Document Title	Specification of Watchdog Interface
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	41

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• Editorial changes
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• Editorial changes
2021-11-25	R21-11	AUTOSAR Release Management	• Editorial changes
2020-11-30	R20-11	AUTOSAR Release Management	• Updated the structure and tables of the error sections
2019-11-28	R19-11	AUTOSAR Release Management	• Corrected error codes • Removed the error code WDGIF_E_INV_POINTER • Corrected uptrace of [SWS_Wdgif_00046] • Cleanup of diagrams in chapter 10 • Changed Document Status from Final to published

△

2018-10-31	4.4.0	AUTOSAR Release Management	• Header File cleanup • Editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• Renaming – dafeault errors changed to development errors
2016-11-30	4.3.0	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2015-07-31	4.2.2	AUTOSAR Release Management	• Minor fixes
2014-10-31	4.2.1	AUTOSAR Release Management	• Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Editorial changes • Removed chapter(s) on change documentation
2013-10-31	4.1.2	AUTOSAR Release Management	• Minor corrections • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Artifact path fixed • Reworked according to the new SWS_BSWGeneral • New indexing scheme for requirements
2011-12-22	4.0.3	AUTOSAR Administration	• Modification in DeviceIndex • new template with requirements traceability
2010-09-30	3.1.5	AUTOSAR Administration	• Update of module version check, addition of invalid pointer as error code and checking for null pointer
2010-02-02	3.1.4	AUTOSAR Administration	• Modifications for windowed watchdog concept • Further maintenance for R4.0 • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised

▽

△

2007-12-21	3.0.1	AUTOSAR Administration	• The main bullets summarizing the changes are • Tables of chapter 8 has been replaced with • Contents generated from AUTOSAR BSW model • Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• In chapter 5.1.2 the file include structure has been changed to comply with the SPAL general include structure. • Legal disclaimer added • Release Notes added • “Advice for users” revised • “Revision Information” added
2006-05-16	2.0	AUTOSAR Administration	• Document structure adapted to common Release 2.0 SWS Template
2005-05-31	1.0	AUTOSAR Administration	• Initial Release

Disclaimer

Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration of the AUTOSAR Basic Software module Watchdog Interface.

In case of more than one watchdog device and watchdog driver (e.g. both an internal software watchdog and an external hardware watchdog) being used on an ECU, this module allows the watchdog manager (or any other client of the watchdog) to select the correct watchdog driver – and thus the watchdog device – while retaining the API and functionality of the underlying driver.

The Watchdog Interface is part of the Onboard Device Abstraction Layer [1].

[SWS_WdgIf_00026]

Upstream requirements: SRS_Wdg_12165, SRS_Wdg_12167, SRS_MemHwAb_14019

⌈The Watchdog Interface provides uniform access to services of the underlying watchdog drivers like mode switching and setting trigger conditions⌋

2 Acronyms and Abbreviations

Note: For this module there are no local acronyms and abbreviations. All used acronyms and abbreviations should be contained in the [2, TR-Glossary].

3 Related documentation

3.1 Input documents & related standards and norms

[1] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[4] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[5] Requirements on Memory Hardware Abstraction Layer
AUTOSAR_CP_RS_MemoryHWAbstractionLayer

[6] General Requirements on SPAL
AUTOSAR_CP_RS_SPALGeneral

[7] Specification of Watchdog Driver
AUTOSAR_CP_SWS_WatchdogDriver

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [3], which is also valid for Watchdog Interface.

Thus, the specification [3, SWS BSW General] shall be considered as additional and required specification for Watchdog Interface.

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

The Watchdog Interface is part of the ECU Abstraction Layer. It allows the upper layer, especially the watchdog manager, to uniformly access one or more watchdog drivers. The implementation of the Watchdog Interface therefore depends on the number of watchdog drivers below.

5.1 File structure

5.1.1 Code file structure

For details refer to the chapter 5.1.6 “Code file structure” in [3, SWS BSW General].

5.1.2 Version check

For details refer to the chapter 5.1.8 “Version Check” in [3, SWS BSW General].

