

|                                   |                                  |
|-----------------------------------|----------------------------------|
| <b>Document Title</b>             | Specification of Watchdog Driver |
| <b>Document Owner</b>             | AUTOSAR                          |
| <b>Document Responsibility</b>    | AUTOSAR                          |
| <b>Document Identification No</b> | 039                              |
| <b>Document Classification</b>    | Standard                         |
|                                   |                                  |
| <b>Document Status</b>            | Final                            |
| <b>Part of AUTOSAR Standard</b>   | Classic Platform                 |
| <b>Part of Standard Release</b>   | 4.3.0                            |

| Document Change History |         |                            |   |
|-------------------------|---------|----------------------------|---|
| Date                    | Release | Changed by                 | Change Description  |
| 2016-11-30              | 4.3.0   | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>Removed chapter 10.2.1 “Variants” including req SWS_Wdg_00157, SWS_Wdg_00158 SWS_Wdg_00159</li> <li>Removed Chapter “7.8 Debugging”</li> <li>In table ECUC_Wdg_00073 added row for "Supported Config Variants"</li> <li>minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation</li> </ul> |
| 2015-07-31              | 4.2.2   | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>Debugging support marked as obsolete</li> <li>minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation</li> </ul>   |
| 2014-10-31              | 4.2.1   | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>Adapt specification of extended production errors.</li> <li>WDG_E_INIT_FAILED added (error code is referenced by SWS_BSWGeneral)</li> </ul>  |
| 2014-03-31              | 4.1.3   | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>Minor editorial changes</li> </ul>   |
| 2013-10-31              | 4.1.2   | AUTOSAR Release Management | <ul style="list-style-type: none"> <li>Shift Dem_ReportErrorStatus from mandatory to optional interfaces</li> <li>Editorial changes</li> <li>Removed chapter(s) on change documentation</li> </ul>  |

| Document Change History |         |                        |  |
|-------------------------|---------|------------------------|--|
| Date                    | Release | Changed by             | Change Description   |
| 2013-03-15              | 4.1.1   | AUTOSAR Administration | <ul style="list-style-type: none"><li>• Add chapter for production errors</li><li>• Rename MemMap.h to Wdg_MemMap.h</li><li>• Remove GPT usage</li><li>• Added Subchapter 3.x due to SWS General Rollout</li><li>• Reworked according to the new SWS_BSWGeneral</li><li>• Reworded SWS_Wdg_00018, SWS_Wdg_00019, SWS_Wdg_00052 for debugging purpose</li></ul>   |
| 2011-12-22              | 4.0.3   | AUTOSAR Administration | <ul style="list-style-type: none"><li>• DET-Error for Wdg_GetVersionInfo added</li></ul>   |
| 2010-09-30              | 3.1.5   | AUTOSAR Administration | <ul style="list-style-type: none"><li>• Requirement WDG141/WDG143 removed</li></ul>  |
| 2010-02-02              | 3.1.4   | AUTOSAR Administration | <ul style="list-style-type: none"><li>• Modifications for windowed watchdog concept</li><li>• Further maintenance for R4.0: see Chapter 11</li><li>• Legal disclaimer revised</li></ul>  |
| 2008-08-13              | 3.1.1   | AUTOSAR Administration | <ul style="list-style-type: none"><li>• Legal disclaimer revised</li></ul>   |
| 2007-12-21              | 3.0.1   | AUTOSAR Administration | <ul style="list-style-type: none"><li>• Section 5.1.2 the file include structure has been changed.</li><li>• Section 8.6.2 Dem_ReportErrorStatus added as optional interfaces.</li><li>• Rephrased the requirementsWDG019, SWS_Wdg_00031, SWS_Wdg_00034.</li><li>• Modified sequence diagrams in chapter 9.</li><li>• Document meta information extended</li><li>• Small layout adaptations made</li></ul> |

| Document Change History |         |                        |   |
|-------------------------|---------|------------------------|---|
| Date                    | Release | Changed by             | Change Description  |
| 2007-01-24              | 2.1.15  | AUTOSAR Administration | <ul style="list-style-type: none"><li>• In chapter 5.1.2 the file include structure has been changed to comply with the SPAL general include structure.</li><li>• In chapter WdgDefaultMode has been added as PC variant and WDG003 has been changed to allow passing NULL pointer.</li><li>• For WDG037 the requirement was changed to allow configuration of activation code if the H/W allows for the same.</li><li>• For SWS_Wdg_00078 the requirement was changed to add reference to SPI/DIO for accessing the external watchdog</li><li>•</li><li>• Legal disclaimer revised</li><li>• Release Notes added</li><li>• “Advice for users” revised</li><li>• “Revision Information” added</li></ul> |
| 2006-05-16              | 2.0     | AUTOSAR Administration | <ul style="list-style-type: none"><li>• Document structure adapted to common Release 2.0 SWS Template</li></ul>   |
| 2005-05-31              | 1.0     | AUTOSAR Administration | <ul style="list-style-type: none"><li>• Initial Release</li></ul>   |

## Disclaimer

This specification 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 specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification 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 specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

## Table of Contents

|       |  |    |
|-------|--|----|
| 1     | Introduction and functional overview .....             | 7  |
| 2     | Acronyms and abbreviations .....                       | 8  |
| 3     | Related documentation.....                             | 9  |
| 3.1   | Input documents .....                                  | 9  |
| 3.2   | Related standards and norms .....                      | 9  |
| 3.3   | Related specification .....                            | 9  |
| 4     | Constraints and assumptions .....                      | 10 |
| 4.1   | Limitations .....                                      | 10 |
| 4.2   | Applicability to car domains .....                     | 10 |
| 5     | Dependencies to other modules.....                     | 11 |
| 5.1   | File structure.....                                    | 11 |
| 5.1.1 | Code file structure.....                               | 11 |
| 5.1.2 | Header file structure.....                             | 12 |
| 5.1.3 | Version check .....                                    | 13 |
| 5.2   | System clock .....                                     | 13 |
| 5.3   | Onboard communication handlers.....                    | 13 |
| 6     | Requirements traceability .....                        | 14 |
| 7     | Functional specification .....                         | 21 |
| 7.1   | General design rules .....                             | 21 |
| 7.2   | Error classification .....                             | 22 |
| 7.2.1 | Development Errors .....                               | 22 |
| 7.2.2 | Runtime errors .....                                   | 22 |
| 7.2.3 | Transient Faults .....                                 | 22 |
| 7.2.4 | Production errors .....                                | 22 |
| 7.2.5 | Extended production errors.....                        | 22 |
| 7.3   | Error detection .....                                  | 23 |
| 7.4   | Error notification .....                               | 23 |
| 7.5   | External watchdog driver .....                         | 23 |
| 7.6   | Internal watchdog driver .....                         | 24 |
| 7.7   | Triggering concept to support windowed watchdogs ..... | 25 |
| 8     | API specification .....                                | 27 |
| 8.1   | Imported types.....                                    | 27 |
| 8.2   | Type definitions .....                                 | 27 |
| 8.2.1 | Wdg_ConfigType .....                                   | 27 |
| 8.3   | Function definitions.....                              | 27 |
| 8.3.1 | Wdg_Init.....  | 27 |
| 8.3.2 | Wdg_SetMode .....                                      | 29 |
| 8.3.3 | Wdg_SetTriggerCondition .....                          | 30 |
| 8.3.4 | Wdg_GetVersionInfo.....                                | 31 |
| 8.4   | Call-back Notifications .....                          | 32 |
| 8.5   | Scheduled functions .....                              | 32 |
| 8.6   | Expected interfaces .....                              | 32 |

|        |   |    |
|--------|---|----|
| 8.6.1  | Mandatory interfaces .....  | 32 |
| 8.6.2  | Optional interfaces .....   | 32 |
| 8.6.3  | Configurable interfaces .....                                     | 33 |
| 9      | Sequence diagrams .....   | 34 |
| 9.1    | Watchdog initialization, setting trigger condition and mode ..... | 34 |
| 9.2    | Data exchange between watchdog driver and hardware .....          | 35 |
| 10     | Configuration specification .....                                 | 36 |
| 10.1   | How to read this chapter .....                                    | 36 |
| 10.2   | Containers and configuration parameters .....                     | 37 |
| 10.2.1 | Wdg .....   | 37 |
| 10.2.2 | WdgDemEventParameterRefs .....                                    | 37 |
| 10.2.3 | WdgGeneral .....  | 38 |
| 10.2.4 | WdgSettingsConfig .....   | 40 |
| 10.2.5 | WdgSettingsFast .....   | 41 |
| 10.2.6 | WdgSettingsSlow .....   | 41 |
| 10.2.7 | WdgSettingsOff .....  | 42 |
| 10.2.8 | WdgExternalConfiguration .....                                    | 42 |
| 10.3   | Published information .....                                       | 42 |
| 10.3.1 | WdgPublishedInformation .....                                     | 42 |
| 11     | Not applicable requirements .....                                 | 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

