Callback notifications

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

8.5 Scheduled functions

None.

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

None.

8.6.2 Optional Interfaces

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

[ICU213] ⌈

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.
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:

- **[ICU055]** ⌈The static configuration parameter `IcuReportWakeupSource` is set to "ON" ⌋(BSW12069, BSW00410)
- **[ICU056]** ⌈The module is in mode `ICU_MODE_SLEEP`⌋(BSW12069)
- **[ICU057]** ⌈A wakeup event occurs on a wakeup capable ICU channel.⌋ (BSW12069)

[ICU228] ⌈`EcuM_CheckWakeup` shall be called within the Interrupt Service Routine servicing the ICU channel wakeup event on wakeup-capable channel.⌋()

[ICU229] ⌈The ISR's, providing the wakeup events, shall be responsible for resetting the interrupt flags if required by hardware.⌋()

8.6.3 Configurable interfaces

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

[ICU119] ⌈The ISRs shall reset the interrupt flags (if needed by hardware) and call the corresponding notification functions.⌋(BSW12129)

[ICU018] ⌈The Icu notification functions shall be configurable as function pointers within the initialization data structure (`Icu_ConfigType`).⌋(BSW12056)

[ICU187] ⌈The Icu module's notification functions shall have no parameters and no return value.⌋(BSW00359)

[ICU214] ⌈

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	None

(inout):	
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).

⌋ ()

[ICU348] ⌈Re-entrancy of operation `Icu_SignalNotification_<Channel>` is not relevant for this module (In general it is in this case not re-entrant).⌋ ()

[ICU021] ⌈According to the last call of `Icu_EnableNotification()`, the Icu module shall call the notification function Icu_SignalNotification_<Channel> if the requested signal edge (rising / falling / both edges) occurs (once per edge).⌋ (BSW157, BSW12369)

[ICU044] ⌈Only those edge notifications shall be provided, which are supported by hardware.⌋ (BSW12305)

[ICU042] ⌈After a call of `Icu_DisableNotification` , the Icu module shall not call the notification function Icu_SignalNotification_<Channel>.⌋ (BSW12305)

[ICU215] ⌈

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().

⌋ (BSW12444)

[ICU349] ⌈Re-entrancy of the `Icu_TimestampNotification_<Channel>` is not relevant for this module (in general it is in this case not re-entrant).⌋ ()

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

[ICU217] ⌈After a call of `Icu_DisableNotification` the Icu module shall NOT call the notification `Icu_TimestampNotification_<Channel>`.⌋ ()

[ICU218] ⌈The Icu module's notification `Icu_TimestampNotification_<Channel>` depends on pre-processor switch `IcuTimestampApi`⌋ ()

