Elektrobit

EB tresos® AutoCore Generic 8
Watchdog Stack documentation

product release 8.8.7

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

Technical support

https://www.elektrobit.com/support

Legal disclaimer

Confidential information.

Table of Contents

1. Overview of EB tresos AutoCore Generic 8 Watchdog Stack documentation

Welcome to the EB tresos AutoCore Generic 8 Watchdog Stack (ACG8 Watchdog Stack) product documentation.

This document provides:

- Chapter 2, “Supported features“: list of features supported by the ACG8 Watchdog Stack
- Chapter 3, “ACG8 Watchdog Stack release notes“: release notes for the ACG8 Watchdog Stack modules
- Chapter 4, “ACG8 Watchdog Stack user guide“: background information and instructions
- Chapter 5, “ACG8 Watchdog Stack module references“: information about configuration parameters and the application programming interface

2. Supported features

ACG8 Watchdog Stack supports the following features:

- **Extended logical monitoring on several cores:** Support for multiple supervision graphs per checkpoint to allow monitoring fork/join constructs with truly concurrent execution on different cores.
- **Logical monitoring on several cores:** Support for monitoring of control flow graphs that are distributed across several cores.
- **Deadline monitoring on several cores:** Support for deadline monitoring on several cores with start and end checkpoints located on the same core.
- **Alive supervision on several cores:** Support for alive supervision on several cores with a single main function on a master core.
- **Alive supervision:** Support for cyclic triggering of a watchdog hardware via the MCAL driver Wdg.

3. ACG8 Watchdog Stack release notes

3.1. Overview

This chapter provides the ACG8 Watchdog Stack product specific release notes. General release notes that are applicable to all products are provided in the EB tresos AutoCore Generic documentation. Refer to the general release notes in addition to the product release notes documented here.

3.2. Scope of the release

3.2.1. Configuration tool

Your release of EB tresos AutoCore is compatible with the release of the EB tresos Studio configuration tool:

- EB tresos Studio: 29.2.0 b220916-0321

3.2.2. AUTOSAR modules

The following table lists the AUTOSAR modules that are part of this ACG8 Watchdog Stack release.

Module name	AUTOSAR version and revision	SWS version and revision	Module version	Supplier
WdgIf	4.0.3 []	2.5.0 [0000]	6.1.30	Elektrobit Automotive GmbH
WdgM	4.0.3 []	2.2.0 [0000]	6.1.43	Elektrobit Automotive GmbH

Table 3.1. Hardware-Independent Modules specified by the AUTOSAR standard

3.2.3. EB (Elektrobit) modules

The following table lists all modules which are part of this release but are not specified by the AUTOSAR standard. These modules include tooling developed by EB or they may hold files shared by all other modules.

Module name	Module version	Supplier
No EB modules available		

Table 3.2. Modules not specified by the AUTOSAR standard

3.2.4. MCAL modules and EB tresos AutoCore OS

For information about MCAL modules and OS, refer to the respective documentation, which is available as PDF at `$TRESOS_BASE/doc/3.0_EB_tresos_AutoCore_OS` and `$TRESOS_BASE/doc/5.0_MCAL_modules`[1]. It is also available in the online help in EB tresos Studio. Browse to the folders `EB tresos AutoCore OS` and `MCAL modules`.

3.3. Module release notes

3.3.1. Wdglf module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 2.5.0
- Module version: 6.1.30.B567464
- Supplier: Elektrobit Automotive GmbH

3.3.1.1. Change log

This chapter lists the changes between different versions.

Module version 6.1.30
2022-10-26

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.1.29
2022-07-04

[1] `$TRESOS_BASE` is the location at which you installed EB tresos Studio.

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.1.28
2022-03-09

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.1.27
2021-10-08

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.1.26
2021-06-25

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.1.25
2021-03-05

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.1.24
2020-10-23

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.1.23
2020-06-19

- Change all NO_INIT memory sections to CLEARED.
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.1.21
2020-05-04

- Time 2.1 RFM Release.

Module version 6.1.20
2020-02-21

- ASCWDGIF-321 Fixed known issue: Wrong checks in the configuration for VendorApiInfix will lead to generation fails.

Module version 6.1.19
2019-12-17

- Time 2.0 RFD Release.

Module version 6.0.19
2019-10-11

- Implemented ASIL tags.

Module version 6.0.18
2019-06-14

- Implemented multicore support.
- Support for BSWMD VendorApiInfix.

Module version 6.0.17
2019-04-18

- Update generator to disable vendor infix for a single referenced driver.

Module version 6.0.16
2018-10-25

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.0.15
2018-05-25

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.0.14
2018-03-02

- Implemented macro-redefinition guards.
- Implemented compliance to MISRA-C:2012.
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.0.13
2017-09-22

Module version 6.0.12
2017-04-03

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.0.11
2016-11-07

- Changed WdgIf_ModeType form enumaration to macro definition.

Module version 6.0.10
2014-07-11

- Improved ECUC parameter checks

Module version 6.0.9
2013-12-13

- Added non-functional code improvements
- Updated the Basic Software Module Description to specify all external and internal APIs (Basic Software Module Entities)

Module version 6.0.8
2013-07-04

- Added non-functional code improvements

Module version 6.0.7
2013-06-25

- Added non-functional code improvements

Module version 6.0.6
2013-06-14

- Added non-functional code improvements

Module version 6.0.5
2013-05-10

- ASCWDGIF-207 Fixed known issue: WdgIf uses wrong API calls to the Wdg driver if multiple Wdgs are configured

Module version 6.0.4
2013-03-15

- Added non-functional code improvements

Module version 6.0.3
2013-02-08

- Added specification of memory mappings to Basic Software Module Description

Module version 6.0.2
2012-10-12

- Changed the top-level structure of the software-component description in the ARXML files from `/AUTOSAR/WdgIf` to `/AUTOSAR_WdgIf`
- Added non-functional code improvements

Module version 6.0.1
2012-03-16

- Added macro definition for single Wdg driver independent of Det
- Removed compiler warnings for redefined `WdgIf_SetMode` and `WdgIf_SetTriggerCondition`
- Added symbolic name values for `WdgIfDeviceIndex`
- Added generation of BSWMD

Module version 6.0.0
2012-02-17

- Initial AUTOSAR 4.0 version

3.3.1.2. New features

- No new features have been added since the last release.

3.3.1.3. Elektrobit-specific enhancements

This chapter lists the enhancements provided by the module.

- This module provides no Elektrobit-specific enhancements.

3.3.1.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- No consistency check between code files and header files

  Description:

  The inter-module version checks as specified by the WdgIf SWS are not implemented.

  Rationale:

  - The required compile-time version checks would result in an inflexible, hardly integratable basic software stack.
  - EB tresos AutoCore is an already integrated product.
  - The project handling of EB tresos Studio provides means to enforce that only modules with the same EB tresos AutoCore release version can be added to the project.

Requirements:

WDGIF005

- No AUTOSAR Debugging support

Description:

WdgIf is not instrumented for the usage with AUTOSAR Debugging.

WDGIF052, WDGIF053, WDGIF054, WDGIF055

3.3.1.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- For this module no limitations are known.

3.3.1.6. Open-source software

WdgIf does not use open-source software.

3.3.2. WdgM module release notes

- AUTOSAR R4.0 Rev 3
- AUTOSAR SWS document version: 2.2.0
- Module version: 6.1.43.B567464
- Supplier: Elektrobit Automotive GmbH

3.3.2.1. Change log

This chapter lists the changes between different versions.

Module version 6.1.43
2022-10-26

- ASCWDGM-1016 Fixed known issue: Wrong usage of P2CONST creates compilation problems for some compilers.
- ASCWDGM-1014 Fixed known issue: WdgM does not compile when Master Instance is mapped to the highest core.
- ASCWDGM-1006 Fixed known issue: WdgM_GetAllExpiredSEID may not return all expired supervised entities.
- ASCWDGM-1008 Fixed known issue: WdgM_GetFirstExpiredSEID() may not provide the first expired supervised entity when multicore is enabled.

Module version 6.1.42
2022-07-04

- ASCWDGM-979 Fixed known issue: Wrong supervised entity IDs are stored when the WdgMGetAllExpiredSEIDs feature is enabled.
- Added support for multicore mixed criticality.
- Removed memory mapping for AUTOSAR 3.2 compatibility.
- Fixed non-compliant use of memory sections.

Module version 6.1.41
2022-03-09

- ASCWDGM-949 Fixed known issue: WdgM does not compile when SEs are not mapped to all WdgM core instances via an OsApp ref.
- TimE Protection license only: Removed TimE license for Logical Supervision and Deadline Supervision.
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.1.40
2021-10-08

- ASCWDGM-911 Fixed known issue: Missing memory sections for multi-core main functions declaration.
- ASCWDGM-930 Added support for slave instance triggering of the watchdog drivers.
- ASCWDGM-914 Added support for partition reset.
- ASCWDGM-923 Added new callout to signal a MainFunction violation.
- ASCWDGM-905 Implemented new API to retrieve all expired Supervised Entities.
- Improved the generation of sattelite MainFunction BSWMD information.

Module version 6.1.39
2021-06-25

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.1.38
2021-03-05

- Internal module improvement. This module version update does not affect module functionality.
- Added generation of MainFunction Timing Event for TimE.
- Behavior improvement: Decrease execution time for configurations which have many Supervised Entities configured.

Module version 6.1.37
2020-10-23

- Internal module improvement. This module version update does not affect module functionality.
- ASCWDGM-826: Added ASR 4.3 service component compatibility for WdgM.
- ASCWDGM-865 Fixed known issue: Code generator mixes Index and WdgMSupervisedEntityId for symbol names.

Module version 6.1.36
2020-06-19

- Change all NO_INIT memory sections to CLEARED and restriction on Deadline Supervision in the Limitations.xml file due to multicore use.
- Internal module improvement. This module version update does not affect module functionality.

Module version 6.1.34
2020-05-04

- Time 2.1 RFM Release.

Module version 6.1.33
2020-04-03

- ASCWDGM-812 Fixed known issue: WdgM_Mainfunction not generated for slaves

Module version 6.1.32
2020-02-21

- ASCWDGM-803 Fixed known issue: Different memory mapping area for definition and declaration of WdgM_EB_GlobalStatus
- ASCWDGM-795 Fixed known issue: WdgM service component does not work with multi-core distribution
- ASCWDGM-813 Fixed known issue: "Space" character present in the last line of the file WdgM_Lcfg.h leads to compiler error

Module version 6.1.31
2019-12-17

- TimE 2.0 RFD Release.

Module version 6.0.31
2019-10-11

- ASCWDGM-758 Fixed known issue: BSWMD does not generate the BSW implementations of all cores.

Module version 6.0.30
2019-06-14

- Implemented multicore support.

Module version 6.0.29
2018-10-25

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.0.28
2018-06-22

- ASCWDGM-654 Fixed known issue: Variables are not assigned to a memory section.
- Internal module improvement. This module version update does not affect module functionality

- Removed AUTOSAR 3.1 support from the module.

Module version 6.0.27
2018-03-02

- Internal module improvement. This module version update does not affect module functionality
- Invert logic for AUTOSAR 4.0.2 and remove AUTOSAR 3.x legacy support for symbolic names
- ASCWDGM-629 Fixed known issue: Behaviour changed when the same alive supervision is used after switching the mode.
- Added immediate mode switch when calling WdgM_SetMode().

Module version 6.0.26
2017-09-22

- ASCWDGM-594 Fixed known issue: Dem event in WdgM_Cfg.h causes compilation error
- Comply to MISRA-C:2012

Module version 6.0.25
2017-03-31

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.0.24
2017-03-10

- Internal module improvement. This module version update does not affect module functionality.

Module version 6.0.23
2016-11-07

- Extended license checks for logical supervision or deadline supervision to cover new license strings EB_-TIME_CFM and EB_TIME_DM as well.

Module version 6.0.22
2015-11-06

- Added non-functional code improvements to fix compiler warnings.

Module version 6.0.21
2015-10-23

- Added non-functional code improvements to fix compiler warnings.

Module version 6.0.20
2015-06-19

- ASCWDGM-515 Fixed known issue: Support for automatic configuration of DEM events

Module version 6.0.19
2015-01-07

- Added non-functional code improvements to fix MISRA violations.

Module version 6.0.18
2014-04-25

- Added non-functional code improvements to improve text of some container descriptions

Module version 6.0.17
2013-12-13

- Added support for the integration into an AUTOSAR 3.1 environment
- ASCWDGM-467 Fixed known issue: Compilation fails on case-sensitive file systems
- ASCWDGM-493 Fixed known issue: Configuration for reporting of DEM events `WDGM_E_MONITORING` and `WDGM_E_SET_MODE` is not possible if DEM event `WDGM_E_IMPROPER_CALLER` is not used

Module version 6.0.16
2013-10-11

- Added non-functional code improvements to update source code documentation for production error reporting

- ASCWDGM-474 Fixed known issue: The WdgM may not compile if reporting a production error to the Diagnostic Event Manager is configured
- Added non-functional code improvements to ease the integration of the WdgM into an ASR31 environment
- TimE Protection license only: Removed obsolete feature for the configuration of DEM callouts

Module version 6.0.15
2013-08-07

- Added non-functional code improvements to fix incorrect generation of macro values `0.0U` to `0U`

Module version 6.0.14
2013-07-24

- Added non-functional code improvements to deal with tasking compiler bug on XC2361E (V2.5 r1)

Module version 6.0.13
2013-07-23

- ASCWDGM-447 Fixed known issue: The WdgM uses incorrect compiler abstraction

Module version 6.0.12
2013-07-11

- Fixed minor code issues (non-function changes) for future debugging support

Module version 6.0.11
2013-06-25

- Fixed compiler warning on XC2k derivate reported from a tasking compiler
- Added non-functional code improvements
- ASCWDGM-412 Fixed known issue: TimE Protection license only: A data corruption in the internal WdgM data may not result in Global Supervision Status `EXPIRED`

Module version 6.0.10
2013-06-14

- ASCWDGM-379 Fixed known issue: Inconsistent starting point of the Supervision Reference Cycle for the first evaluation of Alive Supervision
- ASCWDGM-378 Fixed known issue: TimE Protection license only: The WdgM never recovers from `FAILED` state in case the Error Recovery feature is enabled and only Deadline Supervision or/and Logical Supervision are configured for the current WdgM mode
- ASCWDGM-383 Fixed known issue: Wrong individual mode switch notification behavior during initialization and de-initialization phases
- ASCWDGM-384 Fixed known issue: TimE Protection license only: Missing checks allow an inconsistent configuration leading to undefined behavior if Deadline or Logical Supervision is used
- ASCWDGM-385 Fixed known issue: TimE Protection license only: Deadline Supervision may not be evaluated according to the specification with respect to parameters `WdgMMainFunctionPeriodTolerance`, `WdgMSupervisionCycle`, `WdgMDeadlineMax`, and `WdgMDeadlineMin`
- Added non-functional code improvements
- Added simple Dbg instrumentation for tracing of function enter/exit points
- ASCWDGM-385 Fixed known issue: Unspecified behavior if `WdgIf_SetMode` fails during initialization of the WdgM.
- ASCWDGM-393 Fixed known issue: TimE Protection license only: Unspecified behavior of Alive Supervision if configured together with Logical or Deadline Supervision and an enabled Error Recovery
- ASCWDGM-397 Fixed known issue: TimE Protection license only: The WdgM does not compile for some specific Deadline Supervision configurations
- ASCWDGM-404 Fixed known issue: The WdgM does not compile if more than one CallerId ID is configured for a mode switch request
- ASCWDGM-406 Fixed known issue: TimE Protection license only: Unspecified behavior if the WdgM switches back to an old mode while some Supervisions are still active

Module version 6.0.9
2013-04-30

- Added non-functional code improvements
- ASCWDGM-359 Fixed known issue: TimE Protection license only: WdgM does not compile if callout API is configured for `GetExpectedWdgMMode`
- ASCWDGM-361 Fixed known issue: TimE Protection license only: Deadline Supervision is not performed for Checkpoints which are configured to be both Start Checkpoint and End Checkpoint
- Implemented tracking of a failed Logical Supervision Status for the successor Checkpoints of a Logical Supervision Graph
- ASCWDGM-363 Fixed known issue: TimE Protection license only: A failed Deadline Supervision may not lead to global state `EXPIRED` if Error Recovery is enabled

- ASCWDGM-369 Fixed known issue: TimE Protection license only: The individual mode switch callout API contains an incorrect old mode if the Supervised Entity is deactivated
- ASCWDGM-371 Fixed known issue: The WdgM does not overwrite an active initialization request when required to by a call to `WdgM_DeInit()`

Module version 6.0.8
2013-03-15

- Added support for Alive Supervision of multiple Checkpoints of a Supervised Entity
- TimE Protection license only: Added support for Logical Supervision
- TimE Protection license only: Added support for Deadline Monitoring
- TimE Protection license only: Implemented detection of deadline violation of Supervised Entities in the granularity of the main function cycle
- Implemented smooth error reaction without a reset for individual Supervised Entities
- TimE Protection license only: Implemented detection of timing violation of schedule main function
- Added non-functional code improvements

Module version 6.0.7
2013-02-08

- Added specification of Memory Mappings to Basic Software Module Description

Module version 6.0.6
2012-12-21

- Added non-functional code improvements
- Implemented usage of AUTOSAR conform type naming for `ModeDeclarationGroups`
- TimE Protection license only: Added support for integration in safety projects up to ASIL level D

Module version 6.0.5
2012-10-12

- Added AUTOSAR 3.2 support of Rte Interface and SWCD
- Added non-functional code improvements

- Added support for optional generation of Service APIs according to AUTOSAR 3.2

Module version 6.0.4
2012-08-17

- Added definition of Exclusive Area Activation in Basic Software Module Description
- Fixed minor issues in generated WdgM Service Component Description
- Added support of symbolic name generation for Checkpoint IDs via Rte

Module version 6.0.3
2012-07-13

- Added internal mode switch to a final mode within `WdgM_DeInit` is now optional

Module version 6.0.2
2012-06-15

- Updated WdgM to derive value for `WdgMWatchdogName` from container name
- Fixed compiler warning in case no Wdgs are configured
- ASCWDGM-248 Fixed known issue: The Watchdog trigger conditions are incorrectly updated in the `MainFunction()` after an enforced reset via `WdgM_PerformReset()`
- Added switches for optional containers on same tabs as lists
- ASCWDGM-256 Fixed known issue: Inconsistent storage class specification for `WdgM_EB_UpdateTriggerConditions()` and `WdgM_EB_SetMode()` cause an error or warning

Module version 6.0.1
2012-03-16

- Added non-functional code improvements
- Added generation of BSWMD

Module version 6.0.0
2012-02-17

- Initial AUTOSAR 4.0 version

3.3.2.2. New features

- No new features have been added since the last release.

3.3.2.3. Elektrobit-specific enhancements

This chapter lists the enhancements provided by the module.

- Optimized usage of WdgM in case no RTE is used

  Description:

  The parameter `WdgMRteUsage` (see `WDGM.EB.WdgMRteUsage_Conf`) can be used to disable the RTE interface of the Watchdog Manager. This means that the Watchdog Manager module can be used without an RTE if needed.

  Rationale:

  Disabling the usage of the Rte makes the integration of the WdgM at the beginning easier.

- Enhanced production error reporting

  Description:

  An enhanced production error reporting mechanism has been introduced. This allows to configure the following options independently for each Dem event:

  - Report production errors to the Diagnostics Event Manager (Dem).
  - Report production errors to the Development Error Tracer (Det) as development errors.
  - Do not report production errors at all.

  If a production error is redirected towards the Det, you may configure the reported Det error-ID.

  Rationale:

  This enhancement implements the HIS requirements concerning fault operation and error detection: HisGen0007, HisGen0008, and HisGen0009.

- Watchdog Manager mode switch to a sleep mode

  Description:

  The configuration parameter `WdgMSleepMode` in the general tab of a `WdgMConfigSet` entry (see `WDGM.EB.WdgM_DeInit.2`) allows the integrator to specify a sleep mode. The `WdgM_DeInit` function then updates the trigger conditions via a Watchdog Manager mode switch to this sleep mode.

  Rationale:

The integrator of the WdgM can specify the WdGM sleep mode. The EcuM does not need to explicitly switch the WdgM mode before it calls the `DeInit` function.

- Provision of Checkpoint IDs via WdgM Service Component Description

  Description:

  The WdgM Service Component Description specifies the name of the configured Checkpoint IDs of each Supervision Entity as a constant within the interface description as follows: `WdgMConf_WdgMCheck-point_<CheckpointName>`.

  Rationale:

  A SWC usually includes only the corresponding Rte header file and not the BSW header file of the Watchdog Manager. Therefore, the Rte must have knowledge about the configured Checkpoint IDs in order to generate the symbolic name values for the Checkpoint IDs used in the SWC.

- Support for optional generation of AUTOSAR 3.2 Service Component Description

  Description:

  Support for the generation of AUTOSAR 3.2, or AUTOSAR 4.0 service APIs and Software Component Description as well as default service APIs and Software Component Description which can be configured to adhere either to AUTOSAR 3.2 or 4.0 schema version.

  Rationale:

  AUTOSAR 3.2/3.1 application SWCs of Tier-1 shall be deployed in an AUTOSAR 4.0 environment or AUTOSAR 3.2/3.1 environment.

- Support for the configuration of callout functions for the integration into projects with ASIL level up to ASIL D

  Description:

  The Watchdog Manager provides the configuration of callout functions for the following:

  - Error indication (instead of DET and DEM reporting).
  - Periodically polling the information regarding the Watchdog Manager state (e.g. Initialization, Watchdog Manager Mode, etc.) from an external entity (e.g. Safety Manager) instead of providing the APIs `WdgM_Init`, `WdgM_DeInit`, or `WdgM_SetMode`.

  Rationale:

  Support for the integration of Time and Execution Protection for automotive projects with ASIL level up to ASIL D.

