Document Title	Specification of OCU Driver
Document Category	Software specification (SWS)
Document UID	615
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Removed GetVersionInfo API • Editorial changes
2025-11-27	R25-11	AUTOSAR Release Management	• Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• Editorial changes
2023-11-23	R23-11	AUTOSAR Release Management	• Editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	• Editorial changes
2021-11-25	R21-11	AUTOSAR Release Management	• Removed the Ocu_ConfigType data structure specification
2020-11-30	R20-11	AUTOSAR Release Management	• Added missing specification for enum values • Updated return value for Ocu_StartChannel • Minor changes in Error Tables



△

2019-11-28	R19-11	AUTOSAR Release Management	• Error OCU_E_BUSY classifed as Runtime Error • Added reference to OcuHWSpecificSettings in OcuChannel. Multiplicity of OcuHWSpecificSettings changed • Introduced MCAL Multicore Distribution • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• OcuGroup removed (ECUC_Ocu_00161, ECUC_Ocu_00162, ECUC_Ocu_00163) • Updated Header File Structure • Multicore feature (SWS_Ocu_00170, SWS_Ocu_CONSTR_00001, SWS_Ocu_CONSTR_00002)
2017-12-08	4.3.1	AUTOSAR Release Management	• OcuGroup set to obsolete (ECUC_Ocu_00161, ECUC_Ocu_00162 and ECUC_Ocu_00163) • Renamed "default error detection" to "development error detection"
2016-11-30	4.3.0	AUTOSAR Release Management	• Removed SWS_Ocu_00134 and SWS_Ocu_00135 • Renamed "SRS_BSW_000386" to "SRS_BSW_00386" • 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 • Removed SRS_BSW_00431 • Changed "SRS_SPAL12448" to "SRS_SPAL_12448"

▽

△

2015-07-31	4.2.2	AUTOSAR Release Management	• DET has been renamed • SWS_Ocu_00041 and SWS_Ocu_00042 requirements are removed • OCU_E_PARAM_CONFIG is removed. Added OCU_E_INIT_FAILED • Invalid requirement IDs: Updated SWS_Ocu_156, SWS_Ocu_169
2014-10-31	4.2.1	AUTOSAR Release Management	• Set the postBuildVariantValue and postBuildVariantMultiplicity to false and also • Set the valueConfigClass and the multiplicityConfigClass for all variants to preCompile. • Removal of automatically supported BSW requirement. Reference to SWS_BSW_00380 is removed.
2013-10-31	4.1.2	AUTOSAR Release Management	• Minor update of the document structure • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Release Management	• Initial release

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 8
2 Acronyms and Abbreviations 9
3 Related documentation 10
3.1 Input documents & related standards and norms . . . 10
3.2 Related specification . . . 10
4 Constraints and assumptions 11
4.1 Assumptions . . . 11
4.1.1 Clock . . . 11
4.1.2 Resources . . . 11
4.1.3 Counting and comparing . . . 11
4.2 Limitations . . . 12
4.3 Applicability to car domains . . . 12
5 Dependencies to other modules 13
5.1 Module DET . . . 13
5.2 Module DEM . . . 13
5.3 Module MCU Driver . . . 13
5.4 Module PORT . . . 13
5.5 File structure . . . 13
5.5.1 Code file structure . . . 13
5.5.2 Header file structure . . . 14
6 Requirements Tracing 15
7 Functional specification 18
7.1 General behavior . . . 18
7.2 Version check . . . 19
7.2.1 Background & Rationale . . . 19
7.3 Time Unit Ticks . . . 20
7.3.1 Background & Rationale . . . 20
7.3.2 Requirements . . . 20
7.4 Error Classification . . . 20
7.4.1 Development Errors . . . 20
7.4.2 Runtime Errors . . . 21
7.4.3 Production Errors . . . 21
7.5 Error Detection . . . 22
7.6 Error Notification . . . 22
7.7 Debug Support . . . 22

8 API specification 23
8.1 Imported types . . . 23
8.2 Type definitions . . . 23
8.2.1 Ocu_ChannelType . . . 23
8.2.2 Ocu_ValueType . . . 24
8.2.3 Ocu_PinStateType . . . 24
8.2.4 Ocu_PinActionType . . . 25
8.2.5 Ocu_ConfigType . . . 25
8.2.6 Ocu_ReturnType . . . 26
8.3 Function definitions . . . 26
8.3.1 Ocu_Init . . . 26
8.3.2 Ocu_DeInit . . . 28
8.3.3 Ocu_StartChannel . . . 29
8.3.4 Ocu_StopChannel . . . 31
8.3.5 Ocu_SetPinState . . . 32
8.3.6 Ocu_SetPinAction . . . 34
8.3.7 Ocu_GetCounter . . . 35
8.3.8 Ocu_SetAbsoluteThreshold . . . 36
8.3.9 Ocu_SetRelativeThreshold . . . 40
8.3.10 Ocu_DisableNotification . . . 42
8.3.11 Ocu_EnableNotification . . . 43
8.4 Scheduled functions . . . 45
8.5 Expected interfaces . . . 45
8.5.1 Mandatory interfaces . . . 45
8.5.2 Optional interfaces . . . 45
8.5.3 Configurable interfaces . . . 45
9 Sequence diagrams 47
9.1 Initialization . . . 47
9.2 De-initialization . . . 48
9.3 Using the Ocu Notifications . . . 49
9.4 Ocu_SetPinState . . . 49
9.5 Ocu_SetPinAction . . . 50
9.6 Setting a new compare threshold . . . 50
10 Configuration specification 51
10.1 How to read this chapter . . . 51
10.1.1 Configuration and configuration parameters . . . 51
10.1.2 Containers . . . 51
10.1.3 Specification template for configuration parameters . . . 52
10.2 Containers and configuration parameters . . . 53
10.2.1 Ocu . . . 53
10.2.2 OcuGeneral . . . 54
10.2.3 OcuConfigurationOfOptionalApis . . . 56

10.2.4 OcuConfigSet . . . 59
10.2.5 OcuChannel . . . 60
10.2.6 OcuHWSpecificSettings . . . 66
10.3 Published Information . . . 68
A Not applicable requirements 69
B Change history of AUTOSAR traceable items 70
B.1 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 70
B.1.1 Added Specification Items in R26-11 . . . 70
B.1.2 Changed Specification Items in R26-11 . . . 70
B.1.3 Deleted Specification Items in R26-11 . . . 70
B.1.4 Added Constraints in R26-11 . . . 70
B.1.5 Changed Constraints in R26-11 . . . 70
B.1.6 Deleted Constraints in R26-11 . . . 71
B.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 71
B.2.1 Added Specification Items in R25-11 . . . 71
B.2.2 Changed Specification Items in R25-11 . . . 71
B.2.3 Deleted Specification Items in R25-11 . . . 71
B.2.4 Added Constraints in R25-11 . . . 71
B.2.5 Changed Constraints in R25-11 . . . 71
B.2.6 Deleted Constraints in R25-11 . . . 71
B.3 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 71
B.3.1 Added Specification Items in R24-11 . . . 71
B.3.2 Changed Specification Items in R24-11 . . . 72
B.3.3 Deleted Specification Items in R24-11 . . . 72
B.3.4 Added Constraints in R24-11 . . . 72
B.3.5 Changed Constraints in R24-11 . . . 72
B.3.6 Deleted Constraints in R24-11 . . . 72
B.4 Traceable item history of this document according to AUTOSAR Release R23-11 . . . 72
B.4.1 Added Specification Items in R23-11 . . . 72
B.4.2 Changed Specification Items in R23-11 . . . 72
B.4.3 Deleted Specification Items in R23-11 . . . 72
B.4.4 Added Constraints in R23-11 . . . 73
B.4.5 Changed Constraints in R23-11 . . . 73
B.4.6 Deleted Constraints in R23-11 . . . 73

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.

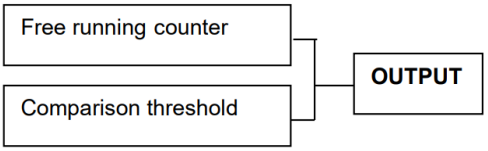


Figure 1.1: Abstract view of an OCU channel

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the OCU Driver module that are not included in the [1, AUTOSAR glossary].

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.

Acronym/Abbreviation	Description
OCU	Output Compare Unit
DMA	Direct Memory Access
SPAL	Standard Peripheral Abstraction Layer

Term definition:	Description:
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 maxium) 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 Ocu_SetAbsolut Threshold API, and used as base to compute the return information.

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[3] Specification of ECU Configuration
AUTOSAR_CP_TPS_ECUConfiguration

[4] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

[5] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2], which is also valid for OCU Driver.

Thus, General Specification on Basic Software modules [2] shall be considered as additional and required specification for OCU Driver.

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 compre 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).

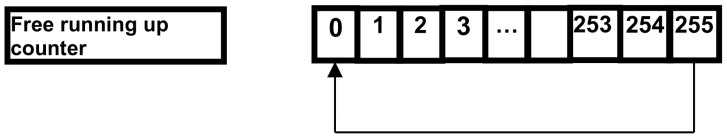


Figure 4.1: Abstraction model of the free running counter for this driver

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 comparaison can occur when entering a state of the counter or while exiting from a state, as shown in the picture below

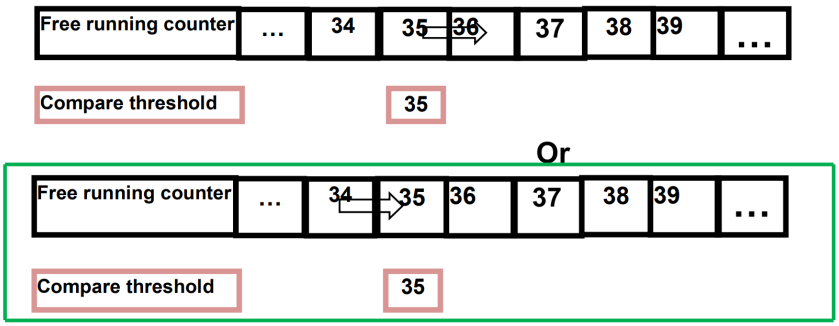


Figure 4.2: Abstraction model of the comparison process expected in driver

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

5.1 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.

