Document Title	Specification of Watchdog Driver
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	2.1.0
Document Status	Draft
Part of Release	2.1
Revision	0014

Document Change History

Date	Version	Changed by	Change Description
31.01.2007	2.1.0	AUTOSAR Administration	• In chapter 5.1.2 the file include structure has been changed to comply with the SPAL general include structure. • In chapter 10.2.2 WDG_DEFAULT_MODE has been added as PC variant and WDG003 has been changed to allow passing NULL pointer. • For WDG037 the requirement was changed to allow configuration of activation code if the H/W allows for the same. • For WDG078 the requirement was changed to add reference to SPI/DIO for accessing the external watchdog • Legal disclaimer revised • Release Notes added • “Advice for users” revised • “Revision Information” added
20.03.2006	2.0.0	AUTOSAR Administration	Document structure adapted to common Release 2.0 SWS Template
31.05.2005	1.0.0	AUTOSAR Administration	Initial Release

Release Notes

Errata and known deficiencies

The current concept of how to trigger the watchdog hardware does not represent the startup, wakeup, sleep and shutdown phases properly.

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents 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 Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, 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

This document specifies the functionality, API and the configuration of the AUTOSAR Basic Software module Watchdog Driver.

This module provides services for initialization, changing the operation mode and triggering the watchdog.

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

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:
WDG	Watchdog (module specific prefix)
DET	Development Error Tracer – module to catch development errors.
DEM	Diagnostic Event Manager – module to handle diagnostic relevant events.

Definitions needed for understanding of the concepts

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

3 Related documentation

3.1 Input documents

[1] Layered Software Architecture
https://svn.autosar.org/repos/10Releases/
AUTOSAR_LayeredSoftwareArchitecture.pdf

[2] General Requirements on Basic Software Modules
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_General.pdf

[3] General Requirements on SPAL
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_SPAL_General.pdf

[4] Requirements on Watchdog Driver
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_WatchdogDriver.pdf

[5] Specification of Watchdog Interface,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_WatchdogInterface.pdf

3.2 Related standards and norms

None

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

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

A driver for an external watchdog uses handlers or drivers of other modules (e.g. SPI) to access the external watchdog device. Such a driver is located in the ECU Abstraction Layer.

WDG055: The source code of an external watchdog driver shall be independent of the microcontroller platform.

5.1 File structure

5.1.1 Code file structure

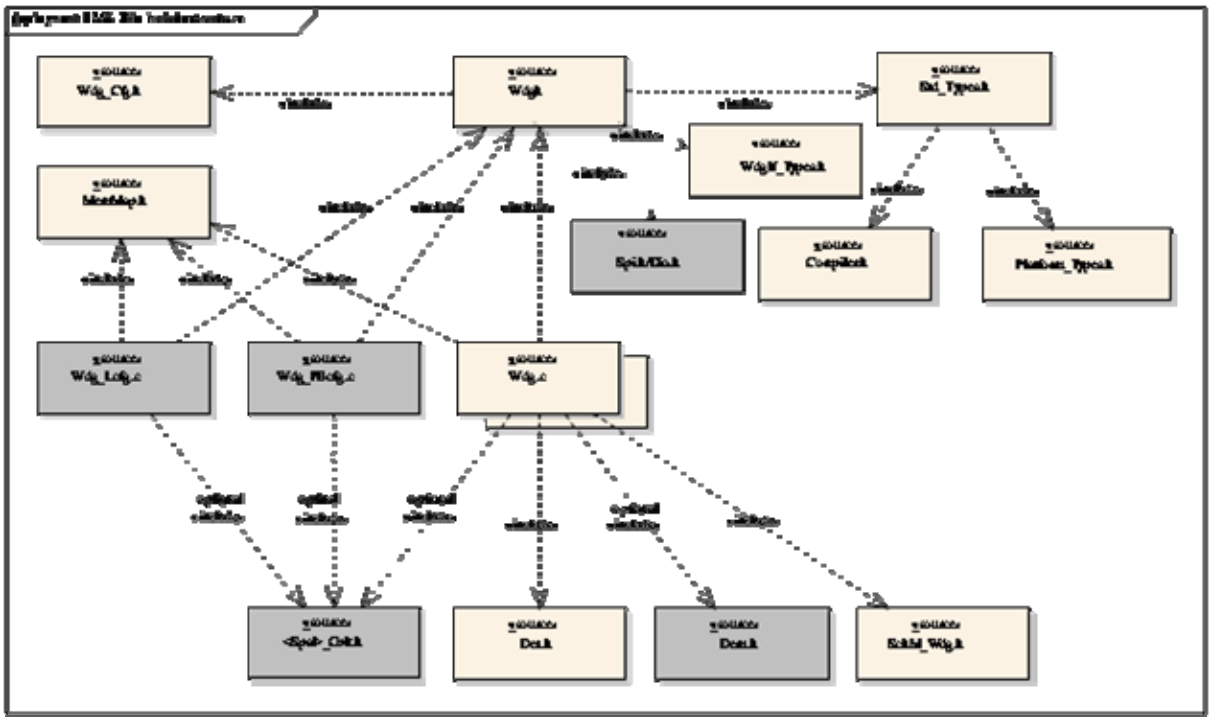
WDG079: The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files named:

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

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

5.1.2 Header file structure

WDG061: The file include structure shall be as follows:



The grey boxes are optional:

Figure 1: File include structure

WDG080: The module shall include the Dem.h file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

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 driver 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

