

|                                   |                             |
|-----------------------------------|-----------------------------|
| <b>Document Title</b>             | Specification of OCU Driver |
| <b>Document Owner</b>             | AUTOSAR                     |
| <b>Document Responsibility</b>    | AUTOSAR                     |
| <b>Document Identification No</b> | 615                         |
| <b>Document Status</b>            | published                   |
| <b>Part of AUTOSAR Standard</b>   | Classic Platform            |
| <b>Part of Standard Release</b>   | R21-11                      |

| <b>Document Change History</b> |                |                                  |  |
|--------------------------------|----------------|----------------------------------|--|
| <b>Date</b>                    | <b>Release</b> | <b>Changed by</b>                | <b>Change Description</b>  |
| 2021-11-25                     | R21-11         | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>Removed the Ocu_ConfigType data structure specification</li> </ul>  |
| 2020-11-30                     | R20-11         | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>Added missing specification for enum values</li> <li>Updated return value for Ocu_StartChannel</li> <li>Minor changes in Error Tables</li> </ul>  |
| 2019-11-28                     | R19-11         | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>Error OCU_E_BUSY classified as Runtime Error.</li> <li>Added reference to OcuHWSpecificSettings in OcuChannel. Multiplicity of OcuHWSpecificSettings changed.</li> <li>Introduced MCAL Multicore Distribution</li> <li>Changed Document Status from Final to published</li> </ul> |
| 2018-10-31                     | 4.4.0          | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>OcuGroup removed (ECUC_Ocu_00161, ECUC_Ocu_00162, ECUC_Ocu_00163)</li> <li>Updated Header File Structure</li> <li>Multicore feature (SWS_Ocu_00170, SWS_Ocu_CONSTR_00001, SWS_Ocu_CONSTR_00002)</li> </ul>  |

| Document Change History |         |                                  |   |
|-------------------------|---------|----------------------------------|---|
| Date                    | Release | Changed by                       | Change Description  |
| 2017-12-08              | 4.3.1   | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>OcuGroup set to obsolete (ECUC_Ocu_00161, ECUC_Ocu_00162 and ECUC_Ocu_00163)</li> <li>Renamed “default error detection” to “development error detection”</li> </ul>  |
| 2016-11-30              | 4.3.0   | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>Removed SWS_Ocu_00134 and SWS_Ocu_00135</li> <li>Renamed “SRS_BSW_000386” to “SRS_BSW_00386”</li> <li>Removed SRS_BSW_00157, SRS_BSW_00326, SRS_BSW_00329, SRS_BSW_00338, SRS_BSW_00355, SRS_BSW_00370, SRS_BSW_00376, SRS_BSW_00434</li> <li>Removed SRS_BSW_0431</li> <li>Changed “SRS_SPAL12448” to “SRS_SPAL_12448”</li> </ul> |
| 2015-07-31              | 4.2.2   | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>DET has been renamed.</li> <li>SWS_Ocu_00041 and SWS_Ocu_00042 requirements are removed.</li> <li>OCU_E_PARAM_CONFIG is removed. Added OCU_E_INIT_FAILED</li> <li>Invalid requirement IDs: Updated SWS_Ocu_156, SWS_Ocu_169</li> </ul>   |
| 2014-10-31              | 4.2.1   | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"> <li>Set the postBuildVariantValue and postBuildVariantMultiplicity to false and also</li> <li>set the valueConfigClass and the multiplicityConfigClass for all variants to preCompile.</li> <li>Removal of automatically supported BSW requirement. Reference to SWS_BSW_00380 is removed.</li> </ul>                                  |

| <b>Document Change History</b> |                |                                  |   |
|--------------------------------|----------------|----------------------------------|---|
| <b>Date</b>                    | <b>Release</b> | <b>Changed by</b>                | <b>Change Description</b>   |
| 2013-10-31                     | 4.1.2          | AUTOSAR<br>Release<br>Management | <ul style="list-style-type: none"><li>• Minor update of the document structure</li><li>• Editorial changes</li><li>• Removed chapter(s) on change documentation</li></ul> |
| 2013-03-15                     | 4.1.1          | AUTOSAR<br>Administration        | <ul style="list-style-type: none"><li>• Initial Release</li></ul>   |

## Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Table of Contents

|       |  |    |
|-------|--|----|
| 1     | Introduction and functional overview ..... | 7  |
| 2     | Acronyms and abbreviations.....            | 8  |
| 3     | Related documentation .....                | 9  |
| 3.1   | Input documents.....                       | 9  |
| 4     | Constraints and assumptions.....           | 10 |
| 4.1   | Assumptions.....                           | 10 |
| 4.1.1 | Clock.....                                 | 10 |
| 4.1.2 | Resources .....                            | 10 |
| 4.1.3 | Counting and comparing .....               | 10 |
| 4.2   | Limitations .....                          | 11 |
| 4.3   | Applicability to car domains .....         | 11 |
| 5     | Dependencies to other modules .....        | 12 |
| 5.1   | File structure .....                       | 12 |
| 5.1.1 | Code file structure .....                  | 12 |
| 5.1.2 | Header file structure .....                | 12 |
| 6     | Requirements traceability .....            | 13 |
| 7     | Functional specification.....              | 22 |
| 7.1   | General behavior.....                      | 22 |
| 7.2   | Version check.....                         | 23 |
| 7.2.1 | Background & Rationale .....               | 23 |
| 7.3   | Time Unit Ticks .....                      | 23 |
| 7.3.1 | Background & Rationale .....               | 23 |
| 7.3.2 | Requirements .....                         | 23 |
| 7.4   | Error classification.....                  | 24 |
| 7.4.1 | Development Errors .....                   | 24 |
| 7.4.2 | Runtime Errors .....                       | 25 |
| 7.4.3 | Transient Faults.....                      | 25 |
| 7.4.4 | Production Errors .....                    | 25 |
| 7.4.5 | Extended Production Errors .....           | 25 |
| 7.5   | Error Detection .....                      | 25 |
| 7.6   | Error Notification .....                   | 25 |
| 7.7   | Debug Support.....                         | 26 |
| 8     | API specification.....                     | 27 |
| 8.1   | Imported types .....                       | 27 |
| 8.2   | Type definitions .....                     | 27 |
| 8.2.1 | Ocu_ChannelType .....                      | 27 |
| 8.2.2 | Ocu_ValueType.....                         | 27 |
| 8.2.3 | Ocu_PinStateType .....                     | 28 |
| 8.2.4 | Ocu_PinActionType.....                     | 28 |
| 8.2.5 | Ocu_ConfigType .....                       | 28 |
| 8.2.6 | Ocu_ReturnType .....                       | 29 |

|        |   |    |
|--------|---|----|
| 8.3    | Function definitions .....                                | 29 |
| 8.3.1  | Ocu_Init .....  | 29 |
| 8.3.2  | Ocu_DeInit.....   | 31 |
| 8.3.3  | Ocu_StartChannel.....                                     | 33 |
| 8.3.4  | Ocu_StopChannel.....                                      | 34 |
| 8.3.5  | Ocu_SetPinState.....                                      | 35 |
| 8.3.6  | Ocu_SetPinAction .....                                    | 37 |
| 8.3.7  | Ocu_GetCounter .....                                      | 38 |
| 8.3.8  | Ocu_SetAbsoluteThreshold .....                            | 39 |
| 8.3.9  | Ocu_SetRelativeThreshold .....                            | 43 |
| 8.3.10 | Ocu_DisableNotification .....                             | 46 |
| 8.3.11 | Ocu_EnableNotification .....                              | 47 |
| 8.3.12 | Ocu_GetVersionInfo .....                                  | 48 |
| 8.4    | Callback notifications .....                              | 49 |
| 8.5    | Scheduled functions.....                                  | 49 |
| 8.6    | Expected Interfaces .....                                 | 49 |
| 8.6.1  | Mandatory Interfaces.....                                 | 49 |
| 8.6.2  | Optional Interfaces .....                                 | 50 |
| 8.6.3  | Configurable interfaces .....                             | 50 |
| 9      | Sequence and Timing diagrams .....                        | 52 |
| 9.1    | Initialization .....                                      | 52 |
| 9.2    | De-initialization.....                                    | 53 |
| 9.3    | Using the Ocu Notifications .....                         | 54 |
| 9.4    | Ocu_SetPinState.....                                      | 55 |
| 9.5    | Ocu_SetPinAction.....                                     | 56 |
| 9.6    | Setting a new compare threshold .....                     | 56 |
| 10     | Configuration specification .....                         | 57 |
| 10.1   | How to read this chapter.....                             | 57 |
| 10.1.1 | Configuration and configuration parameters .....          | 57 |
| 10.1.2 | Containers.....   | 57 |
| 10.1.3 | Specification template for configuration parameters ..... | 58 |
| 10.2   | Containers and configuration parameters .....             | 58 |
| 10.2.1 | Ocu.....  | 58 |
| 10.2.2 | OcuGeneral.....   | 60 |
| 10.2.3 | OcuConfigurationOfOptionalApis .....                      | 62 |
| 10.2.4 | OcuConfigSet.....   | 65 |
| 10.2.5 | OcuChannel .....  | 67 |
| 10.2.6 | OcuHWSpecificSettings.....                                | 73 |
| 10.3   | Published Information .....                               | 75 |
| 11     | Not applicable requirements .....                         | 76 |

## 1 Introduction and functional overview

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

Each OCU software channel is linked to a hardware OCU peripheral which belongs to the microcontroller. An output pin can be optionally attached to this channel.

The driver provides functions for initialization and control of the microcontroller internal OCU functionality (Output Compare Unit). The OCU driver allows comparing and acting automatically when the value of a counter matches a defined threshold. The OCU driver provides services and configuration parameters for:

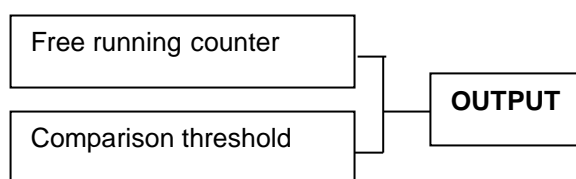
- Starting and stopping a comparison process
- Setting comparison threshold
- Enabling and disabling notification mechanisms
- Getting counter values
- Changing output pin states
- Triggering some hardware resources (ADC, DMA) if available.

□

The tick duration of a channel counter depends on the channel specific settings (part of OCU driver) as well as on the system clock and settings of the clock tree controlled by the MCU module. The tick duration is not limited by this specification.

Some microcontrollers don't have a dedicated OCU hardware cell, but instead a generic timer module that can be configured to provide the OCU functionality and other timer functionalities as well. This specification does not assume the hardware architecture. Instead; it defines parameters and APIs so that they can be implemented on any suitable hardware architecture. The picture below shows a typical representation of an OCU channel.

The 'output' is the action that is actually done upon compare match.



**Figure 1: Abstract view of an OCU channel**

## 2 Acronyms and abbreviations

Acronyms and abbreviations that have a local scope appear in the glossary below. Those that have a global scope are contained in the AUTOSAR glossary.

| <b>Acronym/Abbreviation</b> | <b>Description</b>                    |
|-----------------------------|---------------------------------------|
| OCU                         | Output Compare Unit                   |
| DMA                         | Direct Memory Access                  |
| MCAL                        | Microcontroller Abstraction Layer     |
| MCU                         | Microcontroller Unit                  |
| DEM                         | Diagnostic Event Manager.             |
| DET                         | Default Error Tracer.                 |
| SPAL                        | Standard Peripheral Abstraction Layer |
| MCU                         | Microcontroller Unit.                 |
| ISR                         | Interrupt Service Routine.            |

| <b>Term definition:</b> | <b>Description:</b>   |
|-------------------------|---|
| OCU channel             | Represents a logical entity composed of a free running counter a comparison threshold and the action that is done as a result of the comparison process.  |
| Compare threshold.      | Target value that is compared with the content of the counter each time the counter is increased by one unit.   |
| Free running counter    | A counter that runs from a minimum (respectively a maximum) to a maximum (respectively a minimum) value and restarts automatically from the minimum (respectively a maximum) after reaching the maximum (respectively the minimum) value. |
| Reference Interval      | Interval (in ticks) given by the caller of the <i>Ocu_SetAbsolutThreshold</i> API, and used as base to compute the return information.  |



### 3 Related documentation

#### 3.1 Input documents

- [1] Layered Software Architecture  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf
- [2] General Requirements on SPAL  
AUTOSAR\_SRS\_SPALGeneral.pdf
- [3] General Requirements on Basic Software Modules  
AUTOSAR\_SRS\_BSWGeneral.pdf
- [4] Specification of Default Error Tracer  
AUTOSAR\_SWS\_DefaultErrorTracer.pdf
- [5] Specification of MCU Driver  
AUTOSAR\_SWS\_MCUDriver.pdf
- [6] Specification of ECU Configuration,  
AUTOSAR\_TPS\_ECUConfiguration.pdf
- [7] Basic Software Module Description Template,  
AUTOSAR\_TPS\_BSWModuleDescriptionTemplate.pdf
- [8] List of Basic Software Modules  
AUTOSAR\_TR\_BSWModuleList.pdf

## 4 Constraints and assumptions

### 4.1 Assumptions

#### 4.1.1 Clock

The driver does not support dynamic changes of the clock.

Since the system clock is fully managed by the MCU module, any dynamic change in the system clock settings will impact this module.

The module does not run in the sleep mode.

#### 4.1.2 Resources

The allocation of resources is made exclusively by SW or HW to avoid shared resource issues.

e.g: usage of the API `Ocu_SetPinState`. This API cannot be called to change the state of a pin for a channel that is in the RUNNING state, otherwise there might be a conflict between the state set automatically by the hardware upon compare match and the one set by the API.

#### 4.1.3 Counting and comparing

Our assumption is that the hardware that will operate this driver has the following counter abstraction model (example for an eight-bit counter).

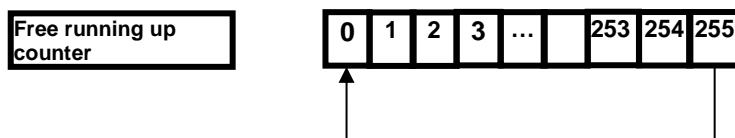


Figure 2: Abstraction model of the free running counter for this driver

Minimum value is 0

Maximum value is 255

The counter is reloaded with 0 when it exceeds the maximum value. That means it has 256 count steps.

Due to the quantization of counting, two different cases are possible when comparing the content of the counter with the threshold. The comparison can occur when entering a state of the counter or while exiting from a state, as shown in the picture below

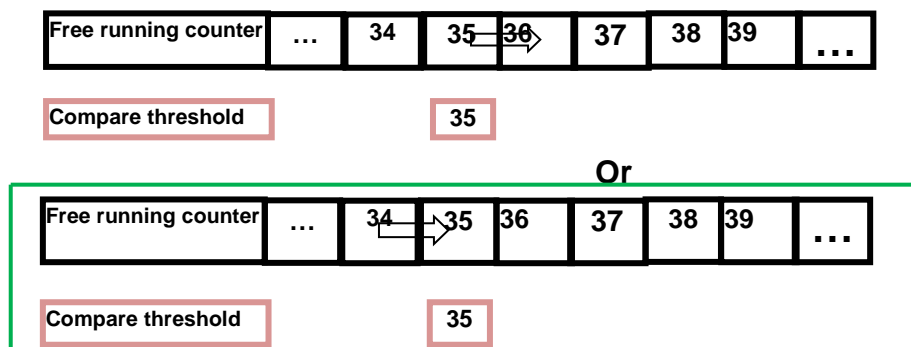


Figure 3: Abstraction model of the comparison process expected in driver.

The expected behavior of this driver is to have the comparison done **on entering** the state represented by the threshold.

## 4.2 Limitations

No limitations.

## 4.3 Applicability to car domains

No restrictions.

## 5 Dependencies to other modules

### Module DET

If development error detection is enabled for the OCU driver, then the driver shall raise errors to the Default Error Tracer (DET) whenever a development error is encountered by this module.

### Module DEM

The OCU driver shall report production errors to the Diagnostic Event Manager (DEM).

### Module MCU Driver

The Microcontroller Unit Driver (MCU Driver) is primarily responsible for initializing and controlling the chip internal clock sources and clock prescalers. The OCU depends on the system clock. Thus, changes of the system clock (e.g. PLL on → PLL off) also affect the clock settings of the OCU hardware.

The MCU driver will set global prescalers, and the OCU clock. The OCU driver will not take care of setting the registers that configure the global clock, global prescalers and PLL in its initialization function. This has to be done by the MCU module. The OCU driver only configures local (OCU peripheral specific) resources.

Document [6] AUTOSAR\_TPS\_ECUConfiguration contains a chapter '4.8 - Clock Tree Configuration', which details the mechanism to deliver reference clock signals to peripherals.

### Module PORT

The configuration of port pins used for the OCU as outputs is done by the PORT driver. Hence the PORT driver has to be initialized prior to the use of OCU functions.

## 5.1 File structure

### 5.1.1 Code file structure

**[SWS\_Ocu\_00001]** [The code file structure shall not be defined completely within this specification. At this point it shall be pointed out that the code-file structure shall include the following files

- Ocu\_Lcfg.c – for link time configurable parameters and
- Ocu\_PBcfg.c – for post build time configurable parameters.

