Document Title	Specification of Watchdog Driver
Document Category	Software specification (SWS)
Document UID	39
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. • Changed the error classification from Extended Production Errors to Runtime Errors.
2025-11-27	R25-11	AUTOSAR Release Management	• No content changes.
2024-11-27	R24-11	AUTOSAR Release Management	• Added support for I2C External WDG Configuration
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• Changed ID from [SWS_Wdg_00175] to [SWS_Wdg_NA_00175]
2021-11-25	R21-11	AUTOSAR Release Management	• Removed "E_NOT_OK" return code when development error is reported
2020-11-30	R20-11	AUTOSAR Release Management	• Editorial - reshaped "Error classification" chapter
2019-11-28	R19-11	AUTOSAR Release Management	• Reworked items requiring that triggering of the WDG shall be done from interrupt routine • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Added ECUC_Wdg_00353: WdgEcucPartitionRef • minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation



△

2017-12-08	4.3.1	AUTOSAR Release Management	• minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	• Removed chapter 10.2.1 "Variants" including req SWS_Wdg_00157, SWS_Wdg_00158 SWS_Wdg_00159 • Removed Chapter "7.8 Debugging" • In table ECUC_Wdg_00073 added row for "Supported Config Variants" • minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2015-07-31	4.2.2	AUTOSAR Release Management	• Debugging support marked as obsolete • minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2014-10-31	4.2.1	AUTOSAR Release Management	• Adapt specification of extended production errors. • WDG_E_INIT_FAILED added (error code is referenced by SWS_BSWGeneral)
2014-03-31	4.1.3	AUTOSAR Release Management	• Minor editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Shift Dem_ReportErrorStatus from mandatory to optional interfaces • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Add chapter for production errors • Rename MemMap.h to Wdg_MemMap.h • Remove GPT usage • Added Subchapter 3.x due to SWS General Rollout • Reworked according to the new SWS_BSWGeneral • Reworded SWS_Wdg_00018, SWS_Wdg_00019, SWS_Wdg_00052 for debugging purpose

▽

△

2011-12-22	4.0.3	AUTOSAR Administration	• DET-Error for Wdg_GetVersionInfo added
2010-09-30	3.1.5	AUTOSAR Administration	• Requirement WDG141/WDG143 removed
2010-02-02	3.1.4	AUTOSAR Administration	• Modifications for windowed watchdog concept • Further maintenance for R4.0: see Chapter 11 • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• Section 5.1.2 the file include structure has been changed. • Section 8.6.2 Dem_ReportErrorStatus added as optional interfaces. • Rephrased the requirements WDG019, SWS_Wdg_00031, SWS_Wdg_00034. • Modified sequence diagrams in chapter 9. • 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. • In chapter WdgDefaultMode has been added as PC variant and WDG003 has been changed to allow passing NULL pointer. • For WDG037 the requirement was changed to allow configuration of activation code if the H/W allows for the same. • For SWS_Wdg_00078 the requirement was changed to add reference to SPI/DIO for accessing the external watchdog • Legal disclaimer revised • 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

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 Limitations . . . 11
4.2 Applicability to car domains . . . 11
5 Dependencies to other modules 12
5.1 File structure . . . 12
5.1.1 Code file structure . . . 12
5.1.2 Header file structure . . . 13
5.2 System clock . . . 13
5.3 Onboard communication handlers . . . 13
6 Requirements Tracing 14
7 Functional specification 17
7.1 General design rules . . . 17
7.2 External watchdog driver . . . 18
7.3 Internal watchdog driver . . . 18
7.4 Triggering concept to support windowed watchdogs . . . 19
7.5 Error Classification . . . 21
7.5.1 Development Errors . . . 21
7.5.2 Runtime Errors . . . 21
7.5.3 Production Errors . . . 22
8 API specification 23
8.1 Imported types . . . 23
8.2 Type definitions . . . 23
8.2.1 Wdg_ConfigType . . . 23
8.3 Function definitions . . . 24
8.3.1 Wdg_Init . . . 24
8.3.2 Wdg_SetMode . . . 25
8.3.3 Wdg_SetTriggerCondition . . . 28
8.3.4 Callback notifications . . . 29
8.4 Scheduled functions . . . 29
8.5 Expected interfaces . . . 29
8.5.1 Mandatory interfaces . . . 29
8.5.2 Optional interfaces . . . 29
8.5.3 Configurable interfaces . . . 29

9 Sequence diagrams 30
9.1 Watchdog initialization, setting trigger condition and mode. 30
9.2 Data exchange between watchdog driver and hardware 30
10 Configuration specification 32
10.1 How to read this chapter 32
10.2 Containers and configuration parameters 32
10.2.1 Wdg 32
10.2.2 WdgSettingsFast 36
10.3 Published Information 38
10.3.1 WdgPublishedInformation 38
A Not applicable requirements 40
B Change history of AUTOSAR traceable items 41
B.1 Traceable item history of this document according to AUTOSAR Release R26-11 41
B.1.1 Added Specification Items in R26-11 41
B.1.2 Changed Specification Items in R26-11 41
B.1.3 Deleted Specification Items in R26-11 42
B.2 Traceable item history of this document according to AUTOSAR Release R24-11 42
B.2.1 Added Specification Items in R25-11 42
B.2.2 Changed Specification Items in R25-11 42
B.2.3 Deleted Specification Items in R25-11 42
B.3 Traceable item history of this document according to AUTOSAR Release R24-11 43
B.3.1 Added Specification Items in R24-11 43
B.3.2 Changed Specification Items in R24-11 43
B.3.3 Deleted Specification Items in R24-11 43
B.4 Traceable item history of this document according to AUTOSAR Release R23-11 43
B.4.1 Added Specification Items in R22-11 43
B.4.2 Changed Specification Items in R22-11 43
B.4.3 Deleted Specification Items in R22-11 44
B.4.4 Added Specification Items in R23-11 44
B.4.5 Changed Specification Items in R23-11 44
B.4.6 Deleted Specification Items in R23-11 44

1 Introduction and functional overview

This document specifies the functionality, API and the configuration of the AUTOSAR Basic Software module watchdog driver (Wdg).

This module provides services for initialization, changing the operation mode and setting the trigger condition (timeout).

The functional requirements and the functional scope are the same for both internal and external watchdog drivers. Hence the API is semantically identical.

An internal watchdog driver belongs to the Microcontroller Abstraction Layer (MCAL), whereas an external watchdog driver belongs to the Onboard Device Abstraction Layer. Therefore, an external watchdog driver needs other drivers (in MCAL) in order to access the microcontroller hardware.

