

MICROSAR WDGM

Technical Reference

Version 4.00.02

Authors	Peter Lang
Status	Released

1 Document Information

1.1 History

	Date	Version	Remarks
Peter Paulus	2007-11-13	1.0	Creation of the document (AUTOSAR 2.0)
Peter Paulus	2006-12-12	1.1	Update due to user manual review
Peter Paulus	2007-06-18	1.2	Update due to AUTOSAR 2.1 release, revision of all chapters
Peter Paulus	2007-07-04	1.3	Revision of complete document
Peter Paulus	2007-07-06	1.4	Headlines formatted, change of label at configuration view of WDGM regarding 'Interrupt Handling'
Peter Paulus	2007-07-09	1.5	Addition of configuration parameter of setting the mode of the underlying watchdog instances during initialization of the WDGM
Peter Paulus	2007-07-26	1.6	Addition of chapter for description of the Service Port functionality
Peter Paulus	2007-08-29	1.7	Description of service port names (e.g. WdgM_Runtime_1_3 added) modification of chapter 1.5.1.5 addition of chapter 1.6.7.1
Peter Paulus	2007-08-31	1.8	Modification of Table 4-3 Compiler abstraction and memory mapping addition of compiler keywords WDGM_CONST, WDGM_MCU_CODE and memmap keyword WDGM_START_SEC_CONST_UNSPECIFIED
Peter Paulus	2007-12-07	2.0.0	Changing the version number to a three digit number
Peter Paulus, Heike Bischof	2008-03-25	3.00.00	- Conversion to Technical Reference - Introduction of ASR3 release (WDGM AR version 1.2.0) - Changing to five-digit version number - Set document status to Released
Peter Paulus	2008-06-03	3.00.01	Added parameter "Use Rte" to "General

			Settings" tab (chapter 6.1.6.2).
Bethina Mausz	2008-09-17	3.00.02	Changing description fields ESCAN00029212
Bethina Mausz	2009-01-08	3.01.00	Insert Mode Management feature ESCAN00030954

Bethina Mausz	2009-08-13	3.02.00	<p>AUTOSAR figure updated ESCAN00034456</p> <p>removed the column 'Configuration Variant'</p> <p>ESCAN00036347: "5.7.1 Export of Software Component Template files" additional information about SW-C generation via DaVinci Configurator Pro inserted</p> <p>correction in table headings done</p> <p>ESCAN00035126: update the figure about include structure</p> <p>ESCAN00032744: updated the description about development error handling</p> <p>ESCAN00033644, ESCAN00033990, ESCAN00034698: updated the memory mapping table</p> <p>Updates in description for DaVinci Configurator Pro.</p> <p>ESCAN00036296: Add chapter "Configuration Variant"</p> <p>Insert some more information for DaVinci Configurator Pro and update some description for usage of WgM configuration</p> <p>Insert Update for Compiler Abstraction and Memory Mapping</p>
---------------	------------	---------	--

Bethina Mausz	2009-10-21	3.02.01	ESCAN00037853: insert additional information about the handling of two or more modes with different activation mode.
Bethina Mausz	2010-08-23	3.02.02	ESCAN00043729: inserted more information about watchdog mode
Martin Froschhammer	2012-07-21	4.00.00	Reworked whole document for AUTOSAR 4 release of the component
Peter Lang	2014-01-14	4.00.02	Modifications due to new description file

Table 1-1 History of the document

1.2 Reference Documents

No.	Title	Version
[1]	AUTOSAR_SWS_WatchdogManager.pdf	V2.2.0
[2]	AUTOSAR_SWS_DevelopmentErrorTracer.pdf	V3.2.0
[3]	AUTOSAR_SWS_DiagnosticEventManager.pdf	V4.2.0
[4]	AUTOSAR_TR_BSWModuleList.pdf	V1.6.0
[5]	AUTOSAR_SWS_WatchdogInterface.pdf	V2.5.0
[6]	AUTOSAR_SWS_WatchdogDriver.pdf	V2.5.0

Table 1-2 Reference documents



Please note

We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector's release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

1	Document Information	2
1.1	History	2
1.2	Reference Documents.....	5
2	Introduction.....	12
2.1	Architecture Overview	13
3	Functional Description	15
3.1	Features.....	15
3.2	Initialization.....	16
3.3	States	17
3.3.1	Global Supervision Status.....	17
3.3.2	Local Supervision Status.....	18
3.3.3	Module States	20
3.4	Main Functions	20
3.4.1	Alive Supervision	20
3.5	Error Handling.....	20
3.5.1	Development Error Reporting.....	20
3.5.1.1	Parameter Checking.....	22
3.5.2	Production Code Error Reporting	24
4	Integration	26
4.1	Scope of Delivery	26
4.1.1	Static Files	26

4.1.2	Dynamic Files	26
4.2	Include Structure	27
4.3	Compiler Abstraction and Memory Mapping.....	27
4.4	Dependencies on SW modules.....	28
4.4.1	AUTOSAR OS	28
4.4.2	DEM.....	28
4.4.3	DET	28
4.4.4	WDGIF	29
4.4.5	SchM	29
4.4.6	MCU	29
4.4.7	RTE	29
4.5	Dependencies on HW modules	29
5	API Description.....	30
5.1	Interfaces Overview	30
5.2	Type Definitions.....	30
5.3	Services provided by WDGm	32
5.3.1	WdgM_Init	32
5.3.2	WdgM_DeInit	32
5.3.3	WdgM_GetVersionInfo	33
5.3.4	WdgM_SetMode.....	34
5.3.5	WdgM_GetMode	36
5.3.6	WdgM_CheckpointReached.....	36
5.3.7	WdgM_UpdateAliveCounter	37

5.3.8	WdgM_GetLocalStatus	38
5.3.9	WdgM_GetGlobalStatus	39
5.3.10	WdgM_PerformReset	40
5.3.11	WdgM_GetFirstExpiredSEID	41
5.3.12	WdgM_MainFunction	41
5.4	Callback Functions	42
5.5	Configurable Interfaces	42
5.6	Service Ports	43
5.6.1	Client Server Interface	43
5.6.1.1	Provide Ports on WDGM Side	43
5.6.1.1.1	WdgM_AliveSupervision	43
5.6.1.2	Require Ports on WDGM Side	43
5.6.1.2.1	WdgM_IndividualMode	44
5.6.1.2.2	WdgM_GlobalMode	44
6	Configuration	45
6.1	Configuration Variants	45
6.2	Configuration of WDGM	45
7	AUTOSAR Standard Compliance	46
7.1	Additions/ Extensions	46
7.1.1	Parameter Checking	46
7.2	Limitations	46
7.2.1	Link-time configuration is not supported	46
7.2.2	Deadline and Logical Supervision are not supported	46

- 7.2.3 Partition reset is not supported 46

- 8 Glossary and Abbreviations47**
 - 8.1 Glossary..... 47
 - 8.2 Abbreviations 47

- 9 Contact49**

Illustrations

Figure 2-1	AUTOSAR architecture	13
Figure 2-2	Interfaces to adjacent modules of the WDM.....	14
Figure 3-1	State machine overview of the Global Supervision Status.....	17
Figure 3-2	State machine overview of the Local Supervision Status.....	19
Figure 4-1	Include structure.....	27

Tables

Table 1-1	History of the document	5
Table 1-2	Reference documents	5
Table 3-1	Supported SWS features	15
Table 3-2	Not supported SWS features	16
Table 3-3	Global Supervision Status	18
Table 3-4	Individual Supervision Status	20
Table 3-5	Module states	20
Table 3-6	Mapping of service IDs to services	21
Table 3-7	Errors reported to DET	22
Table 3-8	Development Error Reporting: Assignment of checks to services	23
Table 3-9	Errors reported to DEM.....	25
Table 4-1	Static files	26
Table 4-2	Generated files	26
Table 4-3	Compiler abstraction and memory mapping	28
Table 5-1	Type definitions	31
Table 5-2	WdgM_Init	32
Table 5-3	WdgM_DeInit	33
Table 5-4	WdgM_GetVersionInfo	34

Table 5-5	WdgM_SetMode	35
Table 5-6	WdgM_GetMode	36
Table 5-7	WdgM_CheckpointReached.....	37
Table 5-8	WdgM_UpdateAliveCounter.....	38
Table 5-9	WdgM_GetLocalStatus	39
Table 5-10	WdgM_GetGlobalStatus	40
Table 5-11	WdgM_PerformReset.....	41
Table 5-12	WdgM_GetFirstExpiredSEID	41
Table 5-13	WdgM_MainFunction.....	42
Table 5-14	Configurable interfaces	42
Table 5-15	Adaptation of Provide Ports.....	43
Table 5-16	WdgM_AliveSupervision	43
Table 5-17	WdgM_IndividualMode.....	44
Table 5-18	WdgM_IndividualMode.....	44
Table 8-1	Glossary	47
Table 8-2	Abbreviations	48

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module WDGM as specified in [1].