These files shall contain all link time and post-build time configurable parameters.] (SRS\_BSW\_00419, SRS\_BSW\_00346, SRS\_BSW\_00314)

### 5.1.2 Header file structure

**[SWS\_Ocu\_00006]** [Ocu.c shall include Ocu.h and Det.h.] ()

## 6 Requirements traceability

| Requirement   | Description  | Satisfied by  |
|---------------|--|---------------|
| SRS_BSW_00003 | All software modules shall provide version and identification information  | SWS_Ocu_00156 |
| SRS_BSW_00004 | All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files   | SWS_Ocu_00012 |
| SRS_BSW_00005 | Modules of the $\hat{\mu}$ C Abstraction Layer (MCAL) may not have hard coded horizontal interfaces  | SWS_Ocu_00156 |
| SRS_BSW_00006 | The source code of software modules above the $\hat{\mu}$ C Abstraction Layer (MCAL) shall not be processor and compiler dependent.            | SWS_Ocu_00156 |
| SRS_BSW_00007 | All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.   | SWS_Ocu_00156 |
| SRS_BSW_00009 | All Basic SW Modules shall be documented according to a common standard.   | SWS_Ocu_00156 |
| SRS_BSW_00010 | The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.                    | SWS_Ocu_00156 |
| SRS_BSW_00101 | The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function                             | SWS_Ocu_00036 |
| SRS_BSW_00159 | All modules of the AUTOSAR Basic Software shall support a tool based configuration   | SWS_Ocu_00156 |
| SRS_BSW_00160 | Configuration files of AUTOSAR Basic SW module shall be readable for human beings  | SWS_Ocu_00156 |
| SRS_BSW_00161 | The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers | SWS_Ocu_00156 |
| SRS_BSW_00162 | The AUTOSAR Basic Software shall provide a hardware abstraction layer  | SWS_Ocu_00156 |

|               |  |  |
|---------------|--|--|
| SRS_BSW_00164 | The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules               | SWS_Ocu_00156  |
| SRS_BSW_00167 | All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks               | SWS_Ocu_00156  |
| SRS_BSW_00168 | SW components shall be tested by a function defined in a common API in the Basis-SW  | SWS_Ocu_00156  |
| SRS_BSW_00170 | The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands         | SWS_Ocu_00156  |
| SRS_BSW_00171 | Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time         | SWS_Ocu_00049, SWS_Ocu_00070, SWS_Ocu_00079, SWS_Ocu_00088, SWS_Ocu_00094, SWS_Ocu_00103, SWS_Ocu_00111, SWS_Ocu_00118 |
| SRS_BSW_00172 | The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system | SWS_Ocu_00156  |
| SRS_BSW_00300 | All AUTOSAR Basic Software Modules shall be identified by an unambiguous name  | SWS_Ocu_00156  |
| SRS_BSW_00301 | All AUTOSAR Basic Software Modules shall only import the necessary information   | SWS_Ocu_00156  |
| SRS_BSW_00302 | All AUTOSAR Basic Software Modules shall only export information needed by other modules   | SWS_Ocu_00156  |
| SRS_BSW_00304 | All AUTOSAR Basic Software Modules shall use only AUTOSAR data types instead of native C data types                              | SWS_Ocu_00156  |
| SRS_BSW_00305 | Data types naming convention   | SWS_Ocu_00156  |
| SRS_BSW_00306 | AUTOSAR Basic Software Modules shall be compiler and platform independent  | SWS_Ocu_00156  |
| SRS_BSW_00307 | Global variables naming convention   | SWS_Ocu_00156  |
| SRS_BSW_00308 | AUTOSAR Basic Software Modules shall not define  | SWS_Ocu_00156  |

|               |   |  |
|---------------|---|--|
|               | global data in their header files, but in the C file  |  |
| SRS_BSW_00309 | All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword | SWS_Ocu_00156  |
| SRS_BSW_00310 | API naming convention   | SWS_Ocu_00156  |
| SRS_BSW_00312 | Shared code shall be reentrant  | SWS_Ocu_00156  |
| SRS_BSW_00314 | All internal driver modules shall separate the interrupt frame definition from the service routine                                  | SWS_Ocu_00001  |
| SRS_BSW_00323 | All AUTOSAR Basic Software Modules shall check passed API parameters for validity   | SWS_Ocu_00016, SWS_Ocu_00056, SWS_Ocu_00064, SWS_Ocu_00071, SWS_Ocu_00072, SWS_Ocu_00073, SWS_Ocu_00075, SWS_Ocu_00080, SWS_Ocu_00081, SWS_Ocu_00082, SWS_Ocu_00089, SWS_Ocu_00096, SWS_Ocu_00105, SWS_Ocu_00113, SWS_Ocu_00120, SWS_Ocu_00126 |
| SRS_BSW_00325 | The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short                   | SWS_Ocu_00156  |
| SRS_BSW_00327 | Error values naming convention  | SWS_Ocu_00016, SWS_Ocu_00156   |
| SRS_BSW_00328 | All AUTOSAR Basic Software Modules shall avoid the duplication of code  | SWS_Ocu_00156  |
| SRS_BSW_00330 | It shall be allowed to use macros instead of functions where source code is used and runtime is critical                            | SWS_Ocu_00156  |
| SRS_BSW_00331 | All Basic Software Modules shall strictly separate error and status information   | SWS_Ocu_00016, SWS_Ocu_00156   |
| SRS_BSW_00333 | For each callback function it shall be specified if it is called from interrupt context or not                                      | SWS_Ocu_00156  |
| SRS_BSW_00334 | All Basic Software Modules shall provide an XML file that contains the meta data  | SWS_Ocu_00156  |
| SRS_BSW_00335 | Status values naming convention   | SWS_Ocu_00156  |
| SRS_BSW_00336 | Basic SW module shall be able to shutdown   | SWS_Ocu_00046  |
| SRS_BSW_00337 | Classification of development errors  | SWS_Ocu_00015, SWS_Ocu_00016, SWS_Ocu_00017  |

|               |  |  |
|---------------|--|--|
| SRS_BSW_00339 | Reporting of production relevant error status  | SWS_Ocu_00017, SWS_Ocu_00019, SWS_Ocu_00020, SWS_Ocu_00021 |
| SRS_BSW_00341 | Module documentation shall contain all needed informations   | SWS_Ocu_00156  |
| SRS_BSW_00342 | It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed | SWS_Ocu_00156  |
| SRS_BSW_00343 | The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit                   | SWS_Ocu_00013  |
| SRS_BSW_00344 | BSW Modules shall support link-time configuration  | SWS_Ocu_00036  |
| SRS_BSW_00346 | All AUTOSAR Basic Software Modules shall provide at least a basic set of module files  | SWS_Ocu_00001  |
| SRS_BSW_00347 | A Naming separation of different instances of BSW drivers shall be in place  | SWS_Ocu_00156  |
| SRS_BSW_00348 | All AUTOSAR standard types and constants shall be placed and organized in a standard type header file                                | SWS_Ocu_00156  |
| SRS_BSW_00350 | All AUTOSAR Basic Software Modules shall allow the enabling/disabling of detection and reporting of development errors.              | SWS_Ocu_00156  |
| SRS_BSW_00353 | All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header             | SWS_Ocu_00156  |
| SRS_BSW_00357 | For success/failure of an API call a standard return type shall be defined   | SWS_Ocu_00156  |
| SRS_BSW_00358 | The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void                                      | SWS_Ocu_00156  |
| SRS_BSW_00359 | All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible                           | SWS_Ocu_00128, SWS_Ocu_00156                               |
| SRS_BSW_00360 | AUTOSAR Basic Software Modules callback functions are allowed to have parameters   | SWS_Ocu_00128, SWS_Ocu_00156                               |



|               |   |  |
|---------------|---|--|
| SRS_BSW_00361 | All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header | SWS_Ocu_00156  |
| SRS_BSW_00369 | All AUTOSAR Basic Software Modules shall not return specific development error codes via the API  | SWS_Ocu_00018, SWS_Ocu_00019   |
| SRS_BSW_00373 | The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention                                | SWS_Ocu_00156  |
| SRS_BSW_00375 | Basic Software Modules shall report wake-up reasons   | SWS_Ocu_00156  |
| SRS_BSW_00377 | A Basic Software Module can return a module specific types  | SWS_Ocu_00156  |
| SRS_BSW_00378 | AUTOSAR shall provide a boolean type  | SWS_Ocu_00156  |
| SRS_BSW_00383 | The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description    | SWS_Ocu_00156  |
| SRS_BSW_00385 | List possible error notifications   | SWS_Ocu_00016, SWS_Ocu_00017   |
| SRS_BSW_00386 | The BSW shall specify the configuration for detecting an error  | SWS_Ocu_00016, SWS_Ocu_00017, SWS_Ocu_00018, SWS_Ocu_00019, SWS_Ocu_00043, SWS_Ocu_00050, SWS_Ocu_00056, SWS_Ocu_00057, SWS_Ocu_00064, SWS_Ocu_00065, SWS_Ocu_00071, SWS_Ocu_00072, SWS_Ocu_00073, SWS_Ocu_00074, SWS_Ocu_00075, SWS_Ocu_00080, SWS_Ocu_00081, SWS_Ocu_00082, SWS_Ocu_00083, SWS_Ocu_00089, SWS_Ocu_00090, SWS_Ocu_00095, SWS_Ocu_00096, SWS_Ocu_00104, SWS_Ocu_00105, SWS_Ocu_00112, SWS_Ocu_00113, SWS_Ocu_00119, SWS_Ocu_00120, SWS_Ocu_00126 |
| SRS_BSW_00401 | Documentation of multiple instances of configuration parameters shall be available  | SWS_Ocu_00156  |
| SRS_BSW_00404 | BSW Modules shall support post-build configuration  | SWS_Ocu_00036  |
| SRS_BSW_00405 | BSW Modules shall support multiple configuration sets   | SWS_Ocu_00033  |
| SRS_BSW_00406 | A static status variable denoting if a BSW module is initialized shall be initialized   | SWS_Ocu_00043, SWS_Ocu_00050, SWS_Ocu_00055, SWS_Ocu_00057, SWS_Ocu_00065, SWS_Ocu_00074,  |

|               |  |  |
|---------------|--|--|
|               | with value 0 before any APIs of the BSW module is called   | SWS_Ocu_00083, SWS_Ocu_00090, SWS_Ocu_00095, SWS_Ocu_00104, SWS_Ocu_00112, SWS_Ocu_00119 |
| SRS_BSW_00407 | Each BSW module shall provide a function to read out the version information of a dedicated module implementation  | SWS_Ocu_00123, SWS_Ocu_00124   |
| SRS_BSW_00408 | All AUTOSAR Basic Software Modules configuration parameters shall be named according to a specific naming rule     | SWS_Ocu_00156  |
| SRS_BSW_00410 | Compiler switches shall have defined values  | SWS_Ocu_00156  |
| SRS_BSW_00411 | All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API       | SWS_Ocu_00124  |
| SRS_BSW_00413 | An index-based accessing of the instances of BSW modules shall be done   | SWS_Ocu_00156  |
| SRS_BSW_00414 | Init functions shall have a pointer to a configuration structure as single parameter                               | SWS_Ocu_00156  |
| SRS_BSW_00415 | Interfaces which are provided exclusively for one module shall be separated into a dedicated header file           | SWS_Ocu_00156  |
| SRS_BSW_00416 | The sequence of modules to be initialized shall be configurable  | SWS_Ocu_00156  |
| SRS_BSW_00417 | Software which is not part of the SW-C shall report error events only after the DEM is fully operational.          | SWS_Ocu_00156  |
| SRS_BSW_00419 | If a pre-compile time configuration parameter is implemented as "const" it should be placed into a separate c-file | SWS_Ocu_00001  |
| SRS_BSW_00423 | BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template                       | SWS_Ocu_00156  |
| SRS_BSW_00424 | BSW module main processing functions shall not be allowed to enter a wait state                                    | SWS_Ocu_00156  |
| SRS_BSW_00425 | The BSW module description template shall provide means to model the defined trigger                               | SWS_Ocu_00156  |

|               |  |  |
|---------------|--|--|
|               | conditions of schedulable objects  |  |
| SRS_BSW_00426 | BSW Modules shall ensure data consistency of data which is shared between BSW modules                          | SWS_Ocu_00156  |
| SRS_BSW_00427 | ISR functions shall be defined and documented in the BSW module description template                           | SWS_Ocu_00156  |
| SRS_BSW_00428 | A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence | SWS_Ocu_00156  |
| SRS_BSW_00429 | Access to OS is restricted   | SWS_Ocu_00156  |
| SRS_BSW_00432 | Modules should have separate main processing functions for read/receive and write/transmit data path           | SWS_Ocu_00156  |
| SRS_BSW_00433 | Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler         | SWS_Ocu_00156  |
| SRS_BSW_00438 | Configuration data shall be defined in a structure   | SWS_Ocu_00033  |
| SRS_Ocu_00002 | The OCU driver shall support the following basic static configurations per channel                             | SWS_Ocu_00033  |
| SRS_Ocu_00005 | The OCU Driver shall provide the functionality to de-initialize OCU driver                                     | SWS_Ocu_00045  |
| SRS_Ocu_00007 | The OCU driver shall allow enabling /disabling notifications for an OCU channel during runtime                 | SWS_Ocu_00108, SWS_Ocu_00109, SWS_Ocu_00115, SWS_Ocu_00116 |
| SRS_Ocu_00008 | The OCU driver shall provide services for starting and stopping a channel                                      | SWS_Ocu_00051, SWS_Ocu_00052, SWS_Ocu_00058, SWS_Ocu_00059 |
| SRS_Ocu_00009 | The OCU driver shall provide a synchronous service for reading the value of the counter                        | SWS_Ocu_00085, SWS_Ocu_00086                               |
| SRS_Ocu_00010 | The OCU driver shall provide services to modify the value of the threshold of a channel                        | SWS_Ocu_00091, SWS_Ocu_00092, SWS_Ocu_00100                |
| SRS_Ocu_00011 | The OCU driver shall provide a synchronous service to set the state of the output pin attached to a channel    | SWS_Ocu_00066, SWS_Ocu_00067                               |
| SRS_Ocu_00012 | The OCU driver shall provide a service to set the action that will be performed by the pin                     | SWS_Ocu_00076, SWS_Ocu_00077                               |

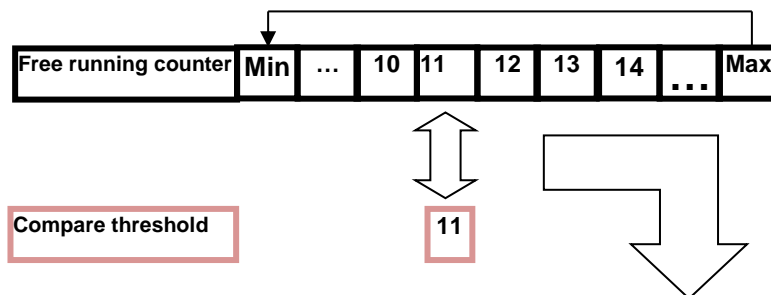
|                |  |  |
|----------------|--|--|
|                | attached to a channel upon comparison match  |  |
| SRS_SPAL_00157 | All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers | SWS_Ocu_00128, SWS_Ocu_00129                               |
| SRS_SPAL_12056 | All driver modules shall allow the static configuration of notification mechanism                                      | SWS_Ocu_00132  |
| SRS_SPAL_12057 | All driver modules shall implement an interface for initialization   | SWS_Ocu_00036, SWS_Ocu_00037, SWS_Ocu_00039, SWS_Ocu_00040 |
| SRS_SPAL_12063 | All driver modules shall only support raw value mode   | SWS_Ocu_00029  |
| SRS_SPAL_12064 | All driver modules shall raise an error if the change of the operation mode leads to degradation of running operations | SWS_Ocu_00156  |
| SRS_SPAL_12067 | All driver modules shall set their wake-up conditions depending on the selected operation mode                         | SWS_Ocu_00156  |
| SRS_SPAL_12068 | The modules of the MCAL shall be initialized in a defined sequence   | SWS_Ocu_00156  |
| SRS_SPAL_12069 | All drivers of the SPAL that wake up from a wake-up interrupt shall report the wake-up reason                          | SWS_Ocu_00156  |
| SRS_SPAL_12075 | All drivers with random streaming capabilities shall use application buffers   | SWS_Ocu_00156  |
| SRS_SPAL_12077 | All drivers shall provide a non blocking implementation  | SWS_Ocu_00156  |
| SRS_SPAL_12078 | The drivers shall be coded in a way that is most efficient in terms of memory and runtime resources                    | SWS_Ocu_00156  |
| SRS_SPAL_12092 | The driver's API shall be accessed by its handler or manager   | SWS_Ocu_00156  |
| SRS_SPAL_12125 | All driver modules shall only initialize the configured resources  | SWS_Ocu_00010, SWS_Ocu_00011, SWS_Ocu_00037, SWS_Ocu_00136 |
| SRS_SPAL_12129 | The ISRs shall be responsible for resetting the interrupt flags and calling the according notification function        | SWS_Ocu_00130  |

|                |  |   |
|----------------|--|---|
| SRS_SPAL_12163 | All driver modules shall implement an interface for de-initialization  | SWS_Ocu_00046, SWS_Ocu_00047  |
| SRS_SPAL_12169 | All driver modules that provide different operation modes shall provide a service for mode selection                   | SWS_Ocu_00156   |
| SRS_SPAL_12263 | The implementation of all driver modules shall allow the configuration of specific module parameter types at link time | SWS_Ocu_00033   |
| SRS_SPAL_12265 | Configuration data shall be kept constant  | SWS_Ocu_00156   |
| SRS_SPAL_12267 | Wakeup sources shall be initialized by MCAL drivers and/or the MCU driver  | SWS_Ocu_00156   |
| SRS_SPAL_12448 | All driver modules shall have a specific behavior after a development error detection                                  | SWS_Ocu_00043, SWS_Ocu_00050, SWS_Ocu_00055, SWS_Ocu_00056, SWS_Ocu_00057, SWS_Ocu_00064, SWS_Ocu_00065, SWS_Ocu_00071, SWS_Ocu_00072, SWS_Ocu_00073, SWS_Ocu_00074, SWS_Ocu_00075, SWS_Ocu_00080, SWS_Ocu_00081, SWS_Ocu_00082, SWS_Ocu_00083, SWS_Ocu_00089, SWS_Ocu_00090, SWS_Ocu_00095, SWS_Ocu_00096, SWS_Ocu_00104, SWS_Ocu_00105, SWS_Ocu_00112, SWS_Ocu_00113, SWS_Ocu_00119, SWS_Ocu_00120, SWS_Ocu_00126 |
| SRS_SPAL_12461 | Specific rules regarding initialization of controller registers shall apply to all driver implementations              | SWS_Ocu_00038, SWS_Ocu_00156  |
| SRS_SPAL_12462 | The register initialization settings shall be published  | SWS_Ocu_00156   |
| SRS_SPAL_12463 | The register initialization settings shall be combined and forwarded   | SWS_Ocu_00156   |

## 7 Functional specification

### 7.1 General behavior

The OCU channel is composed of two main elements: a free running counter and a compare threshold. These elements act together to generate actions required by the user. The free running counter can be provided by hardware or software whereas the threshold is a value set by the user. It is then compared with the current content of the counter each time the counter is increased by one unit.



The driver compares both values each time the counter is increased by one unit. In case of equality, two different types of action can be done:

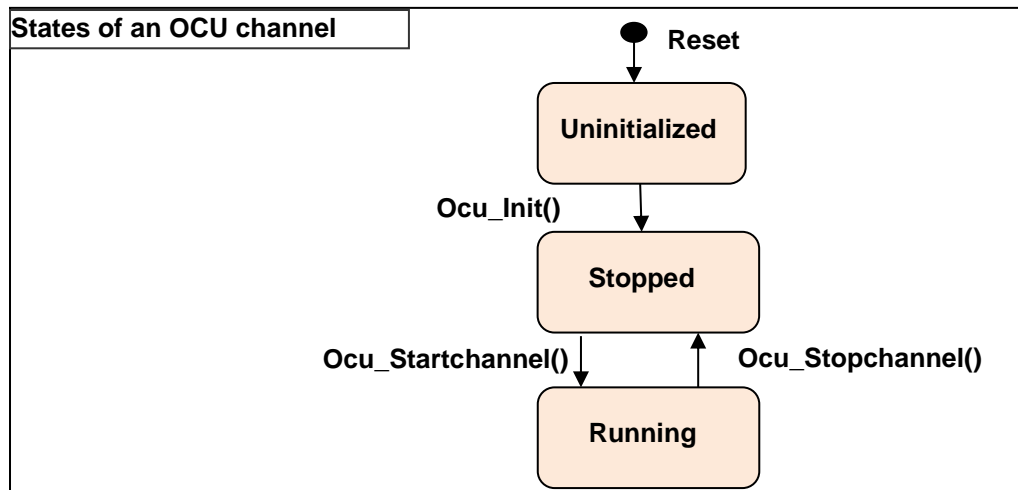
- report the information to the upper layer through a notification function.
- act on a configured output pin

**Figure 4: General behavior of the driver**

The OCU driver provides the following services for managing a channel:

- Starting a channel
- Stopping a channel
- Setting the comparison threshold value
- Enabling and disabling a notification function for a channel
- Getting counter values
- Changing output pin states

The states and the state transitions of an output compare channel are shown in the figure below.