Acronyms and abbreviations that have a local scope are not contained in the AUTOSAR glossary. These must appear in a local glossary.

| Abbreviation / Acronym: | Description:  |
|-------------------------|---|
| DIP                     | Digital Input/Output  |
| DET                     | Default Error Tracer  |
| DEM                     | Diagnostic Event Manager – module to handle diagnostic relevant events. |
| SPI                     | Serial Peripheral Interface   |
| WDG                     | Watchdog (module specific prefix)                                       |

Definitions needed for understanding of the concepts

| Definition: | Description:  |
|-------------|---|
| Off-Mode    | The watchdog hardware is disabled / shut down.<br>This might be necessary in order to shut down the complete ECU and not get cyclic resets from a still running external watchdog.<br>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.<br>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.<br>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. |



## 3 Related documentation

### 3.1 Input documents

- [1] Layered Software Architecture  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf
- [2] General Requirements on Basic Software Modules  
AUTOSAR\_SRS\_BSWGeneral.pdf
- [3] General Requirements on SPAL  
AUTOSAR\_SRS\_SPALGeneral.pdf
- [4] Requirements on Watchdog Driver  
AUTOSAR\_SRS\_WatchdogDriver.pdf
- [5] Specification of Watchdog Interface  
AUTOSAR\_SWS\_WatchdogInterface.pdf
- [6] Basic Software Module Description Template  
AUTOSAR\_TPS\_BSWModuleDescriptionTemplate.pdf
- [7] Specification of RTE Software Specification of Watchdog Driver  
AUTOSAR\_SWS\_RTE.pdf
- [8] List of Basic Software Modules  
AUTOSAR\_TR\_BSWModuleList
- [9] General Specification of Basic Software Modules  
AUTOSAR\_SWS\_BSWGeneral.pdf

### 3.2 Related standards and norms

None

### 3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [9] (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 [1]).

**[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]** [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
- Wdg\_Irq.c – for holding the interrupt frames in case an internal watchdog servicing is implemented as interrupt routine (and not via timer callback)

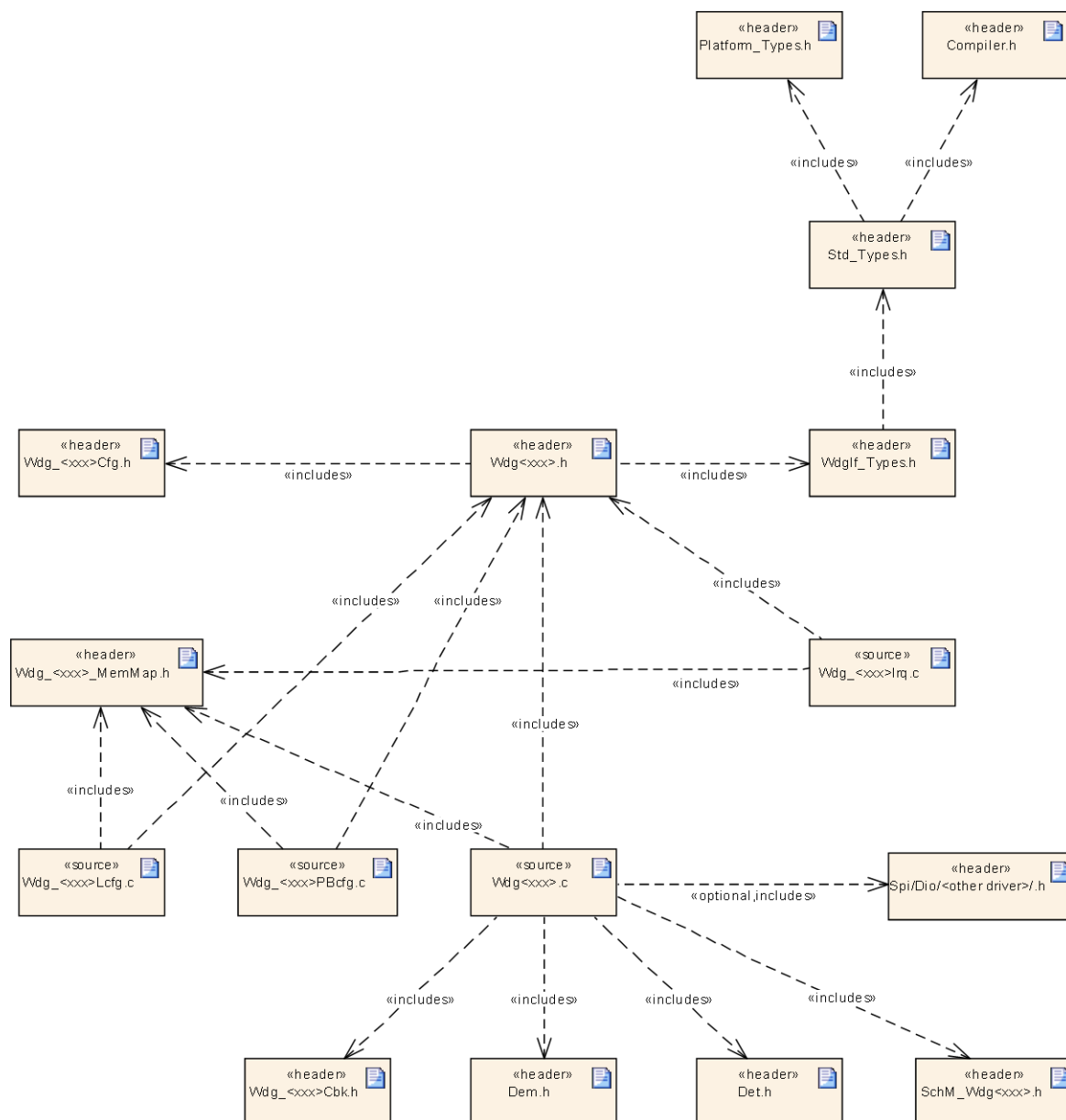
These files shall contain all link time and post-build time configurable parameters.]  
(SRS\_BSW\_00346, SRS\_BSW\_00158, SRS\_BSW\_00314, SRS\_SPAL\_12263)

Note: These names are required by [SRS\\_BSW\\_00314](#) and [SRS\\_BSW\\_00346](#)

**[SWS\_Wdg\_00169]** [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](#).]  
(SRS\_BSW\_00347)

### 5.1.2 Header file structure

**[SWS\_Wdg\_00061]** [The Wdg module shall adhere to the following file structure:



### Figure 1: File include structure

| (SRS\_BSW\_00345, SRS\_BSW\_00159, SRS\_BSW\_00381, SRS\_BSW\_00412, SRS\_BSW\_00346, SRS\_BSW\_00158, SRS\_BSW\_00301)

Notes to the figure:

- The possible name expansion for multiple driver modules are indicated as <xxx> in Figure 1.
- Since the API names are also to be expanded, the header Wdg<xxx>.h will become instance specific.

- Wdg<xxx>.h contains the pre-compile configuration macros. It is not expected that, source code is required to implement the pre-compile configuration for the watchdog driver module. This file also contains – if required – references to the c-data for link-time and/or post-build configuration.
- Wdg\_<xxx>Cbk.h contains the declaration of callback functions from other modules implemented by the Wdg module (see [SRS\\_BSW\\_00370](#)). This file is mandatory, even if there are no callback functions required.
- SchM\_Wdg<xxx>.h is a mandatory include file (see [SRS\\_BSW\\_00335](#)) provided by the RTE generator. Though the Watchdog Driver has no scheduled function, this header is e.g. needed, if the Wdg module defines critical section.
- The need to include headers from SPI-, DIO- or other drivers depends on how the watchdog is serviced and the watchdog hardware is accessed, see chapters 7.5 and 7.6.

[SWS\_Wdg\_00170] [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](#).]  
(SRS\_BSW\_00347)

Note:

In case of multiple watchdog driver instances, the Event Id symbols for production errors defined in this specification (see [SWS\\_Wdg\\_00010](#) and [ECUC\\_Wdg\\_00148](#)) might be expanded in the configuration of the DEM in order to make them unique.

### 5.1.3 Version check

For details refer to the chapter 5.1.8 “Version Check” in *SWS\_BSWGeneral*.