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	post-build	
Vendor ID:	WDGM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	WDGM_MODULE_ID	13 (according to ref. [4])

* For the precise AUTOSAR Release 4.x please see the release specific documentation.

This document describes the functionality and API of the Watchdog Manager (WDGM) as a hardware independent module.

The WDGM has two major tasks:

- > Supervise all applications (Supervised Entities) for liveliness configured within the WDGM.
- > Triggering all available hardware watchdog instances within the ECU if all activated Supervised Entities behave in the expected way.

2.1 Architecture Overview

The following figure shows where the WDM is located in the AUTOSAR architecture.

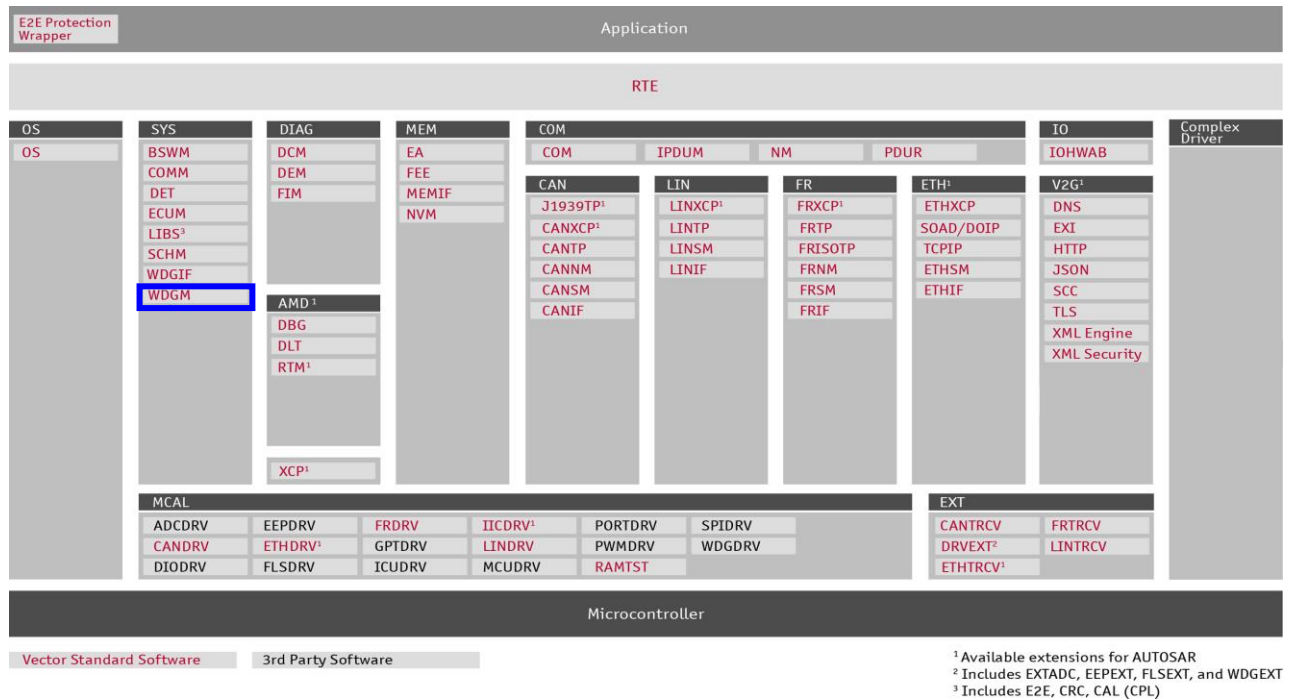


Figure 2-1 AUTOSAR architecture

The next figure shows the interfaces to adjacent modules of the WDM. These interfaces are described in chapter 5.

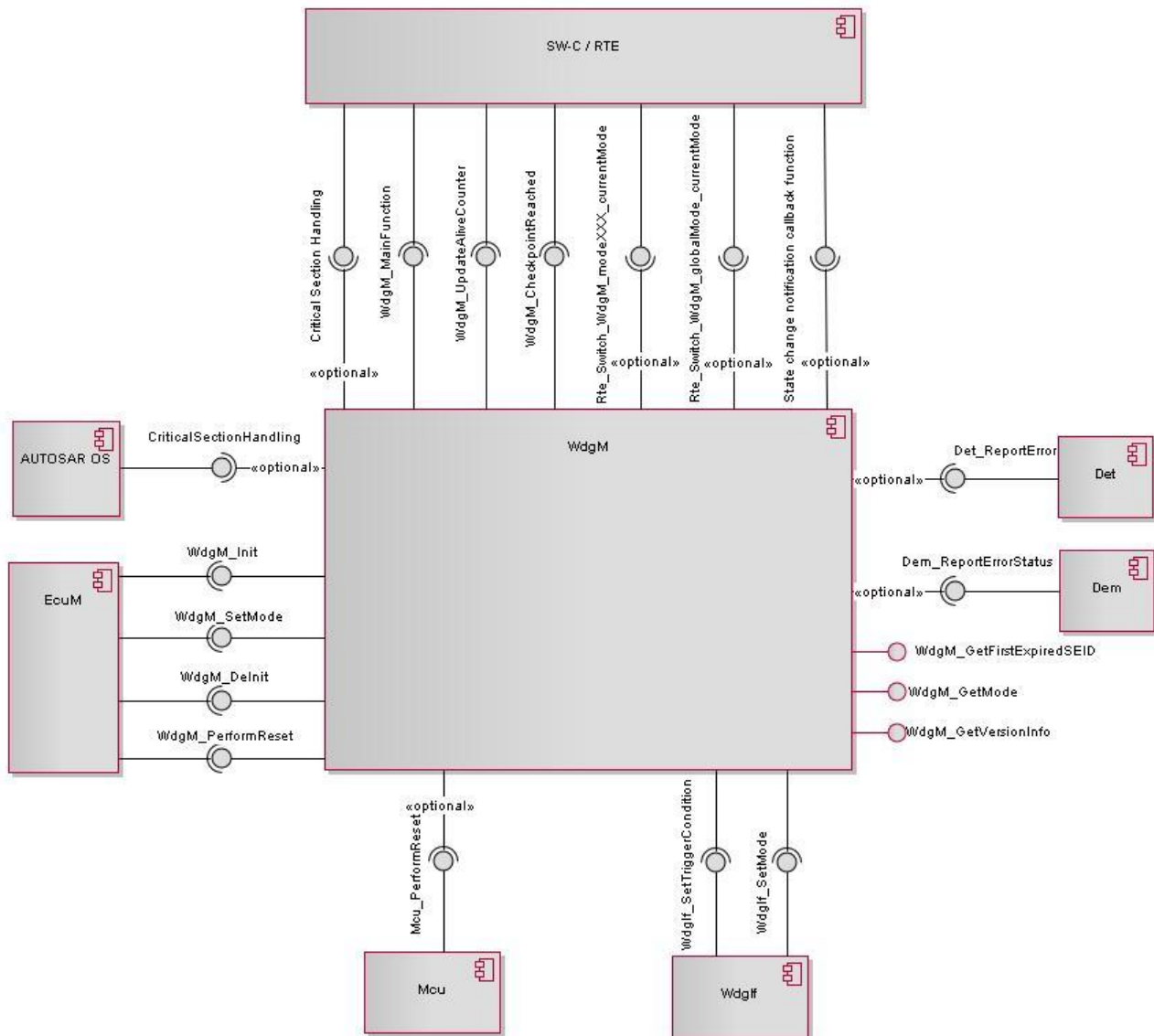


Figure 2-2 Interfaces to adjacent modules of the WDM

Applications do not access the services of the BSW modules directly. They use the service ports provided by the BSW modules via the RTE. The service ports provided by the WDM are listed in chapter 5.6 and are defined in [1].

3 Functional Description

3.1 Features

The features listed in this chapter cover the complete functionality specified in [1].

The "supported" and "not supported" features are presented in the following two tables. For further information of not supported features also see chapter 7.

