MCAL User Manual for Wdg_17_Scu

32-bit TriCore™ AURIX™ TC3xx microcontroller

About this document

Scope and purpose

This User Manual is intended to enable users to integrate the Microcontroller Abstraction Layer (MCAL) software for the TriCore™ AURIX™ family of 32-bit microcontrollers.

This document describes responsibilities of integrator in-charge of integrating MCAL software with the basic software (BSW) stack. This document also provides detailed information on safety, configuration and functions along with examples of usage of significant features.

***Note**: Detailed information about package installation, safety and other generic information that are common across all modules are provided in MCAL User Manual General.*

Intended audience

This document is intended for anyone using the Wdg_17_Scu module of the TC3xx MCAL software.

Document conventions

Table 1 Conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, table and figure captions, screen names, windows, dialog boxes, menus, sub-menus
Italics	Denotes variable(s) and reference(s)
`Courier New`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
>	Indicates that a cascading sub-menu opens when you select a menu item
[cover parentID=<alpha numeric value>]	Used for traceability completeness. Reader should ignore these.

Reference documents

This User Manual should be read in conjunction with the following documents:

- AURIX™ TC3xx MCAL User Manual General
- Specification of WDG Driver, AUTOSAR_SWS_WDG_Driver, AUTOSAR Release 4.2.2
- Specification of WDG Driver, AUTOSAR_SWS_WDG_Driver, AUTOSAR Release 4.4.0

Table of contents

Table of contents

Table of contents

1 Wdg driver

1.1 User information

1.1.1 Description

In AUTOSAR, the watchdog stack implements three monitoring mechanisms: Alive supervision (for supervision of timing of periodic software), Deadline supervision (for aperiodic software), Logical supervision (for supervision of the correctness of the execution sequence). The Watchdog(WDG) driver implements watchdog SWS as specified by AUTOSAR, which provides services for initialization, changing the operation mode and setting the trigger condition (timeout), for use by the user of watchdog hardware (WDT). The users of the WDG driver are: Mode management modules (for initialization), WdgM through WdgIf for setting trigger condition and driver operation Mode setting. The WdgM implements the above mentioned supervision mechanisms. AUTOSAR extends triggering concept to support windowed watchdog mode. This calls for use of hardware timers, to self-trigger WDT until the user-set trigger condition is met. For this purpose, the WDG driver provides options to select one of the two timers, either STM or GTM timers. Since code base would be common for all CPUs, to avoid presence of unused code, the timer selection is made static (precompile) for all WDTs (the driver uses either GTM or STM for all WDTs).

1.1.2 Hardware-software mapping

This section describes the system view of the WDG driver and peripherals administered by it.

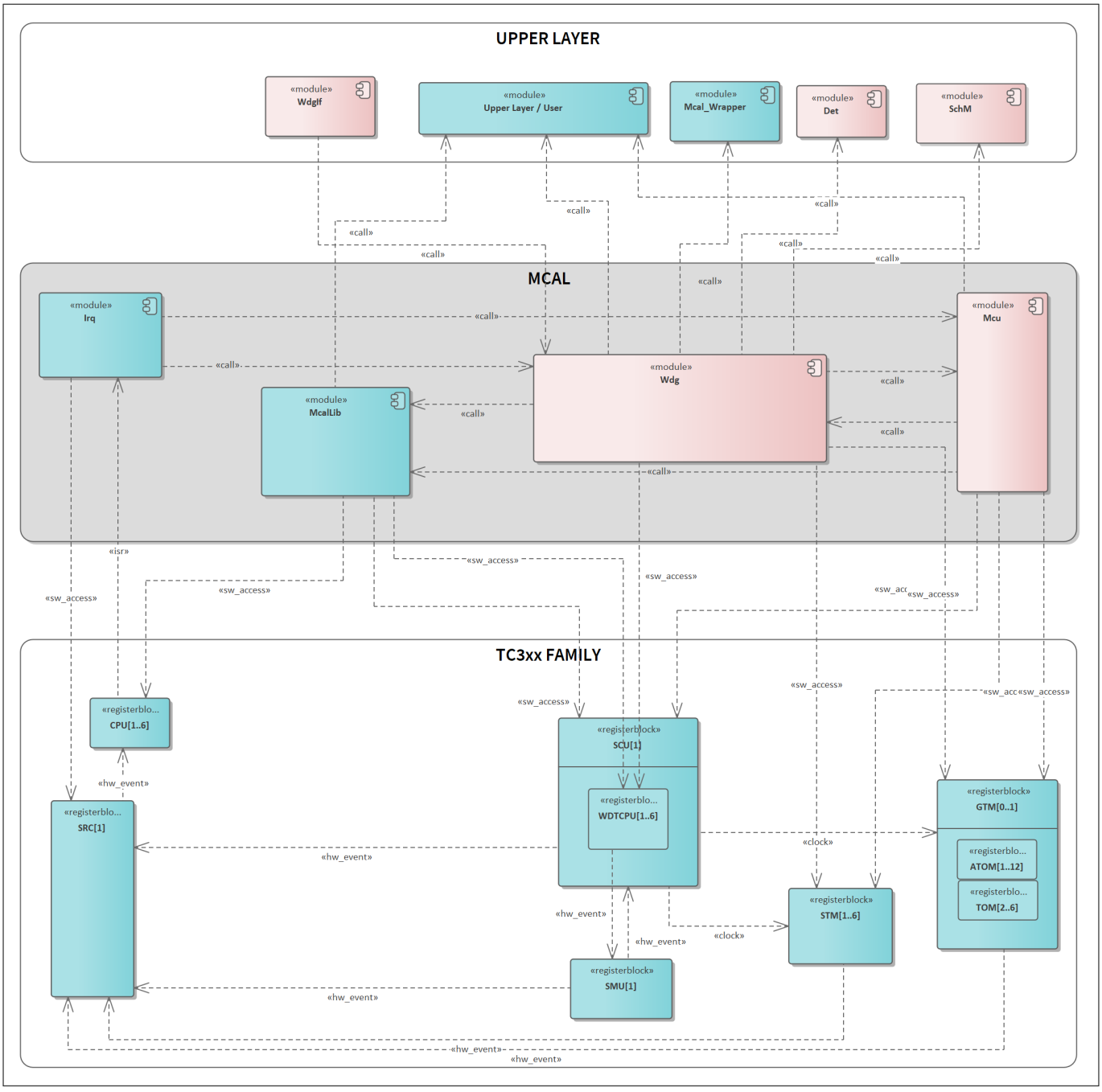

Figure 1 Mapping of hardware-software interfaces

1.1.2.1 GTM: dependent hardware peripheral

Hardware functional features

The WDG driver depends on the GTM for compare match event, which is required for servicing the WDT.

Each WDT is mapped to a unique TOM/ATOM channel.

The GTM TOM/ATOM functional block feature configured/accessed by the WDG driver is:

ATOM/TOM compare match interrupt events feature is used to achieve WDG functionality. Rest of the GTM features are not used.

Users of the hardware

The TOM/ATOM channels are used by the WDG, PWM, GPT and ADC drivers. To avoid conflicts for common GTM registers, the access requests to the GTM channels is performed via the MCU driver. For GTM channel specific registers, WDG exclusively own it and use to restart the GTM counters.

Hardware diagnostic features

Not applicable

Hardware events

The WDG driver uses the following hardware event from the GTM:

- Compare match event is used from the GTM to service the WDT.

1.1.2.2 STM: dependent hardware peripheral

Hardware functional features

The WDG driver depends on the STM for the compare match event which is required for servicing the watchdog timer.

The STMxTIM0 (x = Core Id) register is used by watchdog timer for calculating mode-specific refresh time to service the watchdog timer. Each watchdog timer is mapped to a unique STM timer compare register.

Users of the hardware

The McalLib, MCU and WDG drivers are users of the STM. To avoid conflicts, the access to the STM registers takes place through APIs provided by the MCU driver.

Hardware diagnostic features

Not applicable.

Hardware events

The WDG driver uses the following hardware events from the STM for compare match:

- STMIR0 is used when CMP0 is configured for the watchdog
- STMIR1 is used when CMP1 is configured for the watchdog

1.1.2.3 SMU: dependent hardware peripheral

Hardware functional features

The following features of SMU can be used

- An overflow of the WDT can trigger an alarm request to the SMU
- SMU can be configured to reset the system in case of the WDT overflow

Users of the hardware

The SMU configuration is handled only by the SMU driver.

Hardware diagnostic features

The SMU alarms are not monitored by the WDG driver.

Hardware events

Not applicable.

1.1.2.4 SCU: primary hardware peripheral

Hardware functional features

The TC3xx contains the following WDTs:

- One safety WDT
- One WDT per CPU

The WDG driver uses the SCU IP for the clock, ENDINIT and reset functionalities. The driver requires the fSPB clock signal for functioning.

The key SCU features used by the WDG driver are:

- The WDG driver uses the WDTs for servicing at defined intervals as per application requirements.
- 16-bit WDT counter is used to implement watchdog functionality (WDTCPUxSR).
- 16-bit user-definable reload value is used for normal WDT operation (WDTCPUxCON0). The reload value for normal mode shall be computed based on trigger condition at run time (A fixed reload value provided in the hardware for timeout mode shall only be used for updating CPU critical registers).
- Supports disabling of WDT(WDTCPUxCON1).
- Selectable input frequency is fSPB/64, fSPB/256 or fSPB/16384 (WDTCPUxCON1). Frequency fSPB/64 is not used (fast mode frequency is fSPB/256 and slow mode frequency is fSPB/16384).

The unsupported features of the WDT are:

- Safety WDT (Support is provided for CPU WDT only).
- Incorporation of the corresponding ENDINIT bit and monitoring of its modifications. The WDG ENDINIT feature will only be used for CPU critical registers and will not be supported by WDG driver.
- Automatic password sequencing mechanism is not supported. A fixed reload password will be used.
- The UR bit in WDTCPU0CON1 register will not be configurable through WDG driver. Updating this bit depends on the state of the SMU which is not monitored by the WDG driver.
- Time Check Password feature is not supported.

Users of the hardware

The SCU IP supplies clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, updates to all the ENDINIT protected registers are performed using the MCALLIB APIs.

The WDG and MCALLIB drivers exclusively utilize the WDT.

The MCALLIB driver uses the WDT to implement the ENDINIT protection feature in the timeout mode.

To avoid conflicts, the WDG and MCALLIB drivers use the critical section with the same exclusive area name while accessing the shared resources. User shall ensure that core-specific interrupts are disabled in the critical sections since CON0 register is updated inside these critical sections by both the drivers. User shall ensure that the software on each core accesses only the core-specific WDT registers.

Hardware diagnostic features

Following are the hardware error status flags monitored in the WDG driver:

Overflow error detection.

Access error detection.

The SMU alarms configured for the WDG are not monitored by the WDG driver.

Hardware events

Hardware events from the SCU are not used by the WDG driver.

1.1.3 File structure

1.1.3.1 C file structure

This section provides details of the C files of the WDG driver.

