VECTOR

MICROSAR Wdg

Technical Reference

General

Version 3.01.00

Authors	Matthias Scheid
Status	Released

Document Information

History

Author	Date	Version	Remarks
Matthias Scheid	2016-02-05	3.00.00	Creation of DocTechRef
Matthias Scheid	2016-07-26	3.01.00	Renamed MemMap sections

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_WatchdogDriver.pdf	V2.5.0
[2]	AUTOSAR	AUTOSAR_SWS_DET.pdf	V3.2.0
[3]	AUTOSAR	AUTOSAR_SWS_DEM.pdf	V4.2.0
[4]	AUTOSAR	AUTOSAR_BasicSoftwareModules.pdf	V1.0.0
[5]	AUTOSAR	AUTOSAR_SWS_WatchdogInterface.pdf	V2.5.0

Scope of the Document

This technical reference describes the general use of the Watchdog driver basis software. All aspects which are Watchdog controller specific are described in the hardware specific document, which is also part of the delivery.

Caution
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

Illustrations

Tables

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
3.00.00	Development of Wdg__coreAsr according to SafeBSW process
3.02.00	Improved ASR3 compatibilty (MemMap sections, reference targets)

Table 1-1 Component history

2 Introduction

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

Supported AUTOSAR Release*:	4.0.3	
Supported Configuration Variants:	pre-compile	
Vendor ID:	Wdg_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	WDG_MODULE_ID	102 decimal (according to ref. [4])

* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

The module provides services for initializing a watchdog. Furthermore it supports changing the operating mode and triggering the watchdog.

Note
As this document describes the common aspects that apply to all watchdog modules provided by Vector, no infix is applied to names of APIs, macros, parameters, etc. in this general technical reference.

The infix applied to a hardware-specific watchdog driver module is mentioned in the according hardware-specific supplement to this technical reference.