2 Acronyms and Abbreviations

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

Abbreviation / Acronym:	Description:
WDG	Watchdog (module specific prefix)

Table 2.1: Acronyms and abbreviations used in the scope of this Document

Definitions needed for understanding of the concepts

Definition:	Description:
Off-Mode	The watchdog hardware is disabled / shut down. This might be necessary in order to shut down the complete ECU and not get cyclic resets from a still running external watchdog. This mode might not be allowed for safety critical systems. In this case, the Wdg module has to be configured to prevent switching to this mode.
Slow-Mode	Triggering the watchdog hardware can be done with a long timeout period. This mode can e.g. be used during system startup / initialization phase. E.g. the watchdog hardware is configured for toggle mode (no constraints on the point in time at which the triggering is done) and a timeout period of 20 milliseconds.
Fast-Mode	Triggering the watchdog hardware has to be done with a short timeout period. This mode can e.g. be used during normal operations of the ECU. E.g. the watchdog hardware is configured for window mode (triggering the watchdog has to occur within certain minimum / maximum boundaries within the timeout period) and a timeout period of 5 milliseconds.

Table 2.2: Terms used in the scope of this Document

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] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

[4] Requirements on Watchdog Driver
AUTOSAR_CP_RS_WatchdogDriver

[5] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[6] General Requirements on SPAL
AUTOSAR_CP_RS_SPALGeneral

3.2 Related specification

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

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

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

A Wdg module for an internal (on-chip) watchdog accesses the microcontroller hardware directly and is located in the Microcontroller Abstraction layer.

A Wdg module for an external watchdog uses other modules (e.g. SPI) to access the external watchdog device. Such a Wdg module is located in the Onboard Device Abstraction Layer (see [3]).

[SWS_Wdg_00055] ⌈The Wdg module for an external watchdog driver shall have source code that is independent of the microcontroller platform.⌋

5.1 File structure

5.1.1 Code file structure

[SWS_Wdg_00079]

Upstream requirements: SRS_BSW_00346, SRS_BSW_00314, SRS_SPAL_12263

⌈The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files (as far as required; for name expansion see SWS_Wdg_00169):

- Wdg_Lcfg.c - for link time configurable parameters
- Wdg_PBcfg.c - for post build time configurable parameters

In case an internal watchdog servicing is implemented as interrupt routine, code-file structure shall include Wdg_Irq.c for holding the interrupt frames.

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

Note: These names are required by SRS_BSW_00314 and SRS_BSW_00346

[SWS_Wdg_00169]

Upstream requirements: SRS_BSW_00347

⌈If more than one watchdog driver instance exists on an ECU (namely an external and an internal one) the implementer shall provide unique code file names by expanding the names according to SRS_BSW_00347.⌋

5.1.2 Header file structure

[SWS_Wdg_00170]

Upstream requirements: SRS_BSW_00347

⌈If more than one watchdog driver instance exists on an ECU (namely an external and an internal one) the implementer shall provide unique header file names by expanding the names according to SRS_BSW_00347.⌋

5.2 System clock

If the hardware of the internal watchdog depends on the system clock, changes to the system clock (e.g. PLL on PLL off) may also affect the clock settings of the watchdog hardware.

5.3 Onboard communication handlers

A Wdg module for an external watchdog device depends on the API and capabilities of the used onboard communication handlers or drivers (e.g. SPI handler).

6 Requirements Tracing

The following tables reference the requirements specified in [4] 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_00004]	All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files	[SWS_Wdg_00086]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_Wdg_00001]
[SRS_BSW_00167]	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	[SWS_Wdg_00086]
[SRS_BSW_00314]	All internal driver modules shall separate the interrupt frame definition from the service routine	[SWS_Wdg_00079]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_Wdg_00090] [SWS_Wdg_00091] [SWS_Wdg_00092]
[SRS_BSW_00327]	Error values naming convention	[SWS_Wdg_00010] [SWS_Wdg_00180]
[SRS_BSW_00331]	All Basic Software Modules shall strictly separate error and status information	[SWS_Wdg_00010] [SWS_Wdg_00180]
[SRS_BSW_00335]	Status values naming convention	[SWS_Wdg_00017] [SWS_Wdg_00018] [SWS_Wdg_00019]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_Wdg_00031]
[SRS_BSW_00337]	Classification of development errors	[SWS_Wdg_00010] [SWS_Wdg_00035] [SWS_Wdg_00052]
[SRS_BSW_00343]	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	[SWS_Wdg_00155]
[SRS_BSW_00346]	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	[SWS_Wdg_00079]
[SRS_BSW_00347]	A Naming seperation of different instances of BSW drivers shall be in place	[SWS_Wdg_00169] [SWS_Wdg_00170] [SWS_Wdg_00172]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[SWS_Wdg_00010]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_Wdg_00106]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_Wdg_00017] [SWS_Wdg_00090] [SWS_Wdg_00091] [SWS_Wdg_00092] [SWS_Wdg_00146]
[SRS_BSW_00385]	List possible error notifications	[SWS_Wdg_00010] [SWS_Wdg_00180]



△

Requirement	Description	Satisfied by
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_Wdg_00017] [SWS_Wdg_00018] [SWS_Wdg_00019] [SWS_Wdg_00035] [SWS_Wdg_00052] [SWS_Wdg_00090] [SWS_Wdg_00091] [SWS_Wdg_00092] [SWS_Wdg_00146]
[SRS_BSW_00388]	Containers shall be used to group configuration parameters that are defined for the same object	[SWS_Wdg_00051]
[SRS_BSW_00389]	Containers shall have names	[SWS_Wdg_00051]
[SRS_BSW_00390]	Parameter content shall be unique within the module	[SWS_Wdg_00051]
[SRS_BSW_00392]	Parameters shall have a type	[SWS_Wdg_00051]
[SRS_BSW_00393]	Parameters shall have a range	[SWS_Wdg_00051]
[SRS_BSW_00395]	The Basic Software Module specifications shall list all configuration parameter dependencies	[SWS_Wdg_00051]
[SRS_BSW_00400]	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	[SWS_Wdg_00001]
[SRS_BSW_00403]	The Basic Software Module specifications shall specify for each parameter/container whether it supports different values or multiplicity in different configuration sets	[SWS_Wdg_00051]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_Wdg_00019]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_Wdg_00106] [SWS_Wdg_00171]
[SRS_BSW_00426]	BSW Modules shall ensure data consistency of data which is shared between BSW modules	[SWS_Wdg_00040]
[SRS_BSW_00429]	Access to OS is restricted	[SWS_Wdg_00040]
[SRS_SPAL_12057]	All driver modules shall implement an interface for initialization	[SWS_Wdg_00100] [SWS_Wdg_00101]
[SRS_SPAL_12064]	All driver modules shall raise an error if the change of the operation mode leads to degradation of running operations	[SWS_Wdg_00016] [SWS_Wdg_00017]
[SRS_SPAL_12092]	The driver's API shall be accessed by its handler or manager	[SWS_Wdg_00076]
[SRS_SPAL_12125]	All driver modules shall only initialize the configured resources	[SWS_Wdg_00100] [SWS_Wdg_00101]
[SRS_SPAL_12163]	All driver modules shall implement an interface for de-initialization	[SWS_Wdg_00031]
[SRS_SPAL_12263]	The implementation of all driver modules shall allow the configuration of specific module parameter types at link time	[SWS_Wdg_00079]
[SRS_SPAL_12448]	All driver modules shall have a specific behavior after a development error detection	[SWS_Wdg_00017] [SWS_Wdg_00090] [SWS_Wdg_00091] [SWS_Wdg_00092]

