Safe Watchdog Interface

User Manual

Version:	1.9.4
Date:	21.05.2014
Document number:	D-MSP-M-70-006

TTTech Automotive GmbH
Schoenbrunner Str. 7, A-1040 Vienna, Austria, Tel. + 43 1 585 34 34-0, Fax +43 1 585 34 34-90, support@tttech-automotive.com

Table of Contents

1 Introduction

The **Safe Watchdog Interface (S-WdgIf)** is a part of the **Safe Watchdog Manager Stack**.

Note: This is a user manual and does *not* cover safety-related topics. For safety-related projects that need to fulfill ISO 26262 requirements, refer to the *Safe Watchdog Interface Safety Manual* [5] 20.

Note: The `<infix>` placeholder used in this document stands for the *infix* part of the names of the Watchdog driver functions to which the S-WdgIf interfaces. Depending on the version of the used AUTOSAR environment, the S-WdgIf can consist of the following:

- In **AUTOSAR 4.0** compatible environment, the S-WdgIf consists of the vendor ID and device name strings, where vendo ID is the ID of the vendor of the Watchdog driver and device name is the name of the configured Watchdog driver device .
- In **AUTOSAR 3.1** compatible environment, the S-WdgIf consists of the device name string, where device name is the name of the configured Watchdog driver device

2 Safe Watchdog Interface

2.1 Basic Functionality of the S-WdgIf

The **S-WdgIf** was developed according AUTOSAR version 4.0.1 [2] 20 and adapted for the AUTOSAR 3.1.4 environment [1] 20. The S-WdgIf is compatible with both AUTOSAR versions, but not fully compliant. For the deviations, see section *Deviations from the AUTOSAR Watchdog Interface* 5.

The S-WdgIf is designed to be integrated into an AUTOSAR 3.1.4 and 4.0.1 system. However, it is not restricted to this AUTOSAR version. The software module can also be integrated into other versions of AUTOSAR and other system software architectures if the integration related requirements listed in the **Safe Watchdog Interface Safety Manual** [5] 20 are met.

The S-WdgIf provides a standard interface to all the configured watchdog devices. The Safe Watchdog Manager (**S-WdgM**) accesses each configured watchdog through its **Device Index**.

The **S-WdgIf** is hardware-independent and abstracts one or more Safe Watchdog Driver modules for the **S-WdgM**. The S-WdgM calls the S-WdgIf with a device index parameter (`DeviceIndex`). It is translated by the S-WdgIf into a S-Wdg driver instance. If necessary, additional driver parameters are provided.

Figure 1 shows the layered structure of the **S-WdgM**. The attached watchdog devices can be internal, external, or both.