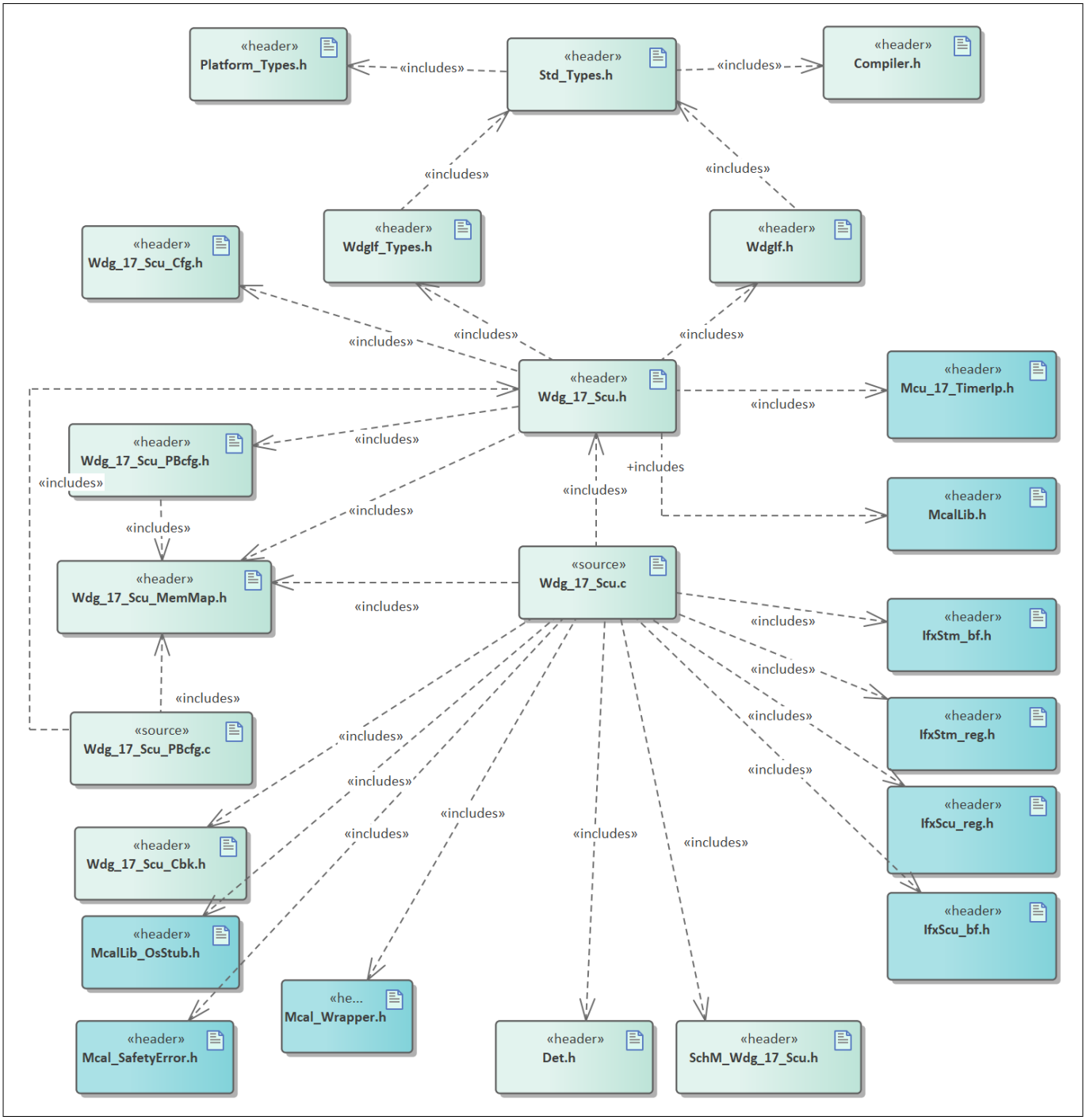

Figure 2 Wdg_C_File_Structure-1.png

Table 2 C file structure

File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Det.h	Provides the exported interfaces of Development Error Tracer
IfxScu_bf.h	SFR header file for SCU
IfxScu_reg.h	SFR header file for SCU
IfxStm_bf.h	SFR header file for STM
IfxStm_reg.h	SFR header file for STM
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB.
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore. This shall be included by other drivers to call OS APIs.
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Mcal_Wrapper.h	Provides the exported interfaces for Production Error and Runtime Development Errors. Implemented by default to include functions of Dem.h and Det.h files. This file can be modified by the user but function prototype is not user modifiable.
Mcu_17_TimerIp.h	Header file defining prototypes of data structures and APIs of Timer IPs (GTM, CCU6 and GPT12), containing functions such as initialization, enable, interrupt handlers and other services and is included by Mcu_17_TimerIp.c source file
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Wdg_17_Scu.h	Export header for SchM functions of WDG driver.
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.
WdgIf.h	Provides type definitions of the Watchdog Interface *Note: This file is included only for AUTOSAR version 4.4.0*
WdgIf_Types.h	Provides type definitions of the Watchdog Interface *Note: This file is included only for AUTOSAR version 4.2.2*
Wdg_17_Scu.c	Source file providing implementation of APIs.
Wdg_17_Scu.h	Header file providing prototypes of APIs, data types and interrupt handlers.
Wdg_17_Scu_Cbk.h	Header file contains call back function of the WDG driver.
Wdg_17_Scu_Cfg.h	Generated header file containing macros.
Wdg_17_Scu_MemMap.h	File (Static) containing the memory section definitions used by the WDG driver.
Wdg_17_Scu_PBcfg.c	Generated source file containing configuration data of the user.
Wdg_17_Scu_PBcfg.h	Generated header file containing configuration data of the user.

1.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the WDG driver.

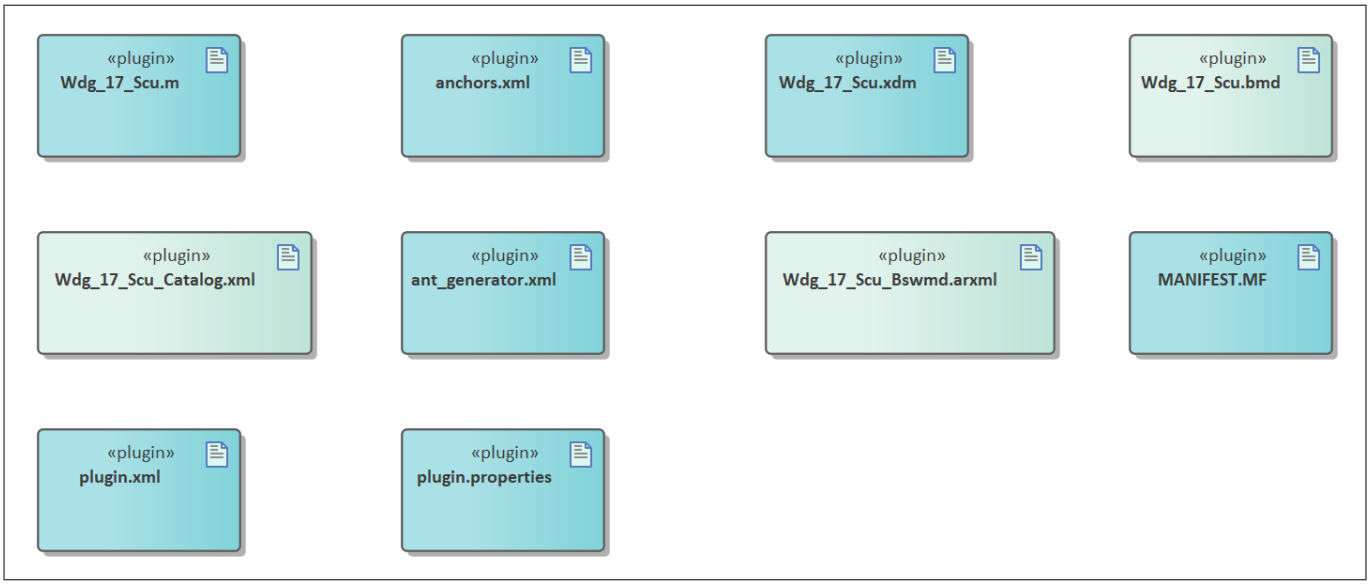

Figure 3 **Wdg_Code_Generator_Plugin_Files-1.png**

Table 3 **Code generator plugin files**

File name	Description
`MANIFEST.MF`	Tresos plugin support file containing the metadata for the WDG driver.
`Wdg_17_Scu.bmd`	AUTOSAR format XML data model schema file (for each device).
`Wdg_17_Scu.m`	Code template macro file for WDG driver.
`Wdg_17_Scu.xdm`	Tresos format XML data model schema file.
`Wdg_17_Scu_Bswmd.arxml`	AUTOSAR format module description file.
`Wdg_17_Scu_Catalog.xml`	AUTOSAR format catalog file.
`anchors.xml`	Tresos anchors support file for the WDG driver.
`ant_generator.xml`	Tresos support file to generate and rename multiple post-build configuration when using variation point.
`plugin.properties`	Tresos plugin support file for the WDG driver.
`plugin.xml`	Tresos plugin support file for the WDG driver.

1.1.4 Integration hints

This section lists the key points that an integrator or user of the WDG driver must consider.

1.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the WDG driver.

- **EcuM**

  The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of the ECU. Specifically, in the context of MCAL, EcuM is used for initialization and de-initialization of the software drivers.The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.
- **Memory mapping**

Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Wdg_17_Scu_MemMap.h` file.

The `Wdg_17_Scu_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements are re-located to the correct memory region. A sample implementation listing the memory-section macros is shown as follows:

```
#define MEMMAP_ERROR
/*To be used for all global or static variables.*/
/*Variable to be cleared at startup or reset is placed here - .bss*/
#if defined WDG_17_SCU_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
 /*User pragmas here*/
 #undef WDG_17_SCU_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
 #undef MEMMAP_ERROR

#elif defined WDG_17_SCU_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
 /*User pragmas here*/
 #undef WDG_17_SCU_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_32
 #undef MEMMAP_ERROR

/*Constants with attributes that show that they reside in one segment for module
configuration.*/
#elif defined WDG_17_SCU_START_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
/*User pragmas here*/
 #undef WDG_17_SCU_START_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
 #undef MEMMAP_ERROR

#elif defined WDG_17_SCU_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
 /*User pragmas here*/
 #undef WDG_17_SCU_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
 #undef MEMMAP_ERROR

/* Code Sections */
#elif defined WDG_17_SCU_START_SEC_CODE_ASIL_B_GLOBAL
/*User pragmas here*/
 #undef WDG_17_SCU_START_SEC_CODE_ASIL_B_GLOBAL
 #undef MEMMAP_ERROR

#elif defined WDG_17_SCU_STOP_SEC_CODE_ASIL_B_GLOBAL
/*User pragmas here*/
 #undef WDG_17_SCU_STOP_SEC_CODE_ASIL_B_GLOBAL
 #undef MEMMAP_ERROR

#endif
#if defined MEMMAP_ERROR
#error "Wdg_17_Scu_MemMap.h, wrong pragma command"
#endif
```

- **DET**

The DET module is a part of the AUTOSAR stack that handles all the development errors reported by the BSW modules. The WDG driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the WDG driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **Mcal_Wrapper**

  This Driver performs reporting of the Production and Runtime errors. The Handling of the reported errors shall be done by the user. The `Mcal_Wrapper_Det_ReportRuntimeError()` API, `Mcal_Wrapper_Dem_SetEventStatus()` API and `Mcal_Wrapper_Dem_ReportErrorStatus()` API are provided in the `Mcal_Wrapper.c` and `Mcal_Wrapper.h` files as a stub code, and can be updated by the integrator to handle the reported errors. The files `Mcal_Wrapper.c` and `Mcal_Wrapper.h` are user modifiable but function prototype is not user modifiable and by default the Mcal Wrapper function shall calls AUTOSAR DEM and DET Modules.

  The user of the WDG driver shall process all the production errors (fail/pass) reported to the Mcal_Wrapper module. The interface used for reporting production error In AUTOSAR version 4.2.2 is `Mcal_Wrapper_Dem_ReportErrorStatus()` and for AUTOSAR version 4.4.0 is `Mcal_Wrapper_Dem_SetEventStatus()` API. The `Mcal_Wrapper.c` and `Mcal_Wrapper.h` files are provided in the MCAL package as a stub code and can be replaced with a user specific production error handling module during the integration phase. The WDG driver process no runtime errors and no runtime errors are reported.

- **SchM**

  The SchM module is a part of the RTE that manages the BSW Scheduler. The WDG driver uses the exclusive areas defined in `SchM_Wdg_17_Scu.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the WDG driver are:

  - TimerHandling
  - ChangeMode
  - CpuEndInit

  The `SchM_Wdg_17_Scu.h` and `SchM_Wdg_17_Scu.c` files are provided in the MCAL package as an example code and needs to updated by the integrator. The user must implement the SchM functions defined by the WDG

driver as **suspend / resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is shown as follows:

```
/*Sample implementation of SchM_Wdg_17_Scu.c*/

#include "IFX_Os.h"
#include "SchM_Wdg_17_Scu.h"

void SchM_Enter_Wdg_17_Scu_TimerHandling(void)
{
 SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Wdg_17_Scu_TimerHandling(void)
{
 ResumeAllInterrupts(); /* Resume CPU core interrupt */
}

void SchM_Enter_Wdg_17_Scu_ChangeMode(void)
{
 SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Wdg_17_Scu_ChangeMode(void)
{
 ResumeAllInterrupts(); /* Resume CPU core interrupt */
}

void SchM_Enter_Wdg_17_Scu_CpuEndInit(void)
{
 SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Wdg_17_Scu_CpuEndInit(void)
{
 ResumeAllInterrupts(); /* Resume CPU core interrupt */
}
```

- **Safety error**

  The WDG driver will report all the detected safety errors through the `Mcal_ReportSafetyError()` API.