The following features described in [1] are supported:

Supported Features
Alive Supervision of the Supervised Entities according to its configurations.
Enable/disable Alive Supervision of Supervised Entities via Mode Switch.
Give the permission for triggering the watchdog instances according to their configurations.
Prevent mode setting of watchdog instances if the watchdog does not provide multiple mode changes.
Support of one/multiple watchdog instances (code/runtime improvements).
Allow/disallow the transition of watchdogs in Off-Mode.
Enable/disable development error detection and reporting to the DET (by default).
Enable/disable immediate resets via the MCU at faulty conditions recognized by the WDGM.
Perform mode changes initiated by the ECUM.
Selection of handling mechanism to protect critical sections.
Generation of the SW-C description needed for the generation of the RTE.
Configuration of service port name of the WDGM within the SW-C description file.
Enable/disable the availability of the API function WdgM_GetVersionInfo().
Reporting Local Supervision Status changes of Supervised Entities.
Reporting Global Supervision Status change.
Evaluation of first expired Supervised Entity after watchdog reset
Service for performing a watchdog reset

Table 3-1 Supported SWS features

The following features described in [1] are not supported:

Not Supported Features
The Main Supervision Cycle is configurable only once in case the RTE functionality is used, because the RTE supports currently only one value for cycle time.

Not Supported Features

Deadline Supervision and Logical Supervision is not supported

Partition reset by calling BswM_WdgM_RequestPartitionReset() is not supported.

Table 3-2 Not supported SWS features

3.2 Initialization

The WDGM is initialized and operational after the API function WdgM_Init() has been called.



Caution

Subject:

Erroneous initialization of the WDGM

Content:

The WdgM tries to set a new mode of the underlying watchdog instances at specific states of the ECU, especially at the startup of the ECU via the function WdgMf_SetMode() and hence Wdg_SetMode(). It is generally possible that this mode-setting procedure fails sporadically (independent of the reason), i.e. the call of the function WdgMf_SetMode() returns with E_NOT_OK.

If mode-setting procedure fails while initialization of the WDGM, the WDGM is not initialized and the ECUM does not recognize this (by default). Thus, the ECUM continues to startup the system or continues code execution generally.

Therefore, **it is recommended to the user/integrator, that the “Enable Immediate Reset” feature should be used/configured within the WDGM** in order to prevent to return to the caller.

Additionally, not using the MCU (or a similar component) to perform resets would lead to unpredictable behaviors of the ECU, especially if the startup of the WDGM fails.

Note:

In case of the startup of the ECU, the system is not yet able to store production errors to the DEM, because the system is not fully initialized at all nor is enough time to store the mode-setting error (generate by the faulty watchdog instance) on the non-volatile memory. Hence, it is not possible to trace the reason of the shortly performed reset of the ECU by the MCU (if configured) or by a watchdog reset.

3.3 States

The Watchdog Manager is internally organized with a state machine which is shown in the following chapters.

3.3.1 Global Supervision Status

All states shown below are handled by the Global Supervision State, whereas the Global Supervision State can only embrace one of the states at one time.

The Global Supervision Status represents the status of the WDGM at all. Triggering the configured watchdog instances depends on this state.

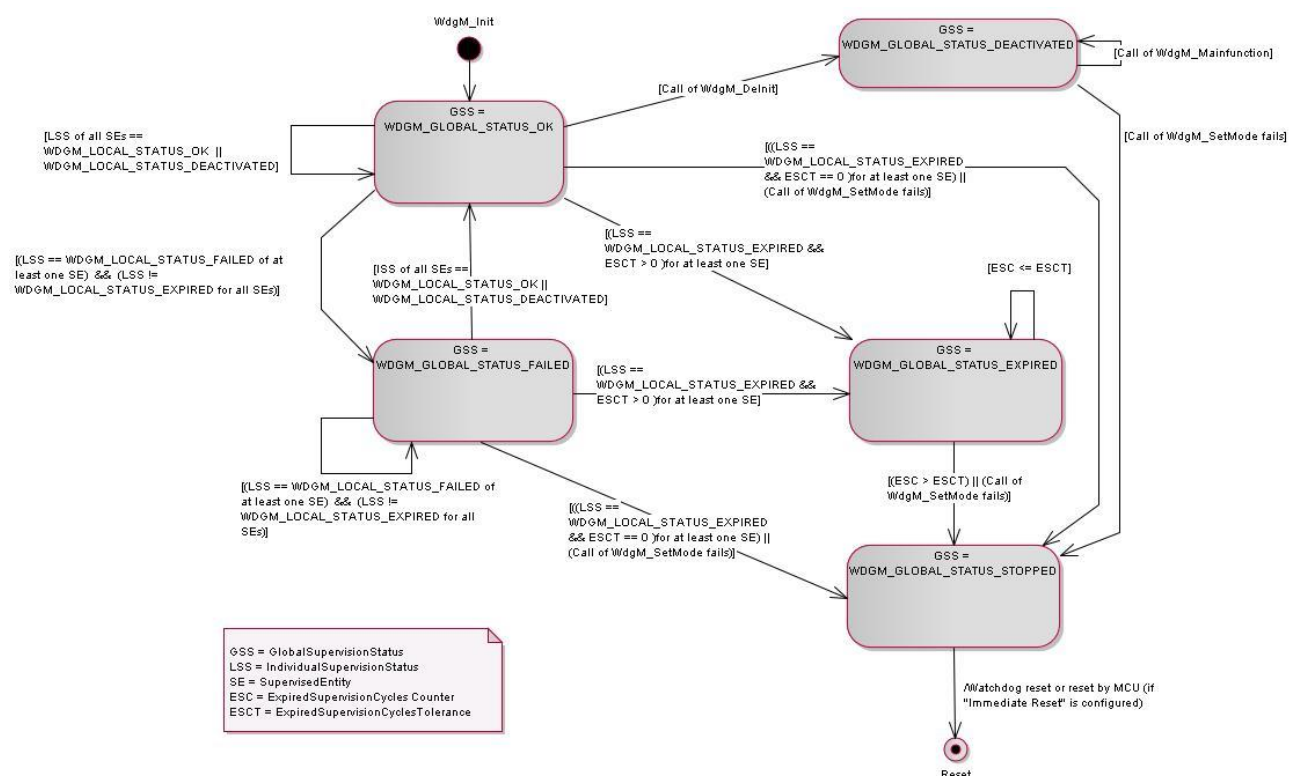


Figure 3-1 State machine overview of the Global Supervision Status

Global Supervision Status	Activities	Point in Time
WDGM_GLOB AL_STATUS_D EACTIVATED	No Alive Supervision is performed. Watchdogs are not managed by WdgM.	> Initial state before Initialization and after de-initialization.

WDGM_GLOB AL_STATUS_OK	Perform Alive Supervision of all enabled Supervised Entities and refresh watchdog trigger condition of the watchdog instances configured.	<ul style="list-style-type: none"> > Initial state after successful initialization > All Supervised Entities matches its configured Expected Alive Indications (including margins)
WDGM_GLOB AL_STATUS_FAILED	Perform Alive Supervision of all enabled Supervised Entities and refresh watchdog trigger condition of the watchdog instances configured.	<ul style="list-style-type: none"> > At least one Supervised Entity does not match its configured Expected Alive Indications (including margins). Additionally, the Failed Reference Cycle Tolerance value of this Supervised Entity is greater than 0. > The Individual Supervision Status of no Supervised Entity is equal to the state WDGM_LOCAL_STATUS_EXPIRED
WDGM_GLOB AL_STATUS_EXPIRED	Perform Alive Supervision of all enabled Supervised Entities and refresh watchdog trigger condition of the watchdog instances configured, but only a transition to WDGM_GLOBAL_STATUS_STOPPED is possible.	<ul style="list-style-type: none"> > The Individual Supervision Status of at least one Supervised Entity is equal to WDGM_LOCAL_STATUS_EXPIRED . I.e. the Failed Reference Cycle Tolerance value of this Supervised Entity has exceeded.
WDGM_GLOB AL_STATUS_STOPPED	Stop Alive Supervision of all Supervised Entities and deny permission for watchdog triggering. A watchdog reset will occur shortly or an immediate reset is performed, depending on configuration.	<ul style="list-style-type: none"> > The configured Expired Supervision Cycles Tolerance value has exceeded.

Table 3-3 Global Supervision Status

3.3.2 Local Supervision Status

Each Supervised Entity supervised by the WDGM is handled by its Local Supervision State which represents the current result of Alive Supervision.

The Global Supervision Status is calculated by the WDGM depending on the Local Supervision Status of all activated Supervised Entities.