5.2 Module DEM

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

5.3 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 AUTOSAR_TPS_ECUConfiguration [3] contains the chapter '4.8 Clock Tree Configuration', which details the mechanism to deliver reference clock signals to peripherals.

5.4 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.

5.5 File structure

5.5.1 Code file structure

[SWS_Ocu_00001]

Upstream requirements: SRS_BSW_00419, SRS_BSW_00346, SRS_BSW_00314

⌈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.⌋

5.5.2 Header file structure

[SWS_Ocu_00006]

Upstream requirements: SRS_BSW_00415, SRS_BSW_00456, SRS_BSW_00447

⌈Ocu.c shall include Ocu.h and Det.h.⌋

6 Requirements Tracing

The following tables reference the requirements specified in <CITATIONS_OF_CONTRIBUTED_DOCUMENTS> and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00003]	All software modules shall provide version and identification information	[SWS_Ocu_00169]
[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_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_Ocu_00035] [SWS_Ocu_00036]
[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_00312]	Shared code shall be reentrant	[SWS_Ocu_00053] [SWS_Ocu_00061] [SWS_Ocu_00068] [SWS_Ocu_00078] [SWS_Ocu_00087] [SWS_Ocu_00093] [SWS_Ocu_00102] [SWS_Ocu_00110] [SWS_Ocu_00117]
[SRS_BSW_00314]	All internal driver modules shall separate the interrupt frame definition from the service routine	[SWS_Ocu_00001]
[SRS_BSW_00318]	Each AUTOSAR Basic Software Module file shall provide version numbers in the header file	[SWS_Ocu_00169]
[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_00114] [SWS_Ocu_00120] [SWS_Ocu_00121]
[SRS_BSW_00327]	Error values naming convention	[SWS_Ocu_00016]
[SRS_BSW_00331]	All Basic Software Modules shall strictly separate error and status information	[SWS_Ocu_00016]
[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] [SWS_Ocu_00127] [SWS_Ocu_91001]
[SRS_BSW_00339]	Reporting of production relevant error status	[SWS_Ocu_00017] [SWS_Ocu_00019] [SWS_Ocu_00020] [SWS_Ocu_00021]
[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]



△

Requirement	Description	Satisfied by
[SRS_BSW_00344]	BSW Modules shall support link-time configuration	[SWS_Ocu_00035] [SWS_Ocu_00036]
[SRS_BSW_00345]	BSW Modules shall support pre-compile configuration	[SWS_Ocu_00035]
[SRS_BSW_00346]	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	[SWS_Ocu_00001]
[SRS_BSW_00357]	For success/failure of an API call a standard return type shall be defined	[SWS_Ocu_00027]
[SRS_BSW_00359]	Callback Function Return Types for AUTOSAR BSW	[SWS_Ocu_00128]
[SRS_BSW_00360]	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	[SWS_Ocu_00128]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_Ocu_00018] [SWS_Ocu_00019] [SWS_Ocu_00127]
[SRS_BSW_00377]	A Basic Software Module can return a module specific types	[SWS_Ocu_00138]
[SRS_BSW_00385]	List possible error notifications	[SWS_Ocu_00016] [SWS_Ocu_00017] [SWS_Ocu_91001] [SWS_Ocu_91002]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions 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_00114] [SWS_Ocu_00119] [SWS_Ocu_00120] [SWS_Ocu_00121]
[SRS_BSW_00402]	Each module shall provide version information	[SWS_Ocu_00169]
[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] [SWS_Ocu_00035]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_Ocu_00043] [SWS_Ocu_00050] [SWS_Ocu_00055] [SWS_Ocu_00057] [SWS_Ocu_00065] [SWS_Ocu_00074] [SWS_Ocu_00083] [SWS_Ocu_00090] [SWS_Ocu_00095] [SWS_Ocu_00104] [SWS_Ocu_00112] [SWS_Ocu_00119]
[SRS_BSW_00415]	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	[SWS_Ocu_00006] [SWS_Ocu_00024]
[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]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00438]	Configuration data shall be defined in a structure	[SWS_Ocu_00033]
[SRS_BSW_00447]	Standardizing Include file structure of BSW Modules Implementing Autosar Service	[SWS_Ocu_00006]
[SRS_BSW_00456]	A Header file shall be defined in order to harmonize BSW Modules	[SWS_Ocu_00006]
[SRS_BSW_00487]	Errors for module initialization shall follow a naming rule	[SWS_Ocu_91001]
[SRS_Ocu_00002]	The OCU driver shall support the following basic static configurations per channel	[SWS_Ocu_00028] [SWS_Ocu_00033] [SWS_Ocu_00133]
[SRS_Ocu_00005]	The OCU Driver shall provide the functionality to de-initialize OCU driver	[SWS_Ocu_00044] [SWS_Ocu_00045] [SWS_Ocu_00048]
[SRS_Ocu_00006]	The OCU driver shall provide a notification for an OCU channel when the current value of the counter matches the threshold	[SWS_Ocu_00133]
[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] [SWS_Ocu_00133]
[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] [SWS_Ocu_00060]
[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] [SWS_Ocu_00101]
[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_00031] [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 attached to a channel upon comparison match	[SWS_Ocu_00032] [SWS_Ocu_00076] [SWS_Ocu_00077]
[SRS_SPAL_00157]	All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers	[SWS_Ocu_00128]
[SRS_SPAL_12063]	All driver modules shall only support raw value mode	[SWS_Ocu_00029]
[SRS_SPAL_12263]	The implementation of all driver modules shall allow the configuration of specific module parameter types at link time	[SWS_Ocu_00033]

Table 6.1: Requirements Tracing

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 hardaware 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.

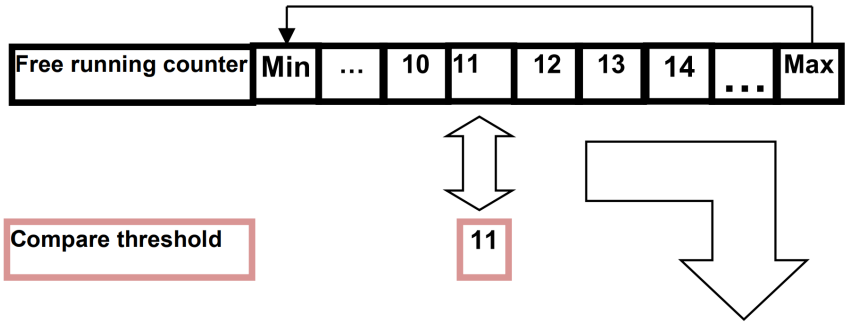


Figure 7.1: General behavior of the driver

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

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.

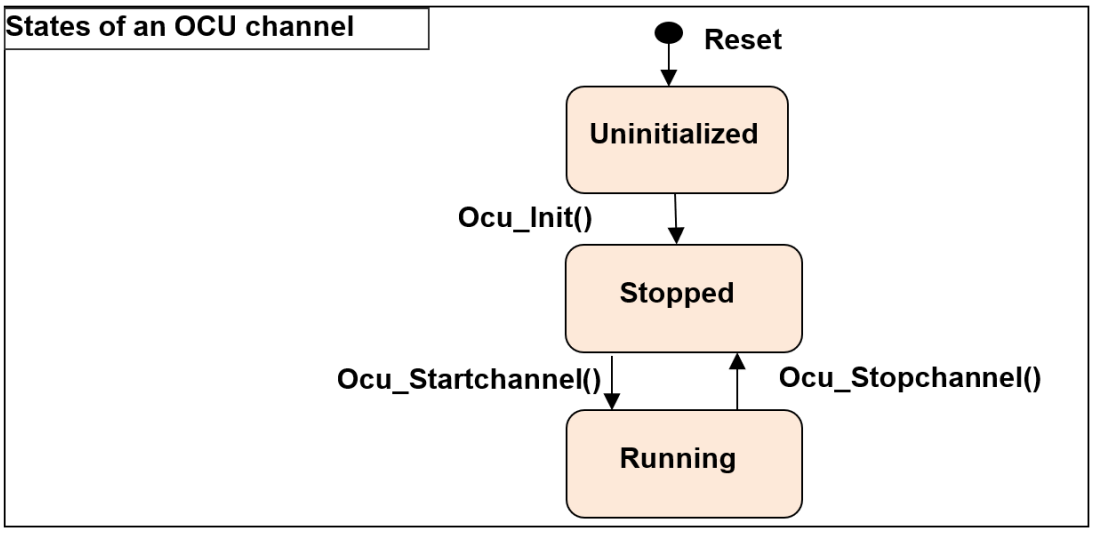


Figure 7.2: State diagram of an OCU channel

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 inialize 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]

Upstream requirements: SRS_BSW_00004

⌈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.⌋

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]

Upstream requirements: SRS_BSW_00343

⌈All time units used within the API services of the OCU driver shall be of the unit ticks.⌋

7.4 Error Classification

Chapter [2, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

[SWS_Ocu_00016]

Upstream requirements: SRS_BSW_00337, SRS_BSW_00323, SRS_BSW_00327, SRS_BSW_00331, SRS_BSW_00385, SRS_BSW_00386

⌈The following errors shall be detectable by the OCU driver depending on its build version (development / production mode).⌋

7.4.1 Development Errors

[SWS_Ocu_00015]

Upstream requirements: SRS_BSW_00337

⌈Development error values are of type uint8.⌋

[SWS_Ocu_91001] Definition of development errors in module Ocu

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00487

⌈

Type of error	*Related error code*	*Error value*
API services other than 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
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

⌋

[SWS_Ocu_00017]

Upstream requirements: SRS_BSW_00337, SRS_BSW_00339, SRS_BSW_00385, SRS_BSW_00386

⌈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.⌋

7.4.2 Runtime Errors

[SWS_Ocu_91002] Definition of runtime errors in module Ocu

Upstream requirements: SRS_BSW_00385

⌈

Type of error	*Related error code*	*Error value*
API Ocu_StartChannel() called on a channel that is in state RUNNING	OCU_E_BUSY	0x09

⌋

7.4.3 Production Errors

There are no production errors.