6 Requirements Tracing

The following tables reference the requirements specified in [4, General Specification of Basic Software Modules], [5, Memory Hardware Abstraction Layer], [6, Requirements on SPAL] 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_00301]	All AUTOSAR Basic Software Modules shall only import the necessary information	[SWS_WdgIf_00041]
[SRS_BSW_00304]	All AUTOSAR Basic Software Modules shall use only AUTOSAR data types instead of native C data types	[SWS_WdgIf_00013]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_WdgIf_00028]
[SRS_BSW_00327]	Error values naming convention	[SWS_WdgIf_00006]
[SRS_BSW_00337]	Classification of development errors	[SWS_WdgIf_00006]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[SWS_WdgIf_00028]
[SRS_BSW_00357]	For success/failure of an API call a standard return type shall be defined	[SWS_WdgIf_00046]
[SRS_BSW_00384]	The Basic Software Module specifications shall specify at least in the description which other modules they require	[SWS_WdgIf_00047] [SWS_WdgIf_00048]
[SRS_BSW_00385]	List possible error notifications	[SWS_WdgIf_00006]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_WdgIf_00006]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_WdgIf_00006]
[SRS_BSW_00480]	Null pointer errors shall follow a naming rule	[SWS_WdgIf_00006]
[SRS_MemHwAb_-14019]	The Memory Abstraction Interface shall provide uniform access to the API services of the underlying memory abstraction modules	[SWS_WdgIf_00017] [SWS_WdgIf_00026]
[SRS_MemHwAb_-14020]	The Memory Abstraction Interface shall allow the selection of an underlying memory abstraction module by using a device index	[SWS_WdgIf_00018]
[SRS_MemHwAb_-14021]	The Memory Abstraction Interface shall allow the pre-compile time configuration of the number of underlying memory abstraction modules	[SWS_WdgIf_00019] [SWS_WdgIf_00020]

▽

△

Requirement	Description	Satisfied by
[SRS_MemHwAb_-14022]	The Memory Abstraction Interface shall preserve the functionality of the underlying memory abstraction module	[SWS_WdgIf_00003]
[SRS_MemHwAb_-14023]	The Memory Abstraction Interface shall only check those parameters that are used within the interface itself	[SWS_WdgIf_00028]
[SRS_MemHwAb_-14024]	The Memory Abstraction Interface shall preserve the timing behavior of the underlying memory abstraction modules and their APIs	[SWS_WdgIf_00003]
[SRS_SPAL_12448]	All driver modules shall have a specific behavior after a development error detection	[SWS_WdgIf_00028]
[SRS_Wdg_12018]	The watchdog driver shall provide a service for selecting the watchdog mode	[SWS_WdgIf_00016] [SWS_WdgIf_00042] [SWS_WdgIf_00057] [SWS_WdgIf_00061]
[SRS_Wdg_12165]	For an external watchdog driver the same requirements shall apply like for an internal watchdog driver	[SWS_WdgIf_00017] [SWS_WdgIf_00026]
[SRS_Wdg_12167]	The external watchdog driver shall have a semantically identical API as an internal watchdog driver	[SWS_WdgIf_00017] [SWS_WdgIf_00026]
[SRS_Wdg_13500]	The watchdog driver shall provide a service to set the watchdog trigger condition	[SWS_WdgIf_00044]
[SWS_BSW_00212]	NULL pointer checking	[SWS_WdgIf_00006]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 General behavior

[SWS_WdgIf_00003]

Upstream requirements: SRS_MemHwAb_14022, SRS_MemHwAb_14024

⌈The Watchdog Interface shall not add functionality to the watchdog drivers. Also the Watchdog Interface does not abstract from watchdog properties like toggle or window mode, timeout periods etc. that is it does not hide any features of the underlying watchdog driver and watchdog hardware.⌋

7.2 Error Classification

Section “Error Handling” of the document [4, General Specification of Basic Software Modules] 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.

7.2.1 Development Errors

[SWS_Wdgif_00006] Definiton of development errors in module Wdgif

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_-00327, SRS_BSW_00414, SRS_BSW_00480, SWS_BSW_00212

⌈

Type of error	*Related error code*	*Error value*
API service called with wrong device index parameter	WDGIF_E_PARAM_DEVICE	0x01
NULL pointer checking	WDGIF_E_PARAM_POINTER	0x03

⌋

[SWS_WdgIf_00028]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00350, SRS_SPAL_12448, SRS_MemHwAb_14023

⌈If more than one watchdog driver is configured and the development error detection is enabled for this module, the parameter `DeviceIndex` shall be checked for being an existing device within the module's services. Detected errors shall be reported to the Default Error Tracer (DET) with the error code `WDGIF_E_PARAM_DEVICE` and the called service shall not be executed.⌋

7.2.2 Runtime Errors

There are no runtime errors.

7.2.3 Production Errors

There are no production errors.