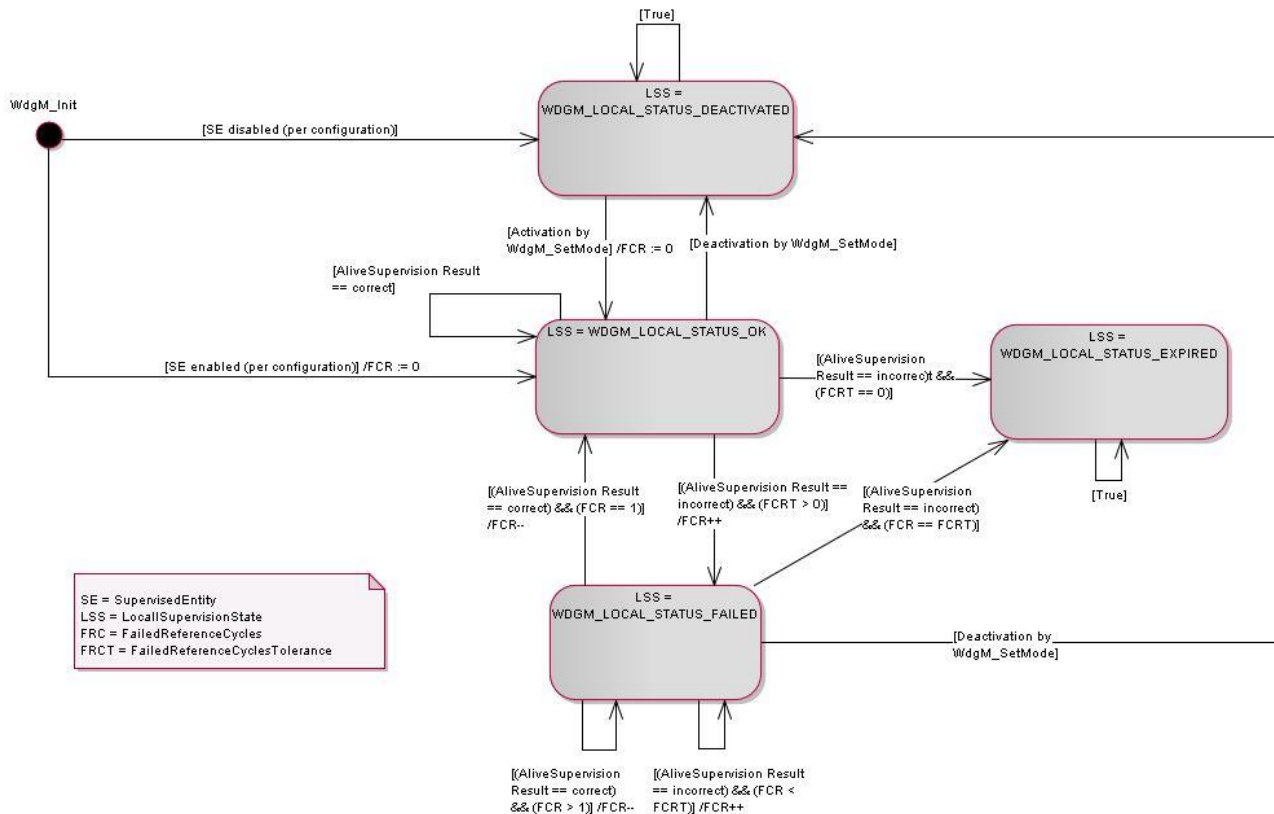


Figure 3-2 State machine overview of the Local Supervision Status

Individual Supervision Status	Activities	Point in Time
WDMG_LOCAL_STATUS_DEACTIVATED	Alive Supervision of Supervised Entity is disabled.	> Supervised Entity is deactivated in the current WdgM mode after initialization or mode switch
WDMG_LOCAL_STATUS_OK	Perform Alive Supervision of the Supervised Entity.	> Supervised Entity is activated and exposed to Alive Supervision. > The Supervised Entities matches its configured Expected Alive Indications (including margins)
WDMG_LOCAL_STATUS_FAILED	Perform Alive Supervision of the Supervised Entity.	> Supervised Entity is activated and exposed to Alive Supervision. > The Supervised Entity does not match its configured Expected Alive Indications (including margins) and the Failed Reference Cycle Tolerance value is greater than 0.

WDGM_LOCA L_STATUS_ EXPIRED	Ignore Alive Supervision of the Supervised Entity.	> The Failed Reference Cycle Tolerance value configured is exceeded of this Supervised Entity.
-----------------------------------	--	--

Table 3-4 Individual Supervision Status

3.3.3 Module States

Module State	Activities	Point in Time
Uninitialized	Calling any API service (except WdgM_Init() WdgM_GetFirstExpiredSEID() and WdgM_GetVersionInfo()) is not allowed.	The API service WdgM_Init() has not been called yet or WdgM_DeInit has been called.
Initialized	The Watchdog Manager is initialized and calling any API service is allowed. The Watchdog Manager performs its work/task configured.	The API service WdgM_Init() has been called and has been finished successfully.

Table 3-5 Module states

3.4 Main Functions

3.4.1 Alive Supervision

The Alive Supervision of the Watchdog Manager offers a mechanism to periodically check the execution reliability of one or several Supervised Entities. This mechanism supports a checkup of cyclic timing constraints of independent Supervised Entities, tracks the checkup result as their Local Supervision Status and provides a Global Supervision Status to support decisions for refreshing the timeouts of the watchdog instances or not. Supervised entities in this context are entities of the application layer which are under supervision of the Watchdog Manager and their status is denoted as the Individual Supervision Status.

3.5 Error Handling

3.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service Det_ReportError() as specified in [2], if development error reporting is enabled.

The reported service IDs identify the services of WDMG which are described in [1]. The following table presents the service IDs and the related services:

Service ID	Service
0x00	WdgM_Init
0x01	WdgM_DeInit
0x02	WdgM_GetVersionInfo

Service ID	Service
0x03	WdgM_SetMode
0x04	WdgM_UpdateAliveCounter
0x0B	WdgM_GetMode
0x0C	WdgM_GetLocalStatus
0x0D	WdgM_GetGlobalStatus
0x0E	WdgM_CheckpointReached
0x0F	WdgM_PerformReset
0x10	WdgM_GetFirstExpiredSEID

Table 3-6 Mapping of service IDs to services

The errors reported to DET are described in the following table:

Error Code		Description
0x10	WDGM_E_NO_INIT	API service used in wrong context, i.e. WDM has not been initialized yet.
0x11	WDGM_E_PARAM_CONFIG	API service WdgM_Init() called with "NULL pointer" parameter.
0x12	WDGM_E_PARAM_MODE	API service WdgM_SetMode() called with wrong "Mode" parameter.
0x13	WDGM_E_PARAM_SEID	API service called with wrong "Supervised Entity identifier" parameter.
0x14	WDGM_E_INV_POINTER	API service called with "NULL pointer" parameter
0x15	WDGM_E_DISABLE_NOT_ALLOWED	Disabling of watchdog not allowed (e.g. in safety relevant systems). This error code is not used by the Watchdog Manager, because the error can be caught while configuration time.
0x16	WDGM_E_CPID	API service WdgM_CheckpointReached() called with an invalid CheckpointId.
0x17	WDGM_E_DEPRECATED	Deprecated API service WdgM_UpdateAliveCounter() was used.
0x18	WDGM_E_AMBIGIOUS	Function WdgM_UpdateAliveCounter() cannot determine the Checkpoint, because there are more than one alive supervisions configured in the current mode for the given Supervised Entity.
0x19	WDGM_E_SEDEACTIVATED	API service used with a checkpoint of a Supervised Entity that is deactivated in the current Watchdog Manager mode.

Error Code		Description
0x24	WDGM_E_ILLEGAL_CALLBACK_HANDLING	<p>One of the API services WdgM_UpdateAliveCounter() or WdgM_CheckpointReached() is called while a callback routine is running, used to report a change of a Individual Supervision Status of a Supervised Entity. This is used to prevent the manipulation of the status of a Supervised Entity.</p> <p>This error code is created additionally to the AUTOSAR specification.</p>

Table 3-7 Errors reported to DET

3.5.1.1 Parameter Checking

AUTOSAR requires that API functions check the validity of their parameters. The checks in Table 3-8 are internal parameter checks of the API functions. These checks are for development error reporting and can be en-/disabled separately. En-/disabling of single checks are an addition to the AUTOSAR standard which requires to en-/disable the complete parameter checking via the parameter WDGM_DEV_ERROR_DETECT.