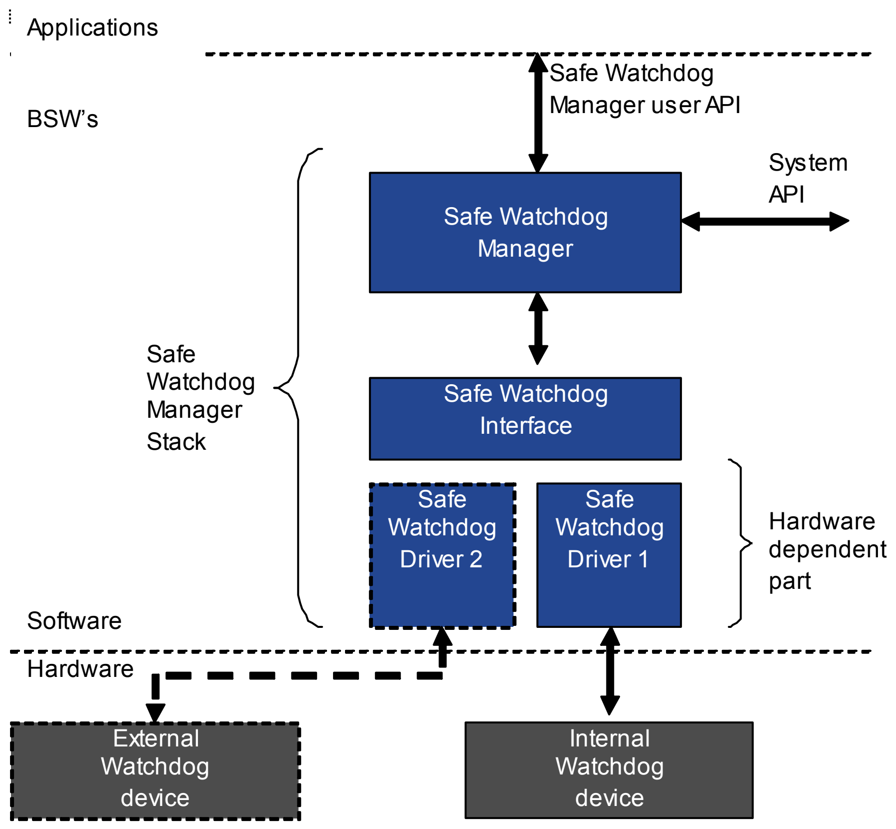

Fig. 1: Layered structure of the Safe Watchdog Manager

2.2 Extensions to the AUTOSAR Watchdog Interface

The S-WdgIf implements and extends the **AUTOSAR Watchdog Interface** module. It implements the following two interfaces to the Watchdog Driver:

- **AUTOSAR**-compatible interface
  - `Wdg_<infix>_SetTriggerCondition()`
  - `Wdg_<infix>_SetMode()`
- **TTTech**-compatible interface
  - `Wdg_<infix>_SetTriggerWindow()`
  - `Wdg_<infix>_SetMode()`

Switching between the two interfaces with the watchdog driver is done by setting the parameter `WdgIfUseAutosarDrvApi` 6 in a correspoding way. For details, see section Integration with Fully AUTOSAR Compliant Drivers 5 and the description of parameter `WdgIfUseAutosarDrvApi` 6 in section Configuration Parameters for the S-WdgIf 5.

Function `WdgIf_GetTickCounter()` is an additional extension to AUTOSAR, providing access to the corresponding S-Wdg function `Wdg_<infix>_GetTickCounter()`, if supported by the hardware and if the configuration parameter `WdgIfInternalTickCounterRef` 7 is set in a correspoding way. For details, see the description of parameter `WdgIfInternalTickCounterRef` 7 in section Configuration Parameters for the

S-WdgIf[5].

2.3 Deviations from the AUTOSAR Watchdog Interface

For safety reasons, the S-WdgIf module should not depend on external modules. This is why the AUTOSAR module **Development Error Tracer (DET)** is called in the presence of development errors only if the preprocessor switch `WDGIF_DEV_ERROR_DETECT`[8] is set to `STD_ON`.

The S-WdgIf calls function `Appl_Det_ReportError()`[15] in order to report detected DET errors instead of calling function `Det_ReportError()`[15] specified in AUTOSAR. For details, see section Expected Interfaces[15].

2.4 Integration with Fully AUTOSAR Compliant Drivers

In order to integrate the **S-WdgIf** with a fully AUTOSAR-compliant watchdog driver set the configuration parameter `WdgIfUseAutosarDrvApi`[6] to `STD_ON`. This will result in the following:

- The AUTOSAR `Wdg_<infix>_SetMode()` is called out of `WdgIf_SetMode()`.
- The `Wdg_<infix>_SetTriggerCondition()` is called out of `WdgIf_SetTriggerCondition()`.
- The `Wdg_<infix>_SetTriggerCondition()` is called out of `WdgIf_SetTriggerWindow()` as well, however, the parameter WindowStart is not passed.

Note: If the S-WdgM is the caller of the S-WdgIf (i.e., function `WdgIf_SetTriggerWindow()` is used to service the watchdog device), then the parameter `WindowStart` (`WdgMTriggerWindowStart`) has no effect, because it cannot be passed to an AUTOSAR-compliant driver. It is then good practice to set it to **0**, because this would be the functional meaning of its absence.

For more information about the configuration parameter `WdgIfUseAutosarDrvApi`[6], see section Configuration Parameters for the S-WdgIf[5].

2.5 Configuration Parameters for the S-WdgIf

Parameter Name	`WdgIfDeviceIndex`
Path	`WdgIf/WdgIfDevice/WdgIfDeviceIndex`
Group	Watchdog device
Type	Integer
Range	`0...65535`
Compatibility	AUTOSAR
Description	Represents the Watchdog Device ID so that it can be referenced by the S-WdgM.