## 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 traceability

| 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_00005 | Modules of the $\mu$ C Abstraction Layer (MCAL) may not have hard coded horizontal interfaces  | SWS_Wdg_00175                |
| SRS_BSW_00006 | The source code of software modules above the $\mu$ C Abstraction Layer (MCAL) shall not be processor and compiler dependent.                  | SWS_Wdg_00175                |
| SRS_BSW_00007 | All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.   | SWS_Wdg_00175                |
| SRS_BSW_00009 | All Basic SW Modules shall be documented according to a common standard.   | SWS_Wdg_00175                |
| SRS_BSW_00010 | The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.                    | SWS_Wdg_00175                |
| 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_00158 | All modules of the AUTOSAR Basic Software shall strictly separate configuration from implementation  | SWS_Wdg_00061, SWS_Wdg_00079 |
| SRS_BSW_00159 | All modules of the AUTOSAR Basic Software shall support a tool based configuration   | SWS_Wdg_00061                |
| SRS_BSW_00161 | The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers | SWS_Wdg_00175                |
| SRS_BSW_00162 | The AUTOSAR Basic Software shall provide a hardware abstraction layer  | SWS_Wdg_00175                |
| SRS_BSW_00164 | The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules                             | SWS_Wdg_00166                |
| SRS_BSW_00167 | All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks                             | SWS_Wdg_00086                |
| SRS_BSW_00168 | SW components shall be tested by a function defined in a common API in the Basis-SW  | SWS_Wdg_00175                |
| SRS_BSW_00170 | The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities,                                      | SWS_Wdg_00175                |

|               |   |   |
|---------------|---|---|
|               | driver demands  |   |
| SRS_BSW_00172 | The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system    | SWS_Wdg_00175   |
| SRS_BSW_00301 | All AUTOSAR Basic Software Modules shall only import the necessary information  | SWS_Wdg_00061   |
| SRS_BSW_00302 | All AUTOSAR Basic Software Modules shall only export information needed by other modules  | SWS_Wdg_00175   |
| SRS_BSW_00304 | All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types                                | SWS_Wdg_00175   |
| SRS_BSW_00306 | AUTOSAR Basic Software Modules shall be compiler and platform independent   | SWS_Wdg_00175   |
| SRS_BSW_00307 | Global variables naming convention  | SWS_Wdg_00175   |
| SRS_BSW_00308 | AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file                                | SWS_Wdg_00175   |
| SRS_BSW_00309 | All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword | SWS_Wdg_00175   |
| SRS_BSW_00312 | Shared code shall be reentrant  | SWS_Wdg_00175   |
| SRS_BSW_00314 | All internal driver modules shall separate the interrupt frame definition from the service routine                                  | SWS_Wdg_00079   |
| SRS_BSW_00321 | The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules                                  | SWS_Wdg_00175   |
| SRS_BSW_00323 | All AUTOSAR Basic Software Modules shall check passed API parameters for validity   | SWS_Wdg_00025, SWS_Wdg_00026, SWS_Wdg_00090, SWS_Wdg_00091, SWS_Wdg_00092 |
| SRS_BSW_00325 | The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short                   | SWS_Wdg_00166   |
| SRS_BSW_00327 | Error values naming convention  | SWS_Wdg_00010, SWS_Wdg_00180, SWS_Wdg_00181, SWS_Wdg_00182, SWS_Wdg_00183 |
| SRS_BSW_00328 | All AUTOSAR Basic Software Modules shall avoid the duplication of code  | SWS_Wdg_00175   |
| SRS_BSW_00330 | It shall be allowed to use macros instead of functions where source code is used and runtime is critical                            | SWS_Wdg_00175   |
| SRS_BSW_00331 | All Basic Software Modules shall strictly separate error and status information   | SWS_Wdg_00010, SWS_Wdg_00180, SWS_Wdg_00181, SWS_Wdg_00182, SWS_Wdg_00183 |
| SRS_BSW_00333 | For each callback function it shall be  | SWS_Wdg_00175   |

|               |  |   |
|---------------|--|---|
|               | specified if it is called from interrupt context or not  |   |
| SRS_BSW_00334 | All Basic Software Modules shall provide an XML file that contains the meta data   | SWS_Wdg_00175                               |
| 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_00339 | Reporting of production relevant error status  | SWS_Wdg_00175                               |
| SRS_BSW_00341 | Module documentation shall contains all needed informations  | SWS_Wdg_00175                               |
| 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_00344 | BSW Modules shall support link-time configuration  | SWS_Wdg_00175                               |
| SRS_BSW_00345 | BSW Modules shall support pre-compile configuration  | SWS_Wdg_00061                               |
| SRS_BSW_00346 | All AUTOSAR Basic Software Modules shall provide at least a basic set of module files                                    | SWS_Wdg_00061, SWS_Wdg_00079                |
| SRS_BSW_00347 | A Naming separation of different instances of BSW drivers shall be in place  | SWS_Wdg_00169, SWS_Wdg_00170, SWS_Wdg_00172 |
| SRS_BSW_00348 | All AUTOSAR standard types and constants shall be placed and organized in a standard type header file                    | SWS_Wdg_00175                               |
| 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_00353 | All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header | SWS_Wdg_00175                               |
| SRS_BSW_00358 | The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void                          | SWS_Wdg_00106                               |
| SRS_BSW_00359 | All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible               | SWS_Wdg_00175                               |
| SRS_BSW_00360 | AUTOSAR Basic Software Modules callback functions are allowed to have parameters   | SWS_Wdg_00175                               |
| SRS_BSW_00361 | All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a                  | SWS_Wdg_00175                               |



|               |  |   |
|---------------|--|---|
|               | compiler specific type and keyword header  |   |
| SRS_BSW_00371 | The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules  | SWS_Wdg_00175   |
| SRS_BSW_00373 | The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention                             | SWS_Wdg_00175   |
| SRS_BSW_00375 | Basic Software Modules shall report wake-up reasons  | SWS_Wdg_00175   |
| SRS_BSW_00377 | A Basic Software Module can return a module specific types   | SWS_Wdg_00175   |
| SRS_BSW_00378 | AUTOSAR shall provide a boolean type   | SWS_Wdg_00175   |
| SRS_BSW_00381 | The pre-compile time parameters shall be placed into a separate configuration header file  | SWS_Wdg_00061   |
| SRS_BSW_00383 | The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description | SWS_Wdg_00175   |
| SRS_BSW_00385 | List possible error notifications  | SWS_Wdg_00010, SWS_Wdg_00180, SWS_Wdg_00181, SWS_Wdg_00182, SWS_Wdg_00183 |
| SRS_BSW_00400 | Parameter shall be selected from multiple sets of parameters after code has been loaded and started  | SWS_Wdg_00001   |
| SRS_BSW_00401 | Documentation of multiple instances of configuration parameters shall be available   | SWS_Wdg_00175   |
| SRS_BSW_00404 | BSW Modules shall support post-build configuration   | SWS_Wdg_00175   |
| SRS_BSW_00405 | BSW Modules shall support multiple configuration sets  | SWS_Wdg_00175   |
| SRS_BSW_00406 | A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called | SWS_Wdg_00019   |
| SRS_BSW_00410 | Compiler switches shall have defined values  | SWS_Wdg_00175   |
| SRS_BSW_00412 | References to c-configuration parameters shall be placed into a separate h-file  | SWS_Wdg_00061   |
| SRS_BSW_00413 | An index-based accessing of the instances of BSW modules shall be done   | SWS_Wdg_00175   |
| SRS_BSW_00414 | Init functions shall have a pointer to a configuration structure as single parameter   | SWS_Wdg_00106, SWS_Wdg_00171  |
| SRS_BSW_00415 | Interfaces which are provided  | SWS_Wdg_00175   |

|               |  |               |
|---------------|--|---------------|
|               | exclusively for one module shall be separated into a dedicated header file   |               |
| SRS_BSW_00416 | The sequence of modules to be initialized shall be configurable  | SWS_Wdg_00175 |
| SRS_BSW_00417 | Software which is not part of the SW-C shall report error events only after the DEM is fully operational.                        | SWS_Wdg_00175 |
| SRS_BSW_00419 | If a pre-compile time configuration parameter is implemented as "const" it should be placed into a separate c-file               | SWS_Wdg_00175 |
| SRS_BSW_00422 | Pre-de-bouncing of error status information is done within the DEM   | SWS_Wdg_00175 |
| SRS_BSW_00423 | BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template                                     | SWS_Wdg_00175 |
| SRS_BSW_00424 | BSW module main processing functions shall not be allowed to enter a wait state  | SWS_Wdg_00175 |
| SRS_BSW_00425 | The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects           | SWS_Wdg_00175 |
| SRS_BSW_00426 | BSW Modules shall ensure data consistency of data which is shared between BSW modules  | SWS_Wdg_00040 |
| SRS_BSW_00427 | ISR functions shall be defined and documented in the BSW module description template   | SWS_Wdg_00166 |
| SRS_BSW_00428 | A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence                   | SWS_Wdg_00175 |
| SRS_BSW_00429 | BSW modules shall be only allowed to use OS objects and/or related OS services   | SWS_Wdg_00040 |
| SRS_BSW_00432 | Modules should have separate main processing functions for read/receive and write/transmit data path                             | SWS_Wdg_00175 |
| SRS_BSW_00433 | Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler                           | SWS_Wdg_00175 |
| SRS_BSW_00437 | Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup               | SWS_Wdg_00175 |
| SRS_BSW_00439 | Enable BSW modules to handle interrupts  | SWS_Wdg_00166 |
| SRS_BSW_00440 | The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via Rte_Call API | SWS_Wdg_00175 |
| SRS_BSW_00441 | Naming convention for type, macro and function   | SWS_Wdg_00175 |