The following table shows which parameter checks are performed on which services:

Check/Error code	WDGM_E_NO_INIT	WDGM_E_PARAM_CONFIG	WDGM_E_PARAM_MODE	WDGM_E_PARAM_SEID	WDGM_E_INV_POINTER	WDGM_E_DISABLE_NOT_ALLOWED	WDGM_E_CPID	WDGM_E_DEPRECATED	WDGM_E_AMBIGUOUS	WDGM_E_SEDEACTIVATED	WDGM_E_ILLEGAL_CALLBACK_HANDLING
Service											
WdgM_Init		■									
WdgM_DeInit	■										
WdgM_GetVersionInfo					■						
WdgM_SetMode	■		■								

Check/Error code											
Service	WDGM_E_NO_INIT	WDGM_E_PARAM_CONFIG	WDGM_E_PARAM_MODE	WDGM_E_PARAM_SEID	WDGM_E_INV_POINTER	WDGM_E_DISABLE_NOT_ALLOWED	WDGM_E_CPID	WDGM_E_DEPRECATED	WDGM_E_AMBIGIOUS	WDGM_E_SEDEACTIVATED	WDGM_E_ILLEGAL_CALLBACK_HANDLING
WdgM_GetMode	■				■						
WdgM_CheckpointReached	■			■			■			■	■
WdgM_UpdateAliveCounter	■			■				■	■		■
WdgM_GetLocalStatus	■			■	■						
WdgM_GetGlobalStatus	■				■						
WdgM_PerformReset	■										
WdgM_GetFirstExpiredSEID					■						
WdgM_MainFunction	■										

Table 3-8 Development Error Reporting: Assignment of checks to services


Detected development errors are reported to the development error tracer by default. The called service is aborted immediately if an error has been detected. Services with return value of type Std_ReturnType return the value E_NOT_OK. Services with return type void just abort execution of the function and return to the caller.

3.5.2 Production Code Error Reporting

By default, production code related errors are reported to DEM using the service `Dem_ReportErrorStatus()` (specified in [3]), if production error reporting is enabled.


If another module is used for production code error reporting, the function prototype for reporting the error can be configured by the user/integrator, but must have the same signature as the service `Dem_ReportErrorStatus()`.

The errors reported to DEM are described in the following table:

Error Code	Description
WDGM_E_MONITORING	<p>Alive Supervision of at least one Supervised Entity has failed.</p> <p>The production code error is already reported if the Global Supervision Status traverses to state EXPIRED for the first time. If this would be done when the status traverses to state STOPPED as required by the Watchdog Manager specification [1], there would be definitely not enough time to store the error on the non-volatile memory.</p>
WDGM_E_SET_MODE	<p>Mode-setting of the Watchdog Manager has failed, i.e. the API service <code>WdgM_SetMode()</code> is called while the Global Supervision Status is different to <code>WDGM_GLOBAL_STATUS_OK</code> or <code>WDGM_GLOBAL_STATUS_FAILED</code>.</p> <div>  <p>Info The Watchdog Manager specification [1] requires additionally reporting this failure if mode-switching of the watchdog instance has failed. According to the specification of the watchdog driver (refer to [6]), a production error is already reported by the watchdog driver in that case. Hence, it would be redundant if the Watchdog Manager reports the same error again. Consequently, no error is reported to the DEM (by default) by the Watchdog Manager if mode-switching of a watchdog instance has failed.</p> </div>

WDGM_E_IMPROPER_CALLER	<p>Defensive behavior checks have detected an improper caller of the service WdgM_SetMode(). That means that the value of the parameter CallerID does not exists caller ID list of the configuration.</p> <p>The check can be activated by the configuration switch "Defensive Behaviour".</p>
------------------------	--

Table 3-9 Errors reported to DEM

	<p>Info</p> <p>The production error codes listed above must be defined by the DEM and are included into the Watchdog Manager by the file Dem.h (by default).</p>
---	---

4 Integration

This chapter gives detailed information for the integration of the MICROSAR WDGM into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the WDGM contains the files which are described in the chapters 4.1.1 and 4.1.2.

4.1.1 Static Files

File Name	Description
WdgM.c	Contains the implementation of the interfaces of the WDGM.
WdgM.h	Declares the interface of the WDGM.
WdgM_bswmd.arxml	Contains the formal notation of all information, which belongs to the WDGM.

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration and generation tool MICROSAR EAD or DaVinci Configurator Pro.

File Name	Description
WdgM_Cfg.h	Contains the static configuration part of the WDGM.
WdgM_PrivateCfg.h	Contains configuration data which is only relevant for the WDGM implementation. This file must be only included in the WDGM implementation files.
WdgM_PBCfg.c	Contains the post-build configuration information of this module.
<ServiceComponentName>_swc.arxml	Contains the configuration parameters of the Software Components which are necessary to generate the RTE. The Service Component Name, configured within the WDGM, serves as prefix of the generated file name.
<Projectname>_ecuc.arxml	Contains ECU specific information which is written to by several configuration tools.

Table 4-2 Generated files

4.2 Include Structure

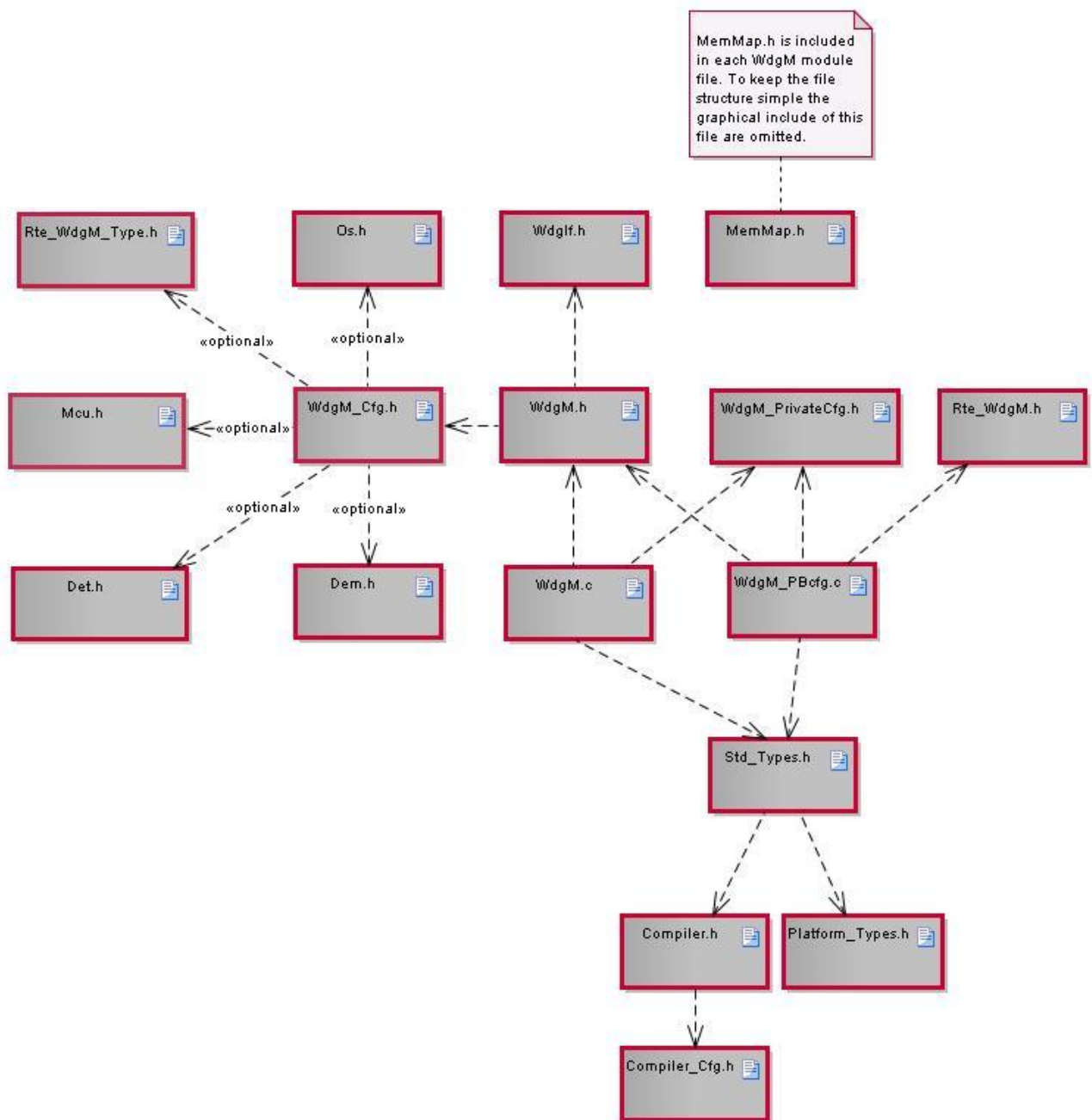


Figure 4-1 Include structure

4.3 Compiler Abstraction and Memory Mapping

The objects (e.g. variables, functions, constants) are declared by compiler independent definitions – the compiler abstraction definitions. Each compiler abstraction definition is assigned to a memory section.