Document: General Requirements on Basic Software Modules

Requirement	***Satisfied by***
[BSW00344] Reference to link-time configuration	WDG082
[BSW00404] Reference to post build time configuration	WDG001, WDG004, WDG082
[BSW00405] Reference to multiple configuration sets	WDG001, WDG004
[BSW00345] Pre-compile-time configuration	WDG045, WDG073, WDG082
[BSW159] Tool-based configuration	Chapter 10.2
[BSW167] Static configuration checking	WDG027
[BSW171] Configurability of optional functionality	WDG069, WDG070, WDG071, WDG081
[BSW170] Data for reconfiguration of SW-components	Not applicable (this module does not depend on faults, signals, …)
[BSW00380] Separate C-File for configuration parameters	WDG079
[BSW00419] Separate C-Files for pre-compile time configuration parameters	Not applicable (only #define's as pre-compile time parameters)
BSW00381] Separate configuration header file for pre-compile time parameters	WDG061
[BSW00412] Separate H-File for configuration parameters	WDG061
[BSW00382] Not-used configuration elements need to be listed	Not applicable (there are no not-used configuration elements for this module)
[BSW00383] List dependencies of configuration files	Not applicable (this module does not use configuration files from other modules)
[BSW00384] List dependencies to other modules	Chapter 5
[BSW00387] Specify the configuration class of callback function	Not applicable (this module does not provide any callback functions)
[BSW00388] Introduce containers	Chapter 10.2
[BSW00389] Containers shall have names	Chapter 10.2.2
[BSW00390] Parameter content shall be unique within the module	Chapter 10.2.2
[BSW00391] Parameter shall have unique names	Chapter 10.2.2
[BSW00392] Parameters shall have a type	Chapter 10.2.2
[BSW00393] Parameters shall have a range	Chapter 10.2.2
[BSW00394] Specify the scope of the parameters	Chapter 10.2.2
[BSW00395] List the required parameters (per parameter)	Chapter 10.2.2
[BSW00396] Configuration classes	Chapter 10.2.2
[BSW00397] Pre-compile-time parameters	Chapter 10.2.2
[BSW00398] Link-time parameters	Chapter 10.2.1, WDG082
[BSW00399] Loadable Post-build time parameters	Chapter 10.2.1, WDG082, WDG083
[BSW00400] Selectable Post-build time parameters	WDG001, WDG004, WDG082
[BSW00402] Published information	Chapter 10.2.4
[BSW00375] Notification of wake-up reason	Not applicable (this module does not provide any wake-up reason)
[BSW101] Initialization interface	WDG001
[BSW00416] Sequence of Initialization	Not applicable

	(requirement on system design, not on a single module)
[BSW00406] Check module initialization	WDG019
[BSW168] Diagnostic Interface of SW components	Not applicable (this module does not support a special diagnostic interface)
[BSW00407] Function to read out published parameters	Chapter 8.3.4
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (this module does not provide an AUTOSAR interface)
[BSW00424] BSW main processing function task allocation	Not applicable (this module does not provide a schedulable main function)
[BSW00425] Trigger conditions for schedulable objects	Not applicable (this module does not provide any schedulable objects)
[BSW00426] Exclusive areas in BSW modules	Not applicable (no exclusive areas specified for this module)
[BSW00427] ISR description for BSW modules	Not applicable (this module does not provide any ISRs)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (this module does not provide a schedulable main function)
[BSW00429] Restricted BSW OS functionality access	Not applicable (this module doesn't use any OS objects or services)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (requirement on the BSW scheduler module)
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (this module does not provide a schedulable main function)
[BSW00433] Calling of main processing functions	Not applicable (requirement on system design, not a single module)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (this is not the schedule module)
[BSW00336] Shutdown interface	WDG031
[BSW00337] Classification of errors	WDG010, WDG013
[BSW00338] Detection and Reporting of development errors	WDG003, WDG008, WDG017, WDG018, WDG019, WDG052, WDG025, WDG026, WDG035
[BSW00369] Do not return development error codes via API	WDG066, WDG012
[BSW00339] Reporting of production relevant error status	Not applicable (no production relevant error status, only error events)
[BSW00421] Reporting of production relevant error events	WDG012
[BSW00422] Debouncing of production relevant error status	Not applicable (requirement on the DEM, not a general requirement)
[BSW00420] Production relevant error event rate detection	Not applicable (requirement on the DEM, not a general requirement)
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable (this is a basic software module)

[BSW00323] API parameter checking	WDG003, WDG008, WDG025, WDG026
[BSW004] Version check	WDG027
[BSW00409] Header files for production code error IDs	WDG062
[BSW00385] List possible error notificatons	WDG010, WDG013
[BSW00386] Configuration for detecting an error	WDG045, WDG064, WDG065
[BSW161] Microcontroller abstraction	Not applicable (requirement on AUTOSAR architecture, not a single module)
[BSW162] ECU layout abstraction	Not applicable (requirement on AUTOSAR architecture, not a single module)
[BSW00324] Do not use HIS I/O Library	Not applicable (architecture decision)
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (requirement on AUTOSAR architecture, not a single module)
[BSW00415] User dependent include files	Not applicable (only one user for this module)
[BSW164] Implementation of interrupt service routines	Not applicable (this module does not implement any ISRs)
[BSW00325] Runtime of interrupt service routines	Not applicable (this module does not implement any ISRs)
[BSW00326] Transition from ISRs to OS tasks	Not applicable (this module does not implement any ISRs)
[BSW00342] Usage of source code and object code	Not applicable (requirement on AUTOSAR architecture, not a single module)
[BSW00343] Specification and configuration of time	Not applicable (no configurable timings)
[BSW160] Human-readable configuration data	Not applicable (requirement on documentation, not on specification)
[BSW007] HIS MISRA C	Not applicable (requirement on implementation, not on specification)
[BSW00300] Module naming convention	Not applicable (requirement on implementation, not on specification)
[BSW00413] Accessing instances of BSW modules	Not implementable in R2.0 timeframe.
[BSW00347] Naming separation of different instances of BSW drivers	Not applicable (requirement on the implementation, not on the specification)
[BSW00305] Self-defined data types naming convention	Chapter 8.2.1
[BSW00307] Global variables naming convention	Not applicable (requirement on the implementation, not on the specification)
[BSW00310] API naming convention	Chapters 8.3.1, 8.3.2, 8.3.3
[BSW00373] Main processing function naming convention	Not applicable (no main processing function)
[BSW00327] Error values naming convention	WDG010, WDG013
[BSW00335] Status values naming convention	Not applicable (status value not seen outside of this module)