|                |  |  |
|----------------|--|--|
| SRS_BSW_00447  | Standardizing Include file structure of BSW Modules Implementing Autosar Service                                       | SWS_Wdg_00175  |
| SRS_BSW_00449  | BSW Service APIs used by Autosar Application Software shall return a Std_ReturnType                                    | SWS_Wdg_00175  |
| SRS_BSW_00450  | A Main function of a un-initialized module shall return immediately  | SWS_Wdg_00175  |
| SRS_BSW_00466  | Classification of extended production errors   | SWS_Wdg_00180, SWS_Wdg_00181, SWS_Wdg_00182, SWS_Wdg_00183 |
| SRS_SPAL_00157 | All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers | SWS_Wdg_00175  |
| SRS_SPAL_12056 | All driver modules shall allow the static configuration of notification mechanism                                      | SWS_Wdg_00175  |
| SRS_SPAL_12057 | All driver modules shall implement an interface for initialization   | SWS_Wdg_00100, SWS_Wdg_00101                               |
| SRS_SPAL_12063 | All driver modules shall only support raw value mode   | SWS_Wdg_00175  |
| 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_12067 | All driver modules shall set their wake-up conditions depending on the selected operation mode                         | SWS_Wdg_00175  |
| SRS_SPAL_12068 | The modules of the MCAL shall be initialized in a defined sequence   | SWS_Wdg_00175  |
| SRS_SPAL_12069 | All drivers of the SPAL that wake up from a wake-up interrupt shall report the wake-up reason                          | SWS_Wdg_00175  |
| SRS_SPAL_12075 | All drivers with random streaming capabilities shall use application buffers   | SWS_Wdg_00175  |
| SRS_SPAL_12077 | All drivers shall provide a non blocking implementation  | SWS_Wdg_00175  |
| SRS_SPAL_12078 | The drivers shall be coded in a way that is most efficient in terms of memory and runtime resources                    | SWS_Wdg_00175  |
| 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_12129 | The ISRs shall be responsible for resetting the interrupt flags and calling the according notification function        | SWS_Wdg_00166  |
| SRS_SPAL_12163 | All driver modules shall implement an interface for de-initialization  | SWS_Wdg_00025, SWS_Wdg_00026, SWS_Wdg_00031                |
| SRS_SPAL_12263 | The implementation of all driver modules shall allow the configuration of specific module parameter types at link      | SWS_Wdg_00079  |

|                |  |   |
|----------------|--|---|
|                | time   |   |
| SRS_SPAL_12265 | Configuration data shall be kept constant  | SWS_Wdg_00175   |
| SRS_SPAL_12267 | Wakeup sources shall be initialized by MCAL drivers and/or the MCU driver  | SWS_Wdg_00175   |
| 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  |
| SRS_SPAL_12461 | Specific rules regarding initialization of controller registers shall apply to all driver implementations                                | SWS_Wdg_00100, SWS_Wdg_00101  |
| SRS_SPAL_12462 | The register initialization settings shall be published  | SWS_Wdg_00175   |
| SRS_SPAL_12463 | The register initialization settings shall be combined and forwarded   | SWS_Wdg_00175   |
| 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, SWS_Wdg_00166 |
| 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_12106  | The disabling of the watchdog shall not be possible  | SWS_Wdg_00025, SWS_Wdg_00026  |
| 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   |
| SRS_Wdg_12167  | The external watchdog driver shall have a semantically identical API as an internal watchdog driver                                      | SWS_Wdg_00175   |
| SRS_Wdg_12168  | The source code of the external watchdog driver shall be independent from the underlying microcontroller                                 | SWS_Wdg_00175   |

## 7 Functional specification

### 7.1 General design rules

**[SWS\_Wdg\_00086]** [The Wdg module shall statically check the configuration parameters (at the latest during compile time) for correctness.] (SRS\_BSW\_00167, SRS\_BSW\_00004)

**[SWS\_Wdg\_00031]** [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_SetMode` routine with OFF mode parameter.] (SRS\_BSW\_00336, SRS\_SPAL\_12163)

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\_00034]** [The start address of the watchdog trigger routine shall be statically configurable to a fixed memory location by the user. The user needs to take care that Configured memory location is valid for the platform on which driver is being implemented on. This configuration parameter shall only be given if supported/needed by the hardware.] ()

Rationale: This allows the watchdog device to identify the correct trigger input if supported by the hardware.

**[SWS\_Wdg\_00040]** [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](#).] (SRS\_BSW\_00426, 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 Error classification

### 7.2.1 Development Errors

[SWS\_Wdg\_00010] [The Wdg module shall detect the following development errors and exceptions depending on its configuration (development/production mode):

| Type or error   | Related error code                     | Value [hex]  |
|---|--|--------------|
| 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<br>WDG_E_PARAM_CONFIG | 0x11<br>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         |

] (SRS\_BSW\_00337, SRS\_BSW\_00350, SRS\_BSW\_00385, SRS\_BSW\_00327, SRS\_BSW\_00331)

### 7.2.2 Runtime errors

There are no runtime errors.

### 7.2.3 Transient Faults

There are no transient errors.

### 7.2.4 Production errors

There are no production errors.

### 7.2.5 Extended production errors

[SWS\_Wdg\_00178][

|                              |  |   |
|------------------------------|--|---|
| <b>Error Name:</b>           | WDG_E_MODE_FAILED  |   |
| <b>Short Description:</b>    | Setting watchdog mode failed   |   |
| <b>Long Description:</b>     | Setting a watchdog mode failed (during initialization or mode switch). |   |
| <b>Detection Criteria:</b>   | Fail   | Setting watchdog mode failed (see <a href="#">SWS_Wdg_00180</a> )     |
|                              | Pass   | Setting watchdog mode not failed (see <a href="#">SWS_Wdg_00181</a> ) |
| <b>Secondary Parameters:</b> | N/A  |   |
| <b>Time Required:</b>        | N/A  |   |
| <b>Monitor Frequency</b>     | Depends on upper layer   |   |

] ()

[SWS\_Wdg\_00180] The extended production error WDG\_E\_MODE\_FAILED shall be reported with FAILED when setting of the watchdog mode failed.]  
(SRS\_BSW\_00327, SRS\_BSW\_00331, SRS\_BSW\_00466, SRS\_BSW\_00385)

[SWS\_Wdg\_00181] The extended production error WDG\_E\_MODE\_FAILED shall be reported with PASSED when setting of the watchdog mode not failed.]  
(SRS\_BSW\_00327, SRS\_BSW\_00331, SRS\_BSW\_00466, SRS\_BSW\_00385)

[SWS\_Wdg\_00179]

|                              |  |   |
|------------------------------|--|---|
| <b>Error Name:</b>           | WDG_E_DISABLE_REJECTED   |   |
| <b>Short Description:</b>    | Disabling watchdog mode failed   |   |
| <b>Long Description:</b>     | Initialization or watchdog mode switch failed because it would disable the watchdog though this is not allowed in this configuration |   |
| <b>Detection Criteria:</b>   | Fail   | Disabling watchdog mode failed (see <a href="#">SWS_Wdg_00182</a> )     |
|                              | Pass   | Disabling watchdog mode not failed (see <a href="#">SWS_Wdg_00183</a> ) |
| <b>Secondary Parameters:</b> | N/A  |   |
| <b>Time Required:</b>        | N/A  |   |
| <b>Monitor Frequency</b>     | Depends on upper layer   |   |

] ()

[SWS\_Wdg\_00182] The extended production error WDG\_E\_DISABLE\_REJECTED shall be reported with FAILED when disabling of the watchdog mode failed.] (SRS\_BSW\_00327, SRS\_BSW\_00331, SRS\_BSW\_00466, SRS\_BSW\_00385)

[SWS\_Wdg\_00183] The extended production error WDG\_E\_DISABLE\_REJECTED shall be reported with PASSED when disabling of the watchdog mode not failed.] (SRS\_BSW\_00327, SRS\_BSW\_00331, SRS\_BSW\_00466, SRS\_BSW\_00385)

## 7.3 Error detection

For details refer to the chapter 7.3 “Error Detection” in *SWS\_BSWGeneral*.