2.1 Architecture Overview

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

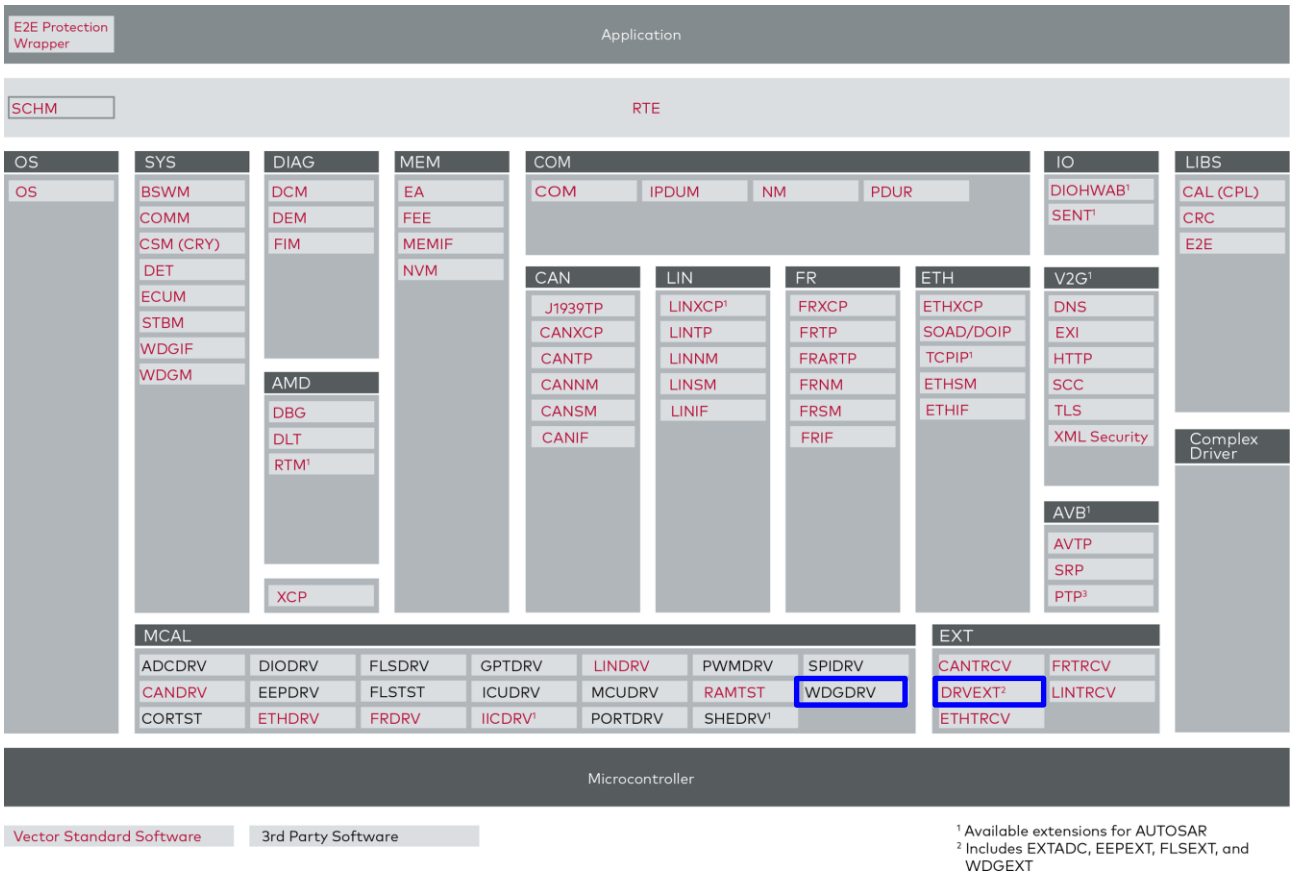

Figure 2-1 AUTOSAR 4.x Architecture Overview

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for the Wdg.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported AUTOSAR standard conform features

> Table 3-2 Not supported AUTOSAR standard conform features

Vector Informatik provides further Wdg functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Initialization of the watchdog driver
Control of watchdog hardware using DIO/SPI/register access (see specific TechRef)
Setting operation mode of watchdog hardware
Trigger concept for windowed watchdogs

Table 3-1 Supported AUTOSAR standard conform features

3.1.1 Deviations

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

Not Supported AUTOSAR Standard Conform Features
API `Wdg_GetVersionInfo()` is not provided as preprocessor macro.

Table 3-2 Not supported AUTOSAR standard conform features

3.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
ASR3 compatibility to allow the usage in a corresponding environment.

Table 3-3 Features provided beyond the AUTOSAR standard

3.1.3 ASR3 compatibility

Even though this watchdog driver module is released according ASR4, it can also be used in ASR3 projects/environments. For further informations see 4.4.

3.1.4 Modes

The watchdog driver module supports three different modes, specified [5]:

- `WDGIF_OFF_MODE`: Watchdog is disabled / hardware switched off
- `WDGIF_SLOW_MODE`: Watchdog is set up for long timeout period (slow triggering)
- `WDGIF_FAST_MODE`: Watchdog is set up for short timeout period (fast triggering)

Depending on constraints of the watchdog hardware the modes `WDGIF_OFF_MODE` / `WDGIF_SLOW_MODE` may not be supported.

3.2 Initialization

After power on the watchdog hardware has to be initialized. The module provides the function `Wdg_Init`. It is necessary to call this function before using any other service of the module by setting. It initializes the watchdog hardware and sets the configured default mode and initial timeout.

3.3 Initial timeout

The initial timeout is set every time the module is in `WDGIF_OFF_MODE` and another mode (see 3.1.4) is requested via the API `Wdg_SetMode`. As the initial internal mode of the watchdog driver module is WDGIF_OFF_MODE, the trigger condition is set to the initial timeout during module initialization.

3.4 Error Handling

3.4.1 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 (i.e. pre-compile parameter `Wdg_DEV_ERROR_DETECT==STD_ON`).

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

The reported Wdg ID is 102.

The reported service IDs identify the services which are described in 5.2. The following table presents the service IDs and the related services:

Service ID	Service
0x00	Wdg_Init
0x01	Wdg_SetMode
0x03	Wdg_SetTriggerCondition
0x04	Wdg_GetVersionInfo
0x05	Wdg_CbkGptTrigger
0x07	Wdg_Trigger

Table 3-4 Service IDs

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

Error Code	Description
0x10	WDG_E_DRIVER_STATE: API service used in wrong context (.e.g. module not initialized)
0x11	WDG_E_PARAM_MODE: API service called with wrong / inconsistent parameter(s)
0x12	WDG_E_PARAM_CONFIG: API service called with wrong / inconsistent parameter(s)
0x13	WDG_E_PARAM_TIMEOUT: The passed timeout value is higher than the maximum timeout value.
0x14	WDG_E_PARAM_POINTER: API is called with wrong pointer value (e.g. NULL_PTR).