[BSW00350] Development error detection keyword	WDG045, WDG069
[BSW00408] Configuration parameter naming convention	Chapter 10.2.2, Chapter 10.2.3
[BSW00410] Compiler switches shall have defined values	Chapter 10.2.2
[BSW00411] Get version info keyword	Chapter 10.2.2
[BSW00346] Basic set of module files	WDG061
[BSW158] Separation of configuration from implementation	WDG061
[BSW00314] Separation of interrupt frames and service routines	Not applicable (this module does not implement any ISRs)
[BSW00370] Separation of callback interface from API	Not applicable (this module does not provide any callback routines)
[BSW00348] Standard type header	Not applicable (standard header files included via interface header file)
[BSW00353] Platform specific type header	Not applicable (standard header files included via interface header file)
[BSW00361] Compiler specific language extension header	Not applicable (standard header files included via interface header file)
[BSW00301] Limit imported information	WDG061
[BSW00302] Limit exported information	Not applicable (requirement on the implementation, not on the specification)
[BSW00328] Avoid duplication of code	Not applicable (requirement on the implementation, not on the specification)
[BSW00312] Shared code shall be reentrant	Not applicable (requirement on the implementation, not on the specification)
[BSW006] Platform independency	Not applicable (this is a module of the microcontroller abstraction layer)
[BSW00357] Standard API return type	Chapter 8.1.1, Chapter 8.3.2
[BSW00377] Module specific API return types	Not applicable (no module specific return types)
[BSW00304] AUTOSAR integer data types	Not applicable (requirement on implementation, not for specification)
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable (requirement on implementation, not for specification)
[BSW00378] AUTOSAR boolean type	Not applicable (requirement on implementation, not for specification)
[BSW00306] Avoid direct use of compiler and platform specific keywords	Not applicable (requirement on implementation, not for specification)
[BSW00308] Definition of global data	Not applicable (requirement on implementation, not for specification)
[BSW00309] Global data with read-only constraint	Not applicable (requirement on implementation, not for specification)
[BSW00371] Do not pass function pointers via API	Not applicable

	(no function pointers in this specification)
[BSW00358] Return type of init() functions	Chapter 8.3.1
[BSW00376] Return type and parameters of main processing functions	Not applicable (this module does not provide a main processing function)
[BSW00359] Return type of callback functions	Not applicable (this module does not provide any callback routines)
[BSW00360] Parameters of callback functions	Not applicable (this module does not provide any callback routines)
[BSW00329] Avoidance of generic interfaces	Chapters 8.3.1, 8.3.2, 8.3.3 (explicit interfaces defined)
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable (requirement on implementation, not for specification)
[BSW00331] Separation of error and status values	WDG010, WDG013
[BSW009] Module User Documentation	Not applicable (requirement on documentation, not on specification)
[BSW00401] Documentation of multiple instances of configuration parameters	Not applicable (all configuration parameters are single instance only)
[BSW172] Compatibility and documentation of scheduling strategy	Not applicable (no internal scheduling policy)
[BSW010] Memory resource documentation	Not applicable (requirement on documentation, not on specification)
[BSW00333] Documentation of callback function context	Not applicable (this module does not provide any callback routines)
[BSW00374] Module vendor identification	WDG074
[BSW00379] Module identification	WDG074
[BSW003] Version identification	WDG074
[BSW00318] Format of module version numbers	WDG074
[BSW00321] Enumeration of module version numbers	Not applicable (requirement on implementation, not for specification)
[BSW00341] Microcontroller compatibility documentation	Not applicable (requirement on documentation, not on specification)
[BSW00334] Provision of XML file	Not applicable (requirement on documentation, not on specification)
[BSW00435] Module Header File Structure for the Basic Software Scheduler	Chapter 5.1.2
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	Chapter 5.1.2

Document: General Requirements on SPAL

Requirement	***Satisfied by***
[BSW12263] Object code compatible configuration concept	WDG004, WDG073, WDG082
[BSW12056] Configuration of notification mechanisms	Not applicable (this module does not support any notification mechanism)

[BSW12267] Configuration of wake-up sources	Not applicable (this module does not wake up the ECU / MCU)
[BSW12057] Driver module initialization	WDG028
[BSW12125] Initialization of hardware resources	WDG028
[BSW12163] Driver module de-initialization	WDG025, WDG026, WDG031
[BSW12058] Individual initialization of overall registers	WDG028
[BSW12059] General initialization of overall registers	WDG028
[BSW12060] Responsibility for initialization of one-time writable registers	WDG028
[BSW12461] Responsibility for register initialization	WDG028
[BSW12462] Provide settings for register initialization	Not applicable (requirement on implementation, not on specification)
[BSW12463] Combine and forward settings for register initialization	Not applicable (requirement on configuration, not on specification)
[BSW12062] Selection of static configuration sets	WDG001
[BSW12068] MCAL initialization sequence	Not applicable (requirement for system integration, not for a single module)
[BSW12069] Wake-up notification of ECU State Manager	Not applicable (this module does not wake up the ECU / MCU)
[BSW157] Notification mechanisms of drivers and handlers	Not applicable (this module does not support any notification mechanism)
[BSW12155] Prototypes of callback functions	Not applicable (this module does not provide any callback functions)
[BSW12169] Control of operation mode	WDG004
[BSW12063] Raw value mode	Not applicable (this module does not provide any data to the user)
[BSW12075] Use of application buffers	Not applicable (this module does not operate on buffers)
[BSW12129] Resetting of interrupt flags	Not applicable (this module does not implement any ISRs)
[BSW12064] Change of operation mode during running operation	WDG032, WDG016, WDG017
[BSW12448] Behavior after development error detection	WDG003, WDG025, WDG008, WDG017, WDG026
[BSW12067] Setting of wake-up conditions	Not applicable (this module does not wake up the ECU / MCU)
[BSW12077] Non-blocking implementation	Not applicable (no long term loops)
[BSW12078] Runtime and memory efficiency	Not applicable (requirement for implementation, not for specification)
[BSW12092] Access to drivers	WDG076
[BSW12265] Configuration data shall be kept constant	WDG001
[BSW12264] Specification of configuration items	WDG073