7.5 Error Detection

[SWS_Ocu_00018]

Upstream requirements: SRS_BSW_00369, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00019]

Upstream requirements: SRS_BSW_00386, SRS_BSW_00369, SRS_BSW_00339

⌈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.⌋

[SWS_Ocu_00020]

Upstream requirements: SRS_BSW_00339

⌈The detection of production errors cannot be switched off.⌋

7.6 Error Notification

[SWS_Ocu_00021]

Upstream requirements: SRS_BSW_00339

⌈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.⌋

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]

Upstream requirements: SRS_BSW_00415

⌈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

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

[SWS_Ocu_00027] Definition of imported datatypes of module Ocu

Upstream requirements: SRS_BSW_00357

⌈

Module	*Header File*	*Imported Type*
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

8.2.1 Ocu_ChannelType

[SWS_Ocu_00028] Definition of datatype Ocu_ChannelType

Upstream requirements: SRS_Ocu_00002

⌈

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

⌋

8.2.2 Ocu_ValueType

[SWS_Ocu_00029] Definition of datatype Ocu_ValueType

Upstream requirements: SRS_SPAL_12063

⌈

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

⌋

8.2.3 Ocu_PinStateType

[SWS_Ocu_00031] Definition of datatype Ocu_PinStateType

Upstream requirements: SRS_Ocu_00011

⌈

Name	Ocu_PinStateType		
Kind	Enumeration		
Range	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.
Description	Output state of the pin linked to an OCU channel.		
Available via	Ocu.h		

⌋

8.2.4 Ocu_PinActionType

[SWS_Ocu_00032] Definition of datatype Ocu_PinActionType

Upstream requirements: SRS_Ocu_00012

⌈

Name	Ocu_PinActionType		
Kind	Enumeration		
Range	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.
Description	Automatic action (by hardware) to be performed on a pin attached to an OCU channel.		
Available via	Ocu.h		

⌋

8.2.5 Ocu_ConfigType

[SWS_Ocu_00033] Definition of datatype Ocu_ConfigType

Upstream requirements: SRS_Ocu_00002, SRS_SPAL_12263, SRS_BSW_00405, SRS_BSW_00438

⌈

Name	Ocu_ConfigType	
Kind	Structure	
Elements	Hardware dependent	
	Type	–
	Comment	The contents of the initialization data structure are hardware specific.
Description	This is the type of the data structure containing the initialization data for the OCU driver.	
Available via	Ocu.h	

⌋

8.2.6 Ocu_ReturnType

[SWS_Ocu_00138] Definition of datatype Ocu_ReturnType

Upstream requirements: SRS_BSW_00377

⌈

Name	Ocu_ReturnType		
Kind	Enumeration		
Range	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.
Description	Return information after setting a new threshold value.		
Available via	Ocu.h		

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 Ocu_Init

[SWS_Ocu_00035] Definition of API function Ocu_Init

Upstream requirements: SRS_BSW_00101, SRS_BSW_00344, SRS_BSW_00405, SRS_BSW_00345

⌈

Service Name	Ocu_Init	
Syntax	`void Ocu_Init ( const Ocu_ConfigType* ConfigPtr )`	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the configuration set
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service for OCU initialization.	
Available via	Ocu.h	

⌋

[SWS_Ocu_00036]

Upstream requirements: SRS_BSW_00344, SRS_BSW_00404, SRS_BSW_00101

⌈The function Ocu_Init shall initialize all internals variables and the used Ocu structure of the microcontroller according to a configuration set referenced by ConfigPtr.⌋

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.⌋

[SWS_Ocu_00011] ⌈The API Ocu_Init shall start all free-running counters, which are exclusively used by this driver.⌋

[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.⌋

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.⌋
- 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.⌋

[SWS_Ocu_00040] ⌈The function Ocu_Init shall disable all notifications.⌋

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]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00044]

Upstream requirements: SRS_Ocu_00005

⌈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] Definition of API function Ocu_DeInit

Upstream requirements: SRS_Ocu_00005

⌈

Service Name	Ocu_DeInit

Syntax	`void Ocu_DeInit (`
	`  void`
	`)`

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

⌋

[SWS_Ocu_00046]

Upstream requirements: SRS_BSW_00336

⌈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.⌋

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

[SWS_Ocu_00047] ⌈The function Ocu_DeInit shall disable all used interrupts and notifications.⌋

[SWS_Ocu_00048]

Upstream requirements: SRS_Ocu_00005

⌈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.⌋

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]

Upstream requirements: SRS_BSW_00171

⌈The function Ocu_DeInit shall be pre compile time configurable On/Off by the configuration parameter: OcuDeInitApi {OCU_DE_INIT_API}.⌋

[SWS_Ocu_00050]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00386

⌈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.⌋

8.3.3 Ocu_StartChannel

[SWS_Ocu_00051] Definition of API function Ocu_StartChannel

Upstream requirements: SRS_Ocu_00008

⌈

Service Name	Ocu_StartChannel	
Syntax	`Std_ReturnType Ocu_StartChannel ( Ocu_ChannelType ChannelNumber )`	
Service ID [hex]	0x02	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU
Parameters (inout)	None	

▽

△

Parameters (out)	None	
Return value	Std_ReturnType	`E_OK` Channel was started `E_NOT_OK` Channel was not started
Description	Service to start an OCU channel.	
Available via	Ocu.h	

⌋

[SWS_Ocu_00052]

Upstream requirements: SRS_Ocu_00008

⌈The function Ocu_StartChannel shall start an OCU channel by allowing all compare match configured actions to be performed.⌋

[SWS_Ocu_00053]

Upstream requirements: SRS_BSW_00312

⌈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]

Upstream requirements: SRS_BSW_00406

⌈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.⌋

[SWS_Ocu_00056]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00057]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00386

⌈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.⌋

8.3.4 Ocu_StopChannel

[SWS_Ocu_00058] Definition of API function Ocu_StopChannel

Upstream requirements: SRS_Ocu_00008

⌈

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

⌋

[SWS_Ocu_00059]

Upstream requirements: SRS_Ocu_00008

⌈The function Ocu_StopChannel shall stop an OCU channel by halting compare match configured actions for this channel.⌋

[SWS_Ocu_00060]

Upstream requirements: SRS_Ocu_00008

⌈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]

Upstream requirements: SRS_BSW_00312

⌈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]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00065]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00386

⌈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.⌋

8.3.5 Ocu_SetPinState

[SWS_Ocu_00066] Definition of API function Ocu_SetPinState

Upstream requirements: SRS_Ocu_00011

⌈

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

⌋

[SWS_Ocu_00067]

Upstream requirements: SRS_Ocu_00011

⌈The function Ocu_SetPinState shall set the pin associated with the channel to the level indicated by "PinState".⌋

[SWS_Ocu_00068]

Upstream requirements: SRS_BSW_00312

⌈The fuction 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 requirerment also means that it shall be possible to alter the state of a STOPPED channel by this API.

[SWS_Ocu_00070]

Upstream requirements: SRS_BSW_00171

⌈The function Ocu_SetPinState shall be pre compile time configurable On/Off by the configuration parameter: OcuSetPinStateApi {OCU_SET_PIN_STATE_API}.⌋

[SWS_Ocu_00071]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00072]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈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_Set PinState shall raise the error OCU_E_PARAM_NO_PIN and return without any action.⌋

[SWS_Ocu_00073]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈If development error detection is enabled for the OCU driver: If the parameter Pin State 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.⌋

[SWS_Ocu_00074]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00075]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈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.⌋

8.3.6 Ocu_SetPinAction

[SWS_Ocu_00076] Definition of API function Ocu_SetPinAction

Upstream requirements: SRS_Ocu_00012

⌈

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

⌋

[SWS_Ocu_00077]

Upstream requirements: SRS_Ocu_00012

⌈The function Ocu_SetPinAction shall set the action to be performed by hardware automatically, at the next compare match in the corresponding OCU channel.⌋

[SWS_Ocu_00078]

Upstream requirements: SRS_BSW_00312

⌈The fuction OCU Ocu_SetPinAction shall be reentrant if it is called for different channels.⌋

[SWS_Ocu_00079]

Upstream requirements: SRS_BSW_00171

⌈The function Ocu_SetPinAction shall be pre compile time configurable by the configuration parameter: OcuSetPinActionApi {OCU_SET_PIN_ACTION_API}.⌋

[SWS_Ocu_00080]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00081]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00082]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈If development error detection is enabled for the OCU driver: If the parameter Pin Action is invalid (not within the range specified by the type), the function Ocu_SetPin Action shall raise the error OCU_E_PARAM_INVALID_ACTION and return without any action.⌋

[SWS_Ocu_00083]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00386

⌈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.⌋

[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] Definition of API function Ocu_GetCounter

Upstream requirements: SRS_Ocu_00009

⌈

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

⌋

[SWS_Ocu_00086]

Upstream requirements: SRS_Ocu_00009

⌈The function Ocu_GetCounter shall read and return the value of the counter of the channel indicated by ChannelNumber.⌋

[SWS_Ocu_00087]

Upstream requirements: SRS_BSW_00312

⌈The function Ocu_GetCounter shall be re-entrant.⌋

[SWS_Ocu_00088]

Upstream requirements: SRS_BSW_00171

⌈The function Ocu_GetCounter shall be pre compile time configurable by the configuration parameter: OcuGetCounterApi {OCU_GET_COUNTER_API}.⌋

[SWS_Ocu_00089]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈If development error detection is enabled for the OCU driver: If the parameter Channel Number 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".⌋

