VECTOR

# MICROSAR WDG

## Technical Reference

### MCAL Emulation in VTT

Version 1.2.0


| | |
|---|---|
| Authors | Christian Leder, Bethina Mausz |
| Status | Released |

# Document Information

## History

| Author | Date | Version | Remarks |
|---|---|---|---|
| Peter Lang | 2013-09-17 | 1.00.00 | Initial version |
| Christian Leder | 2015-02-09 | 1.01.00 | > Global renaming of Vip to Vtt<br>> Usage of template 5.11.0 for the Technical reference |
| Bethina Mausz | 18.06.2016 | 1.02.00 | > FEAT-1842, support of external driver. Restriction: Only one driver per system. |

## Reference Documents

| No. | Source | Title | Version |
|---|---|---|---|
| [1] | AUTOSAR | AUTOSAR_SWS_WatchdogDriver.pdf | V2.5.0 |
| [2] | AUTOSAR | AUTOSAR_SWS_DevelopmentErrorTracer.pdf | V3.2.0 |
| [3] | AUTOSAR | AUTOSAR_SWS_DiagnosticEventManager.pdf | V4.2.0 |
| [4] | AUTOSAR | AUTOSAR_TR_BSWModuleList.pdf | V1.6.0 |

**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 |
|---|---|
| 1.0.x | Initial version of the Vip WDG driver |
| 2.0.x | Global renaming of Vip to Vtt |

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 | |
| **Supported Configuration Variants:** | pre-compile | |
| **Vendor ID:** | WDG_VENDOR_ID | 30 decimal<br>(= Vector-Informatik, according to HIS) |
| **Module ID:** | WDG_MODULE_ID | 102 decimal<br>(according to ref. [4]) |

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

This driver provides services for initializing, changing the operating mode and triggering the watchdog in the VTT MCAL emulation.