  The driver performs only detection and reporting of the safety errors. The handling of the reported errors shall be done by the user. The `Mcal_ReportSafetyError()` is provided in the `Mcal_SafetyError.c` and `Mcal_SafetyError.h` files as a stub code, and must be updated by the integrator to handle the reported errors.

  Note: All DET errors are also reported as safety errors (error code used is same as DET).
- **Notifications and call backs**

  The WDG driver does not expect any call backs from application but it needs a call back from MCU driver for ISR handling.
- **Operating system (OS)**

The OS or the application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or the application.

The OS files provided by the MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

1.1.4.2 Multicore and Resource Manager

Multicore concept for WDG driver

Every CPU has one dedicated WDT. The global status variable is used to maintain the core-specific reload value, status (IDLE, BUSY, UNINIT), timeout counter value and mode (FAST, SLOW, OFF) of the WDT. Initialization is core specific (CPUx can update/modify registers and variables for WDTx only. x = 0,1,2,3,4,5). At least one CPU WDT must be configured if the WDG driver is used in an application. The key points to be considered with respect to multicore in the driver are as follows:

- Locating the constants, variables and configuration data to correct memory space should be carried out by the user. Memory sections are marked GLOBAL (common to all cores) and CORE[x] (specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

  Code section:

  The executable code of the WDG driver is placed under single MemMap section. It can be relocated to any PFlash region.

  Data section:

  The RAM variable memory sections marked as specific to a core should be relocated to the DSPR/DLMU of the same core. The sections marked as global should be relocated to the non-cached LMU region.

  Configuration data and constants:

  The configuration data sections marked as specific to a core should be relocated to the PFlash of the same core. The sections marked as global should be relocated to the PFlash of the master core.

  Note: *Relocating code, data or constants to a distant memory region would impact execution timings.*

  Note: *If the driver operates from a single (master) core, all the sections may be relocated to the PFlash/DSPR/DLMU of the same CPU core.*

Resource manager for WDG

The user should allocate resources of WDG to CPU cores at pre-compile time using the Resource manager module. WDG driver depends on resource manager if STM timer is configured for its timing needs. An STM timer should be allocated to the same core in the resource manager for which a CPU WDG is configured.

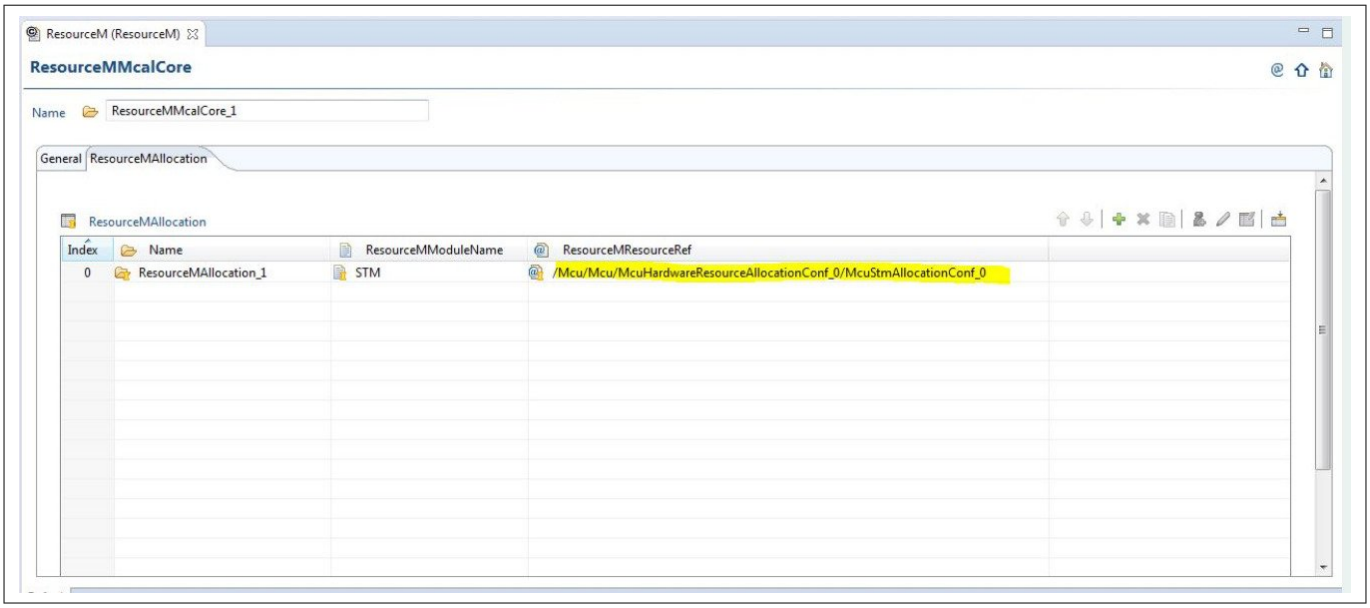

Figure 4 STM timer allocation in Resource Manager

1.1.4.3 MCU support

Timer support

The WDG driver is dependent on the MCU driver for clock configuration and GTM and STM timer services. The initialization of the WDG driver must be started only after completion of the MCU initialization.