[SWS_Ocu_00090]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00386

⌈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".⌋

8.3.8 Ocu_SetAbsoluteThreshold

[SWS_Ocu_00091] Definition of API function Ocu_SetAbsoluteThreshold

Upstream requirements: SRS_Ocu_00010

⌈

Service Name	Ocu_SetAbsoluteThreshold
Syntax	`Ocu_ReturnType Ocu_SetAbsoluteThreshold ( Ocu_ChannelType ChannelNumber, Ocu_ValueType ReferenceValue, Ocu_ValueType AbsoluteValue )`
Service ID [hex]	0x07
Sync/Async	Synchronous
Reentrancy	Reentrant for different channel numbers

▽

△

Parameters (in)	ChannelNumber	Numeric identifier of the OCU channel
	ReferenceValue	Value given by the upper layer and used as a base to determine whether to call the notification before the function exits or not.
	AbsoluteValue	Value to compare with the content of the counter. This value is in ticks.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Ocu_ReturnType	Tells the caller whether the compare match will occur (or has already occured) during the current Refence Interval, as a result of setting the new threshold value.
Description	Service to set the value of the channel threshold using an absolute input data.	
Available via	Ocu.h	

⌋

[SWS_Ocu_00092]

Upstream requirements: SRS_Ocu_00010

⌈The function Ocu_SetAbsoluteThreshold shall set the channel threshold (the compare value) to the value given by AbsoluteValue.⌋

[SWS_Ocu_00093]

Upstream requirements: SRS_BSW_00312

⌈The fuction Ocu_SetAbsoluteThreshold shall be reentrant if it is called for different channels.⌋

[SWS_Ocu_00094]

Upstream requirements: SRS_BSW_00171

⌈The function Ocu_SetAbsoluteThreshold shall be pre compile time configurable On/ Off by the configuration parameter: OcuSetAbsoluteThresholdApi {OCU_SET_ABSOLUTE_THRESHOLD_API}.⌋

[SWS_Ocu_00095]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00096]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈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.⌋

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.

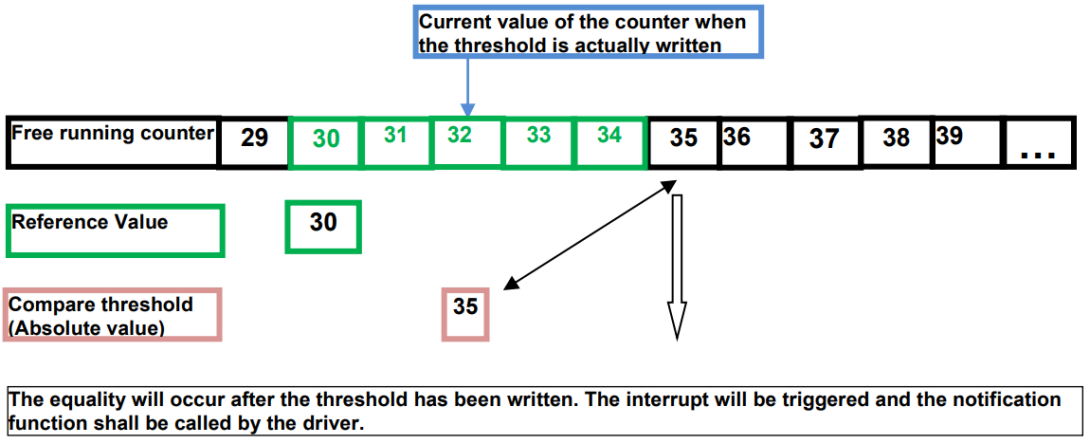


Figure 8.1: Threshold actually written before the target compare match occurs

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

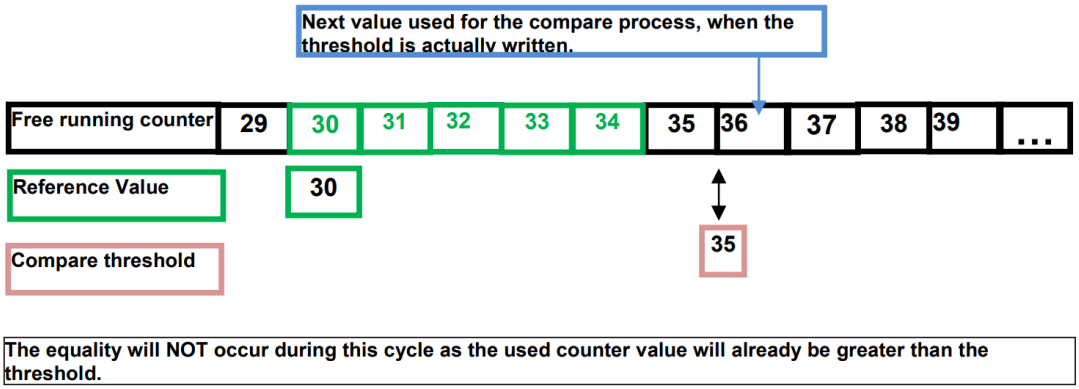


Figure 8.2: Threshold actually written after the target compare match occurs

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.

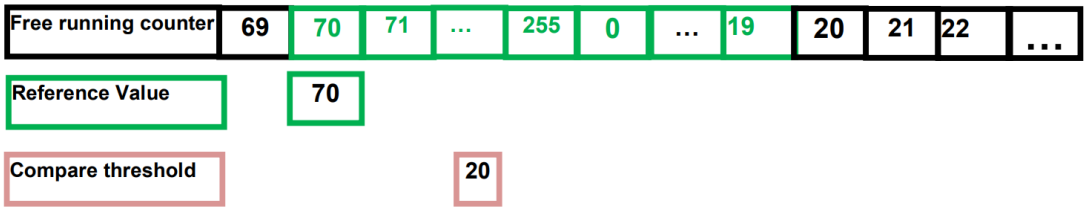


Figure 8.3: Definition of a Reference Interval

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_SetAbsolute Threshold shall return a status to inform the caller whether the compare match will occur (or has already occured) 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_SetAbsolute Threshold, 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] Definition of API function Ocu_SetRelativeThreshold

Upstream requirements: SRS_Ocu_00010

⌈

Service Name	Ocu_SetRelativeThreshold	
Syntax	`Ocu_ReturnType Ocu_SetRelativeThreshold ( Ocu_ChannelType ChannelNumber, Ocu_ValueType RelativeValue )`	
Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU channel
	RelativeValue	Value to use for computing the new threshold.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Ocu_ReturnType	Tells the caller whether the compare match will occur (or has already occured) during the current Refence Interval, as a result of setting the new threshold value.
Description	Service to set the value of the channel threshold relative to the current value of the counter.	
Available via	Ocu.h	

⌋

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:

NewThresholdValue = ReadValue + 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.

Example with Ocu_SetRelativeThreshold(1,5);

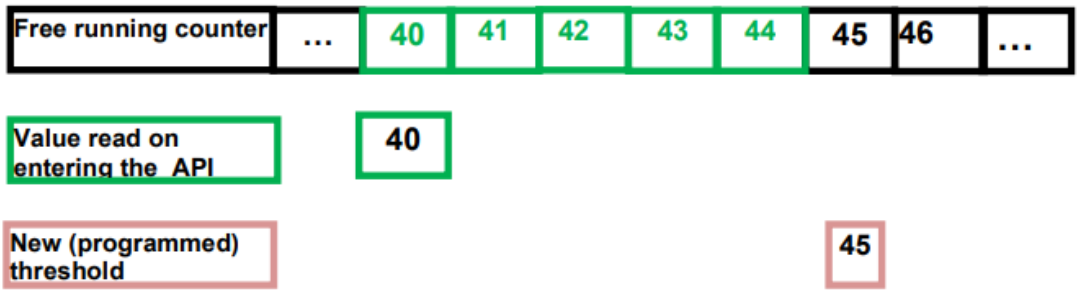


Figure 8.4: Example with OcuSetRelativeThreshold(1,5)

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

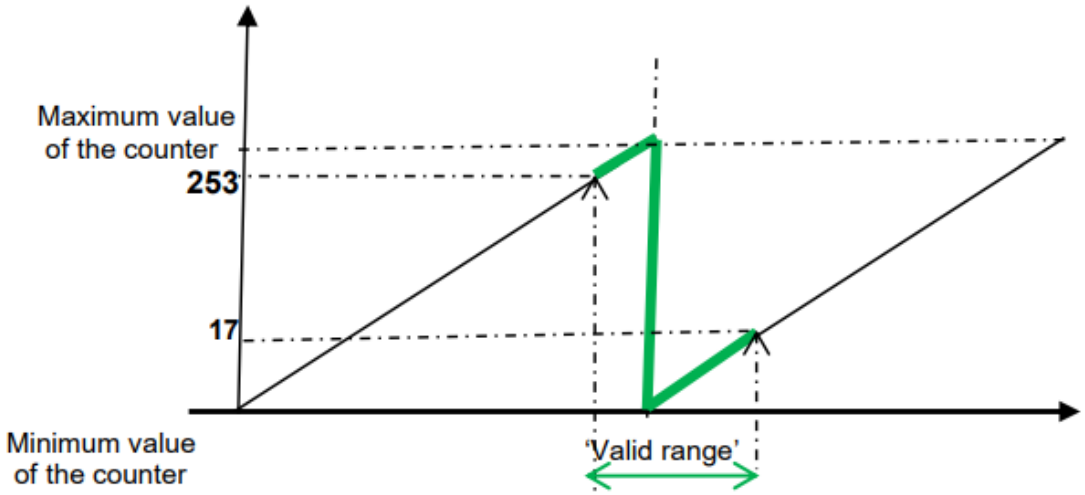


Figure 8.5: Taking into account the roll over of the counter

[SWS_Ocu_00101]

Upstream requirements: SRS_Ocu_00010

⌈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_SetRelative Threshold shall return a status to inform the caller whether the compare match will occur (or has already occured) 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]