▽

△

Requirement	Description	Satisfied by
[SRS_SPAL_12461]	Specific rules regarding initialization of controller registers shall apply to all driver implementations	[SWS_Wdg_00100] [SWS_Wdg_00101]
[SRS_Wdg_12015]	The watchdog driver shall allow the static configuration of watchdog modes	[SWS_Wdg_00051] [SWS_Wdg_00160]
[SRS_Wdg_12018]	The watchdog driver shall provide a service for selecting the watchdog mode	[SWS_Wdg_00160]
[SRS_Wdg_12019]	The watchdog driver shall provide a watchdog trigger routine.	[SWS_Wdg_00093] [SWS_Wdg_00094] [SWS_Wdg_00095] [SWS_Wdg_00134] [SWS_Wdg_00135] [SWS_Wdg_00144]
[SRS_Wdg_12105]	The watchdog driver shall provide an initialization service that allows the selection of one of the statically configured watchdog modes	[SWS_Wdg_00001] [SWS_Wdg_00100] [SWS_Wdg_00101]
[SRS_Wdg_12165]	For an external watchdog driver the same requirements shall apply like for an internal watchdog driver	[SWS_Wdg_00077]
[SRS_Wdg_12166]	A driver for an external SPI watchdog shall allow the static configuration of the required SPI parameters	[SWS_Wdg_00078]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 General design rules

[SWS_Wdg_00086]

Upstream requirements: SRS_BSW_00167, SRS_BSW_00004

⌈The Wdg module shall statically check the configuration parameters (at the latest during compile time) for correctness.⌋

[SWS_Wdg_00031]

Upstream requirements: SRS_BSW_00336, SRS_SPAL_12163

⌈The Wdg module shall not implement an interface for de-initialization/shutdown. If the watchdog supports a de-initialization/shutdown and the environment allows the usage of this feature, the de-initialization/shutdown shall be achieved by calling the Wdg_Set Mode routine with OFF mode parameter.⌋

Rationale: Some watchdogs do not support the de-initialization/shutdown functionality and in some environments this feature must not be used (e.g. in safety critical systems).

[SWS_Wdg_00040]

Upstream requirements: SRS_BSW_00426, SRS_BSW_00429

⌈If interrupts have to be disabled in order to ensure data consistency or correct functionality of this module (e.g. while switching the watchdog mode or during the watchdog trigger routine), this shall be done by using the corresponding BSW Scheduler functionality if possible (this means definition of an exclusive area). The internal watchdog driver (because it belongs to MCAL) may also directly disable interrupts - see SRS_ BSW_00429.⌋

[SWS_Wdg_00168] ⌈Depending on a static configuration (see ECUC_Wdg_00147), the code of the Wdg module is executed either from ROM or from RAM.⌋

Motivation: For certain use cases, e.g. for flash programming in bootloader mode, the watchdog module has to be part of an executable which runs in RAM.

Hint: This is more a requirement for the build environment than for the watchdog module itself. However, since it might also influence the implementation of the code, it is stated here and a corresponding configuration parameter is given.

7.2 External watchdog driver

[SWS_Wdg_00076]

Upstream requirements: SRS_SPAL_12092

⌈To access the external watchdog hardware, the corresponding Wdg module instance shall use the functionality and API of the corresponding handler or driver, e.g. the SPI handler or DIO driver.⌋

Note:

The routine servicing an external watchdog may be implemented by usage of an own internal hardware timer to be independent from other peripherals or by using a GPT driver callback.

Hint: An external watchdog driver is part of the Onboard Device Abstraction Layer (see [3]), which excludes direct hardware access.

This architectural discrepancy will be resolved in an upcoming release.

[SWS_Wdg_00077]

Upstream requirements: SRS_Wdg_12165

⌈A Wdg module for an external watchdog shall satisfy the same functional requirements and offer the same functional scope as a Wdg module for an internal watchdog. Hence their respective APIs are semantically identical.⌋

[SWS_Wdg_00078]

Upstream requirements: SRS_Wdg_12166

⌈The Wdg module shall add all parameters required for accessing the external watchdog hardware, e.g. the used SPI channel or DIO port, to the module's published parameters and to the module's configuration parameters.⌋

7.3 Internal watchdog driver

[SWS_Wdg_00161] ⌈To access the internal watchdog hardware, the corresponding Wdg module instance shall access the hardware for watchdog servicing directly.⌋

Hint: An internal watchdog driver is part of the Microcontroller Abstraction Layer (see [3]), which allows direct hardware access.

Note:

The routine servicing an internal watchdog may be implemented by usage of an internal hardware timer to be independent from other peripherals or by using a GPT driver callback."

If the watchdog servicing routine is implemented as an interrupt routine (i.e. as a cat1 or cat2 interrupt routine and not via the GPT), it shall be described in the Basic Software Module Description and the implementation shall follow the requirements for interrupt handling as given by [5] and [6] (SRS_BSW_00427, SRS_BSW_00325, SRS_BSW_00439, SRS_BSW_00314, SRS_BSW_00429, SRS_SPAL_12129).

7.4 Triggering concept to support windowed watchdogs

In former versions of this specification, the watchdog servicing routine was called from an upper layer of the software which made it difficult to guarantee timing constraints namely for windowed watchdog conditions. This concept has been changed leading to the requirements explained in this chapter.

The basic idea of this concept is to decouple the timing for servicing the watchdog hardware from the logical control.

The time base for triggering the watchdog may be provided by means of hardware. This ensures minimum timing jitter. Servicing of the watchdog hardware directly from a timer ISR ensures minimum latencies."

These two conditions - minimum jitter and latencies - ensure that the time window of a windowed watchdog can be met.

The Wdg Driver expects, that the logical control of the watchdog (whether the watchdog shall be triggered or not) shall be the responsibility of the environment, e.g. the Wdg Manager, so that the basic concepts of the Wdg Manager (alive supervision) shall remain unchanged.

[SWS_Wdg_00144]

Upstream requirements: SRS_Wdg_12019

⌈The Wdg Manager (or other entities) shall control the watchdog driver via a so called trigger condition: as long as the trigger condition is valid the Wdg Driver services the watchdog hardware, if the trigger condition becomes invalid the Wdg Driver stops triggering and the watchdog expires.

The semantics of the trigger condition can be interpreted as a "permission to service the watchdog for the next n milliseconds". Within this time frame the trigger condition has to be updated by the controlling entity else the watchdog will expire.