Figure 5 WDG configuration with a reference to GTM TOM/ATOM channel in MCU

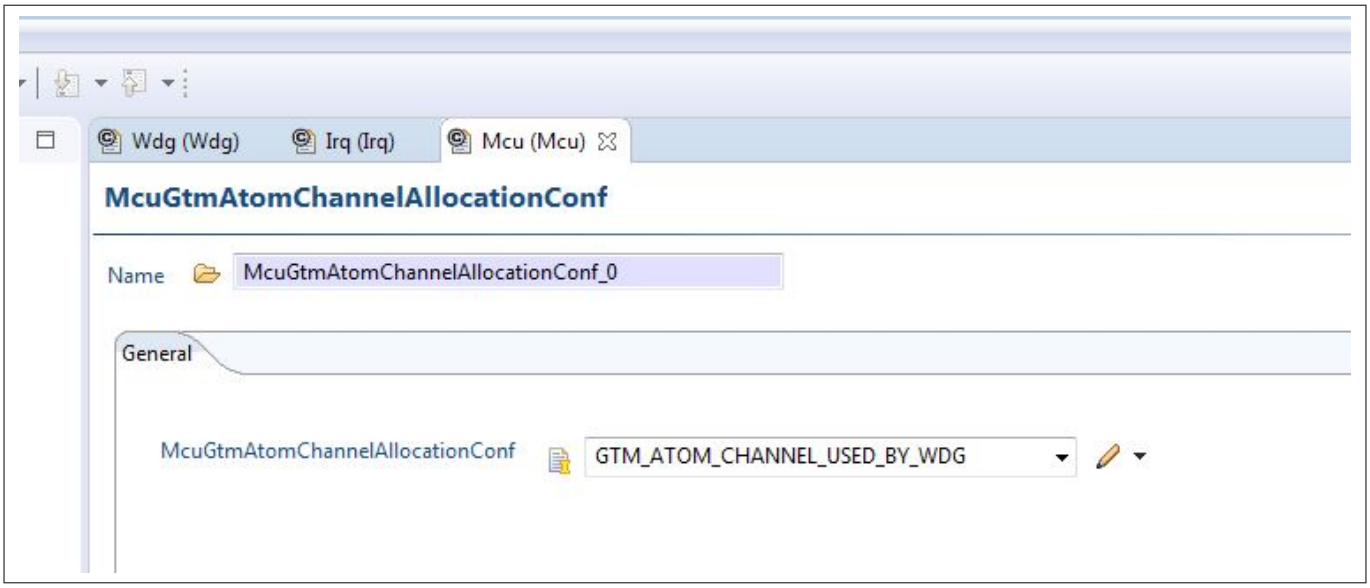

Figure 6 MCU configuration for GTM TOM/ATOM channel

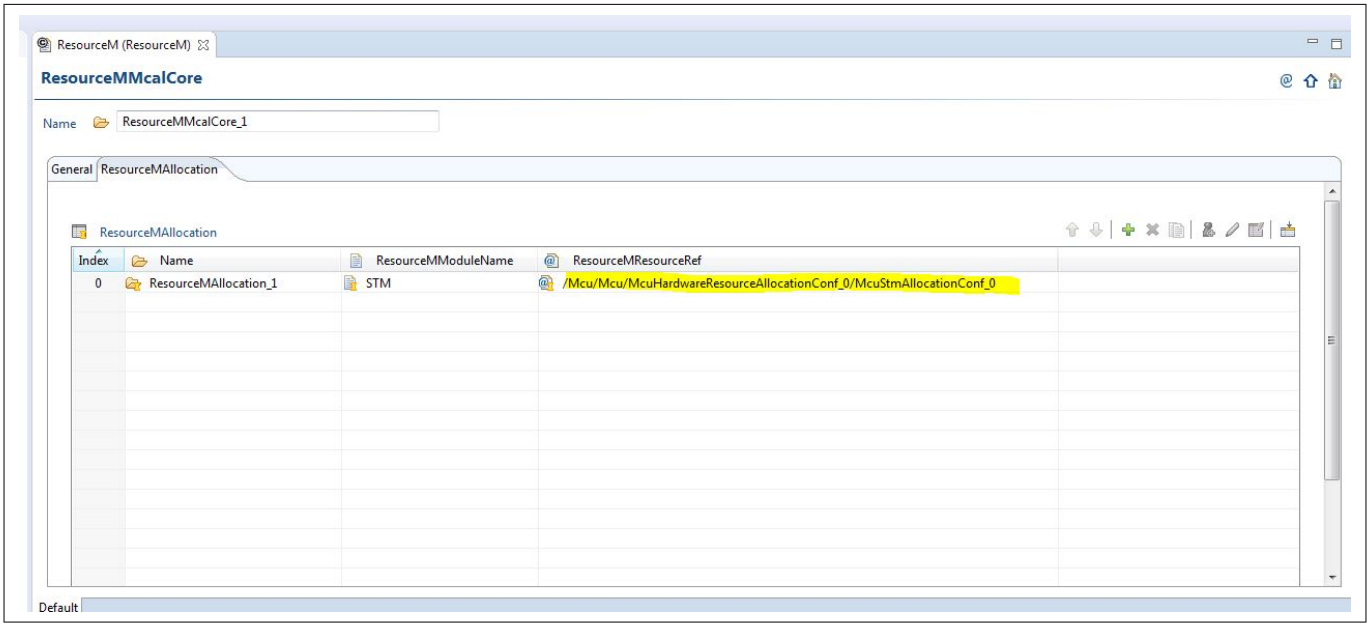

Figure 7 STM timer reference in Resource Manager to MCU

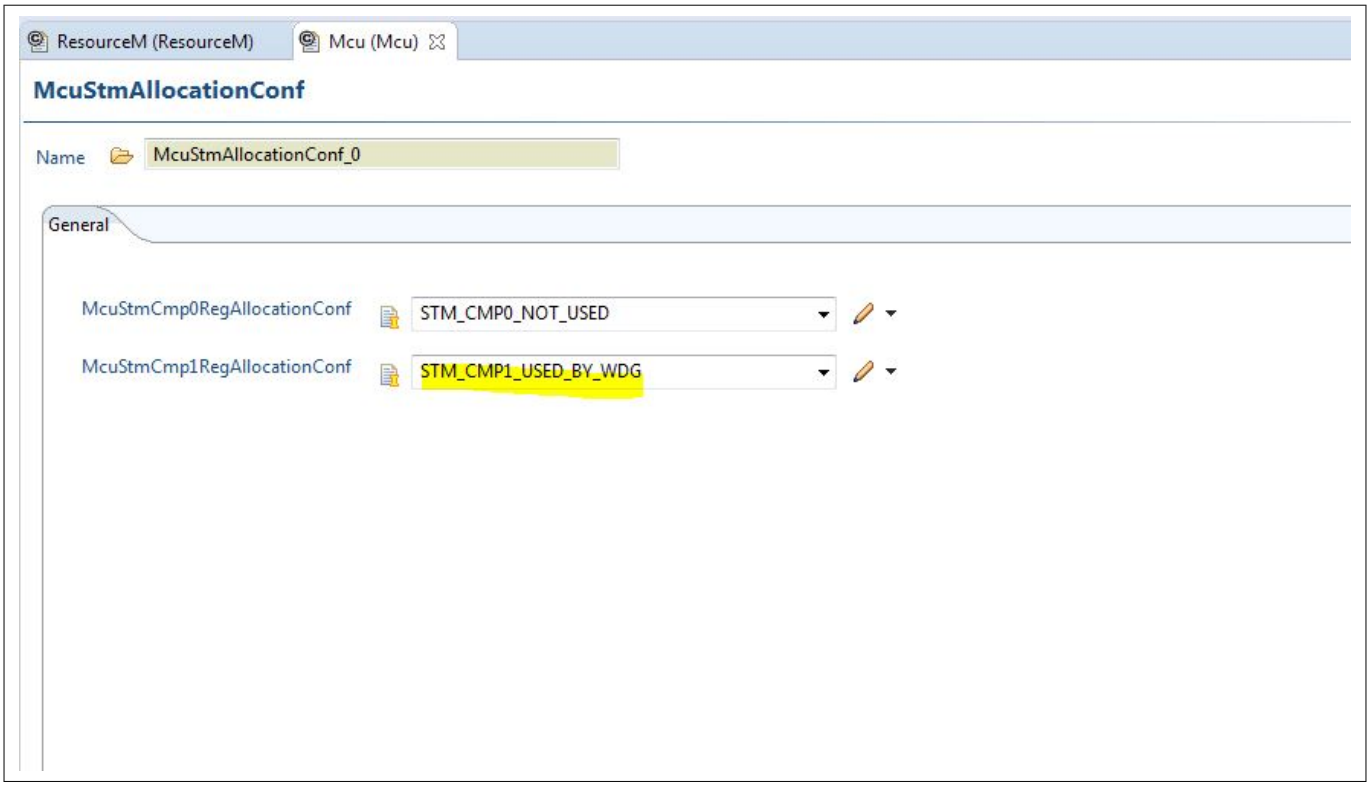

Figure 8 STM Compare Register allocation for WDG in MCU

1.1.4.4 Port support

The WDG driver does not use any services provided by the PORT driver.

1.1.4.5 DMA support

The WDG driver does not use any services provided by the DMA driver.

1.1.4.6 Interrupt connections

The interrupt connections of the Wdg driver are described in this section.

GTM timer

If GTM timer is used, the priority number for the interrupt must be configured in the IRQ module based on the TOM or ATOM channel configured.

For example, if ATOM 0 and Channel 0 is used for WDT, the interrupts have to be configured as shown in the following image:

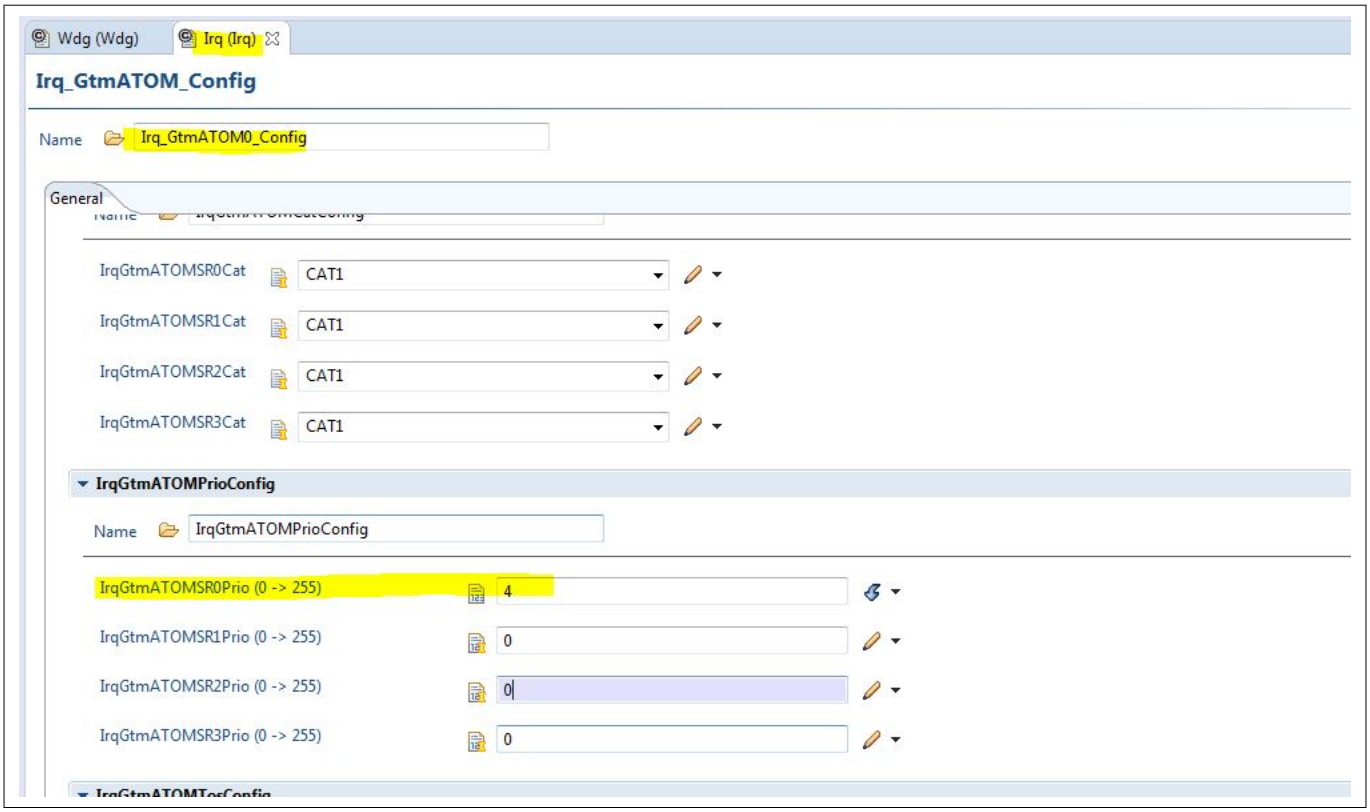

Figure 9 IRQ module configured for interrupt from ATOM 0 Channel 0 with priority number 4

Invoking the interrupt handlers provided by the driver must be done by the user. A sample invocation is shown as follows:

```
/* include MCU timer header file */
#include "Mcu_17_TimerIp.h"
/*********SRC_ GTMTOM0SR0*********/
ISR(GTMTOM0SR0_ISR)
{
/* Enable Global Interrupts */
 ENABLE();
 /* Parameter is Channel Number */
 Mcu_17_Gtm_TomChannelIsr(0, 0);
}

/*********SRC_ GTMATOM0SR0*********/
ISR(GTMATOM0SR0_ISR)
{
 /* Enable Global Interrupts */
 ENABLE();
 /* Parameter is Channel Number */
 Mcu_17_Gtm_AtomChannelIsr(0, 0);
}
```

STM timer

If the STM timer is used, the priority number for the interrupt must be configured in the IRQ driver.

For example, if STM used for WDT, the interrupts have to be configured as shown in the following image:

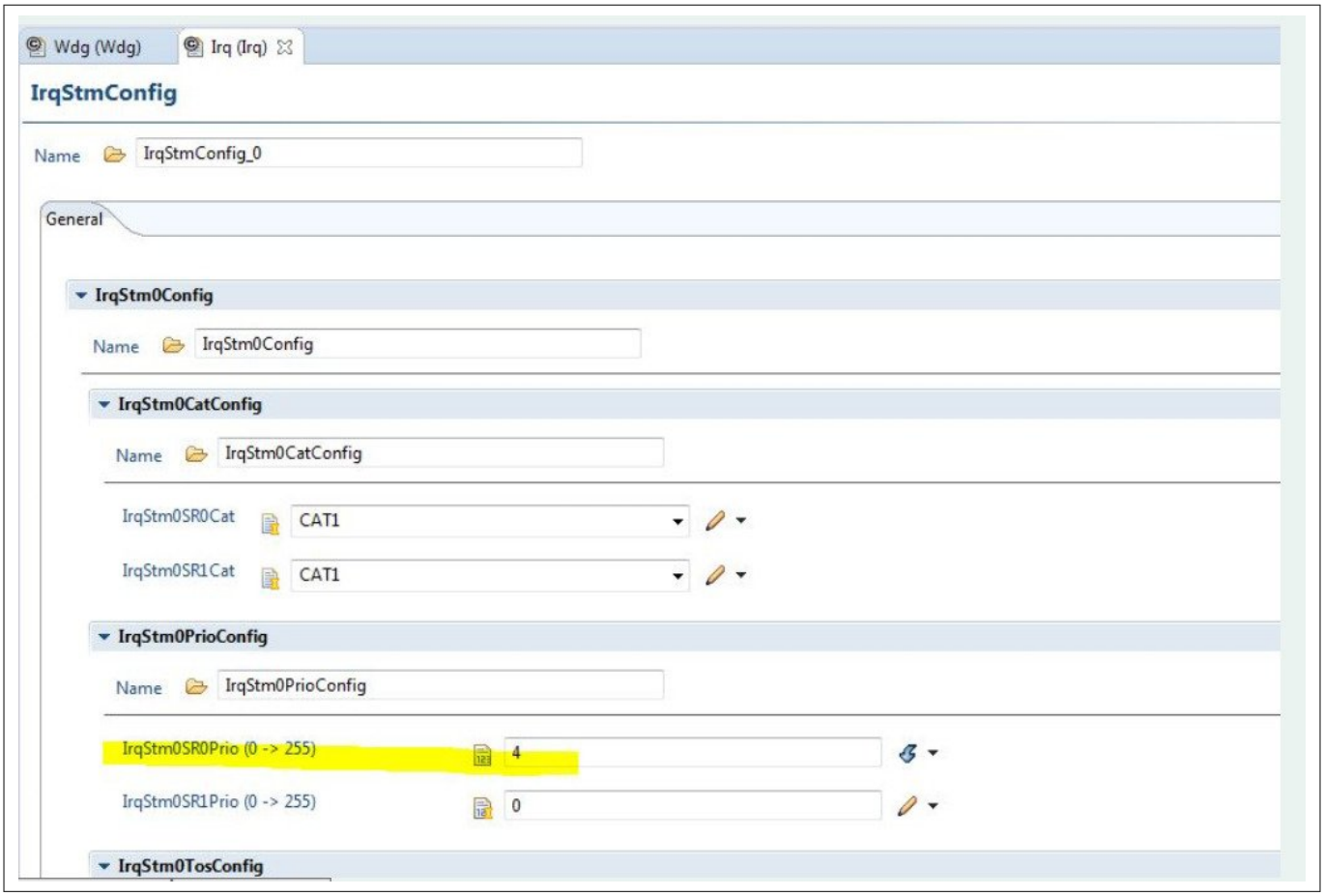

Figure 10 IRQ module configured for interrupt from to STM0_CMP0 with priority number 4

Invoking the interrupt handlers provided by the driver must be done by the user. A sample invocation is shown as follows:

```
/* include MCU timer header file */
#include "Mcu_17_TimerIp.h"
/*********SRC_ STM0SR0*********/
ISR(STM0SR0_ISR)
{
 /* Enable Global Interrupts */
 ENABLE();
 /* Call Interrupt function */
 Mcu_17_Stm_CompareMatchIsr(0U, 0U);
}
/*********SRC_ STM1SR0*********/
ISR(STM1SR0_ISR)
{
 /* Enable Global Interrupts */
 ENABLE();
 /* Call Interrupt function */
 Mcu_17_Stm_CompareMatchIsr(1U, 0U);
}
```

1.1.4.7 Example usage

Configuration

All the configuration parameters in the WDG driver mentioned in the Reference information section must be configured. If the GTM timer is used, the MCU must be configured with the TOM/ATOM channel and the corresponding IRQ must be configured with a priority number. If the STM timer is used, the MCU and Resource Manager must be configured with the corresponding STM compare register and the STM timer to the core, respectively. Also, the corresponding IRQ must be configured with a priority number.