- Detection of deadline violations in the granularity of the main function cycle

  Description:

In addition to deadline supervision of AUTOSAR, the Watchdog Manager detects a deadline violation in the granularity of the main function period in case the Stop Checkpoint of an active deadline monitoring is never called.

Rationale:

Required for projects having ASIL level up to ASIL D.

- Detection of timing violation of scheduled main function

Description:

The Watchdog Manager monitors its own main function period and reports an error in case a timing violation is detected.

Rationale:

Required for projects having ASIL level up to ASIL D.

- Smooth error reaction without a reset for individual Supervised Entities.

Description:

The Watchdog Manager provides the configuration of a smooth error reaction with error recovery for individual Supervised Entities. In this case, the Watchdog Manager reports the status of a failed Supervision without entering the `Expired` state such that a Watchdog reset will not be performed.

Rationale:

Projects may require a different error strategy than provided by the Watchdog Manager (Watchdog reset) in case a Supervised Entity fails.

- Tracking of a failed Logical Supervision Status for the successor Checkpoints of a Logical Supervision Graph

Description:

If the Watchdog Manager detects a failed Logical Supervision for a called Checkpoint participating in an active Supervision Graph, then a call to the API `WdgM_CheckpointReached()` shall return `E_NOT_OK` for all successor Checkpoints of this Supervision Graph.

Rationale:

A Software Component participating in a control loop (e.g. responsible for controlling an actuator) may trust on the return value of the API `WdgM_CheckpointReached()` to decide whether or not the input values are produced in the correct sequence. Thus the input values can be used independent of the schedule period of Watchdog Manager `MainFunction`.

3.3.2.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- BswM_WdgM_RequestPartitionReset API is not supported

  Description:

  In contrast to WDGM225 which states that the WdgM shall restart/shutdown the partition of an OS Application which is configured for a Supervised Entity by calling `BswM_WdgM_RequestPartitionReset` of the Basic Software Mode Manager module, the WdgM will call `WdgMRequestPartitionResetCallout` (see WDGM.EB.TIMEPM.WDGM020121_Conf) of container `WdgMSupervisorCallouts` instead.

  Requirements:

  WDGM225, WDGM162

- Post-build time configuration

  Description:

  Only pre-compile time configuration is supported (reference to product description: ASCPD-77)

  Requirements:

  WDGM127, WDGM004, WDGM042, WDGM010, WDGM029, WDGM266, WDGM255

- The WdgM does not check if a Supervised Entity ID equals some AUTOSAR module ID

  Description:

  In contrast to AUTOSAR which does not allow the configuration of SWC Supervised Entities with a Supervised Entity ID value that is equal to the module ID of any AUTOSAR BSW module, there is no such constraint on the Supervised Entity ID values. Users/integrators are responsible for the correct configuration of Supervised Entity ID values and the possible consequences.

  Rationale:

  There may exist use-cases in legacy projects where some Supervised Entities are not associated to SWCs. Therefore these Supervised Entities may require the configuration of the ID of some AUTOSAR BSW module.

  Requirements:

  WDGM307

- Optional detection of production code errors

  Description:

In contrast to AUTOSAR which does not allow to switch off the detection of production code errors for each individual production code error, the following is possible:

- To keep the production code error as specified.
- To report the production code error to the Development Error Tracer (DET) instead.
- To completely switch off the detection of the production code error.

Rationale:

User-specific configuration of production code errors.

Requirements:

WDGM015

- De-initialization of the Watchdog Manager independent of global Supervision Status

Description:

In contrast to AUTOSAR which allows the de-initialization only from global Supervision Status `WDGM_GLOBAL_STATUS_OK`, the Watchdog Manager can be de-initialized independent of the actual global Supervision Status.

Rationale:

The caller of `WdgM_DeInit` must be aware of the actual global Supervision Status.

Requirements:

WDGM286

- Consistent interpretation of parameter `WdgMFailedAliveSupervisionRefCycleTol`

Description:

AUTOSAR generally defines the parameter `WdgMFailedAliveSupervisionRefCycleTol` as the number of allowed failed reference cycles until a Supervised Entity goes into state `WDGM_LOCAL_STATUS_EXPIRED`. However, AUTOSAR specifies a state machine where one additional failed reference cycle is allowed. In contrast to the state machine specified in AUTOSAR, the implementation allows at most `WdgMFailedAliveSupervisionRefCycleTol` failed reference cycles until a Supervised Entity goes into state `WDGM_LOCAL_STATUS_EXPIRED`.

See Bugzilla entry http://www.autosar.org/bugzilla/show_bug.cgi?id=58303

Requirements:

WDGM206, WDGM204

- Consistent interpretation of parameter `WdgMExpiredSupervisionCycleTol`

Description:

AUTOSAR generally defines the parameter `WdgMExpiredSupervisionCycleTol` as the number of allowed main function cycles in global state `WDGM_GLOBAL_STATUS_EXPIRED` until the WdgM enters the global state `WDGM_GLOBAL_STATUS_STOPPED`. However, AUTOSAR specifies a state machine where two additional main function cycles in global state `WDGM_GLOBAL_STATUS_EXPIRED` are allowed. In contrast to the state machine specified in AUTOSAR, the implementation allows at most `WdgMExpiredSupervisionCycleTol` main function cycles in global state `WDGM_GLOBAL_STATUS_EXPIRED` until the WdgM enters the global state `WDGM_GLOBAL_STATUS_STOPPED`.

See Bugzilla entry http://www.autosar.org/bugzilla/show_bug.cgi?id=58303

Requirements:

WDGM077, WDGM219, WDGM220

- Symbolic port names instead of numeric port name numbering

Description:

In contrast to requirements WDGM.ASR40.WDGM147 and WDGM.ASR40.WDGM149, the RTE ports are named by their symbolic short name taken from the configuration.

Rationale:

Symbolic port names do not change when ports are deleted or inserted as it is the case for numeric names because they get renumbered and need to be reconnected. Also the symbolic name can be chosen to reflect the purpose of a port which makes the port connection process easier and less error prone.

Requirements:

WDGM147, WDGM149

- `WdgM_GetVersionInfo` as a function

Description:

In contrast to WDGM262 which suggests to implement the API as a macro in case caller and callee of `WdgM_GetVersionInfo` are available at compile time, the WdgM always implements the API as a function.

Requirements:

WDGM262

- No AUTOSAR Debugging support

Description:

WdgM is not instrumented for the usage with AUTOSAR Debugging.

Requirements:

WDGM238, WDGM239, WDGM240, WDGM241, WDGM242, WDGM234, WDGM235, WDGM236, WDGM237

- WdgM does not check the versions of other modules

Description:

In contrast to WDGM013, the WdgM does not check the version numbers of included header files from other modules.

Rationale:

In general, the modules are delivered within a whole AutoCore delivery, in which the versions are consistent and therefore do not have to be checked.

Furthermore, this allows the combination of the module with other AUTOSAR compatible but not fully compliant modules. This might e.g., permit to combine the module with (adapted) modules from different AUTOSAR releases or with non-AUTOSAR modules that simulate the necessary behavior.

Requirements:

WDGM013

- Mode switch is done synchronously or asynchronously (synchronously to `MainFunction`) if the WdgMSetModeSynchron is set respectively not set.

Description:

In contrast to AUTOSAR which specifies that a call to `WdgM_SetMode` immediately switches the WdgM mode (WDGM186), the `WdgM_SetMode` request is applied either at the end of the next `MainFunction` call, either like AUTOSAR specifies, depending on how the WdgMSetModeSynchron parameter is configured.

See Bugzilla entry http://www.autosar.org/bugzilla/show_bug.cgi?id=57805.

Requirements:

WDGM154

- (De-)Initialization is done synchronously to `MainFunction`

Description:

In contrast to AUTOSAR which specifies that a call to `WdgM_Init` or `WdgM_DeInit` immediately initializes or de-initializes the Watchdog Manager, the Watchdog Manager is (de-)initialized at the next `MainFunctionCycle`.

Requirements:

WDGM268, WDGM269, WDGM285, WDGM298, WDGM296, WDGM151, WDGM018, WDGM135, WDGM350, WDGM286, WDGM261

- Non-compliant deviations in vendor-specific module definition file

  Description:

  The vendor-specific module definition file (VSMD) has non-compliant deviations to the AUTOSAR specification:

  Violations against Rule EcucSws_1014: Additional vendor specific parameter definitions (using ParameterTypes), container definitions and references shall be added to the VSMD according to the alphabetical order.

  This affects variables and containers in following StMD-Nodes:

  - `/AUTOSAR/WdgM`
  - `/AUTOSAR/WdgM/WdgMGeneral`

  Rationale:

  A merge of AUTOSAR and vendor specific variables in these containers intentionally results in a different order for a clear arrangement of vendor specific parameters in EB tresos Studio.

3.3.2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- Restriction of numbering of Checkpoint IDs

  Description:

  WDGM306_Conf does not specify any constraints for the parameter `WdgMCheckpointId` except that the ID must be unique within the Supervised Entity. In contrast to WDGM306_Conf, all Checkpoint IDs within a Supervised Entity must be zero-based and dense.

  Rationale:

  This reduces the consumption of RAM and ROM.

Requirements:

WDGM306_Conf

- Restriction of number of configurable Checkpoints per Supervision Entity

Description:

The AUTOSAR Specification of the Watchdog Manager specifies that the container `WdgMCheckpoint` as part of `WdgMSupervisedEntity` has a maximum multiplicity of 65535. In contrast to this, at most 256 checkpoints can be configured for a Supervision Entity.

Rationale:

This reduces the consumption of RAM and ROM.

Requirements:

WDGM305_Conf

- Restriction of number of configurable Supervision Entities

Description:

The AUTOSAR Specification of the Watchdog Manager specifies that the container `WdgMSupervisedEntity` as part of `WdgMGeneral` has a maximum multiplicity of 65535. In contrast to this, at most 256 Supervised Entities can be configured.

Rationale:

This reduces the consumption of RAM and ROM.

Requirements:

WDGM303_Conf

- Restriction on ID range of Checkpoints and Supervised Entities

Description:

AUTOSAR states that the range of valid IDs depends on the number of configured Supervised Entities and on the chosen platform type. In contrast to this, the WdgM always uses the maximum possible range of `uint16` instead of `uint8`.

Requirements:

WDGM038

- Restriction on dynamic change of the main function period

Description:

In contrast to AUTOSAR which specifies a mode-dependent `WdgM_MainFunction` period, the `MainFunction` period is always defined via the `WdgMSupervisionCycle` parameter of the first configured mode.

- Restriction on architectural assumption regarding data access

  Description:

  The CPU of the ECU where WdgM is integrated provides 32-bit data types which can be read and written in an atomic way.

- Restriction on Deadline Supervision

  Description:

  Checkpoints used in Deadline Supervision must be called from the same core.

3.3.2.6. Open-source software

WdgM does not use open-source software.

4. ACG8 Watchdog Stack user guide

4.1. Overview

This user guide describes the concepts and the configuration of the modules:

- Watchdog Interface (`WdgIf`)
- Watchdog Manager (`WdgM`)

This user guide is intended for readers who have good knowledge of AUTOSAR and about the purpose of the ACG8 Watchdog Stack modules.

4.2. Background information

4.2.1. Stack overview

The watchdog stack of EB tresos AutoCore contains a group of modules that simplify the watchdog task on an automotive ECU. Figure 4.1, “Watchdog stack overview” illustrates the structure of the watchdog stack.

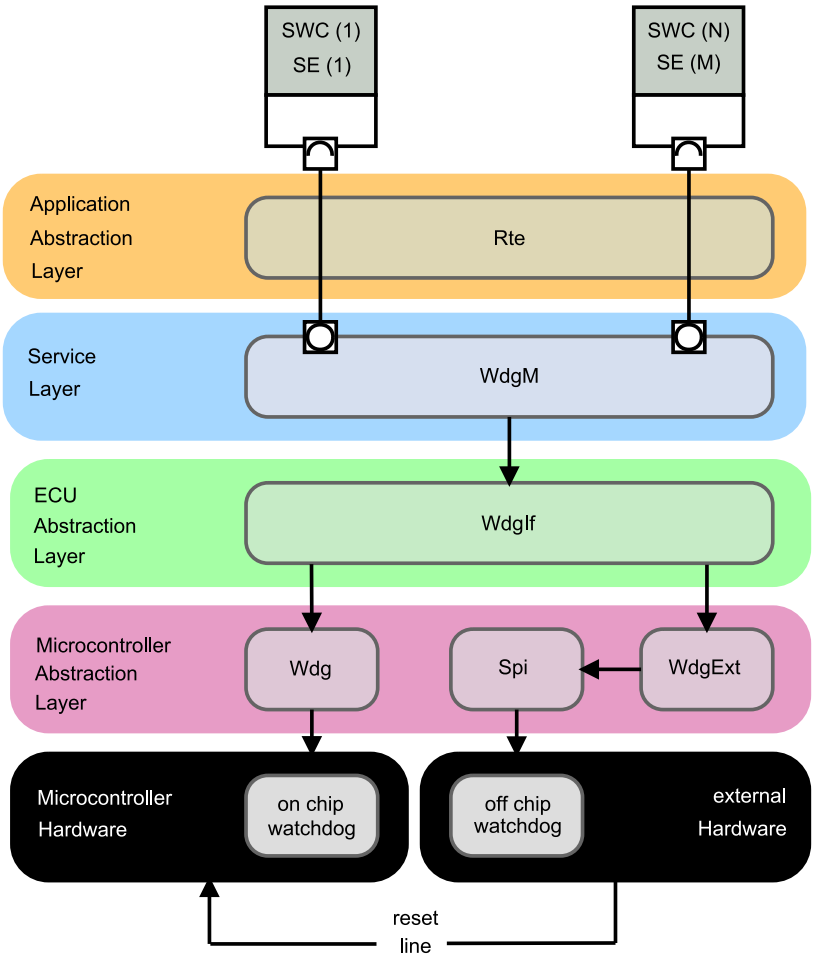

Figure 4.1. Watchdog stack overview

The image shows two software components, the SWC (1) and SWC (N). The `WdgM` module can supervise each of these software components. A software component which is supervised is also called *supervised entity* (SE (1) and SE (M) in Figure 4.1, "Watchdog stack overview").

Details about the software component alive supervision are available in Section 4.6.2.3, "Alive supervision of supervised entities".

The Watchdog Manager (`WdgM`) reports via the Watchdog Interface (`WdgIf`) a triggering condition to the Watchdog Drivers (`Wdg`). The Watchdog Driver then is responsible for triggering the watchdog hardware as long as the triggering condition is true.

For details about triggering the watchdog, see Section 4.6.2.4, "Triggering the watchdog".

4.2.2. Module dependencies

This chapter summarizes the modules that are needed by the watchdog stack to simplify the integration on your ECU.

4.2.2.1. Development Error Detection (Det)

The watchdog stack uses the following API function of the `Det` module:

- `Det_ReportError()`

4.2.2.2. Diagnostic Event Manager (Dem)

The watchdog stack uses the following API function of the `DEM` module:

- `Dem_ReportErrorStatus()`

4.2.2.3. Micro-Controller Unit (Mcu)

The watchdog stack uses the following API function of the `Mcu` module:

- `Mcu_PerformReset()`

4.2.2.4. Runtime Environment (Rte)

The `WdgM` uses the following API function of the `Rte` module:

- `Rte_Switch_<port>_<mode declaration group prototype>()`

4.2.2.5. Schedule Manager (SchM)

The following main function needs to be called by the `SchM` module:

- `WdgM_MainFunction`

4.2.2.6. ECU State Manager (EcuM)

The following (de)init functions need to be called by the `EcuM` module:

- `WdgM_Init()`
- `WdgM_DeInit()`
- `Wdg_Init()`

Note: In sleep modes, the hardware watchdog is triggered by the `EcuM` module.

4.2.2.7. BSW Mode Manager (BswM)

The watchdog stack uses the following API function of the `BswM` module:

- `BswM_WdgM_RequestPartitionReset()`

4.2.3. Multi-core support

The watchdog stack of EB tresos AutoCore supports the use in multi-core systems. It is based on a master-satellite concept, with one master instance and one or more satellite instances. Figure 4.2, "Multi-core watchdog stack overview" illustrates the structure of the multi-core watchdog stack.

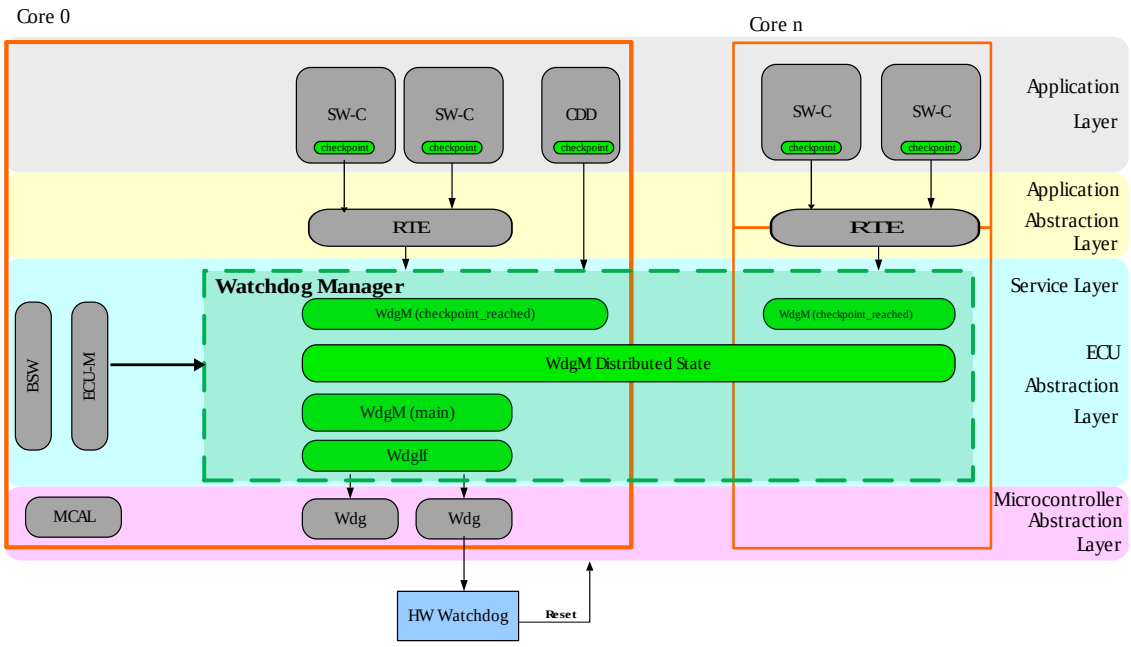

Figure 4.2. Multi-core watchdog stack overview

The following items are processed by the master instance and satellite instances (i.e. all cores in the above picture):

- Get the local status for each supervised entity: retrieve the status of all checkpoints of a supervised entity mapped to a core
- Determine the status of the alive indication for each core
- Determine the local supervision status for each core instance: this is composed of the result of deadline supervision, logical supervision, and alive supervision of that core

The following items are processed only by the master instance (core 0 in the above picture):

- Determine the global supervision status: set the global supervision status (`WdgM` may observe more than one supervised entity spread on multiple cores)
- Update the watchdog trigger condition: if the global supervision status is OK

4.3. Initializing the watchdog stack

The initialization of the watchdog stack is called by the `EcuM` module. Therefore the `WdgM` and the `Wdg` modules have to be inserted in the start-up sequence *Startup One*. For details on this sequence, see the EB tresos AutoCore Generic Mode Management documentation.

Because the initialization of the `WdgM` is done asynchronously with `WdgM_MainFunction()`, the `WdgM` module is considered to be initialized only after the `WdgM_MainFunction()` is called. If any service of `WdgM` is called before, the module reports the error code `WDGM_E_NO_INIT`.

4.4. Configuring the watchdog stack

To simplify the configuration work, we recommend to start in the lowest layer and move your way up. In this way, you can configure the references between the modules without switching from one configuration job to another.

For details about individual configuration parameters, see the module references chapter.

1. Configure the hardware-dependent Watchdog Driver `Wdg`. For configuration instructions, see the folder `plugins/Wdg_<TargetId>/doc/` in your EB tresos Studio installation tree.
2. Configure the Watchdog Interface `WdgIf`. For configuration instructions, see Section 4.5, “WdgIf module user guide”.
3. Configure the Watchdog Manager `WdgM`. For configuration instructions, see Section 4.6, “WdgM module user guide”.

4.5. WdgIf module user guide

4.5.1. Configuring the WdgIf module

The `WdgIf` module contains a list of references to all `Wdg` modules. The screenshot below shows an example for two watchdog drivers: Wdg and WdgExt.

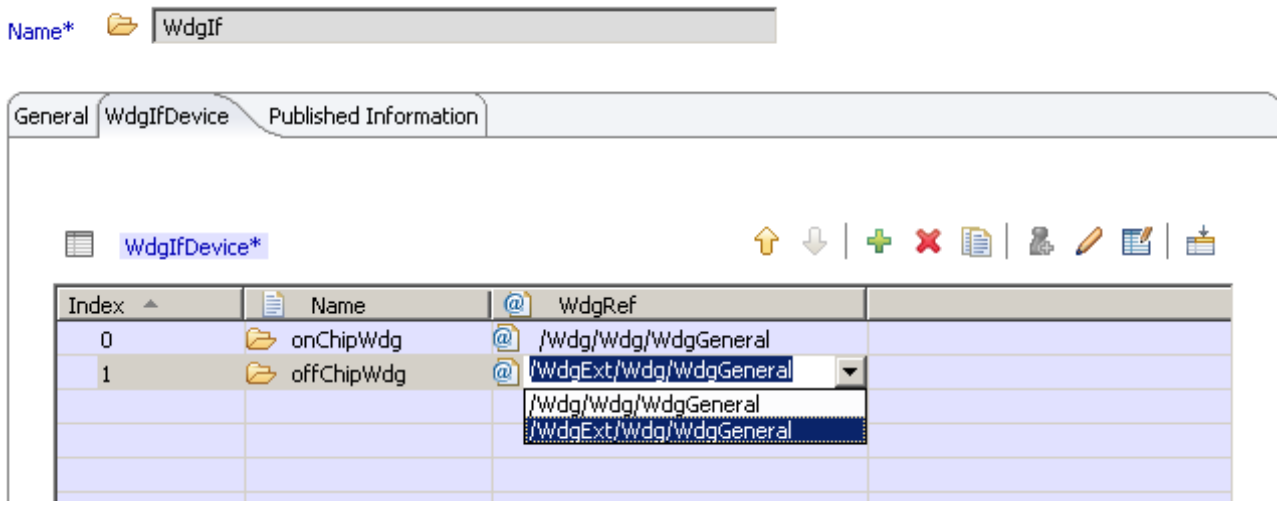

Figure 4.3. Watchdog Interface tab WdgIfDevice

Adding a reference to a watchdog

Step 1
Open the `WdgIf` editor on the `WdgIfDevice` tab.