9 Sequence diagrams

9.1 Icu_Init

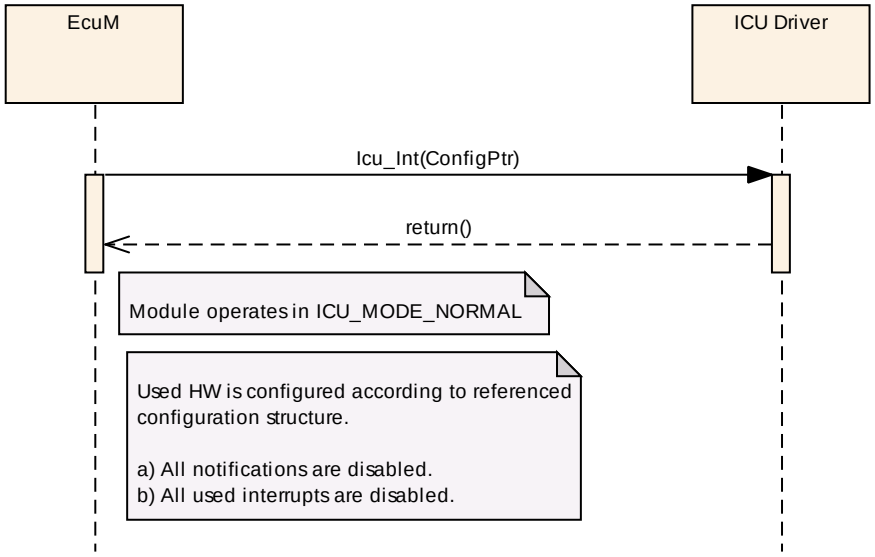


Figure 9.1: Initialization of the ICU driver

9.2 Icu_DeInit

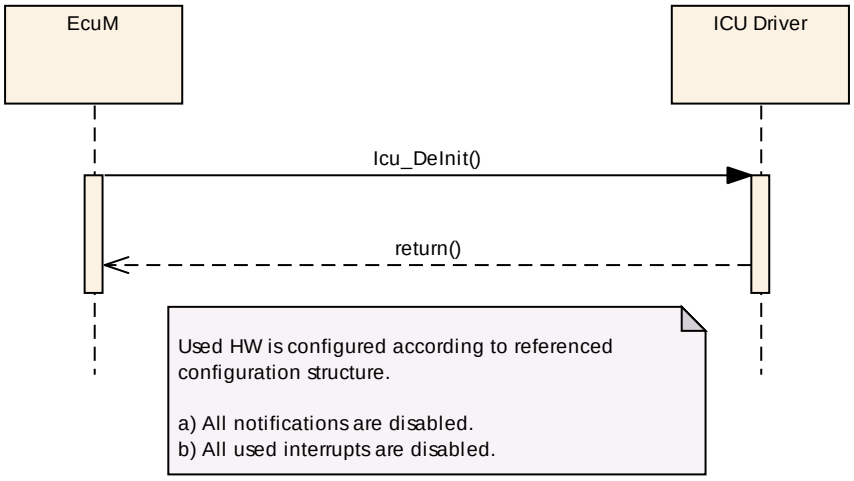


Figure 9.2: De-Initialization of the ICU driver

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

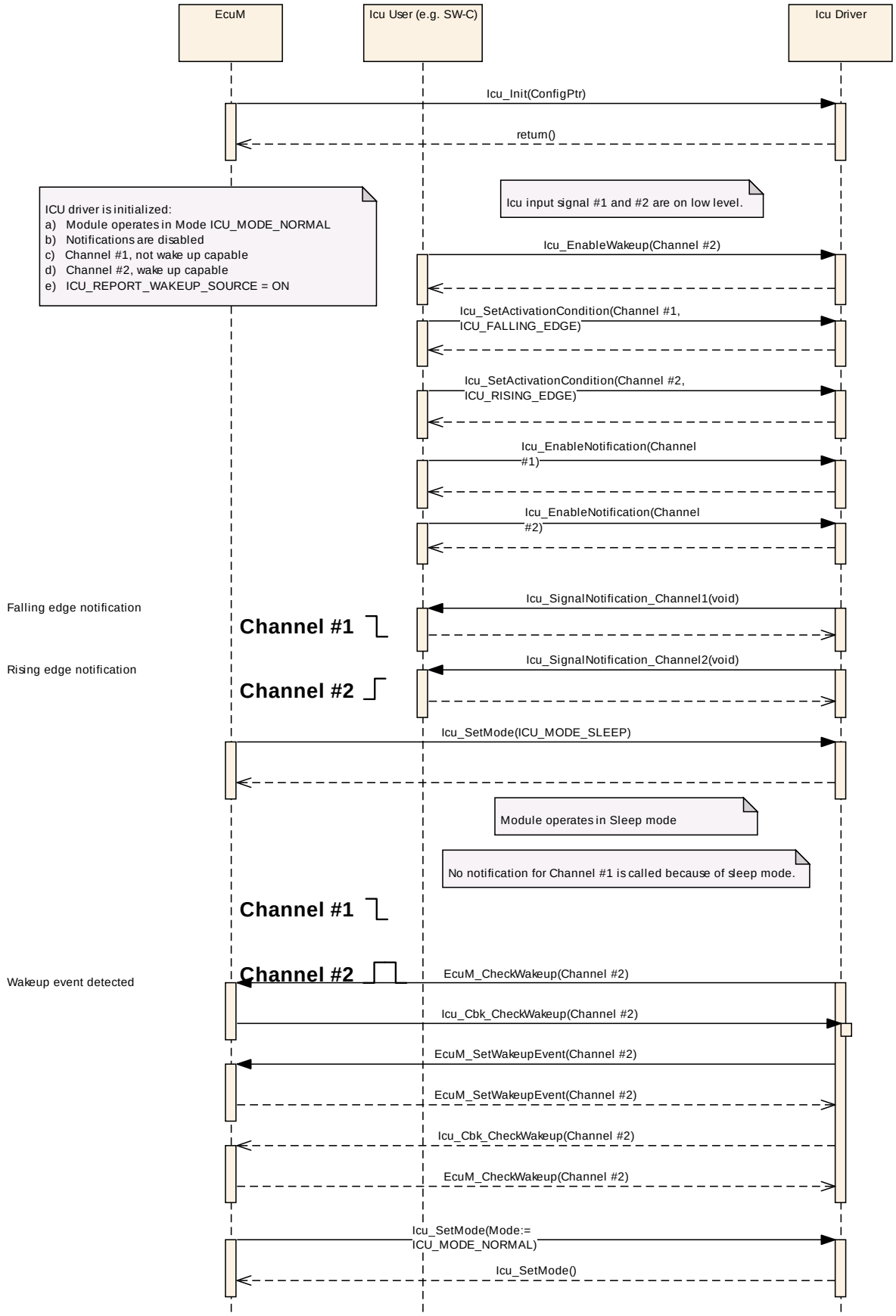


Figure 9.3: Enabled notifications in SLEEP mode

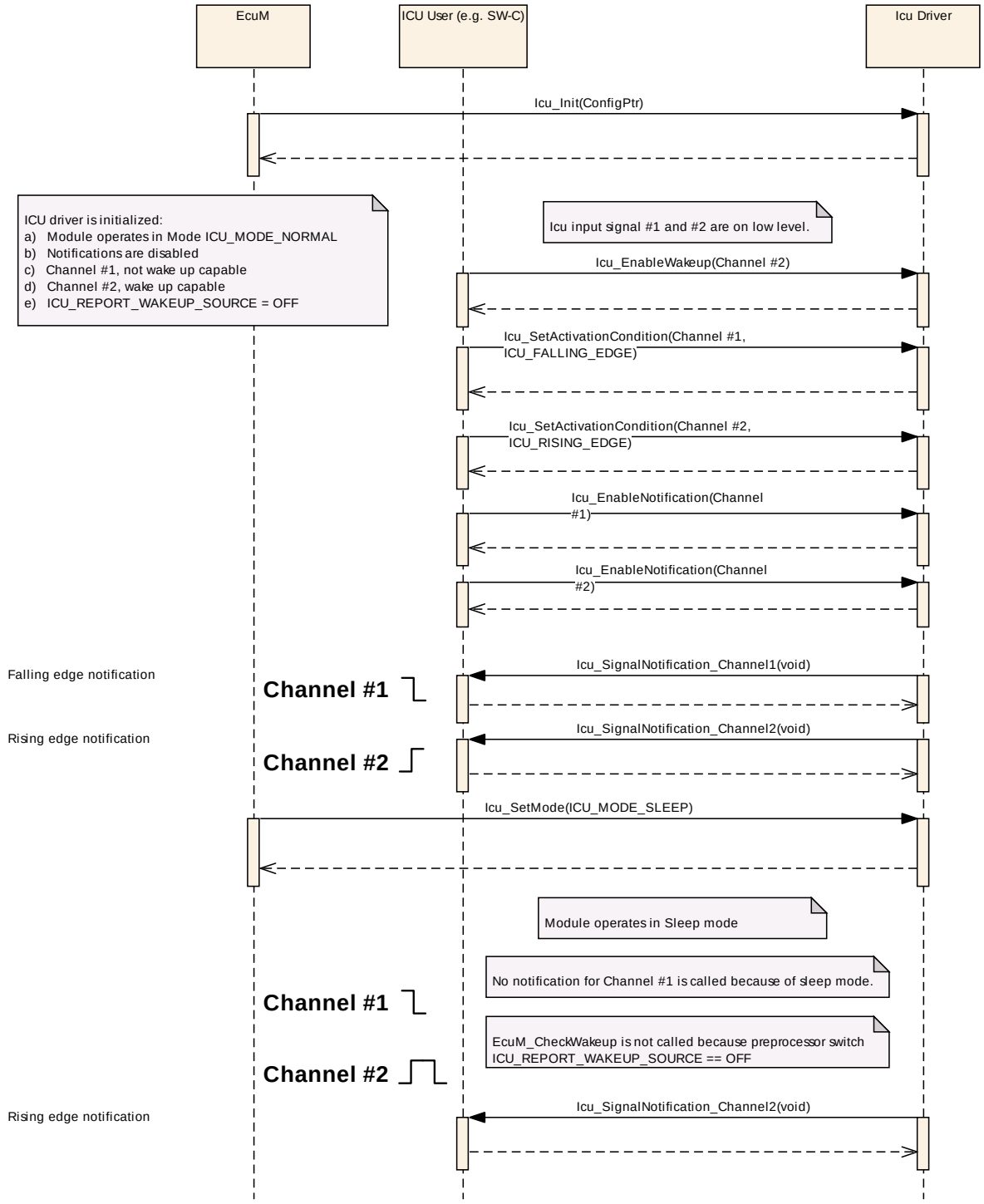


Figure 9.4: Disabled reporting of wakeup sources in SLEEP mode

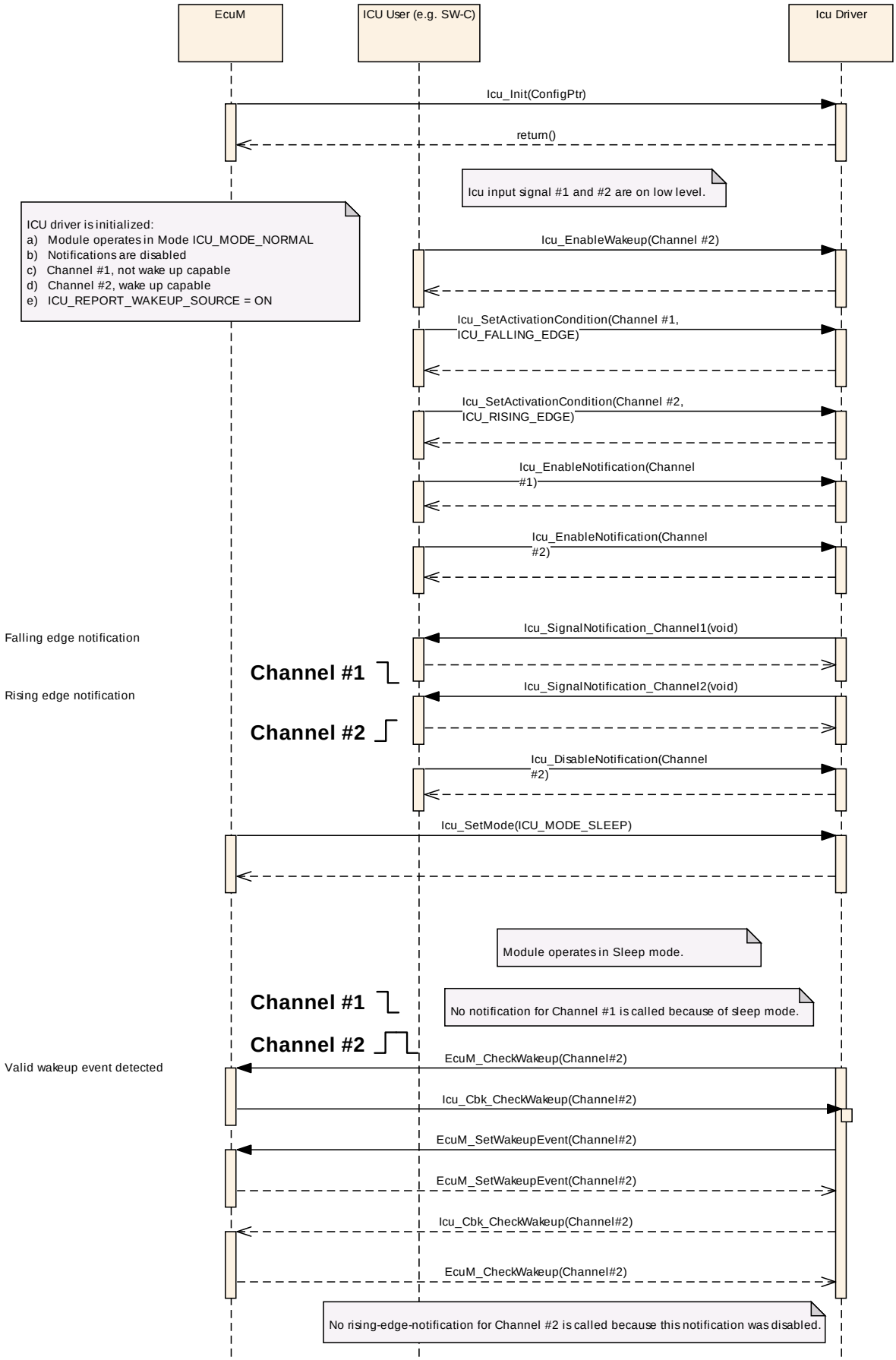


Figure 9.5: Disabled edge notification in SLEEP mode