Code must be generated without any errors.

Summery of WDG driver time related configuration parameters and their relationship		
Configuration Parameter	**Range**	**Relationship (must be adhered to avoid code generation errors)**
WdgCPUInitialTimeout	0.001s - 65.535s	**Less Than** WdgCPUMaxTimeout
WdgCPUMaxTimeout	0.001s - 65.535s	**NA**
WdgFastModeTimeoutValue	0.001s - 0.16777216s (fSPB = 100MHz, Clock divider = 256)	**Less Than** WdgSlowModeTimeoutValue. **Greater Than** WdgFastRefreshTime.
WdgSlowModeTimeoutValue	0.001s - 10.73741824s (fSPB = 100MHz, Clock divider = 16384)	**Greater Than** WdgSlowtRefreshTime.
WdgFastRefreshTime	0.001s - WdgFastModeTimeoutValue	**Less Than** WdgFastModeTimeoutValue and WdgSlowRefreshTime
WdgSlowRefreshTime	0.001s - 10.73741824s	**Greater Than** WdgFastRefreshTime **Less Than** WdgSlowModeTimeoutValue

Figure 11 Summary of WDG driver time related configuration parameters and their relationship

Initialization of the WDG driver

Step 1. Include `Wdg_17_Scu.h` header file, to include extern definition of WDG driver configuration data structure.

Step 2. Include `Mcu.h` header file, To include extern definition of MCU driver configuration data structure.

Step 3. Initialize the MCU driver.

```
Mcu_Init(&Mcu_Config);

 Mcu_InitClock(0); /* Initialize PLLs and Clocks by passing clock configuration index */

 while(Mcu_GetPllStatus() != MCU_PLL_LOCKED); /* wait until PLL is locked */

 Mcu_DistributePllClock(); /* change clock source as PLL and ramp up/down to configured clock
frequencies */
```

Step 4. Initialize the IRQ driver and set the SRE bit.

Step 5. Initialize the WDG driver for the current core.

Note: SMU alarm group 8 can be configured to take an action based on the WDT alarms.

```
Wdg_17_Scu_Init(&Wdg_17_Scu_Config_0);
```

Normal operation of WDG driver

Step 1. After initializing MCU and WDG drivers, the mode of the WDG driver can be changed to FAST or SLOW.

Note: Mode of the WDG driver for a core can be modified to OFF mode only if WdgCPUDisableAllowed is enabled, else DET and Production error shall be raised.

```
Wdg_17_Scu_SetMode(WDGIF_FAST_MODE) ;
Wdg_17_Scu_SetMode(WDGIF_SLOW_MODE) ;
```

```
Wdg_17_Scu_SetMode(WDGIF_OFF_MODE); /*to turn OFF the WDG driver at run time if
WdgCPUDisableAllowed is enabled*/
```

Step 2. Call the `Wdg_17_Scu_SetTriggerCondition(timeout in ms)` API to service the WDG timer at every logical step.

Note:Timeout passed as an input parameter to the API must be greater than the mode specific refresh time configured by the user. Otherwise, the trigger counter will be zero, WDT will not be serviced and this will lead to WDG alarm when the timer overflows.

```
Wdg_17_Scu_SetTriggerCondition (1000); /*timeout in ms*/
```

Note : Once the WDG counter expires and if this API is not called then WDG timer will overflow. CPU will reset if SMU is configured to take the necessary action based on the WDG Alarm.

STM timer plausibility check

The application software shall perform a plausibility check of the STM counter value with a different timer/ counter, before using it.

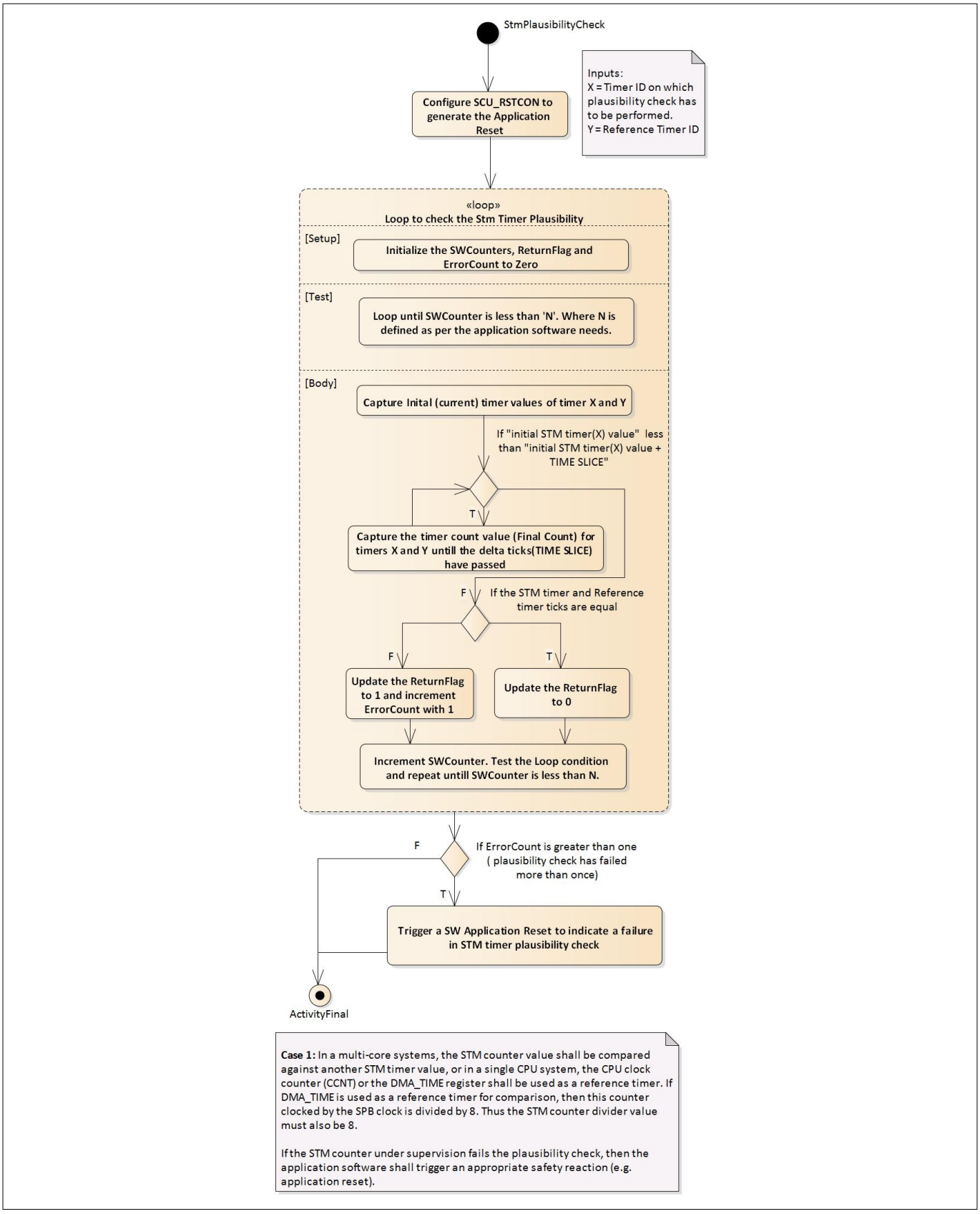

Figure 12 An example to perform STM timer plausibility check

```
/* Case1: Example to implement STM plausibility check */
```

```
/*****************************************************************************
** Example to implement STM plausibility check **
** **
** Syntax : void Stm_PlausiblityCheck(const uint32 StmTimer, **
** const uint32 ReferenceTimer) **
** **
** Description : This function performs plausibility check of STM **
** counter by comparing with another STM timer. **
** **
** **
** Parameters(in) : StmTimer : STM timer ID to on which plausibility **
** check has to be performed. **
** ReferenceTimer :Reference STM Timer ID **
** **
** Parameters (out) : none **
** **
** Return value : 0: If the Plausibility Check is Successful **
** 1: If the Plausibility Check is Unsuccessful **
** **
*****************************************************************************/
uint8 Stm_PlausiblityCheck(const uint32 StmTimer, const uint32 ReferenceTimer)
{
 uint8 ReturnFlag = 0x1U;
 uint8 ErrorCount = 0x0U;
 SWCounter = 0x0U;

 /*Configure SCU RSTCON to generate Application reset*/
 Ifx_SCU_SWRSTCON SwResetReq;
 Ifx_SCU_RSTCON ResetConfig;
 ResetConfig.U = SCU_RSTCON.U;
 ResetConfig.B.SW = 2U;

 Mcal_WriteSafetyEndInitProtReg(&SCU_RSTCON.U,(uint32)ResetConfig.U);

 /*Loop N times. Until SWCounter is less than STM_N_CHECK_COUNT*/
 while( SWCounter < STM_N_CHECK_COUNT )
 {
 /*Capture the initial timer values*.
 X = Timer ID on which plausibility check has to be performed.
 Y = Reference Timer ID */
 X1 = StmTim0Info[StmTimer]->U;
 Y1 = StmTim0Info[ReferenceTimer]->U;
 /*Loop until the STM counter value has reached the final timer value*/

 while(StmTim0Info[StmTimer]->U < (X1 + STM_TIME_SLICE_COUNT))
 {
 /*Capture count till the final timer values is equal to
```

```
 STM_TIME_SLICE_COUNT + initial timer value */
 X2 = StmTim0Info[StmTimer]->U;
 Y2 = StmTim0Info[ReferenceTimer]->U;
 }

 /*If the STM timer and Reference timer ticks are equal*/
 if((uint32)(X1-X2) == (uint32)(Y1-Y2))
 {
 ReturnFlag = 0x00U; /*Plausibility Check is successful*/
 }
 else
 {
 ReturnFlag = 0x01U; /*Plausibility Check is unsuccessful*/
 ErrorCount+=0x01U; /* Increment the Error Count */
 }
 SWCounter++;
 }

 /*If plausibility check has failed more than once.Trigger an
 application reset*/
 if(ErrorCount > 0x1U)
 {
 /*Trigger a SW Application Reset to indicate a consistent
 failure in STM timer value check*/
 SwResetReq.U = SCU_SWRSTCON.U;
 SwResetReq.B.SWRSTREQ = 0x1U;
 Mcal_WritePeripEndInitProtReg(&SCU_SWRSTCON.U,(uint32)SwResetReq.U);
 }
 return ReturnFlag;
}
```