Step 2

Click the button to add a **WdgIfDevice**.

Step 3
Select the desired watchdog driver from the drop-down list box.

Step 4
Repeat these steps for each watchdog driver.

For configuration parameter details, see the module references chapter.

4.6. WdgM module user guide

4.6.1. Overview

This chapter provides you with `WdgM`-specific information:

- Section 4.6.2, “Background information” explains the concepts of the `WdgM` module.
- Section 4.6.3, “Configuring the WdgM module” provides instructions on how to configure the `WdgM` module.

4.6.2. Background information

The Watchdog Manager (`WdgM`) enables you to supervise how reliable an application is executed. It manages the application’s periodicity, the logical constraints, and its timing constraints.

The supervision of application timing constraints is decoupled from the timing constraints of any underlying hardware watchdogs. For details about application timing constraints, see Section 4.6.2.3, “Alive supervision of supervised entities”. For details about the hardware watchdogs timing constraints, see Section 4.6.2.4, “Triggering the watchdog”.

The hardware watchdogs are accessed via the homogenous `WdgIf` module. The `Wdg` init function is neither abstracted by the `WdgIf` nor by the `WdgM` module.

4.6.2.1. State machines and interfaces

The following illustration shows the state machines of the `WdgM` module and their interfaces.

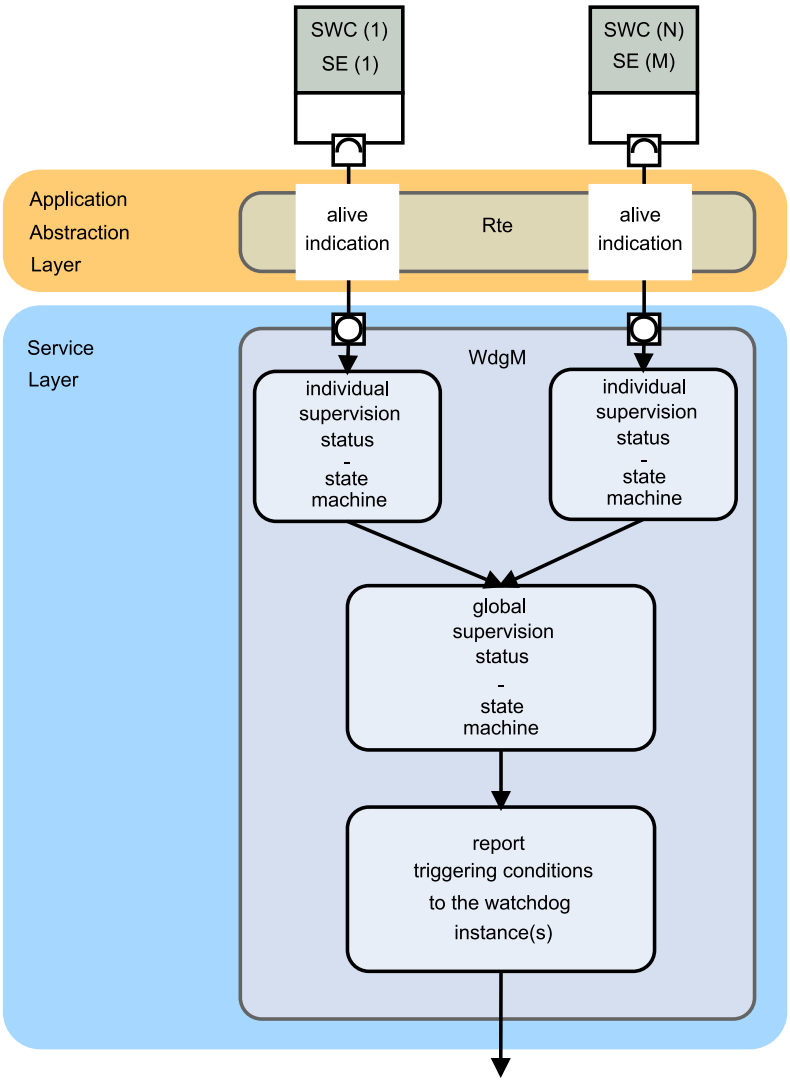

Figure 4.4. Watchdog Manager overview

4.6.2.2. Concepts of supervision

The `WdgM` provides the following concepts of supervision:

- *Supervision of multiple entities:*

  The `WdgM` supervises many entities for execution reliability. The `WdgM` provides an individual port for each supervised entity. At each respective port, the supervised entities have to indicate their aliveness for any configured checkpoint.

- *Supervision of a single entity:*

The `WdgM` supervises a single entity for execution reliability. The `WdgM` provides a port for the supervised entity. There, the supervised entity has to indicate its aliveness for the configured checkpoints.

- *No supervision:*

  No entity is supervised for execution reliability. The proof of execution reliability follows the cyclic scheduling of the `WdgM`.

4.6.2.3. Alive supervision of supervised entities

The alive supervision of supervised entities is optional for the watchdog stack. If there is no supervised entity, the triggering of the watchdog is independent of any software component and simply follows the cyclic scheduling of the `WdgM`.

For details about the watchdog triggering, see Section 4.6.2.4, “Triggering the watchdog”.

If one or more software components are supervised, the triggering of the watchdog follows the state of the *global supervision status* as depicted in Figure 4.4, “Watchdog Manager overview”.

4.6.2.3.1. Supervision status state machines

The `WdgM` handles two different types of state machines:

- Individual supervision status state machine
- Global supervision status state machine

For each supervised entity, there is exactly one *individual supervision status state machine*, which has one of the following states:

- *WDGM_LOCAL_STATUS_OK*

  The supervised entity sent its alive indication correctly.

- *WDGM_LOCAL_STATUS_FAILED*

  The supervised entity has not sent its alive indication correctly but the number of failed alive supervisions has not yet exceeded a configurable boundary.

- *WDGM_LOCAL_STATUS_EXPIRED*

  The supervised entity has not sent its alive indication correctly and the number of failed alive supervisions has been exceeded.

- *WDGM_LOCAL_STATUS_DEACTIVATED*

The alive supervision is not performed on deactivated supervised entities. The global supervision status does not consider deactivated supervised entities.

Besides the individual supervision status state machine(s), there is exactly one *global supervision status state machine*, which depends on the state(s) of the individual supervision status state machine(s). The following states are defined:

- *WDGM_GLOBAL_STATUS_OK*: All activated individual supervision status state machines are in the state WDGM_LOCAL_STATUS_OK.
- *WDGM_GLOBAL_STATUS_FAILED*: One or more activated individual supervision status state machines are in the state WDGM_LOCAL_STATUS_FAILED. There is no individual supervision status state machine that is in the state WDGM_LOCAL_STATUS_EXPIRED.
- *WDGM_GLOBAL_STATUS_EXPIRED*: One or more activated individual supervision status state machines are in the state WDGM_LOCAL_STATUS_EXPIRED, but a tolerance window has not yet been exceeded. The tolerance window is configured with the parameter **WdgMExpiredSupervisionCycleTol**. See Figure 4.13, “Watchdog Manager WdgMConfigSet General tab” for details.
- *WDGM_GLOBAL_STATUS_STOPPED*: One or more activated individual supervision status state machines are in the state WDGM_LOCAL_STATUS_EXPIRED and a tolerance window has been exceeded. The tolerance window is configured with the parameter **WdgMExpiredSupervisionCycleTol**. See Figure 4.13, “Watchdog Manager WdgMConfigSet General tab” for details.

4.6.2.3.2. Effects of mode switching on alive supervision

If `WdgM_SetMode` is called with a new mode that reuses an alive supervision from the previous mode, the alive indications from the previous mode are ignored. This means that, after changing the mode, alive supervision starts from the beginning.

The mode switch can be done asynchronously at the end of the `WdgM_MainFunction` by setting the parameter `WdgMSetModeSynchron` to FALSE. Then, after alive supervision is verified, the following applies:

- If **WdgMSupervisionReferenceCycle** is 1, the verification of alive supervision is always done.
- If **WdgMSupervisionReferenceCycle** is greater than 1, the verification depends on when `WdgM_SetMode` is called. If `WdgM_SetMode` is called between the last two **WdgMSupervisionReferenceCycle**s, the verification is done. Otherwise, verification is ignored.

4.6.2.3.3. Example of the alive supervision

For two examples of alive supervision, see chapter 7.2.3.1 in the AUTOSAR 4.0.3 Specification of Watchdog Manager, V2.2.0.

4.6.2.4. Triggering the watchdog

The `WdgM` module reports the triggering condition of the watchdog instances. The triggering of the watchdog hardware with the right timing is performed by the Watchdog Drivers as long as the reported trigger condition is true. The triggering condition is a counter value that the `WdgM` sets cyclically. This enables the `WdgM` to supervise the program execution abstracted from the triggering of the hardware watchdog entities.

The correct trigger condition is reported cyclically if the global supervision status equals

- WDGM_LOCAL_STATUS_OK,
- WDGM_LOCAL_STATUS_FAILED or
- WDGM_LOCAL_STATUS_EXPIRED.

The watchdog is *not* triggered if the global supervision status equals WDGM_GLOBAL_STATUS_STOPPED.

You may configure timing constraints for each individual watchdog instance within the `WdgM` module. The timing constraints enable:

- cyclical triggering within a maximum time period
- triggering within a defined time window

If the `WdgM` module is configured to execute on multiple cores, only the **WdgM Master Instance** triggers the watchdog drivers. If each core has its own watchdog, the **WdgM Master Instance** must have access to each watchdog that has to be triggered.

The `WdgIf` module provides uniform access to services of the underlying watchdog drivers such as mode switching and triggering. Select the appropriate watchdog driver via a device index as described in Section 4.5, “WdgIf module user guide”.

4.6.2.5. WdgM modes

The `WdgM` module allows to configure different modes as shown in Figure 4.5, “Watchdog Manager modes”. This enables you to configure different supervision and trigger settings for different modes of the watchdog stack. For example, you may use another `WdgM` mode during ECU start-up than during normal operation.

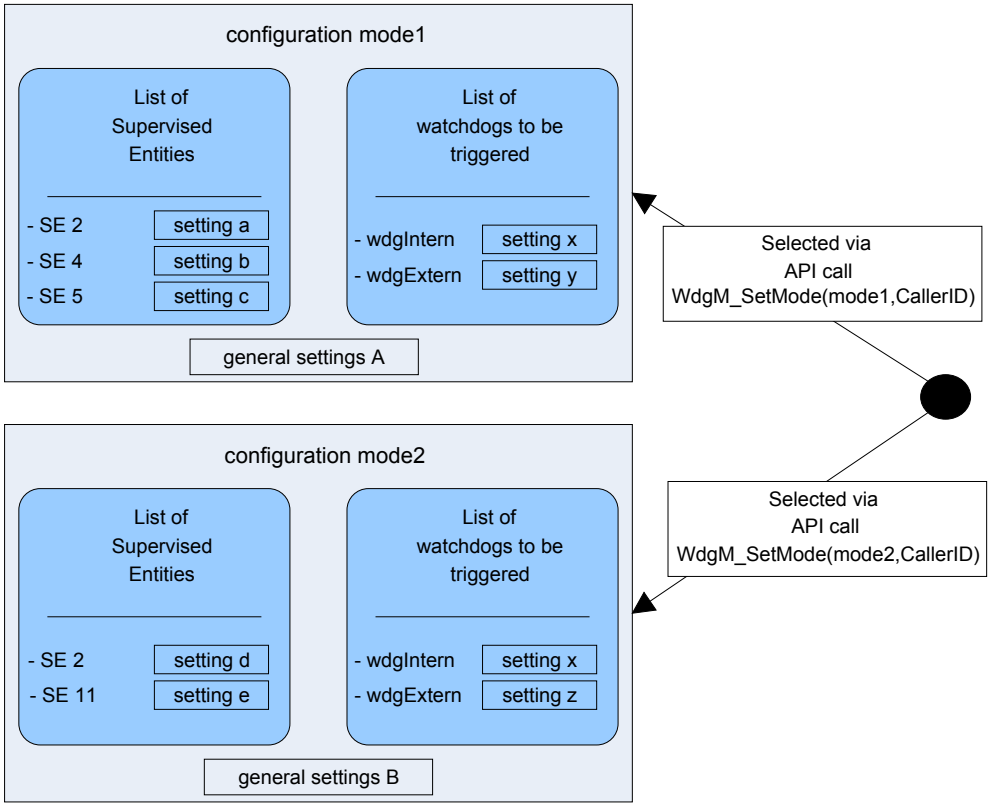

Figure 4.5. Watchdog Manager modes

You can configure a list of supervised entities for each mode. This list of entities is to be supervised for the mode specified. For each supervised entity, it is additionally possible to configure specific settings such as timing parameters or tolerance settings.

Furthermore, you can configure a list of watchdogs that are to be triggered for each mode. Again, you can configure specific settings for each watchdog, such as timing or the mode used.

You can switch between the settings configured during run-time with the API call `WdgM_SetMode()`.

The mode switch can be performed either synchronously with the call or asynchronously at the end of `WdgM_MainFunction`, depending if `WdgMSetModeSynchron` is set to TRUE, respectively set to FALSE.

If multi-core is configured, the mode switch is done as follows:

1. The `WdgM` master instance does a mode switch.
2. If the mode switch was successful, the master instance prompts all `WdgM` satellite instances to change to the new mode.

Because the instances run in parallel, it may be possible that not all satellite instances change the mode at the same time. During the time the satellites change to the new mode, the `WdgM` master instance computes the global supervision status based on the local statuses of the instances that finished the mode switch.

There is a protection mechanism if the satellite instances do not switch to the new mode. By configuring the `WdgMMasterWaitSlaveModeSwitch` parameter, you can specify how long to wait for the satellites to change to the new mode before an error is given. This mechanism is also used when the `WdgM` instances are initialized.

4.6.2.6. Development error detection

If development error detection is enabled, the driver's services perform regular development error checks. A development error is an error that must be detected and fixed during the development phase. It must not occur in the production code.

All development errors are reported to the `Det` module, a central error hook function within the AUTOSAR environment.

If an error occurs, the error hook routine is called and the error code together with the service-ID and the module-ID are passed on as parameters. See the `Det` module user guide in the EB tresos AutoCore Generic Base documentation for details about the `Det` and its API.

If you enabled development error detection, the following development error checks are performed by the services of the `WdgM` module:

- Checks the pointer to configuration data when calling the `WdgM_Init()` function.

  Reports the error code `WDGM_E_PARAM_CONFIG (0x11)` and service-ID `0x00` if the pointer to configuration data is a NULL pointer.
- Checks the parameter `mode` for being within the valid range when calling the `WdgM_SetMode()` function.

  Reports the error code `WDGM_E_PARAM_MODE (0x12)` and service-ID `0x03` if the `mode` exceeds the valid range.
- Checks the parameter `SEID` for being a valid Supervised Entity ID when calling the `WdgM_GetLocalStatus()` function.

  Reports the error code `WDGM_E_PARAM_SEID (0x13)` and service-ID `0x0c` if the `SEID` is not valid.
- Checks the parameter `SEID` for being a valid Supervised Entity ID when calling the `WdgM_UpdateAliveCounter()` function.

  Reports the error code `WDGM_E_PARAM_SEID (0x13)` and service-ID `0x04` if the `SEID` is not valid.
- Checks the parameter `SEID` for being a valid Supervised Entity ID when calling the `WdgM_CheckpointReached()` function.

  Reports the error code `WDGM_E_PARAM_SEID (0x13)` and service-ID `0x0e` if the `SEID` is not valid.

- Checks the parameter `CheckpointID` for being a valid Checkpoint ID when calling the `WdgM_CheckpointReached()` function.

  Reports the error code `WDGM_E_CPID (0x16)` and service-ID `0x0e` if the `CheckpointID` is not valid.
- Reports the error code `WDGM_E_DEPRECATED (0x17)` and service-ID `0x04` when calling the `WdgM_UpdateAliveCounter()` function.
- Checks if in the current mode there are more than one Alive Supervisions (`WdgMAliveSupervision`) configured for the Supervised Entity `SEID` when calling the `WdgM_UpdateAliveCounter()` function.

  Reports the error code `WDGM_E_AMBIGIOUS (0x18)` and service-ID `0x04` when calling the `WdgM_UpdateAliveCounter()` function if in the current mode there are more than one Alive Supervisions (WdgMAliveSupervision) configured.
- Checks the parameter `SEID` for being an activated Supervised Entity in the current watchdog mode when calling the `WdgM_CheckpointReached()` function.

  Reports the error code `WDGM_E_SEDEACTIVATED (0x19)` and service-ID `0x0e` if the Supervised Entity with ID `SEID` is deactivated in the current watchdog mode.
- Checks for unsupported calls of any `WdgM` function except functions `WdgM_GetFirstExpiredSEID()` and `WdgM_Init()`

  Reports the error code `WDGM_E_NO_INIT (0x10)`, if the `WdgM` function is called in any other than the initialized state.
- Checks the parameter `Status` to be a valid pointer when calling the `WdgM_GetLocalStatus()` function.

  Reports the error code `WDGM_E_INV_POINTER (0x14)` and service-ID `0x0c` if the parameter `Status` variable is a NULL pointer.
- Checks the parameter `Status` to be a valid pointer when calling the `WdgM_GetGlobalStatus()` function.

  Reports the error code `WDGM_E_INV_POINTER (0x14)` and service-ID `0x0d` if the parameter `Status` variable is a NULL pointer.
- Checks the parameter `Mode` to be a valid pointer when calling the `WdgM_GetMode()` function.

  Reports the error code `WDGM_E_INV_POINTER (0x14)` and service-ID `0x0b` if the parameter `Mode` variable is a NULL pointer.
- Checks the parameter `SEID` to be a valid pointer when calling the `WdgM_GetFirstExpiredSEID()` function.

  Reports the error code `WDGM_E_INV_POINTER (0x14)` and service-ID `0x10` if the parameter `SEID` variable is a NULL pointer.
- Checks the parameter `VersionInfo` to be a valid pointer when calling the `WdgM_GetVersionInfo()` function.

Reports the error code `WDGM_E_INV_POINTER (0x14)` and service-ID `0x02` if the parameter `VersionInfo` variable is a NULL pointer.

- Checks for invalid disabling of the underlying watchdog drivers when calling the `WdgM_SetMode()` function.

  Reports the error code `WDGM_E_DISABLE_NOT_ALLOWED (0x15)` and service-ID `0x03`, if the parameter `mode` disables the Watchdog Driver (`WDGIF_OFF_MODE`), while the parameter `WdgM_WdgOffModeEnabled` prohibits disabling of underlying watchdog drivers.

- Checks for an invalid core ID calling of a supervised entity checkpoint when calling the `WdgM_CheckpointReached()` function.

  Reports the error code `WDGM_E_PARAM_WRONG_CORE_ID (0x84)` and service-ID `0x0e`, if the parameter `SEID` is mapped to a different core than the one that calls it.

- Checks if the satellite instances switched to the new mode, in the time specified by parameter `WdgMMasterWaitSlaveModeSwitch` after calling `WdgM_SetMode()` with a new mode.

  Reports the error code `WDGM_EB_E_SLAVE_FAILED_CHANGEMODE (0x85)` and service-ID `0x08`, if the satellite did not switch to the new mode in the configured time.

4.6.2.7. Run-time error detection

If the global supervision status of the `WdgM` module reaches the state `WDGM_GLOBAL_STATUS_STOPPED` (i.-e. alive supervision has failed and a reset occurs), the `WdgM` reports an error event to the Diagnostic Error Manager (`Dem`) module with status FAILED. The symbolic name of the event is `WDGM_E_SUPERVISION` and is defined by the `Dem` module.

If a switch mode failure occurs for at least one of the configured watchdog drivers, the `WdgM` reports an error event to the `Diagnostic Error Manager` (`Dem`) with status failed. The symbolic name of the reported error event is `WDGM_E_SET_MODE` and is defined by the `Dem` module.

If an improper caller is detected for the `WdgM_SetMode()` function, the `WdgM` reports the error event `WDGM E_IMPROPER_CALLER` to the `Dem` with status FAILED.

4.6.2.8. Software component description

This chapter describes the data types and interfaces provided by the SWC description of the `WdgM`.

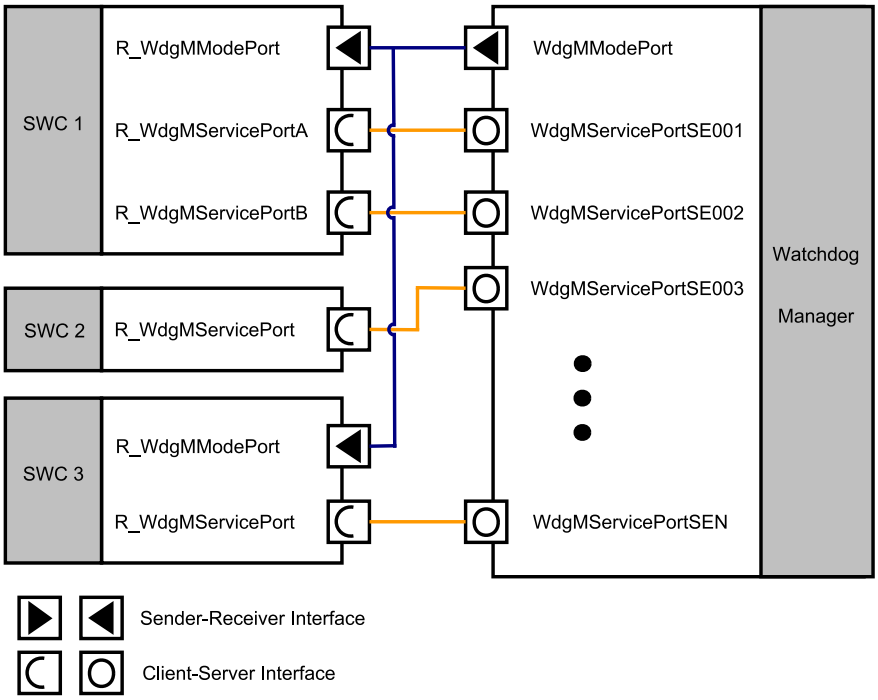

Figure 4.6. Overview of Watchdog Manager ports

4.6.2.8.1. Data types

The following data types are used within the Watchdog Managers software component description.