## 7.4 Error notification

For details refer to the chapter 7.4 “Error notification” in *SWS\_BSWGeneral*.

## 7.5 External watchdog driver



**[SWS\_Wdg\_00076]** [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.] (SRS\_SPAL\_12092)

**[SWS\_Wdg\_00162]** [The routine servicing an external watchdog shall 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 [1]), which excludes direct hardware access.

This architectural discrepancy will be resolved in an upcoming release.

**[SWS\_Wdg\_00077]** [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.] (SRS\_Wdg\_12165)

**[SWS\_Wdg\_00078]** [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.] (SRS\_Wdg\_12166)

## 7.6 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 [1]), which allows direct hardware access.

**[SWS\_Wdg\_00166]** [The routine servicing an internal watchdog shall be implemented as an interrupt routine driven by a hardware timer] (SRS\_BSW\_00427, SRS\_BSW\_00164, SRS\_BSW\_00325, SRS\_BSW\_00439, SRS\_SPAL\_12129, SRS\_Wdg\_12019)

Notes:

In both cases, the watchdog servicing routine runs in interrupt context.

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 [2] and [2] [3] ([SRS BSW 00427](#), [SRS BSW 00325](#), [SRS BSW 00439](#), [SRS BSW 00314](#), [SRS BSW 00429](#), [SRS SPAL 12129](#)).



## 7.7 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.

As already stated by [SWS\\_Wdg\\_00162](#) and [SWS\\_Wdg\\_00166](#), the time base for triggering the watchdog shall be provided by means of a hardware. This ensures minimum timing jitter.

These two requirements [SWS\\_Wdg\\_00162](#) and [SWS\\_Wdg\\_00166](#) also imply that servicing of the watchdog hardware is done directly from a timer ISR. This 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]** [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).] (SRS\_Wdg\_12019)

**[SWS\_Wdg\_00134]** [If the trigger counter is greater than zero, the watchdog servicing routine shall decrement the trigger counter and trigger the hardware watchdog.] (SRS\_Wdg\_12019)

**[SWS\_Wdg\_00135]** [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).] (SRS\_Wdg\_12019)

**[SWS\_Wdg\_00093]** [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.] (SRS\_Wdg\_12019)

**[SWS\_Wdg\_00094]** [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 (see **Figure 3**).

] (SRS\_Wdg\_12019)

**[SWS\_Wdg\_00095]** [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.] (SRS\_Wdg\_12019)

**[SWS\_Wdg\_00035]** [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`.] (SRS\_BSW\_00337)

**[SWS\_Wdg\_00052]** [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.] (SRS\_BSW\_00337)

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.

## 8 API specification

[SWS\_Wdg\_00172] [If more than one watchdog driver instance exists 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](#). ] (SRS\_BSW\_00347)

### 8.1 Imported types

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

[SWS\_Wdg\_00105] [

| Module    | Imported Type       |
|-----------|---------------------|
| Dem       | Dem_EventIdType     |
|           | Dem_EventStatusType |
| Std_Types | Std_ReturnType      |
|           | Std_VersionInfoType |
| WdgIf     | WdgIf_ModeType      |

] ()

### 8.2 Type definitions

#### 8.2.1 Wdg\_ConfigType

[SWS\_Wdg\_00171] [

|                     |   |  |
|---------------------|---|--|
| <b>Name:</b>        | Wdg_ConfigType  |  |
| <b>Type:</b>        | Structure   |  |
| <b>Range:</b>       | Hardware dependent structure  | Structure to hold the watchdog driver configuration set. |
| <b>Description:</b> | Used for pointers to structures holding configuration data provided to the Wdg module initialization routine for configuration of the module and watchdog hardware. |  |

] (SRS\_BSW\_00414)

### 8.3 Function definitions

#### 8.3.1 Wdg\_Init

[SWS\_Wdg\_00106] [

|                         |   |                               |
|-------------------------|---|-------------------------------|
| <b>Service name:</b>    | Wdg_Init  |                               |
| <b>Syntax:</b>          | <pre>void Wdg_Init(     const Wdg_ConfigType* ConfigPtr )</pre> |                               |
| <b>Service ID[hex]:</b> | 0x00  |                               |
| <b>Sync/Async:</b>      | Synchronous   |                               |
| <b>Reentrancy:</b>      | Non Reentrant   |                               |
| <b>Parameters (in):</b> | ConfigPtr   | Pointer to configuration set. |
| <b>Parameters</b>       | None  |                               |

|                          |                         |
|--------------------------|-------------------------|
| <b>(inout):</b>          |                         |
| <b>Parameters (out):</b> | None                    |
| <b>Return value:</b>     | None                    |
| <b>Description:</b>      | Initializes the module. |

] (SRS\_BSW\_00358, SRS\_BSW\_00414)

**[SWS\_Wdg\_00001]** [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.] (SRS\_BSW\_00400, SRS\_BSW\_00101, SRS\_Wdg\_12105)

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]** [The `Wdg_Init` function shall initialize all global variables of the Wdg module and set the default watchdog mode and initial timeout period] (SRS\_SPAL\_12057, SRS\_SPAL\_12125, SRS\_SPAL\_12461, SRS\_Wdg\_12105)

**[SWS\_Wdg\_00101]** [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.] (SRS\_SPAL\_12057, SRS\_SPAL\_12125, SRS\_SPAL\_12461, SRS\_Wdg\_12105)

**[SWS\_Wdg\_00025]** [If disabling the watchdog is not allowed (because pre-compile configuration parameter `WdgDisableAllowed==OFF`) and if the default mode given in the provided configuration set disables the watchdog, the `Wdg_Init` function shall not execute the initialization but raise the extended production error `WDG_E_DISABLE_REJECTED`.] (SRS\_BSW\_00323, SRS\_SPAL\_12163, SRS\_Wdg\_12106)

**[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 extended production error `WDG_E_MODE_FAILED`.] ()

**[SWS\_Wdg\_00090]** [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`.] (SRS\_BSW\_00323, SRS\_SPAL\_12448)

**[SWS\_Wdg\_00019]** [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.] (SRS\_BSW\_00406, SRS\_BSW\_00335)

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] [

|                            |  |  |
|----------------------------|--|--|
| <b>Service name:</b>       | Wdg_SetMode  |  |
| <b>Syntax:</b>             | <pre>Std_ReturnType Wdg_SetMode(     WdgIf_ModeType Mode )</pre> |  |
| <b>Service ID[hex]:</b>    | 0x01   |  |
| <b>Sync/Async:</b>         | Synchronous  |  |
| <b>Reentrancy:</b>         | Non Reentrant  |  |
| <b>Parameters (in):</b>    | Mode   | One of the following statically configured modes:<br>1. WDGIF_OFF_MODE<br>2. WDGIF_SLOW_MODE<br>3. WDGIF_FAST_MODE |
| <b>Parameters (inout):</b> | None   |  |
| <b>Parameters (out):</b>   | None   |  |
| <b>Return value:</b>       | Std_ReturnType   | Std_ReturnType.  |
| <b>Description:</b>        | Switches the watchdog into the mode Mode.                        |  |

] ()

[SWS\_Wdg\_00160] [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 ] (SRS\_Wdg\_12015, SRS\_Wdg\_12018)

[SWS\_Wdg\_00051] [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.] (SRS\_Wdg\_12015)

[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] [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 extended production error `WDG_E_MODE_FAILED`.] (SRS\_SPAL\_12064)

**[SWS\_Wdg\_00026]** [If disabling the watchdog is not allowed (e.g. in safety relevant systems, see [ECUC\\_Wdg\\_00115](#)) the `Wdg_SetMode` function shall check whether the settings for the requested mode would disable the watchdog. In this case, the function shall not execute the mode switch but raise the extended production error `WDG_E_DISABLE_REJECTED` and return with the value `E_NOT_OK`.] (SRS\_BSW\_00323, SRS\_SPAL\_12163, SRS\_Wdg\_12106)

**[SWS\_Wdg\_00091]** [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` and return with the value `E_NOT_OK`] (SRS\_BSW\_00323, SRS\_SPAL\_12448)

**[SWS\_Wdg\_00092]** [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` and return with the value `E_NOT_OK`. ] (SRS\_BSW\_00323, SRS\_SPAL\_12448)

**[SWS\_Wdg\_00017]** [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` and return with the value `E_NOT_OK`. ] (SRS\_BSW\_00335, SRS\_SPAL\_12064, SRS\_SPAL\_12448)