**Figure 5: State diagram of an OCU channel**

An Ocu channel has a simple state diagram with the states shown above. All the channels of the driver are initialized at once with the API `Ocu_Init()`. There's no API to initialize individually each channel.

Depending on the hardware architecture, the hardware tied to an Ocu channel may be managed by the OCU cell or any other timer module in the microcontroller.

## 7.2 Version check

### 7.2.1 Background & Rationale

The integration of incompatible files is to be avoided. Minimum implementation is the version check of the header file inside the .c file (version numbers of .c and .h files must be identical).

**[SWS\_Ocu\_00012]** [The OCU driver shall perform Inter Module Checks to avoid integration of incompatible files. The imported included files shall be checked by preprocessing 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 OCU driver.

If the values are not identical to the expected values, an error shall be reported.]

(SRS\_BSW\_00004)

## 7.3 Time Unit Ticks

### 7.3.1 Background & Rationale

To get times out of register values it is necessary to know the oscillator frequency, prescalers and some other settings of the whole system clock. Since these settings are made in MCU 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.

### 7.3.2 Requirements

**[SWS\_Ocu\_00013]** [All time units used within the API services of the OCU driver shall be of the unit ticks.] (SRS\_BSW\_00343)

## 7.4 Error classification

**[SWS\_Ocu\_00015]** [Development error values are of type uint8. ]  
 (SRS\_BSW\_00337)

**[SWS\_Ocu\_00016]** [The following errors shall be detectable by the OCU driver depending on its build version (development / production mode).]  
 (SRS\_BSW\_00337, SRS\_BSW\_00323, SRS\_BSW\_00327, SRS\_BSW\_00331, SRS\_BSW\_00385, SRS\_BSW\_00386)

### 7.4.1 Development Errors

**[SWS\_Ocu\_91001]**

| <i>Type of error</i>   | <i>Related error code</i>   | <i>Error value</i> |
|--|-----------------------------|--------------------|
| API services other than Ocu_GetVersionInfo() and Ocu_init() used without module initialization   | OCU_E_UNINIT                | 0x02               |
| API service used with an invalid channel Identifier.   | OCU_E_PARAM_INVALID_CHANNEL | 0x03               |
| API Ocu_SetPinState() called with an invalid pin state or when the channel is in the RUNNING state.  | OCU_E_PARAM_INVALID_STATE   | 0x04               |
| API Ocu_SetPinAction() called with an invalid pin action.  | OCU_E_PARAM_INVALID_ACTION  | 0x05               |
| Usage of Ocu_DisableNotification() or Ocu_EnableNotification() on a channel where a NULL pointer is configured as the notification function. | OCU_E_NO_VALID_NOTIF        | 0x06               |
| API Ocu_Init() called while the OCU driver has already been initialized  | OCU_E_ALREADY_INITIALIZED   | 0x07               |
| API Ocu_GetVersionInfo() is called with a NULL parameter.  | OCU_E_PARAM_POINTER         | 0x08               |
| Ocu_SetPinState() or Ocu_SetPinAction() called for a channel that doesn't have an associated output pin.                                     | OCU_E_PARAM_NO_PIN          | 0x0A               |
| OCU initialization has been failed, e.g. selected configuration set doesn't exist.   | OCU_E_INIT_FAILED           | 0x0B               |

l()

**[SWS\_Ocu\_00017]** [Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the OCU device specific implementation specification. The classification and enumeration shall be compatible to the errors listed above.] (SRS\_BSW\_00337, SRS\_BSW\_00339, SRS\_BSW\_00385, SRS\_BSW\_00386)



## 7.4.2 Runtime Errors

### [SWS\_Ocu\_91002]

| <i>Type of error</i>  | <i>Related error code</i> | <i>Error value</i> |
|---|---------------------------|--------------------|
| API Ocu_StartChannel() called on a channel that is in state RUNNING | OCU_E_BUSY                | 0x09               |

]()

## 7.4.3 Transient Faults

There are no transient faults.

## 7.4.4 Production Errors

There are no production errors.

## 7.4.5 Extended Production Errors

There are no extended production errors.

## 7.5 Error Detection

**[SWS\_Ocu\_00018]** [The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch `OcuDevErrorDetectApi` shall activate or deactivate the detection of all development errors.] (SRS\_BSW\_00369, SRS\_BSW\_00386)

**[SWS\_Ocu\_00019]** [If the switch `OcuDevErrorDetectApi` is enabled, then API parameter checking is enabled. The detailed description of the detected errors can be found in chapter [Error classification](#) and chapter [API specification](#).] (SRS\_BSW\_00386, SRS\_BSW\_00369, SRS\_BSW\_00339)

**[SWS\_Ocu\_00020]** [The detection of production errors cannot be switched off.] (SRS\_BSW\_00339)

## 7.6 Error Notification

**[SWS\_Ocu\_00021]** [Detected development errors shall be reported with the service `Det_ReportError` of the Default Error Tracer (DET) if the pre-processor switch `OcuDevErrorDetectApi` is set.] (SRS\_BSW\_00339)

## 7.7 Debug Support

**[SWS\_Ocu\_00023]** [ Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable. ] ()

**[SWS\_Ocu\_00024]** [ All type definitions of variables which shall be debugged shall be accessible by the header file Ocu.h. ] ()

**[SWS\_Ocu\_00025]** [ 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". ] ()

**[SWS\_Ocu\_00026]** [ Variables available for debugging shall be described in the respective OCU driver Description. ] ()

## 8 API specification

### 8.1 Imported types

This chapter lists all types included from other modules.

#### [SWS\_Ocu\_00027]

| <i>Module</i> | <i>Header File</i> | <i>Imported Type</i> |
|---------------|--------------------|----------------------|
| Std           | Std_Types.h        | Std_ReturnType       |
|               | Std_Types.h        | Std_VersionInfoType  |

]()

### 8.2 Type definitions

#### 8.2.1 Ocu\_ChannelType

##### [SWS\_Ocu\_00028]

|                      |                                       |        |   |
|----------------------|---------------------------------------|--------|---|
| <b>Name</b>          | Ocu_ChannelType                       |        |   |
| <b>Kind</b>          | Type                                  |        |   |
| <b>Derived from</b>  | uint                                  |        |   |
| <b>Range</b>         | 8 / 16 / 32 bits                      | -<br>- | This is implementation specific but not all values may be valid within the type. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform. |
| <b>Description</b>   | Numeric identifier of an OCU channel. |        |   |
| <b>Available via</b> | Ocu.h                                 |        |   |

]()

#### 8.2.2 Ocu\_ValueType

##### [SWS\_Ocu\_00029]

|                     |   |        |   |
|---------------------|---|--------|---|
| <b>Name</b>         | Ocu_ValueType   |        |   |
| <b>Kind</b>         | Type  |        |   |
| <b>Derived from</b> | uint  |        |   |
| <b>Range</b>        | 8 / 16 / 32 bits  | -<br>- | This is implementation specific but not all values may be valid within the type. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform. |
| <b>Description</b>  | Type for reading the counter and writing the threshold values (in number of ticks). |        |   |

|                      |       |
|----------------------|-------|
| <b>Available via</b> | Ocu.h |
|----------------------|-------|

](SRS\_SPAL\_12063)

### 8.2.3 Ocu\_PinStateType

[SWS\_Ocu\_00031][

|                      |   |      |  |
|----------------------|---|------|--|
| <b>Name</b>          | Ocu_PinStateType                                  |      |  |
| <b>Kind</b>          | Enumeration                                       |      |  |
| <b>Range</b>         | OCU_HIGH  | 0x00 | The pin associated to an OCU channel is in high state. |
|                      | OCU_LOW   | 0x01 | The pin associated to an OCU channel is in low state.  |
| <b>Description</b>   | Output state of the pin linked to an OCU channel. |      |  |
| <b>Available via</b> | Ocu.h   |      |  |

]()

### 8.2.4 Ocu\_PinActionType

[SWS\_Ocu\_00032][

|                      |   |      |   |
|----------------------|---|------|---|
| <b>Name</b>          | Ocu_PinActionType   |      |   |
| <b>Kind</b>          | Enumeration   |      |   |
| <b>Range</b>         | OCU_SET_HIGH  | 0x00 | The channel pin will be set HIGH upon compare match.                                      |
|                      | OCU_SET_LOW   | 0x01 | The channel pin will be set LOW upon compare match.                                       |
|                      | OCU_TOGGLE  | 0x02 | The channel pin will be set to the opposite of its current level HIGH upon compare match. |
|                      | OCU_DISABLE   | 0x03 | The channel pin will remain at its current level upon compare match.                      |
| <b>Description</b>   | Automatic action (by hardware) to be performed on a pin attached to an OCU channel. |      |   |
| <b>Available via</b> | Ocu.h   |      |   |

]()

### 8.2.5 Ocu\_ConfigType

[SWS\_Ocu\_00033][

|             |                |
|-------------|----------------|
| <b>Name</b> | Ocu_ConfigType |
|-------------|----------------|

|                      |   |  |
|----------------------|---|--|
| <b>Kind</b>          | Structure   |  |
| <b>Elements</b>      | Hardware dependent  |  |
|                      | <b>Type</b>   | --   |
|                      | <b>Comment</b>  | The contents of the initialization data structure are hardware specific. |
| <b>Description</b>   | This is the type of the data structure containing the initialization data for the OCU driver. |  |
| <b>Available via</b> | Ocu.h   |  |

] (SRS\_Ocu\_00002, SRS\_SPAL\_12263, SRS\_BSW\_00405, SRS\_BSW\_00438)

### 8.2.6 Ocu\_ReturnType

[SWS\_Ocu\_00138]

|                      |   |      |   |
|----------------------|---|------|---|
| <b>Name</b>          | Ocu_ReturnType  |      |   |
| <b>Kind</b>          | Enumeration   |      |   |
| <b>Range</b>         | OCU_CM_IN_REF_INTERVAL                                  | 0x00 | The compare match will occur inside the current Reference Interval.     |
|                      | OCU_CM_OUT_REF_INTERVAL                                 | 0x01 | The compare match will not occur inside the current Reference Interval. |
| <b>Description</b>   | Return information after setting a new threshold value. |      |   |
| <b>Available via</b> | Ocu.h   |      |   |

]()

## 8.3 Function definitions

### 8.3.1 Ocu\_Init

[SWS\_Ocu\_00035]

|                         |  |                                  |
|-------------------------|--|----------------------------------|
| <b>Service Name</b>     | Ocu_Init   |                                  |
| <b>Syntax</b>           | <pre>void Ocu_Init (     const Ocu_ConfigType* ConfigPtr )</pre> |                                  |
| <b>Service ID [hex]</b> | 0x00   |                                  |
| <b>Sync/Async</b>       | Synchronous  |                                  |
| <b>Reentrancy</b>       | Non Reentrant  |                                  |
| <b>Parameters (in)</b>  | ConfigPtr  | Pointer to the configuration set |

|                           |                                 |
|---------------------------|---------------------------------|
| <b>Parameters (inout)</b> | None                            |
| <b>Parameters (out)</b>   | None                            |
| <b>Return value</b>       | None                            |
| <b>Description</b>        | Service for OCU initialization. |
| <b>Available via</b>      | Ocu.h                           |

]()

**[SWS\_Ocu\_00036]** [The function `Ocu_Init` shall initialize all internal variables and the used `Ocu` structure of the microcontroller according to a configuration set referenced by `ConfigPtr`.] (SRS\_BSW\_00344, SRS\_BSW\_00404, SRS\_BSW\_00101, SRS\_SPAL\_12057)

Note: All the channels are initialized at once by the API `Ocu_Init`. There's no API to individually initialize each channel.

**[SWS\_Ocu\_00010]** [If a free-running counter of the OCU cell can be used by another timer module then the `Ocu` driver must not start nor stop the free-running counter.] (SRS\_SPAL\_12125)

**[SWS\_Ocu\_00011]** [The API `Ocu_Init` shall start all free-running counters, which are exclusively used by this driver.](SRS\_SPAL\_12125)

**[SWS\_Ocu\_00037]** [the function `Ocu_Init` shall only initialize the configured resources and shall not touch resources that are not configured in the configuration file.] (SRS\_SPAL\_12057, SRS\_SPAL\_12125)

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

- **[SWS\_Ocu\_00038]** [If the hardware allows for only one usage of the register (register dedicated only to the OCU resource), then the OCU driver is responsible for initializing the register.] (SRS\_SPAL\_12461)
- **Note1:** If the register can affect several hardware modules and if it is not an I/O register it shall be initialized by the MCU driver. (SRS\_SPAL\_12461)

- **Note2:** One-time writable registers that require initialization directly after reset shall be initialized by the start-up code. (SRS\_SPAL\_12461)
- **Note3:** All other registers shall be initialized by the startup code. (SRS\_SPAL\_12461).
- **Note4:** If a register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver. (SRS\_SPAL\_12461)

**[SWS\_Ocu\_00039]** [The function `Ocu_Init` shall stop all channels.]  
(SRS\_SPAL\_12057).

**[SWS\_Ocu\_00040]** [The function `Ocu_Init` shall disable all notifications.]  
(SRS\_SPAL\_12057)

The reason is that the users of these notifications may not be ready. They can call `Ocu_EnableNotification()` to start getting notifications.

**[SWS\_Ocu\_00043]** [If development error detection is enabled for the OCU driver and the function `Ocu_Init` is called when the OCU driver and hardware are already initialized, the function `Ocu_Init` shall raise development error `OCU_E_ALREADY_INITIALIZED` and return without any action.]  
(SRS\_BSW\_00406, SRS\_BSW\_00386, SRS\_SPAL\_12448)

**[SWS\_Ocu\_00044]** [A re-initialization of the OCU driver by executing the function `Ocu_Init` requires a de-initialization before by executing the function `Ocu_DeInit.`] ()

### 8.3.2 Ocu\_DeInit

**[SWS\_Ocu\_00045]**

|                           |   |
|---------------------------|---|
| <b>Service Name</b>       | Ocu_DeInit                              |
| <b>Syntax</b>             | <pre>void Ocu_DeInit (     void )</pre> |
| <b>Service ID [hex]</b>   | 0x01                                    |
| <b>Sync/Async</b>         | Synchronous                             |
| <b>Reentrancy</b>         | Non Reentrant                           |
| <b>Parameters (in)</b>    | None                                    |
| <b>Parameters (inout)</b> | None                                    |

|                         |  |
|-------------------------|--|
| <b>Parameters (out)</b> | None   |
| <b>Return value</b>     | None   |
| <b>Description</b>      | This function de-initializes the OCU module. |
| <b>Available via</b>    | Ocu.h  |

](SRS\_Ocu\_00005)

**[SWS\_Ocu\_00046]** [The function `Ocu_DeInit` shall deinitialize the OCU variables and registers that were initialized by `Ocu_Init` to a state comparable to their power on reset state. Values of registers which are not writeable are excluded.]  
 (SRS\_BSW\_00336, SRS\_SPAL\_12163)

Note: It's the responsibility of the hardware design that the state does not lead to undefined activities in the  $\mu$ C.

**[SWS\_Ocu\_00047]** [The function `Ocu_DeInit` shall disable all used interrupts and notifications.] (SRS\_SPAL\_12163)

**[SWS\_Ocu\_00048]** [The function `Ocu_DeInit` shall influence only the peripherals which are allocated by static configuration and/or the runtime configuration set passed by the previous call of `Ocu_Init()`.] ()

**[SWS\_Ocu\_00136]** [The API `Ocu_DeInit` shall stop all free-running counters, which are exclusively used by this driver.] (SRS\_SPAL\_12125)

Note: To prevent undefined behaviour during de-initialization, the user must stop all RUNNING channels (by calling the function `Ocu_StopChannel`) before calling the API `Ocu_DeInit`. Hence the requirement below.

**[SWS\_Ocu\_00137]** [If development error detection is enabled for the OCU driver: if a channel is still in the RUNNING state when the function `Ocu_DeInit` is called, then the function shall raise the development error 'OCU\_E\_PARAM\_INVALID\_STATE' and return without any action.] ()

**[SWS\_Ocu\_00049]** [The function `Ocu_DeInit` shall be pre compile time configurable On/Off by the configuration parameter: `OcuDeInitApi` {OCU\_DE\_INIT\_API}.] (SRS\_BSW\_00171).

**[SWS\_Ocu\_00050]** [If development error detection is enabled for the OCU driver: If the driver is not initialized, the function `Ocu_DeInit` shall raise the error `OCU_E_UNINIT`.] (SRS\_BSW\_00406, SRS\_BSW\_00386, SRS\_SPAL\_12448)



### 8.3.3 Ocu\_StartChannel

#### [SWS\_Ocu\_00051]

|                           |   |  |
|---------------------------|---|--|
| <b>Service Name</b>       | Ocu_StartChannel  |  |
| <b>Syntax</b>             | Std_ReturnType Ocu_StartChannel (<br>Ocu_ChannelType ChannelNumber<br>) |  |
| <b>Service ID [hex]</b>   | 0x02  |  |
| <b>Sync/Async</b>         | Synchronous   |  |
| <b>Reentrancy</b>         | Reentrant for different channel numbers                                 |  |
| <b>Parameters (in)</b>    | ChannelNumber   | Numeric identifier of the OCU                                |
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | None  |  |
| <b>Return value</b>       | Std_ReturnType  | E_OK Channel was started<br>E_NOT_OK Channel was not started |
| <b>Description</b>        | Service to start an OCU channel.  |  |
| <b>Available via</b>      | Ocu.h   |  |

](SRS\_Ocu\_00008)

**[SWS\_Ocu\_00052]** [The function Ocu\_StartChannel shall start an OCU channel by allowing all compare match configured actions to be performed.] (SRS\_Ocu\_00008)

**[SWS\_Ocu\_00053]** [The function Ocu\_StartChannel shall be reentrant if it is called for different channels.] ()

**[SWS\_Ocu\_00054]** [The state of the selected channel shall be set to “RUNNING” if the function Ocu\_StartChannel has been successfully performed.] ()

**[SWS\_Ocu\_00055]** [ If the function Ocu\_StartChannel is called on a channel in the state "RUNNING", then the function shall raise the error OCU\_E\_BUSY and return without any action.] (SRS\_BSW\_00406, SRS\_SPAL\_12448).

**[SWS\_Ocu\_00056]** [If development error detection is enabled for the OCU driver: If the parameter ChannelNumber is invalid (not within the range specified by the configuration), the function Ocu\_StartChannel shall raise the error OCU\_E\_PARAM\_INVALID\_CHANNEL and return without any action.] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448).