4.6.2.8.1.1. WdgMModeType

Element:	**ModeDeclarationGroup**
Modes:	SUPERVISION_OK, SUPERVISION_FAILED, SUPERVISION_EXPIRED, SUPERVISION_STOPPED, SUPERVISION_DEACTIVATED
Initial mode:	SUPERVISION_OK
Description:	These modes represent the global Watchdog Manager alive supervision status which is reported to the RTE.

Table 4.1. TS_WDGM_70764

4.6.2.8.2. Ports

4.6.2.8.2.1. WdgMModePort

Interface type:	**ModeSwitchInterface**
Interface name:	WdgM_IndividualMode
Mode declarations group prototypes:	currentMode (data type WdgMModeType)
Description:	This mode port reports the current Local Supervision Status of a single Supervised Entity.

Interface type:	**ModeSwitchInterface**
Interface name:	WdgM_GlobalMode
Mode declarations group prototypes:	currentMode (data type WdgMModeType)
Description:	This mode port reports the current mode of the Watchdog Manager which represents the Global Supervision Status that is combined from all individual Supervised Entities.

4.6.2.8.2.2. WdgMServicePortSE<nnn>

Interface type:	**ClientServerInterface**
Interface name:	WdgM_AliveSupervision
Operation prototypes:	UpdateAliveCounter(ERR/{E_NOT_OK/}) CheckpointReached(IN WdgM_CheckpointIdType, ERR/{E_NOT_OK/})
Description:	This port allows each supervised entity (SE) to trigger its alive counter (CheckpointReached).

4.6.3. Configuring the WdgM module

For information about the `WdgM` module initialization, see Section 4.3, “Initializing the watchdog stack”.

4.6.3.1. Referencing all configured watchdogs

Referencing all watchdogs configured in WdgIf

Step 1
Open the `WdgM` editor on the **WdgMWatchdog** tab.

Step 2

Click the button to add a **WdgMWatchdog** list entry.

Step 3
Select the desired watchdog interface from the drop-down list box. See the following image for a configuration example.

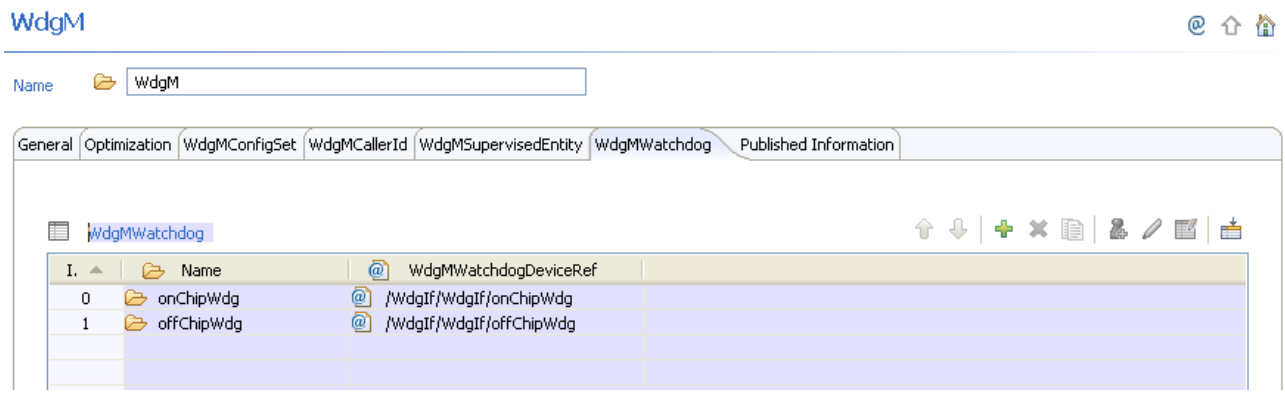

Figure 4.7. Watchdog manager tab WdgMWatchdog

4.6.3.2. Adding all supervised entities

Adding a supervised entity

Step 1
On the **WdgMSupervisedEntity** tab, click the button to add a **WdgMSupervisedEntity** list entry.

The **WdgMSupervisedEntityId** is set automatically when you add a new element.

Step 2
For a multi-core distribution of `WdgM`, configure the **WdgMSupervisedEntityCoreId** with the core ID on which the supervised entity resides. See the following image for a configuration example.

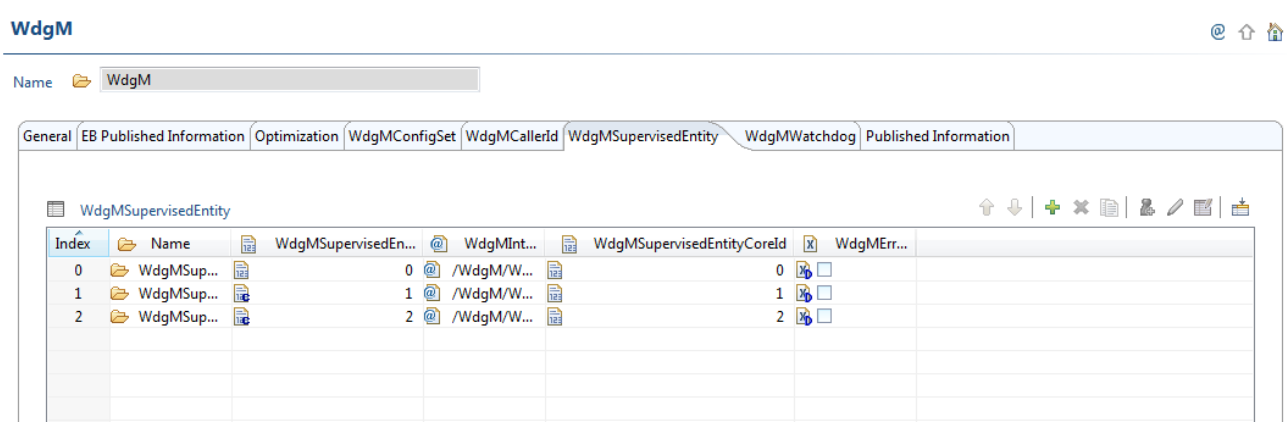

Figure 4.8. Watchdog manager tab WdgMSupervisedEntity

4.6.3.3. Adding the checkpoints for a supervised entity

Adding a checkpoint for a supervised entity

Step 1
On the **WdgMSupervisedEntity** tab, open an entry for editing.

Step 2

On the **WdgMCheckpoint** tab, click the button to add a **WdgMCheckpoint** list entry.

Step 3
Set the **WdgMCheckpointId** to a unique ID in the scope of a supervised entity. See the following image for a configuration example.

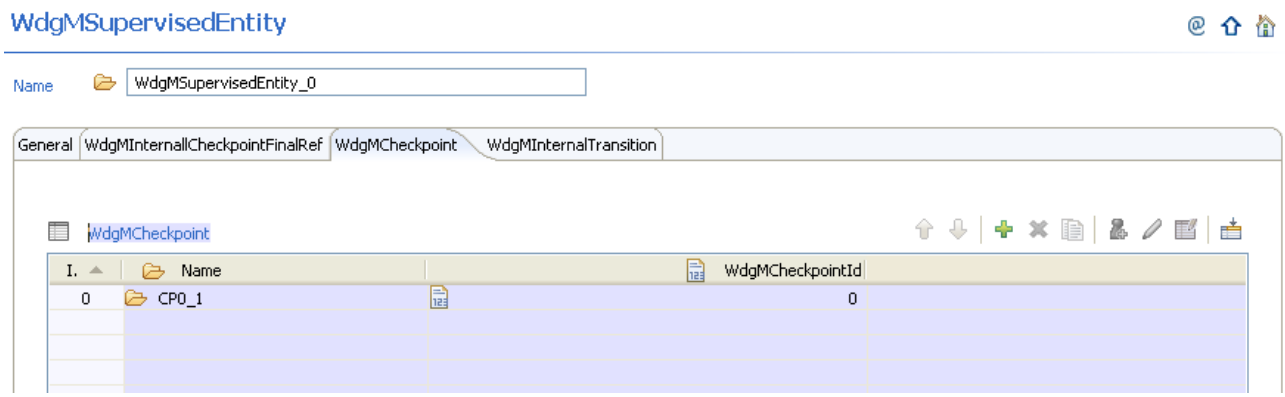

Figure 4.9. Watchdog manager tab WdgMCheckpoint

Step 4
Add more checkpoints as required.

4.6.3.4. Defining the initial and final checkpoints for logical supervision

Adding the initial and final checkpoints

Step 1
Within the **WdgMSupervisedEntity**, go to the **WdgMInternalCheckpointFinalRef** tab.

Step 2
Click the button to add a **WdgMInternalCheckpointFinalRef** list entry.

Step 3
Select the reference to the final checkpoint from the drop-down list box. See the following image for a configuration example.

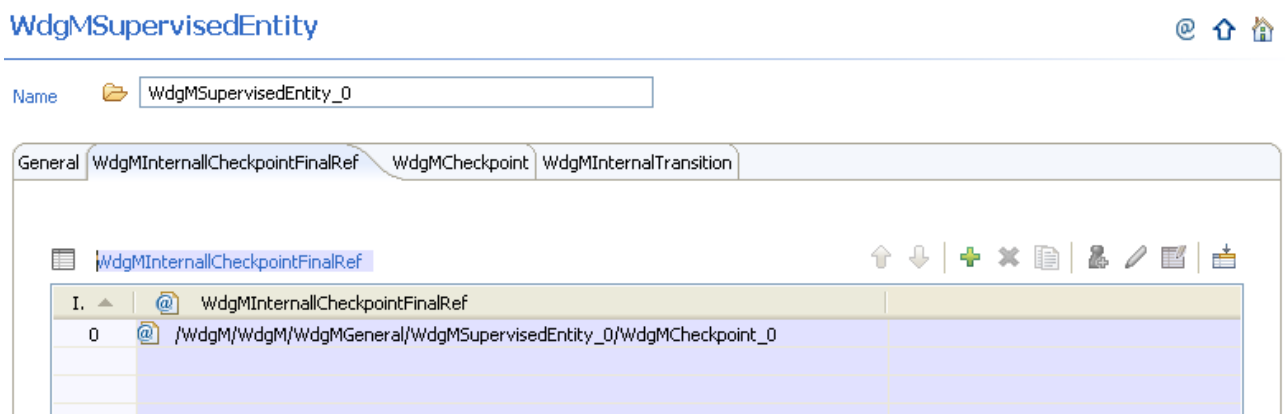

Figure 4.10. Watchdog manager tab WdgMInternalCheckpointFinalRef

Step 4
Switch to the **General** tab.

Step 5
In **WdgMInternalCheckpointInitialRef**, select the initial checkpoint from the drop-down list box.

4.6.3.5. Defining a configuration set

You can define different configuration sets for different modes. For background information, see Section 4.6.2.5, "WdgM modes". You configure the configuration set in the **WdgMConfigSet** container. Only one multiple configuration container is allowed. To configure the multiple configuration container, double-click it.

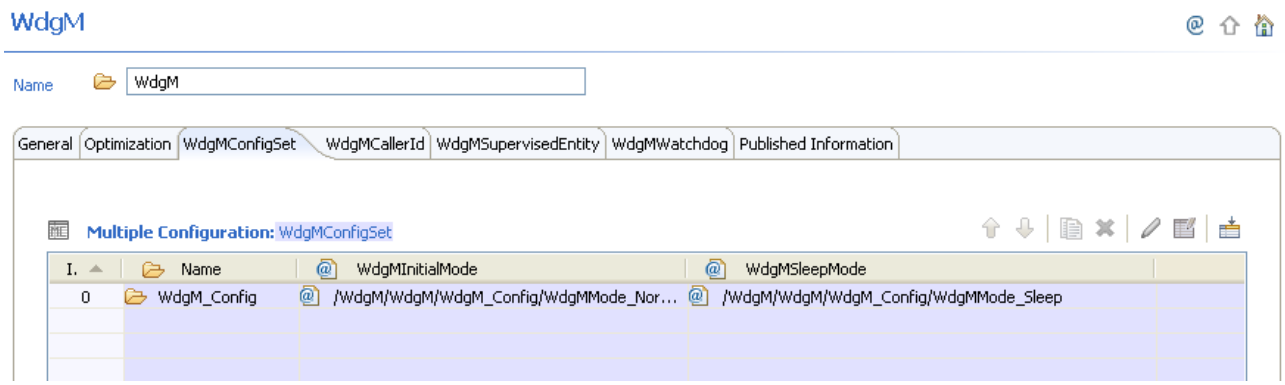

Figure 4.11. Watchdog Manager WdgMConfigSet

Defining a configuration set

Step 1
Open the **WdgMConfigSet** entry for editing.

Step 2
Go to the **WdgMMode** tab.

Step 3

Click on + to add at least one mode.

Step 4
Set the **WdgMModeId** to a unique ID. See the following image for a configuration example.

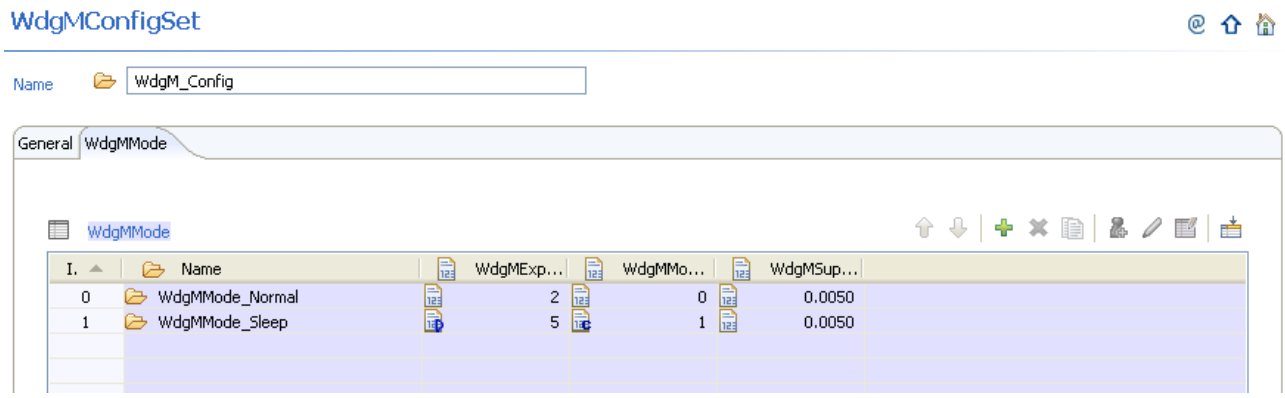

Figure 4.12. Watchdog Manager modes

Step 5
Go to the General tab

Step 6
Set the **WdgMInitialMode** to a configured WdgMMode that the Watchdog Manager is in after it has been initialized.

Step 7
Set the **WdgMSleepMode** to a configured WdgMMode that the Watchdog Manager is in after `WdgM_DeInit()` has been called. See the following image for a configuration example.

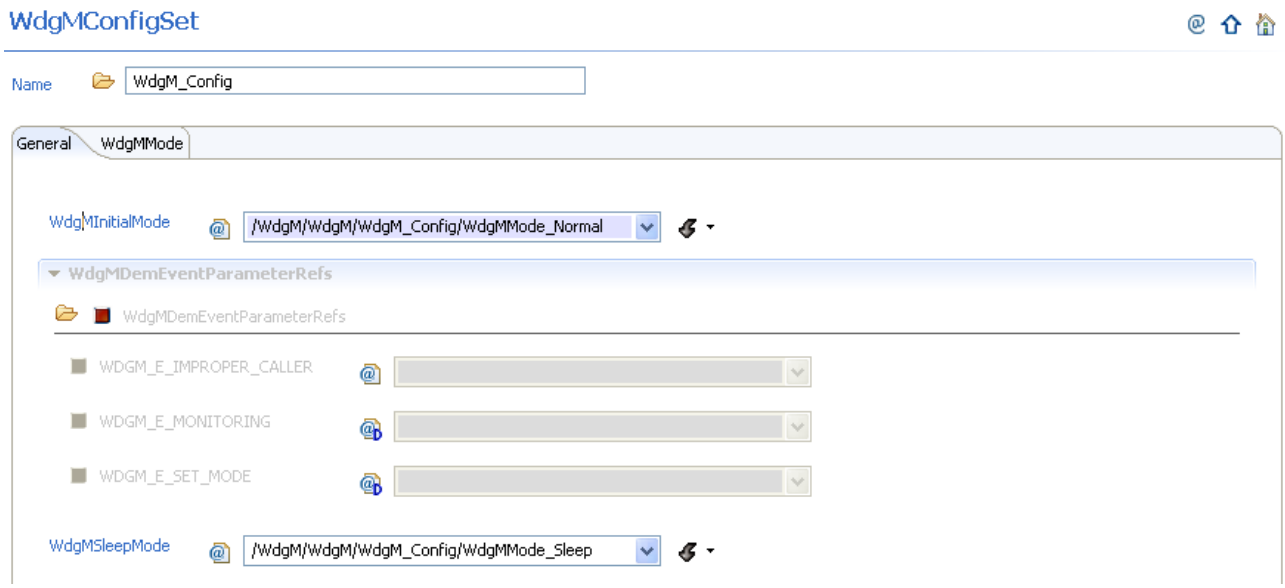

Figure 4.13. Watchdog Manager WdgMConfigSet General tab

4.6.3.6. Configuring a mode

The concept of the different modes is explained in Section 4.6.2.5, “WdgM modes”. For details about each configuration parameter, see the module references chapter.

Configuring a mode

Step 1
On the **WdgMMode** tab, double-click a mode entry.

Step 2
On the **General** tab, configure the general settings for the mode. See the following image for a configuration example.

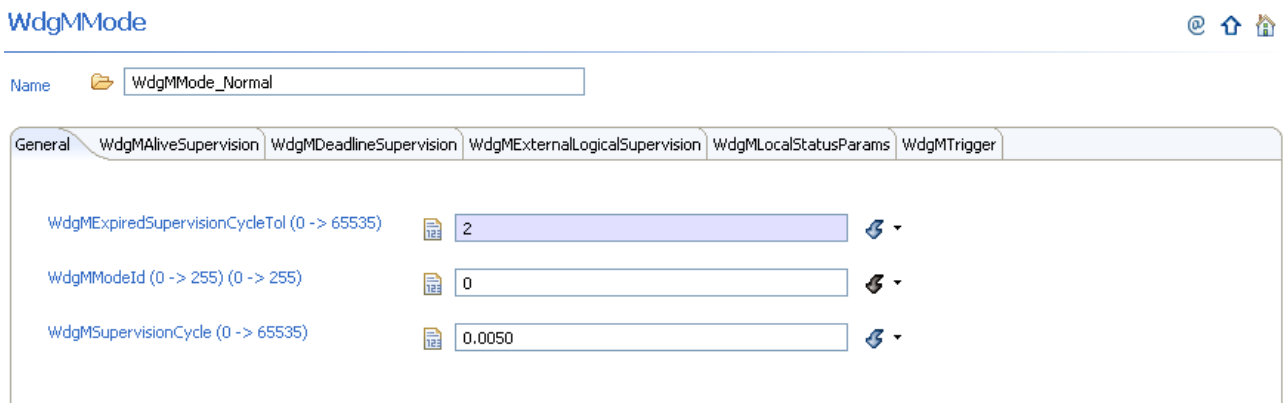

Figure 4.14. Watchdog Manager mode General

Step 3
On the **WdgMAliveSupervision** tab, add all software components that require an alive supervision in this mode.

Step 4
On the **WdgMDeadlineSupervision** tab, add all software components that require a deadline supervision in this mode.

Step 5
On the **WdgMExternalLogicalSupervision** tab, add all software components that require an external logical supervision in this mode.

Step 6
On the **WdgMTrigger** tab, add all watchdog drivers that shall be triggered in this mode.

Step 7

On the **WdgMLocalStatusParams**, click the button to add a **WdgMLocalStatusParams** list entry for each configured supervised entity in this mode, and set the **WdgMLocalStatusSupervisedEntityRef** to this supervised entity.

Step 8
Double-click each **WdgMLocalStatusParams** list entry and set the **WdgMFailedAliveSupervisionRefCycleTol** to configure the acceptable amount of reference cycles with incorrect/failed alive supervisions for the referenced supervised entity.

4.6.3.7. Configuring the WdgM for multi-core

The **General Multicore Configuration Parameters** container is an optional container for the general configuration of `WdgMGeneralMulticore`. This shall be configured only if `WdgM` is used on multiple cores.

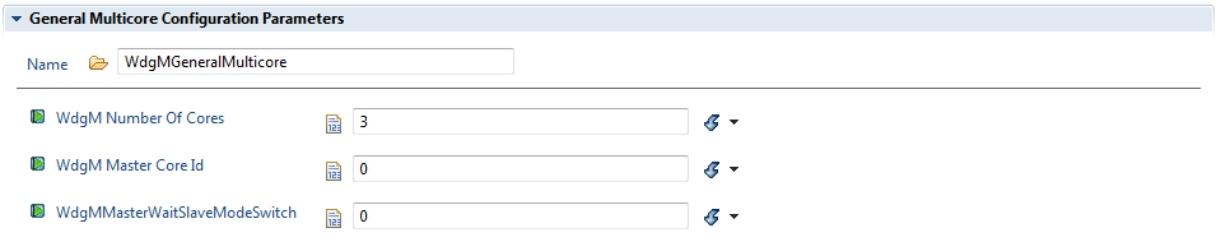

Figure 4.15. General Multi-core Configuration Parameters tab

Configuring the WdgM for multi-core

Step 1
Set the **WdgMNumberOfCores** to a value that specifies on how many cores `WdgM` shall execute.