**[SWS\_Wdg\_00018]** [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. ] (SRS\_BSW\_00335)

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]** [

|                      |   |
|----------------------|---|
| <b>Service name:</b> | <code>Wdg_SetTriggerCondition</code>                          |
| <b>Syntax:</b>       | <pre>void Wdg_SetTriggerCondition(     uint16 timeout )</pre> |



|                            |  |
|----------------------------|--|
| <b>Service ID[hex]:</b>    | 0x03   |
| <b>Sync/Async:</b>         | Synchronous  |
| <b>Reentrancy:</b>         | Non Reentrant  |
| <b>Parameters (in):</b>    | timeout    Timeout value (milliseconds) for setting the trigger counter. |
| <b>Parameters (inout):</b> | None   |
| <b>Parameters (out):</b>   | None   |
| <b>Return value:</b>       | None   |
| <b>Description:</b>        | Sets the timeout value for the trigger counter.                          |

] (SRS\_BSW\_00343)

**[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]** [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 Wdg\_GetVersionInfo

**[SWS\_Wdg\_00109]** [

|                            |  |
|----------------------------|--|
| <b>Service name:</b>       | <code>Wdg_GetVersionInfo</code>  |
| <b>Syntax:</b>             | <pre>void Wdg_GetVersionInfo(<br/>    Std_VersionInfoType* versioninfo<br/>)</pre> |
| <b>Service ID[hex]:</b>    | 0x04   |
| <b>Sync/Async:</b>         | Synchronous  |
| <b>Reentrancy:</b>         | Reentrant  |
| <b>Parameters (in):</b>    | None   |
| <b>Parameters (inout):</b> | None   |

|                          |  |   |
|--------------------------|--|---|
| <b>Parameters (out):</b> | versioninfo                                    | Pointer to where to store the version information of this module. |
| <b>Return value:</b>     | None   |   |
| <b>Description:</b>      | Returns the version information of the module. |   |

] ()

**[SWS\_Wdg\_00174]** [If development error detection is enabled for the Wdg Driver module, the function `Wdg_GetVersionInfo` shall raise `WDG_E_PARAM_POINTER`, if the argument is a NULL pointer and return without any action.] ()

## 8.4 Call-back Notifications

This chapter lists all functions provided by the Wdg module to lower layer modules.

The Wdg module has no call back notifications

## 8.5 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.

## 8.6 Expected interfaces

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

### 8.6.1 Mandatory interfaces

This module does not require any mandatory interfaces.

### 8.6.2 Optional interfaces

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

**[SWS\_Wdg\_00111]** [

| API function                    | Description   |
|---------------------------------|---|
| <code>Dem_SetEventStatus</code> | Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling <code>Dem_SetEventStatus</code> can safely ignore the return value. |
| <code>Det_ReportError</code>    | 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.

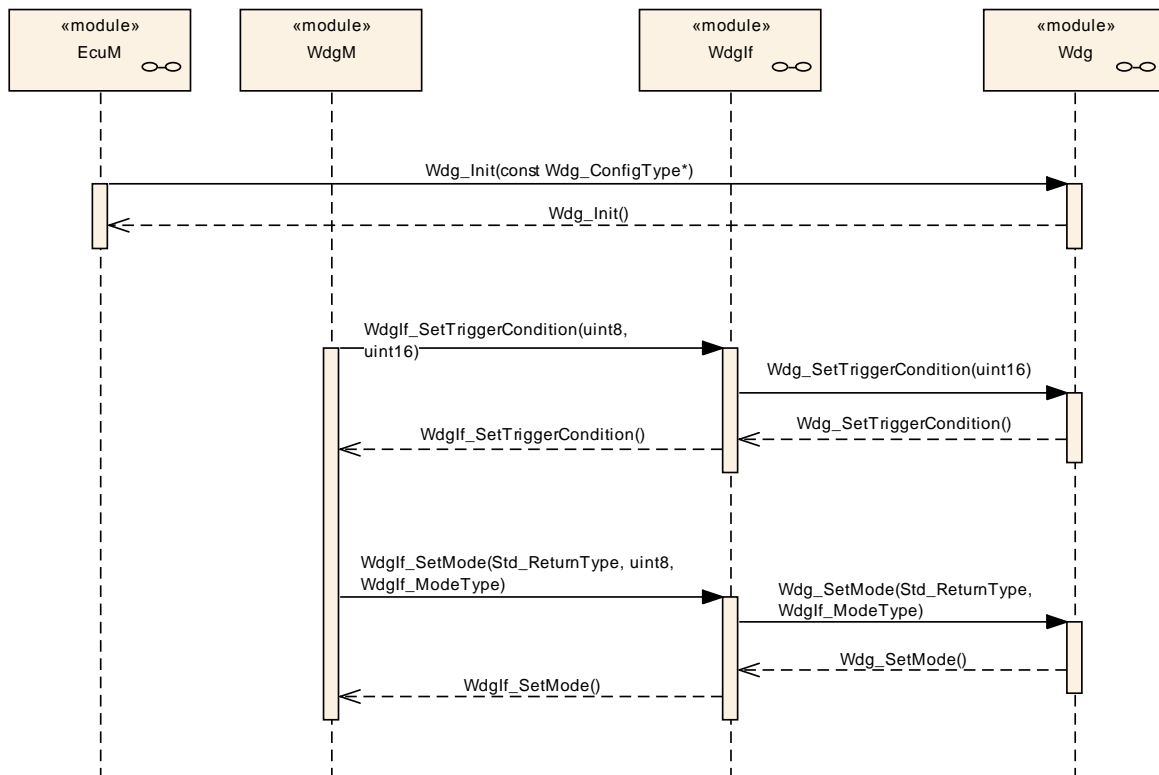
### **8.6.3 Configurable interfaces**

This module does not require any configurable interfaces.

## 9 Sequence diagrams

### 9.1 Watchdog initialization, setting trigger condition and mode.

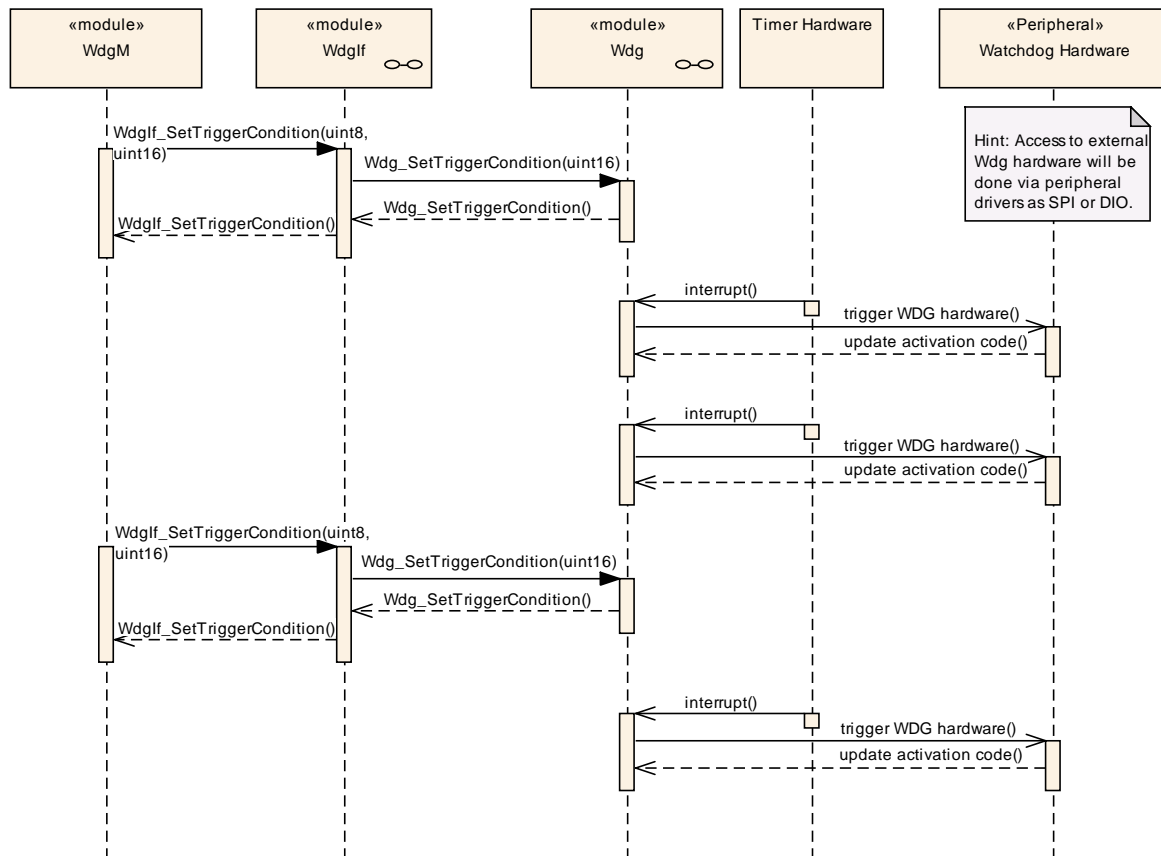
The diagram shows the sequence to initialize the Wdg module, to set the trigger condition and to change the watchdog mode. Note that this is only an example. Especially, another “client” module than the Watchdog Manager (WdgM) could set the trigger condition.