7.2.4 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 Imported types

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

[SWS_WdgIf_00041] Definition of imported datatypes of module WdgIf

Upstream requirements: SRS_BSW_00301

⌈

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

⌋

8.2 Type definitions

Note: The implementer of the Watchdog Interface shall not change or extend the type definitions of the Watchdog Interface for a specific watchdog device or platform.

8.2.1 `WdgIf_ModeType`

[SWS_WdgIf_00061] Definition of datatype WdgIf_ModeType

Upstream requirements: SRS_Wdg_12018

⌈

Name	WdgIf_ModeType		
Kind	Enumeration		
Range	WDGIF_OFF_MODE	–	In this mode, the watchdog driver is disabled (switched off).
	WDGIF_SLOW_MODE	–	In this mode, the watchdog driver is set up for a long timeout period (slow triggering).
	WDGIF_FAST_MODE	–	In this mode, the watchdog driver is set up for a short timeout period (fast triggering).
Description	Mode type of the WdgIf module		
Available via	WdgIf.h		

⌋

[SWS_WdgIf_00016]

Upstream requirements: SRS_Wdg_12018

⌈The `WdgIf_ModeType` values shall be passed as parameters to the watchdog drivers mode switching function (`Wdg_SetMode`).⌋

Note: The hardware specific settings behind these modes are given in the watchdog drivers configuration set.

8.3 Function definitions

[SWS_WdgIf_00017]

Upstream requirements: SRS_Wdg_12165, SRS_Wdg_12167, SRS_MemHwAb_14019

⌈The Watchdog Interface shall map the APIs specified in this chapter to the API of the underlying drivers. For functional behavior refer to the specification of the watchdog driver⌋

[SWS_WdgIf_00018]

Upstream requirements: SRS_MemHwAb_14020

⌈The Watchdog Interface shall use the parameter `DeviceIndex` for selection of watchdog drivers. If only one watchdog driver is configured, the parameter `DeviceIndex` shall be ignored.⌋

[SWS_WdgIf_00013]

Upstream requirements: SRS_BSW_00304

⌈The data type for the watchdog device index shall be `uint8`. `DeviceIndex` shall provide a zero-based consecutive index.⌋

[SWS_WdgIf_00019]

Upstream requirements: SRS_MemHwAb_14021

⌈If only one watchdog driver is configured, the Watchdog Interface shall cause no runtime overhead when mapping the Watchdog Interface API to the API of the corresponding Watchdog Driver.⌋

Implementation hint: This could be done by using macros as for example

```
#define WdgIf_SetMode(DeviceIndex, WdgMode) \
        Wdg_SetMode(WdgMode)
```

[SWS_WdgIf_00020]

Upstream requirements: SRS_MemHwAb_14021

⌈If more than one watchdog driver is configured, the Watchdog Interface shall use efficient mechanisms to map the API calls to the appropriate watchdog driver.⌋

Implementation hint: One solution is to use tables of pointers to functions where the parameter `DeviceIndex` is used as array index, for example

```
#define WdgIf_SetMode(DeviceIndex, WdgMode) \
        SetModeFctPtr[DeviceIndex](WdgMode)
```

Note: The service IDs are related to the service IDs of the watchdog driver specification [7]. For that reason, they may not start with 0.

8.3.1 `WdgIf_SetMode`

[SWS_WdgIf_00042] Definition of API function WdgIf_SetMode

Upstream requirements: SRS_Wdg_12018

⌈

Service Name	WdgIf_SetMode	
Syntax	`Std_ReturnType WdgIf_SetMode ( uint8 DeviceIndex, WdgIf_ModeType WdgMode )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	DeviceIndex	Identifies the Watchdog Driver instance.
	WdgMode	The watchdog driver mode (see Watchdog Driver).
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	–
Description	Map the service WdgIf_SetMode to the service Wdg_SetMode of the corresponding Watchdog Driver.	
Available via	WdgIf.h	

⌋

[SWS_WdgIf_00057]

Upstream requirements: SRS_Wdg_12018

⌈`WdgIf_SetMode` shall return the value which it gets from the service `Wdg_SetMode` of the corresponding Watchdog Driver.⌋

Possible content of the return value is specified by the Watchdog Driver, see [7].