Document: Requirements on Watchdog Driver

Requirement	***Satisfied by***
[BSW12015] Configuration of watchdog modes	WDG004, WDG051
[BSW12105] Watchdog initialization	WDG001, WDG028
[BSW12106] Prohibit disabling of watchdog	WDG025, WDG026
[BSW12018] Watchdog mode selection service	WDG004, WDG032
[BSW12019] Watchdog trigger service	WDG036, WDG037
[BSW12165] Functional scope	WDG077
[BSW12166] SPI channel configuration	WDG078
[BSW12167] Common Watchdog API	Not applicable (only interface to watchdog drivers)
[BSW12168] Microcontroller independency	Not applicable (requirement for implementation, not for specification)

7 Functional specification

7.1 General design rules

WDG027: The configuration parameters shall be checked statically (at the latest during compile time) for correctness. The version information in the module header and source files shall be validated and consistent (e.g. by comparing the version information in the module header and source files with a pre-processor macro).

WDG031: The watchdog driver shall not implement an interface for de-initialization / shutdown since some watchdogs do not support this functionality and in some environments this feature must not be used (e.g. in safety critical systems). If the watchdog supports a de-initialization / shutdown and the environment allows the usage of this feature it shall be achieved by calling the Wdg_SetMode routine with the corresponding mode parameter.

WDG034: The start address of the watchdog trigger routine shall be statically configurable to a fixed memory location. This allows the watchdog device to identify the correct trigger input if supported by the hardware. If the watchdog hardware requires a certain location for the trigger routine, it shall be given in the watchdog driver's documentation and has to be matched by the driver's configuration. This configuration parameter shall only be given if supported / needed by the hardware.

WDG040: 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 operating system functionality if possible.

7.2 Error classification

WDG062: Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file Dem_IntErrId.h and included via Dem.h.

WDG063: Development error values are of type uint8.

WDG010: The watchdog driver shall be able to detect the following errors and exceptions depending on its configuration (development/production mode):

Type or error	*Relevance*	*Related error code*	*Value [hex]*
API service used in wrong context (e.g. driver not initialized).	Development	`WDG_E_DRIVER_STATE`	0x10
API service called with wrong / inconsistent parameter(s)	Development	`WDG_E_PARAM_MODE` `WDG_E_PARAM_CONFIG`	0x11 0x12
Switching between watchdog modes failed.	Production	`WDG_E_MODE_SWITCH_FAILED`	Assigned by DEM
Disabling of watchdog not allowed (e.g. in safety relevant systems)	Production	`WDG_E_DISABLE_REJECTED`	Assigned by DEM

7.3 Error detection

WDG045: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch *WDG_DEV_ERROR_DETECT* (see chapter 10) shall activate or deactivate the detection of all development errors.

WDG064: If the *WDG_DEV_ERROR_DETECT* switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.2 and chapter 8.

WDG065: The detection of production code errors cannot be switched off.

7.4 Error notification

WDG066: Detected evelopment errors shall be reported to the Development Error Tracer (DET) if the pre-processor switch WDG_DEV_ERROR_DETECT is set. The error codes shall not be used as return values of the called function.

WDG012: Detected production relevant error events shall be reported to the Diagnostic Event Manager (DEM). The error codes shall not be used as return values of the called function.

WDG013: Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the module's implementation documentation. The classification and enumeration shall be compatible to the errors listed above [WDG010]

7.5 External watchdog driver

WDG076: To access the external watchdog hardware, the watchdog driver has to use the functionality and API of the corresponding handler or driver, e.g. the SPI handler or DIO driver.

WDG077: An external watchdog driver is based on the same functional requirements and offers the same functional scope as an internal watchdog driver hence their respective APIs are semantically identical.

WDG078: All parameters required for this access, e.g. the used SPI channel or DIO port, have to be added to the list of the drivers published parameters and also to the watchdog drivers configuration set.

8 API specification

8.1 Imported types

8.1.1 Standard types

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

- Std_Types.h

- Std_ReturnType
- Std_VersionInfoType

8.1.2 WdgIf types

In this chapter all types included from module WdgIf are listed.

- WdgIf_ModeType
- WdgIf_StatusType

8.2 Type definitions

8.2.1 Wdg_ConfigType

Wdg_ConfigType

Type:	struct
Range:	`Hardware dependent structure` Structure to hold the watchdog driver configuration set.
Description:	A pointer to such a structure is provided to the watchdog driver initialization routine for configuration of the driver and watchdog hardware.