Step 2
Set the **WdgMMasterCoreId** to a value that corresponds to the core ID on which the master instance shall execute.

Step 3
Set the **WdgMMasterWaitSlaveModeSwitch** to a value that specifies how many main functions the master instance shall wait for mode switch synchronization between cores.

5. ACG8 Watchdog Stack module references

5.1. Overview

This chapter provides module references for the ACG8 Watchdog Stack product modules. These include a detailed description of all configuration parameters. Furthermore this chapter lists the application programming interface with all data types, constants and functions.

The content of the sections is sorted alphabetically according the EB tresos AutoCore Generic module names.

For further information on the functional behavior of these modules, refer to the chapter ACG8 Watchdog Stack user's guide.

5.1.1. Notation in EB module references

EB notation may differ from the AUTOSAR standard notation in the software specification documents (SWS). This section describes the notation of *default value* and *range* fields in the EB module references.

5.1.1.1. Default value of configuration parameters

If there is no default value specified for a parameter, the default value field is omitted to prevent ambiguity with parameters that have `--` as default values.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has no default value field, therefore it is omitted.

5.1.1.2. Range information of configuration parameters

The range of a configuration parameter contains an upper and a lower boundary. However, in special cases the range of allowed values can be computed by means of an XPath function that is evaluated at configuration time. An XPath function can either be a standard `xpath:<function>()` or a custom `cxpath:<function>()` function. The range of a configuration parameter may be computed based on other configuration parameters

that are referenced from the XPath function. For more information on custom XPath functions, see section *Custom XPath Functions API* of the EB tresos Studio developer's guide.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has the custom XPath function `cxpath:getCompuMethodsVT()` in the range field which provides the allowed values.

5.2. WdgIf

5.2.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.
WdgIfDevice	1..255	It contains the information for the selection of a particular Watchdog device in case multiple Watchdog drivers are connected.
WdgIfGeneral	1..1	This container collects all generic watchdog interface parameters.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	**IMPLEMENTATION_CONFIG_VARIANT**
Label	Config Variant
Multiplicity	1..1
Type	ENUMERATION
Default value	VariantPreCompile

Range	VariantPreCompile

5.2.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	**ArMajorVersion**
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	5

Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	6
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version

Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	30
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	43
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.2.1.2. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	PbcfgMSupport	
Label	PbcfgM support	
Description	Specifies whether or not the WdgIf can use the PbcfgM module for post-build support.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PublishedInformation:**	
Origin	Elektrobit Automotive GmbH	

5.2.1.3. WdgIfDevice

Parameters included	
Parameter name	**Multiplicity**
WdgIfDeviceIndex	1..1
WdgIfDriverRef	1..1
WdgIfDrvBswImplementationRef	0..1

Parameter Name	WdgIfDeviceIndex	
Description	Represents the watchdog interface ID so that it can be referenced by the watchdog manager.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	WdgIfDriverRef
Description	Reference to the watchdog drivers that are controlled by the watchdog interface.

Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgIfDrvBswImplementationRef**	
Description	Reference to the BswImplementation of the underlying driver which contains the vendorId and vendorApiInfix.	
Multiplicity	0..1	
Type	FOREIGN-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.2.1.4. WdgIfGeneral

Parameters included	
Parameter name	**Multiplicity**
WdgIfDevErrorDetect	1..1
WdgIfVersionInfoApi	1..1
WdgIfDriverAPIInfixEnable	1..1

Parameter Name	**WdgIfDevErrorDetect**	
Description	Pre-processor switch for enabling the development error detection and reporting. true: Development error detection enabled false: Development error detection disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgIfVersionInfoApi**
Description	Pre-processor switch to enable / disable the service returning the version information.

	true: Version information service enabled false: Version information service disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgIfDriverAPIInfixEnable**	
Description	This parameter defines if WdgIf shall use the Vendor Id and the API Infix for accessing the Wdg Driver module in case a single Wdg driver is configured. true: WdgIf uses the Vendor Id and the API Infix of the Wdg Driver for accessing the Driver API (e.g. Wdg_1_driver) in case only a single Wdg driver is used. In addtion this name mangling is also used for including the Wdg Driver header file (e.g. Wdg_1_driver.h) false: WdgIf does not use the Vendor Id and the API Infix of the Wdg Driver in case only a single Wdg driver is used. Note: If more than one Wdg driver is configured, name mangling must be used.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile

5.2.2. Application programming interface (API)

5.2.2.1. Type definitions

5.2.2.1.1. WdgIf_ModeType

Purpose	Mode type of the WdgIf module.
Type	`uint8`

Description	This uint8 type holds the mode types that are passed as parameters to Wdg_SetMode().

5.2.2.2. Macro constants

5.2.2.2.1. WDGIF_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.2.2.2.2. WDGIF_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	0U

5.2.2.2.3. WDGIF_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	3U

5.2.2.2.4. WDGIF_E_INV_POINTER

Purpose	DET: API service called with null pointer parameter.
Value	0x02U

5.2.2.2.5. WDGIF_E_PARAM_DEVICE

Purpose	DET: API service called with wrong device index parameter.
Value	0x01U

5.2.2.2.6. WDGIF_FAST_MODE

Purpose	In this mode, the watchdog driver is set up for a short timeout period (fast triggering).

Value	2u

5.2.2.2.7. WDGIF_MODULE_ID

Purpose	AUTOSAR module identification.
Value	43U

5.2.2.2.8. WDGIF_OFF_MODE

Purpose	In this mode, the watchdog driver is disabled (switched off).
Value	0u

5.2.2.2.9. WDGIF_SID_GETVERSIONINFO

Purpose	API service id for WdgIf_GetVersionInfo().
Value	0x03U

5.2.2.2.10. WDGIF_SID_SETMODE

Purpose	API service id for WdgIf_SetMode().
Value	0x01U

5.2.2.2.11. WDGIF_SID_SETTRIGGERCOND

Purpose	API service id for WdgIf_SetTriggerCondition().
Value	0x02U

5.2.2.2.12. WDGIF_SLOW_MODE

Purpose	In this mode, the watchdog driver is set up for a long timeout period (slow triggering).

Value	1u

5.2.2.2.13. WDGIF_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	6U

5.2.2.2.14. WDGIF_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	1U

5.2.2.2.15. WDGIF_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	30U

5.2.2.2.16. WDGIF_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.2.2.3. Functions

5.2.2.3.1. WdgIf_GetVersionInfo

Purpose	Get version information of the Watchdog Interface.
Synopsis	`void WdgIf_GetVersionInfo ( Std_VersionInfoType *const VersionInfoPtr );`
Service ID	WDGIF_SID_GETVERSIONINFO
Sync/Async	Synchronous
Reentrancy	Reentrant

Parameters (out)	`VersionInfoPtr`	Pointer to where to store the version information of this module
Description	This service returns the version information of this module.	

5.2.2.3.2. WdgIf_SetMode

Purpose	Set mode of the Watchdog driver.	
Synopsis	`Std_ReturnType WdgIf_SetMode ( uint8 DeviceIndex , WdgIf_ModeType WdgMode );`	
Service ID	WDGIF_SID_SETMODE	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Parameters (in)	`DeviceIndex`	index of requested watchdog driver
	`WdgMode`	requested mode of watchdog driver
Parameters (in,out)	`DeviceIndex`	index of requested watchdog driver
	`WdgMode`	requested mode of watchdog driver
Return Value	Std_ReturnType	
Description	This function provides access to the mode switching services of the underlying watchdog drivers. The function is mapped to the service Wdg_SetMode(). This function is implemented as macro if development error detection is turned off.	

5.2.2.3.3. WdgIf_SetTriggerCondition

Purpose	Trigger the Watchdog driver.	
Synopsis	`void WdgIf_SetTriggerCondition ( uint8 DeviceIndex , uint16 Timeout );`	
Service ID	WDGIF_SID_SETTRIGGERCOND	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Parameters (in)	`DeviceIndex`	Identifies the Watchdog Driver instance.
	`Timeout`	Timeout value (milliseconds) for setting the trigger counter.

Description	This service maps the service WdgIf_SetTriggerCondition to the service Wdg_SetTriggerCondition of the corresponding Watchdog Driver. This function is implemented as macro if development error detection is turned off.

5.2.3. Integration notes

5.2.3.1. Exclusive areas

Exclusive areas are not used by the `WdgIf` module.

5.2.3.2. Production errors

Production errors are not reported by the `WdgIf` module.

5.2.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CODE_ASIL_D
CONST_ASIL_D_UNSPECIFIED

5.2.3.4. Integration requirements

WARNING

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

Integration requirements are not listed for the WdgIf module.

5.3. WdgM

5.3.1. Configuration parameters

Containers included		
Container name	**Multiplicity**	**Description**
CommonPublishedInformation	1..1	**Label:** Common Published Information Common container, aggregated by all modules. It contains published information about vendor and versions.
PublishedInformation	1..1	**Label:** EB Published Information Additional published parameters not covered by CommonPublishedInformation container.
WdgMDefensiveProgramming	1..1	**Label:** Defensive Programming Options Parameters for defensive programming
ReportToDem	1..1	**Label:** Production error handling Production error handling
WdgMConfigSet	1..n	This container describes one of multiple configuration sets of WdgM. This is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
WdgMGeneral	1..1	**Label:** General Configuration Parameters Container defines all general configuration parameters of the Watchdog Manager.

Parameters included	
Parameter name	**Multiplicity**
IMPLEMENTATION_CONFIG_VARIANT	1..1

Parameter Name	**IMPLEMENTATION_CONFIG_VARIANT**
Label	Config Variant
Multiplicity	1..1
Type	ENUMERATION
Default value	VariantPreCompile

Range	VariantPreCompile

5.3.1.1. CommonPublishedInformation

Parameters included	
Parameter name	**Multiplicity**
ArMajorVersion	1..1
ArMinorVersion	1..1
ArPatchVersion	1..1
SwMajorVersion	1..1
SwMinorVersion	1..1
SwPatchVersion	1..1
ModuleId	1..1
VendorId	1..1
Release	1..1

Parameter Name	**ArMajorVersion**
Label	AUTOSAR Major Version
Description	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArMinorVersion**
Label	AUTOSAR Minor Version
Description	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	2

Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ArPatchVersion**
Label	AUTOSAR Patch Version
Description	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	0
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMajorVersion**
Label	Software Major Version
Description	Major version number of the vendor specific implementation of the module.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	6
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwMinorVersion**
Label	Software Minor Version
Description	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**SwPatchVersion**
Label	Software Patch Version

Description	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	43
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**ModuleId**
Label	Numeric Module ID
Description	Module ID of this module from Module List
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	13
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**VendorId**
Label	Vendor ID
Description	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
Multiplicity	1..1
Type	INTEGER_LABEL
Default value	1
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

Parameter Name	**Release**
Label	Release Information
Multiplicity	1..1
Type	STRING_LABEL
Default value	
Configuration class	**PublishedInformation:**

Origin	Elektrobit Automotive GmbH

5.3.1.2. PublishedInformation

Parameters included	
Parameter name	**Multiplicity**
PbcfgMSupport	1..1

Parameter Name	**PbcfgMSupport**
Label	PbcfgM support
Description	Specifies whether or not the WdgM can use the PbcfgM module for post-build support.
Multiplicity	1..1
Type	BOOLEAN
Default value	false
Configuration class	**PublishedInformation:**
Origin	Elektrobit Automotive GmbH

5.3.1.3. WdgMDefensiveProgramming

Parameters included	
Parameter name	**Multiplicity**
WdgMDefProgEnabled	1..1
WdgMPrecondAssertEnabled	1..1
WdgMPostcondAssertEnabled	1..1
WdgMStaticAssertEnabled	1..1
WdgMUnreachAssertEnabled	1..1
WdgMInvariantAssertEnabled	1..1

Parameter Name	**WdgMDefProgEnabled**
Label	Enable Defensive Programming
Description	Enables or disables the defensive programming feature for the module WdgM. Note: This feature is dependent on the use of the development error detection module. To use the defensive programming feature, proceed as follows:

	1. Enable development error detection 2. Enable defensive programming 3. Enable assertions as required	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMPrecondAssertEnabled**	
Label	Enable Precondition Assertions	
Description	Enables handling of precondition assertion checks reported from the module WdgM. Dependency on parameter(s): ► Enable Development Error Detection (`WdgMDevErrorDetect`): must be enabled ► Enable Defensive Programming (`WdgMDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMPostcondAssertEnabled**
Label	Enable Postcondition Assertions
Description	Enables handling of postcondition assertion checks reported from the module WdgM. Dependency on parameter(s): ► Enable Development Error Detection (`WdgMDevErrorDetect`): must be enabled ► Enable Defensive Programming (`WdgMDefProgEnabled`): must be enabled

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMStaticAssertEnabled**	
Label	Enable Static Assertions	
Description	Enables handling of static assertion checks reported from the module WdgM. Dependency on parameter(s): ► Enable Development Error Detection (`WdgMDevErrorDetect`): must be enabled ► Enable Defensive Programming (`WdgMDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMUnreachAssertEnabled**	
Label	Enable Unreachable Code Assertions	
Description	Enables handling of unreachable code assertion checks reported from the module WdgM. Dependency on parameter(s): ► Enable Development Error Detection (`WdgMDevErrorDetect`): must be enabled ► Enable Defensive Programming (`WdgMDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Origin	Elektrobit Automotive GmbH

Parameter Name	**WdgMInvariantAssertEnabled**	
Label	Enable Invariant Assertions	
Description	Enables handling of invariant assertion checks reported from functions of the module WdgM. Dependency on parameter(s): ► Enable Development Error Detection (`WdgMDevErrorDetect`): must be enabled ► Enable Defensive Programming (`WdgMDefProgEnabled`): must be enabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.4. ReportToDem

Parameters included	
Parameter name	**Multiplicity**
WdgMSupervisionReportToDem	1..1
WdgMSupervisionDemDetErrId	1..1
WdgMSetModeReportToDem	1..1
WdgMSetModeDemDetErrId	1..1
WdgMImproperCallerReportToDem	1..1
WdgMImproperCallerDemDetErrId	1..1
WdgMMFTimingViolationReportToDem	1..1
WdgMMFTimingViolationDemDetErrId	1..1
WdgMDataCorruptionReportToDem	1..1
WdgMDataCorruptionDemDetErrId	1..1

Parameter Name	**WdgMSupervisionReportToDem**
Label	WdgM Supervision Failure

Description	Selects the handling of the production error: *WDGM_E_MONITORING (formerly WDGM_E_SUPERVISION)* ► `DEM`: The production error WDGM_E_MONITORING is reported to the Diagnostics Event Manager (Dem). ► `DET`: The production error WDGM_E_MONITORING is reported to the Development Error Tracer (Det) if enabled. ► `DISABLE`: The production error WDGM_E_MONITORING is not reported at all.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	DEM	
Range	DEM DET DISABLE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMSupervisionDemDetErrId**	
Label	WdgM Supervision Failure DemToDet ErrorId	
Description	If a production error is reported towards the Det, this parameter defines the error id which is reported towards the Det. The preprocessor define WDGM_E_DEMTODET_E_SUPERVISION is generated holding the value of `WdgMSupervisionDemDetErrId`.	
Multiplicity	1..1	
Type	INTEGER	
Default value	30	
Range	<=255 >=30	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMSetModeReportToDem**
Label	WdgM SetMode Failure
Description	Selects the handling of the production error: *WDGM_E_SET_MODE*

	▶ `DEM`: The production error WDGM_E_SET_MODE is reported to the Diagnostics Event Manager (Dem). ▶ `DET`: The production error WDGM_E_SET_MODE is reported to the Development Error Tracer (Det) if enabled. ▶ `DISABLE`: The production error WDGM_E_SET_MODE is not reported at all.
Multiplicity	1..1
Type	ENUMERATION
Default value	DEM
Range	DEM DET DISABLE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**WdgMSetModeDemDetErrId**
Label	WdgM SetMode Failure DemToDet ErrorId
Description	If a production error is reported towards the Det, this parameter defines the error id which is reported towards the Det. The preprocessor define WDGM_EB_E_DEMTODET_SET_MODE is generated holding the value of `WdgMSetModeDemDetErrId`.
Multiplicity	1..1
Type	INTEGER
Default value	30
Range	<=255 >=30
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**WdgMImproperCallerReportToDem**
Label	WdgM ImproperCaller Failure
Description	Selects the handling of the production error: *WDGM_E_IMPROPER_CALLER* ▶ `DEM`: The production error WDGM_E_IMPROPER_CALLER is reported to the Diagnostics Event Manager (Dem).

	▶ `DET`: The production error WDGM_E_IMPROPER_CALLER is reported to the Development Error Tracer (Det) if enabled. ▶ `DISABLE`: The production error WDGM_E_IMPROPER_CALLER is not reported at all.
Multiplicity	1..1
Type	ENUMERATION
Default value	DEM
Range	DEM DET DISABLE
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**WdgMImproperCallerDemDetErrId**
Label	WdgM ImproperCaller Failure DemToDet ErrorId
Description	If a production error is reported towards the Det, this parameter defines the error id which is reported towards the Det. The preprocessor define WDGM_EB_E_DEMTODET_IMPROPER_CALLER is generated holding the value of `WdgMImproperCallerDemDetErrId`.
Multiplicity	1..1
Type	INTEGER
Default value	30
Range	<=255 >=30
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**WdgMMFTimingViolationReportToDem**
Label	WdgM MainFunction Timing Failure
Description	Selects the handling of the production error: *WDGM_E_MF_TIMINGVIOLATION* ▶ `DEM`: The production error WDGM_E_MF_TIMINGVIOLATION is reported to the Diagnostics Event Manager (Dem). ▶ `DET`: The production error WDGM_E_MF_TIMINGVIOLATION is reported to the Development Error Tracer (Det) if enabled.

	▶ `DISABLE`: The production error WDGM_E_MF_TIMINGVIOLATION is not reported at all.
Multiplicity	1..1
Type	ENUMERATION
Default value	DEM
Range	DEM DET DISABLE
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**WdgMMFTimingViolationDemDetErrId**
Label	WdgM MainFunction Timing Failure DemToDet ErrorId
Description	If a production error is reported towards the Det, this parameter defines the error id which is reported towards the Det. The preprocessor define WDGM_E_DEMTODET_E_MF_TIMINGVIOLATION is generated holding the value of `WdgMMFTimingViolationDemDetErrId`.
Multiplicity	1..1
Type	INTEGER
Default value	30
Range	<=255 >=30
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**WdgMDataCorruptionReportToDem**
Label	WdgM Data Corruption Failure
Description	Selects the handling of the production error: *WDGM_E_DATA_CORRUPTION* ▶ `DEM`: The production error WDGM_E_DATA_CORRUPTION is reported to the Diagnostics Event Manager (Dem). ▶ `DET`: The production error WDGM_E_DATA_CORRUPTION is reported to the Development Error Tracer (Det) if enabled. ▶ `DISABLE`: The production error WDGM_E_DATA_CORRUPTION is not reported at all.

Multiplicity	1..1	
Type	ENUMERATION	
Default value	DEM	
Range	DEM	
	DET	
	DISABLE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMDataCorruptionDemDetErrId**	
Label	WdgM Data Corruption Failure DemToDet ErrorId	
Description	If a production error is reported towards the Det, this parameter defines the error id which is reported towards the Det. The preprocessor define WDGM_E_DEMTODET_E_DATA_CORRUPTION is generated holding the value of `WdgMDataCorruptionDemDetErrId`.	
Multiplicity	1..1	
Type	INTEGER	
Default value	30	
Range	<=255	
	>=30	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.5. WdgMConfigSet

Containers included		
Container name	**Multiplicity**	**Description**
WdgMDemEventParameter-Refs	1..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.

Containers included		
WdgMMode	1..255	The container describes one of several modes of the Watchdog Manager.

Parameters included	
Parameter name	**Multiplicity**
WdgMInitialMode	1..1
WdgMSleepMode	0..1

Parameter Name	**WdgMInitialMode**	
Description	The mode that the Watchdog Manager is in after it has been initialized.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMSleepMode**	
Description	The Watchdog Manager switches to this WdgM mode at the beginning of WdgM_DeInit and executes the configured trigger conditions before finally deactivating the Watchdog Manager. This final mode switch usually has no Supervision Entities configured and may be necessary for a relaxed Watchdog triggering during the shutdown or sleep phase. If parameter is disabled, this final mode switch has to be performed externally via the WdgM_SetMode() API and at least one main function has to be executed afterwards such that the Watchdog Manager calls the required trigger conditions before finally executing WdgM_DeInit.	
Multiplicity	0..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.6. WdgMDemEventParameterRefs

Parameters included	
Parameter name	**Multiplicity**
WDGM_E_DATA_CORRUPTION	1..1
WDGM_E_IMPROPER_CALLER	1..1

Parameters included	
WDGM_E_MF_TIMINGVIOLATION	1..1
WDGM_E_MONITORING	1..1
WDGM_E_SET_MODE	1..1

Parameter Name	WDGM_E_DATA_CORRUPTION	
Description	Reference to the DemEventParameter that shall be issued when data corruption is detected in the internal WdgM data, which is stored double inverse. Dependency on parameter(s): ► `WdgMDataCorruptionReportToDem`: Select DEM to enable the reporting of `WDGM_E_DATA_CORRUPTION`. Further notes: ► Activation: Thrown, if data corruption is detected in the internal WdgM data. ► Healing: None. The error resides in memory until it is deleted. ► Trigger debounce: None. The error is reported on first occurrence. ► Rate of diagnostic checks: Checked on every call of the service that reports this error.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	WDGM_E_IMPROPER_CALLER
Description	Reference to the DemEventParameter that shall be issued when the defensive behavior checks detect an improper caller. Dependency on parameter(s): ► `WdgMImproperCallerReportToDem`: Select DEM to enable the reporting of `WDGM_E_IMPROPER_CALLER`. Further notes: ► Activation: Thrown, if the passed CallerID is not in the list of the configured list of allowed CallerIDs. ► Healing: None. The error resides in memory until it is deleted. ► Trigger debounce: None. The error is reported on first occurrence.