**[SWS\_Ocu\_00057]** [If development error detection is enabled for the OCU driver: If the driver is not initialized, the function `Ocu_StartChannel` shall raise the error `OCU_E_UNINIT` and return without any action.] (SRS\_BSW\_00406, SRS\_BSW\_00386, SRS\_SPAL\_12448).

### 8.3.4 Ocu\_StopChannel

**[SWS\_Ocu\_00058]**

|                           |   |                               |
|---------------------------|---|-------------------------------|
| <b>Service Name</b>       | Ocu_StopChannel   |                               |
| <b>Syntax</b>             | <pre>void Ocu_StopChannel (     Ocu_ChannelType ChannelNumber )</pre> |                               |
| <b>Service ID [hex]</b>   | 0x03  |                               |
| <b>Sync/Async</b>         | Synchronous   |                               |
| <b>Reentrancy</b>         | Reentrant for different channel numbers                               |                               |
| <b>Parameters (in)</b>    | ChannelNumber   | Numeric identifier of the OCU |
| <b>Parameters (inout)</b> | None  |                               |
| <b>Parameters (out)</b>   | None  |                               |
| <b>Return value</b>       | None  |                               |
| <b>Description</b>        | Service to stop an OCU channel.                                       |                               |
| <b>Available via</b>      | Ocu.h   |                               |

](SRS\_Ocu\_00008)

**[SWS\_Ocu\_00059]** [The function `Ocu_StopChannel` shall stop an OCU channel by halting compare match configured actions for this channel.] (SRS\_Ocu\_00008)

**[SWS\_Ocu\_00060]** [The function `Ocu_StopChannel` shall not stop the free-running counter associated with a channel.] ()

Note: This is due to the fact that a free-running counter can be associated with more than one Ocu channel. Therefore, stopping that counter will harm the operation of the other channel(s).

**[SWS\_Ocu\_00061]** [The function `Ocu_StopChannel` shall be reentrant if it is called for different channels.] ()

**[SWS\_Ocu\_00062]** [The state of the selected channel shall be set to “STOPPED” if the function `Ocu_StopChannel` is successfully performed.] ()

**[SWS\_Ocu\_00063]** [ If the function `Ocu_StopChannel` is called on a channel in the state "STOPPED", then the function shall leave without any action (no change of the channel state), and shall **not** raise a development error.] ()

**[SWS\_Ocu\_00064]** [ If development error detection is enabled for the OCU driver: If the parameter `ChannelNumber` is invalid (not within the range specified by the configuration), the function `Ocu_StopChannel` shall raise the error `OCU_E_PARAM_INVALID_CHANNEL` and return without any action.] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448).

**[SWS\_Ocu\_00065]** [ If development error detection is enabled for the OCU driver: If the driver is not initialized, the function `Ocu_StopChannel` shall raise the error `OCU_E_UNINIT` and return without any action.] (SRS\_BSW\_00406, SRS\_BSW\_00386, SRS\_SPAL\_12448).

### 8.3.5 Ocu\_SetPinState

**[SWS\_Ocu\_00066]**[

|                           |  |                               |
|---------------------------|--|-------------------------------|
| <b>Service Name</b>       | Ocu_SetPinState  |                               |
| <b>Syntax</b>             | <pre>void Ocu_SetPinState (     Ocu_ChannelType ChannelNumber,     Ocu_PinStateType PinState )</pre> |                               |
| <b>Service ID [hex]</b>   | 0x04   |                               |
| <b>Sync/Async</b>         | Synchronous  |                               |
| <b>Reentrancy</b>         | Reentrant for different channel numbers  |                               |
| <b>Parameters (in)</b>    | ChannelNumber  | Numeric identifier of the OCU |
|                           | PinState   | OCU_LOW, OCU_HIGH             |
| <b>Parameters (inout)</b> | None   |                               |
| <b>Parameters (out)</b>   | None   |                               |
| <b>Return value</b>       | None   |                               |
| <b>Description</b>        | Service to set immediately the level of the pin associated to an OCU channel.                        |                               |
| <b>Available via</b>      | Ocu.h  |                               |

](SRS\_Ocu\_00011)

**[SWS\_Ocu\_00067]** [The function `Ocu_SetPinState` shall set the pin associated with the channel to the level indicated by "PinState".] (SRS\_Ocu\_00011)

**[SWS\_Ocu\_00068]** [The function `Ocu_SetPinState` shall be reentrant if it is called for different channels.] ()

**[SWS\_Ocu\_00069]** [The function `Ocu_SetPinState` shall be used only if the channel is not in the `RUNNING` state.] ()

Note: The previous requirement also means that it shall be possible to alter the state of a `STOPPED` channel by this API.

**[SWS\_Ocu\_00070]** [The function `Ocu_SetPinState` shall be pre compile time configurable On/Off by the configuration parameter: `OcuSetPinStateApi` {`OCU_SET_PIN_STATE_API`}.] (SRS\_BSW\_00171)

**[SWS\_Ocu\_00071]** [If development error detection is enabled for the OCU driver: If the parameter `ChannelNumber` is invalid (not within the range specified by the configuration), the function `Ocu_SetPinState` shall raise the error `OCU_E_PARAM_INVALID_CHANNEL` and return without any action.] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448)

**[SWS\_Ocu\_00072]** [If development error detection is enabled for the OCU driver: If a pin is not associated with the channel (not defined in the configuration of the channel), the function `Ocu_SetPinState` shall raise the error `OCU_E_PARAM_NO_PIN` and return without any action.] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448)

**[SWS\_Ocu\_00073]** [If development error detection is enabled for the OCU driver: If the parameter `PinState` is invalid (not within the range specified by the configuration), the function `Ocu_SetPinState` shall raise the error `OCU_E_PARAM_INVALID_STATE` and return without any action.] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448)

**[SWS\_Ocu\_00074]** [If development error detection is enabled for the OCU driver: If the driver is not initialized, the function `Ocu_SetPinState` shall raise the error `OCU_E_UNINIT` and return without any action.] (SRS\_BSW\_00406, SRS\_BSW\_00386, SRS\_SPAL\_12448).

**[SWS\_Ocu\_00075]** [If development error detection is enabled for the OCU driver: If the channel is in the `RUNNING` state, the function `Ocu_SetPinState` shall raise the error `OCU_E_PARAM_INVALID_STATE` and return without any action.] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448).

### 8.3.6 Ocu\_SetPinAction

#### [SWS\_Ocu\_00076]

|                           |  |  |
|---------------------------|--|--|
| <b>Service Name</b>       | Ocu_SetPinAction   |  |
| <b>Syntax</b>             | <pre>void Ocu_SetPinAction (     Ocu_ChannelType ChannelNumber,     Ocu_PinActionType PinAction )</pre>        |  |
| <b>Service ID [hex]</b>   | 0x05   |  |
| <b>Sync/Async</b>         | Synchronous  |  |
| <b>Reentrancy</b>         | Reentrant for different channel numbers  |  |
| <b>Parameters (in)</b>    | ChannelNumber  | Numeric identifier of the OCU                      |
|                           | PinAction  | OCU_SET_LOW, OCU_SET_HIGH, OCU_TOGGLE, OCU_DISABLE |
| <b>Parameters (inout)</b> | None   |  |
| <b>Parameters (out)</b>   | None   |  |
| <b>Return value</b>       | None   |  |
| <b>Description</b>        | Service to indicate the driver what shall be done automatically by hardware (if supported) upon compare match. |  |
| <b>Available via</b>      | Ocu.h  |  |

](SRS\_Ocu\_00012)

**[SWS\_Ocu\_00077]** [The function Ocu\_SetPinAction shall set the action to be performed by hardware automatically, at the next compare match in the corresponding OCU channel. ] (SRS\_Ocu\_00012)

**[SWS\_Ocu\_00078]** [The fuction OCU Ocu\_SetPinAction shall be reentrant if it is called for different channels.] ()

**[SWS\_Ocu\_00079]** [The function Ocu\_SetPinAction shall be pre compile time configurable by the configuration parameter: OcuSetPinActionApi {OCU\_SET\_PIN\_ACTION\_API}.] (SRS\_BSW\_00171)

**[SWS\_Ocu\_00080]** [If development error detection is enabled for the OCU driver: If the parameter ChannelNumber is invalid (not within the range specified by the configuration), the function Ocu\_SetPinAction shall raise the error OCU\_E\_PARAM\_INVALID\_CHANNEL and return without any action.] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448)

**[SWS\_Ocu\_00081]** [ If development error detection is enabled for the OCU driver: If a pin is not associated with the channel (not defined in the configuration of the channel), the function `Ocu_SetPinAction` shall raise the error

`OCU_E_PARAM_NO_PIN` and return without any action.] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448)

**[SWS\_Ocu\_00082]** [ If development error detection is enabled for the OCU driver: If the parameter `PinAction` is invalid (not within the range specified by the type), the function `Ocu_SetPinAction` shall raise the error `OCU_E_PARAM_INVALID_ACTION`

and return without any action.] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448)

**[SWS\_Ocu\_00083]** [ If development error detection is enabled for the OCU driver: If the driver is not initialized, the function `Ocu_SetPinAction` shall raise the error

`OCU_E_UNINIT` and return without any action.] (SRS\_BSW\_00406, SRS\_BSW\_00386, SRS\_SPAL\_12448)

**[SWS\_Ocu\_00084]** [ If a pin is associated with a channel; the relevant action with this pin shall be performed upon compare match.] ( )

### 8.3.7 Ocu\_GetCounter

**[SWS\_Ocu\_00085]**[

|                           |   |                                       |
|---------------------------|---|---------------------------------------|
| <b>Service Name</b>       | Ocu_GetCounter  |                                       |
| <b>Syntax</b>             | <pre>Ocu_ValueType Ocu_GetCounter (     Ocu_ChannelType ChannelNumber )</pre> |                                       |
| <b>Service ID [hex]</b>   | 0x06  |                                       |
| <b>Sync/Async</b>         | Synchronous   |                                       |
| <b>Reentrancy</b>         | Reentrant   |                                       |
| <b>Parameters (in)</b>    | ChannelNumber   | Numeric identifier of the OCU channel |
| <b>Parameters (inout)</b> | None  |                                       |
| <b>Parameters (out)</b>   | None  |                                       |
| <b>Return value</b>       | Ocu_ValueType   | Content of the counter in ticks       |
| <b>Description</b>        | Service to read the current value of the counter.                             |                                       |
| <b>Available via</b>      | Ocu.h   |                                       |

](SRS\_Ocu\_00009)

**[SWS\_Ocu\_00086]** [The function `Ocu_GetCounter` shall read and return the value of the counter of the channel indicated by `ChannelNumber`.] (SRS\_Ocu\_00009)

**[SWS\_Ocu\_00087]** [The function `Ocu_GetCounter` shall be re-entrant.] ()

**[SWS\_Ocu\_00088]** [The function `Ocu_GetCounter` shall be pre compile time configurable by the configuration parameter: `OcuGetCounterApi` {`OCU_GET_COUNTER_API`}.] (SRS\_BSW\_00171)

**[SWS\_Ocu\_00089]** [If development error detection is enabled for the OCU driver: If the parameter `ChannelNumber` is invalid (not within the range specified by the configuration), the function `Ocu_GetCounter` shall raise the error `OCU_E_PARAM_INVALID_CHANNEL` and shall return the value “0”.] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448).

**[SWS\_Ocu\_00090]** [If development error detection is enabled for the OCU driver: if the driver is not initialized, then the function `Ocu_GetCounterValue` shall raise the error `OCU_E_UNINIT` and shall return the value “0”.] (SRS\_BSW\_00406, SRS\_BSW\_00386, SRS\_SPAL\_12448).

### 8.3.8 Ocu\_SetAbsoluteThreshold

**[SWS\_Ocu\_00091]**[

|                         |   |   |
|-------------------------|---|---|
| <b>Service Name</b>     | Ocu_SetAbsoluteThreshold  |   |
| <b>Syntax</b>           | <pre>Ocu_ReturnType Ocu_SetAbsoluteThreshold (     Ocu_ChannelType ChannelNumber,     Ocu_ValueType ReferenceValue,     Ocu_ValueType AbsoluteValue )</pre> |   |
| <b>Service ID [hex]</b> | 0x07  |   |
| <b>Sync/Async</b>       | Synchronous   |   |
| <b>Reentrancy</b>       | Reentrant for different channel numbers   |   |
| <b>Parameters (in)</b>  | Channel Number  | Numeric identifier of the OCU channel   |
|                         | Reference Value   | Value given by the upper layer and used as a base to determine whether to call the notification before the function exits or not. |
|                         | Absolute Value  | Value to compare with the content of the counter. This value is in ticks.   |

|                           |   |  |
|---------------------------|---|--|
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | None  |  |
| <b>Return value</b>       | Ocu_Return-Type   | Tells the caller whether the compare match will occur (or has already occurred) during the current Reference Interval, as a result of setting the new threshold value. |
| <b>Description</b>        | Service to set the value of the channel threshold using an absolute input data. |  |
| <b>Available via</b>      | Ocu.h   |  |

](SRS\_Ocu\_00010)

**[SWS\_Ocu\_00092]** [The function `Ocu_SetAbsoluteThreshold` shall set the channel threshold (the compare value) to the value given by `AbsoluteValue`. ]

(SRS\_Ocu\_00010)

**[SWS\_Ocu\_00093]** [The function `Ocu_SetAbsoluteThreshold` shall be reentrant if it is called for different channels.] ()

**[SWS\_Ocu\_00094]** [The function `Ocu_SetAbsoluteThreshold` shall be pre compile time configurable On/Off by the configuration parameter:

`OcuSetAbsoluteThresholdApi {OCU_SET_ABSOLUTE_THRESHOLD_API}.`]

(SRS\_BSW\_00171)

**[SWS\_Ocu\_00095]** [If development error detection is enabled for the OCU driver: If the driver is not initialized, the function `Ocu_SetAbsoluteThreshold` shall raise the error `OCU_E_UNINIT` and return without any action.] (SRS\_BSW\_00406, SRS\_BSW\_00386, SRS\_SPAL\_12448).

**[SWS\_Ocu\_00096]** [If development error detection is enabled for the OCU driver: If the parameter `ChannelNumber` is invalid (not within the range specified by the configuration), the function `Ocu_SetAbsoluteThreshold` shall raise the error

`OCU_E_PARAM_INVALID_CHANNEL` and return without any action.]

(SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448).

Note: `ReferenceValue` is information from the upper layer. With the combination of the `ReferenceValue` and the `AbsoluteValue` an interval (defined as 'Reference Interval', green area in the pictures below) is provided to take into account the fact that the counter is running continuously and there might be a delay between the request from a caller to update the compare threshold and the actual modification of this threshold.



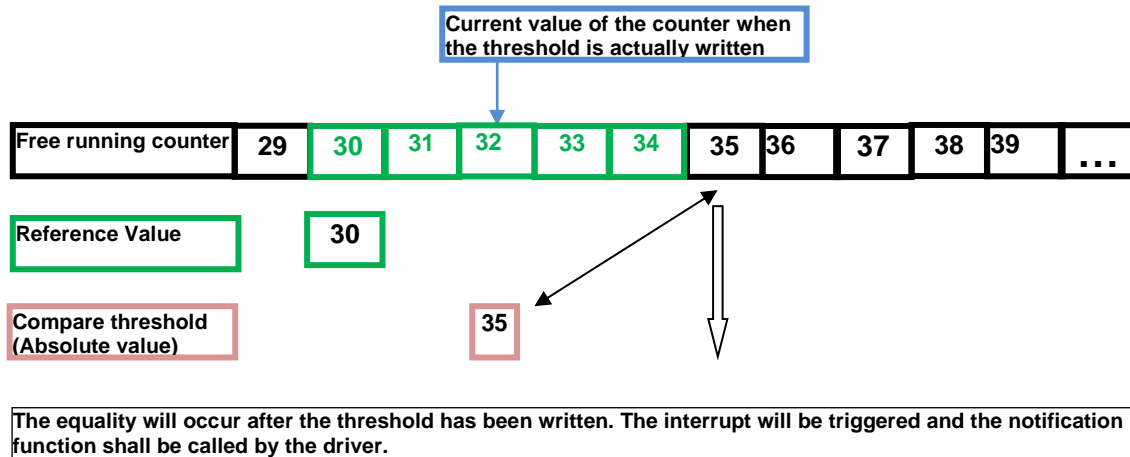
To simplify the description here, we postulate that due to internal MCU and peripheral timings the write action to a HW compare register is always done:

- before the actual compare is made, this might even be within the same clock cycle (**case1**).

- after the actual compare is made, this might even be within the same clock cycle. (**case2**).

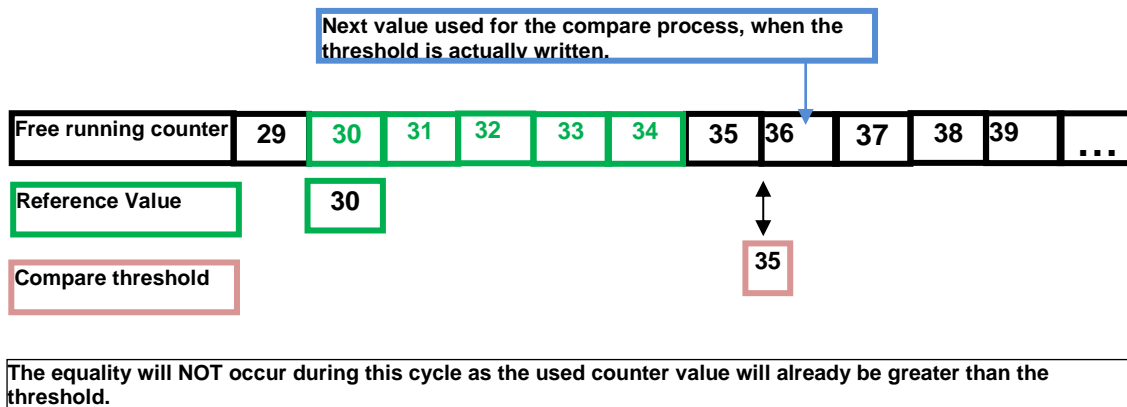
As shown with the following example `Ocu_SetAbsoluteThreshold(1,30,35);` in the pictures below.

**Case 1:** the threshold is actually written before the target compare match occurs.



**Figure 6:** threshold actually written before the target compare match occurs

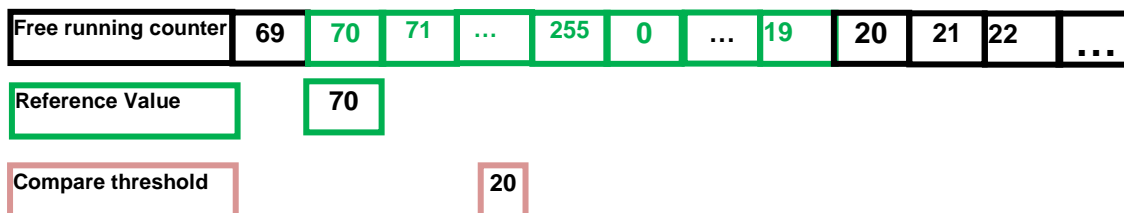
**Case 2:** the threshold is written after the targeted compare match has occurred.