Table 3-5 Errors reported to DET

3.4.2 Production Code Error Reporting

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

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

Error Code	Description
Assigned by DEM (WDG_DISABLE_REJECTED)	Initialization or watchdog mode switch failed because it would disable the watchdog though this is not allowed in this configuration.
Assigned by DEM (WDG_MODE_FAILED)	Setting a watchdog mode failed (during initialization or mode switch).

Table 3-6 Errors reported to DEM

4 Integration

This chapter gives necessary information for the integration of the MICROSAR Wdg into an application environment of an ECU.

4.1 Scope of Delivery

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

4.1.1 Static Files

File Name	Description
Wdg.c	This is the source file of Wdg
Wdg.h	This is the header file of Wdg>
Wdg_Cbk.h	This header file defines the callback functions of Wdg
Wdg_LL.c	This is the source file of sub-module Wdg_LL
Wdg_LL.h	This is the header file of sub-module Wdg_LL
Wdg_Mode.c	This is the source file of sub-module Wdg_Mode
Wdg_Mode.h	This is the header file of sub-module Wdg_Mode
Wdg_Timer.c	This is the source file of sub-module Wdg_Timer
Wdg_Timer.h	This is the header file of sub-module Wdg_Timer
Wdg_TrgCnd.c	This is the source file of sub-module Wdg_TrgCnd
Wdg_TrgCnd.h	This is the source file of sub-module Wdg_TrgCnd
Wdg_TrgCnd_Cbk.h	This header file defines the callback functions of sub-module Wdg_TrgCnd
Wdg_Types.h	This header file defines the global types used by Wdg.

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool [config tool].

File Name	Description
Wdg_Lcfg.c	This is the configuration source file of Wdg
Wdg_Cfg.h	This is the configuration header file of Wdg

Table 4-2 Generated files

4.2 Critical Sections

- `WDG_EXCLUSIVE_AREA_0`: Ensures consistency during mode change
- `WDG_EXCLUSIVE_AREA_1`: Ensures consistency during modification of trigger condition.

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 of the Wdg and illustrates their assignment among each other.

Compiler Abstraction Definitions / Memory Mapping Sections	WDG_CODE	WDG_VAR	WDG_CONST
WDG_START_SEC_CODE WDG_STOP_SEC_CODE	■		
WDG_STOP_SEC_VAR_INIT_8BIT WDG_STOP_SEC_VAR_INIT_8BIT		■	
WDG_START_SEC_VAR_ZERO_INIT_8BIT WDG_STOP_SEC_VAR_ ZERO_INIT_8BIT		■	
WDG_START_SEC_VAR_ ZERO_INIT_32BIT WDG_START_SEC_VAR_ ZERO_INIT_32BIT		■	
WDG_START_SEC_VAR_ZERO_INIT_UNSPECIFIED WDG_STOP_SEC_VAR_ZERO_INIT_UNSPECIFIED		■	
WDG_START_SEC_CONST_UNSPECIFIED WDG_STOP_SEC_CONST_UNSPECIFIED			■

Table 4-3 Compiler Abstraction and Memory Mapping

4.4 Autosar 3 compatibility

If the module is configured for usage in ASR3 environements (Configuration container `WdgAsr3Compatibility` exists in DaVinci Configurator 5), the additional API `Wdg_Trigger` is provided.

According to ASR3 specification, this function gets called cyclically from WdgIf in order to trigger the watchdog hardware. This differs from ASR4, as the watchdog driver module is responsible itself for cyclic triggering. WdgIf here uses the API `Wdg_SetTriggerCondition` to set the period in which Wdg is allowed to trigger the hardware (timeout) to prevent the watchdog from expiring.

To allow the usage of this Wdg in ASR3 environments, the API `Wdg_Trigger` is mapped to API `Wdg_SetTriggerCondition`, which gets called with the timeout value set in parameter `WdgTriggerTimeout`. Instead of periodical direct triggering of the watchdog hardware, the cyclic call of `Wdg_Trigger` leads to a continuous renewal of the timeout. As consequence the watchdog will expire, when WdgIf does not call Wdg_Trigger within the configured timeout.

5 API Description

5.1 Type Definitions

The types defined by the Wdg are described in this chapter.

Type Name	C-Type	Description	Value Range
Wdg_StateType	enum	Module states of Wdg	`WDG_UNINIT` Module is not initialized yet.
			`WDG_IDLE` Module is initialized and not busy.
			`WDG_BUSY#` Module is busy.