**Figure 2:** 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. Note that this is only an example. For an external watchdog, the watchdog hardware cannot be accessed directly, but only via drivers of the MCAL layer, like SPI or DIO.



**Figure 3:** 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 the chapter 10.1 “Introduction to configuration specification” in *SWS\_BSWGeneral*.

## 10.2 Containers and configuration parameters

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

### 10.2.1 Wdg

|                                   |  |
|-----------------------------------|--|
| <b>SWS Item</b>                   | <b>ECUC_Wdg_00073 :</b>                                    |
| <b>Module Name</b>                | <i>Wdg</i>   |
| <b>Module Description</b>         | Configuration of the Wdg (Watchdog driver) module.         |
| <b>Post-Build Variant Support</b> | true   |
| <b>Supported Config Variants</b>  | VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE |

| <b>Included Containers</b> |                     |  |
|----------------------------|---------------------|--|
| <b>Container Name</b>      | <b>Multiplicity</b> | <b>Scope / Dependency</b>  |
| WdgDemEventParameterRefs   | 0..1                | Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references. |
| 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.<br>Note: All postbuild parameters are handled via this container.   |

### 10.2.2 WdgDemEventParameterRefs

|                                 |  |
|---------------------------------|--|
| <b>SWS Item</b>                 | <b>ECUC_Wdg_00148 :</b>  |
| <b>Container Name</b>           | WdgDemEventParameterRefs   |
| <b>Description</b>              | Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references. |
| <b>Configuration Parameters</b> |  |

|   |  |    |              |
|---|--|----|--------------|
| <b>SWS Item</b>                         | <b>ECUC_Wdg_00150 :</b>  |    |              |
| <b>Name</b>                             | WDG_E_DISABLE_REJECTED   |    |              |
| <b>Description</b>                      | Reference to the DemEventParameter which shall be issued when the error "Initialization or mode switch failed because it would disable the watchdog" has occurred. |    |              |
| <b>Multiplicity</b>                     | 0..1   |    |              |
| <b>Type</b>                             | Symbolic name reference to [ DemEventParameter ]   |    |              |
| <b>Post-Build Variant Multiplicity</b>  | false  |    |              |
| <b>Post-Build Variant Value</b>         | false  |    |              |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|   | <b>Link time</b>   | -- |              |

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

|  |   |    |              |
|--|---|----|--------------|
| <b>SWS Item</b>                        | <b>ECUC_Wdg_00149 :</b>   |    |              |
| <b>Name</b>                            | WDG_E_MODE_FAILED   |    |              |
| <b>Description</b>                     | Reference to the DemEventParameter which shall be issued when the error "Setting a watchdog mode failed (during initialization or mode switch)" has occurred. |    |              |
| <b>Multiplicity</b>                    | 0..1  |    |              |
| <b>Type</b>                            | Symbolic name reference to [ DemEventParameter ]  |    |              |
| <b>Post-Build Variant Multiplicity</b> | false   |    |              |
| <b>Post-Build Variant Value</b>        | false   |    |              |
| <b>Value Configuration Class</b>       | <b>Pre-compile time</b>   | X  | All Variants |
|  | <b>Link time</b>  | -- |              |
|  | <b>Post-build time</b>  | -- |              |
| <b>Value Configuration Class</b>       | <b>Pre-compile time</b>   | X  | All Variants |
|  | <b>Link time</b>  | -- |              |
|  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>              | scope: local  |    |              |

#### No Included Containers

### 10.2.3 WdgGeneral

|                                 |   |  |  |
|---------------------------------|---|--|--|
| <b>SWS Item</b>                 | <b>ECUC_Wdg_00114 :</b>   |  |  |
| <b>Container Name</b>           | WdgGeneral  |  |  |
| <b>Description</b>              | All general parameters of the watchdog driver are collected here. |  |  |
| <b>Configuration Parameters</b> |   |  |  |

|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Wdg_00115 :</b>   |    |              |
| <b>Name</b>                      | WdgDevErrorDetect   |    |              |
| <b>Description</b>               | Switches the development error detection and notification on or off. <ul style="list-style-type: none"> <li>true: detection and notification is enabled.</li> <li>false: detection and notification is disabled.</li> </ul> |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucBooleanParamDef   |    |              |
| <b>Default value</b>             | false   |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                                  | <b>Link time</b>  | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: local  |    |              |

|                     |   |  |  |
|---------------------|---|--|--|
| <b>SWS Item</b>     | <b>ECUC_Wdg_00116 :</b>   |  |  |
| <b>Name</b>         | WdgDisableAllowed   |  |  |
| <b>Description</b>  | Compile switch to allow / forbid disabling the watchdog driver during runtime.<br>True: Disabling the watchdog driver at runtime is allowed.<br>False: Disabling the watchdog driver at runtime is not allowed. |  |  |
| <b>Multiplicity</b> | 1   |  |  |

|                                  |  |    |              |
|----------------------------------|--|----|--------------|
| <b>Type</b>                      | EcucBooleanParamDef  |    |              |
| <b>Default value</b>             | --   |    |              |
| <b>Post-Build Variant Value</b>  | false  |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|                                  | <b>Link time</b>   | -- |              |
|                                  | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: Safety relevant compile switch, this has to be in accordance with the corresponding settings for the watchdog manager. |    |              |

|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Wdg_00117 :</b>   |    |              |
| <b>Name</b>                      | WdgIndex  |    |              |
| <b>Description</b>               | Represents the watchdog driver's ID so that it can be referenced by the watchdog interface. |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucIntegerParamDef (Symbolic Name generated for this parameter)                            |    |              |
| <b>Range</b>                     | 0 .. 255  |    |              |
| <b>Default value</b>             | --  |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                                  | <b>Link time</b>  | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: local  |    |              |

|                                  |  |    |              |
|----------------------------------|--|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Wdg_00130 :</b>  |    |              |
| <b>Name</b>                      | WdgInitialTimeout  |    |              |
| <b>Description</b>               | The initial timeout (sec) for the trigger condition to be initialized during Init function. It shall be not larger than WdgMaxTimeout. |    |              |
| <b>Multiplicity</b>              | 1  |    |              |
| <b>Type</b>                      | EcucFloatParamDef  |    |              |
| <b>Range</b>                     | [0 .. 65.535]  |    |              |
| <b>Default value</b>             | --   |    |              |
| <b>Post-Build Variant Value</b>  | false  |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>  | X  | All Variants |
|                                  | <b>Link time</b>   | -- |              |
|                                  | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b>        | scope: local   |    |              |

|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Wdg_00131 :</b>   |    |              |
| <b>Name</b>                      | WdgMaxTimeout   |    |              |
| <b>Description</b>               | The maximum timeout (sec) to which the watchdog trigger condition can be initialized. |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucFloatParamDef   |    |              |
| <b>Range</b>                     | [0 .. 65.535]   |    |              |
| <b>Default value</b>             | --  |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                                  | <b>Link time</b>  | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: local  |    |              |

|                    |   |  |  |
|--------------------|---|--|--|
| <b>SWS Item</b>    | <b>ECUC_Wdg_00147 :</b>   |  |  |
| <b>Name</b>        | WdgRunArea  |  |  |
| <b>Description</b> | Represents the watchdog driver execution area is either from ROM(Flash) or RAM as required with the particular microcontroller. |  |  |

|                                  |                         |  |              |
|----------------------------------|-------------------------|--|--------------|
| <b>Multiplicity</b>              | 1                       |  |              |
| <b>Type</b>                      | EcucEnumerationParamDef |  |              |
| <b>Range</b>                     | RAM                     | Watchdog driver to be executed out of RAM area |              |
|                                  | ROM                     | Watchdog driver to be executed out of ROM area |              |
| <b>Post-Build Variant Value</b>  | false                   |  |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b> | X  | All Variants |
|                                  | <b>Link time</b>        | --   |              |
|                                  | <b>Post-build time</b>  | --   |              |
| <b>Scope / Dependency</b>        | scope: local            |  |              |

|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Wdg_00118 :</b>   |    |              |
| <b>Name</b>                      | WdgTriggerLocation  |    |              |
| <b>Description</b>               | Location (memory address) of the watchdog trigger routine.                                  |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucFunctionNameDef   |    |              |
| <b>Default value</b>             | --  |    |              |
| <b>maxLength</b>                 | --  |    |              |
| <b>minLength</b>                 | --  |    |              |
| <b>regularExpression</b>         | --  |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                                  | <b>Link time</b>  | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: Only relevant if provided by hardware and needed by the system. |    |              |

|                                  |   |    |              |
|----------------------------------|---|----|--------------|
| <b>SWS Item</b>                  | <b>ECUC_Wdg_00119 :</b>   |    |              |
| <b>Name</b>                      | WdgVersionInfoApi   |    |              |
| <b>Description</b>               | Compile switch to enable / disable the version information API <ul style="list-style-type: none"> <li>• True: API enabled</li> <li>• False: API disabled</li> </ul> |    |              |
| <b>Multiplicity</b>              | 1   |    |              |
| <b>Type</b>                      | EcucBooleanParamDef   |    |              |
| <b>Default value</b>             | false   |    |              |
| <b>Post-Build Variant Value</b>  | false   |    |              |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                                  | <b>Link time</b>  | -- |              |
|                                  | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b>        | scope: local  |    |              |