	► Rate of diagnostic checks: Checked on every call of the service that reports this error.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WDGM_E_MF_TIMINGVIOLATION**	
Description	Reference to the DemEventParameter that shall be issued when the actual Watchdog Manager main function period deviates from the configured mode-dependent schedule period (`WdgMSupervisionCycle`). Dependency on parameter(s): ► `WdgMMFTimingViolationReportToDem`: Select DEM to enable the reporting of `WDGM_E_MF_TIMINGVIOLATION`. Further notes: ► Activation: Thrown, if the `WdgM_MainFunction()` period deviates from the configured mode-dependent schedule period (`WdgMSupervisionCycle`). ► Healing: None. The error resides in memory until it is deleted. ► Trigger debounce: None. The error is reported on first occurrence. ► Rate of diagnostic checks: Checked cyclically within `WdgM_MainFunction()`.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WDGM_E_MONITORING**
Description	Reference to the DemEventParameter that shall be issued when the following error occurs: *Monitoring has failed and a watchdog reset will occur.* Dependency on parameter(s): ► `WdgMSupervisionReportToDem`: Select DEM to enable the reporting of `WDGM_E_MONITORING`. Further notes:

	▶ Activation: Thrown, if supervision fails for a supervised entity. ▶ Healing: None. The error resides in memory until it is deleted. ▶ Trigger debounce: None. The error is reported on first occurrence. ▶ Rate of diagnostic checks: Checked cyclically within `WdgM_MainFunction()`.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

Parameter Name	**WDGM_E_SET_MODE**
Description	Reference to the DemEventParameter that shall be issued when the following error occurs: *Watchdog drivers' mode switch has failed.* Dependency on parameter(s): ▶ `WdgMSetModeReportToDem`: Select DEM to enable the reporting of `WDGM_E_SET_MODE`. Further notes: ▶ Activation: Thrown, if watchdog drivers' mode switch has failed. ▶ Healing: None. The error resides in memory until it is deleted. ▶ Trigger debounce: None. The error is reported on first occurrence. ▶ Rate of diagnostic checks: Checked cyclically within `WdgM_MainFunction()`.
Multiplicity	1..1
Type	SYMBOLIC-NAME-REFERENCE
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	AUTOSAR_ECUC

5.3.1.7. WdgMMode

Containers included		
Container name	**Multiplicity**	**Description**
WdgMAliveSupervision	0..65535	This container collects all configuration parameters of Alive-Supervision of one Checkpoint. Note that each Checkpoint

Containers included		
		may have different parameters. For example, it may have different min and max margin.
WdgMDeadlineSupervision	0..65535	This container collects all configuration parameters for Deadline Supervision for a Supervised Entity.
WdgMExternalLogicalSupervision	0..65535	This container collects all configuration parameters for Logical Supervision for one external graph.
WdgMLocalStatusParams	0..65535	This container collects all configuration parameters for the Local Status of a Supervised Entity.
WdgMTrigger	0..255	This container collects all configuration parameters for the triggering of hardware watchdogs.

Parameters included	
Parameter name	**Multiplicity**
WdgMExpiredSupervisionCycleTol	1..1
WdgMModeId	1..1
WdgMSupervisionCycle	1..1

Parameter Name	**WdgMExpiredSupervisionCycleTol**	
Description	This parameter shall be used to define a value that fixes the amount of expired supervision cycles for how long the blocking of watchdog triggering shall be postponed, AFTER THE GLOBAL SUPERVISION STATUS HAS REACHED THE STATE EXPIRED.	
Multiplicity	1..1	
Type	INTEGER	
Default value	5	
Range	<=65535	
	>=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMModeId**
Label	WdgMModeId (0 -> 255)
Description	This parameter fixes the identfier for the mode. This identifier is for instance passed as a parameter to the WdgM_SetMode service.
Multiplicity	1..1

Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMSupervisionCycle**	
Description	This parameter defines the schedule period of the main function WdgM_Main-Function. Unit: [s]	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.01	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.3.1.8. WdgMAliveSupervision

Parameters included	
Parameter name	**Multiplicity**
WdgMExpectedAliveIndications	1..1
WdgMMaxMargin	1..1
WdgMMinMargin	1..1
WdgMSupervisionReferenceCycle	1..1
WdgMAliveSupervisionCheckpointRef	1..1

Parameter Name	**WdgMExpectedAliveIndications**	
Description	This parameter contains the amount of expected alive indications of the Check-point within the referenced amount of defined supervision cycles according to corresponding SE.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Range	<=65535 >=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Origin	AUTOSAR_ECUC

Parameter Name	**WdgMMaxMargin**	
Label	WdgMMaxMargin (0 -> 255)	
Description	This parameter contains the amount of alive indications of the Checkpoint that are acceptable to be additional to the expected alive indications within the corresponding supervision reference cycle.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMMinMargin**	
Label	WdgMMinMargin (0 -> 255)	
Description	This parameter contains the amount of alive indications of the Checkpoint that are acceptable to be missed from the expected alive indications within the corresponding supervision reference cycle.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMSupervisionReferenceCycle**	
Description	This parameter shall contain the amount of supervision cycles to be used as reference by the alive-supervision mechanism to perform the checkup with counted alive indications according to corresponding SE.	
Multiplicity	1..1	
Type	INTEGER	
Default value	1	
Range	<=65535 >=1	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	WdgMAliveSupervisionCheckpointRef	
Description	Reference to Checkpoint within a Supervised Entity that shall be supervised.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.3.1.9. WdgMDeadlineSupervision

Parameters included	
Parameter name	**Multiplicity**
WdgMDeadlineMax	1..1
WdgMDeadlineMin	1..1
WdgMDeadlineStartRef	1..1
WdgMDeadlineStopRef	1..1

Parameter Name	WdgMDeadlineMax	
Description	This parameter contains the longest time span after which the deadline is considered to be met. Unit: [s]	
Multiplicity	1..1	
Type	FLOAT	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	WdgMDeadlineMin	
Description	This parameter contains the shortest time span after which the deadline is considered to be met. Unit: [s]	
Multiplicity	1..1	
Type	FLOAT	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	WdgMDeadlineStartRef
Description	This is the reference to the start Checkpoint for Deadline Supervision.

Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMDeadlineStopRef**	
Description	This is the reference to the stop Checkpoint for Deadline Supervision.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.3.1.10. WdgMExternalLogicalSupervision

Containers included		
Container name	**Multiplicity**	**Description**
WdgMExternalTransition	0..65535	This container collects the Checkpoints for an External Transition across Supervised Entities.

Parameters included	
Parameter name	**Multiplicity**
WdgMExternalCheckpointFinalRef	1..65535
WdgMExternalCheckpointInitialRef	1..65535

Parameter Name	**WdgMExternalCheckpointFinalRef**	
Description	This is the reference to the final Checkpoint(s) for this External Graph.	
Multiplicity	1..65535	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMExternalCheckpointInitialRef**
Description	This is the reference to the initial Checkpoint(s) for this External Graph.
Multiplicity	1..65535

Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.3.1.11. WdgMExternalTransition

Parameters included	
Parameter name	**Multiplicity**
WdgMExternalTransitionDestRef	1..1
WdgMExternalTransitionSourceRef	1..1

Parameter Name	**WdgMExternalTransitionDestRef**	
Description	This is the reference to the destination Checkpoint of an External Transition.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMExternalTransitionSourceRef**	
Description	This is the reference to the source Checkpoint of an External Transition.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.3.1.12. WdgMLocalStatusParams

Parameters included	
Parameter name	**Multiplicity**
WdgMFailedAliveSupervisionRefCycleTol	1..1
WdgMLocalStatusSupervisedEntityRef	1..1

Parameter Name	**WdgMFailedAliveSupervisionRefCycleTol**

Description	This parameter shall contain the acceptable amount of reference cycles with incorrect/failed alive supervisions for this Supervised Entity.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=255	
	>=0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMLocalStatusSupervisedEntityRef**	
Description	This is the reference to the Supervised Entity for which the Local Status parameters are specified.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.3.1.13. WdgMTrigger

Parameters included	
Parameter name	**Multiplicity**
WdgMTriggerConditionValue	1..1
WdgMWatchdogMode	1..1
WdgMTriggerWatchdogRef	1..1

Parameter Name	**WdgMTriggerConditionValue**	
Description	This parameter shall contain the value that is passed to WdgIf_SetTriggerCondition for this watchdog.	
Multiplicity	1..1	
Type	INTEGER	
Range	<=65535	
	>=1	
Configuration class	**VariantPreCompile:**	VariantPreCompile

Origin	AUTOSAR_ECUC

Parameter Name	**WdgMWatchdogMode**	
Description	This parameter contains the watchdog mode that shall be used for the referenced watchdog in this Watchdog Manager mode. Implementation Type: WdgIf_ModeType	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	WDGIF_FAST_MODE	
Range	WDGIF_FAST_MODE	
	WDGIF_OFF_MODE	
	WDGIF_SLOW_MODE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMTriggerWatchdogRef**	
Description	This parameter is a reference to the configured watchdog.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.3.1.14. WdgMGeneral

Containers included		
Container name	**Multiplicity**	**Description**
WdgMCallerIds	0..1	Contains the definition of valid CallerIds for the callers who have permission to call the function WdgM_SetMode.
WdgMGeneralMulticore	1..1	**Label:** General Multicore Configuration Parameters
WdgMServiceAPI	1..1	**Label:** Service API Parameters Container for configuration of the service API of WdgM.
WdgMSupervisedEntity	1..65535	This container collects all common (mode-independent) parameters of a Supervised Entity to be supervised by the Watchdog Manager.

Containers included		
WdgMSupervisorCallouts	0..1	Enables the configuration of callouts for WdgM APIs and integration of a trusted component (e.g. &Supervisor;).
WdgMWatchdog	0..255	This container collects all common (mode-independent) parameters of a Watchdog to be triggered by the Watchdog Manager.

Parameters included	
Parameter name	**Multiplicity**
WdgMDefensiveBehavior	1..1
WdgMDemStoppedSupervisionReport	1..1
WdgMDevErrorDetect	1..1
WdgMImmediateReset	1..1
WdgMOffModeEnabled	1..1
WdgMVersionInfoApi	1..1
WdgMSetModeSynchron	1..1
WdgMRteUsage	1..1
WdgMGetAllExpiredSEIDs	1..1
WdgMBSWCompatibilityMode	1..1
WdgMPartitioningEnabled	1..1
WdgMMixedSafetyCriticalityEnabled	1..1

Parameter Name	WdgMDefensiveBehavior	
Description	Preprocessor switch to enable/disable the defensive behavior of the Watchdog Manager module.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	WdgMDemStoppedSupervisionReport
Description	Parameter to enable/disable the error reporting to DEM. ► `true`: A notification to DEM is sent if the Watchdog Manager reaches the state WDGM_GLOBAL_STATUS_STOPPED.

	► `false`: The notification is disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMDevErrorDetect**	
Description	Preprocessor switch to enable/disable development error detection and reporting. Shall be used to remove unneeded code segments regarding DET features ► `true`: Development error detection is enabled ► `false`: Development error detection is disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMImmediateReset**	
Description	This parameter enables/disablse the immediate reset feature in case of alive-supervision failure. ► `true`: Immediate reset is enabled ► `false`: Immediate reset is disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMOffModeEnabled**
Description	This parameter enables/disables the selection of the "OffMode" of the watchdog driver.

	► `true`: "OffMode" selection is allowed ► `false`: "OffMode" selection is disallowed	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMVersionInfoApi**	
Description	Preprocessor switch to enable/disable the existence of the API WdgM_GetVersionInfo. Shall be used to remove unneeded code segments. ► `true`: API is enabled ► `false`: API is disabled	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMSetModeSynchron**	
Description	This parameter enable WdgM_SetMode synchronously switch to the new mode. This behavior is available for the case when the Supervisor Proxy is not used and multicore is disabled.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMRteUsage**
Description	This parameter enables the usage of the RTE for this module. For an easy integration it is recommended to disable the usage of the RTE at the beginning of the integration work.

Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMGetAllExpiredSEIDs**	
Description	This parameter allows the user to retrieve all the supervised entities that have expired.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMBSWCompatibilityMode**
Description	Configures whether the AUTOSAR BSW APIs shall be compatible to AUTOSAR 3.2, AUTOSAR 4.0 or AUTOSAR 4.3. ► `AUTOSAR_32` = AUTOSAR 3.2 BSW APIs are provided (e.g. WdgM_SetMode without CallerId) ► `AUTOSAR_40` = AUTOSAR 4.0 BSW APIs are provided (e.g. WdgM_SetMode with CallerId) ► `AUTOSAR_43` = AUTOSAR 4.3 BSW APIs are provided (e.g. WdgM_SetMode without CallerId)
Multiplicity	1..1
Type	ENUMERATION
Default value	AUTOSAR_40
Range	AUTOSAR_32 AUTOSAR_40 AUTOSAR_43
Configuration class	**VariantPreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

Parameter Name	**WdgMPartitioningEnabled**

Description	This parameter defines whether the environment makes use of partitioning. In this case, the WdgM provides a separate non-AUTOSAR conform memory abstraction which must be configured by the integrator to be read/write accessible from the execution context of all other partitions / Software Components. ► `Enabled`: WdgM runtime data which must be written from different callers outside the execution context of the WdgM_Mainfunction (e.g. within the execution context of a Software Component due to the call to WdgM_CheckpointReached) is allocated to a separate non-AUTOSAR conform memory abstraction identifer. ► `Disabled`: Complete WdgM runtime data are allocated to AUTOSAR conform memory abstraction identifiers.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMMixedSafetyCriticalityEnabled**	
Description	This parameter defines whether the environment makes use of different safety level for cores. In this case, the WdgM provides a separate memory abstraction which must be configured by the integrator to be read/write accessible from the execution context of all cores. The master core must always have the highest safety level required by the system safety level and the satellites cores can have different safety levels depending on the needs of the system.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.15. WdgMCallerIds

Parameters included	
Parameter name	**Multiplicity**
WdgMCallerId	0..255

Parameter Name	WdgMCallerId	
Description	This parameter defines one valid CallerId for the callers who have permission to call the function WdgM_SetMode.	
Multiplicity	0..255	
Type	INTEGER	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.3.1.16. WdgMGeneralMulticore

Parameters included	
Parameter name	**Multiplicity**
WdgMNumberOfCores	0..1
WdgMMasterCoreId	0..1
WdgMMasterWaitSlaveModeSwitch	0..1
WdgMGetFirstExpiredEnable	1..1

Parameter Name	WdgMNumberOfCores	
Label	WdgM Number Of Cores	
Description	This parameter defines the maximum number of cores on which WdgM is distributed.	
Multiplicity	0..1	
Type	INTEGER	
Default value	1	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	WdgMMasterCoreId
Label	WdgM Master Core Id
Description	This parameter maps the master instance of WdgM to a specific Os Core ID.
Multiplicity	0..1
Type	INTEGER
Default value	0

Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMMasterWaitSlaveModeSwitch**	
Description	This parameter defines the amount of time WdgM Master Instance shall wait for WdgM Satellite Instances to finish the mode switch, until it will check to see if the mode switch was succesfully from the time the master changed to the new mode(initialization is considered a particular mode switch operation to WdgMInitialMode). If this parameter is set to zero, the check is disabled. In case the mode swicth was not done succefully WDGM_EB_E_SLAVE_FAILED_CHANGEMODE runtime error will be reported. Unit: Number of WdgM Master Instance main functions.	
Multiplicity	0..1	
Type	INTEGER	
Default value	0	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMGetFirstExpiredEnable**	
Description	Parameter to enable/disable the API WdgM_GetFirstExpiredSEID(). Note: When multicore is enabled WdgM_GetFirstExpiredSEID() may not be fully reliable.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	true	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.17. WdgMServiceAPI

Parameters included	
Parameter name	**Multiplicity**
WdgMEnableASR32ServiceAPI	1..1
WdgMEnableASR40ServiceAPI	1..1
WdgMEnableASR43ServiceAPI	1..1

Parameters included	
WdgMDefaultASRServiceAPI	1..1
WdgMEnableASR32ActivateAliveSupervisionAPI	0..1
WdgMEnableASR32DeactivateAliveSupervisionAPI	0..1

Parameter Name	WdgMEnableASR32ServiceAPI	
Label	Enable AUTOSAR 3.2 service API	
Description	Configures whether the AUTOSAR 3.2 service API shall be provided. ► `TRUE` = Enables AUTOSAR 3.2 service API. ► `FALSE` = Disables AUTOSAR 3.2 service API.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	WdgMEnableASR40ServiceAPI	
Label	Enable AUTOSAR 4.0 service API	
Description	Configures whether the AUTOSAR 4.0 service API shall be provided. ► `TRUE` = Enables AUTOSAR 4.0 service API. ► `FALSE` = Disables AUTOSAR 4.0 service API.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	WdgMEnableASR43ServiceAPI
Label	Enable AUTOSAR 4.3 service API
Description	Configures whether the AUTOSAR 4.3 service API shall be provided. ► `TRUE` = Enables AUTOSAR 4.3 service API. ► `FALSE` = Disables AUTOSAR 4.3 service API.
Multiplicity	1..1

Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMDefaultASRServiceAPI**	
Label	Default AUTOSAR service API	
Description	Defines the default AUTOSAR service API. ► `AUTOSAR_32` = AUTOSAR 3.2 service API is the default one. ► `AUTOSAR_40` = AUTOSAR 4.0 service API is the default one. ► `AUTOSAR_43` = AUTOSAR 4.3 service API is the default one. ► `NONE` = No default AUTOSAR service API is provided.	
Multiplicity	1..1	
Type	ENUMERATION	
Default value	AUTOSAR_40	
Range	AUTOSAR_32 AUTOSAR_40 AUTOSAR_43 NONE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMEnableASR32ActivateAliveSupervisionAPI**
Label	Enable AUTOSAR 3.2 service API ActivateAliveSupervision
Description	Configures whether the AUTOSAR 3.2 service API ActivateAliveSupervision shall be provided for sake of compatibility with existing ASR32 Software Components. ► `Enabled` = The WdgM re-directs the call to WdgM_ActivateAliveSupervision to an externally implemented API with the API name specified within this parameter. (Note: makes only sense if either parameter Default Service API is set to AUTOSAR_32 or generation of ASR32 Service API is enabled.) ► `Disabled` = The WdgM Service Component does not support / provide this service API.

Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMEnableASR32DeactivateAliveSupervisionAPI**
Label	Enable AUTOSAR 3.2 service API DeactivateAliveSupervision
Description	Configures whether the AUTOSAR 3.2 service API DeactivateAliveSupervision shall be provided for sake of compatibility with existing ASR32 Software Components. ► `Enabled` = The WdgM re-directs the call to WdgM_DeactivateAliveSupervision to an externally implemented API with the API name specified within this parameter. (Note: makes only sense if either parameter Default Service API is set to AUTOSAR_32 or generation of ASR32 Service API is enabled.) ► `Disabled` = The WdgM Service Component does not support / provide this service API.
Multiplicity	0..1
Type	STRING
Default value	
Configuration class	**PreCompile:** VariantPreCompile
Origin	Elektrobit Automotive GmbH

5.3.1.18. WdgMSupervisedEntity

Containers included		
Container name	**Multiplicity**	**Description**
WdgMCheckpoint	1..65535	This container collects all Checkpoints of this Supervised Entity. Each Supervised Entity has at least one Checkpoint.
WdgMInternalTransition	0..65535	This container defines the graph of Internal Transitions within this Supervised Entity.