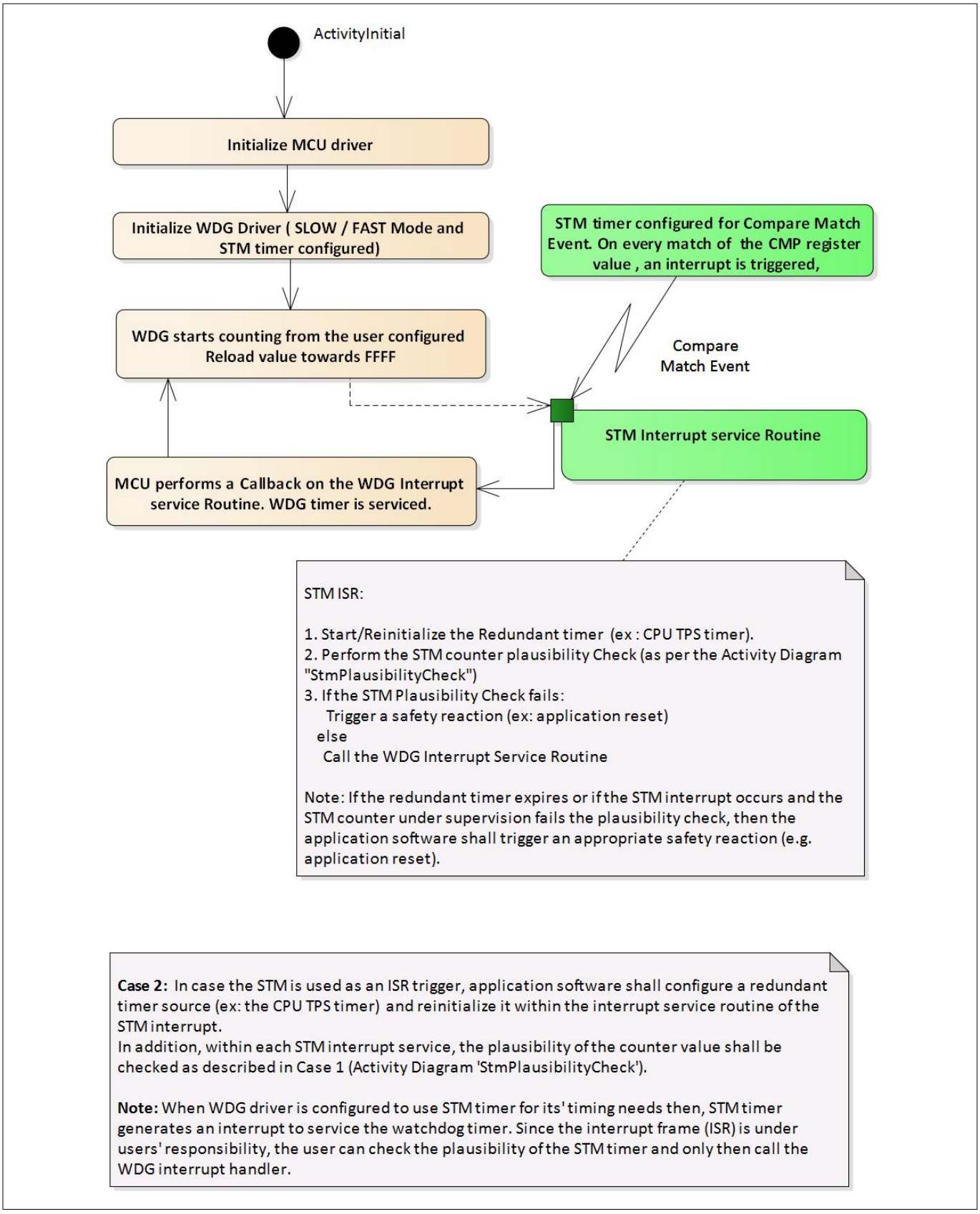

Figure 13 Activity diagram depicting an example to perform STM timer plausibility check and ensure interrupt triggering time if in case STM timer is used as an ISR trigger

1.1.5 Key architectural considerations

1.1.5.1 WDG driver states

The state of the WDG driver will be updated to WDG_UNINIT or WDG_BUSY or WDG_IDLE irrespective of whether the DET or safety error check is enabled or not. The driver state check in the APIs will prevent any unintended SFR or global update.

1.1.5.2 WDG driver critical section

The SCU_WDTCPUxCON0 (x is core Id) register is accessed by both MCALLIB and WDG (MCALLIB driver ensures that the REL bits of this register is not hampered and it writes back with the correct values once it exits the timeout mode). MCALLIB uses this register in the hardware timeout mode and WDG uses it in the hardware normal mode. Thus MCALLIB and WDG use the critical section with the same exclusive area (CpuEndinit).

1.1.5.3 WDG driver timer configuration and services

Every WDT configured for a core must be accompanied with an STM timer/GTM timer resource. All STM and GTM related registers will be configured by invoking MCU driver APIs.

If STM timer is used, the timer is allocated to the core using the Resource manager and the compare register is allocated using the MCU driver. The STM compare register used for WDT needs must be configured in the MCU driver. STM compare match interrupt from CMP0 will be routed through IR0 and CMP1 will be routed through IR1.

If GTM timer is used, the TOM/ATOM channel is allocated in the MCU driver and the respective clock is configured in the WDG driver. All the STM/GTM related registers required by the WDG driver are updated using services provided by the MCU driver. The source of the interrupt will be CMP0 (compare match interrupt) and not CMP1 (period match interrupt). The WDG driver mode specific reload values (SLOW/FAST) provided in the configuration is the timeout in seconds used by the WDG driver and the mode specific refresh (SLOW/FAST) values are the GTM/STM timer callback periods.

1.1.5.4 WDG driver timer dependency

Among the available timers, the GTM or STM timer is chosen to trigger the WDT. The GTM and STM have sufficient number of timers to support WDT on multiple cores. The GPT12 and CC6 timers are not considered due to the unavailability of the required number of timers and due to specific use cases of these timers. In an application, it could be possible that one STM timer is allocated per CPU. The same timer can be used by CPU WDT and is also accessed by the OS. Since the STM timer hardware provides two timer interrupts (CMP0, CMP1), one interrupt could be used by OS and the other by WDT.

1.1.5.5 WDG driver multicore error reporting

When the WDG initialization is done for a wrong core, the WDG_E_PARAM_CONFIG development or safety error is reported.

1.1.5.6 WDG driver password access

The internal WDG hardware shall be accessed in the hardware normal mode by the WDG driver with a static password, which is configured by the user. The entire password access sequence is handled by the WDG driver. The WDG hardware supports both static password and automatic password sequence mechanism. However, according to the AUTOSAR 4.2.2, the program flow monitoring shall be fulfilled by the upper layer (WDG manager) where automatic password sequencing mechanism could be used. Even though the password is configured by the user at runtime, the password to unlock the watchdog timer may be changed by the user or

by another driver. Hence, the password is always recalculated before SCU_WDTCPUxCON0 (x = core Id) is accessed.

1.1.5.7 WDG driver trigger

The timeout passed to the `Wdg_SetTriggerCondition` API must be greater than the mode (SLOW/FAST) specific refresh time (configured by the user), to ensure that the WDT is serviced until the trigger counter value becomes zero. (If the mode is OFF, the WDT will not be serviced). If the timeout is less than the refresh time then the trigger counter will be zero and WDT will not be serviced and this will lead to WDG alarm when the timer overflows.

1.1.5.8 WDG driver code generator dependency

The WDG driver is dependent on MCU, MCALLIB and Resource Manager for successful code generation. In case these modules are not added to the configuration project, an error is raised.

1.1.5.9 WDG driver variation point

The Post-Build-Variant value of the parameters: `WdgInitialTimeout, WdgCPUInitialTimeout, WdgMaxTimeout and WdgCPUMaxTimeout` is set as TRUE since the values may vary across different variants configured across multiple cores.

1.1.5.10 STM timer plausibility check

If the WDG driver uses the STM timer for time-based triggers, the STM timer will generate interrupts to service the WDT as per the configured mode-specific refresh time (WdgFastRefreshTime and WdgSlowRefreshTime). The user shall check the plausibility of the STM timer and only if the plausibility check is successful, the `Wdg_17_Scu_Isr` API shall be invoked.

1.2 Assumptions of Use (AoU)

The AoU for the WDG driver are as follows.

The safety measures which need to be fulfilled by the integrator will be provided as Assumptions of Use (AoU) in order to enable the usage of these drivers in safety relevant applications up to the highest safety level captured in the release notes and to facilitate flexibility in customer's safety concepts.

- **Critical section used while accessing CON0 register**

User shall ensure that the core-specific interrupts are disabled in the following critical sections: -SchM_Enter_Wdg_17_Scu_CpuEndInit and -SchM_Enter_McalLib_CpuEndInit.
[cover parentID WDG={51D8C43C-2314-4b4d-BF4C-263237F16EAB}]

- **GTM channel allocation to WDG timer**

The user shall ensure the following while allocating GTM channels to a particular core-specific WDG timer. The GTM channels sharing the same interrupt node shall be allocated to the same core in the IRQ driver. For example, if GTM channel x is allocated to the WDG driver and channel x+1 is allocated to the PWM driver, both the modules shall be in the same core since x and x+1 share interrupt node.
[cover parentID WDG={B578A86A-6687-4294-B6A7-A0B824FCE690}]

- **GTM channels sharing same interrupt nodes**

User shall ensure that timer channels of the GTM, which share the same interrupt nodes, shall not be configured for different CPU WDG timers.
[cover parentID WDG={49FDBDF3-10F6-4ba2-8F8E-06D92DC37EA4}]

- **WDG driver initialization with the correct configuration pointer**

The user shall ensure that the WDG initialization is done with the correct configuration pointer. The Wdg_17_Scu_InitCheck() API shall be called after the WDG driver initialization and before starting any WDG driver functionality.
[cover parentID WDG={46AC4880-0030-4f96-85AA-465741024800}]

- **WDG overflow**

The user shall ensure that the watchdog overflow shall trigger an SMU alarm and enter into a safe state. However, the overflow error flag will be checked before servicing the WDT and a safety error will be reported to the upper layer.

[cover parentID WDG={CBD32342-5082-4356-9803-61385D57337E}]

1.3 Reference information

1.3.1 Configuration interfaces

Supported configuration variant: Post-Build

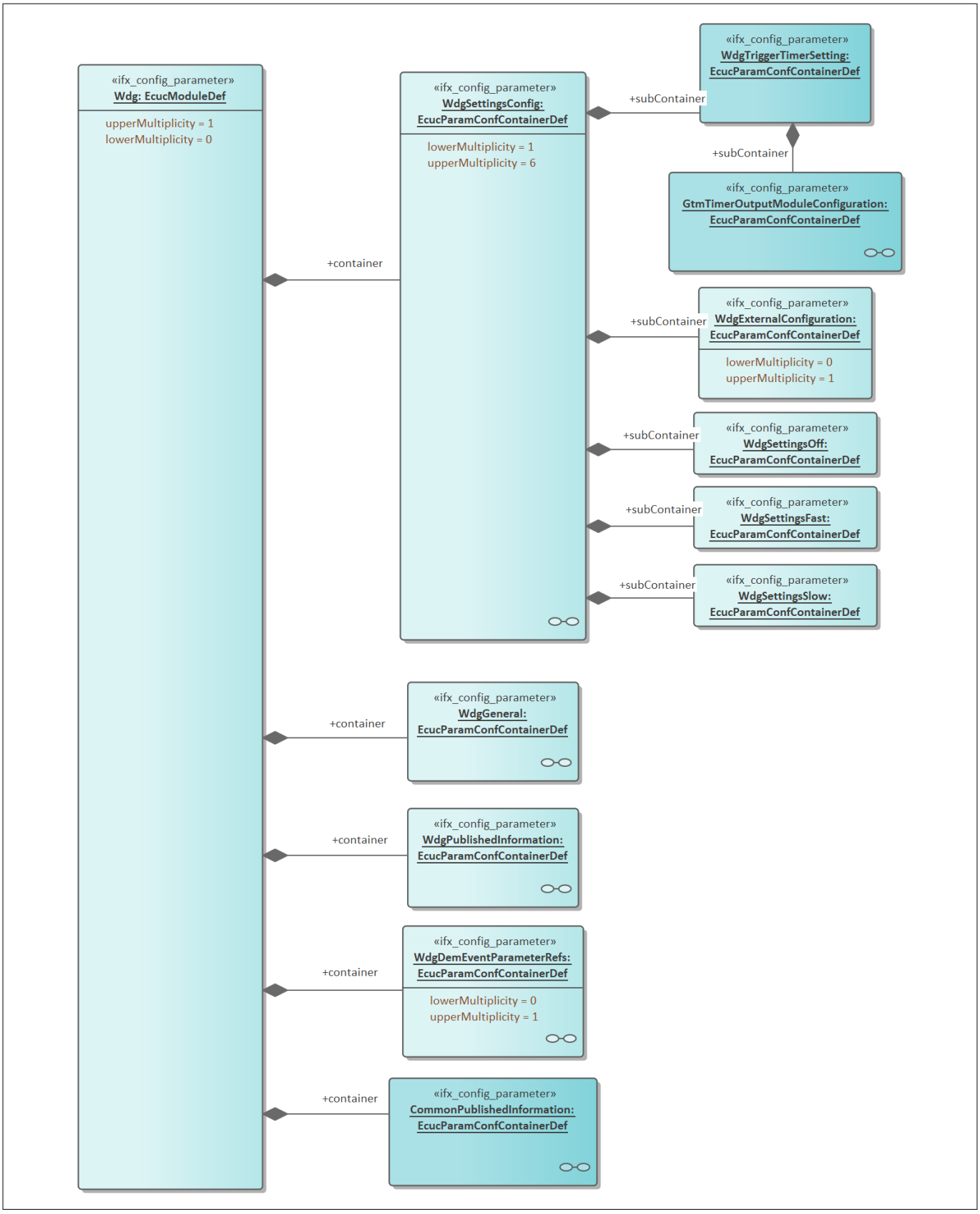

Figure 14 Container hierarchy along with their configuration parameters

1.3.1.1 Container: GtmTimerConfiguration

This container contains the configuration elements for configuring GTM timer hardware.
Post-Build Variant Multiplicity: TRUE
Multiplicity Configuration Class: Pre-Compile

1.3.1.2 Container: WdgSettingsConfig

This is a List of containers of Configuration items for the different watchdog settings.