**Figure 7:** threshold actually written after the target compare match has occurred

The Reference Interval takes into account the possible rollover of the counter as shown in the figure below.

Ocu\_SetAbsoluteThreshold(1,70,20);  
Example for a counter that runs from 0 to 255.



**Figure 8:** definition of a Reference Interval

As a result of the cases explained above, the expected behaviour of the driver is as follows.

- Notification to the upper layer is done only upon Compare Match (hardware): therefore there shall be a unique (at most, see further below about how to manage written threshold values) notification for each written value of the threshold during each Reference Interval.
- The API 'Ocu\_SetAbsoluteThreshold' shall return a status to inform the caller whether:
  - the writing was done inside the current Reference Interval (before actual compare match, it is even possible that the Compare Match might have already happened before the API returns) ([Case 1](#))
  - or the writing was done outside the current Reference Interval. ([Case2](#))

This status will help the caller (application) decide on how to proceed.

**[SWS\_Ocu\_00098]** [After setting a new threshold value, the API `Ocu_SetAbsoluteThreshold` shall return a status to inform the caller whether the compare match will occur (or has already occurred) during the current Reference Interval, as a result of setting the new threshold value. ] ()

For the threshold value written during the previous call of the API `Ocu_SetAbsoluteThreshold`, the expected behaviour of the driver is as follows: The previously written threshold value is erased by the current call.

Note: due to real time behaviour, the previously written threshold value might still produce a compare match; after the API has been called but the threshold value is not yet actually changed.

**[SWS\_Ocu\_00097]** [Upon actual setting of a new threshold value, the previous threshold value (written during the last call of this API) shall no longer produce a compare match. ] ()

### 8.3.9 Ocu\_SetRelativeThreshold

**[SWS\_Ocu\_00100]**

|                         |   |
|-------------------------|---|
| <b>Service Name</b>     | Ocu_SetRelativeThreshold  |
| <b>Syntax</b>           | Ocu_ReturnType Ocu_SetRelativeThreshold (<br>Ocu_ChannelType ChannelNumber,<br>Ocu_ValueType RelativeValue<br>) |
| <b>Service ID [hex]</b> | 0x08  |
| <b>Sync/Async</b>       | Synchronous   |
| <b>Reentrancy</b>       | Reentrant for different channel numbers   |

|                           |   |  |
|---------------------------|---|--|
| <b>Parameters (in)</b>    | Channel Number  | Numeric identifier of the OCU channel  |
|                           | Relative Value  | Value to use for computing the new threshold.  |
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | None  |  |
| <b>Return value</b>       | Ocu_ - Return Type  | Tells the caller whether the compare match will occur (or has already occurred) during the current Reference Interval, as a result of setting the new threshold value. |
| <b>Description</b>        | Service to set the value of the channel threshold relative to the current value of the counter. |  |
| <b>Available via</b>      | Ocu.h   |  |

](SRS\_Ocu\_00010)

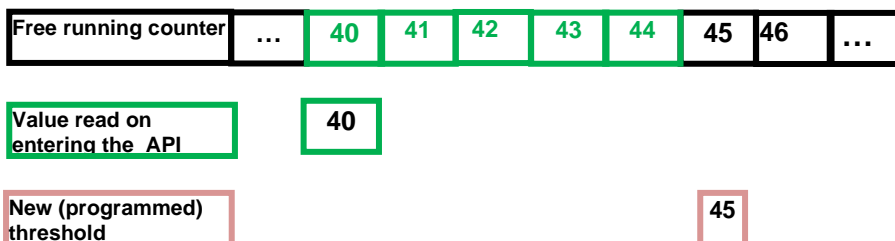
The behaviour of this API is as follows.

- On entry, the API reads the counter value (*ReadValue*). Then the new threshold value is computed and written according to the following formula:  

$$\text{NewThresholdValue} = \text{ReadValue} + \text{RelativeValue}.$$

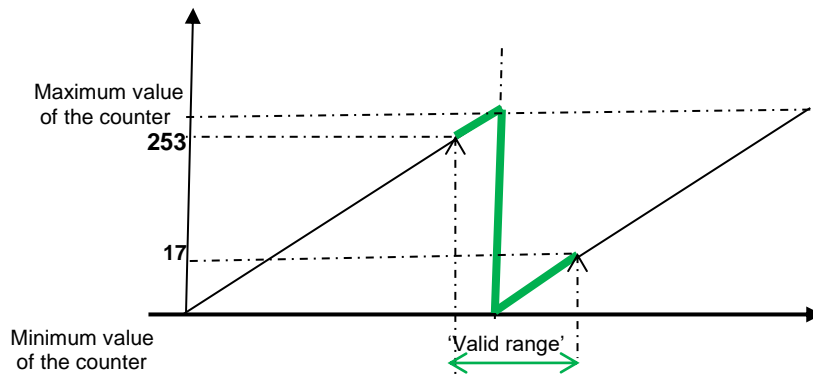
The rest of the behaviour is then the same as for the API `Ocu_SetAbsoluteThreshold` where the reference value is now `ReadValue`, and the Reference Interval is between `ReadValue` and the new programmed threshold (`NewThresholdValue`) as shown in the picture below.

Exemple with `Ocu_SetRelativeThreshold(1,5);`



Note: As for the API `Ocu_SetAbsoluteThreshold`, the possible rollover of the counter is also included in the Reference Interval as shown in the figure below.

Example with `Ocu_SetRelativeThreshold(1,20)`, with `ReadValue` equals to 253.



**Figure 9: Taking into account the roll over of the counter**

As a result, this API behaves like `Ocu_SetAbsoluteThreshold`, hence the requirements below.

**[SWS\_Ocu\_00101]** [The function `Ocu_SetRelativeThreshold` shall add `RelativeValue` to the value of the counter on entering the function to compute the new threshold relative to the counter.] ()

**[SWS\_Ocu\_00106]** [After setting a new threshold value, the API `Ocu_SetRelativeThreshold` shall return a status to inform the caller whether the compare match will occur (or has already occurred) during the current Reference Interval, as a result of setting the new threshold value.] ()

**[SWS\_Ocu\_00107]** [Upon actual setting of a new threshold value (absolute or relative), the previous threshold value shall no longer produce a compare match.] ()

**[SWS\_Ocu\_00102]** [The function `OCU Ocu_SetAbsoluteThreshold` shall be reentrant if it is called for different channels.] ()

**[SWS\_Ocu\_00103]** [The function `Ocu_SetRelativeThreshold` shall be pre compile time configurable On/Off by the configuration parameter:

`OcuSetRelativeThresholdApi {OCU_SET_RELATIVE_THRESHOLD_API}.`] (SRS\_BSW\_00171)

**[SWS\_Ocu\_00104]** [If development error detection is enabled for the OCU driver: If the driver is not initialized, the function `Ocu_SetRelativeThreshold` shall raise the error `OCU_E_UNINIT` and return without any action.] (SRS\_BSW\_00406, SRS\_BSW\_00386, SRS\_SPAL\_12448).

**[SWS\_Ocu\_00105]** [If development error detection is enabled for the OCU driver: if the parameter `ChannelNumber` is invalid (not within the range specified by the configuration), the function `Ocu_SetRelativeThreshold` shall raise the error `OCU_E_PARAM_INVALID_CHANNEL` and return without any action.] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448).

### 8.3.10 Ocu\_DisableNotification

**[SWS\_Ocu\_00108]**

|                           |   |                                       |
|---------------------------|---|---------------------------------------|
| <b>Service Name</b>       | Ocu_DisableNotification   |                                       |
| <b>Syntax</b>             | <pre>void Ocu_DisableNotification (     Ocu_ChannelType ChannelNumber )</pre> |                                       |
| <b>Service ID [hex]</b>   | 0x0a  |                                       |
| <b>Sync/Async</b>         | Synchronous   |                                       |
| <b>Reentrancy</b>         | Reentrant for different channel numbers                                       |                                       |
| <b>Parameters (in)</b>    | ChannelNumber   | Numeric identifier of the OCU channel |
| <b>Parameters (inout)</b> | None  |                                       |
| <b>Parameters (out)</b>   | None  |                                       |
| <b>Return value</b>       | None  |                                       |
| <b>Description</b>        | This service is used to disable notifications from an OCU channel.            |                                       |
| <b>Available via</b>      | Ocu.h   |                                       |

](SRS\_Ocu\_00007)

**[SWS\_Ocu\_00109]** [The function `Ocu_DisableNotification` shall disable the OCU compare match notification.] (SRS\_Ocu\_00007)

**[SWS\_Ocu\_00110]** [The fuction OCU `Ocu_DisableNotification` shall be reentrant if it is called for different channels.] ()

**[SWS\_Ocu\_00111]** [The function `Ocu_DisableNotification` shall be pre compile time configurable On/Off by the configuration parameter: `OcuNotificationSupported` {`OCU_NOTIFICATION_SUPPORTED`}.] (SRS\_BSW\_00171)

**[SWS\_Ocu\_00112]** [If development error detection is enabled for the OCU driver: If the driver is not initialized, the function `Ocu_DisableNotification` shall raise the error `OCU_E_UNINIT` and return without any action.] (SRS\_BSW\_00406, SRS\_BSW\_00386, SRS\_SPAL\_12448).

**[SWS\_Ocu\_00113]** [If development error detection is enabled for the OCU driver: If the parameter `Channel` is invalid (not within the range specified by configuration), the function `Ocu_DisableNotification` shall raise the error

`OCU_E_PARAM_INVALID_CHANNEL` and return without any action.]  
(SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448)

**[SWS\_Ocu\_00114]** [If development error detection is enabled for the OCU driver: If the notification function is the NULL pointer, the function `Ocu_DisableNotification` shall raise the error `OCU_E_NO_VALID_NOTIF` and return without any action.] () (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448).

### 8.3.11 Ocu\_EnableNotification

**[SWS\_Ocu\_00115]**

|                           |  |                                       |
|---------------------------|--|---------------------------------------|
| <b>Service Name</b>       | Ocu_EnableNotification   |                                       |
| <b>Syntax</b>             | <pre>void Ocu_EnableNotification (     Ocu_ChannelType ChannelNumber )</pre> |                                       |
| <b>Service ID [hex]</b>   | 0x0b   |                                       |
| <b>Sync/Async</b>         | Synchronous  |                                       |
| <b>Reentrancy</b>         | Reentrant for different channel numbers                                      |                                       |
| <b>Parameters (in)</b>    | ChannelNumber  | Numeric identifier of the OCU channel |
| <b>Parameters (inout)</b> | None   |                                       |
| <b>Parameters (out)</b>   | None   |                                       |
| <b>Return value</b>       | None   |                                       |
| <b>Description</b>        | This service is used to enable notifications from an OCU channel.            |                                       |
| <b>Available via</b>      | Ocu.h  |                                       |

](SRS\_Ocu\_00007)

**[SWS\_Ocu\_00116]** [The function `Ocu_EnableNotification` shall enable the OCU compare match notification of the indexed channel.] (SRS\_Ocu\_00007)

**[SWS\_Ocu\_00117]** [The function `Ocu_EnableNotification` shall be reentrant if it is called for different channels.] ()

**[SWS\_Ocu\_00118]** [The function `Ocu_EnableNotification` shall be pre compile time configurable On/Off by the configuration parameter: `OcuNotificationSupported` {`OCU_NOTIFICATION_SUPPORTED`}.] (SRS\_BSW\_00171)

**[SWS\_Ocu\_00119]** [If development error detection is enabled for the OCU driver: If the driver is not initialized, the function `Ocu_EnableNotification` shall raise the error `OCU_E_UNINIT` and return without any action.] (SRS\_BSW\_00406, SRS\_BSW\_00386, SRS\_SPAL\_12448).

**[SWS\_Ocu\_00120]** [If development error detection is enabled for the OCU driver: If the parameter `Channel` is invalid (not within the range specified by configuration), then the function `Ocu_EnableNotification` shall raise the error `OCU_E_PARAM_INVALID_CHANNEL` and return without any action.] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448)

**[SWS\_Ocu\_00121]** [If development error detection is enabled for the OCU driver: If the notification function is the NULL pointer, the function `Ocu_EnableNotification` shall raise the error `OCU_E_NO_VALID_NOTIF` and return without any action.] () (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448).

### 8.3.12 Ocu\_GetVersionInfo

**[SWS\_Ocu\_00122]**

|                           |   |  |
|---------------------------|---|--|
| <b>Service Name</b>       | Ocu_GetVersionInfo  |  |
| <b>Syntax</b>             | <pre>void Ocu_GetVersionInfo (     Std_VersionInfoType* versioninfo )</pre> |  |
| <b>Service ID [hex]</b>   | 0x09  |  |
| <b>Sync/Async</b>         | Synchronous   |  |
| <b>Reentrancy</b>         | Reentrant   |  |
| <b>Parameters (in)</b>    | None  |  |
| <b>Parameters (inout)</b> | None  |  |
| <b>Parameters (out)</b>   | versioninfo   | Pointer to where to store the version information of this module |
| <b>Return value</b>       | None  |  |
| <b>Description</b>        | This service returns the version information of this module.                |  |
| <b>Available via</b>      | Ocu.h   |  |

()



**[SWS\_Ocu\_00123]** [The function `Ocu_GetVersionInfo` shall return the version information of this module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers.] (SRS\_BSW\_00407)

**[SWS\_Ocu\_00124]** [The function `Ocu_GetVersionInfo` shall be pre compile time configurable On/Off by the configuration parameter: `OcuVersionInfoApi` {`OCU_VERSION_INFO_API`}.] (SRS\_BSW\_00407, SRS\_BSW\_00411)

**[SWS\_Ocu\_00125]** [If source code for caller and callee of `Ocu_GetVersionInfo` is available; the OCU driver should realize `Ocu_GetVersionInfo` as a macro, defined in the module's header file.] ()

**[SWS\_Ocu\_00126]** [If development error detection is enabled for the OCU driver, the function `Ocu_GetVersionInfo` shall raise development error `OCU_E_PARAM_POINTER` if parameter `versioninfo` is a null pointer, and return without any action] (SRS\_BSW\_00323, SRS\_BSW\_00386, SRS\_SPAL\_12448).

## 8.4 Callback notifications

Since the OCU Driver is a module on the lowest architectural layer it doesn't provide any call-back functions for lower layer modules.

## 8.5 Scheduled functions

The OCU driver offers only synchronous services and therefore doesn't need any scheduled functions.

## 8.6 Expected Interfaces

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

### 8.6.1 Mandatory Interfaces

This module does not require any mandatory interfaces.

### 8.6.2 Optional Interfaces

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

[SWS\_Ocu\_00127]

| API Function    | Header File | Description                           |
|-----------------|-------------|---------------------------------------|
| Det_ReportError | Det.h       | Service to report development errors. |

]()

### 8.6.3 Configurable interfaces

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

[SWS\_Ocu\_00128]

|                           |   |
|---------------------------|---|
| <b>Service Name</b>       | Ocu_Notification_<Channel>  |
| <b>Syntax</b>             | void Ocu_Notification_<Channel> (<br>void<br>)  |
| <b>Sync/Async</b>         | Synchronous   |
| <b>Reentrancy</b>         | Reentrancy of this API call depends on the user code  |
| <b>Parameters (in)</b>    | None  |
| <b>Parameters (inout)</b> | None  |
| <b>Parameters (out)</b>   | None  |
| <b>Return value</b>       | None  |
| <b>Description</b>        | This notification function is called when a compare match occurs on the associated channel. |
| <b>Available via</b>      | <none>  |

[(SRS\_BSW\_00359, SRS\_BSW\_00360, SRS\_SPAL\_00157)

The notification prototype `Ocu_Notification_<channel#>` is for the notification callback function provided by the upper layer and shall be implemented by the user.

[SWS\_Ocu\_00129] [The OCU driver shall call the function `Ocu_Notification_<Channel#>` according to the last call of `Ocu_EnableNotification/Ocu_DisableNotification` for channel `<Channel#>`, if there's a compare match on that channel.] (SRS\_SPAL\_00157)

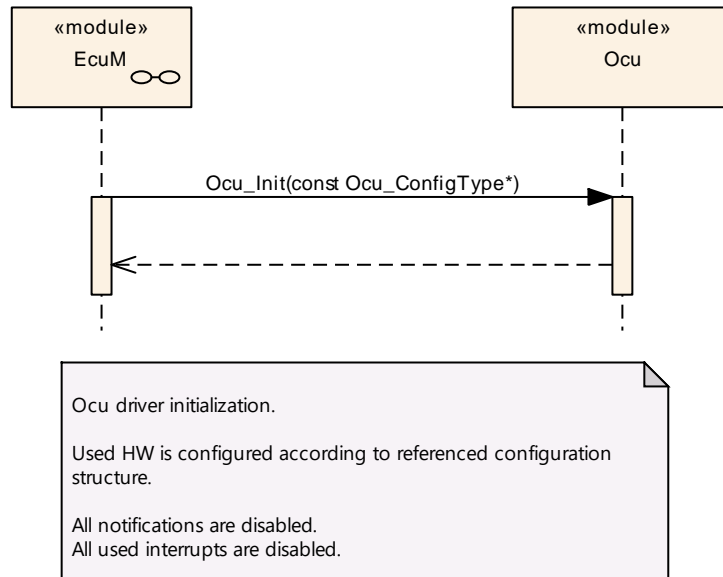
**[SWS\_Ocu\_00130]** [The OCU driver shall reset the interrupt flag (if needed by hardware) associated with the notification Ocu\_Notification\_<Channel#>] (SRS\_SPAL\_12129)

**[SWS\_Ocu\_00132]** [ [If the NULL pointer is configured for a notification call-back, then no call-back shall be executed.] (SRS\_SPAL\_12056)