8.3 Function definitions

8.3.1 Wdg_Init

Wdg_Init

Service name:	Wdg_Init
Syntax:	void Wdg_Init (const Wdg_ConfigType *ConfigPtr)
Service ID [hex]:	0x00
Sync/Async:	Synchronous
Re-entrancy:	Non re-entrant

Parameters (in):	ConfigPtr	Pointer to configuration set.
Parameters (out):	None	--
Return value	None	--
Description:	**WDG001:** This routine initializes the watchdog driver and watchdog hardware, i.e. it sets the default watchdog mode and timeout period as provided in the configuration set. This configuration set shall be chosen from a limited number of statically configured sets. **WDG028:** The routine shall initialize all module global variables and 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 shall be initialized by a common system module. **WDG009:** All timeout periods shall have the same (configured) duration, i.e. the first timeout period shall have the same duration as the following timeout periods. **WDG003:** If development error detection is enabled, the parameter <ConfigPtr> shall be checked for not being a NULL pointer except for Pre-Compiled variant. Also the (hardware specific) contents of the given configuration set shall be checked for being within the allowed boundaries. If an error is detected the initialization of the watchdog shall not be executed and the error shall be reported to the Development Error Tracer with the value WDG_E_PARAM_CONFIG. **WDG025:** If disabling the watchdog is not allowed (e.g. in safety relevant systems, see (WDG070) the routine shall check if the default mode given in the provided configuration set will disable the watchdog. In this case the initialization routine shall not be executed and the error shall be reported to the Diagnostic Event Manager (DEM) with the value WDG_E_DISABLE_REJECTED. **WDG019:** If development error detection is enabled, the routine shall set the drivers internal state from WDG_UNINIT (the default state) to WDG_IDLE if the initialization was successful.	
Caveats:	The watchdog hardware might already be running out of power-on-reset.	
Configuration:	None	

8.3.2 Wdg_SetMode

Wdg_SetMode

Service name:	Wdg_SetMode	
Syntax:	Std_ReturnType Wdg_SetMode (WdgIf_ModeType Mode)	
Service ID [hex]:	0x01	
Sync/Async:	Synchronous	
Re-entrancy:	Non re-entrant	
Parameters (in):	WdgIf_ModeType	One of the following statically configured modes: • WDGIF_OFF_MODE • WDGIF_SLOW_MODE • WDGIF_FAST_MODE
Parameters (out):	None	--
Return value:		Std_ReturnType , .

	E_OK	if the function succeeded i.e. the mode switch has been completely executed
	E_NOT_OK	otherwise
Description:	**WDG004:** By choosing one of a limited number of statically configured settings (e.g. toggle or window watchdog, different timeout periods) the watchdog driver and hardware can be switched between the following three different watchdog modes: • WDGIF_OFF_MODE • WDGIF_SLOW_MODE • WDGIF_FAST_MODE **WDG051:** The parameters needed for the different watchdog modes are hardware / driver specific and contained in the watchdog driver's configuration set provided to the initialization routine. **WDG032:** The routine shall switch the watchdog driver and hardware from the current watchdog mode to the watchdog mode defined by the given parameter. It shall return E_OK if this mode switch has been executed completely and successfully, i.e. all parameters of the watchdog driver and hardware have been set to the new values. **WDG016:** If switching the watchdog driver and 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 routine shall return the value E_NOT_OK. This error shall be reported to the Diagnostic Event Manager (DEM) with the error code WDG_E_MODE_SWITCH_FAILED. **WDG008:** If development error detection is enabled, the parameter <Mode> shall be checked for being within the allowed range. Also the (hardware specific) settings for the requested mode shall be checked for being within the allowed boundaries. In case of an error the mode switch shall not be executed, the error shall be reported to the Development Error Tracer (DET) with the value WDG_E_PARAM_MODE and the routine shall return the value E_NOT_OK. **WDG017:** If development error detection is enabled, the routine shall check whether the driver state is WDG_IDLE (meaning the watchdog driver and hardware are initialized and the watchdog is currently not being triggered or switched). In case of an error the mode switch shall not be executed, the error shall be reported to the Development Error Tracer (DET) with the error code WDG_E_DRIVER_STATE and the routine shall return the value E_NOT_OK. **WDG018:** If development error detection is enabled, the routine shall set the driver state to WDG_BUSY during its runtime. It shall reset the driver state to WDG_IDLE last thing before it returns to the caller. **WDG026:** If disabling the watchdog is not allowed (e.g. in safety relevant systems, see (WDG070) the routine shall check if the settings for the requested mode would disable the watchdog. In this case, the mode switch shall not be executed, the error shall be reported to the Diagnostic Event Manager (DEM) with the error code WDG_E_DISABLE_REJECTED and the routine shall return the value E_NOT_OK.	
Caveats:	None	
Configuration:	None	

8.3.3 Wdg_Trigger

Wdg_Trigger

Service name:	Wdg_Trigger
Syntax:	void Wdg_Trigger (void)
Service ID [hex]:	0x02
Sync/Async:	Synchronous
Re-entrancy:	Non re-entrant
Parameters (in):	None --
Parameters (out):	None --
Return value	None --
Description:	**WDG036:** This routine triggers the watchdog hardware. It has to be called cyclically by some upper layer function (ususally the watchdog manager) in order to prevent the watchdog hardware from expiring. **WDG037:** If the watchdog hardware requires an activation code which can be configured or changed, this shall be handled internally by the watchdog driver. In this case, the driver shall pass the correct activation code to the watchdog hardware and the watchdog hardware in turn shall update the driver internal variable where the next expected access code is stored. The trigger cycle of the watchdog driver shall be thus that updating the activation code by the watchdog hardware can be guaranteed (see Figure 3). If the initial activation code can be configured, it shall be given in the watchdog drivers configuration set.. If the activation code is fixed for a particular hardware the above requirement can be ignored. **WDG035:** If development error detection is enabled, the routine shall check whether the driver state is WDG_IDLE (meaning the watchdog driver and hardware are initialized and the watchdog is currently not being triggered or switched). In case of an error the routine shall not be executed and the error shall be reported to the Development Error Tracer with the value WDG_E_DRIVER_STATE. **WDG052:** If development error detection is enabled, the routine shall set the driver state to WDG_BUSY during its runtime. It shall reset the driver state to WDG_IDLE last thing before it returns to the caller. **WDG041:** This routine shall be kept as short as possible since it might be called on interrupt level.
Caveats:	- The watchdog driver shall have been initialized before this service is called - This routine might be called on interrupt level.
Configuration:	None