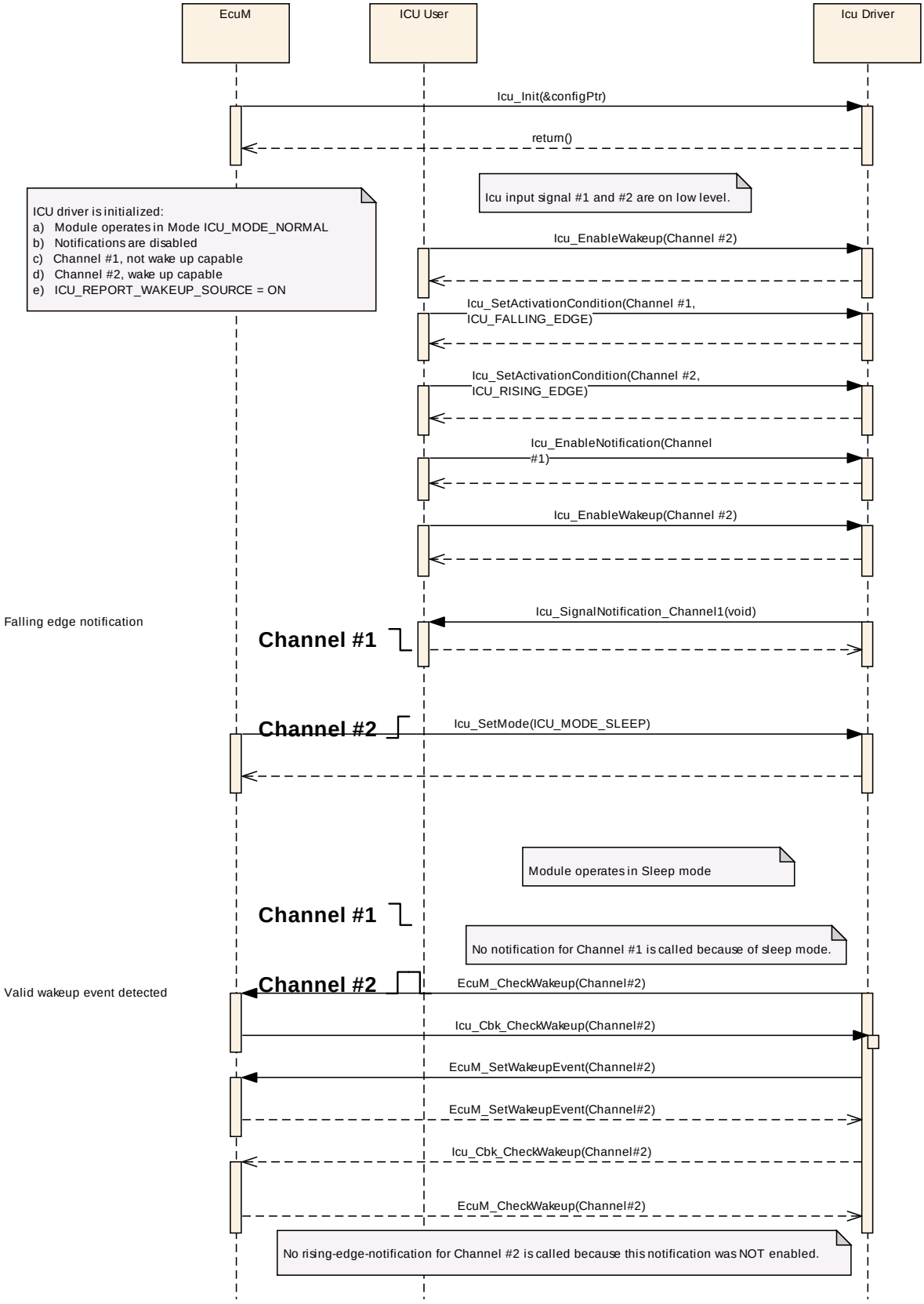


Figure 9.6: Un-Enabled reporting of notifications in SLEEP mode

9.5 Icu_DisableWakeup

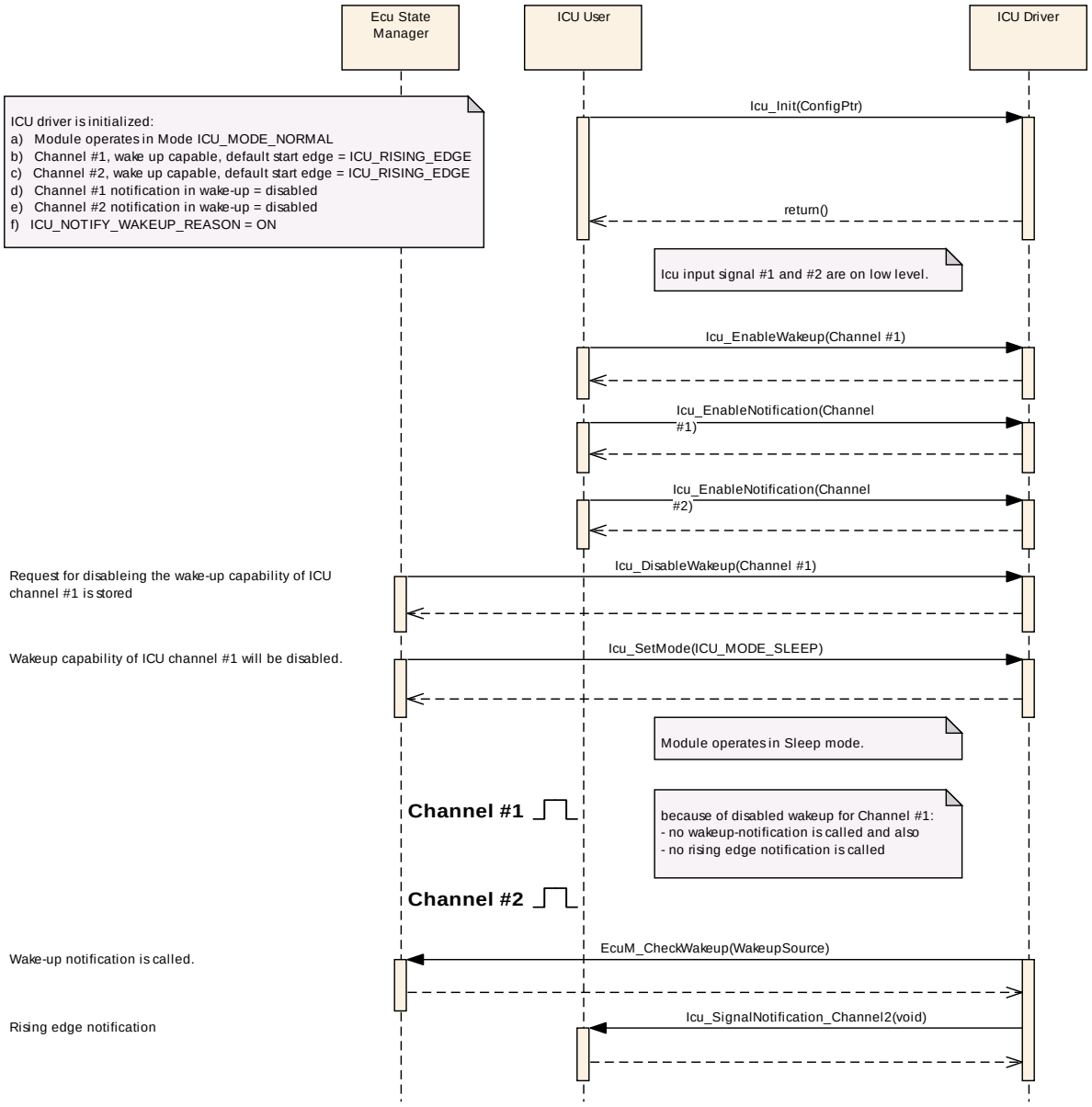


Figure 9.7: Disabling of wakeup-capabilities

9.6 Icu_EnableWakeup

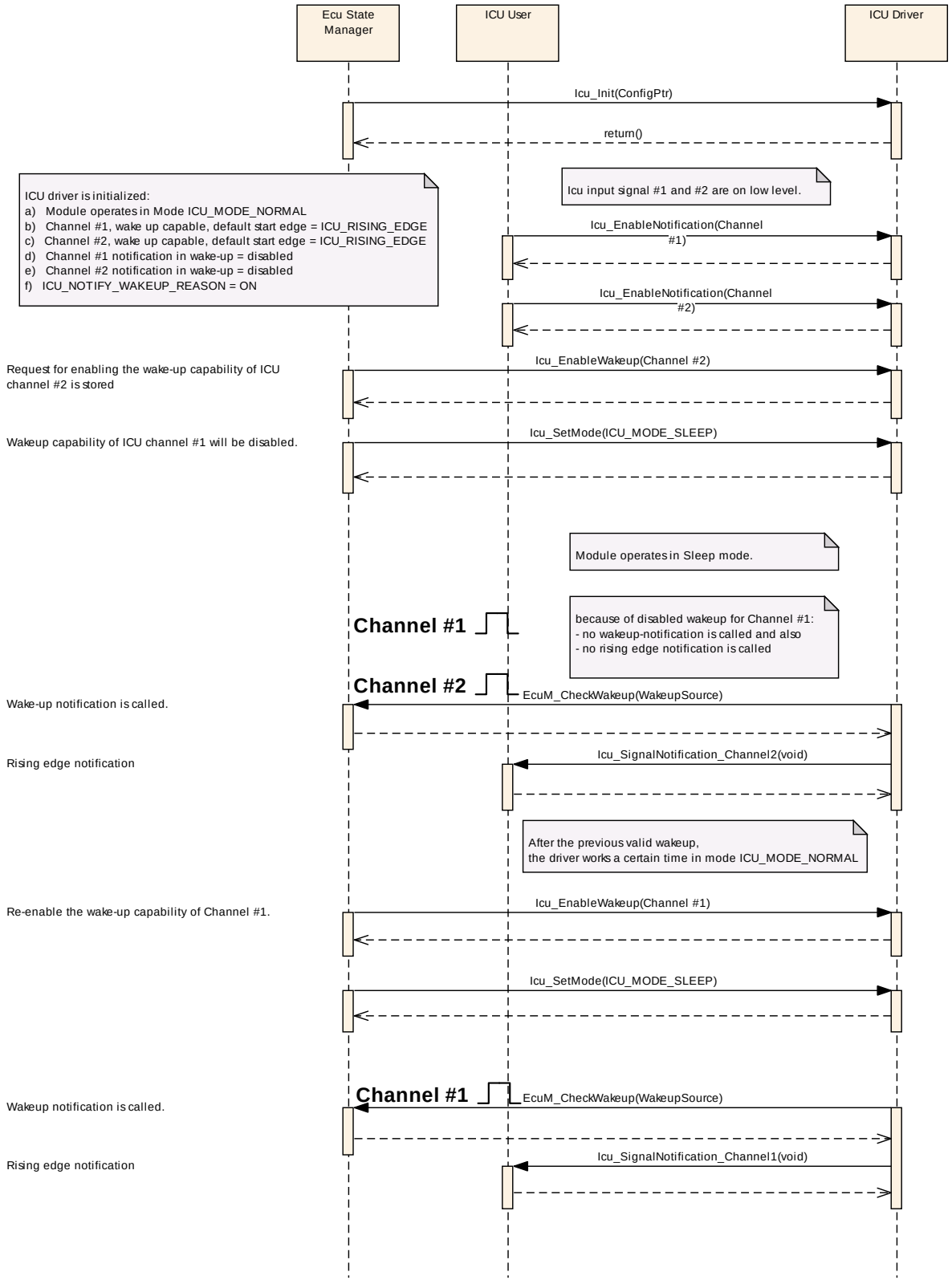


Figure 9.8: Enabling of wakeup-capabilities

9.7 Icu_SetActivationCondition

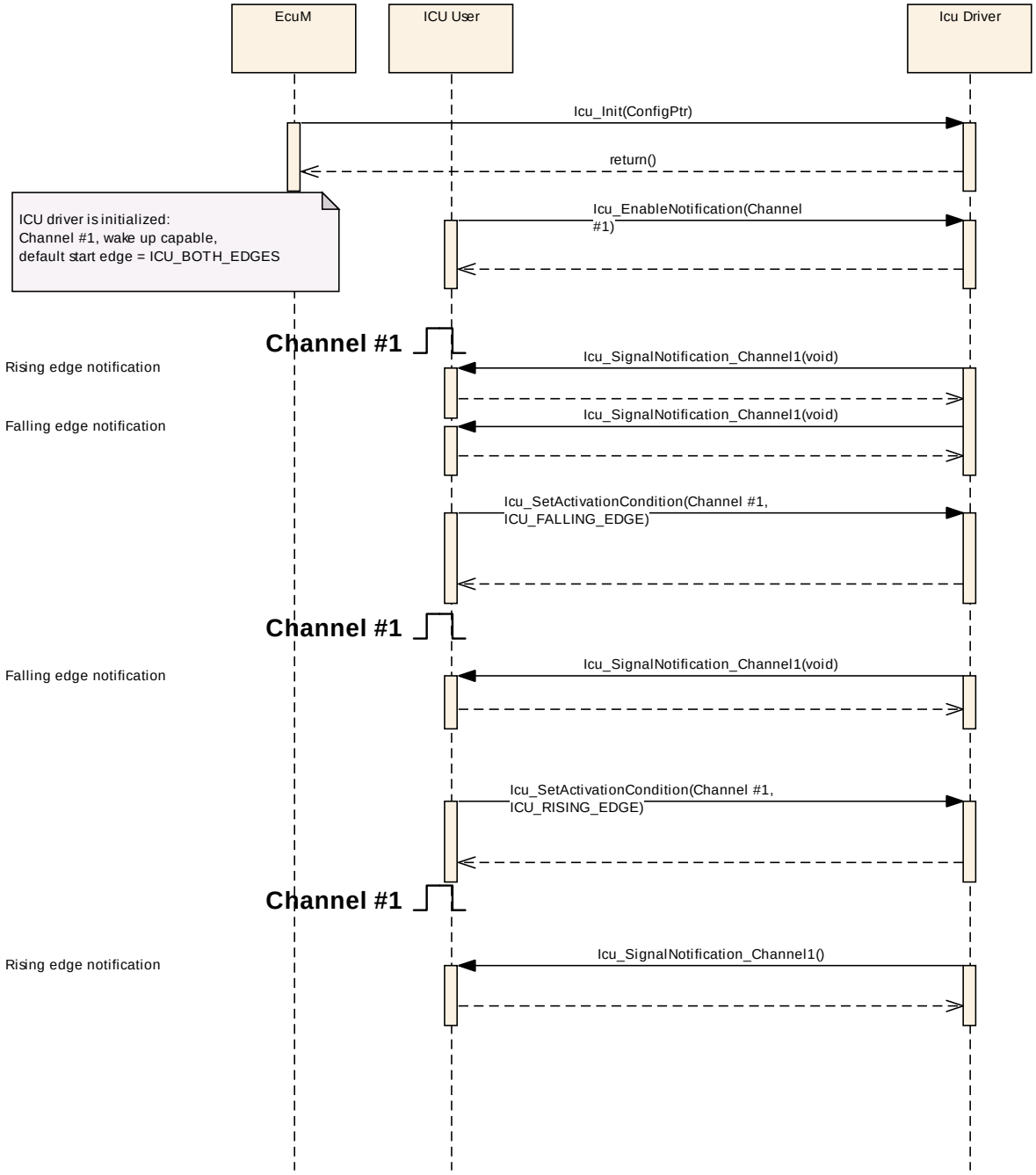


Figure 9.9: Setting up the activation condition for a channel

9.8 Icu_DisableNotification

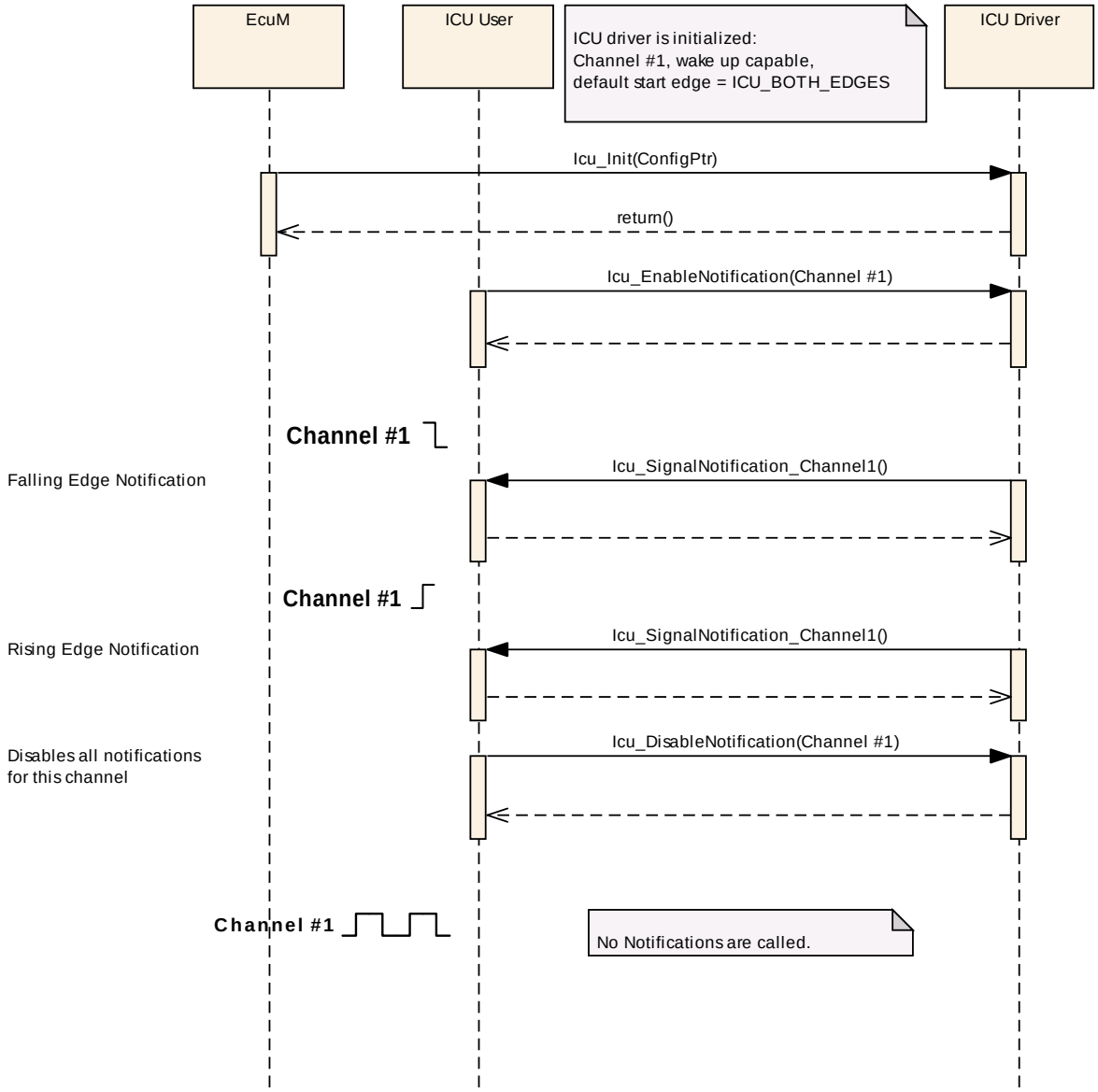