**[SWS\_Ocu\_00133]** [When the notification mechanism is disabled, the OCU driver shall send no notification.] ()

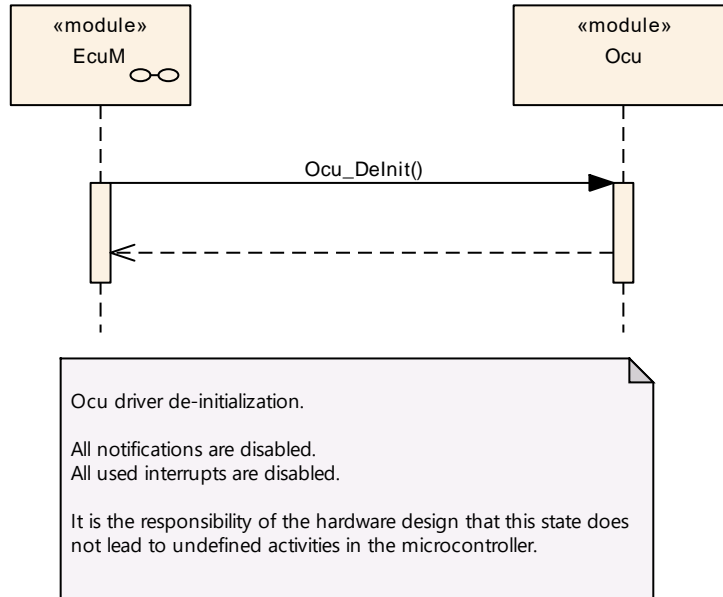
## 9 Sequence and Timing diagrams

### 9.1 Initialization



**Figure 10: Ocu initialization**

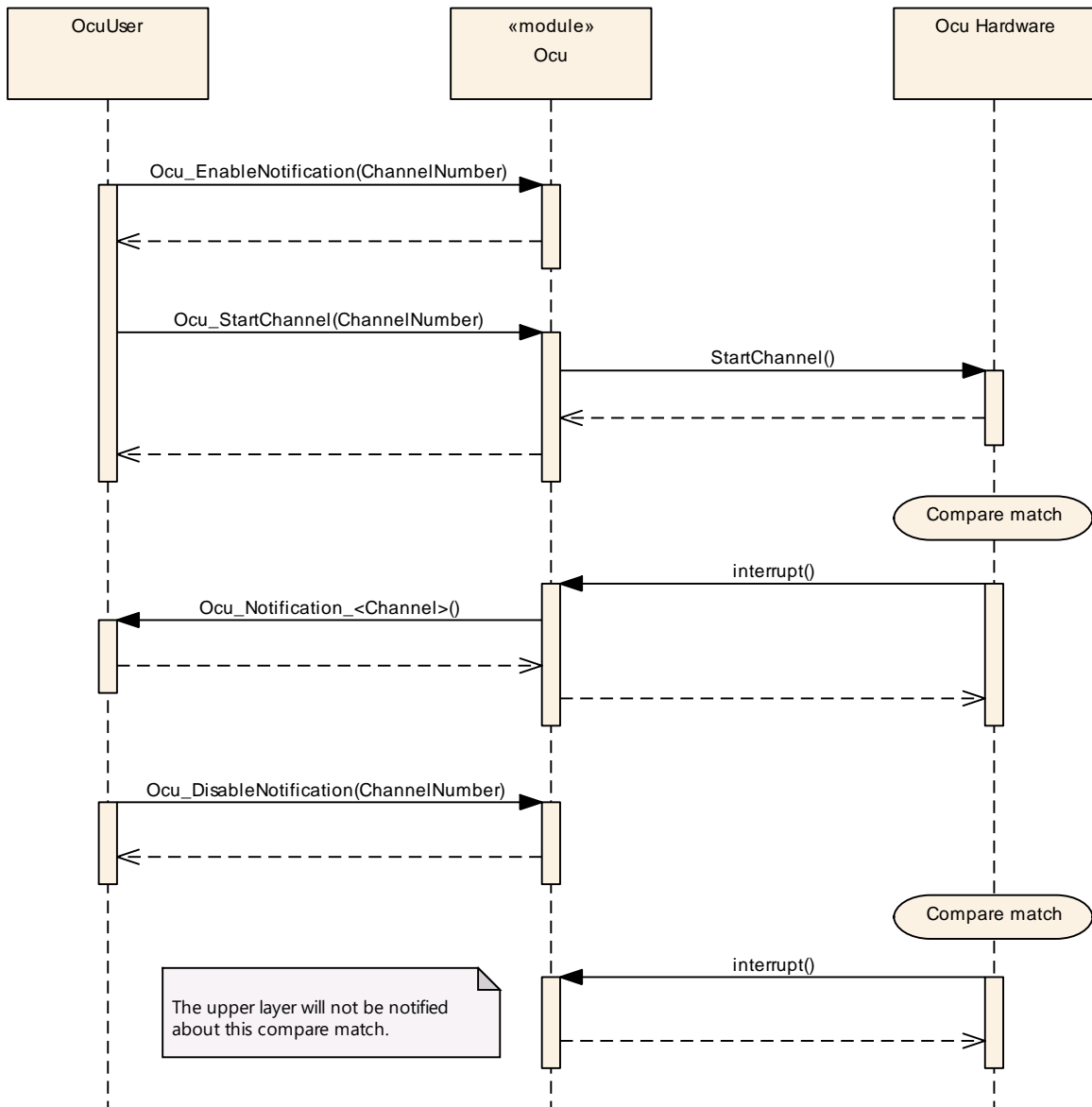
## 9.2 De-initialization



**Figure 11: Ocu de-initialization**

### 9.3 Using the Ocu Notifications

g



**Figure 12: Enable/Disable Notifications**

### 9.4 Ocu\_SetPinState

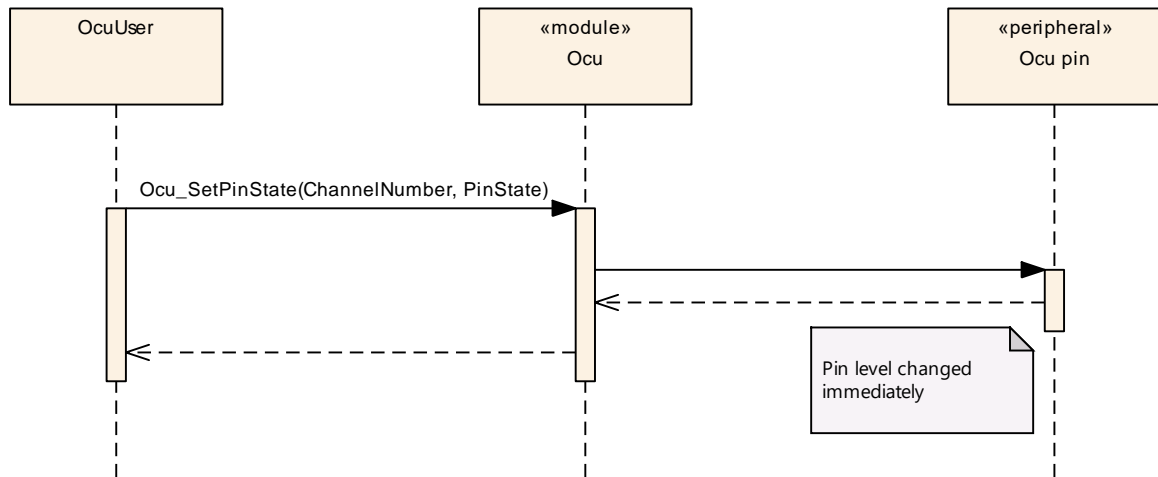


Figure 13: Ocu driver sets the pin state

### 9.5 Ocu\_SetPinAction

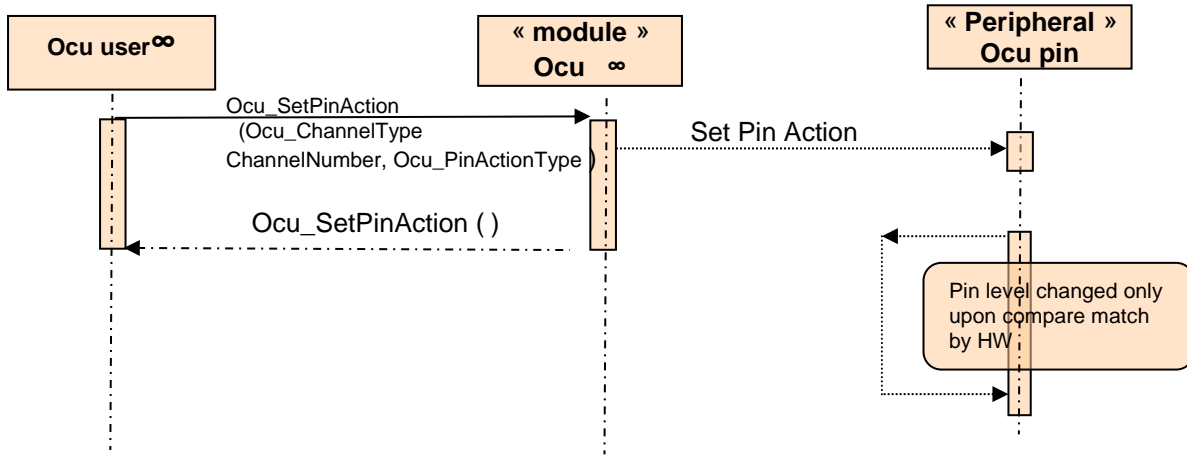


Figure 14: Change the pin state upon compare match

### 9.6 Setting a new compare threshold

Refer to the chapters [Ocu\\_SetAbsoluteThreshold](#) and [Ocu\\_SetRelativeThreshold](#).



## 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 OCU Driver.

Chapter 10.3 specifies published information of the module OCU Driver.

### 10.1 How to read this chapter

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

- AUTOSAR Layered Software Architecture [1]
- AUTOSAR ECU Configuration Specification [6]  
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 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.

### 10.1.3 Specification template for configuration parameters

The following tables consist of three sections:

- the general section
- the configuration parameter section
- the section of included/referenced containers

Pre-compile time - specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

| Label | Description   |
|-------|---|
| x     | The configuration parameter shall be of configuration class <i>Pre-compile time</i> .       |
| --    | The configuration parameter shall never be of configuration class <i>Pre-compile time</i> . |

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

| Label | Description  |
|-------|--|
| x     | The configuration parameter shall be of configuration class <i>Link time</i> .       |
| --    | The configuration parameter shall never be of configuration class <i>Link time</i> . |

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

| Label | Description  |
|-------|--|
| x     | The configuration parameter shall be of configuration class <i>Post Build</i> and no specific implementation is required.  |
| L     | <i>Loadable</i> - the configuration parameter shall be of configuration class <i>Post Build</i> and only one configuration parameter set resides in the ECU.   |
| M     | <i>Multiple</i> - the configuration parameter shall be of configuration class <i>Post Build</i> and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module. |
| --    | The configuration parameter shall never be of configuration class <i>Post Build</i> .  |

## 10.2 Containers and configuration parameters

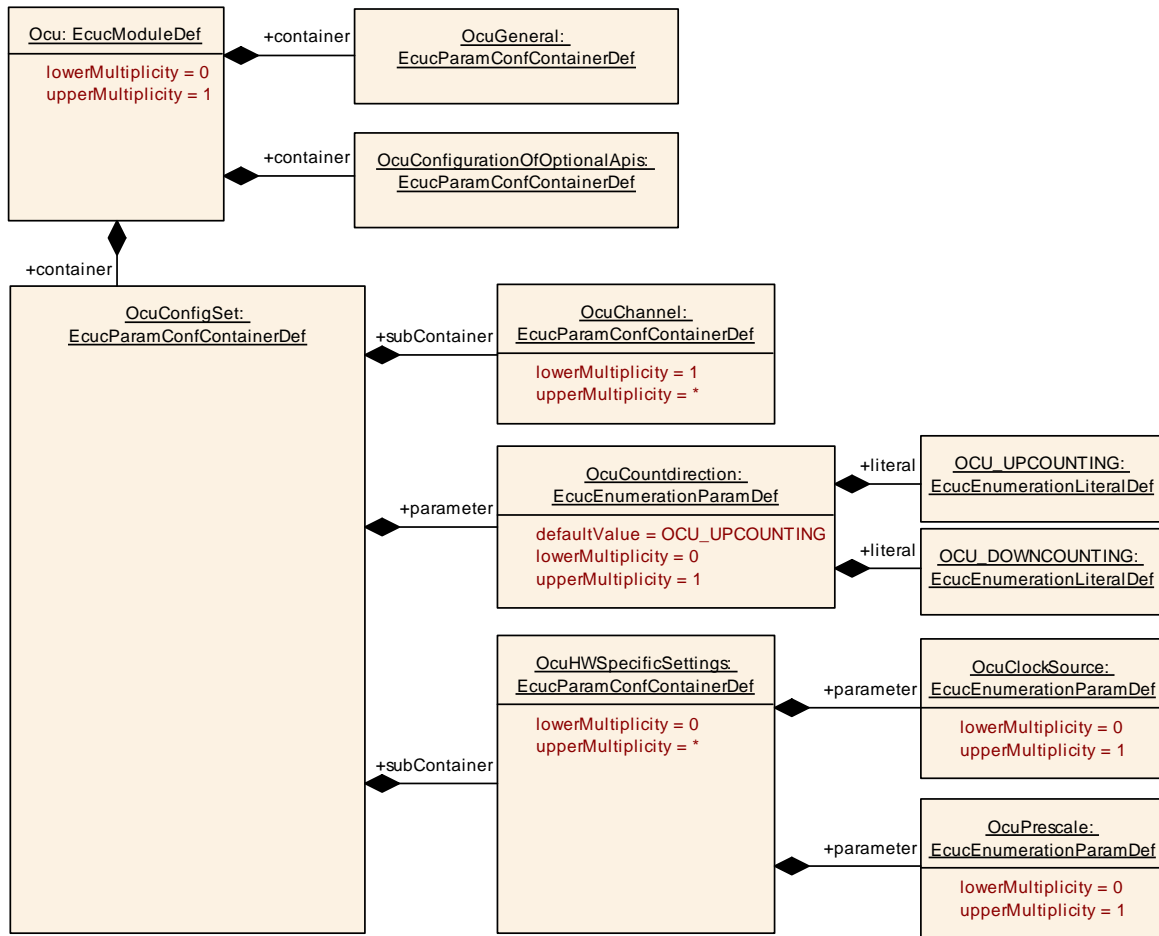
The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe the chapters: [Functional specification](#) and [API specification](#).

[SWS\_Ocu\_00170] 「The OCU module shall reject configurations with partition mappings which are not supported by the implementation.」()

### 10.2.1 Ocu

|                                   |  |
|-----------------------------------|--|
| <b>SWS Item</b>                   | ECUC_Ocu_00136 :                                   |
| <b>Module Name</b>                | Ocu  |
| <b>Module Description</b>         | Configuration of Ocu (Output Compare Unit) module. |
| <b>Post-Build Variant Support</b> | true   |
| <b>Supported Config Variants</b>  | VARIANT-POST-BUILD, VARIANT-PRE-COMPILE            |

| Included Containers            |              |  |
|--------------------------------|--------------|--|
| Container Name                 | Multiplicity | Scope / Dependency   |
| OcuConfigSet                   | 1            | This container is the base of a Configuration Set, which contains the configured OCU channels. This way, different configuration sets can be defined for post-build process. |
| OcuConfigurationOfOptionalApis | 1            | Configuration of optional APIs.  |
| OcuGeneral                     | 1            | This container contains the module-wide configuration parameters of the OCU Driver.  |



## 10.2.2 OcuGeneral

|                                 |   |  |  |
|---------------------------------|---|--|--|
| <b>SWS Item</b>                 | <b>ECUC_Ocu_00137 :</b>   |  |  |
| <b>Container Name</b>           | OcuGeneral  |  |  |
| <b>Parent Container</b>         | Ocu   |  |  |
| <b>Description</b>              | This container contains the module-wide configuration parameters of the OCU Driver. |  |  |
| <b>Configuration Parameters</b> |   |  |  |

|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Ocu_00138 :</b>   |    |              |
| <b>Name</b>                      | OcuDevErrorDetect   |    |              |
| <b>Parent Container</b>          | OcuGeneral  |    |              |
| <b>Description</b>               | Switches the development error detection and notification on or off. <ul style="list-style-type: none"> <li>• true: detection and notification is enabled.</li> <li>• false: detection and notification is disabled.</li> </ul> |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucBooleanParamDef   |    |              |
| <b>Default value</b>             | false   |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                                  | <b>Link time</b>  | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: local  |    |              |

|   |  |    |              |
|---|--|----|--------------|
| <b>SWS Item</b>                         | <b>ECUC_Ocu_00167 :</b>  |    |              |
| <b>Name</b>                             | OcuEcucPartitionRef  |    |              |
| <b>Parent Container</b>                 | OcuGeneral   |    |              |
| <b>Description</b>                      | Maps the OCU driver to zero or multiple ECUC partitions to make the driver API available in the according partition. |    |              |
| <b>Multiplicity</b>                     | 0..*   |    |              |
| <b>Type</b>                             | Reference to [ EcucPartition ]   |    |              |
| <b>Post-Build Variant Multiplicity</b>  | true   |    |              |
| <b>Post-Build Variant Value</b>         | true   |    |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|   | <b>Link time</b>   | -- |              |
|   | <b>Post-build time</b>   | -- |              |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X  | All Variants |
|   | <b>Link time</b>   | -- |              |
|   | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b>               | scope: ECU   |    |              |

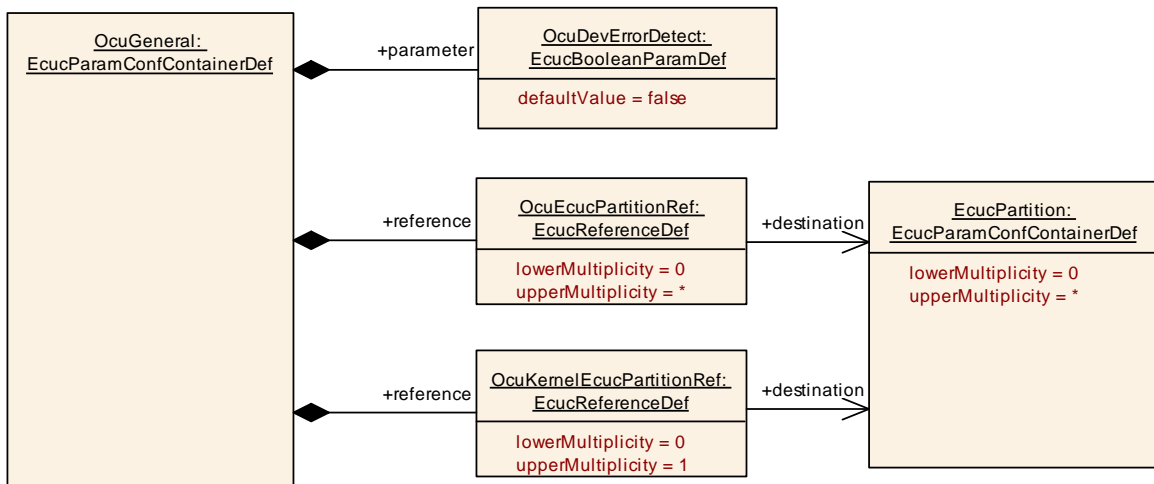
|   |   |    |              |
|---|---|----|--------------|
| <b>SWS Item</b>                         | <b>ECUC_Ocu_00168 :</b>   |    |              |
| <b>Name</b>                             | OcuKernelEcucPartitionRef   |    |              |
| <b>Parent Container</b>                 | OcuGeneral  |    |              |
| <b>Description</b>                      | Maps the OCU kernel to zero or one ECUC partitions to assign the driver kernel to a certain core. The ECUC partition referenced is a subset of the ECUC partitions where the OCU driver is mapped to. |    |              |
| <b>Multiplicity</b>                     | 0..1  |    |              |
| <b>Type</b>                             | Reference to [ EcucPartition ]  |    |              |
| <b>Post-Build Variant Multiplicity</b>  | true  |    |              |
| <b>Post-Build Variant Value</b>         | true  |    |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>   | X  | All Variants |
|   | <b>Link time</b>  | -- |              |

|                                  |                         |    |              |
|----------------------------------|-------------------------|----|--------------|
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b> | X  | All Variants |
|                                  | <b>Link time</b>        | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: ECU              |    |              |