Parameters included	
Parameter name	**Multiplicity**

Parameters included	
WdgMSupervisedEntityId	1..1
WdgMEcucPartitionRef	0..1
WdgMOsApplicationRef	0..1
WdgMInternalCheckpointInitialRef	1..1
WdgMInternalICheckpointFinalRef	1..65535
WdgMSupervisedEntityCoreId	1..1
WdgMErrorRecoveryEnabled	1..1

Parameter Name	WdgMSupervisedEntityId	
Label	WdgMSupervisedEntityId (0 -> 65535)	
Description	This parameter shall contain the unique identifier of the supervised entity.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	WdgMEcucPartitionRef	
Description	Denotes in which "EcucPartition" the supervised entity is executed. When the partition is stopped, the supervised entity shall be de-activated in the WdgM to avoid an ECU reset.	
Multiplicity	0..1	
Type	REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	WdgMOsApplicationRef	
Description	Optional reference to an OS Application. Beware, the Watchdog Manager module will trigger a partition restart of this OS Application when the corresponding Supervised Entity reaches WDGM_LOCAL_STATUS_FAILED.	
Multiplicity	0..1	
Type	REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	WdgMInternalCheckpointInitialRef	
Description	This is the reference to the initial Checkpoint for this Supervised Entity.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	WdgMInternalCheckpointFinalRef	
Description	This is the reference to the final Checkpoint(s) for this Supervised Entity.	
Multiplicity	1..65535	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	WdgMSupervisedEntityCoreId	
Label	WdgMSupervisedEntityCoreId	
Description	This parameter maps to which core the SupervisedEntity belongs to.	
Multiplicity	1..1	
Type	INTEGER	
Default value	0	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	WdgMErrorRecoveryEnabled	
Description	If set to true, then the Supervised Entity never enters the local Supervision Status `WDGM_LOCAL_STATUS_EXPIRED` (i.e. independent of the configured kind of supervision, in case of a detected incorrect supervision, it remains in status `WDGM_LOCAL_STATUS_FAILED` until all supervisions succeed again). If set to false, then the Supervised Entity behaves according to AUTOSAR.	
Multiplicity	1..1	
Type	BOOLEAN	
Default value	false	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.19. WdgMCheckpoint

Parameters included	
Parameter name	**Multiplicity**
WdgMCheckpointId	1..1

Parameter Name	**WdgMCheckpointId**	
Description	This parameter shall contain the unique identifier of Checkpoint.	
Multiplicity	1..1	
Type	INTEGER	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.3.1.20. WdgMInternalTransition

Parameters included	
Parameter name	**Multiplicity**
WdgMInternalTransitionDestRef	1..1
WdgMInternalTransitionSourceRef	1..1

Parameter Name	**WdgMInternalTransitionDestRef**	
Description	This is the reference to the destination Checkpoint of a Internal Transition within this Supervised Entity.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMInternalTransitionSourceRef**	
Description	This is the reference to the source Checkpoint of a Internal Transition within this Supervised Entity.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

5.3.1.21. WdgMSupervisorCallouts

Parameters included	
Parameter name	**Multiplicity**
WdgMGetExpectedInitStateCallout	0..1
WdgMInitRedirectionCallout	1..1
WdgMDeInitRedirectionCallout	1..1
WdgMGetExpectedWdgMModeCallout	0..1
WdgMSetModeRedirectionCallout	1..1
WdgMGetElapsedTimeCallout	0..1
WdgMTimeGranularity	1..1
WdgMMainFunctionPeriodTolerance	1..1
WdgMIsPerformResetCallout	0..1
WdgMSupervisionExpiredCallout	0..1
WdgMIndividualModeSwitchCallout	0..1
WdgMGlobalModeSwitchCallout	0..1
WdgMDetCallout	0..1
WdgMGetCoreIdCallout	0..1
WdgMMainFunctionViolationCallout	0..1
WdgMRequestPartitionResetCallout	0..1
WdgMGetApplicationStateCallout	0..1

Parameter Name	**WdgMGetExpectedInitStateCallout**	
Description	Defines the implemented API name for polling a desired initialization state. This API is usually implemented in the Supervisor and shall have the following syntax: Std_ReturnType [ConfiguredAPIName](WdgM_EB_InitStatusType* InitStatus)	
Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMInitRedirectionCallout**

Description	This parameter is only used if parameter WdgMGetExpectedInitStateCallout is enabled. The WdgM redirects each call to BSW API WdgM_Init to the configured callout API. If this parameter is left empty, then the BSW API WdgM_Init is not provided and no redirection is performed.	
Multiplicity	1..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMDeInitRedirectionCallout**	
Description	This parameter is only used if parameter WdgMGetExpectedInitStateCallout is enabled. The WdgM redirects each call to BSW API WdgM_DeInit to the configured callout API. If this parameter is left empty, then the BSW API WdgM_DeInit is not provided and no redirection is performed.	
Multiplicity	1..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMGetExpectedWdgMModeCallout**	
Description	Description: Defines the implemented API name for polling the expected WdgM-Mode. This API is usually implemented in the Supervisor and shall have the following syntax: Std_ReturnType [ConfiguredAPIName](WdgM_ModeType* Mode)	
Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMSetModeRedirectionCallout**
Description	This parameter is only used if parameter WdgMGetExpectedWdgMModeCallout is enabled. The WdgM redirects each call to BSW API WdgM_SetMode to the

	configured callout API. If this parameter is left empty, then the BSW API WdgM_-SetMode is not provided and no redirection is performed.	
Multiplicity	1..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMGetElapsedTimeCallout**	
Description	Defines the implemented API name for getting the information regarding the elapsed time. The time units of the parameters are expected to be in real-time with the granularity provided in parameter WdgMTimeGranularity. This API is usually implemented in a safe OS and shall have the following syntax: void [ConfiguredAPIName](uint32* PreviousTime, uint32* ElapsedTime)	
Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMTimeGranularity**	
Description	This parameter defines the granularity in real-time between two consecutive units of the time parameters used in the GetElapsedTime API specified via parameter WdgMGetElapsedTimeCallout. Unit: [s]	
Multiplicity	1..1	
Type	FLOAT	
Default value	0.0001	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMMainFunctionPeriodTolerance**
Description	This parameter defines the allowed tolerance (plus / minus) in real-time between two main function calls with respect to the configured schedule time of parameter WdgMSupervisionCycle. Unit: [s]
Multiplicity	1..1

Type	FLOAT	
Default value	0.0001	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMIsPerformResetCallout**	
Description	Defines the implemented API name for getting the authorization of a requested Watchdog reset. This API is usually implemented in the Supervisor and shall have the following syntax: Std_ReturnType [ConfiguredAPIName](void)	
Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMSupervisionExpiredCallout**	
Description	Defines the implemented API name for indicating the Supervisor about a Supervision failure and getting the information for a possible re-initialization. This API is usually implemented in the Supervisor and shall have the following syntax: void [ConfiguredAPIName](WdgM_SupervisedEntityIdType FirstExpiredSEID)	
Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMIndividualModeSwitchCallout**
Description	Defines the implemented API name for indicating the Supervisor about a mode switch of a Supervised Entity (e.g. from WDGM_LOCAL_STATUS_OK to WDGM_LOCAL_STATUS_FAILED, or from WDGM_LOCAL_STATUS_FAILED to WDGM_LOCAL_STATUS_DEACTIVATED, etc.). This API is usually implemented in the Supervisor and shall have the following syntax: void [ConfiguredAPIName](WdgM_SupervisedEntityIdType SEID, WdgM_ModeType OldMode, WdgM_ModeType NewMode)

Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMGlobalModeSwitchCallout**	
Description	Defines the implemented API name for indicating the Supervisor about a global mode switch of the WdgM (e.g. from WDGM_GLOBAL_STATUS_OK to WDGM_GLOBAL_STATUS_FAILED, or from WDGM_GLOBAL_STATUS_-FAILED to WDGM_GLOBAL_STATUS_EXPIRED, etc.). This API is usually implemented in the Supervisor and shall have the following syntax: void [ConfiguredAPIName](WdgM_ModeType OldMode, WdgM_ModeType NewMode)	
Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMDetCallout**	
Description	Defines the implemented API name for indicating the Supervisor about an internal DET error. This API shall have the following syntax: void [ConfiguredAPIName](uint8 ApiID, uint8 ErrorID)	
Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMGetCoreIdCallout**
Description	Defines the implemented API name for retrieving the core id information required for Temporal Program Flow Monitoring and Logical Program Flow Monitoring. This API shall have the following syntax: uint16 [ConfiguredAPIName](void)

Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMMainFunctionViolationCallout**	
Description	Defines the implemented API name for indicating the supervisor about a violation of main function schedule time. This API shall have the following syntax: void [ConfiguredAPIName](void)	
Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMRequestPartitionResetCallout**	
Description	Defines the implemented API name for requesting a restart/shutdown of the corresponding partition. This API shall have the following syntax: Std_ReturnType [ConfiguredAPIName](ApplicationType Application)	
Multiplicity	0..1	
Type	STRING	
Default value		
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMGetApplicationStateCallout**
Description	Defines the implemented API name for retrieving the current state of an OS-Application. This API shall have the following syntax: Std_ReturnType [ConfiguredAPIName](ApplicationType Application, ApplicationStateRefType Value)
Multiplicity	0..1
Type	STRING
Default value	

Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.1.22. WdgMWatchdog

Parameters included	
Parameter name	**Multiplicity**
WdgMWatchdogName	1..1
WdgMWatchdogDeviceRef	1..1
WdgMMulticoreWdgEnable	1..1
WdgMMulticoreWdgCoreId	0..1

Parameter Name	**WdgMWatchdogName**	
Description	This parameter shall contain the symbolic name of the watchdog instance.	
Multiplicity	1..1	
Type	STRING	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMWatchdogDeviceRef**	
Description	Reference to one device container of Watchdog Interface. In the referenced container WdgIfDevice, the parameter WdgIfDeviceIndex contains the Index parameter that WdgM has to use for WdgIf_SetTriggerCondition calls for that watchdog instance.	
Multiplicity	1..1	
Type	SYMBOLIC-NAME-REFERENCE	
Configuration class	**VariantPreCompile:**	VariantPreCompile
Origin	AUTOSAR_ECUC	

Parameter Name	**WdgMMulticoreWdgEnable**
Label	Trigger Wdg driver from all cores
Description	This parameter defines if the referenced watchdog driver is triggered form all the cores, when multicore is enable.
Multiplicity	1..1

Type	BOOLEAN	
Default value	false	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

Parameter Name	**WdgMMulticoreWdgCoreId**	
Label	Core Id - trigger Wdg driver	
Description	This parameter defines the core instance which will trigger the referenced watchdog driver.	
Multiplicity	0..1	
Type	INTEGER	
Default value	0	
Configuration class	**PreCompile:**	VariantPreCompile
Origin	Elektrobit Automotive GmbH	

5.3.2. Application programming interface (API)

5.3.2.1. Type definitions

5.3.2.1.1. WdgM_CheckpointIdType

Purpose	Checkpoint Id Type.
Type	`uint16`
Description	This type identifies a Checkpoint in the context of a Supervised Entity for the Watchdog Manager. Note that an individual Checkpoint can only be identified by the pair of Supervised Entity ID and Checkpoint ID.

5.3.2.1.2. WdgM_ConfigType

Purpose	configuration of WdgM
Type	`uint8`

Description	This is a dummy structure to hold configuration parameters. As the WdgM is not able to be link or post build time configurable this structure is not used. It is defined for compatibility only. A pointer to this structure is passed to the Watchdog Manager initialization function for configuration.

5.3.2.1.3. WdgM_EB_CoreIdType

Purpose	Holding the executing Core ID.
Type	`uint16`

5.3.2.1.4. WdgM_EB_InitStatusType

Purpose	Init status.
Type	`uint8`
Description	This type is used to define the expected / actual initialization status. The two possible states are: WDGM_EB_INIT_STATUS_INIT ..... WdgM is / shall be initialized WDGM_EB_INIT_STATUS_DEINIT ... WdgM is / shall be de-initialized

5.3.2.1.5. WdgM_GlobalStatusType

Purpose	Global Status Type.
Type	`uint8`
Description	This type is used to represent the global supervision status of the Watchdog Manager.

5.3.2.1.6. WdgM_LocalStatusType

Purpose	Local Status Type.
Type	`uint8`
Description	This type is used to represent the local supervision status of the individual supervised entities.

5.3.2.1.7. WdgM_ModeType

Purpose	Mode Type.

Type	`uint8`
Description	This type distinguishes the different modes that were configured for the Watchdog Manager.

5.3.2.1.8. WdgM_SupervisedEntityIdType

Purpose	Supervised Entity Id Type.
Type	`uint16`
Description	This type identifies an individual Supervised Entity for the Watchdog Manager.

5.3.2.2. Macro constants

5.3.2.2.1. WDGM_AR_RELEASE_MAJOR_VERSION

Purpose	AUTOSAR release major version.
Value	4U

5.3.2.2.2. WDGM_AR_RELEASE_MINOR_VERSION

Purpose	AUTOSAR release minor version.
Value	0U

5.3.2.2.3. WDGM_AR_RELEASE_REVISION_VERSION

Purpose	AUTOSAR release revision version.
Value	3U

5.3.2.2.4. WDGM_EB_E_DEINIT_REQUEST

Purpose	Development error: WdgM de-initialization request failed.

Value	0x81U

5.3.2.2.5. WDGM_EB_E_INIT_REQUEST

Purpose	Development error: WdgM initialization request failed.
Value	0x80U

5.3.2.2.6. WDGM_EB_E_REENTRANCY

Purpose	Development error: non-reentrant WdgM main function is executed in parallel.
Value	0x83U

5.3.2.2.7. WDGM_EB_E_SETMODE_REQUEST

Purpose	Development error: WdgM mode change request failed.
Value	0x82U

5.3.2.2.8. WDGM_EB_E_SLAVE_FAILED_CHANGEMODE

Purpose	Development error: Satellite instance failed to change to the new mode in the time given by parameter WdgMMasterWaitSlaveModeSwitch. Initialization is considered a particular mode switch operation to WdgMInitialMode.
Value	0x85U

5.3.2.2.9. WDGM_EB_INIT_STATUS_DEINIT

Purpose	WdgM is / shall be de-initialized.
Value	0xFFU

5.3.2.2.10. WDGM_EB_INIT_STATUS_INIT

Purpose	WdgM is / shall be initialized.

Value	0U

5.3.2.2.11. WDGM_E_AMBIGIOUS

Purpose	Development error: Function WdgM_UpdateAliveIndication cannot determine the Checkpoint, because there are more than one alive supervisions configured in the current mode for the given Supervised Entity.
Value	0x18U

5.3.2.2.12. WDGM_E_CPID

Purpose	Development error: API service used with an invalid CheckpointId.
Value	0x16U

5.3.2.2.13. WDGM_E_DEPRECATED

Purpose	Development error: Deprecated API service was used.
Value	0x17U

5.3.2.2.14. WDGM_E_DISABLE_NOT_ALLOWED

Purpose	Development error: Disabling of watchdog not allowed.
Value	0x15U

5.3.2.2.15. WDGM_E_INV_POINTER

Purpose	Development error: API service called with NULL pointer.
Value	0x14U

5.3.2.2.16. WDGM_E_NOT_AUTHORIZED

Purpose	Development error: API service is not authorized to be executed (e.g. in case of a call to WdgM_PerformReset).

Value	0xF1U

5.3.2.2.17. WDGM_E_NO_INIT

Purpose	Development error: WdgM not initialized.
Value	0x10U

5.3.2.2.18. WDGM_E_PARAM_CONFIG

Purpose	Development error: API service Wdg_Init was called with an erroneous configuration set.
Value	0x11U

5.3.2.2.19. WDGM_E_PARAM_MODE

Purpose	Development error: API service called with invalid mode.
Value	0x12U

5.3.2.2.20. WDGM_E_PARAM_SEID

Purpose	Development error: API service called with invalid supervised entity.
Value	0x13U

5.3.2.2.21. WDGM_E_PARAM_WRONG_CORE_ID

Purpose	Development error: API service called with wrong core id.
Value	0x84U

5.3.2.2.22. WDGM_E_SEDEACTIVATED

Purpose	Development error: API service used with a checkpoint of a Supervised Entity that is deactivated in the current Watchdog Manager mode.
Value	0x19U

5.3.2.2.23. WDGM_GLOBAL_STATUS_DEACTIVATED

Purpose	Supervision deactivated.
Value	4U

5.3.2.2.24. WDGM_GLOBAL_STATUS_EXPIRED

Purpose	At least one Supervised Entity has Local Supervision Status WDGM_LOCAL_STATUS_EXPIRED and the time limit to postpone the blocking of watchdog triggering is not yet exceeded.
Value	2U

5.3.2.2.25. WDGM_GLOBAL_STATUS_FAILED

Purpose	At least one Supervised Entity has Local Supervision Status WDGM_LOCAL_STATUS_FAILED.
Value	1U

5.3.2.2.26. WDGM_GLOBAL_STATUS_OK

Purpose	Alive / Deadline / Logical Supervision of all active Supervised Entities fulfilled.
Value	0U

5.3.2.2.27. WDGM_GLOBAL_STATUS_STOPPED

Purpose	At least one Supervised Entity has Local Supervision Status WDGM_LOCAL_STATUS_EXPIRED and the time limit to postpone the blocking of watchdog triggering is exceeded.
Value	3U

5.3.2.2.28. WDGM_LOCAL_STATUS_DEACTIVATED

Purpose	Supervision deactivated.
Value	4U
Description	Alive, Deadline, and Logical Supervision is disabled for this Supervised Entity.

5.3.2.2.29. WDGM_LOCAL_STATUS_EXPIRED

Purpose	One of Alive, Deadline, or Logical Supervision is not fulfilled.
Value	2U
Description	Timing constraints for Alive Supervision have been violated including the margins for more often than the acceptable amount of failed supervision reference cycles.

5.3.2.2.30. WDGM_LOCAL_STATUS_FAILED

Purpose	Alive Supervision not fulfilled, but Deadline and Logical Supervision are fulfilled.
Value	1U
Description	Timing constraints for Alive Supervision have been violated including the margins, but the amount of failed supervision reference cycles has not been exceeded.

5.3.2.2.31. WDGM_LOCAL_STATUS_OK

Purpose	Alive / Deadline / Logical Supervision fulfilled.
Value	0U
Description	Timing constraints for Alive Supervision are fulfilled within the configured margins.

5.3.2.2.32. WDGM_MODULE_ID

Purpose	AUTOSAR module identification.
Value	13U

5.3.2.2.33. WDGM_SID_CHECKPOINT_REACHED

Purpose	Service id of WdgM_CheckpointReached().
Value	0x0eU

5.3.2.2.34. WDGM_SID_DEINIT

Purpose	Service id of WdgM_DeInit().
Value	0x01U

5.3.2.2.35. WDGM_SID_GET_ALL_EXPIRED_SEID

Purpose	Service id of WdgM_GetAllExpiredSEID().
Value	0x1CU

5.3.2.2.36. WDGM_SID_GET_FIRST_EXPIRED_SEID

Purpose	Service id of WdgM_GetFirstExpiredSEID().
Value	0x10U

5.3.2.2.37. WDGM_SID_GET_GLOBAL_STATUS

Purpose	Service id of WdgM_GetGlobalStatus().
Value	0x0dU

5.3.2.2.38. WDGM_SID_GET_LOCAL_STATUS

Purpose	Service id of WdgM_GetLocalStatus().
Value	0x0cU

5.3.2.2.39. WDGM_SID_GET_MODE

Purpose	Service id of WdgM_GetMode().
Value	0x0bU

5.3.2.2.40. WDGM_SID_GET_VERSION_INFO

Purpose	Service id of WdgM_GetVersionInfo().
Value	0x02U

5.3.2.2.41. WDGM_SID_INIT

Purpose	Service id of WdgM_Init().
Value	0x00U

5.3.2.2.42. WDGM_SID_PERFORM_RESET

Purpose	Service id of WdgM_PerformReset().
Value	0x0fU

5.3.2.2.43. WDGM_SID_SET_MODE

Purpose	Service id of WdgM_SetMode().
Value	0x03U

5.3.2.2.44. WDGM_SID_UPDATE_ALIVE_COUNTER

Purpose	Service id of WdgM_UpdateAliveCounter().
Value	0x04U

5.3.2.2.45. WDGM_SW_MAJOR_VERSION

Purpose	AUTOSAR module major version.
Value	6U

5.3.2.2.46. WDGM_SW_MINOR_VERSION

Purpose	AUTOSAR module minor version.
Value	1U

5.3.2.2.47. WDGM_SW_PATCH_VERSION

Purpose	AUTOSAR module patch version.
Value	43U

5.3.2.2.48. WDGM_VENDOR_ID

Purpose	AUTOSAR vendor identification: Elektrobit Automotive GmbH.
Value	1U

5.3.2.3. Functions

5.3.2.3.1. Supervisor_WdgM_ASR32_SetModeRedirectionCallout

Purpose	Redirected callout API for WdgM_SetMode.	
Synopsis	`Std_ReturnType` **`Supervisor_WdgM_ASR32_SetModeRedirectionCallout`** `( WdgM_ModeType Mode );`	
Service ID	0x03	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Parameters (in)	`Mode`	One of the configured Watchdog Manager modes
Return Value	Success of operation	
	`E_OK`	Successfully changed to the new mode
	`E_NOT_OK`	Changing to the new mode failed

5.3.2.3.2. Supervisor_WdgM_ASR40_SetModeRedirectionCallout

Purpose	Redirected callout API for WdgM_SetMode.	
Synopsis	`Std_ReturnType` **`Supervisor_WdgM_ASR40_SetModeRedirectionCallout`** `( WdgM_ModeType Mode , uint16 CallerID );`	
Service ID	0x03	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Parameters (in)	`Mode`	One of the configured Watchdog Manager modes
	`CallerID`	Module ID of the calling module
Return Value	Success of operation	
	`E_OK`	Successfully changed to the new mode
	`E_NOT_OK`	Changing to the new mode failed

5.3.2.3.3. Supervisor_WdgM_ActivateAliveSupervisionRedirectionCallout

Purpose	Activate alive supervision of a considered entity via a pre-configured callout API.

Synopsis	`Std_ReturnType` **`Supervisor_WdgM_ActivateAliveSupervisionRedirectionCallout`** `( WdgM_ASR40_SupervisedEntityIdType SEId );`	
Parameters (in)	`SEId`	Id of supervised entity whose alive supervision should be activated
Return Value	Success of operation	
	`E_OK`	Operation successful
	`E_NOT_OK`	Operation failed
Description	This function exists for ASR32 compatibility and simply redirects the call to a pre-configured API. Note: In ASR32, this API activated the alive supervision of a supervised entity of the active alive supervision configuration set.	

5.3.2.3.4. Supervisor_WdgM_DeInitRedirectionCallout

Purpose	Redirected callout API for WdgM_DeInit.
Synopsis	`void` **`Supervisor_WdgM_DeInitRedirectionCallout`** `( void );`
Service ID	0x01
Sync/Async	Synchronous
Reentrancy	Non reentrant

5.3.2.3.5. Supervisor_WdgM_DeactivateAliveSupervisionRedirectionCallout

Purpose	Deactivate alive supervision of a considered entity via a pre-configured callout API.	
Synopsis	`Std_ReturnType` **`Supervisor_WdgM_DeactivateAliveSupervisionRedirectionCallout`** `( WdgM_ASR40_SupervisedEntityIdType SEId );`	
Parameters (in)	`SEId`	Id of supervised entity whose alive supervision should be deactivated
Return Value	Success of operation	
	`E_OK`	Operation successful
	`E_NOT_OK`	Operation failed
Description	This function exists for ASR32 compatibility and simply redirects the call to a pre-configured API. Note: In ASR32, this API deactivated the alive supervision of a supervised entity of the active alive supervision configuration set.	