8.3.4 Wdg_GetVersionInfo

Service name:	Wdg_ GetVersionInfo

Syntax:	void Wdg_GetVersionInfo
	{
	  Std_VersionInfoType *versioninfo
	}

Service ID [hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	--
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	--
Description:	**WDG067:** This service returns the version information of this module. The version information includes: - Module Id - Vendor Id - Vendor specific version numbers (BSW00407). **WDG068:** This function shall be pre compile time configurable On/Off by the configuration parameter: <MODULE PREFIX>_VERSION_INFO_API Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.	
Caveats:	None	
Configuration:	This function is only available if configured by the compile switch WDG_VERSION_INFO_API.	

8.4 Call-back Notifications

There are no callback notifications from this module since we're at the lowest layer of the software architecture.

8.5 Scheduled functions

There are no scheduled functions in this module since it's a purely synchronous driver.

8.6 Expected Interfaces

In this chapter all interfaces required from other modules are listed.
To access the external watchdog Dio and Spi interfaces might be used depending on the user's requirement.

8.6.1 Mandatory Interfaces

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

API function	***Module***	***Description***
Dem_ReportErrorStatus	Dem	Reporting of production relevant errors.

8.6.2 Optional Interfaces

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

API function	*Module*	*Description*	*Configuration parameter (description see chapter 10)*
Det_ReportError	Det	Development error notification	WDG_DEV_ERROR_DETECT

Additional to the above SPI and DIO Interfaces might also be used to configure the External Watchdog.

8.6.3 Configurable interfaces

This module does not require any configurable interfaces.