Parameter Name	`WdgIfDriverRef`
Path	`WdgIf/WdgIfDevice/WdgIfDriverRef`
Group	Watchdog device
Type	Reference
Range	n.a.
Compatibility	AUTOSAR
Description	Reference to the Watchdog driver of this Watchdog Device.

Parameter Name	`WdgIfDevErrorDetect`
Parameter Name (Embedded Code)	`WDGIF_DEV_ERROR_DETECT`
Path	`WdgIf/WdgIfGeneral/WdgIfDevErrorDetect`
Group	Preprocessor
Type	`Boolean`
Range	`false/true`
Compatibility	AUTOSAR
Description	Pre-processor switch for enabling the development error reporting.

Parameter Name	`WdgIfVersionInfoApi`
Parameter Name (Embedded Code)	`WDGIF_VERSION_INFO_API`
Path	`WdgIf/WdgIfGeneral/WdgIfVersionInfoApi`
Group	Preprocessor
Type	`Boolean`
Range	`false/true`
Compatibility	AUTOSAR
Description	Pre-processor switch to enable/disable the service returning the version information.

Parameter Name	`WdgIfUseAutosarDrvApi`
Parameter Name (Embedded Code)	`WDGIF_USE_AUTOSAR_DRV_API`
Path	`WdgIf/WdgIfGeneral/WdgIfUseAutosarDrvApi`
Group	Preprocessor
Type	Boolean
Range	`false/true`
Compatibility	TTTech
Description	Pre-processor switch to enable/disable the use of fully AUTOSAR-compliant driver API functions.

Parameter Name	`WdgIfInternalTickCounterRef`
Parameter Name (Embedded Code)	`WDGIF_INTERNAL_TICK_COUNTER`
Path	`WdgIf/WdgIfGeneral/WdgIfInternalTickCounte`
Group	Preprocessor
Type	Reference
Range	n.a.
Compatibility	TTTech
Description	If this parameter references a driver which implements an internal tick counter then the function `WdgIf_InternalTickCounter()` is compiled and can be used by the S-WdgM. Otherwise the API is not compiled. **Note:** If a driver is selected, then it must support the internal tick counter, and its parameter `WdgInternalTickCounter` must be set to `TRUE`.

2.6 Configuring the S-WdgIf in the ECU Description File

Apart from the preprocessor options the user must add a `WdgIfDevice` container to the S-WdgIf module in the ECU description file. This container has two attributes: `WdgIfDeviceIndex` shall be set to zero, and `WdgIfDriverRef` [6] points must be selected so that it points to the Watchdog driver being used.

2.7 Preprocessor Options

Option	Description
`WDGIF_DEV_ERROR_DETECT`	Enables or disables calls to DET in case of development errors. Corresponds to ECU description option `WdgIfDevErrorDetect` 6. ▪ `STD_ON`: Development errors are checked and reported to DET ▪ `STD_OFF`: Development errors are checked but not reported to DET
`WDGIF_VERSION_INFO_API`	Enables the version info API for compiling (can be disabled in order to save resources). Corresponds to ECU description option `WdgIfVersionInfoApi` 8. ▪ `STD_ON`: The S-WdgIf API for version information is provided. ▪ `STD_OFF`: The S-WdgIf API for version information is not provided.
`WDGIF_INTERNAL_TICK_COUNTER`	Enables or disables the usage of an internal tick counter in the S-WdgIf. Corresponds to ECU description option `WdgIfInternalTickCounterRef` 8. If a driver is referenced the usage of an internal tick counter is enabled, otherwise disabled.
`WDGIF_USE_AUTOSAR_DRV_API`	Enables or disables the use of fully AUTOSAR-compliant driver API functions. ▪ `STD_ON`: AUTOSAR-compliant driver API functions are used. **Note:** The parameter `WindowStart` (`WdgMTriggerWindowStart`) of the S-WdgM configuration is then ignored. Therefore, it is good practice to set it to **0**. ▪ `STD_OFF`: TTTech driver API functions are used.

2.8 File Structure

Figure 2 gives an overview of the file structure of the **S-WdgIf**.