5.3.2.3.6. Supervisor_WdgM_DetCallout

Purpose	Indicate an internal error.	
Synopsis	`void` **`Supervisor_WdgM_DetCallout`** `( uint8 SID , uint8 ErrorID );`	
Parameters (in)	`SID`	ID of the API where an internal error was detected
	`ErrorID`	Internal Error Type

5.3.2.3.7. Supervisor_WdgM_GetApplicationStateCallout

Purpose	The function returns the current state of an OS-Application.	
Synopsis	`Std_ReturnType` **`Supervisor_WdgM_GetApplicationStateCallout`** `( ApplicationType Application , ApplicationStateRefType Value );`	
Parameters (in)	`Application`	The OS-Application from which the state is requested.
Parameters (out)	`Value`	The current state of the application.
Return Value	Success of operation.	
	`E_OK:`	No errors.
	`E_NOT_OK:`	Application is not valid

5.3.2.3.8. Supervisor_WdgM_GetCoreIdCallout

Purpose	The function returns a unique core identifier.
Synopsis	`WdgM_EB_CoreIdType` **`Supervisor_WdgM_GetCoreIdCallout`** `( void  );`
Return Value	The return value is the unique ID of the core.

5.3.2.3.9. Supervisor_WdgM_GetExpectedInitStateCallout

Purpose	Provide actual / get expected initialization state.	
Synopsis	`Std_ReturnType` **`Supervisor_WdgM_GetExpectedInitStateCallout`** `( WdgM_EB_InitStatusType * InitStatus );`	
Parameters (in,out)	`InitStatus`	The caller of this API (WdgM) provides the current initial status, the expected initial

		status for the WdgM shall be returned in case E_OK is returned.
Return Value	Success of operation	
	`E_OK`	The returned value in InitStatus is valid. The WdgM changes to the expected state.
	`E_NOT_OK`	The returned value is not valid and will be ignored. WdgM continues normal operation.
Description	The callout is invoked at the beginning of WdgM_MainFunction cycle. Possible values for InitStatus: WDGM_EB_INIT_STATUS_INIT The WdgM shall be initialized, respectively stay initialized. WDGM_EB_INIT_STATUS_DEINIT The WdgM shall be de-initialized, respectively stay de-initialized.	

5.3.2.3.10. Supervisor_WdgM_GetExpectedWdgMModeCallout

Purpose	Provide actual / get expected WdgM Mode.	
Synopsis	`Std_ReturnType` **`Supervisor_WdgM_GetExpectedWdgMModeCallout`** `( WdgM_ModeType * WdgMMode );`	
Parameters (in,out)	`WdgMMode`	The caller of this API (WdgM) provides the current mode, the expected WdgM mode shall be returned in case E_OK is returned.
Return Value	Success of operation	
	`E_OK`	The WdgM shall perform a mode switch to the mode stored in the argument WdgM-Mode.
	`E_NOT_OK`	The returned value is not valid and will be ignored. WdgM continues normal operation.
Description	The callout is invoked at the beginning of WdgM_MainFunction cycle if WdgM is initialized and the prior call to WdgMGetExpectedInitStateCallout also returned WDGM_EB_INIT_STATUS_INIT.	

5.3.2.3.11. Supervisor_WdgM_GetTimeCallout

Purpose	Get elapsed time.

Synopsis	`void` **`Supervisor_WdgM_GetTimeCallout`** `( uint32 * PreviousTime , uint32 * ElapsedTime );`	
Parameters (in,out)	`PreviousTime`	The old time is passed in order to calculate the difference with respect to the actual time. The actual time is returned via this variable.
Parameters (out)	`ElapsedTime`	The elapsed time with respect to the time passed via parameter PreviousTime.

5.3.2.3.12. Supervisor_WdgM_GlobalModeSwitchCallout

Purpose	Indicate a state transition of the global Supervision State.	
Synopsis	`void` **`Supervisor_WdgM_GlobalModeSwitchCallout`** `( WdgM_GlobalStatusType OldMode , WdgM_GlobalStatusType NewMode );`	
Parameters (in)	`OldMode`	Old global WdgMMode
	`NewMode`	New global WdgMMode

5.3.2.3.13. Supervisor_WdgM_IndividualModeSwitchCallout

Purpose	Indicate a state transition of a Supervised Entity.	
Synopsis	`void` **`Supervisor_WdgM_IndividualModeSwitchCallout`** `( WdgM_ASR40_SupervisedEntityIdType SEID , WdgM_LocalStatusType OldMode , WdgM_LocalStatusType NewMode );`	
Parameters (in)	`SEID`	Supervised Entity ID that performed a mode switch
	`OldMode`	Old WdgMMode of the SEID
	`NewMode`	New WdgMMode of the SEID

5.3.2.3.14. Supervisor_WdgM_InitRedirectionCallout

Purpose	Redirected callout API for WdgM_Init.
Synopsis	`void` **`Supervisor_WdgM_InitRedirectionCallout`** `( const WdgM_ConfigType * ConfigPtr );`
Service ID	0x00
Sync/Async	Synchronous

Reentrancy	Non reentrant	
Parameters (in)	`ConfigPtr`	Pointer to configuration data, this parameter is ignored in the current implementation.

5.3.2.3.15. Supervisor_WdgM_IsPerformResetCallout

Purpose	Get authorization for direct reset.	
Synopsis	`Std_ReturnType` **`Supervisor_WdgM_IsPerformResetCallout`** `( void );`	
Return Value	Returns if the caller of the WdgM Perform Reset API is authorized.	
	`E_OK`	Caller of Perform Reset is authorized and therefore reset request will be executed by setting all triggering conditions to 0.
	`E_NOT_OK`	Caller of Perform Reset is not authorized and therefore reset request will be ignored. WdgM continues normal operation.

5.3.2.3.16. Supervisor_WdgM_RequestPartitionResetCallout

Purpose	Function called by WdgM to request a partition reset.	
Synopsis	`Std_ReturnType` **`Supervisor_WdgM_RequestPartitionResetCallout`** `( ApplicationType Application );`	
Parameters (in)	`Application`	The identifier of an OS-Application.
Return Value	`E_OK:`	No errors.
`E_NOT_OK:`	Partition was not restarted	

5.3.2.3.17. Supervisor_WdgM_SupervisionExpiredCallout

Purpose	Indicate an expired Supervised Entity.	
Synopsis	`void` **`Supervisor_WdgM_SupervisionExpiredCallout`** `( WdgM_ASR40_SupervisedEntityIdType ExpiredSEID );`	
Parameters (in)	`ExpiredSEID`	Supervised Entity ID that expired (one of Deadline Supervision, Logical Supervision, or Alive Supervision failed).

5.3.2.3.18. WdgM_CheckpointReached

Purpose	Give alive indications to the Watchdog Manager.	
Synopsis	`Std_ReturnType WdgM_CheckpointReached ( WdgM_SupervisedEntityIdType SEID , WdgM_CheckpointIdType CheckpointID );`	
Service ID	0x0e	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Production Errors	► WDGM_E_DATA_CORRUPTION: thrown, if data corruption is detected in the internal WdgM data	
Parameters (in)	`SEID`	ID of supervised entity whose alive counter is updated
	`CheckpointID`	Identifier of the Checkpoint within a Supervised Entity that has been reached.
Return Value	Success of operation	
	`E_OK`	Operation successful
	`E_NOT_OK`	Operation failed
Description	This function indicates to the Watchdog Manager that a checkpoint within a supervised entity has been reached.	

5.3.2.3.19. WdgM_DeInit

Purpose	De-initialize the Watchdog Manager.
Synopsis	`void WdgM_DeInit ( void );`
Service ID	0x01
Sync/Async	Synchronous
Reentrancy	Non reentrant

5.3.2.3.20. WdgM_GetAllExpiredSEID

Purpose	Get all supervised entities that have expired.
Synopsis	`Std_ReturnType WdgM_GetAllExpiredSEID ( uint8 * ExpiredSEID , uint8 * NoOfExpiredSEID );`

Service ID	0x1c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`ExpiredSEID`	List of expired supervised entities
	`NoOfExpiredSEID`	The number of expired supervised entities
Return Value	Success of operation	
	`E_OK`	Operation successful
	`E_NOT_OK`	Operation failed
Description	Returns all the supervised entities that have expired and their total number.	

5.3.2.3.21. WdgM_GetFirstExpiredSEID

Purpose	Get SEID that first reached WDGM_LOCAL_STATUS_EXPIRED.	
Synopsis	`Std_ReturnType` **`WdgM_GetFirstExpiredSEID`** `( WdgM_SupervisedEntityIdType * SEID );`	
Service ID	0x10	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`SEID`	Supervised entity ID that first reached WDGM_LOCAL_STATUS_EXPIRED
Return Value	Success of operation	
	`E_OK`	Operation successful
	`E_NOT_OK`	Operation failed
Description	Returns SEID that first reached the state WDGM_LOCAL_STATUS_EXPIRED.	

5.3.2.3.22. WdgM_GetGlobalStatus

Purpose	Get global supervision status of the Watchdog Manager.
Synopsis	`Std_ReturnType` **`WdgM_GetGlobalStatus`** `( WdgM_GlobalStatusType * Status );`
Service ID	0x0d
Sync/Async	Synchronous

Reentrancy	Reentrant	
Parameters (out)	`Status`	Global supervision status
Return Value	Success of operation	
	`E_OK`	Operation successful
	`E_NOT_OK`	Operation failed

5.3.2.3.23. WdgM_GetLocalStatus

Purpose	Get supervision status of a specific entity.	
Synopsis	`Std_ReturnType` **`WdgM_GetLocalStatus`** `( WdgM_SupervisedEntityIdType SEID , WdgM_LocalStatusType * Status );`	
Service ID	0x0c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	`SEID`	ID of supervised entity whose status shall be returned
Parameters (out)	`Status`	Status of the given supervised entity
Return Value	Success of operation	
	`E_OK`	Operation successful
	`E_NOT_OK`	Operation failed

5.3.2.3.24. WdgM_GetMode

Purpose	Returns the current mode of the Watchdog Manager.	
Synopsis	`Std_ReturnType` **`WdgM_GetMode`** `( WdgM_ModeType * Mode );`	
Service ID	0x0b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (out)	`Mode`	Current WdgM mode
Return Value	Success of operation	
	`E_OK`	Current mode successfully returned
	`E_NOT_OK`	Returning current mode failed

5.3.2.3.25. WdgM_GetVersionInfo

Purpose	Get version information of the Watchdog Manager.	
Synopsis	`void WdgM_GetVersionInfo ( Std_VersionInfoType * VersionInfo );`	
Service ID	0x02	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Parameters (out)	`VersionInfo`	Version information
Description	This service returns the version information of this module. The version information includes ► Module Id ► Vendor Id ► Vendor specific version numbers	

5.3.2.3.26. WdgM_Init

Purpose	Initialize the Watchdog Manager.	
Synopsis	`void WdgM_Init ( const WdgM_ConfigType * ConfigPtr );`	
Service ID	0x00	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Parameters (in)	`ConfigPtr`	Pointer to configuration data, this parameter is ignored in the current implementation.

5.3.2.3.27. WdgM_MainFunction

Purpose	Cyclic main function for WdgM processing.
Synopsis	`void WdgM_MainFunction ( void );`
Service ID	0x08
Sync/Async	Synchronous
Reentrancy	Non reentrant
Production Errors	► WDGM_E_MONITORING: thrown, if supervision has failed for a supervised entity

	▶ WDGM_E_SET_MODE: thrown, if watchdog drivers' mode switch has failed ▶ WDGM_E_MF_TIMINGVIOLATION: thrown, if WdgM_MainFunction period deviates from the configured mode-dependent schedule period (WdgMSupervisionCycle) ▶ WDGM_E_DATA_CORRUPTION: thrown, if data corruption is detected in the internal WdgM data
Description	Performs the processing of the cyclic Watchdog Manager jobs.

5.3.2.3.28. WdgM_PerformReset

Purpose	Force a watchdog reset.
Synopsis	`void WdgM_PerformReset ( void );`
Service ID	0x0f
Sync/Async	Synchronous
Reentrancy	Non reentrant
Description	Instructs the Watchdog Manager to cause a watchdog reset.

5.3.2.3.29. WdgM_SetMode

Purpose	Set the current mode of the Watchdog Manager.	
Synopsis	`Std_ReturnType WdgM_SetMode ( WdgM_ModeType Mode , uint16 CallerID );`	
Service ID	0x03	
Sync/Async	Synchronous	
Reentrancy	Non reentrant	
Production Errors	▶ WDGM_E_IMPROPER_CALLER: thrown, if the passed CallerID is not in the list of the configured list of allowed CallerIDs	
Parameters (in)	`Mode`	One of the configured Watchdog Manager modes
	`CallerID`	Module ID of the calling module
Return Value	Success of operation	
	`E_OK`	Successfully changed to the new mode

	`E_NOT_OK`	Changing to the new mode failed

5.3.2.3.30. WdgM_UpdateAliveCounter

Purpose	Give alive indications to the Watchdog Manager via AUTOSAR 3.2 API (deprecated).	
Synopsis	`Std_ReturnType` **`WdgM_UpdateAliveCounter`** `( WdgM_SupervisedEntityIdType SEID );`	
Service ID	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Production Errors	► WDGM_E_DATA_CORRUPTION: thrown, if data corruption is detected in the internal WdgM data	
Parameters (in)	`SEID`	ID of supervised entity whose alive counter is updated
Return Value	Success of operation	
	`E_OK`	Operation successful
	`E_NOT_OK`	Operation failed
Description	This function updates the alive counter of a requested supervised entity of the current mode.	

5.3.3. Integration notes

5.3.3.1. Exclusive areas

Exclusive areas are not used by the `WdgM` module.

5.3.3.2. Production errors

WDGM_E_DATA_CORRUPTION	► WdgM_CheckpointReached ► WdgM_MainFunction ► WdgM_UpdateAliveCounter

WDGM_E_IMPROPER_CALLER	▶ WdgM_SetMode
WDGM_E_MF_TIMINGVIOLATION	▶ WdgM_MainFunction
WDGM_E_MONITORING	▶ WdgM_MainFunction
WDGM_E_SET_MODE	▶ WdgM_MainFunction

5.3.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

Memory section
CODE
CALLOUT_CODE_ASIL_D
CODE_ASIL_D
CONST_8
CONST_ASIL_D_8
CONST_16
CONST_ASIL_D_16
CONST_ASIL_D_UNSPECIFIED
VAR_CLEARED_ASIL_D_UNSPECIFIED
VAR_INIT_ASIL_D_UNSPECIFIED
VAR_CLEARED_16
VAR_CLEARED_ASIL_D_16
VAR_CLEARED_ASIL_D_32
VAR_INIT_ASIL_D_8
VAR_POWER_ON_INIT_ASIL_D_8
VAR_INIT_8
VAR_INIT_16
VAR_CLEARED_UNSPECIFIED
VAR_CLEARED_ASIL_D_8
VAR_INIT_ASIL_D_LOCAL_8

VAR_INIT_UNSPECIFIED
VAR_INIT_GLOBAL_32
VAR_POWER_ON_CLEARED_ASIL_D_UNSPECIFIED
VAR_GLOBAL_32
VAR_CLEARED_GLOBAL_UNSPECIFIED
VAR_CLEARED_GLOBAL_SHARED_UNSPECIFIED
VAR_CLEARED_32
VAR_POWER_ON_INIT_ASIL_D_16
VAR_INIT_ASIL_D_32
VAR_POWER_ON_INIT_ASIL_D_UNSPECIFIED
VAR_CLEARED_SHARED_UNSPECIFIED
VAR_SHARED_INIT_16
VAR_INIT_SHARED_UNSPECIFIED
SATELLITE_R_VAR_INIT_8
SATELLITE_R_VAR_INIT_16
SATELLITE_R_VAR_CLEARED_16
VAR_LOCAL_8
VAR_INIT_LOCAL_16
VAR_CLEARED_8
VAR_16

5.3.3.4. Integration requirements

WARNING

Integration requirements list is not exhaustive

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

5.3.3.4.1. lim.WdgM.EB_INTREQ_WdgM_0001

Description	If the WdgM_PerformReset API can interrupt the WdgM_MainFunction, the Watchdog Manager may postpone the update of the Watchdog triggering conditions to 0 by one main function cycle due to race conditions.

Rationale	Existing race condition in case WdgM_PerformReset interrupt has higher priority than WdgM_MainFunction.

5.3.3.4.2. lim.WdgM.EB_INTREQ_WdgM_0002

Description	If the WdgM is integrated in an environment where the individual applications are allocated to different partitions, then the general parameter WdgMPartitioningEnabled must be enabled. The WdgM then makes use of the memory section WDGM_START/STOP_SEC_VAR_CLEARED_SHARED_UNSPECIFIED which is required for the access to the shared graph data (Logical Supervision) in the context of different applications / Supervised Entities. The integrity of the shared data is ensured by holding all values double inverse. Therefore, the integrator must configure this section to be globally accessable (write and readably) from all partitions.
Rationale	This version of the WdgM module is tailored for the use in a single core environment where the shared memory approach is more efficient than splitting the memory among the different partitions.

5.3.3.4.3. lim.WdgM.EB_INTREQ_WdgM_0003

Description	If mode swich is done synchronous (WdgMSetModeSynchron is set to TRUE) then the API WdgM_SetMode should not be preempt by WdgM_DeInit.
Rationale	Both APIs are setting the watchdog to a specific mode. The preemtion of WdgM_SetMode by WdgM_DeInit may lead to inconsistency in the watchdog mode.

5.3.3.4.4. lim.WdgM.EB_INTREQ_WdgM_0004

Description	If mode swich is done synchronous (WdgMSetModeSynchron is set to TRUE) and the WdgM_SetMode API can interrupt the WdgM_CheckpointReached, the Watchdog Manager may postpone the evaluation of the Supervised Entyty results if it is not used in the new mode until it is activated again in a new mode, due to race conditions.
Rationale	Existing race condition in case WdgM_SetMode interrupt has higher priority than WdgM_CheckpointReached.

5.3.3.4.5. lim.WdgM.EB_INTREQ_WdgM_0005

Description	If mode swich is done synchronous (WdgMSetModeSynchron is set to TRUE) and the WdgM_MainFunction API can interrupt the WdgM_SetMode, the Watchdog Manager

	may evaluate results of the Supervised Entities from the old mode and the new mode, until the next main function, due to race conditions.
Rationale	Existing race condition in case WdgM_MainFunction interrupt has higher priority than WdgM_CheckpointReached.

5.3.3.4.6. lim.WdgM.EB_INTREQ_WdgM_0006

Description	If mode swich is done synchronous (WdgMSetModeSynchron is set to TRUE) then WdgM_SetMode should not interrupt the WdgM_MainFunction.
Rationale	WdgMFailedAliveSupervisionRefCycle of a Supervised Entity which is not used in the new mode when WdgM_SetMode is called may have a value of 1 instead of 0 when the Supervised Enity is reactivated in a new mode.

5.3.3.4.7. lim.WdgM.EB_INTREQ_WdgM_0007

Description	If legacy symbolic names (from AUTOSAR versions 3.x or lower than 4.0.3) are used then the macro WDGM_PROVIDE_LEGACY_SYMBOLIC_NAMES shall be defined before including the WdgM header file.
Rationale	The usage of legacy symbolic names is discouraged, but for backwards compatibility the usage of these symbolic names is provided. Recommendation is to go for AUTOSAR 4.0.3 symbolic names (See TPS_ECUC_02108 from AUTOSAR_TPS_ECUConfiguration.pdf) and not enable the macro.

5.3.3.4.8. lim.WdgM.EB_INTREQ_WdgM_0008

Description	If AUTROSAR 4.3 service is used then the swcd arxml files are generated only for the generate_swcd command in Tresos.
Rationale	generate_asr32_swcd is used in projects where AUTOSAR 3.2 model is used and generate_asr40_swcd is the same as generate_swcd.

5.3.3.4.9. lim.WdgM.EB_INTREQ_WdgM_0009

Description	The integrator must assure the natural alignment of the variables, based on their memory mapping.
Rationale	To be able to access the data atomically the data must be align correctly.

5.3.3.4.10. lim.WdgM.EB_INTREQ_WdgM_0010

Description	If multicore feature is enabled, the integrator must assure cache coherency for Inter Core Data exchange.
Rationale	Cache coherency must be assured or cache must be disabled, otherwise inter core data consistency cannot be guaranteed.

5.3.3.4.11. lim.WdgM.EB_INTREQ_WdgM_0011

Description	If Partition Restart feature is used then the integrator shall assure the WdgM_MainFunction executes on a partition not referenced by any Supervised Entity.
Rationale	WdgM_MainFunction is evaluating the partition status. If partition is reset then WdgM_MainFunction does not execute anymore to evaluate the partition status.

5.3.3.4.12. lim.WdgM.EB_INTREQ_WdgM_0012

Description	If multicore feature is enabled, the integrator shall call WdgM_GetAllExpiredSEID, WdgM_GetFirstExpiredSEID, WdgM_GetGlobalStatus, WdgM_GetMode, WdgM_PerformReset and WdgM_SetMode only from the Master Instance of WdgM.
Rationale	These APIs are related with the Global Status of WdgM and must be controlled by the WdgM Master Instance.

5.3.3.4.13. lim.WdgM.EB_INTREQ_WdgM_0013

Description	If multicore feature is enabled and WdgM_PerformReset() is called then WdgM will update the trigger conditions to zero only for the master controlled watchdogs.
Rationale	WdgMImmediateReset can be used to trigger a ECU reset or an watchdog which reset the whole ECU.

5.3.3.4.14. lim.WdgM.EB_INTREQ_WdgM_0014

Description	If multicore feature and configuration parameter WdgMGetAllExpiredSEIDs are enabled then WdgM_GetAllExpiredSEID() API shall be called before WdgM is re-initialized.
Rationale	Saved expired supervised entities are relevant after reset before the re-initialization of the WdgM. Afterwards WdgM will start again supervising.

5.3.3.4.15. lim.WdgM.EB_INTREQ_WdgM_0015

Description	If multicore feature is enabled then WdgM_GetFirstExpiredSEID() may not be fully reliable due to race conditions.
Rationale	Without a timestamp over cores this can not be achieved.