Figure 9.10: Disabling of the notification for a channel

9.9 Icu_EnableNotification

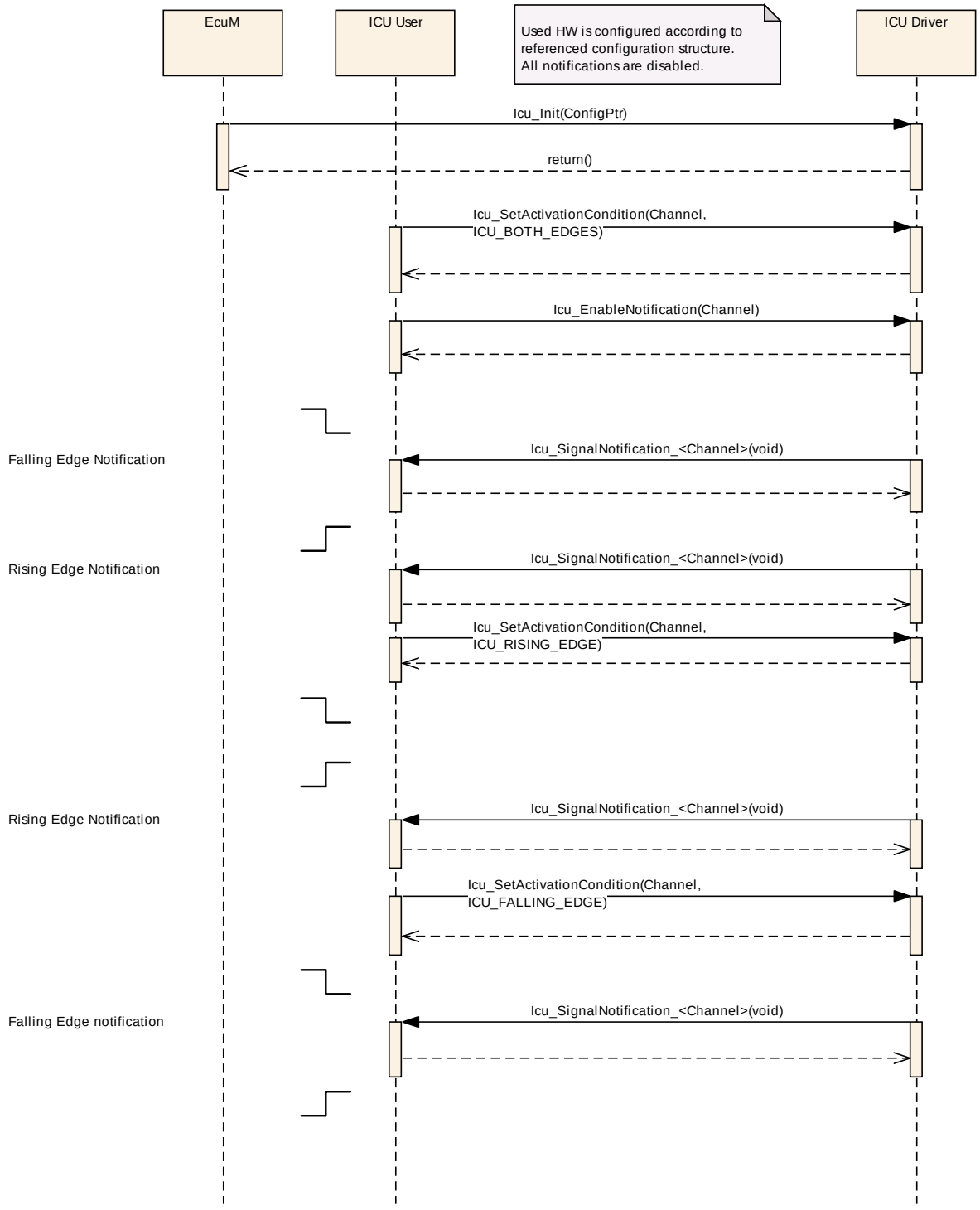


Figure 9.11: Enabling of the edge-notification for a channel

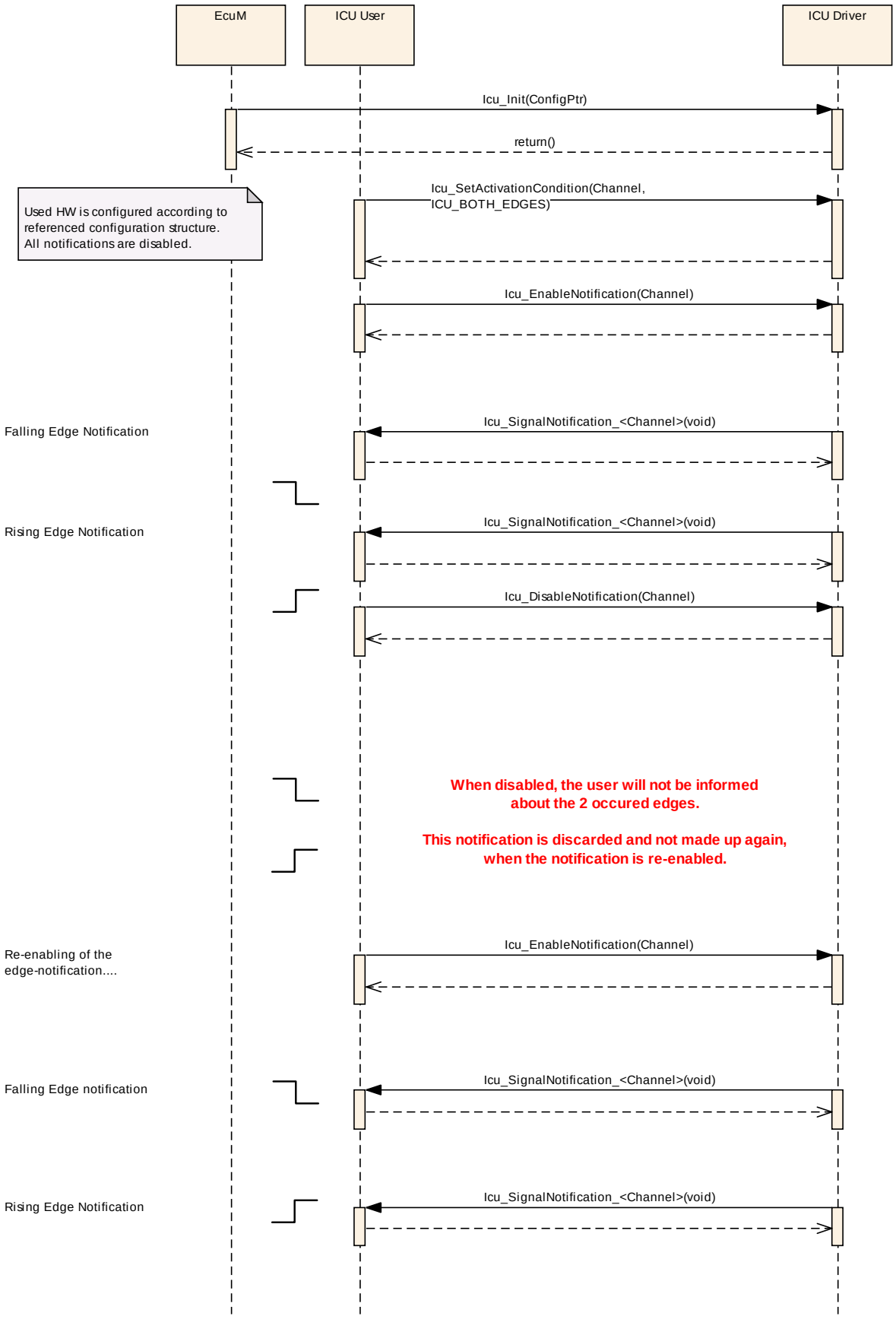

EcuM
ICU User
ICU Driver
Icu_Init(ConfigPtr)
return()
Used HW is configured according to referenced configuration structure. All notifications are disabled.
Icu_SetActivationCondition(Channel, ICU_BOTH_EDGES)
Icu_EnableNotification(Channel)
Falling Edge Notification
Icu_SignalNotification_<Channel>(void)
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