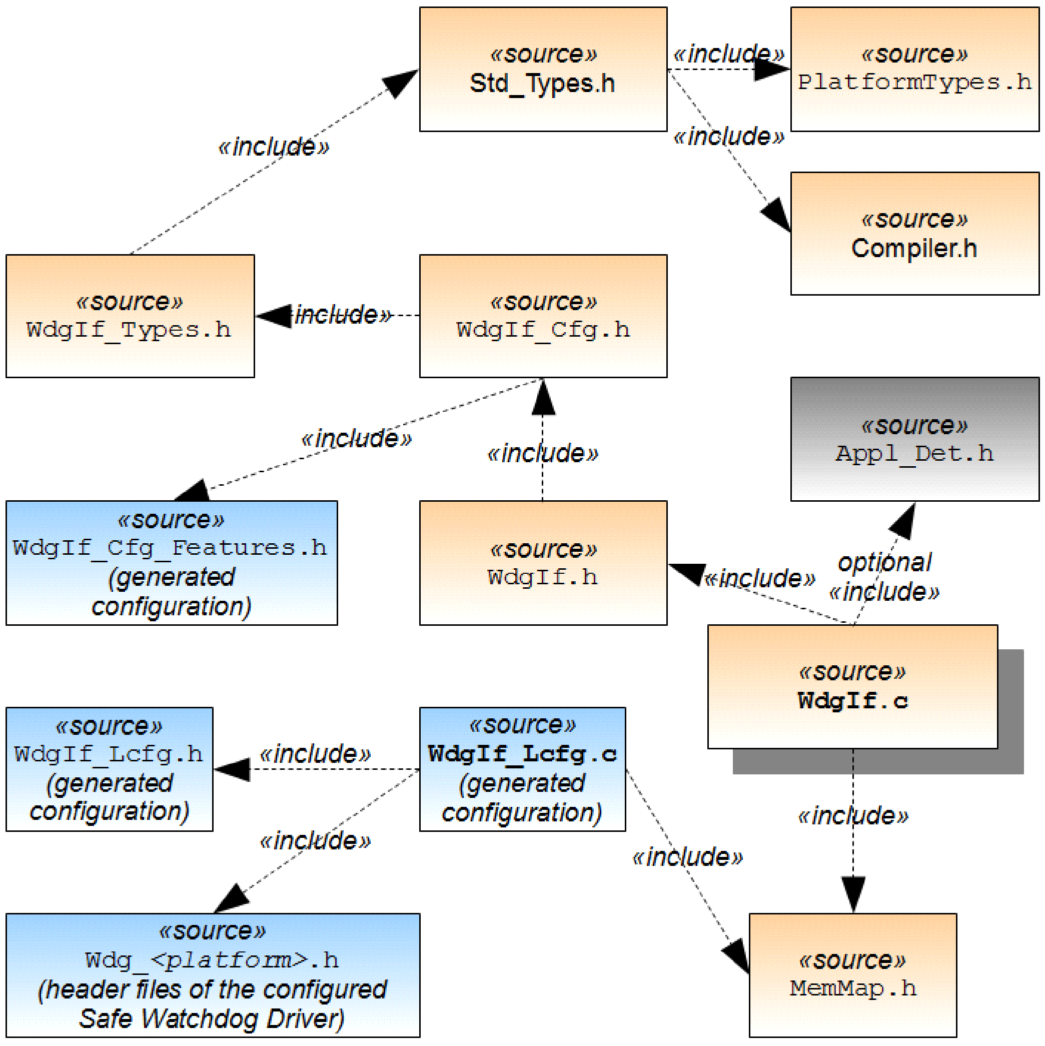

Fig. 2: File structure of the S-WdgIf

The implementation of the S-WdgIf module is in the file `WdgIf.c`. File `WdgIf.c` provides an interface to the **upper layer**, the S-WdgM, and includes the header file `WdgIf.h`. The header files `WdgIf_Types.h` and `Std_Types.h` are included indirectly through the `WdgIf.h`.

File	Description
`WdgIf_Types.h`	S-WdgIf and common Safe Watchdog Stack type definitions
`WdgIf_Cfg.h`	Pre-compile time definitions
`MemMap.h`	Is included directly in the module implementation files to organize code, data and constants in the memory.
`Appl_Det.h`	`Appl_Det.h` is included instead of `Det.h`, because the reporting of development errors is not done by directly calling the DET service `Det_ReportError()` 15, but by calling the user-defined service `Appl_Det_ReportError()` 15. **Note:** This service could just be a direct call to the external module DET, but could also perform more complex operations such as switching the OS context before calling DET.
`WdgIf_Lcfg.c` and `WdgIf_Lcfg.h`	These files contain the generated configuration.
`WdgIf_Cfg_Features.h`	Is generated and contains all preprocessor options for the S-WdgIf module.