## 2.1 Architecture Overview

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

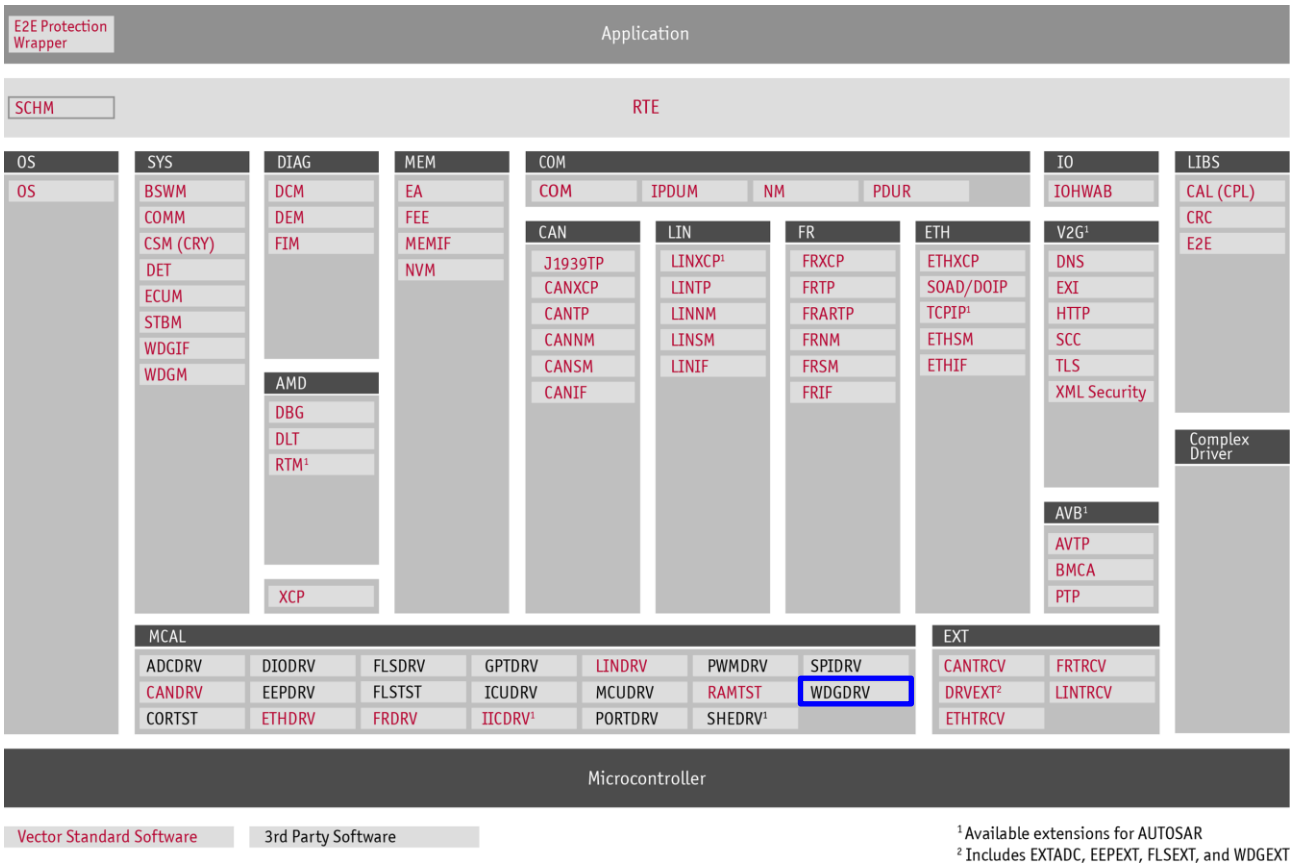


Figure 2-1 AUTOSAR 4.x Architecture Overview

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

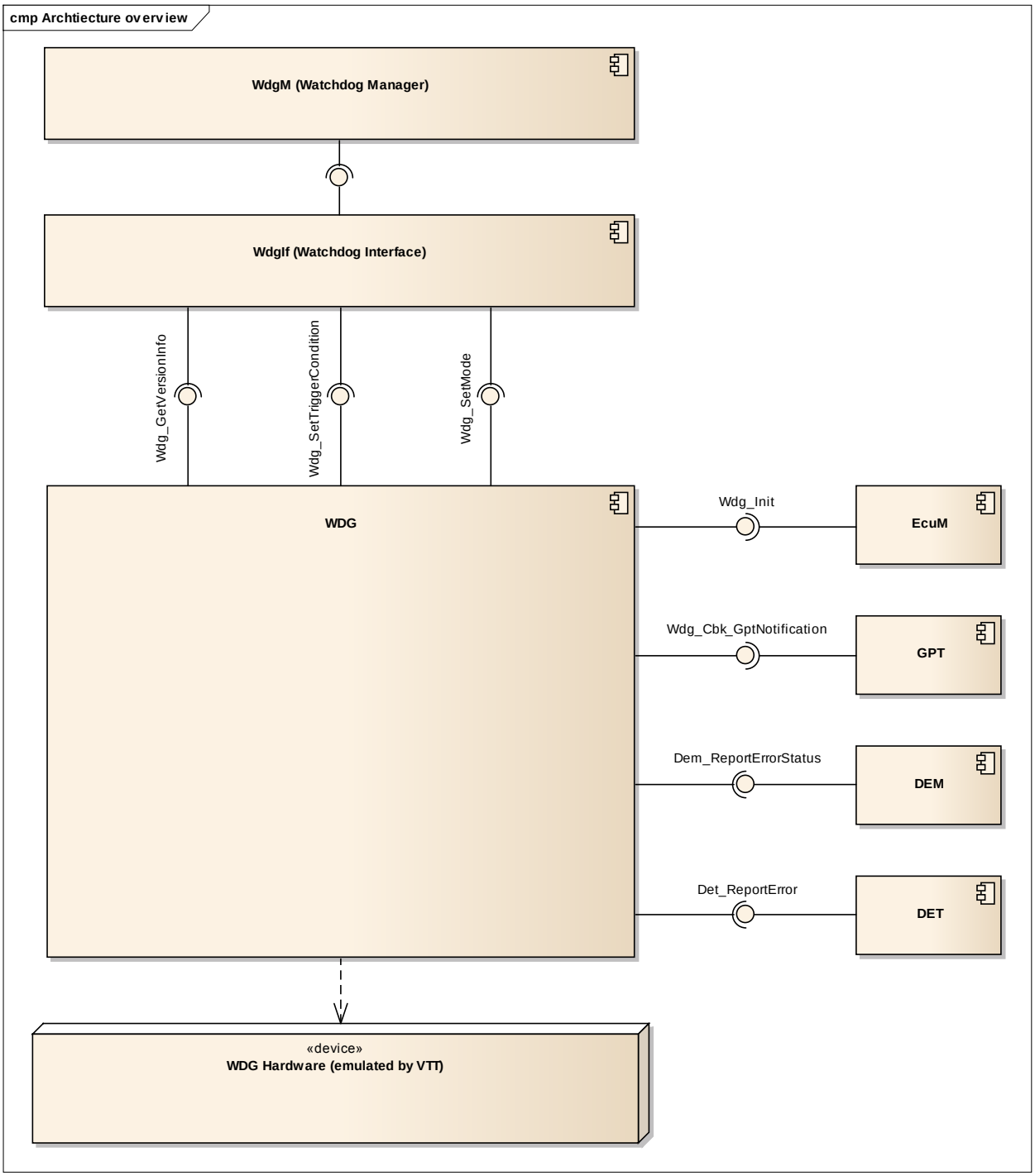


Figure 2-2 Interfaces to adjacent modules of the WDG

# 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 |
| Changing the operating mode |
| Watchdog trigger |

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 |
|---|
| Windowed Watchdog |
| Development Error `WDG_E_PARAM_TIMEOUT` will not be reported in this emulation |
| Due to the fact, that this watchdog is only emulated in the VTT framework, no reset of the device is performed in case the watchdog expires |

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 |
|---|
| In addition to the existing checks required by the AUTOSAR standard, the parameter `versioninfo` passed to the service `Wdg_GetVersionInfo` is checked for not referencing `NULL_PTR`. If it does, the error `WDG_E_PARAM_VINFO` is reported to DET instead of `WDG_E_PARAM_POINTER` |

Table 3-3 Features provided beyond the AUTOSAR standard

### 3.1.3 Limitations

There are no limitations.

## 3.2 Emulation

This driver is an emulation of an WDG module.

**Caution**

Be careful using while loops in order to poll any status.

The user has to ensure, that the application does not block the emulation. So, within every while loop the following function call has to be called:

```
while(ANY_STATUS == temp_status)
{
  Schedule();
}
```

Use the function call Schedule() which is available once the header file of the module WDG is included.

## 3.3 Initialization

The WDG module is being initialized by calling `Wdg_Init(&WdgConfigSet)`. All global variables are initialized by calling `Wdg_InitMemory()`. So, `Wdg_InitMemory()` has to be called prior to `Wdg_Init()`.

If disabling the watchdog is not allowed and the default mode given in the provided configuration set disables the watchdog, the `Wdg_Init()` raises the production error `WDG_DEM_ERROR_DISABLE_WDG_DISABLED`.

## 3.4 States

The module WDG provides the following global states:

- *uninitialized / undefined*: WDG is not initialized
- `WDG_BUSY`: Switching between modes or triggering is still ongoing
- `WDG_IDLE`: Watchdog is not being triggered or mode is not being switched