The following table contains the memory section names and the compiler abstraction definitions defined for the WDM and illustrate their assignment among each other.

Compiler Abstraction Definitions								
	WDGM_CODE	WDGM_VAR	WDGM_VAR_NOINIT	WDGM_DATA	WDGM_APPL_DATA	WDGM_APPL_CODE	WDGM_CONST	WDGM_PBCFG
Memory Mapping Sections								
WDGM_START_SEC_CODE	■							
WDGM_START_SEC_VAR_INIT_UNSPECIFIED		■						
WDGM_START_SEC_VAR_NO_INIT_UNSPECIFIED			■	■				
WDGM_START_SEC_CONST_UNSPECIFIED							■	
WDGM_START_SEC_CONST_32							■	
WDGM_START_SEC_CONFIG_DATA_UNSPECIFIED								■
Application code memory section - access from WDGM into application memory						■		
Application buffer memory section - access from WDGM into application memory					■			

Table 4-3 Compiler abstraction and memory mapping

4.4 Dependencies on SW modules

4.4.1 AUTOSAR OS

The WDGM module depends on the AUTOSAR OS. The WDGM can either use the locking mechanisms of OS or SchM for critical section handling.

4.4.2 DEM

The WDGM depends on the DEM in order to report production errors.

4.4.3 DET

The WDGM depends on the DET (by default) in order to report development errors.

The reporting is only available in case the detection is enabled. The reporting can also be disabled in case the detection is enabled.

The DET can be replaced optionally by an equivalent component which is responsible to recognize development errors, if no DET component is available.

4.4.4 WDGIF

The WDGM depends on the Watchdog Interface in order to set the timeout for permission to trigger for the configured watchdog instances via the WDGIF. The service used therefore is `WdgIf_SetTriggerCondition()`. Furthermore, the mode of watchdog instances is set via the WDGIF and its service `WdgIf_SetMode()` on demand.

4.4.5 SchM

The BSW Scheduler calls the cyclical function `WdgM_MainFunction()` and provides services for critical section handling. Since ASR 4.0 the SchM interface is integrated within the RTE and not a stand-alone module anymore.

4.4.6 MCU

The WDGM depends on the MCU (by default) in order to perform an immediate reset of the ECU in case of Alive Supervision failure.

The MCU can optionally be excluded from the WDGM dependent components by disabling the switch "Enable Immediate Reset" within "Module Dependencies".

The MCU can be replaced optionally by an equivalent component which is responsible to perform the reset instead, if no MCU component is available.

4.4.7 RTE

The WDGM depends on the RTE (by default) in order to notify the mode change.

Notification of mode change can be enabled or disabled by the switch "Enable Mode Switch Notification" on the perspective "Module Dependencies" within the WDGM module. In case notification is used it can be chosen between notifications via Rte or user defined callback function ("Module Dependencies"/ "Enable RTE Mode Switch Notification").

Furthermore, since ASR 4.0, the RTE provides the interface to the module SchM for cyclic main function call and critical section handling.

4.5 Dependencies on HW modules

The Watchdog Manager is hardware independent.

5 API Description

5.1 Interfaces Overview

See chapter 2.1.

5.2 Type Definitions

Type Name	C-Type	Description	Value Range
WdgM_Supervised EntityIdType	uint8 / uint16	<p>This type identifies a Supervised Entity for the Watchdog Manager.</p> <p>The type depends on the number of configured Supervised Entities. It is determined automatically. In case the number of Supervised Entities reaches 256 the corresponding type is uint16 else uint8 is used instead.</p>	0-254 (uint8)
			0-65534 (uint16)
WdgM_CheckpointIdType	uint8 / uint16	<p>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.</p> <p>It is determined automatically. In case the number of Checkpoint reaches 256 the corresponding type is uint16 else uint8 is used instead.</p>	0-254 (uint8)
			0-65534 (uint16)
WdgM_LocalStatus Type	uint8	<p>This type shall be used for variables that represent the current status of supervision for individual Supervised Entities.</p>	WDGM_LOCAL_STATUS_OK
			<p>Refer to chapters 3.3.2 for more detailed information.</p> <p>WDGM_LOCAL_STATUS_FAILED</p> <p>Refer to chapters 3.3.2 for</p>

Type Name	C-Type	Description	Value Range
			more detailed information.
			WDGM_LOCAL_STATUS_EXPIRED Refer to chapters 3.3.2 for more detailed information.
			WDGM_LOCAL_STATUS_DEACTIVATED Refer to chapter 3.3.2 for more detailed information.
WdgM_GlobalStatusType	uint8	This type shall be used for variables that represent the global supervision status of the Watchdog Manager module.	WDGM_GLOBAL_STATUS_OK Refer to chapters 3.3.1 for more detailed information. WDGM_GLOBAL_STATUS_FAILED Refer to chapters 3.3.1 for more detailed information. WDGM_GLOBAL_STATUS_EXPIRED Refer to chapters 3.3.1 for more detailed information. WDGM_GLOBAL_STATUS_STOPPED Refer to chapter 3.3.1 for more detailed information. WDGM_GLOBAL_STATUS_DEACTIVATED Refer to chapter 3.3.1 for more detailed information.
WdgM_ModeType	uint8	This type distinguishes the different modes that were configured for the Watchdog Manager.	0-255

Table 5-1 Type definitions

5.3 Services provided by WDM

The WDM API consists of services, which are realized by function calls.

5.3.1 WdgM_Init

Prototype	
void WdgM_Init (const WdgM_ConfigType *ConfigPtr)	
Parameter	
ConfigPtr	Pointer to configuration set.
Return code	
void	--
Functional Description	
<p>This routine initializes the Watchdog Manager, i.e. it sets the default Watchdog Manager mode as provided in the configuration set. Furthermore, module local variables are initialized and the module is set to initialized state.</p> <p>After execution of this routine, Alive Supervision is activated according to the list of Supervised Entities defined in the configuration set.</p> <p>Also the first expired Supervised Entity which caused the last watchdog reset is cleared in this service.</p> <p>Additionally, if development mode is configured, parameter checks are done and in case of failure they are reported to the DET with the according service ID and the reason of occurrence (see chapter 3.5.1.1).</p>	
Particularities and Limitations	
<ul style="list-style-type: none"> ■ This service is synchronous. ■ This service is non-reentrant. ■ This service is always available. ■ A parameter check of this service is switched on/off via the configuration switch 'Check ConfigPtr Parameter'. ■ This service can support one or multiple watchdog instance, depending on configuration of the Watchdog Manager. 	
Expected Caller Context	
<ul style="list-style-type: none"> ■ Expected to be called in application context. 	

Table 5-2 WdgM_Init

5.3.2 WdgM_DeInit

Prototype	
void WdgM_DeInit ()	
Parameter	
void	--

Return code	
Void	--
Functional Description	
<p>This routine de-initializes the Watchdog Manager. It sets the global status to WDM_GLOBAL_STATUS_DEACTIVATED and sets the trigger condition for all watchdog instances for a last time with the timeout parameter defined in the mode that is configured as Delnit Mode.</p> <p>After execution of this routine, Alive Supervision is activated according to the list of Supervised Entities defined in the configuration set.</p> <p>Additionally, if development mode is configured, parameter checks are done and in case of failure they are reported to the DET with the according service ID and the reason of occurrence (see chapter 3.5.1.1).</p>	
Particularities and Limitations	
<ul style="list-style-type: none"> ■ This service is synchronous. ■ This service is non-reentrant. ■ This service is always available. ■ Checking whether the module has been initialized can be switched off/on via the configuration switch 'Check Initialized Module'. 	
Expected Caller Context	
<ul style="list-style-type: none"> ■ Expected to be called in application context. 	

Table 5-3 WdgM_Delnit

5.3.3 WdgM_GetVersionInfo

Prototype	
void WdgM_GetVersionInfo (Std_VersionInfoType *VersionInfo)	
Parameter	
VersionInfo	Pointer to where to store the version information of the module WdgM.
Return code	
Void	--
Functional Description	
<p>This service returns the version information of the Watchdog Manager.</p> <p>Additionally, if development mode is configured, parameter checks are done and in case of failure they are reported to the DET with the according service ID and the reason of occurrence (see chapter 3.5.1.1).</p>	

Particularities and Limitations

- This service is synchronous.
- This service is non-reentrant.
- This service is available, depending on configuration switch 'Enable WdgM_GetVersionInfo API'.
- A parameter check of this service is switched on/off via the configuration switch 'Check VersionInfo Parameter'.

Expected Caller Context

- Expected to be called in application context.