Handover of the watchdog control logic is simply done by shared usage of the trigger condition (e.g. during startup / shutdown).⌋

[SWS_Wdg_00134]

Upstream requirements: SRS_Wdg_12019

⌈If the trigger counter is greater than zero, the watchdog servicing routine shall decrement the trigger counter and trigger the hardware watchdog.⌋

[SWS_Wdg_00135]

Upstream requirements: SRS_Wdg_12019

⌈If the trigger counter has reached zero, the watchdog servicing routine shall do nothing (i.e. the watchdog is not triggered and will therefore expire).⌋

[SWS_Wdg_00093]

Upstream requirements: SRS_Wdg_12019

⌈If the watchdog hardware requires an activation code which can be configured or changed, the Wdg Driver shall handle the activation code internally. In this case, the Wdg Driver shall pass the correct activation code to the watchdog hardware and the watchdog hardware in turn shall update the Wdg module's internal variable where the next expected access code is stored.⌋

[SWS_Wdg_00094]

Upstream requirements: SRS_Wdg_12019

⌈If the watchdog hardware requires an activation code which can be configured or changed, the trigger cycle of the Wdg Driver shall be defined with a value so that updating the activation code by the watchdog hardware can be guaranteed.⌋

[SWS_Wdg_00095]

Upstream requirements: SRS_Wdg_12019

⌈If the watchdog hardware requires an activation code which can be configured or changed and the initial activation code can be configured, the activation code shall be provided in the Wdg Driver's configuration set. If the activation code is fixed for a particular hardware the above requirement can be ignored.⌋

[SWS_Wdg_00035]

Upstream requirements: SRS_BSW_00337, SRS_BSW_00386

⌈When development error detection is enabled for the Wdg Driver module: the watchdog servicing routine shall check whether the Wdg module's state is WDG_IDLE (meaning the watchdog driver and hardware are initialized and the watchdog is currently not being triggered or switched). If this is not the case, the function shall not trigger the watchdog hardware but raise the development error WDG_E_DRIVER_STATE.⌋

[SWS_Wdg_00052]

Upstream requirements: SRS_BSW_00337, SRS_BSW_00386

⌈When development error detection is enabled for the Wdg Driver module: the watchdog servicing routine shall set the Wdg module's state to WDG_BUSY during its execution (indicating, that the module is busy) and shall reset the module's state to WDG_IDLE (indicating, that the module is initialized and not busy) as last operation before it returns.⌋

Note: This specification prescribes the symbols WDG_IDLE and WDG_BUSY only, if they are externally visible, e.g. for debugging (see SRS_BSW_00335). Choosing the data type for the status variable is up to the implementation.

Hint for the integration: The Wdg module's environment shall make sure that the Wdg Driver module has been initialized before watchdog servicing routine is called.

7.5 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 following error types which may occur in BSW modules.

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

7.5.1 Development Errors

[SWS_Wdg_00010] Definition of development errors in module Wdg

Upstream requirements: SRS_BSW_00337, SRS_BSW_00350, SRS_BSW_00385, SRS_BSW_00327, SRS_BSW_00331

⌈

Type of error	*Related error code*	*Error value*
API service used in wrong context (e.g. module not initialized).	WDG_E_DRIVER_STATE	0x10
API service called with wrong / inconsistent parameter(s)	WDG_E_PARAM_MODE	0x11
API service called with wrong / inconsistent parameter(s)	WDG_E_PARAM_CONFIG	0x12
The passed timeout value is higher than the maximum timeout value	WDG_E_PARAM_TIMEOUT	0x13
API is called with wrong pointer value (e.g. NULL pointer)	WDG_E_PARAM_POINTER	0x14
Invalid configuration set selection	WDG_E_INIT_FAILED	0x15

⌋

7.5.2 Runtime Errors

[SWS_Wdg_91000] Definition of runtime errors in module Wdg ⌈

Type of error	*Related error code*	*Error value*
Setting a watchdog mode failed (during initialization or mode switch).	WDG_E_MODE_FAILED	0x01

⌋

[SWS_Wdg_00180]

Upstream requirements: SRS_BSW_00327, SRS_BSW_00331, SRS_BSW_00385

⌈When the setting of the watchdog mode failed, the Runtime Error WDG_E_MODE_FAILED shall be reported.⌋

7.5.3 Production Errors

There are no production errors.

8 API specification

[SWS_Wdg_00172]

Upstream requirements: SRS_BSW_00347

⌈If more than one watchdog driver instance exits on an ECU (namely an external and an internal one) the API names and instance specific type names specified in this chapter shall be made unique by expansion according to SRS_BSW_00347.⌋

8.1 Imported types

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

[SWS_Wdg_00105] Definition of imported datatypes of module Wdg ⌈

Module	*Header File*	*Imported Type*
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Std	Std_Types.h	Std_ReturnType
WdgIf	WdgIf.h	WdgIf_ModeType

⌋

8.2 Type definitions

8.2.1 Wdg_ConfigType

[SWS_Wdg_00171] Definition of datatype Wdg_ConfigType

Upstream requirements: SRS_BSW_00414

⌈

Name	Wdg_ConfigType	
Kind	Structure	
Elements	Hardware dependent structure	
	Type	–
	Comment	Structure to hold the watchdog driver configuration set.
Description	Used for pointers to structures holding configuration data provided to the Wdg module initialization routine for configuration of the module and watchdog hardware.	
Available via	Wdg.h	

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 Wdg_Init

[SWS_Wdg_00106] Definition of API function Wdg_Init

Upstream requirements: SRS_BSW_00358, SRS_BSW_00414

⌈

Service Name	Wdg_Init	

Syntax	`void Wdg_Init (`
	`  const Wdg_ConfigType* ConfigPtr`
	`)`	

Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to configuration set.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initializes the module.	
Available via	Wdg.h	

⌋

[SWS_Wdg_00001]

Upstream requirements: SRS_BSW_00400, SRS_BSW_00101, SRS_Wdg_12105

⌈The Wdg_Init function shall initialize the Wdg module and the watchdog hardware, i.e. it shall set the default watchdog mode and timeout period as provided in the configuration set.⌋

Note:

Via post-build configuration, the user can choose the configuration set to be used with the Wdg_Init function from a limited number of statically configured sets (see also SRS_BSW_00314).

[SWS_Wdg_00100]

Upstream requirements: SRS_SPAL_12057, SRS_SPAL_12125, SRS_SPAL_12461, SRS_Wdg_12105

⌈The Wdg_Init function shall initialize all global variables of the Wdg module and set the default watchdog mode and initial timeout period⌋

[SWS_Wdg_00101]

Upstream requirements: SRS_SPAL_12057, SRS_SPAL_12125, SRS_SPAL_12461, SRS_Wdg_12105

⌈The Wdg_Init function shall initialize those controller registers that are needed for controlling the watchdog hardware and that do not influence/depend on other (hardware) modules.