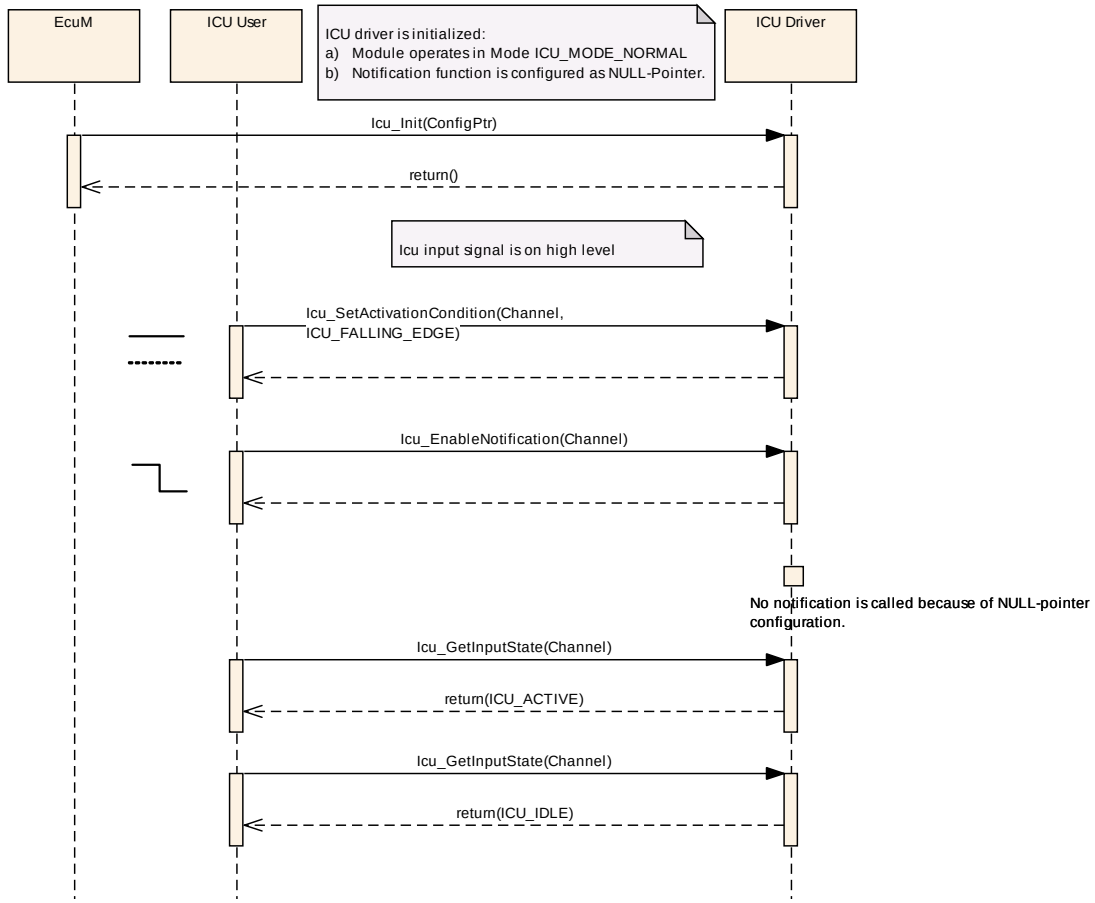


Figure 9.13: Polling of the channel status

9.11 Icu Timestamping

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

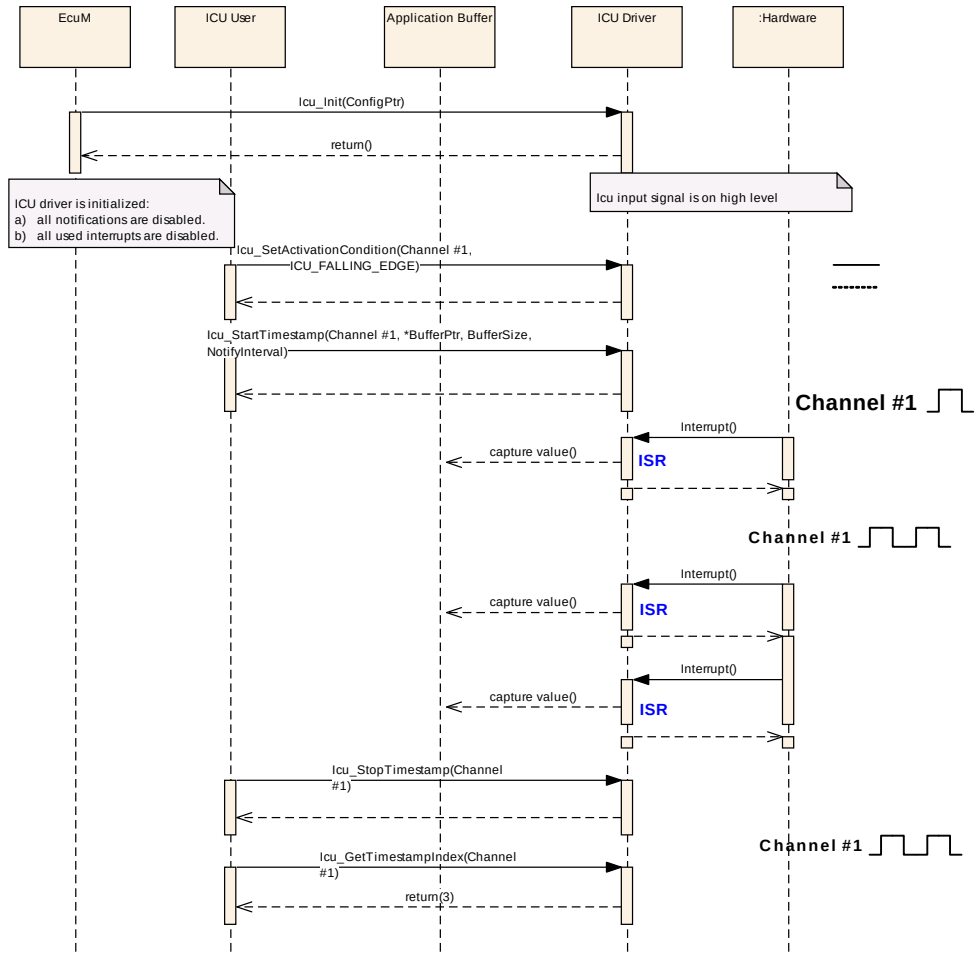


Figure 9.14: Overview of the timestamping functionality of the ICU driver

The Timestamping in general is shown in the following figure:

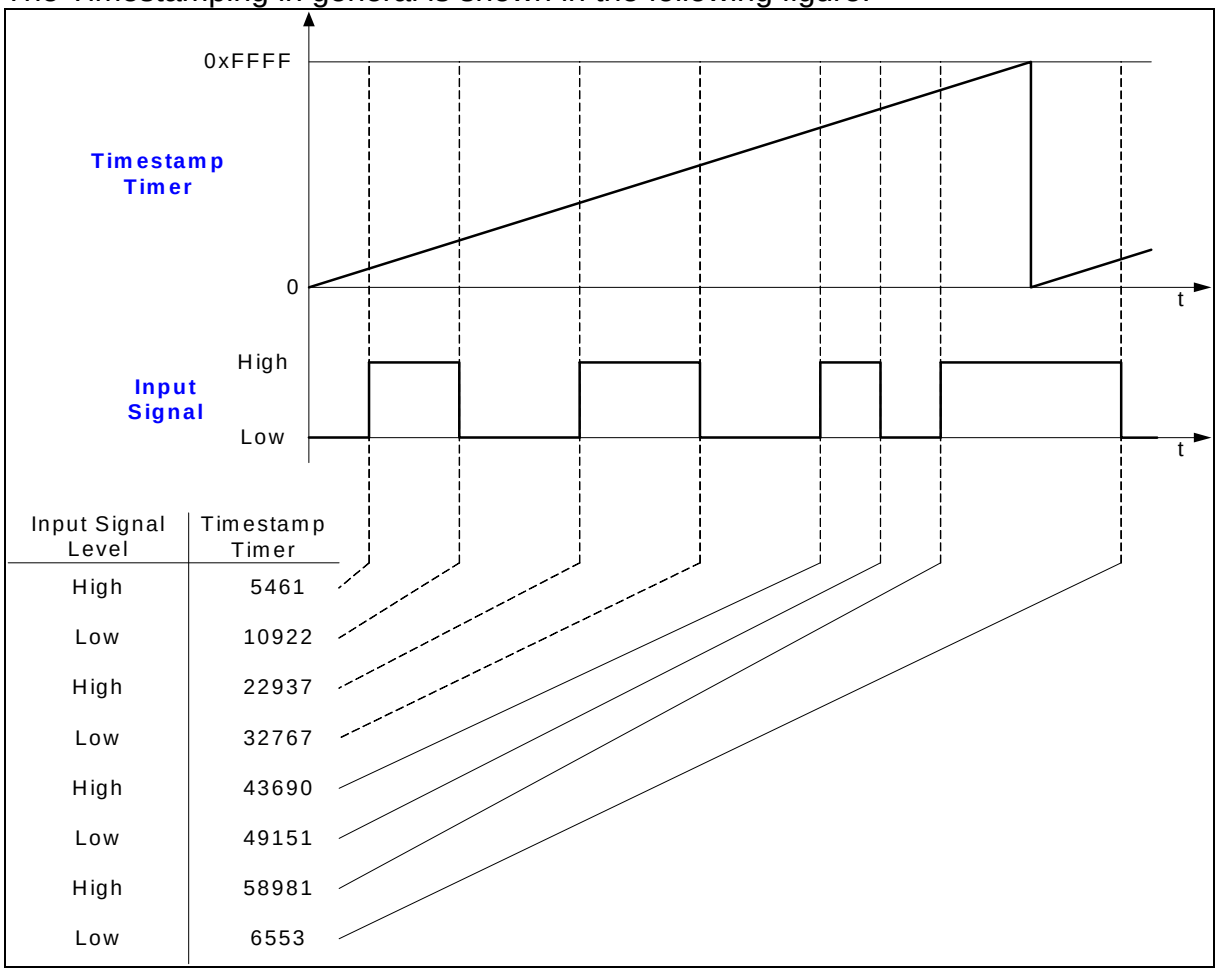


Figure 9.15: Timestamping overview

9.12 Icu Edge Counting

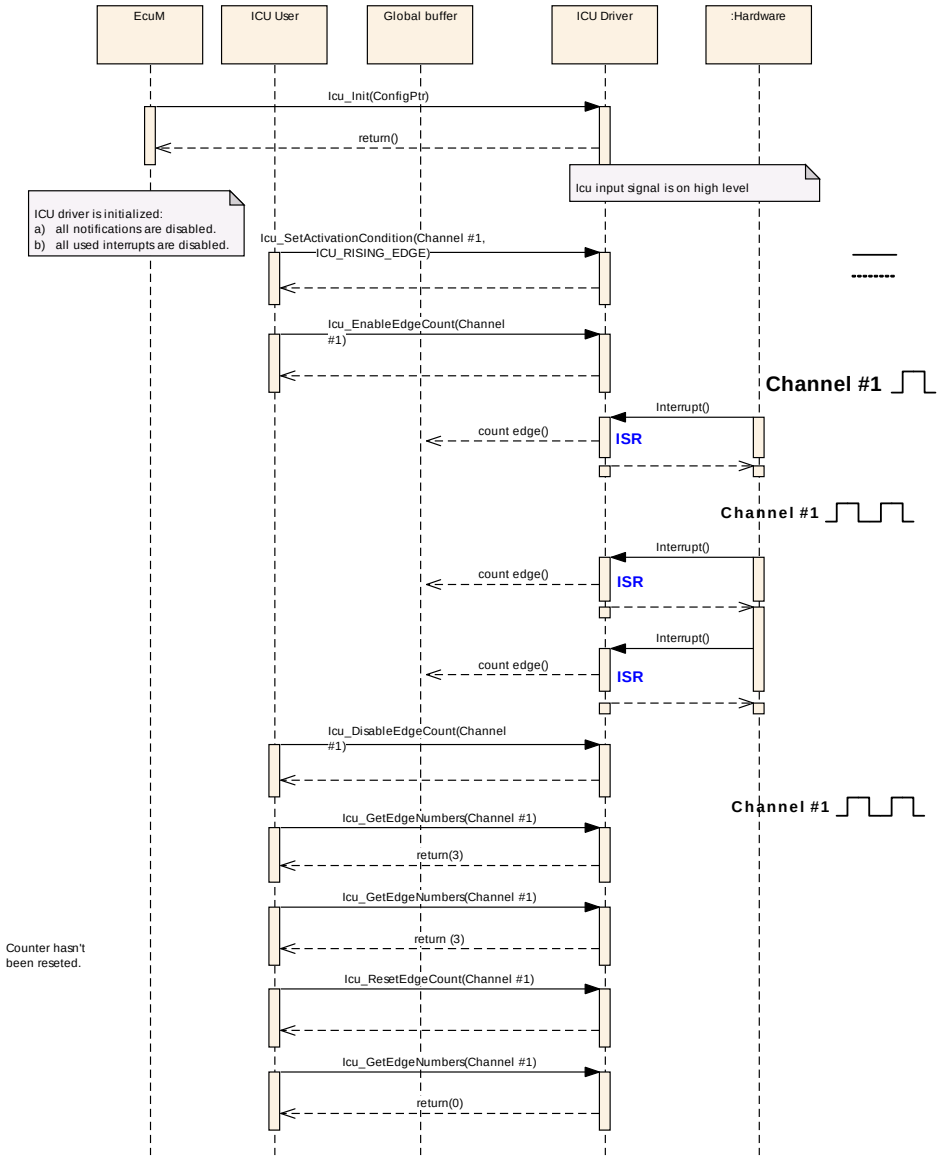


Figure 9.16: Inquire the number of counted edges

9.13 Icu_GetTimeElapsed

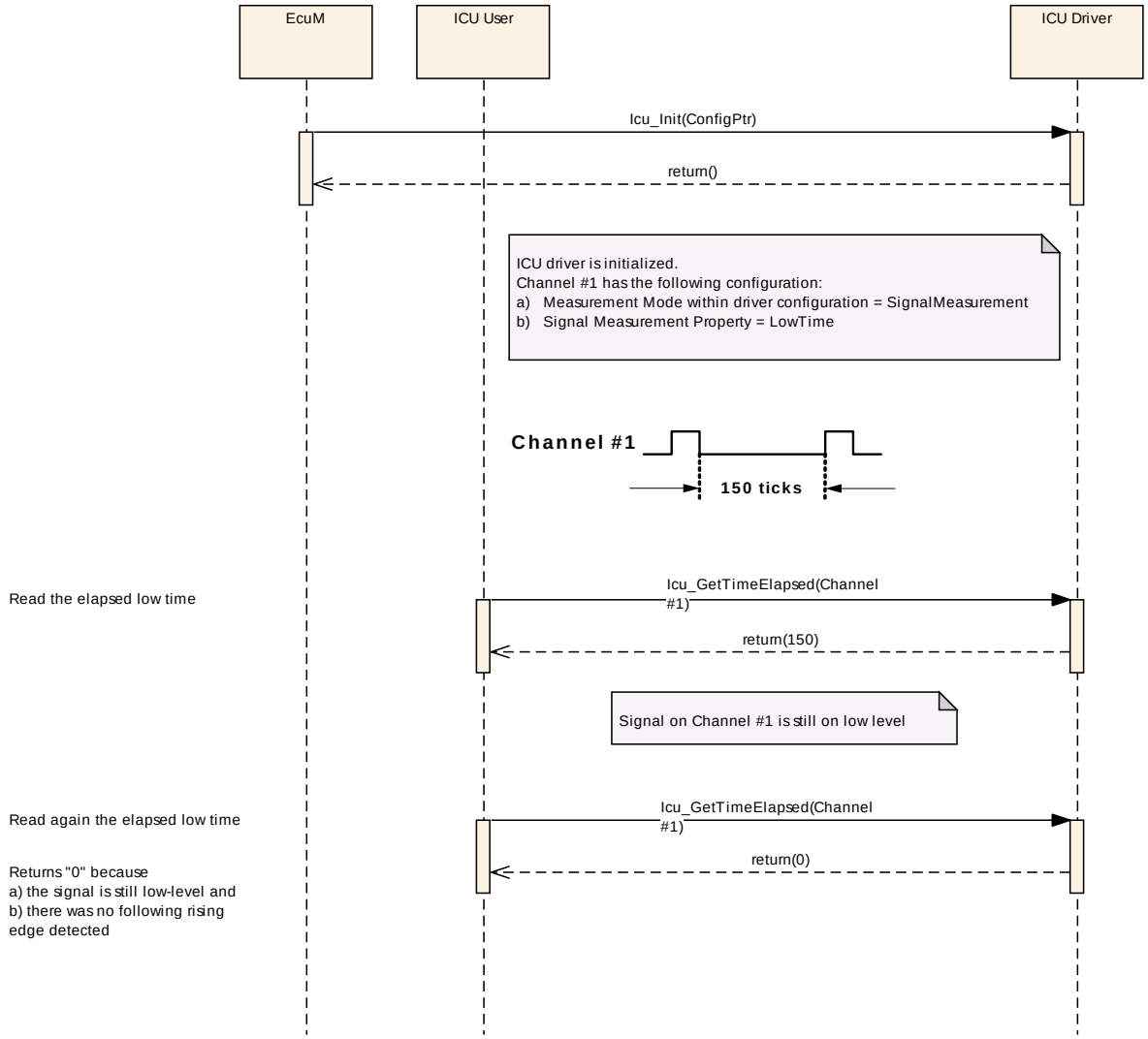


Figure 9.17: Inquire the elapsed level-time of a channel

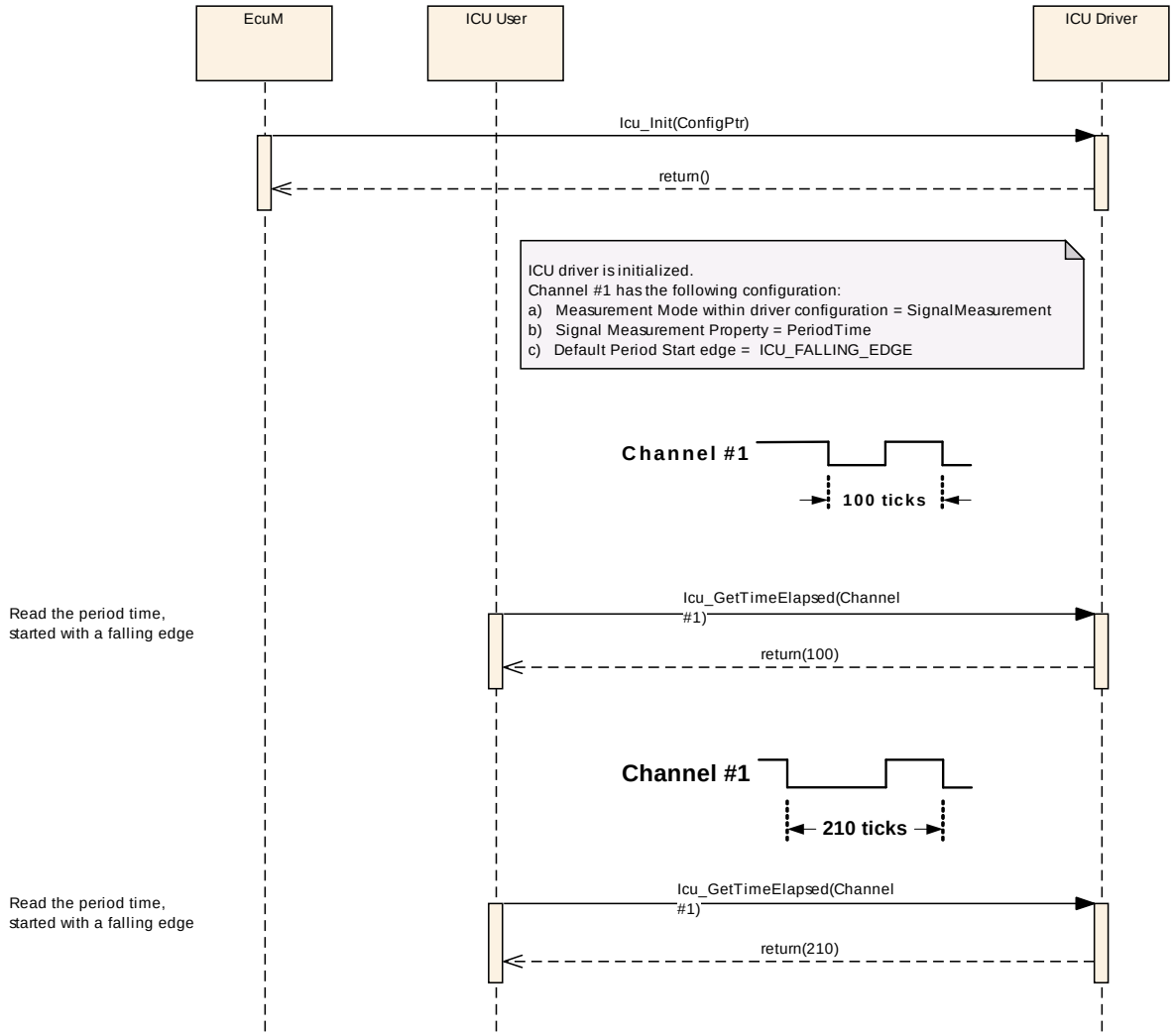


Figure 9.18: Inquire the elapsed period time of a channel

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().