8.3.2 WdgIf_SetTriggerCondition

[SWS_WdgIf_00044] Definition of API function WdgIf_SetTriggerCondition

Upstream requirements: SRS_Wdg_13500

⌈

Service Name	WdgIf_SetTriggerCondition	
Syntax	`void WdgIf_SetTriggerCondition ( uint8 DeviceIndex, uint16 Timeout )`	
Service ID [hex]	0x02	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	DeviceIndex	Identifies the Watchdog Driver instance.
	Timeout	Timeout value (milliseconds) for setting the trigger counter.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Map the service WdgIf_SetTriggerCondition to the service Wdg_SetTriggerCondition of the corresponding Watchdog Driver.	
Available via	WdgIf.h	

⌋

8.3.3 WdgIf_GetVersionInfo

[SWS_WdgIf_00046] Definition of API function WdgIf_GetVersionInfo

Upstream requirements: SRS_BSW_00357

⌈

Service Name	WdgIf_GetVersionInfo
Syntax	`void WdgIf_GetVersionInfo ( Std_VersionInfoType* VersionInfoPtr )`
Service ID [hex]	0x03
Sync/Async	Synchronous
Reentrancy	Reentrant
Parameters (in)	None
Parameters (inout)	None

▽

△

Parameters (out)	VersionInfoPtr	Pointer to where to store the version information of this module.
Return value	None	
Description	Returns the version information.	
Available via	WdgIf.h	

⌋

8.4 Callback notifications

This module does not provide any callback functions.

8.5 Scheduled functions

This module does not need any scheduled functions.

8.6 Expected interfaces

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

8.6.1 Mandatory interfaces

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

[SWS_WdgIf_00047] Definition of mandatory interfaces required by module Wdg If

Upstream requirements: SRS_BSW_00384

⌈

API Function	***Header File***	***Description***
Wdg_SetMode	Wdg.h	Switches the watchdog into the mode Mode.
Wdg_SetTriggerCondition	Wdg.h	Sets the timeout value for the trigger counter.

⌋

8.6.2 Optional interfaces

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

[SWS_WdgIf_00048] Definition of optional interfaces requested by module Wdg If

Upstream requirements: SRS_BSW_00384

⌈

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

⌋

8.6.3 Configurable interfaces

There are no configurable interfaces for this module.

9 Sequence diagrams

Refer to specification of watchdog driver [7].

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 WdgIf.

Chapter 10.3 specifies published information of the module WdgIf.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in [3, SWS BSW General].