#### No Included Containers

### 10.2.4 WdgSettingsConfig

|                       |  |  |  |
|-----------------------|--|--|--|
| <b>SWS Item</b>       | <b>ECUC_Wdg_00082 :</b>  |  |  |
| <b>Container Name</b> | WdgSettingsConfig  |  |  |
| <b>Description</b>    | Configuration items for the different watchdog settings, including those for external watchdog hardware. |  |  |



|                                 |  |
|---------------------------------|--|
|                                 | Note: All postbuild parameters are handled via this container. |
| <b>Configuration Parameters</b> |  |

|                                  |   |                                 |                     |
|----------------------------------|---|---------------------------------|---------------------|
| <b>SWS Item</b>                  | <b>ECUC_Wdg_00120 :</b>   |                                 |                     |
| <b>Name</b>                      | WdgDefaultMode  |                                 |                     |
| <b>Description</b>               | Default mode for watchdog driver initialization.<br>ImplementationType: WdgIf_ModeType            |                                 |                     |
| <b>Multiplicity</b>              | 1   |                                 |                     |
| <b>Type</b>                      | EcucEnumerationParamDef   |                                 |                     |
| <b>Range</b>                     | WDGIF_FAST_MODE   | Default watchdog mode is "fast" |                     |
|                                  | WDGIF_OFF_MODE  | Default watchdog mode is "off"  |                     |
|                                  | WDGIF_SLOW_MODE   | Default watchdog mode is "slow" |                     |
| <b>Post-Build Variant Value</b>  | true  |                                 |                     |
| <b>Value Configuration Class</b> | <b>Pre-compile time</b>   | X                               | VARIANT-PRE-COMPILE |
|                                  | <b>Link time</b>  | X                               | VARIANT-LINK-TIME   |
|                                  | <b>Post-build time</b>  | X                               | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>        | scope: local<br>dependency: "Off" mode only possible if disabling the watchdog driver is allowed. |                                 |                     |

|                            |                     |  |
|----------------------------|---------------------|--|
| <b>Included Containers</b> |                     |  |
| <b>Container Name</b>      | <b>Multiplicity</b> | <b>Scope / Dependency</b>  |
| 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. |

Note:

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.

### 10.2.5 WdgSettingsFast

|                                 |  |
|---------------------------------|--|
| <b>SWS Item</b>                 | <b>ECUC_Wdg_00121 :</b>  |
| <b>Container Name</b>           | WdgSettingsFast  |
| <b>Description</b>              | Hardware dependent settings for the watchdog driver's "fast" mode. |
| <b>Configuration Parameters</b> |  |

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

### 10.2.6 WdgSettingsSlow

|                                 |  |
|---------------------------------|--|
| <b>SWS Item</b>                 | <b>ECUC_Wdg_00123 :</b>  |
| <b>Container Name</b>           | WdgSettingsSlow  |
| <b>Description</b>              | Hardware dependent settings for the watchdog driver's "slow" mode. |
| <b>Configuration Parameters</b> |  |

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

### 10.2.7 WdgSettingsOff

|                                 |   |
|---------------------------------|---|
| <b>SWS Item</b>                 | <b>ECUC_Wdg_00122 :</b>   |
| <b>Container Name</b>           | WdgSettingsOff  |
| <b>Description</b>              | Hardware dependent settings for the watchdog driver's "off" mode. |
| <b>Configuration Parameters</b> |   |

**No Included Containers**

### 10.2.8 WdgExternalConfiguration

|                                 |   |
|---------------------------------|---|
| <b>SWS Item</b>                 | <b>ECUC_Wdg_00112 :</b>                               |
| <b>Container Name</b>           | WdgExternalConfiguration                              |
| <b>Description</b>              | Configuration items for an external watchdog hardware |
| <b>Configuration Parameters</b> |   |

|   |   |   |                     |
|---|---|---|---------------------|
| <b>SWS Item</b>                         | <b>ECUC_Wdg_00113 :</b>   |   |                     |
| <b>Name</b>                             | WdgExternalContainerRef   |   |                     |
| <b>Description</b>                      | Reference to either<br>- a DioChannelGroup container in case the hardware watchdog is connected via DIO pins<br>- an SpiSequenceConfiguration container in case the watchdog hardware is accessed via SPI |   |                     |
| <b>Multiplicity</b>                     | 0..1  |   |                     |
| <b>Type</b>                             | Choice reference to [ DioChannelGroup , SpiSequence ]   |   |                     |
| <b>Post-Build Variant Multiplicity</b>  | true  |   |                     |
| <b>Post-Build Variant Value</b>         | true  |   |                     |
| <b>Multiplicity Configuration Class</b> | <b>Pre-compile time</b>   | X | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>  | X | VARIANT-LINK-TIME   |
|   | <b>Post-build time</b>  | X | VARIANT-POST-BUILD  |
| <b>Value Configuration Class</b>        | <b>Pre-compile time</b>   | X | VARIANT-PRE-COMPILE |
|   | <b>Link time</b>  | X | VARIANT-LINK-TIME   |
|   | <b>Post-build time</b>  | X | VARIANT-POST-BUILD  |
| <b>Scope / Dependency</b>               | scope: local<br>dependency: See DIO resp. SPI SWS   |   |                     |

**No Included Containers**

## 10.3 Published information

For details refer to the chapter 10.3 “Published Information” in *SWS\_BSWGeneral*.

### 10.3.1 WdgPublishedInformation

|                                 |   |
|---------------------------------|---|
| <b>SWS Item</b>                 | <b>ECUC_Wdg_00074 :</b>   |
| <b>Container Name</b>           | WdgPublishedInformation   |
| <b>Description</b>              | Container holding all Wdg specific published information parameters |
| <b>Configuration Parameters</b> |   |

|                 |                         |
|-----------------|-------------------------|
| <b>SWS Item</b> | <b>ECUC_Wdg_00127 :</b> |
| <b>Name</b>     | WdgTriggerMode          |

|                                  |  |    |              |
|----------------------------------|--|----|--------------|
| <b>Description</b>               | Watchdog trigger mode (toggle/window/both) |    |              |
| <b>Multiplicity</b>              | 1  |    |              |
| <b>Type</b>                      | EcucEnumerationParamDef                    |    |              |
| <b>Range</b>                     | WDG_BOTH                                   | -- |              |
|                                  | WDG_TOGGLE                                 | -- |              |
|                                  | WDG_WINDOW                                 | -- |              |
| <b>Post-Build Variant Value</b>  | false                                      |    |              |
| <b>Value Configuration Class</b> | <b>Published Information</b>               | X  | All Variants |
| <b>Scope / Dependency</b>        | scope: local                               |    |              |

**No Included Containers**

**Note:**

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

## 11 Not applicable requirements

**[SWS\_Wdg\_00175]** [ These requirements are not applicable to this specification.]

(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\_BSW\_00161, SRS\_BSW\_00162, SRS\_BSW\_00005, SRS\_BSW\_00415, SRS\_BSW\_00007, SRS\_BSW\_00413, SRS\_BSW\_00441, SRS\_BSW\_00307, SRS\_BSW\_00373, SRS\_BSW\_00410, SRS\_BSW\_00447, SRS\_BSW\_00348, SRS\_BSW\_00353, SRS\_BSW\_00361, SRS\_BSW\_00302, SRS\_BSW\_00328, SRS\_BSW\_00312, SRS\_BSW\_00006, SRS\_BSW\_00449, SRS\_BSW\_00377, SRS\_BSW\_00304, SRS\_BSW\_00378, SRS\_BSW\_00306, SRS\_BSW\_00308, SRS\_BSW\_00309, SRS\_BSW\_00371, SRS\_BSW\_00359, SRS\_BSW\_00360, SRS\_BSW\_00440, SRS\_BSW\_00330, SRS\_BSW\_00009, SRS\_BSW\_00401, SRS\_BSW\_00172, SRS\_BSW\_00010, SRS\_BSW\_00333, SRS\_BSW\_00321, SRS\_BSW\_00341, SRS\_BSW\_00334, 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)