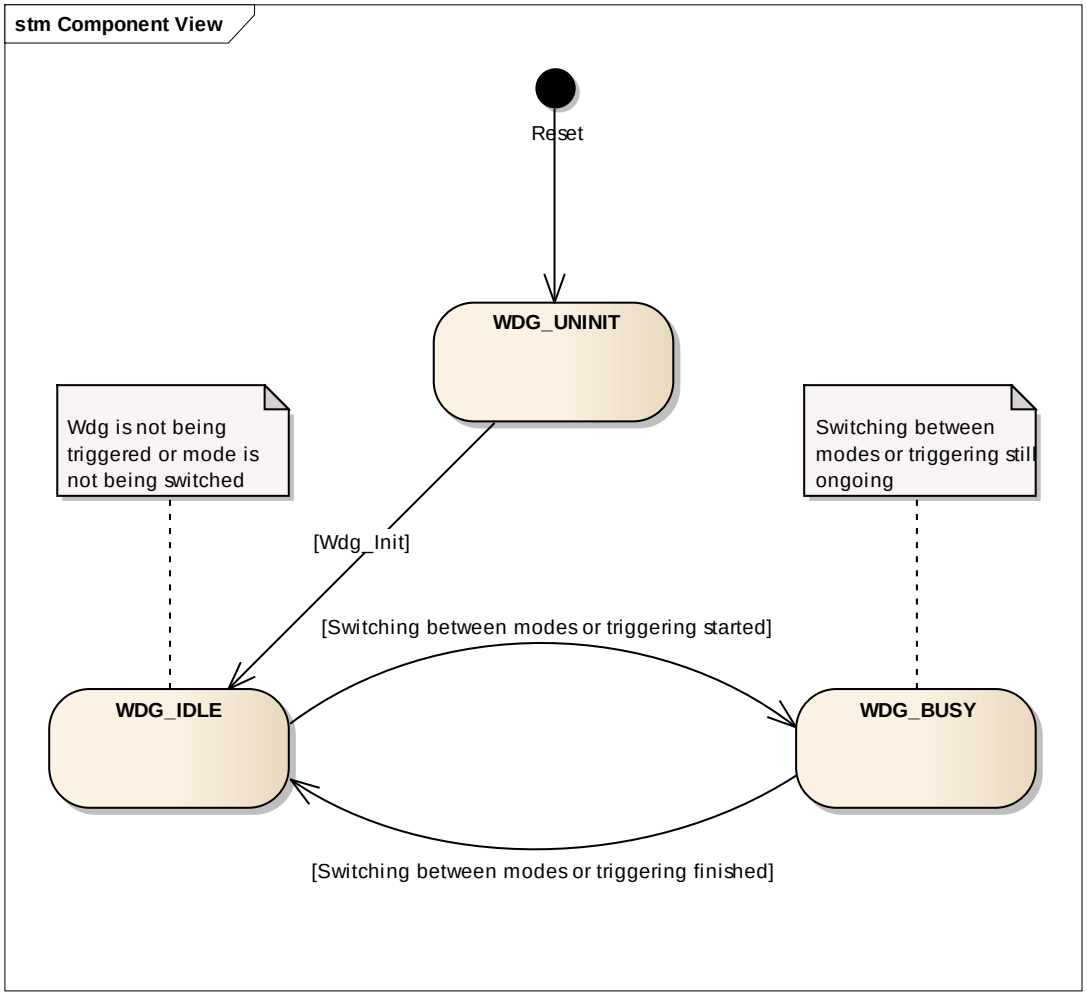


Figure 3-1 Module States

## 3.5 Main Functions

The WDG module does not provide any cyclic main functions.

## 3.6 Error Handling

### 3.6.1 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 |
| 0x02 | Wdg_SetTriggerCondition |
| 0x04 | Wdg_GetVersionInfo |
| 0x05 | Wdg_Cbk_Gpt_Notification |

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` | Parameter of `Wdg_SetMode()` is invalid |
| 0x12 | `WDG_E_PARAM_CONFIG` | Parameter of `Wdg_Init()` references `NULL_PTR` |
| 0x14 | `WDG_E_PARAM_VINFO` | Parameter of `Wdg_GetVersionInfo()` references `NULL_PTR` |

Table 3-5 Errors reported to DET

#### 3.6.1.1 Parameter Checking

AUTOSAR requires that API functions check the validity of their parameters. The checks in Table 3-6 are internal parameter checks of the API functions. These checks are for development error reporting and can be en-/disabled.