3 API Description

This section describes the types, functions and interfaces that are imported or provided by the S-WdgIf software layer.

3.1 Type Definitions

This section describes the **types of the parameters** passed to the API functions of the **S-WdgIf**.

Name	`WdgIf_InterfaceFunctionsType`	
Type	`Structure`	
Elements	`Std_ReturnType (* Wdg_SetMode) (uint8, WdgIf_ModeType)`	Pointer to the platform-specific `SetMode` function
	`void (* Wdg_SetTriggerWindow) (uint8, uint16, uint16)`	Pointer to the platform-specific `SetTriggerWindow` function
Description	Provides pointers to the platform-specific APIs	

Name	`WdgIf_InterfaceFunctionsPerWdgDeviceType`	
Type	`Structure`	
Elements	`const WdgIf_InterfaceFunctionsType* WdgFunctions`	Pointers to the platform-specific functions
	`const uint8 WdgInstance`	Index of the physical watchdog instance within this platform
Description	Connects platform-dependent functions to a physical watchdog in order to allow several watchdogs of the same platform to work simultaneously (e.g., external watchdogs).	

Name	`WdgIf_InterfaceType`	
Type	`Structure`	
Elements	`const uint8 NumOfWdgs`	Number of watchdogs supported in the S-WdgIf
	`const WdgIf_InterfaceFunctionsPerWdgDeviceType* WdgFunctionsPerDevice`	Reference to the functions and physical watchdog indices
	`#if (WDGIF_INTERNAL_TICK_COUNTER`	Function pointer to the `GetTickCounter` driver function if the

	`== STD_ON)` `uint32 (*Wdg_GetTickCounter)` `(void)` `#endif`	internal tick counter is switched on
Description	Main S-WdgIf configuration structure.	

Name	`WdgIf_ModeType`
Type	`Enumeration`
Range	▪ `WDGIF_OFF_MODE:` Watchdog disabled ▪ `WDGIF_SLOW_MODE:` Long timeout period (slow triggering) ▪ `WDGIF_FAST_MODE:` Short timeout period (fast triggering)
Description	Mode of the Watchdog

3.2 S-WdgIf Functions

Syntax	`Std_ReturnType WdgIf_SetMode (uint8 DeviceIndex, WdgIf_ModeType Mode)`
Service ID [hex]	`0x01`
Sync/Async	Synchronous
Reentrant?	No
Parameters (in)	▪ `DeviceIndex`: Identifies the watchdog instance. ▪ `Mode`: One of the following statically configured modes: o `WDGIF_OFF_MODE:` Watchdog disabled o `WDGIF_SLOW_MODE:` Long timeout period (slow triggering) o `WDGIF_FAST_MODE:` Short timeout period (fast triggering)
Parameters (in/out)	none
Parameters (out)	none
Return value	▪ `E_OK`: API finished successfully. ▪ `E_NOT_OK`: An error occurred during execution.
Description	This function maps the `SetMode` service to the corresponding physical watchdog implementation according to the parameter `DeviceIndex`.

Syntax	`Std_ReturnType WdgIf_SetTriggerCondition (uint8 DeviceIndex, uint16 Timeout)`
Service ID[hex]	`0x02`
Sync/Async	Synchronous
Reentrant?	No
Parameters (in)	▪ `DeviceIndex`: Identifies the watchdog instance. ▪ `Timeout`: Timeout value in milliseconds for setting the trigger counter.
Parameters (in/out)	none
Parameters (out)	none
Return value	▪ `E_OK`: API finished successfully. ▪ `E_NOT_OK`: An error occurred during execution.
Description	This function maps the `SetTriggerCondition` service to the corresponding physical watchdog implementation according to the parameter `DeviceIndex`.

Syntax	`Std_ReturnType WdgIf_SetTriggerWindow (uint8 DeviceIndex, uint16 WindowStart, uint16 Timeout)`
Service ID[hex]	`0x04`
Sync/Async	Synchronous
Reentrant?	No
Parameters (in)	▪ `DeviceIndex`: Identifies the watchdog instance. ▪ `WindowStart`: Minimum time until next watchdog service is allowed in milliseconds. ▪ `Timeout`: Timeout value in milliseconds for setting the trigger counter.
Parameters (in/out)	none
Parameters (out)	none
Return value	▪ `E_OK`: API finished successfully. ▪ `E_NOT_OK`: An error occurred during execution.