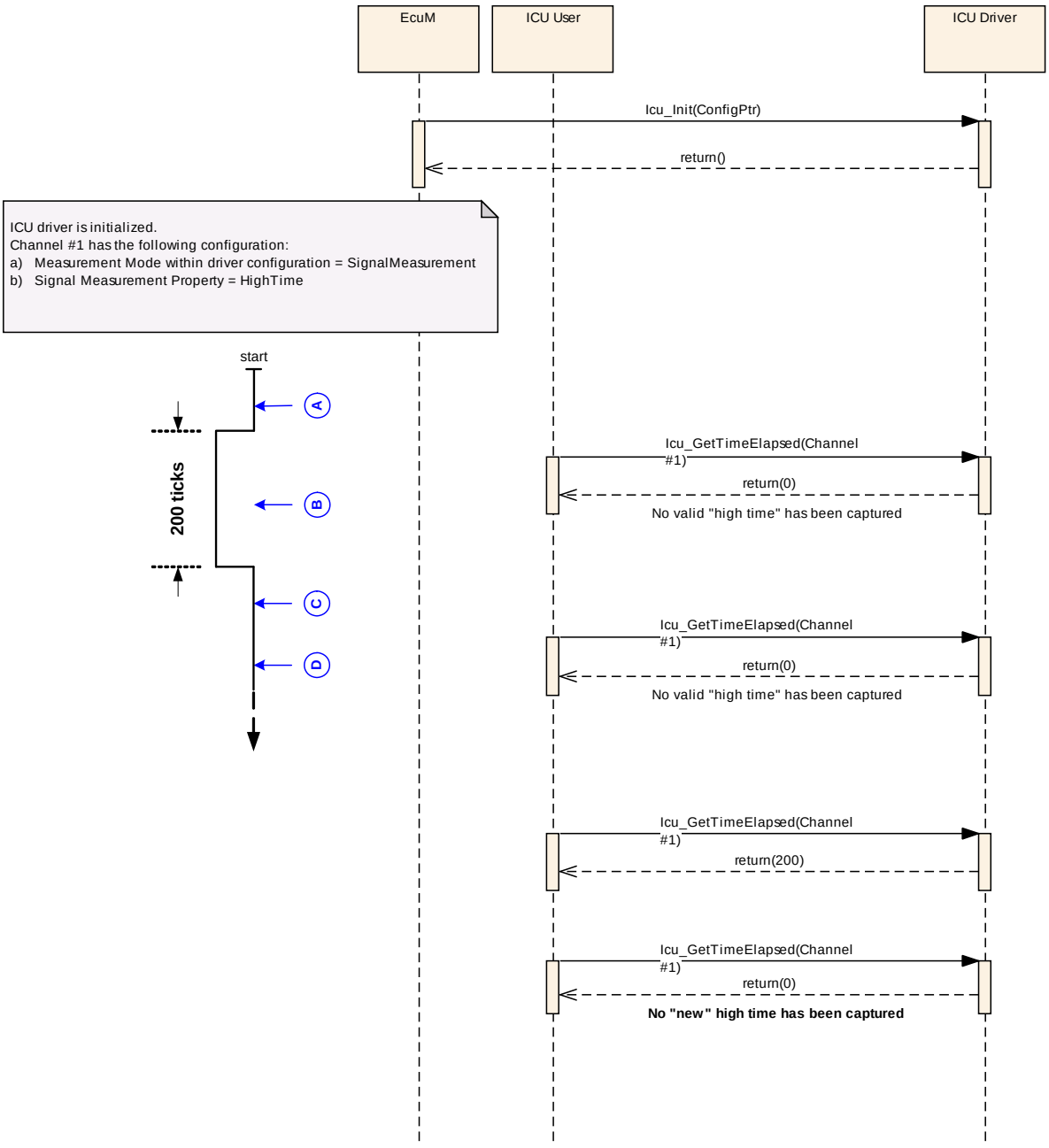
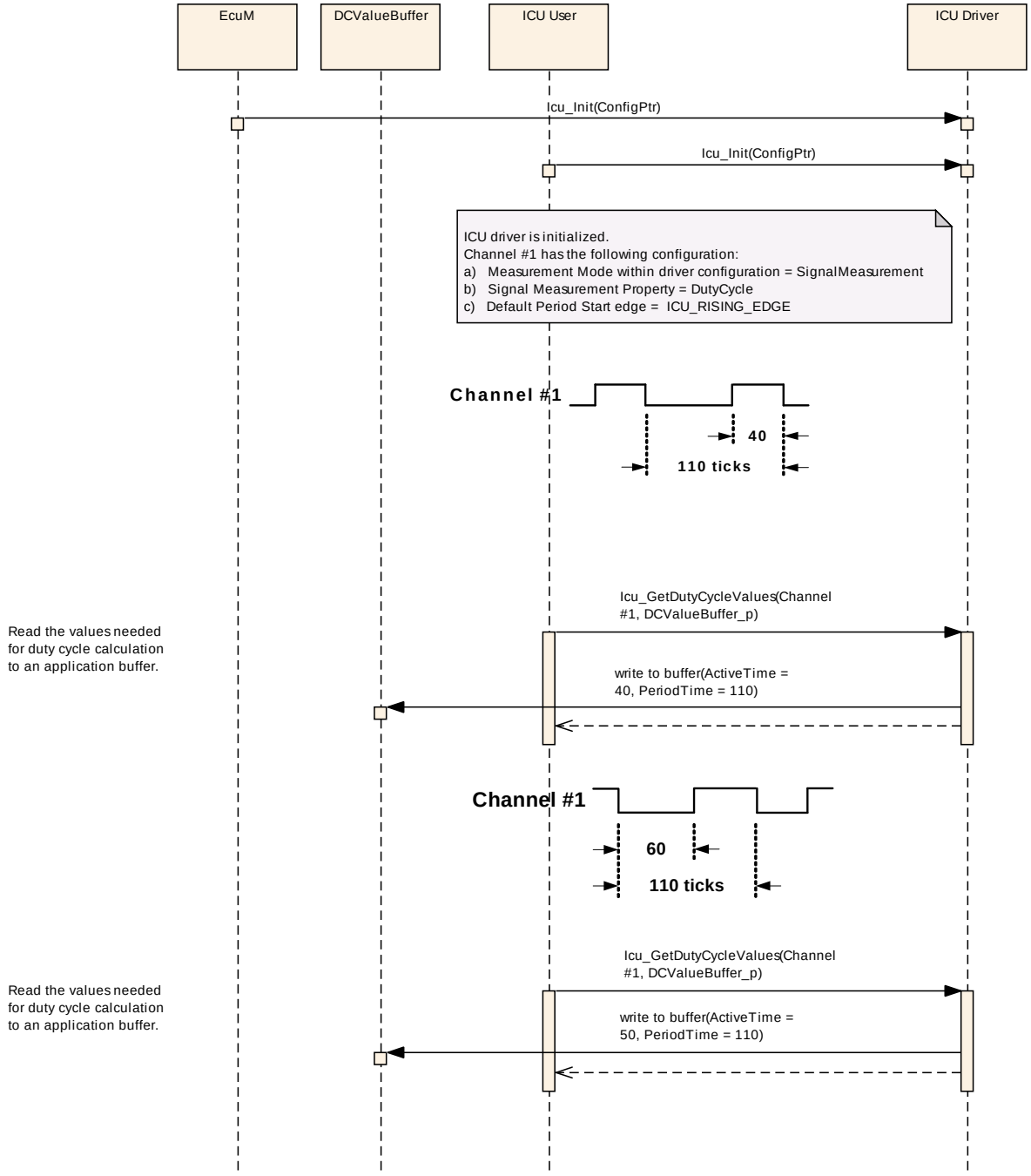