Upstream requirements: SRS_BSW_00312

⌈The fuction OCU Ocu_SetAbsoluteThreshold shall be reentrant if it is called for different channels.⌋

[SWS_Ocu_00103]

Upstream requirements: SRS_BSW_00171

⌈The function Ocu_SetRelativeThreshold shall be pre compile time configurable On/Off by the configuration parameter: OcuSetRelativeThresholdApi {OCU_SET_RELATIVE_ THRESHOLD_API}.⌋

[SWS_Ocu_00104]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00105]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈If development error detection is enabled for the OCU driver: if the parameter Channel Number 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.⌋

8.3.10 Ocu_DisableNotification

[SWS_Ocu_00108] Definition of API function Ocu_DisableNotification

Upstream requirements: SRS_Ocu_00007

⌈

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

⌋

[SWS_Ocu_00109]

Upstream requirements: SRS_Ocu_00007

⌈The function Ocu_DisableNotification shall disable the OCU compare match notification.⌋

[SWS_Ocu_00110]

Upstream requirements: SRS_BSW_00312

⌈The fuction OCU Ocu_DisableNotification shall be reentrant if it is called for different channels.⌋

[SWS_Ocu_00111]

Upstream requirements: SRS_BSW_00171

⌈The function Ocu_DisableNotification shall be pre compile time configurable On/Off by the configuration parameter: OcuNotificationSupported {OCU_NOTIFICATION_SUPPORTED}.⌋

[SWS_Ocu_00112]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00113]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00114]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈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.⌋

8.3.11 Ocu_EnableNotification

[SWS_Ocu_00115] Definition of API function Ocu_EnableNotification

Upstream requirements: SRS_Ocu_00007

⌈

Service Name	Ocu_EnableNotification	
Syntax	`void Ocu_EnableNotification ( Ocu_ChannelType ChannelNumber )`	
Service ID [hex]	0x0b	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the OCU channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	

▽

△

Description	This service is used to enable notifications from an OCU channel.
Available via	Ocu.h

⌋

[SWS_Ocu_00116]

Upstream requirements: SRS_Ocu_00007

⌈The function Ocu_EnableNotification shall enable the OCU compare match notification of the indexed channel.⌋

[SWS_Ocu_00117]

Upstream requirements: SRS_BSW_00312

⌈The function Ocu_EnableNotification shall be reentrant if it is called for different channels.⌋

[SWS_Ocu_00118]

Upstream requirements: SRS_BSW_00171

⌈The function Ocu_EnableNotification shall be pre compile time configurable On/Off by the configuration parameter: OcuNotificationSupported {OCU_NOTIFICATION_SUPPORTED}.⌋

[SWS_Ocu_00119]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00120]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈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.⌋

[SWS_Ocu_00121]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈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.⌋

8.4 Scheduled functions

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

8.5 Expected interfaces

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

8.5.1 Mandatory interfaces

This module does not require any mandatory interfaces.

8.5.2 Optional interfaces

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

[SWS_Ocu_00127] Definition of optional interfaces requested by module Ocu

Upstream requirements: SRS_BSW_00337, SRS_BSW_00369

⌈

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

⌋

8.5.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

[SWS_Ocu_00128] Definition of configurable interface Ocu_Notification_<Channel>

Upstream requirements: SRS_BSW_00359, SRS_BSW_00360, SRS_SPAL_00157

⌈

Service Name	Ocu_Notification_<Channel>
Syntax	`void Ocu_Notification_<Channel> ( void )`

▽

△

Sync/Async	Synchronous
Reentrancy	Reentrancy of this API call depends on the user code
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	This notification function is called when a compare match occurs on the associated channel.
Available via	Ocu_Externals.h

⌋

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.⌋

[SWS_Ocu_00130] ⌈The OCU driver shall reset the interrupt flag

(if needed by hardware) associated with the notification Ocu_Notification_<Channel#>⌋

[SWS_Ocu_00132] ⌈If the NULL pointer is configured for a notification call-back, then no call-back shall be executed.⌋

[SWS_Ocu_00133]

Upstream requirements: SRS_Ocu_00002, SRS_Ocu_00006, SRS_Ocu_00007

⌈When the notification mechanism is disabled, the OCU driver shall send no notification.⌋