Registers that can influence or depend on other modules are initialized by a common system module.⌋

[SWS_Wdg_00173] ⌈If switching the Wdg module and the watchdog hardware into the default mode is not possible, e.g. because of inconsistent mode settings or because some timing constraints have not been met, the Wdg_Init function shall raise the Runtime Error WDG_E_MODE_FAILED.⌋

[SWS_Wdg_00090]

Upstream requirements: SRS_BSW_00323, SRS_SPAL_12448, SRS_BSW_00369, SRS_BSW_00386

⌈When development error detection is enabled for the Wdg module: The Wdg_Init function shall check that the (hardware specific) contents of the given configuration set is within the allowed boundaries. If this error is detected, the function Wdg_Init shall not execute the initialization but raise the extended error WDG_E_PARAM_CONFIG.⌋

[SWS_Wdg_00019]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00335, SRS_BSW_00386

⌈When development error detection is enabled for the Wdg module: The Wdg_Init function shall set the Wdg module's internal state from WDG_UNINIT (the default state indicating a non-initialized module) to WDG_IDLE if the initialization was successful.⌋

Note: This specification prescribes the symbols WDG_IDLE and WDG_UNINIT only, if they are externally visible, e.g. for debugging (see SRS_BSW_00335). Choosing the data type for the status variable is up to the implementation.

8.3.2 Wdg_SetMode

[SWS_Wdg_00107] Definition of API function Wdg_SetMode ⌈

Service Name	Wdg_SetMode
Syntax	`Std_ReturnType Wdg_SetMode ( WdgIf_ModeType Mode )`
Service ID [hex]	0x01
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	Mode	One of the following statically configured modes: 1. WDGIF_OFF_MODE 2. WDGIF_SLOW_MODE 3. WDGIF_FAST_MODE
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Service executed without errors. `E_NOT_OK`: Service executed with errors.
Description	Switches the watchdog into the mode Mode.	
Available via	Wdg.h	

⌋

[SWS_Wdg_00160]

Upstream requirements: SRS_Wdg_12015, SRS_Wdg_12018

⌈The function Wdg_SetMode shall switch the watchdog driver from the current watchdog mode into the mode given by the argument Mode. This means: By choosing one of a limited number of statically configured settings (e.g. toggle or window watchdog, different timeout periods) the Wdg module and the watchdog hardware are switched to one of the following three different modes:

- WDGIF_OFF_MODE
- WDGIF_SLOW_MODE
- WDGIF_FAST_MODE

⌋

[SWS_Wdg_00051]

Upstream requirements: SRS_Wdg_12015, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00403

⌈The configuration set provided to the Wdg module's initialization routine shall contain the hardware / driver specific parameters to be used in the different watchdog modes.⌋

[SWS_Wdg_00145] ⌈The Wdg_SetMode function shall reset the watchdog timeout counter based on the new watchdog mode i.e. the timeout frame remaining shall be recalculated based on a changed trigger period.⌋

[SWS_Wdg_00103] ⌈The Wdg_SetMode function shall return E_OK if the mode switch has been executed completely and successfully, i.e. all parameters of the Wdg module and the watchdog hardware have been set to the new values⌋

[SWS_Wdg_00016]

Upstream requirements: SRS_SPAL_12064

⌈If switching the Wdg module and the watchdog hardware into the requested mode is not possible, e.g. because of inconsistent mode settings or because some timing constraints have not been met, the Wdg_SetMode function shall return the value E_NOT_OK and raise the Runtime Error WDG_E_MODE_FAILED.⌋

[SWS_Wdg_00091]

Upstream requirements: SRS_BSW_00323, SRS_SPAL_12448, SRS_BSW_00369, SRS_BSW_00386

⌈When development error detection is enabled for the Wdg module: The Wdg_SetMode function shall check that the parameter Mode is within the allowed range. If this is not the case, the function shall not execute the mode switch but raise development error WDG_E_PARAM_MODE.⌋

[SWS_Wdg_00092]

Upstream requirements: SRS_BSW_00323, SRS_SPAL_12448, SRS_BSW_00369, SRS_BSW_00386

⌈When development error detection is enabled for the Wdg module: The Wdg_SetMode function shall check that the (hardware specific) settings for the requested mode are within the allowed boundaries. If this is not the case, the function shall not execute the mode switch but raise the development error WDG_E_PARAM_MODE.⌋

[SWS_Wdg_00017]

Upstream requirements: SRS_BSW_00335, SRS_SPAL_12064, SRS_SPAL_12448, SRS_BSW_00369, SRS_BSW_00386

⌈When development error detection is enabled for the Wdg module: The Wdg_SetMode function shall check that the Wdg module's state is WDG_IDLE (meaning the Wdg module and the watchdog hardware are initialized and the watchdog is currently not being triggered or switched). If this is not the case, the function shall not execute the mode switch but raise the development error WDG_E_DRIVER_STATE.⌋

[SWS_Wdg_00018]

Upstream requirements: SRS_BSW_00335, SRS_BSW_00386

⌈When development error detection is enabled for the Wdg module: The function Wdg_SetMode shall set the Wdg module's state to WDG_BUSY during its execution (indicating, that the module is busy) and shall reset the Wdg module's state to WDG_IDLE as last operation before it returns to the caller.⌋

Note: This specification prescribes the symbols WDG_IDLE and WDG_BUSY only, if they are externally visible, e.g. for debugging (see SRS_BSW_00335). Choosing the data type for the status variable is up to the implementation.

8.3.3 Wdg_SetTriggerCondition

[SWS_Wdg_00155] Definition of API function Wdg_SetTriggerCondition

Upstream requirements: SRS_BSW_00343

⌈

Service Name	Wdg_SetTriggerCondition	
Syntax	`void Wdg_SetTriggerCondition ( uint16 timeout )`	
Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	timeout	Timeout value (milliseconds) for setting the trigger counter.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Sets the timeout value for the trigger counter.	
Available via	Wdg.h	

⌋

[SWS_Wdg_00136] ⌈The function Wdg_SetTriggerCondition shall reset the watchdog timeout counter according to the timeout value passed.⌋

[SWS_Wdg_00138] ⌈The timeout value passed shall be interpreted as 'milliseconds'. The conversion from milliseconds to the corresponding counter value shall be done internally by the Wdg module.⌋

[SWS_Wdg_00139] ⌈The current watchdog mode shall be taken into account when calculating the counter value from the timeout parameter.⌋

[SWS_Wdg_00140] ⌈This function shall also allow to set "0" as the time frame for triggering which will result in an (almost) immediate stop of the watchdog triggering and an (almost) instantaneous watchdog reset of the ECU. In case the counter value stored inside watchdog has the value "0", the service Wdg_SetTriggerCondition shall do nothing, which means it shall ignore the counter passed by the parameter to Wdg_SetTriggerCondition.⌋

[SWS_Wdg_00146]

Upstream requirements: SRS_BSW_00369, SRS_BSW_00386

⌈When development error detection is enabled for the module: The function Wdg_SetTriggerCondition shall check that the timeout parameter given is less or equal to the maximum timeout value (WdgMaxTimeout). If this is not the case the function shall not reload the timeout counter but raise the development error WDG_E_PARAM_TIMEOUT and return to the caller.⌋

8.3.4 Callback notifications

The Wdg module has no callback notifications.