Table 5-1 Type definitions

Note
As the configuration type strongly depends on used watchdog hardware it is described in the hardware specific supplement to this technical reference.

5.2 Services provided by Wdg

5.2.1 Wdg_Init

Prototype	
`Wdg_Init(Wdg_ConfigType* ConfigPtr)`	
Parameter	
`ConfigPtr`	Pointer to configuration structure for initializing the module
Return code	
`retCode`	none
Functional Description	
description	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. <if applicable> > This function is non-reentrant. <if applicable>	
Expected Caller Context	
> ANY	

Table 5-2 Wdg_Init

5.2.2 Wdg_SetTriggerCondition

Prototype	
`Wdg_SetTriggerCondition(uint16 timeout)`	
Parameter	
`timeout`	Duration of timeout period in milliseconds
Return code	
`retCode`	none
Functional Description	
> This API is used by WdgIf to set the timeout period in which the watchdog driver is allowed to trigger the watchdog hardware. > In case the value 0 is passed as timeout parameter the module will cause a reset as soon as possible.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
ANY	

Table 5-3 Wdg_SetTriggerCondition

5.2.3 Wdg_SetMode

Prototype	
`Wdg_SetMode(WdgIf_ModeType Mode)`	
Parameter	
`Mode`	Mode to which the module / hardware should be set
Return code	
`Std_ReturnType`	E_OK - Mode switch executed successfully E_NOT_OK - Mode switch failed
Functional Description	
This API is used by WdgIf to set the mode of the watchdog hardware to a given mode and adapt the trigger timing. Depending on the hardware valid modes are WDGIF_OFF_MODE, WDGIF_SLOW_MODE and WDGIF_FAST_MODE.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	
Expected Caller Context	
> ANY	

Table 5-4 Wdg_SetMode

5.2.4 Wdg_GetVersionInfo

Prototype	
`Wdg_GetVersionInfo (Std_VersionInfoType* versioninfo)`	
Parameter	
`versioninfo`	Pointer to where to store the version information
Return code	
`Std_ReturnType`	E_OK - Mode switch executed successfully E_NOT_OK - Mode switch failed
Functional Description	
Returns the version information of this module	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. > The function is only available if switch `WDG_VERSION_INFO_API` is enabled.	
Expected Caller Context	
> ANY	

Table 5-5 Wdg_SetMode

5.2.5 Wdg_Trigger

Prototype	
`Wdg_Trigger (void)`	
Parameter	
-	-
Return code	
-	-
Functional Description	
This API provides a wrapper-functionality that allows this module to be used in ASR3-conform watchdog stacks. It has to be called cyclically from WdgIf to reset the trigger condition with a configured timeout.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > The function is only available if switch `WDG_ASR3X_COMPATIBILITY` is enabled.	
Expected Caller Context	
> ANY	

Table 5-6 Wdg_SetMode

5.3 Services used by Wdg

In the following table services provided by other components, which are used by the Wdg are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET	Det_ReportError()
DEM	Dem_ReportErrorStatus.
SchM/RTE	SchM_Enter_Wdg_WDG_EXCLUSIVE_AREA_x SchM_Exit_Wdg_WDG_EXCLUSIVE_AREA_x

Table 5-7 Services used by the Wdg

Note
Depending on the watchdog hardware the driver module may use additional services (Gpt, Dio, SPI). These are described in the hardware specific supplement of the technical reference.

5.4 Callback Functions

This chapter describes the callback functions that are implemented by the Wdg and can be invoked by other modules. The prototypes of the callback functions are provided in the header file `Wdg_Cbk.h` by the Wdg.

5.4.1 Wdg_Cbk_GptNotificationTrigger

Prototype	
`Wdg_Cbk_GptNotificationTrigger (void)`	
Parameter	
–	-
Return code	
–	-
Functional Description	
The main purpose of this function is to trigger the watchdog hardware. Additionally it triggers the update of the trigger condition and initiates a reset of the timing if the watchdog hardware requires an asynchronous mode change.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.	

Expected Caller Context
> ANY (ISR recommended)

Table 5-8 Wdg_SetMode

6 Glossary and Abbreviations

6.1 Glossary

Term	Description
EAD	Embedded Architecture Designer; generation tool for MICROSAR components
GENy	Generation tool for CANbedded and MICROSAR components

Table 6-1 Glossary

6.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
CFG5	DaVinci Configurator 5
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
HIS	Hersteller Initiative Software
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 6-2 Abbreviations

7 Contact

Visit our website for more information on

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

www.vector.com