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

| Check / Service | WDG_E_DRIVER_STATE | WDG_E_PARAM_MODE | WDG_E_PARAM_CONFIG | WDG_E_PARAM_VINFO |
|---|---|---|---|---|
| Wdg_Init | | | ■ | |
| Wdg_SetMode | ■ | ■ | | |
| Wdg_SetTriggerCondition | | | | |
| Wdg_GetVersionInfo | | | | ■ |
| Wdg_Cbk_GptNotification | ■ | | | |

Table 3-6 Development Error Reporting: Assignment of checks to services

### 3.6.2 Production Code Error Reporting

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

| Error Code | Description |
|---|---|
| WDG_DEM_ERROR_DISABLE_WDG_DISABLED | Disabling of watchdog not allowed (e.g. in safety relevant systems) |

Table 3-7 Errors reported to DEM

This production error code can be reported to DEM out of two WDG services:

- If Wdg_Init is called with default configured mode WDGIF_OFF_MODE, but the WDG is not allowed to be disabled.
- If Wdg_SetMode is called with parameter WDGIF_OFF_MODE, but the WDG is not allowed to be disabled.

# 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.h | The module header defines the interface of the WDG. This file must be included by upper layer software components |
| Wdg.c | This C-source contains the implementation of the module's functionalities |
| DrvWd_VttCanoe01Asr.jar | This jar-file contains the generator and the validator for the DaVinci Configurator |
| VTTWdg_bswmd.arxml | Basic Software Module Description according to AUTOSAR for VTT Emulation |
| Wdg_bswmd.arxml | Optional Basic Software Module Description. Placeholder for real target (semiconductor manufacturer) in VTT only use case |

Table 4-1 Static files

### 4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator.

| File Name | Description |
|---|---|
| Wdg_Cfg.h | The configuration-header contains the static configuration part of this module |
| Wdg_PBcfg.c | The configuration-source contains the object independent part of the runtime configuration |
| Wdg_VendorId_ApiInfix.c | The source contains the wrapper APIs which maps the vendor/infix specific APIs to VTTWdg APIs. This file is generated in case an API-Infix is configured. |
| Wdg_VendorId_ApiInfix.h | The header contains the vendor/infix specific API declarations. It also contains the extern declaration of `Wdg_MainFunction`. This file is generated in case an API-Infix is configured. |