9 Sequence diagrams

9.1 Initialization

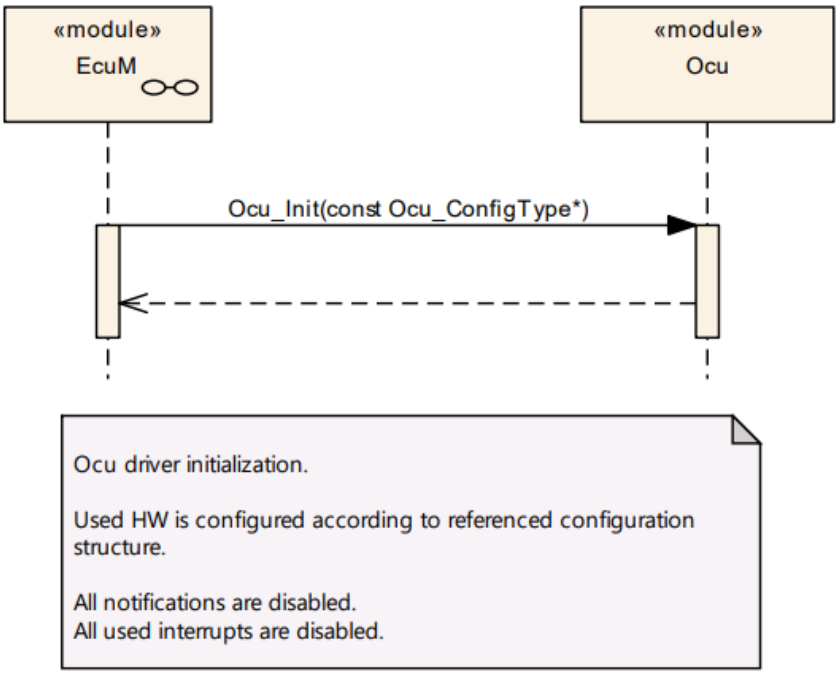


Figure 9.1: Ocu Initialization

9.2 De-initialization

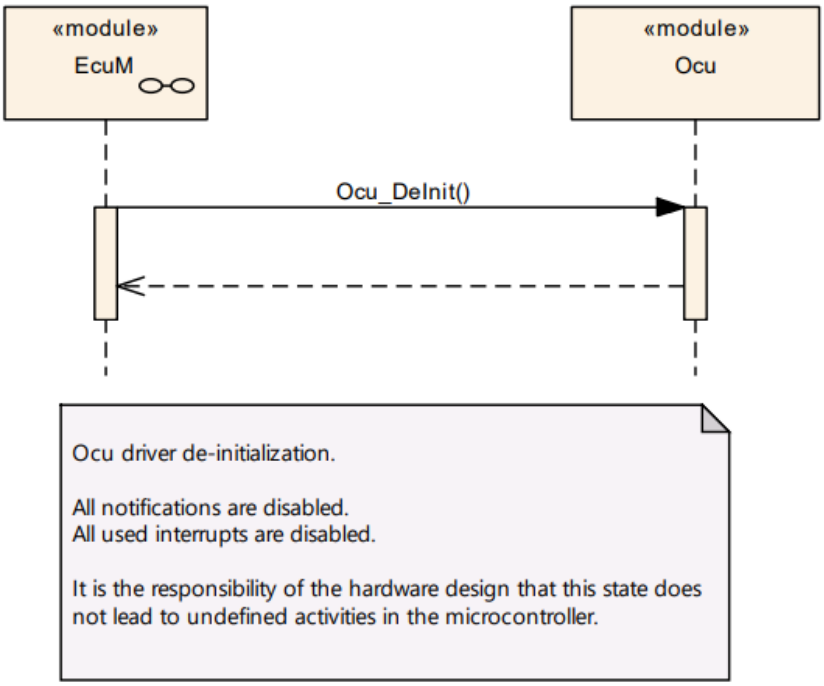


Figure 9.2: Ocu De-initialization

9.3 Using the Ocu Notifications

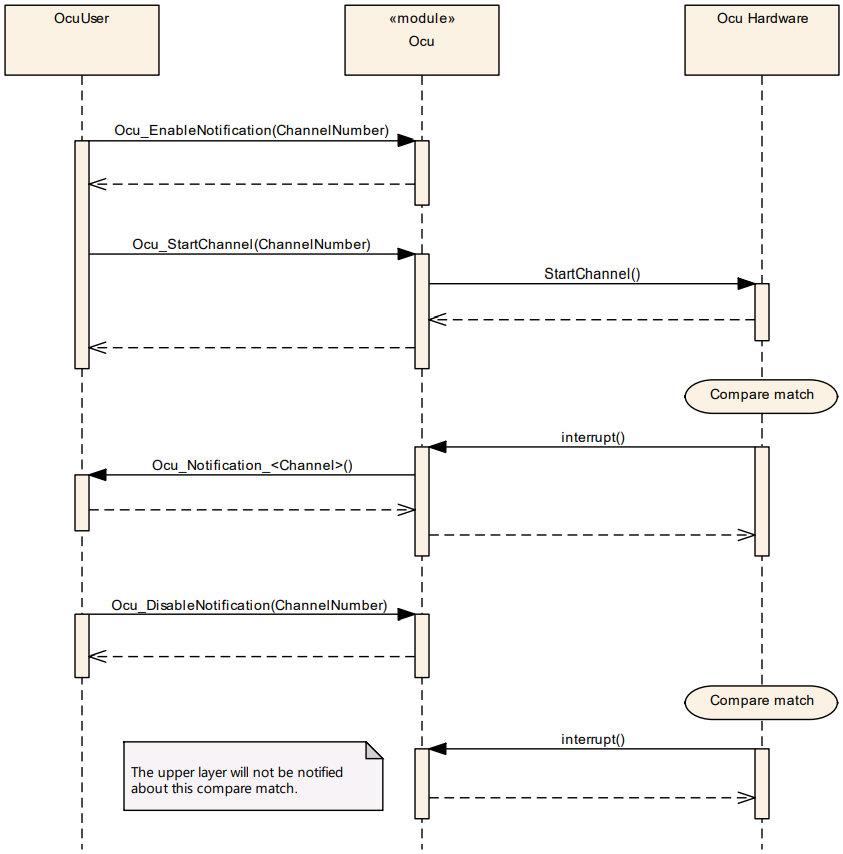


Figure 9.3: Enable and disable notifications

9.4 Ocu_SetPinState

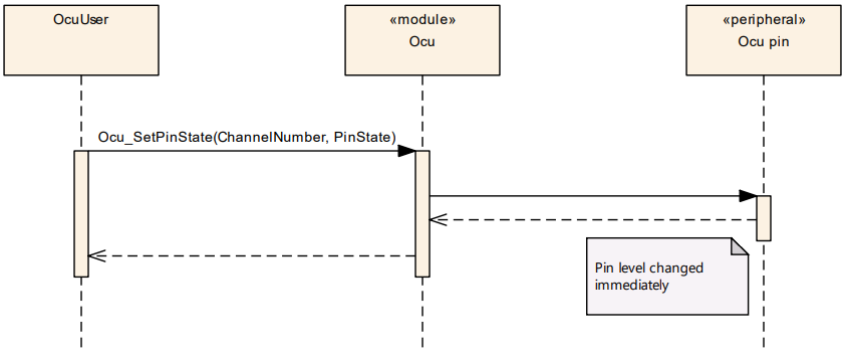


Figure 9.4: Ocu driver sets the pin state

9.5 Ocu_SetPinAction

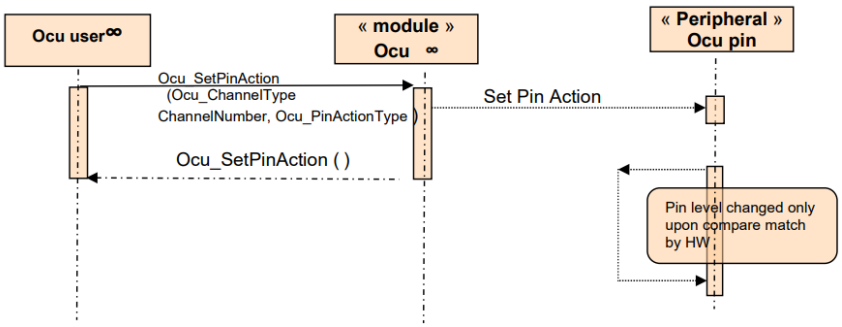


Figure 9.5: Change the pin state upon compare match

9.6 Setting a new compare threshold

Refer to the chapters 8.3.8 (Ocu_SetAbsoliteThreshold) and 8.3.9 (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.

Chapter 10.3 specifies published information of the module OCU.

10.1 How to read this chapter

For details refer to [2] Chapter 10.1 *"Introduction to configuration specification"*.

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

- AUTOSAR Layered Software Architecture [4]
- AUTOSAR ECU Configuration Specification[3]

  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 Pre-compile time.
–	The configuration parameter shall never be of configuration class Pre-compile time.

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 Link time.
–	The configuration parameter shall never be of configuration class Link time.

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 Post Build and no specific implementation is required.
L	Loadable - the configuration parameter shall be of configuration class Post Build and only one configuration parameter set resides in the ECU.
M	Multiple - the configuration parameter shall be of configuration class Post Build 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 Post Build.

10.2 Containers and configuration parameters

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

[SWS_Ocu_00170] ⌈The OCU module shall reject configurations with partition mappings which are not supported by the implementation.⌋

10.2.1 Ocu

[ECUC_Ocu_00136] Definition of EcucModuleDef Ocu ⌈

Module Name	Ocu
Description	Configuration of Ocu (Output Compare Unit) module.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
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.

⌋

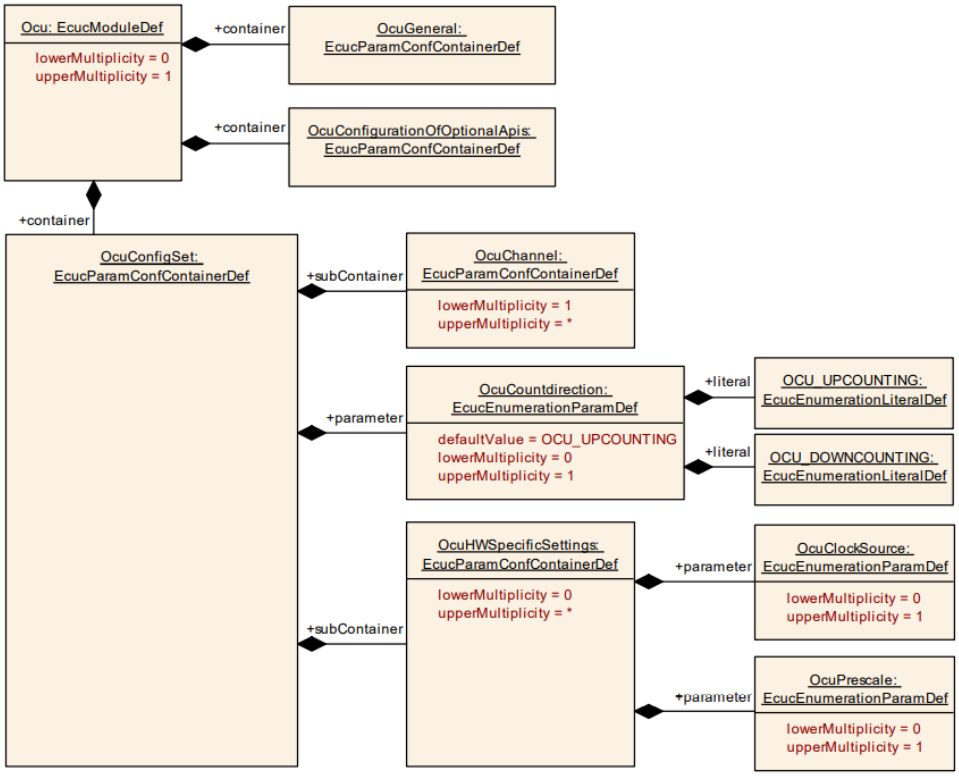


Figure 10.1: Ocu configuration

10.2.2 OcuGeneral

[SWS_Ocu_CONSTR_00001] ⌈The ECUC partitions referenced by OcuKernelEcuc PartitionRef shall be a subset of the ECUC partitions referenced by OcuEcucPartition Ref.⌋

[ECUC_Ocu_00137] Definition of EcucParamConfContainerDef OcuGeneral ⌈

Container Name	OcuGeneral
Parent Container	Ocu
Upstream mapping defined	–
Description	This container contains the module-wide configuration parameters of the OCU Driver.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OcuDevErrorDetect	1	[ECUC_Ocu_00138]
OcuEcucPartitionRef	0..*	[ECUC_Ocu_00167]
OcuKernelEcucPartitionRef	0..1	[ECUC_Ocu_00168]

No Included Containers

⌋

[ECUC_Ocu_00138] Definition of EcucBooleanParamDef OcuDevErrorDetect ⌈

Parameter Name	OcuDevErrorDetect
Parent Container	OcuGeneral
Upstream mapping defined	–
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_Ocu_00167] Definition of EcucReferenceDef OcuEcucPartitionRef ⌈

Parameter Name	OcuEcucPartitionRef	
Parent Container	OcuGeneral	
Upstream mapping defined	–	
Description	Maps the OCU driver to zero or multiple ECUC partitions to make the driver API available in the according partition.	
Multiplicity	0..*	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_Ocu_00168] Definition of EcucReferenceDef OcuKernelEcucPartitionRef ⌈

Parameter Name	OcuKernelEcucPartitionRef	
Parent Container	OcuGeneral	
Upstream mapping defined	–	
Description	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.	
Multiplicity	0..1	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[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⌋

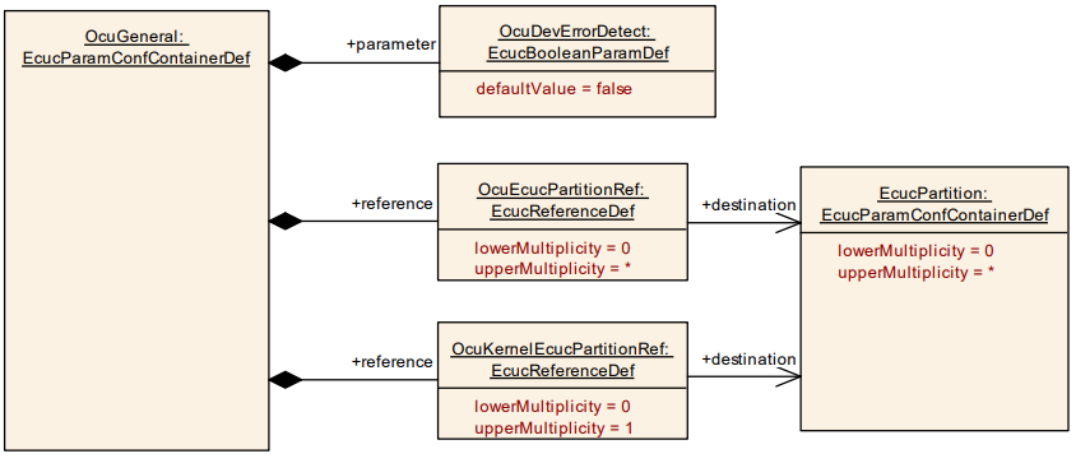


Figure 10.2: Ocu General Configuration

10.2.3 OcuConfigurationOfOptionalApis

[ECUC_Ocu_00139] Definition of EcucParamConfContainerDef OcuConfigurationOfOptionalApis ⌈