10.2 Containers and configuration parameters

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

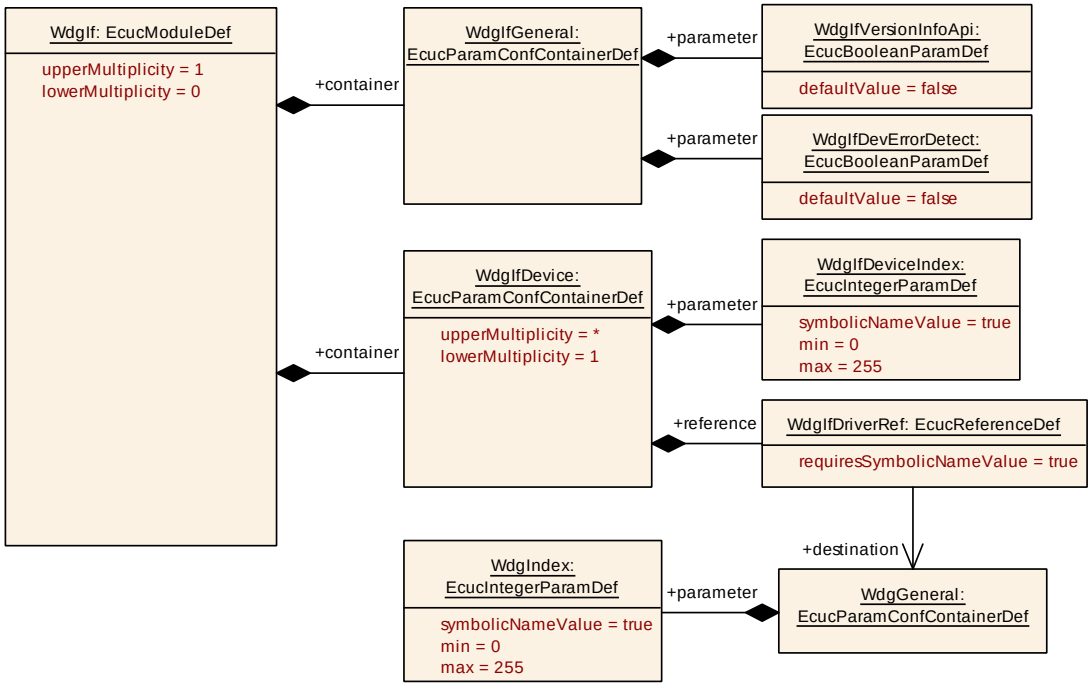

Figure 10.1: Watchdog Interface configuration

10.2.1 WdgIf

[ECUC_WdgIf_00033] Definition of EcucModuleDef WdgIf ⌈

Module Name	WdgIf
Description	Configuration of the WdgIf (Watchdog Interface) module.
Post-Build Variant Support	false
Supported Config Variants	VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
WdgIfDevice	1..*	It contains the information for the selection of a particular Watchdog device in case multiple Watchdog drivers are connected.
WdgIfGeneral	1	This container collects all generic watchdog interface parameters.

⌋

10.2.2 WdgIfGeneral

[ECUC_WdgIf_00001] Definition of EcucParamConfContainerDef WdgIfGeneral ⌈

Container Name	WdgIfGeneral
Parent Container	WdgIf
Description	This container collects all generic watchdog interface parameters.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
WdgIfDevErrorDetect	1	[ECUC_WdgIf_00005]
WdgIfVersionInfoApi	1	[ECUC_WdgIf_00003]

No Included Containers

⌋

[ECUC_WdgIf_00005] Definition of EcucBooleanParamDef WdgIfDevErrorDetect ⌈

Parameter Name	WdgIfDevErrorDetect		
Parent Container	WdgIfGeneral		
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		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_WdgIf_00003] Definition of EcucBooleanParamDef WdgIfVersionInfoApi ⌈

Parameter Name	WdgIfVersionInfoApi		
Parent Container	WdgIfGeneral		
Description	Pre-processor switch to enable / disable the service returning the version information. true: Version information service enabled false: Version information service disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.2.3 WdgIfDevice

[ECUC_WdgIf_00002] Definition of EcucParamConfContainerDef WdgIfDevice ⌈

Container Name	WdgIfDevice
Parent Container	WdgIf
Description	It contains the information for the selection of a particular Watchdog device in case multiple Watchdog drivers are connected.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
WdgIfDeviceIndex	1	[ECUC_WdgIf_00006]
WdgIfDriverRef	1	[ECUC_WdgIf_00007]

No Included Containers

⌋

[ECUC_WdgIf_00006] Definition of EcucIntegerParamDef WdgIfDeviceIndex ⌈

Parameter Name	WdgIfDeviceIndex		
Parent Container	WdgIfDevice		
Description	Represents the watchdog interface ID so that it can be referenced by the watchdog manager.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

⌋

[ECUC_WdgIf_00007] Definition of EcucReferenceDef WdgIfDriverRef ⌈

Parameter Name	WdgIfDriverRef		
Parent Container	WdgIfDevice		
Description	Reference to the watchdog drivers that are controlled by the watchdog interface.		
Multiplicity	1		
Type	Symbolic name reference to WdgGeneral		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in [3, SWS BSW General].

A Not applicable requirements

[SWS_WdgIf_NA_00999]

Upstream requirements: SRS_BSW_00004, SRS_BSW_00101, SRS_BSW_00159, SRS_BSW_-00167, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00171, SRS_BSW_00336, SRS_BSW_00339, SRS_BSW_00344, SRS_BSW_-00345, SRS_BSW_00369, SRS_BSW_00375, SRS_BSW_00380, SRS_BSW_00383, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_-00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00394, SRS_BSW_00395, SRS_BSW_00396, SRS_BSW_00397, SRS_BSW_-00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00402, SRS_BSW_00403, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_-00406, SRS_BSW_00407, SRS_BSW_00408, SRS_BSW_00409, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_-00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_-00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00437, SRS_BSW_00438, SRS_BSW_00450, SRS_BSW_00451, SRS_BSW_-00452, SRS_BSW_00458, SRS_BSW_00461, SRS_BSW_00466, SRS_BSW_00467, SRS_BSW_00469, SRS_BSW_00470, SRS_BSW_-00471, SRS_BSW_00472, SRS_BSW_00478, SRS_BSW_00488, SRS_BSW_00489, SRS_BSW_00490, SRS_BSW_00491, SRS_BSW_-00492, SRS_BSW_00493

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R24-11

B.1.1 Added Specification Items in R24-11

none

B.1.2 Changed Specification Items in R24-11

none

B.1.3 Deleted Specification Items in R24-11

none