**No Included Containers**

**[SWS\_Ocu\_CONSTR\_00001]** [The ECUC partitions referenced by OcuKernelEcucPartitionRef shall be a subset of the ECUC partitions referenced by OcuEcucPartitionRef.]()

**[SWS\_Ocu\_CONSTR\_00004]** [ If OcuEcucPartitionRef references one or more ECUC partitions, OcuKernelEcucPartitionRef shall have a multiplicity of one and reference one of these ECUC partitions as well]()



### 10.2.3 OcuConfigurationOfOptionalApis

|                                 |                                 |  |  |
|---------------------------------|---------------------------------|--|--|
| <b>SWS Item</b>                 | <b>ECUC_Ocu_00139 :</b>         |  |  |
| <b>Container Name</b>           | OcuConfigurationOfOptionalApis  |  |  |
| <b>Parent Container</b>         | Ocu                             |  |  |
| <b>Description</b>              | Configuration of optional APIs. |  |  |
| <b>Configuration Parameters</b> |                                 |  |  |

|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Ocu_00140 :</b>                                 |    |              |
| <b>Name</b>                      | OcuDelInitApi   |    |              |
| <b>Parent Container</b>          | OcuConfigurationOfOptionalApis                          |    |              |
| <b>Description</b>               | Adds / removes the service Ocu_DelInit() from the code. |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucBooleanParamDef                                     |    |              |
| <b>Default value</b>             | --  |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                                 | X  | All Variants |
|                                  | <b>Link time</b>  | -- |              |
|                                  | <b>Post-build time</b>                                  | -- |              |
| <b>Scope / Dependency</b>        | scope: local  |    |              |

|                                  |  |    |              |
|----------------------------------|--|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Ocu_00141 :</b>                                    |    |              |
| <b>Name</b>                      | OcuGetCounterApi   |    |              |
| <b>Parent Container</b>          | OcuConfigurationOfOptionalApis                             |    |              |
| <b>Description</b>               | Adds / removes the service Ocu_GetCounter() from the code. |    |              |
| <b>Multiplicity</b>              | 1  |    |              |
| <b>Type</b>                      | EcucBooleanParamDef  |    |              |
| <b>Default value</b>             | --   |    |              |
| <b>Post-Build Variant Value</b>  | false  |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                                    | X  | All Variants |
|                                  | <b>Link time</b>   | -- |              |
|                                  | <b>Post-build time</b>                                     | -- |              |
| <b>Scope / Dependency</b>        | scope: local   |    |              |

|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Ocu_00142 :</b>   |    |              |
| <b>Name</b>                      | OcuNotificationSupported  |    |              |
| <b>Parent Container</b>          | OcuConfigurationOfOptionalApis  |    |              |
| <b>Description</b>               | Adds / removes the services Ocu_EnableNotification() and Ocu_DisableNotification() from the code. |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucBooleanParamDef   |    |              |
| <b>Default value</b>             | --  |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                                  | <b>Link time</b>  | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: local  |    |              |

|                                 |  |  |  |
|---------------------------------|--|--|--|
| <b>SWS Item</b>                 | <b>ECUC_Ocu_00143 :</b>  |  |  |
| <b>Name</b>                     | OcuSetAbsoluteThresholdApi   |  |  |
| <b>Parent Container</b>         | OcuConfigurationOfOptionalApis                                       |  |  |
| <b>Description</b>              | Adds / removes the service Ocu_SetAbsoluteThreshold() from the code. |  |  |
| <b>Multiplicity</b>             | 1  |  |  |
| <b>Type</b>                     | EcucBooleanParamDef  |  |  |
| <b>Default value</b>            | --   |  |  |
| <b>Post-Build Variant Value</b> | false  |  |  |

|                                  |                         |    |              |
|----------------------------------|-------------------------|----|--------------|
| <b>Value Configuration Class</b> | <b>Pre-compile time</b> | X  | All Variants |
|                                  | <b>Link time</b>        | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: local            |    |              |

|                                  |  |    |              |
|----------------------------------|--|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Ocu_00144 :</b>                                      |    |              |
| <b>Name</b>                      | OcuSetPinActionApi   |    |              |
| <b>Parent Container</b>          | OcuConfigurationOfOptionalApis                               |    |              |
| <b>Description</b>               | Adds / removes the service Ocu_SetPinAction() from the code. |    |              |
| <b>Multiplicity</b>              | 1  |    |              |
| <b>Type</b>                      | EcucBooleanParamDef  |    |              |
| <b>Default value</b>             | --   |    |              |
| <b>Post-Build Variant Value</b>  | false  |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                                      | X  | All Variants |
|                                  | <b>Link time</b>   | -- |              |
|                                  | <b>Post-build time</b>                                       | -- |              |
| <b>Scope / Dependency</b>        | scope: local   |    |              |

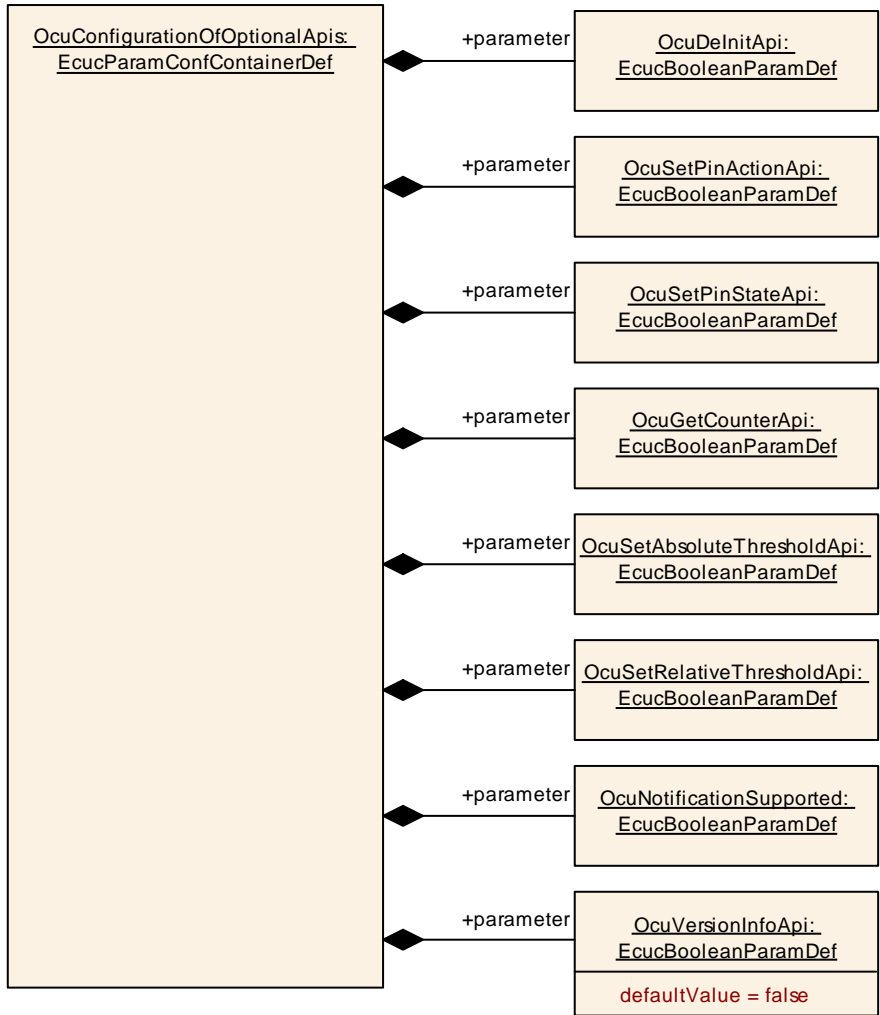
|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Ocu_00145 :</b>                                     |    |              |
| <b>Name</b>                      | OcuSetPinStateApi   |    |              |
| <b>Parent Container</b>          | OcuConfigurationOfOptionalApis                              |    |              |
| <b>Description</b>               | Adds / removes the service Ocu_SetPinState() from the code. |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucBooleanParamDef   |    |              |
| <b>Default value</b>             | --  |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                                     | X  | All Variants |
|                                  | <b>Link time</b>  | -- |              |
|                                  | <b>Post-build time</b>                                      | -- |              |
| <b>Scope / Dependency</b>        | scope: local  |    |              |

|                                  |  |    |              |
|----------------------------------|--|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Ocu_00146 :</b>  |    |              |
| <b>Name</b>                      | OcuSetRelativeThresholdApi   |    |              |
| <b>Parent Container</b>          | OcuConfigurationOfOptionalApis                                       |    |              |
| <b>Description</b>               | Adds / removes the service Ocu_SetRelativeThreshold() from the code. |    |              |
| <b>Multiplicity</b>              | 1  |    |              |
| <b>Type</b>                      | EcucBooleanParamDef  |    |              |
| <b>Default value</b>             | --   |    |              |
| <b>Post-Build Variant Value</b>  | false  |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|                                  | <b>Link time</b>   | -- |              |
|                                  | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b>        | scope: local   |    |              |

|                                  |  |    |              |
|----------------------------------|--|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Ocu_00147 :</b>  |    |              |
| <b>Name</b>                      | OcuVersionInfoApi  |    |              |
| <b>Parent Container</b>          | OcuConfigurationOfOptionalApis                                 |    |              |
| <b>Description</b>               | Switch to indicate that the Ocu_GetVersionInfo() is supported. |    |              |
| <b>Multiplicity</b>              | 1  |    |              |
| <b>Type</b>                      | EcucBooleanParamDef  |    |              |
| <b>Default value</b>             | false  |    |              |
| <b>Post-Build Variant Value</b>  | false  |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|                                  | <b>Link time</b>   | -- |              |
|                                  | <b>Post-build time</b>   | -- |              |

|                           |              |
|---------------------------|--------------|
| <b>Scope / Dependency</b> | scope: local |
|---------------------------|--------------|

**No Included Containers**



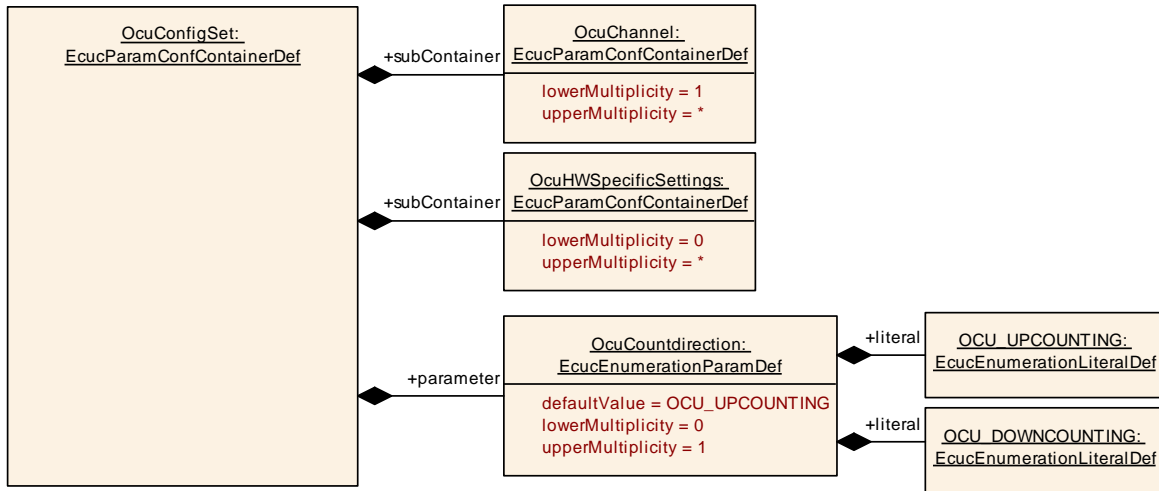


### 10.2.4 OcuConfigSet

|                                 |  |
|---------------------------------|--|
| <b>SWS Item</b>                 | <b>ECUC_Ocu_00148 :</b>  |
| <b>Container Name</b>           | OcuConfigSet   |
| <b>Parent Container</b>         | Ocu  |
| <b>Description</b>              | This container is the base of a Configuration Set, which contains the configured OCU channels. This way, different configuration sets can be defined for post-build process. |
| <b>Configuration Parameters</b> |  |

|   |  |    |  |
|---|--|----|--|
| <b>SWS Item</b>                         | <b>ECUC_Ocu_00149 :</b>  |    |  |
| <b>Name</b>                             | OcuCountdirection  |    |  |
| <b>Parent Container</b>                 | OcuConfigSet   |    |  |
| <b>Description</b>                      | This parameter indicates the count direction for the whole OCU driver. |    |  |
| <b>Multiplicity</b>                     | 0..1   |    |  |
| <b>Type</b>                             | EcucEnumerationParamDef  |    |  |
| <b>Range</b>                            | OCU_DOWNCOUNTING   |    | The OCU counter will reckon from the maximum to the minimum value. |
|   | OCU_UPCOUNTING   |    | The OCU counter will reckon from the minimum to the maximum value. |
| <b>Default value</b>                    | OCU_UPCOUNTING   |    |  |
| <b>Post-Build Variant Multiplicity</b>  | true   |    |  |
| <b>Post-Build Variant Value</b>         | true   |    |  |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X  | VARIANT-PRE-COMPILE  |
|   | <b>Link time</b>   | -- |  |
|   | <b>Post-build time</b>   | X  | VARIANT-POST-BUILD   |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X  | VARIANT-PRE-COMPILE  |
|   | <b>Link time</b>   | -- |  |
|   | <b>Post-build time</b>   | X  | VARIANT-POST-BUILD   |
| <b>Scope / Dependency</b>               | scope: local   |    |  |

|                            |                     |  |
|----------------------------|---------------------|--|
| <b>Included Containers</b> |                     |  |
| <b>Container Name</b>      | <b>Multiplicity</b> | <b>Scope / Dependency</b>  |
| OcuChannel                 | 1..*                | Configuration of an individual OCU channel.  |
| OcuHWSpecificSettings      | 0..*                | This container contains Ocu-specific parameters for selecting the clock source and setting optional prescalers if supported by hardware.<br>Implementation is defined vendor specific. |



### 10.2.5 OcuChannel

|                                 |   |  |  |
|---------------------------------|---|--|--|
| <b>SWS Item</b>                 | <b>ECUC_Ocu_00150 :</b>                     |  |  |
| <b>Container Name</b>           | OcuChannel                                  |  |  |
| <b>Parent Container</b>         | OcuConfigSet                                |  |  |
| <b>Description</b>              | Configuration of an individual OCU channel. |  |  |
| <b>Configuration Parameters</b> |   |  |  |

|                                  |   |    |                     |
|----------------------------------|---|----|---------------------|
| <b>SWS Item</b>                  | <b>ECUC_Ocu_00151 :</b>   |    |                     |
| <b>Name</b>                      | OcuAssignedHardwareChannel  |    |                     |
| <b>Parent Container</b>          | OcuChannel  |    |                     |
| <b>Description</b>               | The physical hardware channel that is assigned to this logical channel. |    |                     |
| <b>Multiplicity</b>              | 1   |    |                     |
| <b>Type</b>                      | EcucIntegerParamDef   |    |                     |
| <b>Range</b>                     | 0 .. 255  |    |                     |
| <b>Default value</b>             | --  |    |                     |
| <b>Post-Build Variant Value</b>  | true  |    |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X  | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>  | -- |                     |
|                                  | <b>Post-build time</b>  | X  | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>        | scope: local  |    |                     |

|                                  |  |    |              |
|----------------------------------|--|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Ocu_00152 :</b>  |    |              |
| <b>Name</b>                      | OcuChannelId   |    |              |
| <b>Parent Container</b>          | OcuChannel   |    |              |
| <b>Description</b>               | Channel Id of the OCU channel. This value will be assigned to the symbolic name derived from the OcuChannel container short name. It defines the assignment of the channel to the physical OCU hardware channel. |    |              |
| <b>Multiplicity</b>              | 1  |    |              |
| <b>Type</b>                      | EcucIntegerParamDef (Symbolic Name generated for this parameter)   |    |              |
| <b>Range</b>                     | 0 .. 65535   |    |              |
| <b>Default value</b>             | --   |    |              |
| <b>Post-Build Variant Value</b>  | false  |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|                                  | <b>Link time</b>   | -- |              |
|                                  | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b>        | scope: local   |    |              |

|                                  |   |    |                     |
|----------------------------------|---|----|---------------------|
| <b>SWS Item</b>                  | <b>ECUC_Ocu_00153 :</b>   |    |                     |
| <b>Name</b>                      | OcuChannelTickDuration  |    |                     |
| <b>Parent Container</b>          | OcuChannel  |    |                     |
| <b>Description</b>               | Specifies the number of input clock edges (rising or falling edges) required to increase the channel counter by one (i.e. one counter tick). The value range depends on the used HW, not all values may be relevant |    |                     |
| <b>Multiplicity</b>              | 1   |    |                     |
| <b>Type</b>                      | EcucIntegerParamDef   |    |                     |
| <b>Range</b>                     | 1 .. 32768  |    |                     |
| <b>Default value</b>             | --  |    |                     |
| <b>Post-Build Variant Value</b>  | true  |    |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X  | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>  | -- |                     |
|                                  | <b>Post-build time</b>  | X  | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>        | scope: local  |    |                     |

|                 |                         |  |  |
|-----------------|-------------------------|--|--|
| <b>SWS Item</b> | <b>ECUC_Ocu_00154 :</b> |  |  |
| <b>Name</b>     | OcuDefaultThreshold     |  |  |

|                                  |  |    |                     |
|----------------------------------|--|----|---------------------|
| <b>Parent Container</b>          | OcuChannel   |    |                     |
| <b>Description</b>               | Value of comparison threshold used for Initialization. |    |                     |
| <b>Multiplicity</b>              | 1  |    |                     |
| <b>Type</b>                      | EcucIntegerParamDef                                    |    |                     |
| <b>Range</b>                     | 0 .. 4294967295  |    |                     |
| <b>Default value</b>             | --   |    |                     |
| <b>Post-Build Variant Value</b>  | true   |    |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>                                | X  | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>                                       | -- |                     |
|                                  | <b>Post-build time</b>                                 | X  | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>        | scope: local   |    |                     |