Note: All postbuild parameters are handled via this container.
Post-Build Variant Multiplicity: -
Multiplicity Configuration Class: -

1.3.1.3 Container: CommonPublishedInformation

Container for common published information. (Based on BSW General)
Post-Build Variant Multiplicity: -
Multiplicity Configuration Class: -

1.3.1.3.1 ArMajorVersion

Table 4 Specification for ArMajorVersion

Name	`ArMajorVersion`		
Description	This parameter provides the major version of the AUTOSAR Specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.2 ArMinorVersion

Table 5 Specification for ArMinorVersion

Name	`ArMinorVersion`		
Description	This parameter provides the minor version of the AUTOSAR Specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef

(table continues...)

Table 5 (continued) Specification for ArMinorVersion			
Range	0 - 255		
Default value	As per the selected Autosar version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.3 ArPatchVersion

Table 6 Specification for ArPatchVersion			
Name	`ArPatchVersion`		
Description	This parameter provides the patch version of the AUTOSAR Specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the selected Autosar version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.4 ModuleId

Table 7 Specification for ModuleId			
Name	`ModuleId`		
Description	This parameter provides the Module Id.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	102		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 7 (continued) Specification for ModuleId

Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.5 Release

Table 8 Specification for Release

Name	`Release`		
Description	Aurix2G derivative used for the implementation Default value will depend on the derivative chosen and will be read from the derivative .properties file.		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	Depends on the Derivative		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.6 SwMajorVersion

Table 9 Specification for SwMajorVersion

Name	`SwMajorVersion`		
Description	This parameter provides the major version of the Software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-

(table continues...)

Table 9 (continued) Specification for SwMajorVersion

Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.7 SwMinorVersion

Table 10 Specification for SwMinorVersion

Name	`SwMinorVersion`		
Description	This parameter provides the minor version of the Software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.8 SwPatchVersion

Table 11 Specification for SwPatchVersion

Name	`SwPatchVersion`		
Description	This parameter provides the patch version of the Software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the driver version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.9 VendorApiInfix

Table 12 Specification for VendorApiInfix

Name	`VendorApiInfix`		
Description	In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name. This parameter is used to specify the vendor specific name. For example, assuming that the VendorId of the implementer is 17 and the implementer chooses VendorApiInfix as Scu then the API name Wdg_SetMode defined in the SWS will translate to Wdg_17_Scu_SetMode. This parameter is mandatory for all modules with upper multiplicity greater than 1. It will not be used for modules with upper multiplicity equal to 1		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	Scu		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3.10 VendorId

Table 13 Specification for VendorId

Name	`VendorId`		
Description	This parameter provides the Vendor Id		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4 Container: GtmTimerOutputModuleConfiguration

This container contains the elements for configuring GTM timer hardware (TOM/ATOM). The settings here are used to configure the timing needs for GTM timer. Availability of this container depends on the underlying derivative.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.4.1 GtmTimerClockSelect

Table 14 Specification for GtmTimerClockSelect

Name	`GtmTimerClockSelect`		
Description	This parameter decides the Clock Source for TOM/ATOM timer. Values: If GtmTimerUsed is TOM module. GTM_FIXED_CLOCK_0: Fixed Clock 0 is selected. GTM_FIXED_CLOCK_1: Fixed Clock 1 is selected. GTM_FIXED_CLOCK_2: Fixed Clock 2 is selected. GTM_FIXED_CLOCK_3: Fixed Clock 3 is selected. GTM_FIXED_CLOCK_4: Fixed Clock 4 is selected. If GtmTimerUsed is ATOM module. GTM_CONFIGURABLE_CLOCK_0: Configurable Clock0 is selected. GTM_CONFIGURABLE_CLOCK_1: Configurable Clock1 is selected. GTM_CONFIGURABLE_CLOCK_2: Configurable Clock2 is selected. GTM_CONFIGURABLE_CLOCK_3: Configurable Clock3 is selected. GTM_CONFIGURABLE_CLOCK_4: Configurable Clock4 is selected. GTM_CONFIGURABLE_CLOCK_5: Configurable Clock5 is selected. GTM_CONFIGURABLE_CLOCK_6: Configurable Clock6 is selected. GTM_CONFIGURABLE_CLOCK_7: Configurable Clock7 is selected. Minimum CLOCK ID is selected as the default value.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 14 (continued) Specification for GtmTimerClockSelect

Range	GTM_CONFIGURABLE_CLOCK_0: Configurable Clock 0 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_1: Configurable Clock 1 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_2: Configurable Clock 2 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_3: Configurable Clock 3 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_4: Configurable Clock 4 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_5: Configurable Clock 5 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_6: Configurable Clock 6 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_7: Configurable Clock 7 will be supplied to the Channel GTM_FIXED_CLOCK_0: Fixed Clock 0 will be supplied to the Channel GTM_FIXED_CLOCK_1: Fixed Clock 1 will be supplied to the Channel GTM_FIXED_CLOCK_2: Fixed Clock 2 will be supplied to the Channel GTM_FIXED_CLOCK_3: Fixed Clock 3 will be supplied to the Channel GTM_FIXED_CLOCK_4: Fixed Clock 4 will be supplied to the Channel		
Default value	GTM_FIXED_CLOCK_0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	GtmTimerUsed		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.2 GtmTimerUsed

Table 15 Specification for GtmTimerUsed

Name	`GtmTimerUsed`		
Description	The TOM/ATOM Channel resource assigned to the watchdog timer. This parameter is list of all the GTM timer channels (TOM/ATOM) used by WDG Driver. Referred timer channel in MCU should have TomChannelUsage/ AtomChannelUsage as USED_BY_WDG_DRIVER. *Note: If referred timer channel is not marked as "GTM_TOM_CHANNEL_USED_BY_WDG/ GTM_ATOM_CHANNEL_USED_BY_WDG" via the parameters McuGtmTomChannelAllocationConf / McuGtmAtomChannelAllocationConf in MCU module an error message will be raised.* *The TOM/ATOM channel should be unreserved in MCU, if WdgTriggerTimerSelection is changed to STM_TIMER or in MCU module an error message will be raised.* *Since the name of the dependent container is user configurable, the default value is kept as NULL.*		
Multiplicity	1..1	**Type**	EcucReferenceDef

(table continues...)

Table 15 (continued) Specification for GtmTimerUsed

Range	Reference to Node: McuGtmTomChannelAllocationConf, McuGtmAtomChannelAllocationConf		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5 Container: Wdg

Configuration of the Wdg module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.6 Container: WdgDemEventParameterRefs

This container lists down all the Production error event configuration parameters for WDG driver.

If this container does not exist in the configuration, then production error handling is not performed.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.6.1 WDG_E_DISABLE_REJECTED

Table 16 Specification for WDG_E_DISABLE_REJECTED

Name	`WDG_E_DISABLE_REJECTED`
Description	The existence of this parameter decides if the WDG module would raise this particular production error. The user has to point to the right DEM event parameter for this fault. The short name of the pointed Production error parameter will be used as the symbol that will be passed as the first parameter to Mcal_Wrapper_Dem_ReportErrorStatus() API for AUTOSAR 4.2.2 and Mcal_Wrapper_Dem_SetEventStatus() API for AUTOSAR 4.4.0. The extended production error WDG_17_SCU_E_DISABLE_REJECTED will be reported with FAILED when disabling of the watchdog mode failed and will be reported with PASSED when disabling of the watchdog mode not failed. The minimum value of multiplicity depends on the WdgSafetyEnable. (if WdgSafetyEnable is enabled then multiplicity is 1..1 and if disabled then multiplicity is 0..1)

(table continues...)

Table 16 (continued) Specification for WDG_E_DISABLE_REJECTED

Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.2 WDG_E_MODE_FAILED

Table 17 Specification for WDG_E_MODE_FAILED

Name	`WDG_E_MODE_FAILED`		
Description	The existence of this parameter decides if the WDG module would raise this particular production error. The user has to point to the right DEM event parameter for this fault.The short name of the pointed Production error parameter will be used as the symbol that will be passed as the first parameter to Mcal_Wrapper_Dem_ReportErrorStatus() API for AUTOSAR 4.2.2 and Mcal_Wrapper_Dem_SetEventStatus() API for AUTOSAR 4.4.0. Reference to DemEventParameter. The extended production error WDG_17_SCU_E_MODE_FAILED will be reported with FAILED when setting of the watchdog mode failed and will be reported with PASSED when setting of the watchdog mode not failed. The minimum value of multiplicity depends on the WdgSafetyEnable. (if WdgSafetyEnable is enabled then multiplicity is 1..1 and if disabled then multiplicity is 0..1)		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	LOCAL

(table continues...)

Table 17 (continued) Specification for WDG_E_MODE_FAILED

Dependency	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.7 Container: WdgExternalConfiguration

This container contains the configuration items for external watchdog hardware.

This container and its parameters are not used. The default container and parameter will be retained from AUTOSAR and disabled.

(This configuration is not provided as the external watchdog is not supported by Infineon.)

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Pre-Compile

1.3.1.8 Container: WdgGeneral

All general parameters of the watchdog driver are collected here.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.8.1 WdgDevErrorDetect

Table 18 Specification for WdgDevErrorDetect

Name	`WdgDevErrorDetect`		
Description	Switches the default error detection and notification ON or OFF. True : enabled (ON). False: disabled (OFF).		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.2 WdgDisableAllowed

Table 19 Specification for WdgDisableAllowed

Name	`WdgDisableAllowed`		
Description	This parameter is disabled. Instead, WdgCPUDisableAllowed is provided to enable/disable CPU specific WDG for every CPU WDG configured. Compile switch to allow / forbid disabling the watchdog driver during runtime. True: Disabling the WDG driver at runtime is allowed. False: Disabling the WDG driver at runtime is not allowed.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.3 WdgEcucPartitionRef

Table 20 Specification for WdgEcucPartitionRef

Name	`WdgEcucPartitionRef`		
Description	Parameter support is added only for AUTOSAR schema compliance, this parameter is not used in code generation logic, hence this parameter is made editable false.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: EcucPartition		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	-		

(table continues...)

Table 20 (continued) Specification for WdgEcucPartitionRef

Autosar Version	Applicable for Autosar version 4.4.0.

1.3.1.8.4 WdgIndex

Table 21 Specification for WdgIndex

Name	`WdgIndex`		
Description	Wdg driver ID which can be used by Wdg Interface as a reference. The default value can be modified by the user as per the needs of the application.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.5 WdgInitApiMode

Table 22 Specification for WdgInitApiMode

Name	`WdgInitApiMode`		
Description	MCAL_SUPERVISOR: Wdg Init API will run in Supervisor mode. MCAL_USER1: Wdg Init API will run in USER1 mode. Since WDG driver accesses the SFRs, it is more efficient to operate the WDG driver in supervisor mode. Hence, the default mode of operation is supervisor.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	WDG_MCAL_SUPERVISOR: APIs runs in Supervisor mode. WDG_MCAL_USER1: APIs runs in USER1 mode.		
Default value	WDG_MCAL_SUPERVISOR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-

(table continues...)