Container Name	OcuConfigurationOfOptionalApis
Parent Container	Ocu
Upstream mapping defined	–
Description	Configuration of optional APIs.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OcuDeInitApi	1	[ECUC_Ocu_00140]
OcuGetCounterApi	1	[ECUC_Ocu_00141]
OcuNotificationSupported	1	[ECUC_Ocu_00142]
OcuSetAbsoluteThresholdApi	1	[ECUC_Ocu_00143]
OcuSetPinActionApi	1	[ECUC_Ocu_00144]
OcuSetPinStateApi	1	[ECUC_Ocu_00145]
OcuSetRelativeThresholdApi	1	[ECUC_Ocu_00146]

No Included Containers

⌋

[ECUC_Ocu_00140] Definition of EcucBooleanParamDef OcuDeInitApi ⌈

Parameter Name	OcuDeInitApi
Parent Container	OcuConfigurationOfOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Ocu_DeInit() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Ocu_00141] Definition of EcucBooleanParamDef OcuGetCounterApi ⌈

Parameter Name	OcuGetCounterApi
Parent Container	OcuConfigurationOfOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Ocu_GetCounter() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Ocu_00142] Definition of EcucBooleanParamDef OcuNotificationSupported ⌈

Parameter Name	OcuNotificationSupported
Parent Container	OcuConfigurationOfOptionalApis
Upstream mapping defined	–
Description	Adds / removes the services Ocu_EnableNotification() and Ocu_DisableNotification() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Ocu_00143] Definition of EcucBooleanParamDef OcuSetAbsoluteThresholdApi ⌈

Parameter Name	OcuSetAbsoluteThresholdApi
Parent Container	OcuConfigurationOfOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Ocu_SetAbsoluteThreshold() from the code.
Multiplicity	1
Type	EcucBooleanParamDef

▽

△

Default value	–
Pre-Compile only	true

⌋

[ECUC_Ocu_00144] Definition of EcucBooleanParamDef OcuSetPinActionApi ⌈

Parameter Name	OcuSetPinActionApi
Parent Container	OcuConfigurationOfOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Ocu_SetPinAction() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Ocu_00145] Definition of EcucBooleanParamDef OcuSetPinStateApi ⌈

Parameter Name	OcuSetPinStateApi
Parent Container	OcuConfigurationOfOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Ocu_SetPinState() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Ocu_00146] Definition of EcucBooleanParamDef OcuSetRelativeThresholdApi ⌈

Parameter Name	OcuSetRelativeThresholdApi
Parent Container	OcuConfigurationOfOptionalApis
Upstream mapping defined	–
Description	Adds / removes the service Ocu_SetRelativeThreshold() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

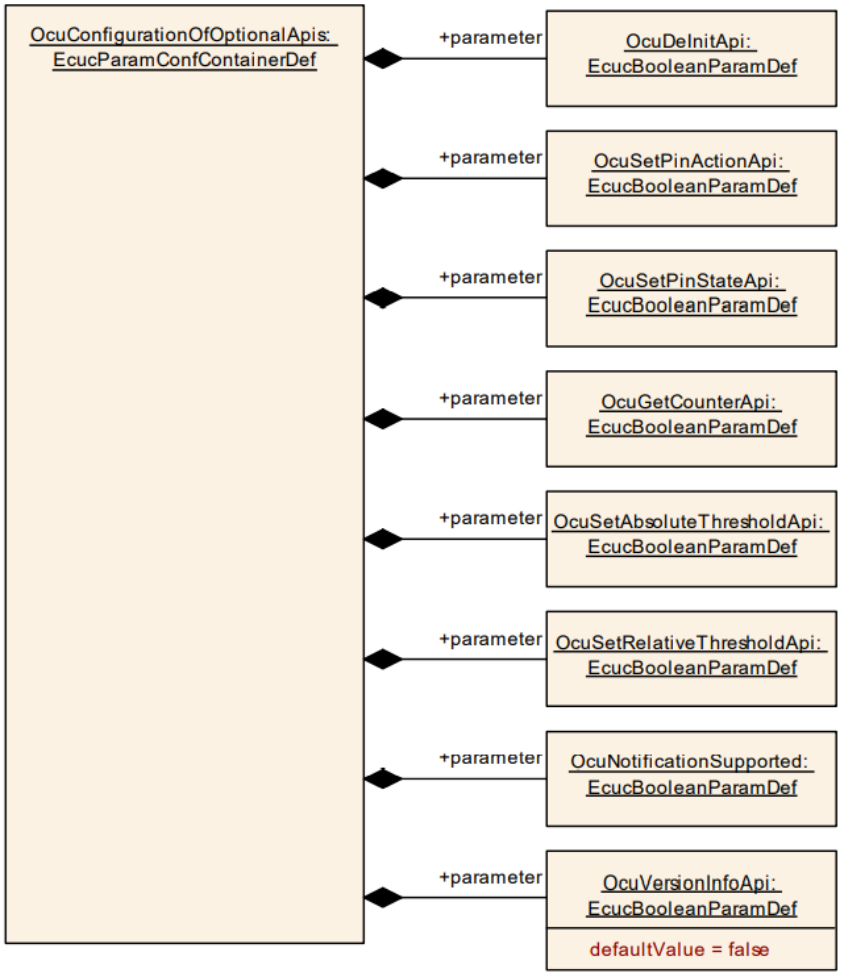


Figure 10.3: Ocu Configuration Of Optional Apis

10.2.4 OcuConfigSet

[ECUC_Ocu_00148] Definition of EcucParamConfContainerDef OcuConfigSet ⌈

Container Name	OcuConfigSet
Parent Container	Ocu
Upstream mapping defined	–
Description	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.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OcuCountdirection	0..1	[ECUC_Ocu_00149]

Included Containers		
Container Name	**Multiplicity**	**Description**
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. Implementation is defined vendor specific.

⌋

[ECUC_Ocu_00149] Definition of EcucEnumerationParamDef OcuCountdirection ⌈

Parameter Name	OcuCountdirection	
Parent Container	OcuConfigSet	
Upstream mapping defined	–	
Description	This parameter indicates the count direction for the whole OCU driver.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	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.
Default value	OCU_UPCOUNTING	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

10.2.5 OcuChannel

[ECUC_Ocu_00150] Definition of EcucParamConfContainerDef OcuChannel ⌈

Container Name	OcuChannel
Parent Container	OcuConfigSet
Upstream mapping defined	–
Description	Configuration of an individual OCU channel.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OcuAssignedHardwareChannel	1	[ECUC_Ocu_00151]
OcuChannelId	1	[ECUC_Ocu_00152]
OcuChannelTickDuration	1	[ECUC_Ocu_00153]
OcuDefaultThreshold	1	[ECUC_Ocu_00154]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OcuHardwareTriggeredAdc	0..1	[ECUC_Ocu_00155]
OcuHardwareTriggeredDMA	0..1	[ECUC_Ocu_00156]
OcuMaxCounterValue	1	[ECUC_Ocu_00157]
OcuNotification	0..1	[ECUC_Ocu_00158]
OcuOutputPinDefaultState	0..1	[ECUC_Ocu_00160]
OcuOutputPinUsed	1	[ECUC_Ocu_00159]
OcuChannelEcucPartitionRef	0..*	[ECUC_Ocu_00169]
OcuHWSpecificSettingsRef	0..1	[ECUC_Ocu_00170]

No Included Containers

⌋

[ECUC_Ocu_00151] Definition of EcucIntegerParamDef OcuAssignedHardwareChannel ⌈

Parameter Name	OcuAssignedHardwareChannel	
Parent Container	OcuChannel	
Upstream mapping defined	–	
Description	The physical hardware channel that is assigned to this logical channel.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Ocu_00152] Definition of EcucIntegerParamDef OcuChannelId ⌈

Parameter Name	OcuChannelId	
Parent Container	OcuChannel	
Upstream mapping defined	–	
Description	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.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Ocu_00153] Definition of EcucIntegerParamDef OcuChannelTickDuration ⌈

Parameter Name	OcuChannelTickDuration	
Parent Container	OcuChannel	
Upstream mapping defined	–	
Description	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	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 32768	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Ocu_00154] Definition of EcucIntegerParamDef OcuDefaultThreshold ⌈

Parameter Name	OcuDefaultThreshold	
Parent Container	OcuChannel	
Upstream mapping defined	–	
Description	Value of comparison threshold used for Initialization.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Ocu_00155] Definition of EcucIntegerParamDef OcuHardwareTriggeredAdc ⌈

Parameter Name	OcuHardwareTriggeredAdc	
Parent Container	OcuChannel	
Upstream mapping defined	–	
Description	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.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	0	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Ocu_00156] Definition of EcucIntegerParamDef OcuHardwareTriggeredDMA ⌈

Parameter Name	OcuHardwareTriggeredDMA	
Parent Container	OcuChannel	
Upstream mapping defined	–	
Description	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.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	0	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Ocu_00157] Definition of EcucIntegerParamDef OcuMaxCounterValue ⌈

Parameter Name	OcuMaxCounterValue	
Parent Container	OcuChannel	
Upstream mapping defined	–	
Description	Maximum value in ticks, the counter of the OCU channel is able to count.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Ocu_00158] Definition of EcucFunctionNameDef OcuNotification ⌈

Parameter Name	OcuNotification	
Parent Container	OcuChannel	
Upstream mapping defined	–	
Description	Definition of a function pointer to a Callback function.	
Multiplicity	0..1	
Type	EcucFunctionNameDef	
Default value	–	
Regular Expression	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Ocu_00160] Definition of EcucEnumerationParamDef OcuOutputPinDefaultState ⌈

Parameter Name	OcuOutputPinDefaultState	
Parent Container	OcuChannel	
Upstream mapping defined	–	
Description	The parameter OcuOutputPinDefaultState represents the state that a pin associated with a channel shall be set to after initialisation.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	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 (0 V) when requested.
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Ocu_00159] Definition of EcucBooleanParamDef OcuOutputPinUsed ⌈