Table 5-4 WdgM_GetVersionInfo

5.3.4 WdgM_SetMode

Prototype

Std_ReturnType **WdgM_SetMode** (WdgM_ModeType Mode, uint16 CallerID)

Parameter

Mode	One of the configured Watchdog Manager modes.
CallerID	Module ID of the calling module.

Return code

Std_ReturnType	> E_OK	Successfully changed to the new mode.
	> E_NOT_OK	Changing to new mode failed.

Functional Description

The Watchdog Manager can be switched between different modes that are statically configured and contained in the Watchdog Manager configuration set provided to the initialization routine.

On failure, the function stops execution immediately and returns to the caller.

Additionally, if development mode is configured, parameter checks are done and in case of failure they are reported to the DET with the according service ID and the reason of occurrence (see chapter 3.5.1.1).



Info

Modes are normally configured for different phases of the ECU, e.g. Startup, Run or Shutdown. The ECUM sets these Modes at the corresponding phase of the ECU.

Within each Mode, it can be configured/selected by the user:

- > the Expired Supervision Reference Cycles
- > which Supervised Entities shall be supervised by the WDGM, which can be a none or subset of all available Supervised Entities
- > the Alive Supervision parameters of the selected Supervised Entities
- > the watchdog instances that shall be served and their watchdog modes
- > the timeout value of the selected watchdog instances

Particularities and Limitations

- This service is synchronous.
- This service is non-reentrant.
- This service is always available.
- Checking whether the module has been initialized can be switched off/on via the configuration switch 'Check Initialized Module'.
- A parameter check of this service is switched on/off via the configuration switch 'Check Mode Parameter'.
- A defensive Behaviour check of the caller ID is switched on/off via the configuration switch 'Defensive Behaviour'.

Expected Caller Context

- Expected to be called in application context.

Table 5-5 WdgM_SetMode

5.3.5 WdgM_GetMode

Prototype		
Std_ReturnType WdgM_GetMode (WdgM_ModeType *Mode)		
Parameter		
Mode	Current mode of the Watchdog Manager.	
Return code		
Std_ReturnType	> E_OK	Current mode successfully returned.
	> E_NOT_OK	Returning current mode failed.
Functional Description		
Returns the current mode of the Watchdog Manager.		
Additionally, if development mode is configured, parameter checks are done and in case of failure they are reported to the DET with the according service ID and the reason of occurrence (see chapter 3.5.1.1).		
Particularities and Limitations		
<ul style="list-style-type: none">■ This service is synchronous.■ This service is reentrant.■ This service is always available.■ Checking whether the module has been initialized can be switched off/on via the configuration switch 'Check Initialized Module'.■ A parameter check of this service is switched on/off via the configuration switch 'Check Mode Parameter'.		
Expected Caller Context		
<ul style="list-style-type: none">■ Expected to be called in application context.		

Table 5-6 WdgM_GetMode

5.3.6 WdgM_CheckpointReached

Prototype		
Std_ReturnType WdgM_CheckpointReached (WdgM_SupervisedEntityType SEID, WdgM_CheckpointIDType CheckpointID)		
Parameter		
SEID	Identifier of the Supervised Entity under control of the WdgM whose Alive Counter shall be updated.	
CheckpointID	Identifier of the Checkpoint within a Supervised Entity that has been reached.	
Return code		
Std_ReturnType	> E_OK	Successfully updated the Alive Counter.


	> E_NOT_OK	Updating failed.
Functional Description		
This routine shall be used by the Supervised Entities under control of the Watchdog Manager to indicate that a Checkpoint has been reached. The Alive Indication counter is updated with this service.		
<div>  <div> Note This service shall replace WdgM_UpdateAliveCounter() in ASR4.0. </div> </div>		
Additionally, if development mode is configured, parameter checks are done and in case of failure they are reported to the DET with the according service ID and the reason of occurrence (see chapter 3.5.1.1).		
Particularities and Limitations		
<ul style="list-style-type: none"> ■ This service is synchronous. ■ This service is reentrant. ■ This service is always available. ■ Checking whether the module has been initialized can be switched off/on via the configuration switch 'Check Initialized Module'. ■ A parameter check of this service is switched on/off via the configuration switch 'Check Supervised Entity ID Parameter'. ■ A parameter check of this service is switched on/off via the configuration switch 'Check Checkpoint ID Parameter'. 		
Expected Caller Context		
<ul style="list-style-type: none"> ■ Expected to be called in application context. 		

Table 5-7 WdgM_CheckpointReached

5.3.7 WdgM_UpdateAliveCounter

Prototype		
Std_ReturnType WdgM_UpdateAliveCounter (WdgM_SupervisedEntityType SEID)		
Parameter		
SEID	Identifier of the Supervised Entity under control of the WdgM whose Alive Counter shall be updated.	
Return code		
Std_ReturnType	> E_OK	Successfully updated the Alive Counter.
	> E_NOT_OK	Updating the Alive Counter failed.

Functional Description

This routine can be used by the Supervised Entities under control of the Watchdog Manager to give Alive Indications to the Watchdog Manager.



Caution

This function is deprecated and should not be used anymore. It is only provided for backward compatibility. Use `WdgM_CheckpointReached()` instead! .

Additionally, if development mode is configured, parameter checks are done and in case of failure they are reported to the DET with the according service ID and the reason of occurrence (see chapter 3.5.1.1).

Particularities and Limitations

- This service is synchronous.
- This service is reentrant.
- This service is always available.
- Checking whether the module has been initialized can be switched off/on via the configuration switch 'Check Initialized Module'.
- Reporting that this deprecated service is used can be switched on/off via the configuration switch 'Check Usage of Deprecated Service'.
- Checking whether the module has been initialized can be switched off/on via the configuration switch 'Check Initialized Module'.
-

Expected Caller Context

- Expected to be called in application context.

Table 5-8 WdgM_UpdateAliveCounter

5.3.8 WdgM_GetLocalStatus

Prototype

```
Std_ReturnType WdgM_GetLocalStatus
(
    WdgM_SupervisedEntityType SEID,
    WdgM_LocalStatusType* Status
)
```

Parameter

SEID	Identifier of the Supervised Entity whose status shall be returned.
------	---

Status	Status of the given Supervised Entity.	
Return code		
Std_ReturnType	> E_OK	Current Supervision Status successfully returned.
	> E_NOT_OK	Returning current Supervision Status failed.
Functional Description		
<p>Returns the current local Supervision Status of the given Supervised Entity.</p> <p>Additionally, if development mode is configured, parameter checks are done and in case of failure they are reported to the DET with the according service ID and the reason of occurrence (see chapter 3.5.1.1).</p> <p>Refer to chapter 3.3.2 for more details about the Local Supervision Status.</p>		
Particularities and Limitations		
<ul style="list-style-type: none">■ This service is synchronous.■ This service is reentrant.■ This service is always available.■ Checking whether the module has been initialized can be switched off/on via the configuration switch 'Check Initialized Module'.■ A parameter check of this service is switched on/off via the configuration switch 'Check Supervised Entity ID Parameter'.■ A parameter check of this service is switched on/off via the configuration switch 'Check Status Parameter'.		
Expected Caller Context		
<ul style="list-style-type: none">■ Expected to be called in application context.		

Table 5-9 WdgM_GetLocalStatus

5.3.9 WdgM_GetGlobalStatus

Prototype		
Std_ReturnType WdgM_GetGlobalStatus (WdgM_AliveSupervisionStatusType* Status)		
Parameter		
Status	Global Supervision Status of the Watchdog Manager.	
Return code		
Std_ReturnType	> E_OK	Current Global Supervision Status successfully returned.
	> E_NOT_OK	Returning Global Supervision Status failed.

Functional Description

Returns the current Global Supervision Status of the Watchdog Manager.

Additionally, if development mode is configured, parameter checks are done and in case of failure they are reported to the DET with the according service ID and the reason of occurrence (see chapter 3.5.1.1).

Refer to chapter 3.3.1 for more details about the Global Supervision Status.

Particularities and Limitations

- This service is synchronous.
- This service is reentrant.
- This service is always available.
- Checking whether the module has been initialized can be switched off/on via the configuration switch 'Check Initialized Module'.
- A parameter check of this service is switched on/off via the configuration switch 'Check Status Parameter'.

Expected Caller Context

- Expected to be called in application context.

Table 5-10 WdgM_GetGlobalStatus

5.3.10 WdgM_PerformReset

Prototype

void **WdgM_PerformReset()**

Parameter

void	--
------	----

Return code

void	--
------	----

Functional Description

This routine instructs the Watchdog Manager to cause an immediate watchdog reset.