|   |  |    |                     |
|---|--|----|---------------------|
| <b>SWS Item</b>                         | <b>ECUC_Ocu_00155 :</b>  |    |                     |
| <b>Name</b>                             | OcuHardwareTriggeredAdc  |    |                     |
| <b>Parent Container</b>                 | OcuChannel   |    |                     |
| <b>Description</b>                      | This parameter is used to allow the OCU channel to trigger an ADC channel upon compare match, if this is supported by hardware. The value of the parameter represents the ADC physical channel to trigger. |    |                     |
| <b>Multiplicity</b>                     | 0..1   |    |                     |
| <b>Type</b>                             | EcucIntegerParamDef  |    |                     |
| <b>Range</b>                            | 0 .. 255   |    |                     |
| <b>Default value</b>                    | 0  |    |                     |
| <b>Post-Build Variant Multiplicity</b>  | true   |    |                     |
| <b>Post-Build Variant Value</b>         | true   |    |                     |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X  | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>   | -- |                     |
|   | <b>Post-build time</b>   | X  | VARIANT-POST-BUILD  |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X  | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>   | -- |                     |
|   | <b>Post-build time</b>   | X  | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>               | scope: local   |    |                     |

|   |   |    |                     |
|---|---|----|---------------------|
| <b>SWS Item</b>                         | <b>ECUC_Ocu_00156 :</b>   |    |                     |
| <b>Name</b>                             | OcuHardwareTriggeredDMA   |    |                     |
| <b>Parent Container</b>                 | OcuChannel  |    |                     |
| <b>Description</b>                      | This parameter is used to allow the OCU channel to trigger a DMA channel upon compare match, if this is supported by hardware. The value of the parameter represents the DMA physical channel to trigger. |    |                     |
| <b>Multiplicity</b>                     | 0..1  |    |                     |
| <b>Type</b>                             | EcucIntegerParamDef   |    |                     |
| <b>Range</b>                            | 0 .. 255  |    |                     |
| <b>Default value</b>                    | 0   |    |                     |
| <b>Post-Build Variant Multiplicity</b>  | true  |    |                     |
| <b>Post-Build Variant Value</b>         | true  |    |                     |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>   | X  | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>  | -- |                     |
|   | <b>Post-build time</b>  | X  | VARIANT-POST-BUILD  |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>   | X  | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>  | -- |                     |
|   | <b>Post-build time</b>  | X  | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>               | scope: local  |    |                     |

|                 |                         |  |  |
|-----------------|-------------------------|--|--|
| <b>SWS Item</b> | <b>ECUC_Ocu_00157 :</b> |  |  |
| <b>Name</b>     | OcuMaxCounterValue      |  |  |

|                                  |  |    |                     |
|----------------------------------|--|----|---------------------|
| <b>Parent Container</b>          | OcuChannel   |    |                     |
| <b>Description</b>               | Maximum value in ticks, the counter of the OCU channel is able to count. |    |                     |
| <b>Multiplicity</b>              | 1  |    |                     |
| <b>Type</b>                      | EcucIntegerParamDef  |    |                     |
| <b>Range</b>                     | 1 .. 4294967295  |    |                     |
| <b>Default value</b>             | --   |    |                     |
| <b>Post-Build Variant Value</b>  | true   |    |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X  | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>   | -- |                     |
|                                  | <b>Post-build time</b>   | X  | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>        | scope: local   |    |                     |

|   |  |    |                     |
|---|--|----|---------------------|
| <b>SWS Item</b>                         | <b>ECUC_Ocu_00158 :</b>                                  |    |                     |
| <b>Name</b>                             | OcuNotification  |    |                     |
| <b>Parent Container</b>                 | OcuChannel   |    |                     |
| <b>Description</b>                      | Definition of a function pointer to a Callback function. |    |                     |
| <b>Multiplicity</b>                     | 0..1   |    |                     |
| <b>Type</b>                             | EcucFunctionNameDef                                      |    |                     |
| <b>Default value</b>                    | --   |    |                     |
| <b>maxLength</b>                        | --   |    |                     |
| <b>minLength</b>                        | --   |    |                     |
| <b>regularExpression</b>                | --   |    |                     |
| <b>Post-Build Variant Multiplicity</b>  | true   |    |                     |
| <b>Post-Build Variant Value</b>         | true   |    |                     |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>                                  | X  | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>   | -- |                     |
|   | <b>Post-build time</b>                                   | X  | VARIANT-POST-BUILD  |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>                                  | X  | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>   | -- |                     |
|   | <b>Post-build time</b>                                   | X  | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>               | scope: local   |    |                     |

|   |  |    |   |
|---|--|----|---|
| <b>SWS Item</b>                         | <b>ECUC_Ocu_00160 :</b>  |    |   |
| <b>Name</b>                             | OcuOutputPinDefaultState   |    |   |
| <b>Parent Container</b>                 | OcuChannel   |    |   |
| <b>Description</b>                      | The parameter OcuOutputPinDefaultState represents the state that a pin associated with a channel shall be set to after initialisation. |    |   |
| <b>Multiplicity</b>                     | 0..1   |    |   |
| <b>Type</b>                             | EcucEnumerationParamDef  |    |   |
| <b>Range</b>                            | OCU_HIGH   |    | The OCU channel output pin will be set to high (3 or 5 V) when requested. |
|   | OCU_LOW  |    | The OCU channel output pin will be set to low (0V) when requested.        |
| <b>Post-Build Variant Multiplicity</b>  | true   |    |   |
| <b>Post-Build Variant Value</b>         | true   |    |   |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X  | VARIANT-PRE-COMPILE   |
|   | <b>Link time</b>   | -- |   |
|   | <b>Post-build time</b>   | X  | VARIANT-POST-BUILD  |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X  | VARIANT-PRE-COMPILE   |
|   | <b>Link time</b>   | -- |   |
|   | <b>Post-build time</b>   | X  | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>               | scope: local   |    |   |

|                                  |  |    |              |
|----------------------------------|--|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Ocu_00159 :</b>  |    |              |
| <b>Name</b>                      | OcuOutputPinUsed   |    |              |
| <b>Parent Container</b>          | OcuChannel   |    |              |
| <b>Description</b>               | Information about the usage of an output pin on this channel.<br>True: the channel uses an output pin.<br><br>False: the channel does not use an output pin. |    |              |
| <b>Multiplicity</b>              | 1  |    |              |
| <b>Type</b>                      | EcucBooleanParamDef  |    |              |
| <b>Default value</b>             | --   |    |              |
| <b>Post-Build Variant Value</b>  | false  |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|                                  | <b>Link time</b>   | -- |              |
|                                  | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b>        | scope: local   |    |              |

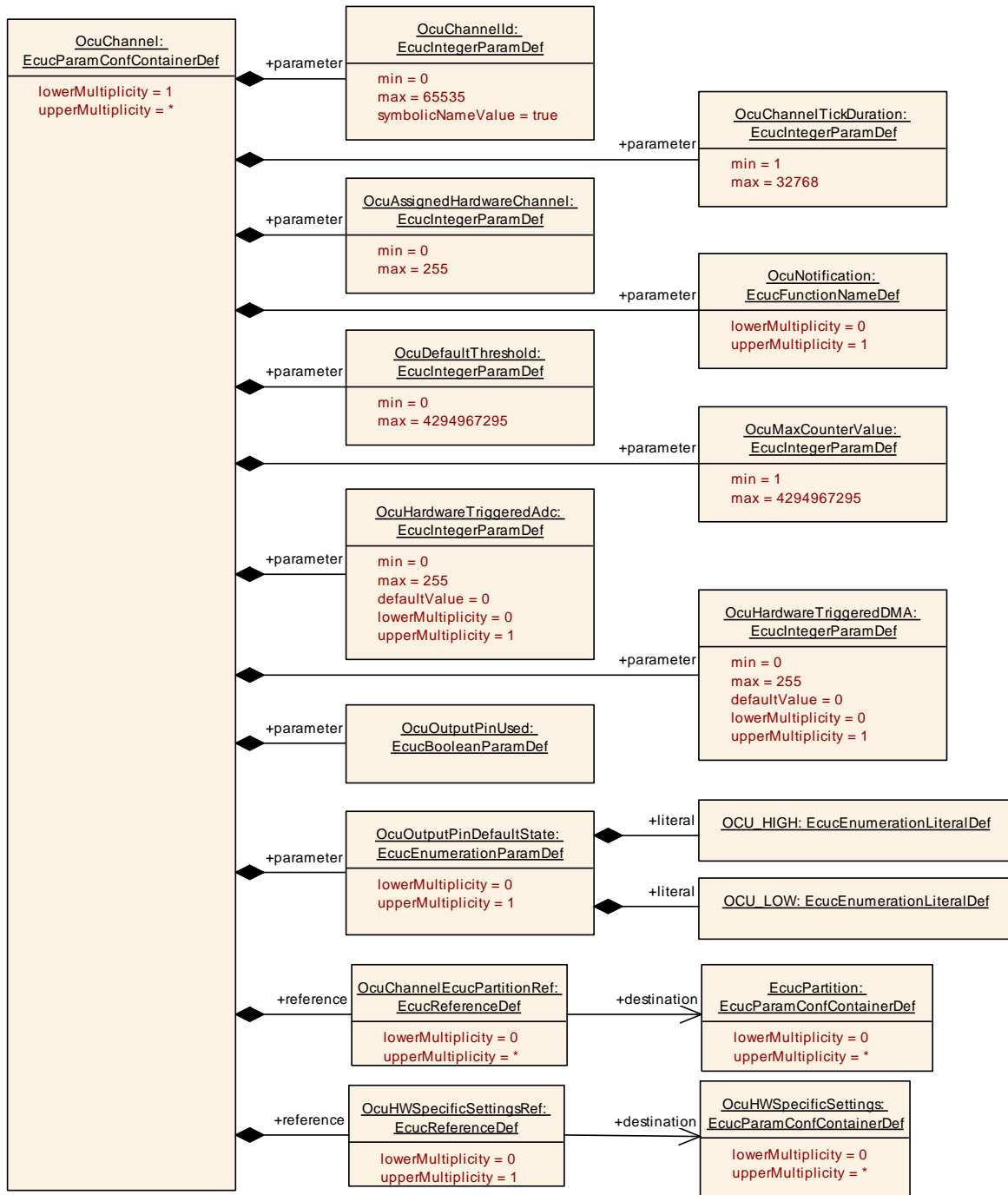
|   |  |    |              |
|---|--|----|--------------|
| <b>SWS Item</b>                         | <b>ECUC_Ocu_00169 :</b>  |    |              |
| <b>Name</b>                             | OcuChannelEcucPartitionRef   |    |              |
| <b>Parent Container</b>                 | OcuChannel   |    |              |
| <b>Description</b>                      | Maps an OCU channel to zero or multiple ECUC partitions to limit the access to this channel. The ECUC partitions referenced are a subset of the ECUC partitions where the OCU driver is mapped to. |    |              |
| <b>Multiplicity</b>                     | 0..*   |    |              |
| <b>Type</b>                             | Reference to [ EcucPartition ]   |    |              |
| <b>Post-Build Variant Multiplicity</b>  | true   |    |              |
| <b>Post-Build Variant Value</b>         | true   |    |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|   | <b>Link time</b>   | -- |              |
|   | <b>Post-build time</b>   | -- |              |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X  | All Variants |
|   | <b>Link time</b>   | -- |              |
|   | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b>               | scope: ECU   |    |              |

|   |  |    |              |
|---|--|----|--------------|
| <b>SWS Item</b>                         | <b>ECUC_Ocu_00170 :</b>  |    |              |
| <b>Name</b>                             | OcuHWSpecificSettingsRef                                       |    |              |
| <b>Parent Container</b>                 | OcuChannel   |    |              |
| <b>Description</b>                      | Reference to the OcuHWSpecificSettings used by the OcuChannel. |    |              |
| <b>Multiplicity</b>                     | 0..1   |    |              |
| <b>Type</b>                             | Reference to [ OcuHWSpecificSettings ]                         |    |              |
| <b>Post-Build Variant Multiplicity</b>  | true   |    |              |
| <b>Post-Build Variant Value</b>         | true   |    |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|   | <b>Link time</b>   | -- |              |
|   | <b>Post-build time</b>   | -- |              |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X  | All Variants |
|   | <b>Link time</b>   | -- |              |
|   | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b>               | scope: local   |    |              |

|                               |
|-------------------------------|
| <b>No Included Containers</b> |
|-------------------------------|

**[SWS\_Ocu\_CONSTR\_00002]** [The ECUC partitions referenced by OcuChannelEcucPartitionRef shall be a subset of the ECUC partitions referenced by OcuEcucPartitionRef.]()

**[SWS\_Ocu\_CONSTR\_00005]** [ If OcuEcucPartitionRef references one or more ECUC partitions, OcuKernelEcucPartitionRef shall have a multiplicity of one and reference one of these ECUC partitions as well.]()





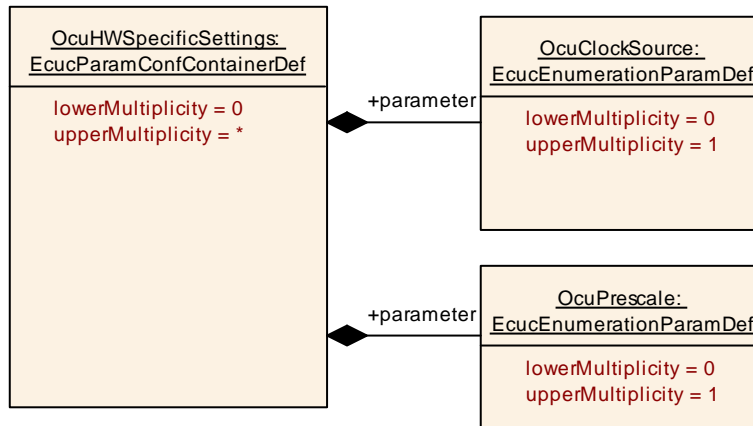
### 10.2.6 OcuHWSpecificSettings

|                                 |  |
|---------------------------------|--|
| <b>SWS Item</b>                 | <b>ECUC_Ocu_00164 :</b>  |
| <b>Container Name</b>           | OcuHWSpecificSettings  |
| <b>Parent Container</b>         | OcuConfigSet   |
| <b>Description</b>              | This container contains Ocu-specific parameters for selecting the clock source and setting optional prescalers if supported by hardware.<br><br>Implementation is defined vendor specific. |
| <b>Configuration Parameters</b> |  |

|   |  |    |                     |
|---|--|----|---------------------|
| <b>SWS Item</b>                         | <b>ECUC_Ocu_00165 :</b>  |    |                     |
| <b>Name</b>                             | OcuClockSource   |    |                     |
| <b>Parent Container</b>                 | OcuHWSpecificSettings  |    |                     |
| <b>Description</b>                      | The OCU driver specific clock input for the unit can statically be configured to select different clock sources if provided by hardware. Enumeration literals are defined vendor specific. |    |                     |
| <b>Multiplicity</b>                     | 0..1   |    |                     |
| <b>Type</b>                             | EcucEnumerationParamDef  |    |                     |
| <b>Range</b>                            | --   |    |                     |
| <b>Post-Build Variant Multiplicity</b>  | true   |    |                     |
| <b>Post-Build Variant Value</b>         | true   |    |                     |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X  | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>   | -- |                     |
|   | <b>Post-build time</b>   | X  | VARIANT-POST-BUILD  |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X  | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>   | -- |                     |
|   | <b>Post-build time</b>   | X  | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>               | scope: local   |    |                     |

|   |  |    |                     |
|---|--|----|---------------------|
| <b>SWS Item</b>                         | <b>ECUC_Ocu_00166 :</b>  |    |                     |
| <b>Name</b>                             | OcuPrescale  |    |                     |
| <b>Parent Container</b>                 | OcuHWSpecificSettings  |    |                     |
| <b>Description</b>                      | Optional OCU driver specific clock prescale factor, if supported by hardware. Implementation is defined vendor specific. |    |                     |
| <b>Multiplicity</b>                     | 0..1   |    |                     |
| <b>Type</b>                             | EcucEnumerationParamDef  |    |                     |
| <b>Range</b>                            | --   |    |                     |
| <b>Post-Build Variant Multiplicity</b>  | true   |    |                     |
| <b>Post-Build Variant Value</b>         | true   |    |                     |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X  | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>   | -- |                     |
|   | <b>Post-build time</b>   | X  | VARIANT-POST-BUILD  |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>  | X  | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>   | -- |                     |
|   | <b>Post-build time</b>   | X  | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>               | scope: local   |    |                     |

|                               |
|-------------------------------|
| <b>No Included Containers</b> |
|-------------------------------|



### 10.3 Published Information

**[SWS\_Ocu\_00169]** [The standardized common published parameters as required by SRS\_BSW\_00402 in the General Requirements on Basic Software Modules [3] 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 [8].] ()

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

## 11 Not applicable requirements

**[SWS\_Ocu\_00156]** [These requirements are not applicable to this specification.]

(SRS\_BSW\_00159, SRS\_BSW\_00167, SRS\_BSW\_00170, SRS\_BSW\_00383, SRS\_BSW\_00375, SRS\_BSW\_00416, SRS\_BSW\_00168, SRS\_BSW\_00423, SRS\_BSW\_00424, SRS\_BSW\_00425, SRS\_BSW\_00426, SRS\_BSW\_00427, SRS\_BSW\_00428, SRS\_BSW\_00429, SRS\_BSW\_00432, SRS\_BSW\_00433, SRS\_BSW\_00417, SRS\_BSW\_00161, SRS\_BSW\_00162, SRS\_BSW\_00005, SRS\_BSW\_00415, SRS\_BSW\_00164, SRS\_BSW\_00325, SRS\_BSW\_00342, SRS\_BSW\_00160, SRS\_BSW\_00007, SRS\_BSW\_00300, SRS\_BSW\_00413, SRS\_BSW\_00347, SRS\_BSW\_00305, SRS\_BSW\_00307, SRS\_BSW\_00310, SRS\_BSW\_00373, SRS\_BSW\_00327, SRS\_BSW\_00335, SRS\_BSW\_00350, SRS\_BSW\_00408, SRS\_BSW\_00410, SRS\_BSW\_00348, SRS\_BSW\_00353, SRS\_BSW\_00361, SRS\_BSW\_00301, SRS\_BSW\_00302, SRS\_BSW\_00328, SRS\_BSW\_00312, SRS\_BSW\_00006, SRS\_BSW\_00357, SRS\_BSW\_00377, SRS\_BSW\_00304, SRS\_BSW\_00378, SRS\_BSW\_00306, SRS\_BSW\_00308, SRS\_BSW\_00309, SRS\_BSW\_00358, SRS\_BSW\_00414, SRS\_BSW\_00359, SRS\_BSW\_00360, SRS\_BSW\_00330, SRS\_BSW\_00331, SRS\_BSW\_00009, SRS\_BSW\_00401, SRS\_BSW\_00172, SRS\_BSW\_00010, SRS\_BSW\_00333, SRS\_BSW\_00003, SRS\_BSW\_00341, SRS\_BSW\_00334, SRS\_SPAL\_12267, SRS\_SPAL\_12461, SRS\_SPAL\_12462, SRS\_SPAL\_12463, SRS\_SPAL\_12068, SRS\_SPAL\_12069, SRS\_SPAL\_12169, SRS\_SPAL\_12075, SRS\_SPAL\_12064, SRS\_SPAL\_12067, SRS\_SPAL\_12077, SRS\_SPAL\_12078, SRS\_SPAL\_12092, SRS\_SPAL\_12265)