Parameter Name	OcuOutputPinUsed
Parent Container	OcuChannel
Upstream mapping defined	–
Description	Information about the usage of an output pin on this channel. True: the channel uses an output pin. False: the channel does not use an output pin.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Ocu_00169] Definition of EcucReferenceDef OcuChannelEcucPartitionRef ⌈

Parameter Name	OcuChannelEcucPartitionRef	
Parent Container	OcuChannel	
Upstream mapping defined	–	
Description	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.	
Multiplicity	0..*	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_Ocu_00170] Definition of EcucReferenceDef OcuHWSpecificSettingsRef
⌈

Parameter Name	OcuHWSpecificSettingsRef	
Parent Container	OcuChannel	
Upstream mapping defined	–	
Description	Reference to the OcuHWSpecificSettings used by the OcuChannel.	
Multiplicity	0..1	
Type	Reference to OcuHWSpecificSettings	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[SWS_Ocu_CONSTR_00002] ⌈The ECUC partitions referenced by OcuChannelEcuc PartitionRef shall be a subset of the ECUC partitions referenced by OcuEcucPartition Ref.⌋

[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⌋

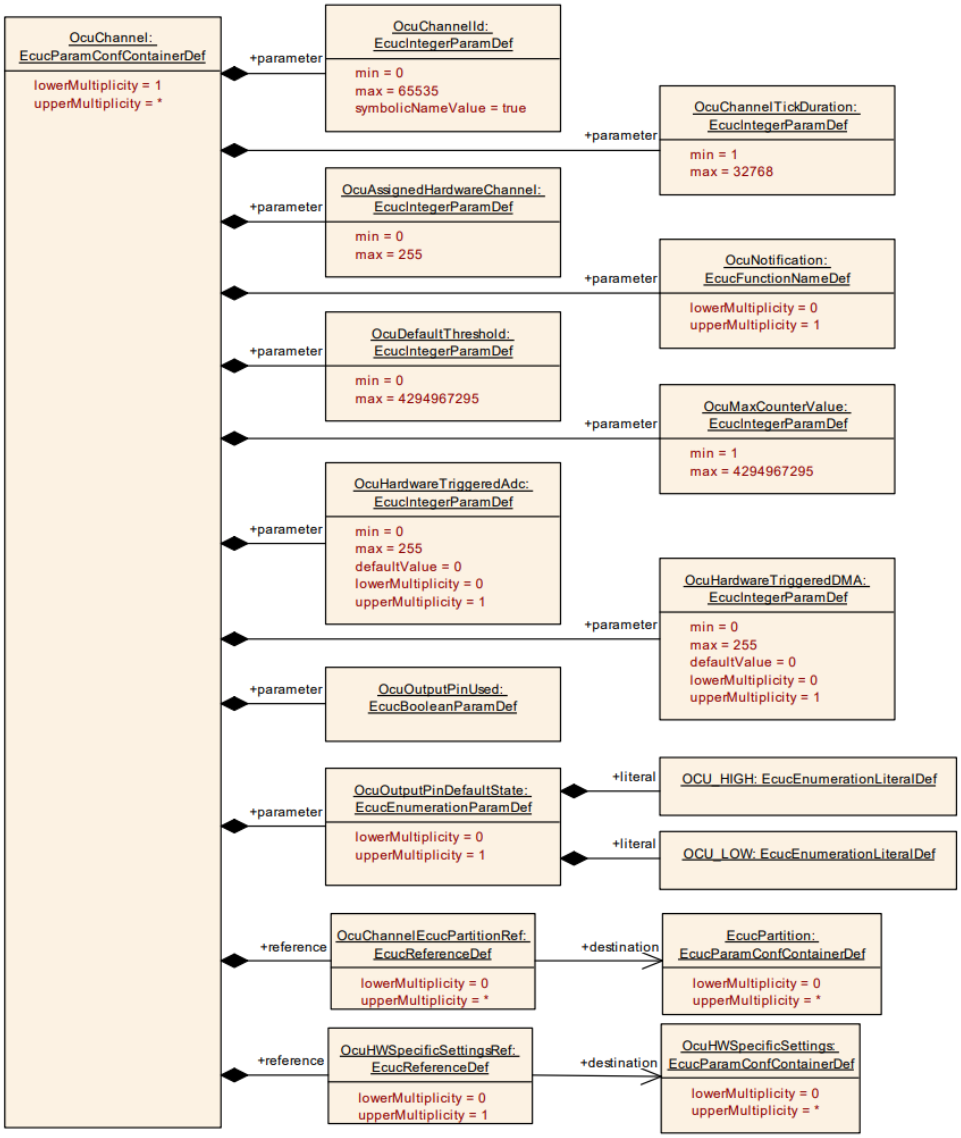


Figure 10.4: Ocu Channel Configuration

10.2.6 OcuHWSpecificSettings

[ECUC_Ocu_00164] Definition of EcucParamConfContainerDef OcuHWSpecific Settings ⌈

Container Name	OcuHWSpecificSettings
Parent Container	OcuConfigSet
Upstream mapping defined	–

▽

△

Description	This container contains Ocu-specific parameters for selecting the clock source and setting optional prescalers if supported by hardware. Implementation is defined vendor specific.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
OcuClockSource	0..1	[ECUC_Ocu_00165]
OcuPrescale	0..1	[ECUC_Ocu_00166]

No Included Containers

⌋

[ECUC_Ocu_00165] Definition of EcucEnumerationParamDef OcuClockSource ⌈

Parameter Name	OcuClockSource	
Parent Container	OcuHWSpecificSettings	
Upstream mapping defined	–	
Description	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.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Ocu_00166] Definition of EcucEnumerationParamDef OcuPrescale ⌈

Parameter Name	OcuPrescale	
Parent Container	OcuHWSpecificSettings	
Upstream mapping defined	–	
Description	Optional OCU driver specific clock prescale factor, if supported by hardware. Implementation is defined vendor specific.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

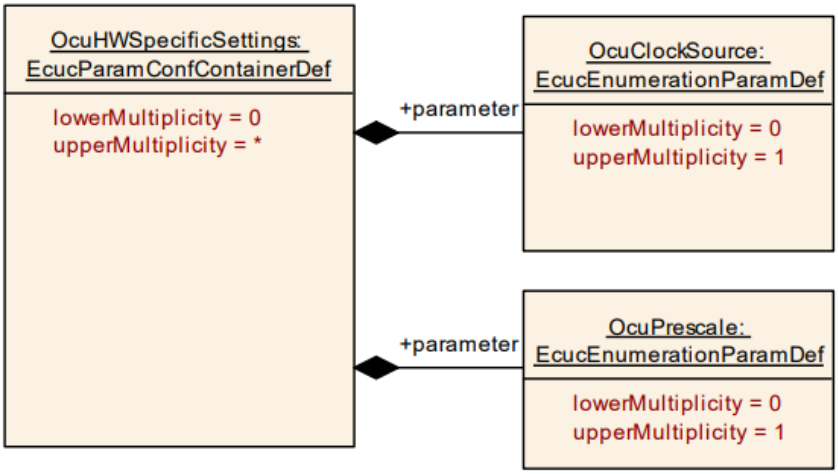


Figure 10.5: Ocu Configuration Of HW Specific Settings

10.3 Published Information

For details refer to [2] Chapter 10.3 *"Published Information"*.

[SWS_Ocu_00169]

Upstream requirements: SRS_BSW_00402, SRS_BSW_00003, SRS_BSW_00318

⌈The standardized common published parameters as required by SRS_BSW_00402 in the General Requirements on Basic Software Modules [5] 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 [2].⌋

Additional module-specific published parameters are listed in the appendix Chapter A if applicable.

A Not applicable requirements

[SWS_Ocu_NA_00156] ⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

B.1 Traceable item history of this document according to AUTOSAR Release R26-11

B.1.1 Added Specification Items in R26-11

none

B.1.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_Ocu_00139]	Definition of EcucParamConfContainerDef OcuConfigurationOfOptionalApis
[SWS_Ocu_00027]	Definition of imported datatypes of module Ocu
[SWS_Ocu_00128]	Definition of configurable interface Ocu_Notification_<Channel>
[SWS_Ocu_91001]	Definition of development errors in module Ocu

Table B.1: Changed Specification Items in R26-11

B.1.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_Ocu_00147]	Definition of EcucBooleanParamDef OcuVersionInfoApi
[SWS_Ocu_00122]	Definition of API function Ocu_GetVersionInfo
[SWS_Ocu_00123]	
[SWS_Ocu_00124]	
[SWS_Ocu_00125]	
[SWS_Ocu_00126]	

Table B.2: Deleted Specification Items in R26-11

B.1.4 Added Constraints in R26-11

none

B.1.5 Changed Constraints in R26-11

none

B.1.6 Deleted Constraints in R26-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R25-11

B.2.1 Added Specification Items in R25-11

none

B.2.2 Changed Specification Items in R25-11

none

B.2.3 Deleted Specification Items in R25-11

none

B.2.4 Added Constraints in R25-11

none

B.2.5 Changed Constraints in R25-11

none

B.2.6 Deleted Constraints in R25-11

none

B.3 Traceable item history of this document according to AUTOSAR Release R24-11

B.3.1 Added Specification Items in R24-11

none

B.3.2 Changed Specification Items in R24-11

Number	Heading
[SWS_Ocu_00169]	

Table B.3: Changed Specification Items in R24-11

B.3.3 Deleted Specification Items in R24-11

none

B.3.4 Added Constraints in R24-11

none

B.3.5 Changed Constraints in R24-11

none

B.3.6 Deleted Constraints in R24-11

none

B.4 Traceable item history of this document according to AUTOSAR Release R23-11

B.4.1 Added Specification Items in R23-11

none

B.4.2 Changed Specification Items in R23-11

none

B.4.3 Deleted Specification Items in R23-11

none

B.4.4 Added Constraints in R23-11

none

B.4.5 Changed Constraints in R23-11

none

B.4.6 Deleted Constraints in R23-11

none