Additionally, if development mode is configured, parameter checks are done and in case of failure they are reported to the DET with the according service ID and the reason of occurrence (see chapter 3.5.1.1).

Particularities and Limitations

- This service is synchronous.
- This service is non-reentrant.
- This service is always available.
- Checking whether the module has been initialized can be switched off/on via the configuration switch 'Check Initialized Module'.

Expected Caller Context
<ul style="list-style-type: none"> Expected to be called in application context.

Table 5-11 WdgM_PerformReset

5.3.11 WdgM_GetFirstExpiredSEID

Prototype		
Std_ReturnType WdgM_GetFirstExpiredSEID (WdgM_SupervisedEntityType* SEID)		
Parameter		
SEID	Identifier of the supervised entity that first reached the state WDGM_LOCAL_STATUS_EXPIRED.	
Return code		
Std_ReturnType	> E_OK	SEID successfully returned
	> E_NOT_OK	Error when returning the SEID
Functional Description		
<p>This service returns SEID that first reached the state WDGM_LOCAL_STATUS_EXPIRED.</p> <p>Usable results are available when the WDGM_GLOBAL_STATUS_EXPIRED has been reached and the module has not been re-initialized. That means also, that after a watchdog reset the information about the first expired supervised entity is still available if memory mapping was done correctly.</p> <p>Additionally, if development mode is configured, parameter checks are done and in case of failure they are reported to the DET with the according service ID and the reason of occurrence (see chapter 3.5.1.1).</p>		
Particularities and Limitations		
<ul style="list-style-type: none">■ This service is synchronous.■ This service is reentrant.■ This service is always available.■ A parameter check of this service is switched on/off via the configuration switch 'Check Pointer Parameter'.		
Expected Caller Context		
<ul style="list-style-type: none">■ Expected to be called in application context.		

Table 5-12 WdgM_GetFirstExpiredSEID

5.3.12 WdgM_MainFunction

Prototype
void WdgM_MainFunction (void)

Parameter	
void	--
Return code	
void	--
Functional Description	
<p>This Main-Function checks the aliveness of the Supervised Entities which are under supervision of the Watchdog Manager. Depending on the Supervised Entities and whether they behave as expected, the WDM allows the configured watchdog instances to be triggered for a certain amount of time according to their configuration parameters. Otherwise the Watchdog timeouts are set to zero and therefore one or more watchdogs will expire which results in a reset of the software.</p> <p>Additionally, if development mode is configured, the call of this API service is checked for faulty call conditions and the failure is reported to the DET with the according service ID and the reason of occurrence (see chapter 3.5.1.1).</p>	
Particularities and Limitations	
<ul style="list-style-type: none"> ■ This service is synchronous. ■ This service is non-reentrant. ■ Checking whether the module has been initialized can be switched off/on via the configuration switch 'Check Initialized Module'. 	
Expected Caller Context	
<ul style="list-style-type: none"> ■ Expected to be called in application context. 	

Table 5-13 WdgM_MainFunction

5.4 Callback Functions

The module does not provide any callback function.

5.5 Configurable Interfaces

API	Description
Function WdgM_GetVersionInfo()	This function can be enabled/disabled by the configuration switch 'Enable WdgM_GetVersionInfo API'.

Table 5-14 Configurable interfaces

5.6 Service Ports

5.6.1 Client Server Interface

A client server interface is related to a Provide Port (PPort) at the server side and a Require Port (RPort) at client side.

The WDM provides PPorts for the services. More detailed information can be retrieved in chapter 5.6.1.1.

Interface name	Comment
WdgM_AliveSupervision	Specified by [1].
WdgM_IndividualMode	Specified by [1].
WdgM_GlobalMode	Specified by [1] .

Table 5-15 Adaptation of Provide Ports

Through the PPorts the Software Components (SW-C) have the possibility to execute services of the WDM with an abstract RTE interface. Hence, the Software Components are independent from the underlying basic software stack.

5.6.1.1 Provide Ports on WDM Side

At the PPorts of the WDM some of the API functions, which are described in 5.3, are available as Runnable Entities. The Runnable Entities are invoked via Operations. The mapping from a SW-C client call to an Operation is performed by the RTE. In this mapping the RTE adds Port Defined Argument Values to the client call of the SW-C, if configured.

The following sub-chapters presents the PPorts defined for the WDM and the Operations defined for the PPorts, the API functions related to the Operations and the Port Defined Argument Values to be added by the RTE.

5.6.1.1.1 WdgM_AliveSupervision

Operation	WDM API Function	Port Defined Argument Values
UpdateAliveCounter	WdgM_UpdateAliveCounter()	WdgM_SupervisedEntityType SEID
CheckpointReached	WdgM_CheckpointReached()	WdgM_SupervisedEntityType SEID

Table 5-16 WdgM_AliveSupervision

5.6.1.2 Require Ports on WDM Side

At its Require Ports the WDM calls Operations. These Operations have to be provided by the SWCs by means of Runnable Entities. These Runnable Entities implement the callback functions expected by the WDM.

The following sub-chapters present the Require Ports defined for the WDGM, the Operations that are called from the WDGM and the related Notifications, which are described in chapter 0.

5.6.1.2.1 WdgM_IndividualMode

Operation	Notification
Local Alive Supervision mode state change	Rte_Switch_WdgM_mode000_<SE-SymbolicName>_currentMode

Table 5-17 WdgM_IndividualMode

5.6.1.2.2 WdgM_GlobalMode

Operation	Notification
Global mode state change	Rte_Switch_WdgM_globalMode_currentMode()

Table 5-18 WdgM_IndividualMode

6 Configuration

The WDM attributes shall be configured in DaVinci Configurator Pro.

6.1 Configuration Variants

The WDM supports the configuration variants

> VARIANT-POST-BUILD

The configuration classes of the WDM parameters depend on the supported configuration variants.

6.2 Configuration of WDM

The WDM is configured with the help of the configuration tool DaVinci Configurator.

The settings for standard use-cases can be done in the “Watchdogs”-Configuration Editor that can be found in the domain “Mode Management”.

For more specific settings the Basic Editor contains all configurable parameters of the WDM.

If the WDM neither appear in the list of Configuration Editors nor in the Basic Editor the module has to be added in the Project Settings Editor (Settings->Modules).



Info

In case of object delivery modifications are without effect, if a parameter is specified as a pre-compile value. Different object code is needed for different settings. Other object files can be obtained at Vector Informatik.

7 AUTOSAR Standard Compliance

7.1 Additions/ Extensions

7.1.1 Parameter Checking

The internal parameter checks of the API functions can be en-/disabled separately. The AUTOSAR standard requires en-/disabling of the complete parameter checking only. For details see chapter 3.5.1.1.

7.2 Limitations

7.2.1 Link-time configuration is not supported

The Watchdog Manager supports only post-build configuration and no link-time configuration.

The consequence of non-observance is that the WdgM_Init() service always requires a pointer to the configuration set which shall be used by the Watchdog Manager.

7.2.2 Deadline and Logical Supervision are not supported

The current sight of Vector is that Deadline and Logical Supervision is only necessary for safety relevant systems. Therefore those features with all their configuration parameters are not implemented yet.

7.2.3 Partition reset is not supported

Resetting of the corresponding partition of an OS application by WDGM is not supported because Basic Software Mode Manager (BSWM) does not support this feature yet.

8 Glossary and Abbreviations

8.1 Glossary

Term	Description
DaVinci Configurator	<p>DaVinci Configurator is a tool for creating ECU configuration descriptions according to the AUTOSAR standard. It includes specific functionality for configuring AUTOSAR basic software modules and the RTE.</p> <p>DaVinci Configurator is delivered as part of a Vector Software Integration Package (SIP). The available tool functionality depends on the tool license and the SIP license.</p>
Supervised Entity	The Watchdog Manager supervises the execution of software. The logical units of supervision are Supervised Entities. Typically a Supervised Entity may represent one SW-C or a Runnable within an SW-C, a BSW module or CDD.

Table 8-1 Glossary

8.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
BSWM	Basic Software Mode Manager
CDD	Complex Device Driver
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
ECUM	ECU State Manager
HIS	Hersteller Initiative Software
MCU	Microcontroller Unit
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
PPort	Provide Port

RPort	Require Port
RTE	Runtime Environment
SCHM	Schedule Manager
SE	Supervised Entity
SRS	Software Requirement Specification
SW-C	Software Component
SWS	Software Specification
WDGIF	Watchdog Interface
WDGM	Watchdog Manager

Table 8-2 Abbreviations

9 Contact

Visit our website for more information on

- > News
- > Products
- > Demo software
- > Support
- > Training data
- > Addresses

www.vector-informatik.com