8.4 Scheduled functions

This chapter lists all functions provided by the Wdg module and called directly by the Basic Software Module Scheduler.

The Wdg module has no scheduled functions.

8.5 Expected interfaces

This chapter lists all functions that the Wdg module requires from other modules.

8.5.1 Mandatory interfaces

This module does not require any mandatory interfaces.

8.5.2 Optional interfaces

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

[SWS_Wdg_00111] Definition of optional interfaces requested by module Wdg ⌈

API Function	*Header File*	*Description*
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)
Det_ReportError	Det.h	Service to report development errors.

⌋

In addition to the functions listed above, further functions might be used to access the external watchdog over Dio or Spi.

8.5.3 Configurable interfaces

This module does not require any configurable interfaces.

9 Sequence diagrams

9.1 Watchdog initialization, setting trigger condition and mode.

The diagram shows the sequence to initialize the Wdg module, to set the trigger condition and to change the watchdog mode. Note that this is only an example. Especially, another "client" module than the Watchdog Manager (WdgM) could set the trigger condition.

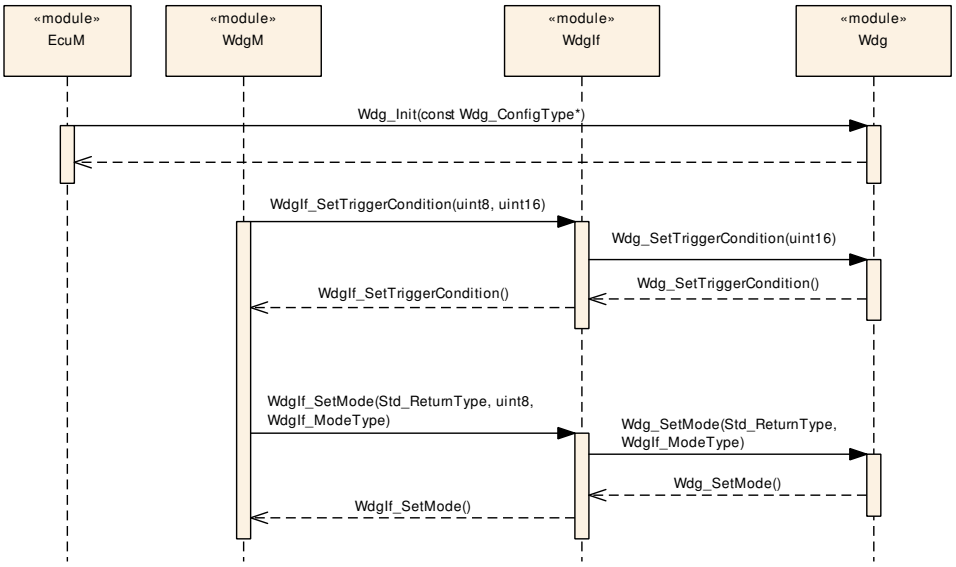


Figure 9.1: Sequence of watchdog initialization, setting trigger condition and mode switching.

9.2 Data exchange between watchdog driver and hardware

The diagram shows the sequence to trigger the watchdog hardware when the WDG servicing routine is implemented as an interrupt routine. Note that this is only an example and the triggering routine is implementation specific. For an external watchdog, the watchdog hardware cannot be accessed directly, but only via drivers of the MCAL layer, like SPI or DIO.

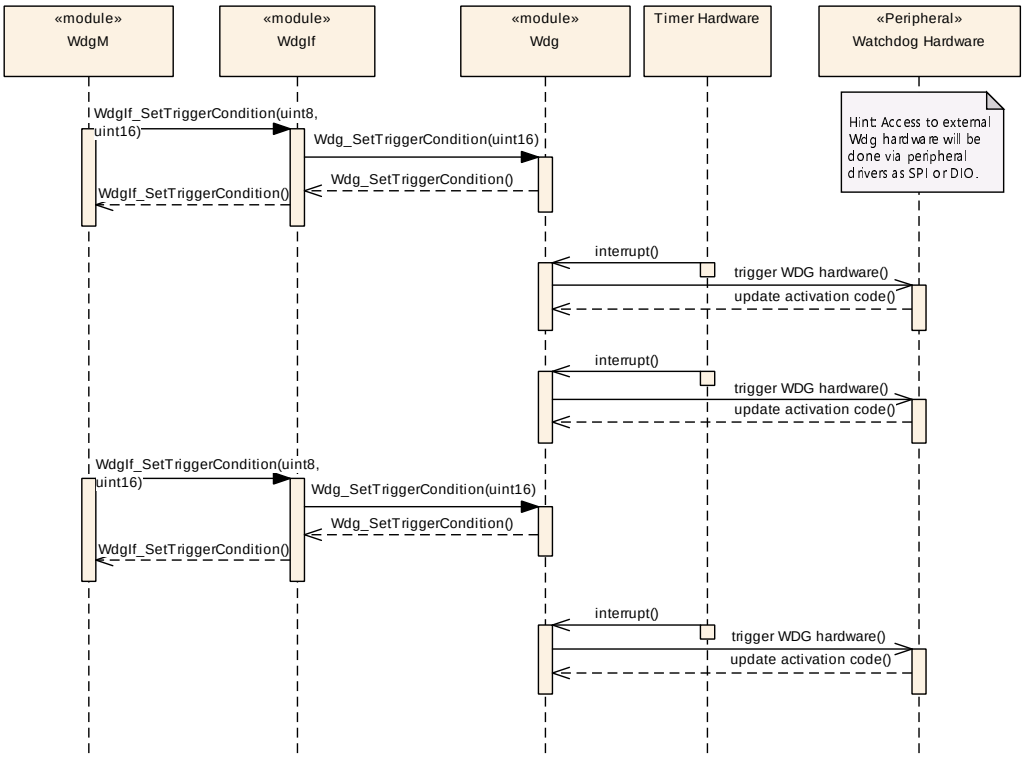


Figure 9.2: Data exchange between watchdog driver and hardware

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 Wdg.

Chapter 10.3 specifies published information of the module Wdg.

10.1 How to read this chapter

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

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.

10.2.1 Wdg

[ECUC_Wdg_00082] Definition of EcucParamConfContainerDef WdgSettingsConfig ⌈

Container Name	WdgSettingsConfig
Parent Container	Wdg
Upstream mapping defined	–
Description	Configuration items for the different watchdog settings, including those for external watchdog hardware. Note: All postbuild parameters are handled via this container.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
WdgDefaultMode	1	[ECUC_Wdg_00120]

Included Containers		
Container Name	**Multiplicity**	**Description**
WdgExternalConfiguration	0..1	Configuration items for an external watchdog hardware
WdgSettingsFast	1	Hardware dependent settings for the watchdog driver's "fast" mode.
WdgSettingsOff	1	Hardware dependent settings for the watchdog driver's "off" mode.
WdgSettingsSlow	1	Hardware dependent settings for the watchdog driver's "slow" mode.