9 Sequence diagrams

9.1 Watchdog initialization, triggering and mode switching

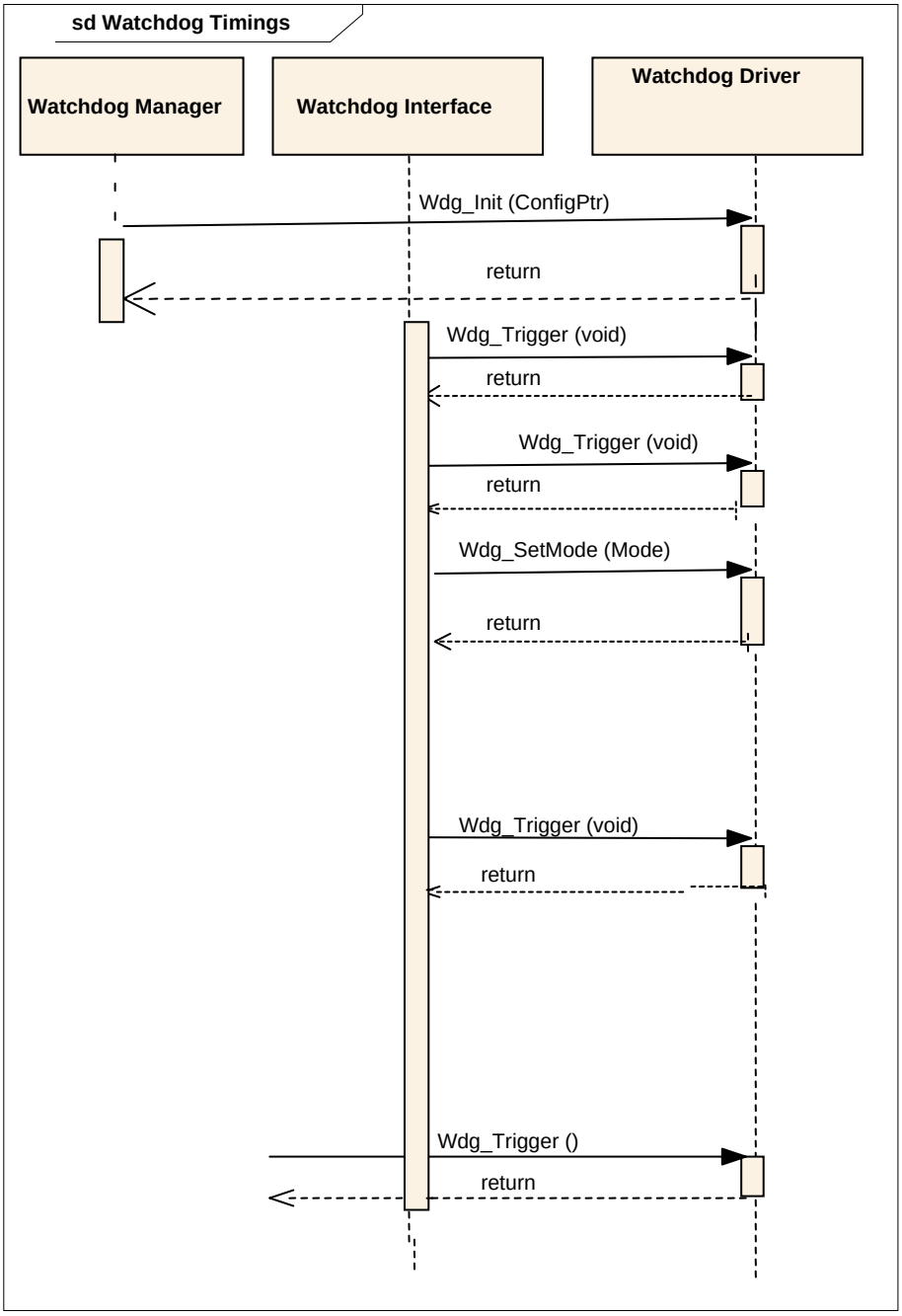


Figure 2: Sequence of watchdog initialization, triggering and mode switching

9.2 Data exchange between watchdog driver and hardware

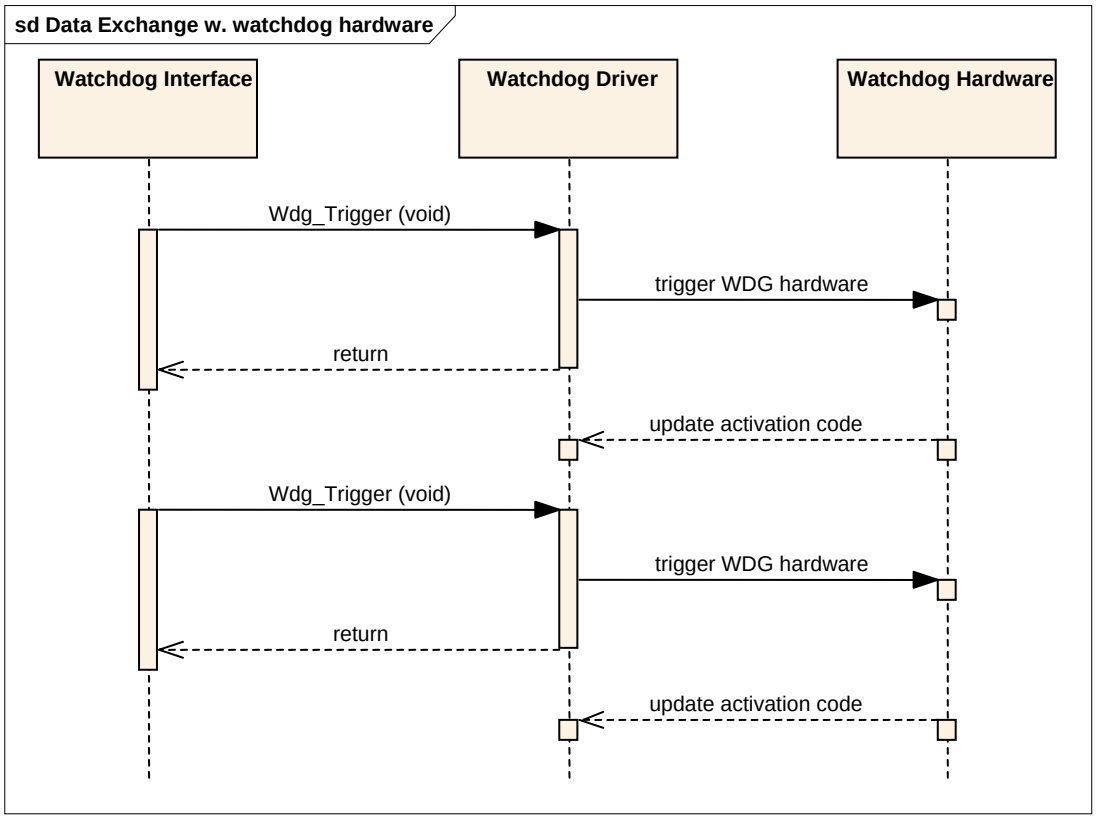


Figure 3: Data exchange between watchdog driver and hardware

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

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

- AUTOSAR Layered Software Architecture
- AUTOSAR ECU Configuration Specification
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.1.3 Specification template for configuration parameters

The following tables consist of three sections:
- the general section
- the configuration parameter section
- the section of included/referenced containers

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

Label	***Description***
x	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

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

Label	***Description***
x	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

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

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

10.2 Containers and configuration parameters

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

10.2.1 Variants

Variant PC: Settings for the different watchdog modes (see WDG082) provided as pre-compile time configuration parameters.

Variant LT: Settings for the different watchdog modes (see WDG082) provided as link-time configuration parameters.

Variant PB: Settings for the different watchdog modes (see WDG082) provided as post build time configuration parameters.

10.2.2 Wdg_Configuration

SWS Item	**WDG073:**
Container Name	Wdg_Configuration
Description	Configuratio items for the watchdog driver.
Configuration Parameters	

Name	WDG_DEV_ERROR_DETECT		
Description	**WDG069:** Compile switch to enable / disable development error detection for this module.		
Type	#define		
Unit	--		
Range	ON	Development error detection enabled	
	OFF	Development error detection disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	WDG_DISABLE_ALLOWED		
Description	**WDG070:** Compile switch to allow / forbid disabling the watchdog driver during runtime.		
Type	#define		
Unit	--		
Range	ON	Disabling the watchdog driver at runtime is allowed.	
	OFF	Disabling the watchdog driver at runtime is not allowed.	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	Safety relevant compile switch, this has to be in accordance with the corresponding settings for the watchdog manager.		

Name	WDG_TRIGGER_LOCATION		
Description	**WDG071:** Location (memory address) of the watchdog trigger routine.		
Type	#define		
Unit	--		
Range	--	Hardware dependent	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	Only relevant if provded by hardware and needed by the system.		

Name	WDG_DEFAULT_MODE		
Description	**WDG083:** Default mode for watchdog driver initialization.		
Type	WdgIf_ModeType		
Unit	--		
Range	WdgIf_FastMode	Default watchdog mode is “fast”	
	WdgIf_SlowMode	Default watchdog mode is “slow”	
	WdgIf_OffMode	Default watchdog mode is “off”	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	X	Variant LT & PB
Scope	Module		
Dependency	“Off” mode only possible if disabling the watchdog driver is allowed.		