Figure 9.19: Inquire the elapsed high time of a channel

9.14 Icu_GetDutyCycleValues



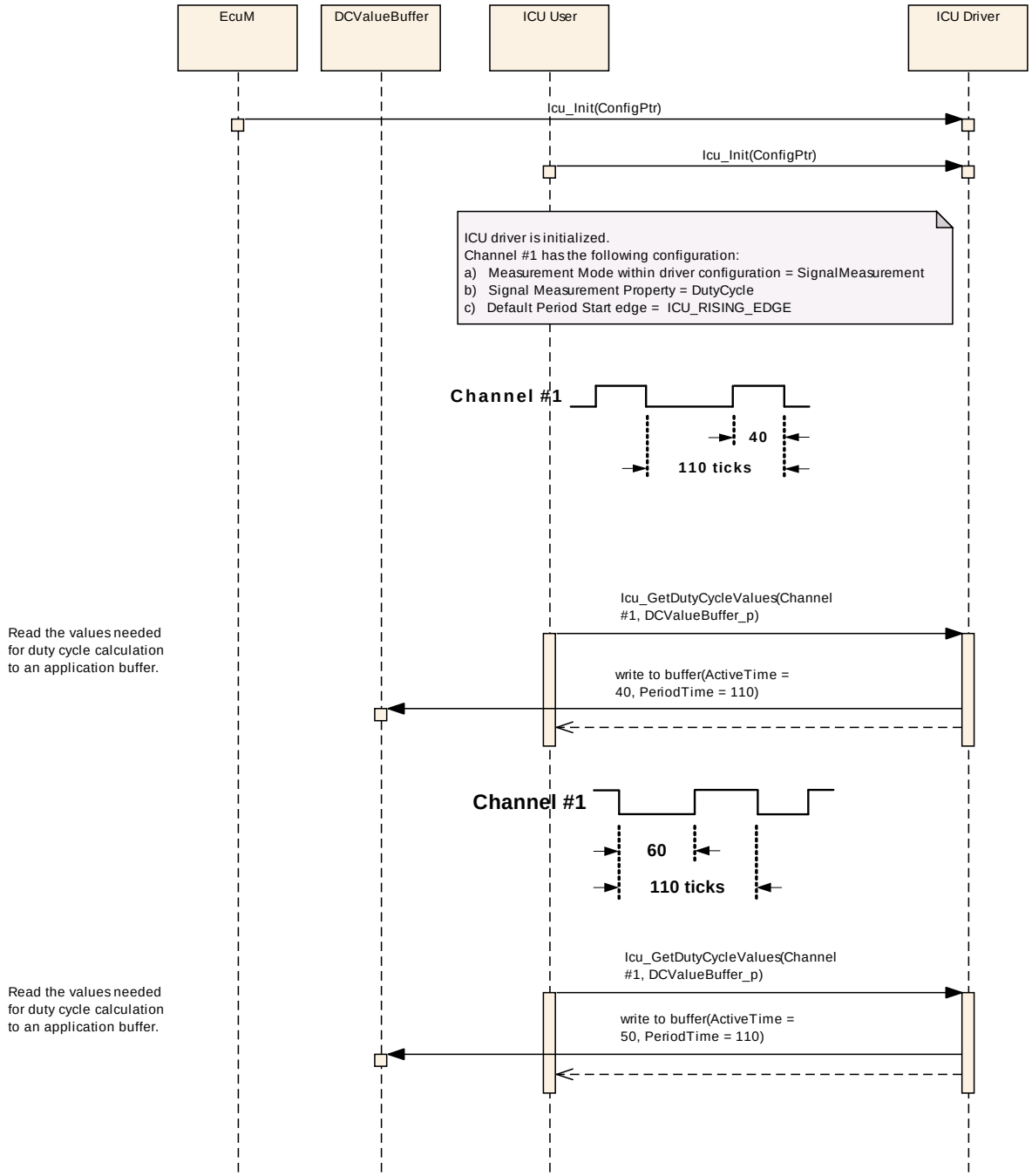


Figure 9.20: Measure the values needed for calculation of duty cycles

9.15 Icu_SignalNotification and Icu_GetInputState

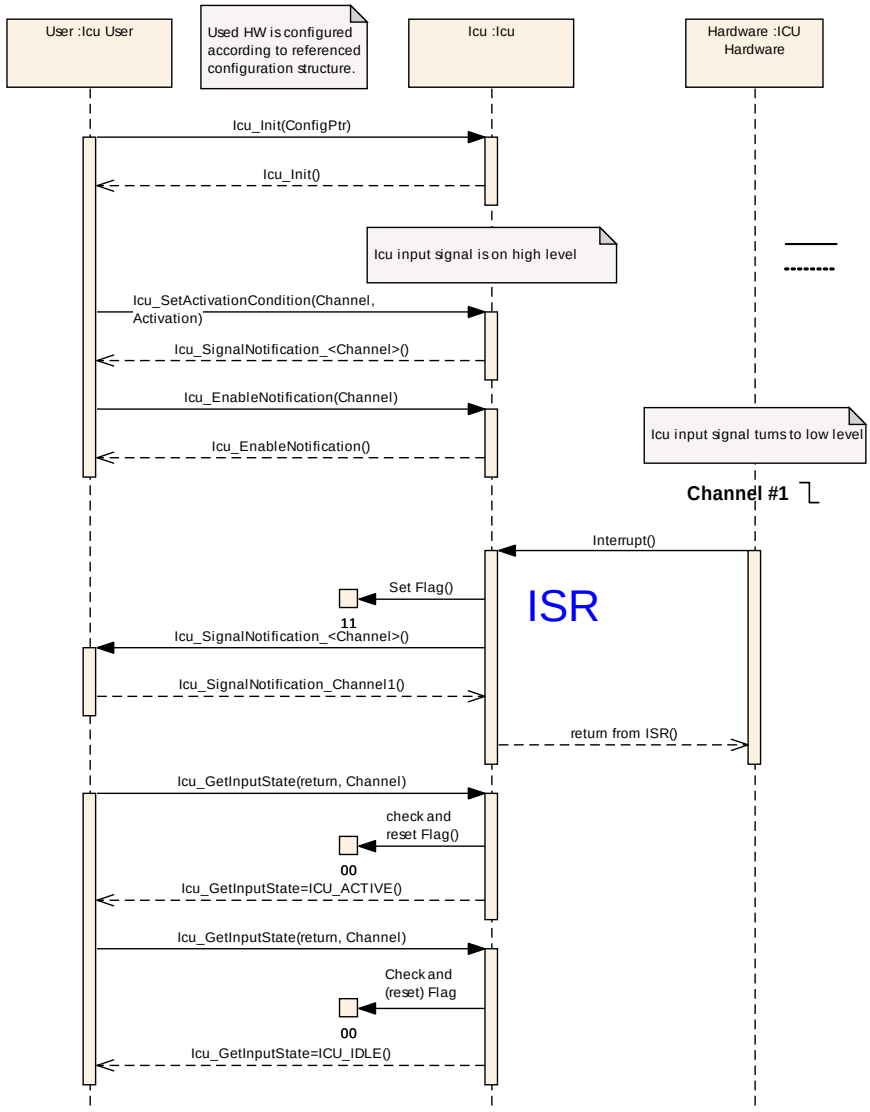


Figure 9.21: Cooperative usage of notification and polling mechanism

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

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

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

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

10.1.1 Configuration and configuration parameters

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

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

10.1.2 Variants

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

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

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

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

10.2 Containers and configuration parameters

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

10.2.1 Variants

[ICU188] ⌈VARIANT-PRE-COMPILE (**P**re **C**ompile): The module ICU shall support a configuration variant pre-compile required for pre-compile time parameters⌋ ()

[ICU189] ⌈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	**ICU026_Conf :**
Container Name	IcuGeneral{General Configuration}
Description	Configuration of general ICU parameters.
Configuration Parameters	

SWS Item	**ICU232_Conf :**		
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: Module		

SWS Item	**ICU221_Conf :**		
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			

SWS Item	**ICU233_Conf :**		
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: Module		

No Included Containers

10.2.4 IcuOptionalApis

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

SWS Item	**ICU234_Conf :**		
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: Module		

SWS Item	**ICU235_Conf :**
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: Module		

SWS Item	**ICU124_Conf :**		
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: Module		

SWS Item	**ICU356_Conf :**		
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: Module		

SWS Item	**ICU236_Conf :**		
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: Module		

SWS Item	**ICU237_Conf :**
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: Module dependency: If IcuSignalMeasurementApi==false this switch shall also be set to false.		

SWS Item	**ICU238_Conf :**		
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: Module		

SWS Item	**ICU239_Conf :**		
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: Module dependency: If IcuSignalMeasurementApi==false this switch shall also be set to false.		

SWS Item	**ICU240_Conf :**		
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: Module		

SWS Item	**ICU241_Conf :**

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: Module		

SWS Item	**ICU242_Conf :**		
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: Module		

SWS Item	**ICU123_Conf :**		
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: Module		

SWS Item	**ICU355_Conf :**		
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: Module		

No Included Containers

10.2.5 IcuChannel

SWS Item	**ICU027_Conf :**
Container Name	IcuChannel{Channel configuration}
Description	Configuration of an individual ICU channel.
Configuration Parameters	

SWS Item	**ICU354_Conf :**		
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: Module		

SWS Item	**ICU222_Conf :**		
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: Module		

SWS Item	**ICU223_Conf :**	
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: Module dependency: The possible measurement modes are depending on the pre-processor switches, which enable/disable optional API services.		

SWS Item	**ICU224_Conf :**
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: Module		

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	**ICU021_Conf :**
Container Name	IcuSignalEdgeDetection{Configuration of Signal Edge Detection}
Description	This container contains the configuration (parameters) in case the measurement mode is "IcuSignalEdgeDetection"
Configuration Parameters	

SWS Item	**ICU225_Conf :**		
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: Module dependency: IcuMeasurementMode		

No Included Containers

10.2.7 IcuSignalMeasurement

SWS Item	**ICU226_Conf :**
Container Name	IcuSignalMeasurement{Configuration of Signal Measurement}
Description	This container contains the configuration (parameters) in case the measurement mode is "IcuSignalMeasurement"

Configuration Parameters

SWS Item	**ICU227_Conf :**		
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_ACTIVE_TIME	The channel is configured for reading the elapsed Signal Active Time	
	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: Module dependency: IcuMeasurementMode, IcuGetDutyCycleValuesApi, IcuGetTimeElapsedApi		

No Included Containers

10.2.8 IcuTimestampMeasurement

SWS Item	**ICU228_Conf :**
Container Name	IcuTimestampMeasurement{Configuration of Timestamp Measurement}
Description	This container contains the configuration (parameters) in case the measurement mode is "IcuTimestamp"
Configuration Parameters	

SWS Item	**ICU229_Conf :**
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: Module dependency: IcuMeasurementMode		

SWS Item	**ICU230_Conf :**		
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: Module dependency: IcuTimestampApi		

No Included Containers

10.2.9 IcuWakeup

SWS Item	**ICU126_Conf :**
Container Name	IcuWakeup{Wakeup Configuration}
Description	This container contains the configuration (parameters) needed to configure a wakeup capable channel
Configuration Parameters	

SWS Item	**ICU231_Conf :**		
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	Reference to [EcuMWakeupSource]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

	dependency: IcuWakeupCapability and IcuReportWakeupSource

No Included Containers

10.2.10 IcuConfigSet

SWS Item	**ICU219_Conf :**
Container Name	IcuConfigSet [Multi Config Container]
Description	This container is the base for a multiple configuration set
Configuration Parameters	

SWS Item	**ICU220_Conf :**		
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: Module		

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

10.3 Published Information

[ICU379] ⌈The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [1] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [4].⌋ ()

Additional module-specific published parameters are listed below if applicable.

[ICU131] ⌈The ICU driver shall describe which other modules (in which versions) are required. This description shall be done by the implementer.⌋ (BSW00384)

11 Not applicable requirements

[ICU380]「These requirements are not applicable to this specification.」(BSW00300, BSW00301, BSW00302, BSW00304, BSW00305, BSW00306, BSW00307, BSW00308, BSW00309, BSW00310, BSW00312, BSW00314, BSW00318, BSW00321, BSW00324, BSW00325, BSW00326, BSW00327, BSW00328, BSW00329, BSW00330, BSW00331, BSW00333, BSW00334, BSW00335, BSW00341, BSW00342, BSW00347, BSW00348, BSW00350, BSW00353, BSW00355, BSW00357, BSW00358, BSW00360, BSW00361, BSW00370, BSW00371, BSW00373, BSW00376, BSW00377, BSW00378, BSW00379, BSW00383, BSW00387, BSW00395, BSW00397, BSW00398, BSW00399, BSW00400, BSW00408, BSW00409, BSW00413, BSW00414, BSW005, BSW006, BSW007, BSW009, BSW010, BSW160, BSW161, BSW162, BSW164, BSW167, BSW168, BSW170, BSW172, BSW00415, BSW00416, BSW00417, BSW00420, BSW00421, BSW00422, BSW00423, BSW00424, BSW00425, BSW00426, BSW00427, BSW00428, BSW00429, BSW00431, BSW00432, BSW00433, BSW00434, BSW00437, BSW00439, BSW00440, BSW00441, BSW12068, BSW12077, BSW12092, BSW12265, BSW12463)