Table 4-2 Generated files

## 4.2 Include Structure

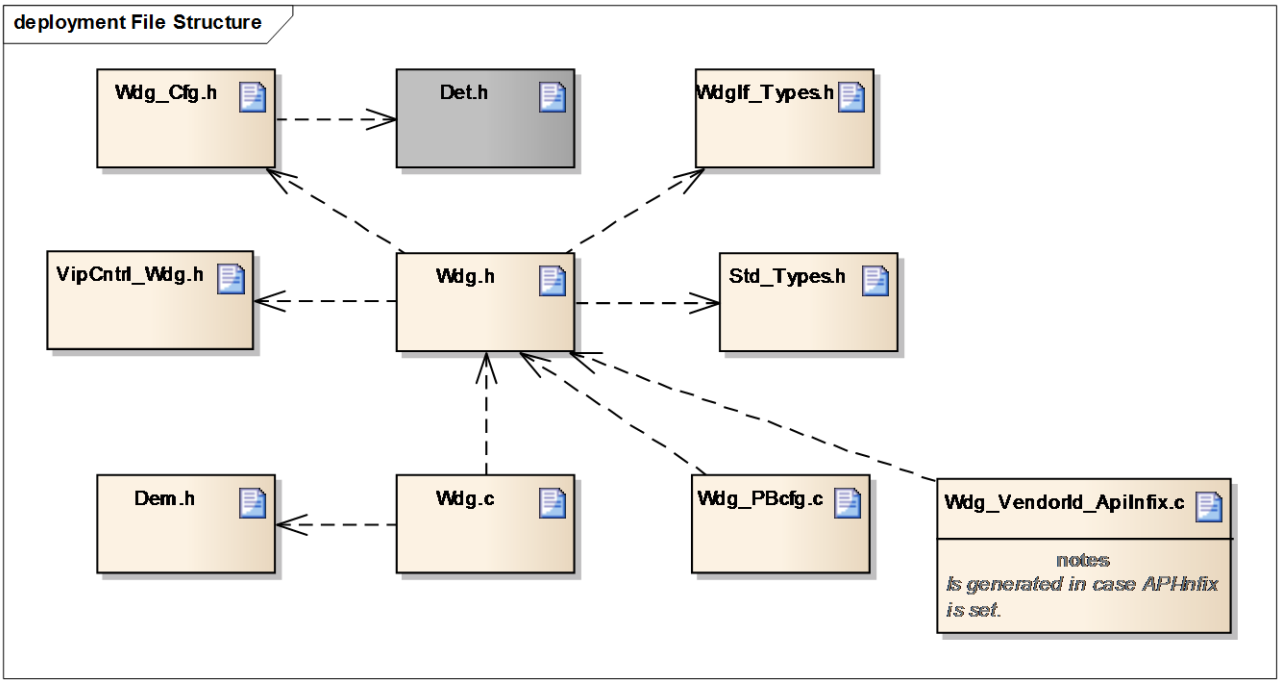


Figure 4-1 Include Structure

## 4.3 Dependencies on SW modules

### 4.3.1 DET (Optional)

The WDG module depends on the DET (by default) in order to report development errors. Detection and reporting of development errors can be enabled or disabled by the switch "Enable Development Error Detection".

### 4.3.2 ECUM

The EcuM cares for the initialization of the module WDG.

### 4.3.3 WDGM

The WDGM triggers all available watchdogs and supervises all registered applications for liveliness. The WDGM uses the WDGIF for accessing the underlying watchdog drivers.

### 4.3.4 WDGIF

The WDGIF provides uniform access to services of underlying watchdog drivers like mode switching and triggering. The appropriate watchdog driver is selected by a device index. The types `WdgIf_ModeType` and `WdgIf_StatusType` are imported from the WDGIF.

### 4.3.5 DEM

The diagnostic event manager records all production errors for diagnostic purposes. The reporting of production errors can be en-/disabled but the detection of production errors cannot be switched off.

# 5 API Description

For an interfaces overview please see Figure 2-2.

## 5.1 Type Definitions

The WDG does not define any types itself. Instead, the WDG uses imported types by WdgIf. The following types from the WdgIf are used within this emulation.

| Type Name | C-Type | Description | Value Range |
|---|---|---|---|
| WdgIf_ModeType | enum | A structure to hold the watchdog driver configuration set | `WDGIF_OFF_MODE`<br>Watchdog disabled |
| | | | `WDGIF_SLOW_MODE`<br>Watchdog runs in slow mode |
| | | | `WDGIF_FAST_MODE`<br>Watchdog runs in fast mode |

Table 5-1 Type definitions

## 5.2 Services provided by WDG

### 5.2.1 Wdg_InitMemory

| Prototype | |
|---|---|
| `void` **`Wdg_InitMemory`** `(void)` | |
| **Parameter** | |
| - | - |
| **Return code** | |
| - | - |
| **Functional Description** | |
| This service initializes the global variables in case the startup code does not work | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non reentrant.<br>> Module must not be initialized. | |
| Expected Caller Context | |
| > Called during startup | |

Table 5-2 Wdg_InitMemory

### 5.2.2 Wdg_Init

| Prototype | |
|---|---|
| `void` **`Wdg_Init`** `(void)` | |
| **Parameter** | |
| `initObject` | Handle of an initialization structure. The generated macros should be used: `kCanInitObjX` (with X = 1 ... Number of generated initialization structures) |
| **Return code** | |
| - | - |
| **Functional Description** | |
| This function initializes the module and the watchdog hardware (emulated in VTT); it sets the default watchdog mode and timeout period as provided in the configuration set. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non reentrant.<br>> Module must not be initialized. | |
| Call context | |
| > ECU State Manager or comparable software module, responsible for driver initialization after startup. | |

Table 5-3 Wdg_Init

### 5.2.3 Wdg_SetMode

| Prototype | |
|---|---|
| `Std_ReturnType` **`Wdg_SetMode`** `(WdgIf_ModeType Mode)` | |
| **Parameter** | |
| `Mode` | Watchdog mode to switch to. |
| **Return code** | |
| `Std_ReturnType` | `E_OK`, Mode switching has been completely executed.<br>`E_NOT_OK`, Mode switching failed |
| **Functional Description** | |
| This function switches the watchdog module and the watchdog hardware from the current watchdog mode to the watchdog mode defined by the parameter `Mode`. This means that the function attempts to set all parameters of the watchdog module and the watchdog hardware to the values defined in the configuration for the new mode. | |
| **Particularities and Limitations** | |
| > This function is synchronous<br>> This function is non reentrant.<br>> This function may only be called if the module has been initialized before. | |
| Call context | |
| > Task Context | |