Description	This function maps the `SetTriggerWindow` service to the corresponding physical watchdog implementation according to the parameter `DeviceIndex`.

Syntax	`void WdgIf_GetVersionInfo (Std_VersionInfoType* VersionInfoPtr)`
Service ID[hex]	`0x03`
Sync/Async	Synchronous
Reentrant?	No
Parameters (in)	`VersionInfoPtr`: Pointer to where to store the version information of this module.
Parameters (in/out)	none
Parameters (out)	none
Return value	`void`
Description	This function is implemented as a macro and returns the version information about this module. This function is only enabled if the preprocessor switch `WDGIF_VERSION_INFO_API` 8 is `STD_ON`.

Syntax	`uint32 WdgIf_GetTickCounter (void)`
Service ID[hex]	none
Sync/Async	Synchronous
Reentrant?	No
Parameters (in)	`TickCounter`: Pointer to where to store the tick counter value provided by the driver.
Parameters (in/out)	none
Parameters (out)	none
Return value	The current hardware tick counter of type `uint32`.
Description	This function returns the current hardware tick counter.

3.3 Expected Interfaces

This section describes the expected interfaces to external modules used by the S-WdgIf and their activation conditions.

Function	Description
`Appl_Det_ReportError()`	If the preprocessor option `WDGIF_DEV_ERROR_DETECT` [8] is set to `STD_ON`, the S-WdgIf references the function `Appl_Det_ReportError()` of the **DET API**. **Note:** If the pre-compiler option `WDGIF_DEV_ERROR_DETECT` is set to `STD_OFF`, the S-WdgIf is self-contained and does not call any functions from external modules.

4 S-WdgIf Configuration

This section describes the **configuration** of the **S-WdgIf**.

4.1 Link Time Configuration

The link time configuration for the S-WdgIf is configured in the ECU description file, e.g., by a tool such as **DaVinci Configurator Pro** . Basically, link time configuration contains all information needed for mapping each underlying watchdog driver to a device index. The configuration can be generated by the **S-WdgIf configuration generator**, described in section The S-WdgIf Configuration Generator 16.

4.2 S-WdgIf Configuration Generator

The **S-WdgIf generator** is a Microsoft Windows console application that can be launched from a **command prompt** window by entering the command `Wdg_If_Cfg_Gen.exe`. The S-WdgIf generator **reads** the S-WdgIf **module information** from the AUTOSAR **ECU description file** (`*.arxml`) and generates configuration structures for the S-WdgIf.
Note: Alternatively, you can use the **DaVinci** Configurator from Vector Informatik GmbH.

To use the S-Wdg generator, enter the following command in a command prompt window:

```
Wdg_If_Cfg_Gen.exe [options] <ECU-DESC-FILE> <OUTPUT-DIR>
<BSWMD_DIR>
```

`[options]`	**Description**
`--version`	Show the application version number and exits.
`-h/--help`	Shows this help message and exits.

Parameter	**Description**
`<ECU-DESC-FILE>`	The ECU description file (`*.arxml`). It is generated by a development tool (e.g.,from the **DaVinci** tool chain).
`<OUTPUT-DIR>`	The destination folder for the generated output. You must specify this parameter.
`<BSWMD_DIR>`	The directory in which the Configuration Generator recursively searches for the BSWMD file(s) that describe the used Watchdog drivers.

The respective configuration for the S-WdgIf is exported to two files:
- `WdgIf_Lcfg.c`
- `WdgIf_Lcfg.h`

4.3 Error Messages

The generator will show an **error message** in the command prompt window and quit if something goes wrong during configuration generation.