Name	WDG_VERSION_INFO_API		
Description	**WDG081:** Compile switch to enable / disable the version information API.		
Type	#define		
Unit	--		
Range	ON	API enabled	
	OFF	API disabled	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.3 Wdg_ModeConfiguration

SWS Item	**WDG082:**
Container Name	Wdg_ModeConfiguration
Description	Configuration items for the different watchdog modes
Configuration Parameters	

Name	WDG_SETTINGS_FAST
Description	Hardware dependent settings for the watchdog driver’s “fast” mode.
Type	Hardware / implementation dependent
Unit	--

Range	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	X	Variant LT
	Post Build	X	Variant PB
Scope	Module		
Dependency	None		

Name	WDG_SETTINGS_SLOW		
Description	Hardware dependent settings for the watchdog driver's "slow" mode.		
Type	Hardware / implementation dependent		
Unit	--		
Range	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	X	Variant LT
	Post Build	X	Variant PB
Scope	Module		
Dependency	None		

Name	WDG_SETTINGS_OFF		
Description	Hardware dependent settings for the watchdog driver's "off" mode.		
Type	Hardware / implementation dependent		
Unit	--		
Range	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	X	Variant LT
	Post Build	X	Variant PB
Scope	Module		
Dependency	Only relevant if disabling the watchdog is allowed.		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.4 Wdg_ExternalConfiguration

SWS Item	
Container Name	Wdg_ExternalConfiguration
Description	Configuration items for an external watchdog hardware
Configuration Parameters	

Name	WDG_EXTERNAL_CONTAINER_REF		
Description	Reference to either - a DioChannelGroup container in case the hardware watchdog is connected via DIO pins - a SpiSequenceConfiguration container in case the watchdog hardware is accessed via SPI		
Type	Hardware / implementation dependent		
Unit	--		
Range	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	X	Variant LT
	Post Build	X	Variant PB
Scope	Module		
Dependency	See DIO resp. SPI SWS.		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

SWS Item	**WDG074:**	
Information elements		
Information element name	***Type / Range***	***Information element description***
`WDG_VENDOR_ID`	`uint16 / --`	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
`WDG_MODULE_ID`	`uint8 / --`	Module ID of this module from Module List
`WDG_AR_MAJOR_VERSION`	`uint8 / --`	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
`WDG_AR_MINOR_VERSION`	`uint8 / --`	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
`WDG_AR_PATCH_VERSION`	`uint8 / --`	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
`WDG_SW_MAJOR_VERSION`	`uint8 / --`	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
`WDG_SW_MINOR_VERSION`	`uint8 / --`	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
`WDG_SW_PATCH_VERSION`	`uint8 / --`	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
`WDG_MIN_TIMEOUT`	`milliseconds`	Minimum timeout period (see WDG059)
`WDG_MAX_TIMEOUT`	`milliseconds`	Maximum timeout period (see WDG059)
`WDG_RESOLUTION`	`milliseconds`	Resolution of watchdog timeout period (see WDG059)
`WDG_TIMEOUT_LIST`	`enumeration`	List of selectable timeout periods in milliseconds (see WDG059)
`WDG_TRIGGER_MODE`	`enumeration`	Watchdog trigger mode (toggle/window/both)

WDG075: If the watchdog hardware provides a uniform timeout resolution over the complete range, this resolution and the minimum and maximum timeout periods that can be selected shall be given. If the timeout resolution is not uniform a list of all possible timeout periods has to be provided.

11 Changes to Release 1

11.1 Deleted SWS Items

SWS Item	*Rationale*
WDG039	New SWS template for release 2.0
WDG002	Bugzilla Entry #4533

11.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
WGD030	WDG069	New SWS template for release 2.0 (copy-paste didn't work on the tags)
WDG058	WDG070	New SWS template for release 2.0 (copy-paste didn't work on the tags)
WDG053	WDG073	New SWS template for release 2.0 (copy-paste didn't work on the tags)
WDG024	WDG074	New SWS template for release 2.0 (copy-paste didn't work on the tags)
WDG054	WDG074	New SWS template for release 2.0 (copy-paste didn't work on the tags)
WDG059	WDG075	New SWS template for release 2.0 (copy-paste didn't work on the tags)
WDG060	WDG076	New SWS template for release 2.0 (copy-paste didn't work on the tags)
WDG049	WDG077	New SWS template for release 2.0 (copy-paste didn't work on the tags)
WDG050	WDG078	New SWS template for release 2.0 (copy-paste didn't work on the tags)

11.3 Changed SWS Items

SWS Item	*Rationale*
WDG045	New SWS template for release 2.0
WDG003	Bugzilla Entry #4077 (4577)
WDG017	Bugzilla Entry #4081 (4580)
WDG004	Bugzilla Entry #4582
WDG035	Bugzilla Entry #4080 (4579)
WDG037	Wrong figure referenced
WDG018, WDG052	Bugzilla Entry #4079 (4578)
WDG012	BSW00421
WDG083	Bugzilla Entry # 12138 (13066)
WDG037	Bugzilla Entry # 12136 (13068)

11.4 Added SWS Items

SWS Item	*Rationale*
WDG062	New SWS template for release 2.0
WDG063	New SWS template for release 2.0

WDG064	New SWS template for release 2.0
WDG065	New SWS template for release 2.0
WDG066	New SWS template for release 2.0
WDG067	New SWS template for release 2.0
WDG068	New SWS template for release 2.0
WDG079	New SWS template for release 2.0
WDG080	New SWS template for release 2.0
WDG081	New SWS template for release 2.0
WDG082	Added some Variants
WDG083	BSW00399