⌋

[ECUC_Wdg_00120] Definition of EcucEnumerationParamDef WdgDefaultMode ⌈

Parameter Name	WdgDefaultMode	
Parent Container	WdgSettingsConfig	
Upstream mapping defined	–	
Description	Default mode for watchdog driver initialization. ImplementationType: WdgIf_ModeType Dependency: "Off" mode only possible if disabling the watchdog driver is allowed.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	WDGIF_FAST_MODE	Default watchdog mode is "fast"
	WDGIF_OFF_MODE	Default watchdog mode is "off"
	WDGIF_SLOW_MODE	Default watchdog mode is "slow"
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Wdg_00114] Definition of EcucParamConfContainerDef WdgGeneral ⌈

Container Name	WdgGeneral
Parent Container	Wdg
Upstream mapping defined	–
Description	All general parameters of the watchdog driver are collected here.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
WdgDevErrorDetect	1	[ECUC_Wdg_00115]
WdgDisableAllowed	1	[ECUC_Wdg_00116]
WdgIndex	1	[ECUC_Wdg_00117]
WdgInitialTimeout	1	[ECUC_Wdg_00130]
WdgMaxTimeout	1	[ECUC_Wdg_00131]
WdgRunArea	1	[ECUC_Wdg_00147]
WdgEcucPartitionRef	0..1	[ECUC_Wdg_00353]

No Included Containers

⌋

[ECUC_Wdg_00115] Definition of EcucBooleanParamDef WdgDevErrorDetect ⌈

Parameter Name	WdgDevErrorDetect
Parent Container	WdgGeneral
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_Wdg_00116] Definition of EcucBooleanParamDef WdgDisableAllowed ⌈

Parameter Name	WdgDisableAllowed
Parent Container	WdgGeneral
Upstream mapping defined	–
Description	Compile switch to allow / forbid disabling the watchdog driver during runtime. True: Disabling the watchdog driver at runtime is allowed. False: Disabling the watchdog driver at runtime is not allowed. Dependency: Safety relevant compile switch, this has to be in accordance with the corresponding settings for the watchdog manager.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Wdg_00117] Definition of EcucIntegerParamDef WdgIndex ⌈

Parameter Name	WdgIndex	
Parent Container	WdgGeneral	
Upstream mapping defined	–	
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Wdg_00130] Definition of EcucFloatParamDef WdgInitialTimeout ⌈

Parameter Name	WdgInitialTimeout	
Parent Container	WdgGeneral	
Upstream mapping defined	–	
Description	The initial timeout (sec) for the trigger condition to be initialized during Init function. It shall be not larger than WdgMaxTimeout.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.535]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Wdg_00131] Definition of EcucFloatParamDef WdgMaxTimeout ⌈

Parameter Name	WdgMaxTimeout	
Parent Container	WdgGeneral	
Upstream mapping defined	–	
Description	The maximum timeout (sec) to which the watchdog trigger condition can be initialized.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.535]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Wdg_00147] Definition of EcucEnumerationParamDef WdgRunArea ⌈

Parameter Name	WdgRunArea	
Parent Container	WdgGeneral	
Upstream mapping defined	–	
Description	Represents the watchdog driver execution area is either from ROM(Flash) or RAM as required with the particular microcontroller.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	RAM	Watchdog driver to be executed out of RAM area
	ROM	Watchdog driver to be executed out of ROM area
Pre-Compile only	true	

⌋

[ECUC_Wdg_00353] Definition of EcucReferenceDef WdgEcucPartitionRef ⌈

Parameter Name	WdgEcucPartitionRef
Parent Container	WdgGeneral
Upstream mapping defined	–

▽

△

Description	Maps the Wdg driver to zero or one ECUC partitions to make the modules API available in this partition.	
Multiplicity	0..1	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_Wdg_00073] Definition of EcucModuleDef Wdg ⌈

Module Name	Wdg
Description	Configuration of the Wdg (Watchdog driver) module.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
WdgGeneral	1	All general parameters of the watchdog driver are collected here.
WdgPublishedInformation	1	Container holding all Wdg specific published information parameters
WdgSettingsConfig	1	Configuration items for the different watchdog settings, including those for external watchdog hardware. Note: All postbuild parameters are handled via this container.

⌋

The three modes are provided as containers for the reason that they might be referred by other modules and hence no parameters are needed. However those containers might be extended by the vendor (resp. hardware) specific configuration parameters, but these could not be standardized.

10.2.2 WdgSettingsFast

[ECUC_Wdg_00112] Definition of EcucParamConfContainerDef WdgExternal Configuration ⌈

Container Name	WdgExternalConfiguration
Parent Container	WdgSettingsConfig
Upstream mapping defined	–
Description	Configuration items for an external watchdog hardware
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
WdgExternalContainerRef	0..1	[ECUC_Wdg_00113]

No Included Containers

⌋

[ECUC_Wdg_00113] Definition of EcucChoiceReferenceDef WdgExternalContainerRef ⌈

Parameter Name	WdgExternalContainerRef	
Parent Container	WdgExternalConfiguration	
Upstream mapping defined	–	
Description	Reference to either • a DioChannelGroup container in case the hardware watchdog is connected via DIO pins • an SpiSequenceConfiguration container in case the watchdog hardware is accessed via SPI • an I2CSequenceConfiguration container in case the watchdog hardware is accessed via I2C Dependency: See DIO resp. SPI SWS resp. I2C SWS	
Multiplicity	0..1	
Type	Choice reference to [DioChannelGroup, I2CSequence, SpiSequence]	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Wdg_00122] Definition of EcucParamConfContainerDef WdgSettingsOff ⌈

Container Name	WdgSettingsOff
Parent Container	WdgSettingsConfig
Upstream mapping defined	–
Description	Hardware dependent settings for the watchdog driver's "off" mode.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

No Included Containers

⌋

[ECUC_Wdg_00123] Definition of EcucParamConfContainerDef WdgSettingsSlow ⌈

Container Name	WdgSettingsSlow
Parent Container	WdgSettingsConfig
Upstream mapping defined	–
Description	Hardware dependent settings for the watchdog driver's "slow" mode.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

No Included Containers

⌋

[ECUC_Wdg_00121] Definition of EcucParamConfContainerDef WdgSettingsFast ⌈

Container Name	WdgSettingsFast
Parent Container	WdgSettingsConfig
Upstream mapping defined	–
Description	Hardware dependent settings for the watchdog driver's "fast" mode.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