Table 5-4 Wdg_SetMode

### 5.2.4 Wdg_SetTriggerCondition

| Prototype | |
|---|---|
| `void` **`Wdg_SetTriggerCondition`** `(uint16 timeout)` | |
| **Parameter** | |
| `timeout` | The parameter specified the allowed trigger window in miliseconds |
| **Return code** | |
| - | - |
| **Functional Description** | |
| This function defines a time window in which the watchdog can be trigger by a GPT timer. | |
| **Particularities and Limitations** | |
| > This function is synchronous and non reentrant.<br>> This function may only be called if the module has been initialized before. | |
| Call context | |
| > Task Context | |

Table 5-5 Wdg_SetTriggerCondition

### 5.2.5 Wdg_GetVersionInfo

| Prototype | |
|---|---|
| `void` **`Wdg_GetVersionInfo`** `(Std_VersionInfoType* versioninfo )` | |
| **Parameter** | |
| `initObject` | Handle of an initialization structure. The generated macros should be used: `kCanInitObjX` (with X = 1 ... Number of generated initialization structures) |
| **Return code** | |
| - | - |
| **Functional Description** | |
| This function returns the version information of the module.<br>The version information includes:<br>> Module Id<br>> Vendor Id<br>> Software version numbers | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is reentrant.<br>> This service is configurable. | |
| Call context | |
| > Task Context | |

Table 5-6 Wdg_GetVersionInfo

### 5.2.6 Wdg_Cbk_GptNotification

| Prototype | |
|---|---|
| `void Wdg_Cbk_GptNotification (void)` | |
| **Parameter** | |
| - | - |
| **Return code** | |
| - | - |
| **Functional Description** | |
| This function is used as notification function for a GPT timer which should be setup to fit to the expected trigger period of the triggered watchdog. | |
| **Particularities and Limitations** | |
| > This function is synchronous<br>> This function is non reentrant.<br>> This function may only be called if the module has been initialized before. | |
| Call context | |
| > Interrupt Context | |

Table 5-7 Wdg_Cbk_GptNotification

## 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 |

Table 5-8 Services used by the WDG

# 6 Configuration

## 6.1 Configuration Variants

The WDG supports the configuration variants

> VARIANT-PRE-COMPILE

The configuration classes of the WDG parameters depend on the supported configuration variants. For their definitions please see the VTTWdg_bswmd.arxml file.

## 6.2 Configuration with DaVinci Configurator 5

The WDG module is configured with the help of the configuration tool DaVinci Configurator 5 (CFG5). The definition of each parameter is given in the corresponding BSWMD file.

# 7 Glossary and Abbreviations

## 7.1 Glossary

| Term | Description |
|---|---|
| CANoe | Tool for simulation and testing of networks and electronic control units. |
| DaVinci Configurator | Configuration and generation tool for MICROSAR components |

Table 7-1 Glossary

## 7.2 Abbreviations

| Abbreviation | Description |
|---|---|
| API | Application Programming Interface |
| AUTOSAR | Automotive Open System Architecture |
| BSW | Basis Software |
| DEM | Diagnostic Event Manager |
| DET | Development Error Tracer |
| ECU | Electronic Control Unit |
| EcuM | ECU State Manager |
| MICROSAR | Microcontroller Open System Architecture (the Vector AUTOSAR solution) |
| OS | Operating System |
| RTE | Runtime Environment |
| SWS | Software Specification |
| VTT | vVIRTUALtarget |
| WDG | Watchdog Driver |
| WDGIF | Watchdog Interface |
| WDGM | Watchdog Manager |

Table 7-2 Abbreviations

# 8 Contact

Visit our website for more information on

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

www.vector.com