4.3.1 Basic Errors

Error No.	Error Message
1	`Bad call syntax.`
2	``Cannot open ECU description file `%s`.``
3	``Cannot convert float parameter `%s/%s` to an Watchdog ticks.``
4	``Cannot convert `%s` to a numerical value.``
5	`Fatal error.`
6	``Method `%s` must be implementd by subclass of `%s`.``
7	`Missing WdgM data.`

4.3.2 Semantic Errors

Error No.	Error Message
1007	``Missing non-zero value for `RTICLK_khz`.``
1008	``Non-supported platform `%s`.``
1009	`%s platform: Bad WDGIF_SLOW_MODE configuration: InitialTimeout_ms = %d > %d = SlowModeMax_ms.`
1010	`%s platform: Bad WDGIF_SLOW_MODE configuration: InitialWindowStart_ms = %d > %d = SlowModeMax_ms.`
1011	`%s platform: Bad WDGIF_FAST_MODE configuration: InitialTimeout_ms = %d > %d = FastModeMax_ms.`
1012	`%s platform: Bad WDGIF_FAST_MODE configuration: InitialWindowStart_ms = %d > %d = FastModeMax_ms.`
1013	`%s platform: Bad WDGIF_FAST_MODE configuration: InitialWindowStart_ms = %d >= %d = InitialTimeout_ms.`
1014	`WDGIF_OFF_MODE not supported for %s.`
1022	``` `Wdg/WdgSettingsConfig/WdgRtiKhz` must be a non-zero value. ```
1023	``` `Wdg/WdgSettingsConfig/WdgFpiKhz` must be a non-zero value. ```
1024	`Missing or unsupported driver. (Also verify the ECU description file's AUTOSAR XML namespace).`
1030	``` `/Wdg/WdgSettingsConfig/fSIRCkhz` must be a non-zero value. ```
1032	``Cannot generate code for unknown driver `%s`.``

	Verify module having `DEFINITION-REF` ending with `.../Wdg`.
1033	Watchdog IF references unknown driver `%s`. Verify module reference having a `DEFINITION-REF` ending with `...WdgIf/WdgIfDevice/WdgIfDriverRef.
1041	No device found for Watchdog `%s`. Verify elements defined by `..WdgIf/WdgIfDevice`.
1042	Device for `%s` platform has no device index. Verify elements defined by `.../WdgIfDeviceIndex`.
1046	Error trying to identify the Watchdog's platform
1047	Error trying to generate Wdg_MemMap.h.
1067	No default `WdgDefaultMode` defined.
1074	The specified ticks per second value for the used WdgMmode (%d Hz) and the RTI clock frequency (%d Hz) must satisfy the following condition: 0 < ticks_per_second <= rti_hz / 2.
1077	The `MPC5643L_MC33904` platform requires `WdgInitialTimeout` = 12 [ms].
1078	The `MPC5643L_MC33904` platform requires `WdgWindowStart` = 6 [ms].
1079	One of the WdgMTrigger elements associated with WdgMWatchdog `%s` has a WdgMTriggerConditionValue value which does not match the WdgInitialTimeout (%d ms) defined for the `%s` driver.
1080	One of the WdgMTrigger elements associated with WdgMWatchdog `%s` has a WdgMTriggerWindowStart value which does not match the `WdgWindowStart` (%d ms) defined for the `%s` driver.
1087	There are more than 1 watchdogs with an internal tick counter (%s)
1088	`%s` has an active internal tick counter but the Watchdog Interface does not reference it via `WdgIfInternalTickCounterRef`.
1089	The driver (`%s`) referenced by the Watchdog Interface via `WdgIfInternalTickCounterRef` has no active internal tick counter.
1117	`/Wdg/WdgSettingsConfig/SysClkKhz` must be a non-zero value.
1118	MPC56xx: using an internal tick counter requires configuring `STMdebugModeControl`.

5 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	AUTOSAR (AUTomotive Open System ARchitecture) is a worldwide development partnership of car manufacturers, suppliers and other companies from the electronics, semiconductor and software industry.
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
MCU	Microcontroller Unit
S-Wdg	Safe Watchdog Driver (implementation by TTTech)
S-WdgIf	Safe Watchdog Interface (implementation by TTTech)
S-WdgM	Safe Watchdog Manager (implementation by TTTech)

6 References

[1] AUTOSAR, *Specification of Watchdog Interface*. V. 2.2.2, Rel. 3.1, Rev. 1

[2] AUTOSAR, *Specification of Watchdog Interface*. V. 2.3.0, Rel. 4.0, Rev. 1

[3] TTTech Automotive GmbH, *Safe Watchdog Manager*, Safety Manual, V. 2.0.2

[4] TTTech Automotive GmbH, *Safe Watchdog Manager*, User Manual, V. 1.8.0

[5] TTTech Automotive GmbH, *Safe Watchdog Interface*, Safety Manual, V. 1.8.0

Index