No Included Containers

⌋

10.3 Published Information

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

10.3.1 WdgPublishedInformation

[ECUC_Wdg_00074] Definition of EcucParamConfContainerDef WdgPublishedInformation ⌈

Container Name	WdgPublishedInformation
Parent Container	Wdg
Upstream mapping defined	–

▽

△

Description	Container holding all Wdg specific published information parameters
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
WdgTriggerMode	1	[ECUC_Wdg_00127]

No Included Containers

⌋

[ECUC_Wdg_00127] Definition of EcucEnumerationParamDef WdgTriggerMode

⌈

Parameter Name	WdgTriggerMode	
Parent Container	WdgPublishedInformation	
Upstream mapping defined	–	
Description	Watchdog trigger mode (toggle/window/both)	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	WDG_BOTH	Window and toggle mode
	WDG_TOGGLE	Toggle mode (watchdog signal has to be toggled between high and low to serve the watchdog).
	WDG_WINDOW	Window mode (watchdog must be served within a specific time window)
Pre-Compile only	true	

⌋

WdgTriggerMode is only published for information purposes; this parameter is not used to configure the Watchdog Driver or the modules using the Watchdog Driver.

A Not applicable requirements

[SWS_Wdg_NA_00175]

Upstream requirements: SRS_BSW_00344, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00170, SRS_BSW_00419, SRS_BSW_00383, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00437, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00428, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00450, SRS_BSW_00339, SRS_BSW_00422, SRS_BSW_00417, SRS_SPAL_12056, SRS_SPAL_12267, SRS_SPAL_12462, SRS_SPAL_12463, SRS_SPAL_12068, SRS_SPAL_12069, SRS_SPAL_00157, SRS_SPAL_12063, SRS_SPAL_12075, SRS_SPAL_12067, SRS_SPAL_12077, SRS_SPAL_12078, SRS_SPAL_12265, SRS_Wdg_12167, SRS_Wdg_12168, SRS_BSW_00461, SRS_BSW_00469, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00478, SRS_BSW_00490, SRS_BSW_00491, SRS_BSW_00384

⌈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

Number	Heading
[SWS_Wdg_91000]	Definition of runtime errors in module Wdg

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

B.1.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_Wdg_00073]	Definition of EcucModuleDef Wdg
[ECUC_Wdg_00074]	Definition of EcucParamConfContainerDef WdgPublishedInformation
[ECUC_Wdg_00082]	Definition of EcucParamConfContainerDef WdgSettingsConfig
[ECUC_Wdg_00112]	Definition of EcucParamConfContainerDef WdgExternalConfiguration
[ECUC_Wdg_00113]	Definition of EcucChoiceReferenceDef WdgExternalContainerRef
[ECUC_Wdg_00114]	Definition of EcucParamConfContainerDef WdgGeneral
[ECUC_Wdg_00115]	Definition of EcucBooleanParamDef WdgDevErrorDetect
[ECUC_Wdg_00116]	Definition of EcucBooleanParamDef WdgDisableAllowed
[ECUC_Wdg_00117]	Definition of EcucIntegerParamDef WdgIndex
[ECUC_Wdg_00120]	Definition of EcucEnumerationParamDef WdgDefaultMode
[ECUC_Wdg_00121]	Definition of EcucParamConfContainerDef WdgSettingsFast
[ECUC_Wdg_00122]	Definition of EcucParamConfContainerDef WdgSettingsOff
[ECUC_Wdg_00123]	Definition of EcucParamConfContainerDef WdgSettingsSlow
[ECUC_Wdg_00127]	Definition of EcucEnumerationParamDef WdgTriggerMode
[ECUC_Wdg_00130]	Definition of EcucFloatParamDef WdgInitialTimeout
[ECUC_Wdg_00131]	Definition of EcucFloatParamDef WdgMaxTimeout
[ECUC_Wdg_00147]	Definition of EcucEnumerationParamDef WdgRunArea
[ECUC_Wdg_00353]	Definition of EcucReferenceDef WdgEcucPartitionRef
[SWS_Wdg_00016]	
[SWS_Wdg_00105]	Definition of imported datatypes of module Wdg
[SWS_Wdg_00173]	

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

B.1.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_Wdg_00119]	Definition of EcucBooleanParamDef WdgVersionInfoApi
[ECUC_Wdg_00148]	Definition of EcucParamConfContainerDef WdgDemEventParameterRefs
[ECUC_Wdg_00149]	Definition of EcucReferenceDef WDG_E_MODE_FAILED
[ECUC_Wdg_00150]	Definition of EcucReferenceDef WDG_E_DISABLE_REJECTED
[SWS_Wdg_00025]	
[SWS_Wdg_00026]	
[SWS_Wdg_00109]	Definition of API function Wdg_GetVersionInfo
[SWS_Wdg_00174]	
[SWS_Wdg_00178]	
[SWS_Wdg_00179]	
[SWS_Wdg_00181]	
[SWS_Wdg_00182]	
[SWS_Wdg_00183]	

Table B.3: Deleted Specification Items in R26-11

B.2 Traceable item history of this document according to AUTOSAR Release R24-11

B.2.1 Added Specification Items in R25-11

none

B.2.2 Changed Specification Items in R25-11

Number	Heading
[SWS_Wdg_00094]	
[SWS_Wdg_00107]	Definition of API function Wdg_SetMode
[SWS_Wdg_00171]	Definition of datatype Wdg_ConfigType

Table B.4: Changed Specification Items in R25-11

B.2.3 Deleted Specification Items 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
[ECUC_Wdg_00113]	Definition of EcucChoiceReferenceDef WdgExternalContainerRef

Table B.5: Changed Specification Items in R24-11

B.3.3 Deleted Specification Items 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 R22-11

Number	Heading
[SWS_Wdg_NA_00175]	

Table B.6: Added Specification Items in R22-11

B.4.2 Changed Specification Items in R22-11

Number	Heading
[SWS_Wdg_00010]	
[SWS_Wdg_00105]	
[SWS_Wdg_00106]	
[SWS_Wdg_00107]	
[SWS_Wdg_00109]	
[SWS_Wdg_00111]	

▽

△

Number	Heading
[SWS_Wdg_00155]	
[SWS_Wdg_00171]	

Table B.7: Changed Specification Items in R22-11

B.4.3 Deleted Specification Items in R22-11

Number	Heading
[SWS_Wdg_00175]	

Table B.8: Deleted Specification Items in R22-11

B.4.4 Added Specification Items in R23-11

none

B.4.5 Changed Specification Items in R23-11

Number	Heading
[SWS_Wdg_00105]	Definition of imported datatypes of module Wdg
[SWS_Wdg_00107]	Definition of API function Wdg_SetMode
[SWS_Wdg_00111]	Definition of optional interfaces in module Wdg
[SWS_Wdg_00171]	Definition of datatype Wdg_ConfigType

Table B.9: Changed Specification Items in R23-11

B.4.6 Deleted Specification Items in R23-11

none