Table 22 (continued) Specification for WdgInitApiMode

Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.6 WdgInitCheckApi

Table 23 Specification for WdgInitCheckApi

Name	`WdgInitCheckApi`		
Description	Switches the InitCheck API ON or OFF. True : enabled (ON). False: disabled (OFF). The detection of safety related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.7 WdgInitialTimeout

Table 24 Specification for WdgInitialTimeout

Name	`WdgInitialTimeout`		
Description	This parameter is disabled. Instead, WdgCPUInitialTimeout is provided as initial timeout for every CPU WDG configured. This initial timeout (sec) for the trigger condition to be initialized during Init function. It will not be larger than WdgMaxTimeout. The default value is retained same as the WdgCPUInitialTimeout parameter. However, this parameter is disabled and not used by the WDG driver for any of its functionality.		
Multiplicity	1..1	**Type**	EcucFloatParamDef

(table continues...)

Table 24 (continued) Specification for WdgInitialTimeout

Range	0.001s - 65.535s		
Default value	5s		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	WdgMaxTimeout		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.8 WdgMaxTimeout

Table 25 Specification for WdgMaxTimeout

Name	`WdgMaxTimeout`		
Description	This parameter is disabled. Instead, WdgCPUMaxTimeout is provided as maximum CPU specific WDG timeout for every CPU WDG configured.This is the maximum window period of the Watchdog timer. The default value is retained same as the WdgCPUMaxTimeout parameter. However, this parameter is disabled and not used by the WDG driver for any of its functionality.		
Multiplicity	1..1	**Type**	EcucFloatParamDef
Range	0.001s - 65.535s		
Default value	32s		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.9 WdgRunArea

Table 26 Specification for WdgRunArea

Name	`WdgRunArea`

(table continues...)

Table 26 (continued) Specification for WdgRunArea

Description	This feature is not provided by Infineon and is disabled . The visibility is false for this parameter. Represents the WDG driver execution area is either from ROM(Flash) or RAM as required with the particular microcontroller. The default value is ROM. However, this parameter is disabled and not used by the WDG driver for any of its functionality.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	RAM: This feature is not provided by Infineon. This visibility is false. WDG driver to be executed out of RAM area ROM: This feature is not provided by Infineon. This parameter is disabled. WDG driver to be executed out of ROM area.		
Default value	ROM		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.10 WdgRuntimeApiMode

Table 27 Specification for WdgRuntimeApiMode

Name	`WdgRuntimeApiMode`		
Description	MCAL_SUPERVISOR: Wdg runtime APIs will run in Supervisor mode. MCAL_USER1: Wdg runtime APIs will run in USER1 mode. Since WDG driver accesses the SFRs, it is more efficient to operate the WDG driver in supervisor mode. Hence, the default mode of operation is supervisor.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	WDG_MCAL_SUPERVISOR: APIs runs in Supervisor mode. WDG_MCAL_USER1: APIs runs in USER1 mode.		
Default value	WDG_MCAL_SUPERVISOR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 27 (continued) Specification for WdgRuntimeApiMode

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.11 WdgSafetyEnable

Table 28 Specification for WdgSafetyEnable

Name	`WdgSafetyEnable`		
Description	Switch to enable reporting of safety DETs. True : enabled (ON). False: disabled (OFF). *Note: when this switch is enabled AUTOSAR DETs are enabled by default.* The detection of safety related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	WDG_E_MODE_FAILED, WDG_E_DISABLE_REJECTED, WdgDemEventParameterRefs		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.12 WdgTriggerLocation

Table 29 Specification for WdgTriggerLocation

Name	`WdgTriggerLocation`

(table continues...)

Table 29 (continued) Specification for WdgTriggerLocation

Description	This feature is not provided by Infineon and is disabled . The visibility is false for this parameter. Location (memory address) of the watchdog trigger routine. The default value is NULL_PTR. However, this parameter is disabled and not used by the WDG driver for any of its functionality.		
Multiplicity	1..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.13 WdgTriggerTimerSelection

Table 30 Specification for WdgTriggerTimerSelection

Name	`WdgTriggerTimerSelection`		
Description	Hardware timer used to service WDG during window period. It is common for all WDTs. GTM_TIMER (Used to service WDTs.) STM_TIMER (Used to service WDTs,) The default value can be modified by the user as per the needs of the application. *Note: GTM_TIMER will not be available for selection in devices without GTM.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	GTM_TIMER: GTM timer is used to service WDTs, in devices with GTM. STM_TIMER: STM timer is used to service WDTs.		
Default value	STM_TIMER		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL

(table continues...)

Table 30 (continued) Specification for WdgTriggerTimerSelection

Dependency	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.8.14 WdgVersionInfoApi

Table 31 Specification for WdgVersionInfoApi

Name	`WdgVersionInfoApi`		
Description	Compile switch to enable / disable the version information API. True : API enabled False: API disabled The optional APIs are disabled by default to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9 Container: WdgPublishedInformation

Container holding all WDG specific published information parameters
Post-Build Variant Multiplicity: -
Multiplicity Configuration Class: -

1.3.1.9.1 WdgTriggerMode

Table 32 Specification for WdgTriggerMode

Name	`WdgTriggerMode`
Description	Watchdog trigger mode (toggle/window/both) The default value is WDG_WINDOW since this the trigger mode supported in the WDG driver. This value is not editable.

(table continues...)

Table 32 (continued) Specification for WdgTriggerMode

Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	WDG_BOTH: Watchdog trigger mode is both Window and Toggle. WDG_TOGGLE: Watchdog trigger mode is Toggle. WDG_WINDOW: Watchdog trigger mode is Window.		
Default value	WDG_WINDOW		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10 Container: WdgSettingsConfig

This is available under WDG main Container. This container configures exactly one watchdog timer per CPU.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.10.1 WdgCPUDisableAllowed

Table 33 Specification for WdgCPUDisableAllowed

Name	`WdgCPUDisableAllowed`		
Description	Compile switch to allow/forbid the Watchdog Driver change mode to WDGIF_OFF_MODE at runtime for the CPU WDT selected by container.This parameter is considered only if the dependent parameter WdgCPUDisableAllowed is True. This parameter is defined by Infineon. The optional features are disabled by default to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-

(table continues...)

Table 33 (continued) Specification for WdgCPUDisableAllowed

Origin	IFX	**Scope**	LOCAL
Dependency	WdgDefaultMode		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10.2 WdgCPUInitialPassowrd

Table 34 Specification for WdgCPUInitialPassowrd

Name	`WdgCPUInitialPassowrd`		
Description	Initial password for the password access of the CPU WDG. The default password after Application Reset is 00000000111100B . The default value is same as the application reset value but can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 16383		
Default value	60		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10.3 WdgCPUInitialTimeout

Table 35 Specification for WdgCPUInitialTimeout

Name	`WdgCPUInitialTimeout`		
Description	This is the initial window period that is active as soon Wdg_17_Scu_Init is called for the core. It is used to calculate the value of the trigger counter which is used to service the WDT just after initialization. This parameter is defined by Infineon. The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	**Type**	EcucFloatParamDef
Range	0.001s - 65.535s		
Default value	5s		

(table continues...)

Table 35 (continued) Specification for WdgCPUInitialTimeout

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	WdgCPUMaxTimeout		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10.4 WdgCPUMaxTimeout

Table 36 Specification for WdgCPUMaxTimeout

Name	`WdgCPUMaxTimeout`		
Description	This is the maximum window period for the core specific watchdog timer. It is considered as the maximum window period for this particular core. This parameter is defined by Infineon**.** The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	**Type**	EcucFloatParamDef
Range	0.001s - 65.535s		
Default value	32s		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10.5 WdgCoreId

Table 37 Specification for WdgCoreId

Name	`WdgCoreId`

(table continues...)

Table 37 (continued) Specification for WdgCoreId

Description	This parameter refers to CORE ID to which the watchdog settings belong. This parameter is defined by Infineon. CPU Watchdog available in each device [Range]: - TC39x = 0-5, - TC38x = 0-3, - TC37x, TC35x = 0-2, - TC36x, TC33x(ED\ADAS) = 0-1, - TC33x = 0. The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - depends on the device		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10.6 WdgDefaultMode

Table 38 Specification for WdgDefaultMode

Name	`WdgDefaultMode`		
Description	Default mode of WDG Driver initialization for WDTx. The Description on each Wdg Mode is given below. - Off-Mode: The watchdog hardware is disabled / shut down. - Slow-Mode: Triggering the watchdog hardware can be done with a long timeout period. This mode for example can be used during system startup / initialization phase. - Fast-Mode: Triggering the watchdog hardware has to be done with a short timeout period. This mode for example can be used during normal operations of the ECU. ImplementationType: WdgIf_ModeType The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 38 (continued) Specification for WdgDefaultMode

Range	WDGIF_FAST_MODE: Default watchdog mode is "fast" WDGIF_OFF_MODE: Default watchdog mode is "Off" Dependency : This mode can be set if the WdgDisableAllowed is True and WdgCPUDisableAllowed is True WDGIF_SLOW_MODE: Default watchdog mode is "slow"		
Default value	WDGIF_SLOW_MODE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	WdgDisableAllowed		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10.7 WdgSystemClockRef

Table 39 Specification for WdgSystemClockRef

Name	`WdgSystemClockRef`		
Description	This parameter refers to the system clock configured, It will refer to a valid McuClockSetting Configuration. *Note: WDG module will work correctly only for this system clock setting. No changes should be done to the system clock if WDG is to be kept functioning.* Since the name of the dependent container is user configurable, the default value is kept as NULL.		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: McuClockReferencePointConfig		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.11 Container: WdgSettingsFast

Hardware dependent settings for the WDG driver fast mode.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.11.1 WdgFastModeTimeoutValue

Table 40 Specification for WdgFastModeTimeoutValue

Name	`WdgFastModeTimeoutValue`		
Description	This parameter is the fast mode timeout in seconds for WDTx. It is used to calculate the reload value of the WDT in fast mode. For TC3xx , fast timeout maximum value can be calculated as follows: - fSPB = 100 MHz - Clock divider(fast) = 256 - WDT frequency = 100 000 000/256 = 390625 Hz - 1 tick = 1/390625 = 0.00000256s Minimum timeout = 0.00000256s Maximum timeout = 65536*0.00000256 = 0.16777216s (approximately) Following condition should be satisfied :- -- WdgFastModeTimeoutValue less than WdgSlowModeTimeoutValue -- WdgFastModeTimeoutValue greater than WdgFastRefreshTime The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	**Type**	EcucFloatParamDef
Range	0.001s - 0.16777216s		
Default value	0.01s		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	WdgFastRefreshTime, WdgSlowModeTimeoutValue		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12 Container: WdgSettingsOff

Hardware dependent settings for the WDG driver off mode.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.13 Container: WdgSettingsSlow

Hardware dependent settings for the WDG driver slow mode.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.13.1 WdgSlowModeTimeoutValue

Table 41 Specification for WdgSlowModeTimeoutValue

Name	`WdgSlowModeTimeoutValue`		
Description	Slow mode timeout in seconds for WDTx.It is used to calculate the reload value of the WDT in slow mode. For TC3xx , Slow timeout maximum value can be calculated as follows: - fSPB = 100 MHz - Clock divider(slow) = 16384 - WDT frequency = 100 000 000/16384 = 6103.515625 Hz - 1 tick = 1/6103.515625 = 0.00016384s Minimum timeout = 0.00016384s Maximum timeout = 65536*0.00016384 = 10.73741824s (approximately) Following condition should be satisfied :- -- WdgSlowModeTimeoutValue greater than WdgSlowRefreshTime The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	**Type**	EcucFloatParamDef
Range	0.001s - 10.73741824s		
Default value	0.02s		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	WdgSlowRefreshTime		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.14 Container: WdgTriggerTimerSetting

This container contains the configuration elements for configuring GTM/STM hardware. The settings here are used to configure the timing needs for WDG module

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.14.1 WdgFastRefreshTime

Table 42 Specification for WdgFastRefreshTime

Name	`WdgFastRefreshTime`

(table continues...)

Table 42 (continued) Specification for WdgFastRefreshTime

Description	Fast mode GTM/STM callback period in seconds for WDTx. It is used to calculate the value of the trigger counter used to service the WDT for fast mode. Following condition should be satisfied :- -- WdgFastRefreshTime less than WdgSlowRefreshTime. -- WdgFastRefreshTime less than WdgFastModeTimeoutValue The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	**Type**	EcucFloatParamDef
Range	0.001s - 0.16777216s		
Default value	0.007s		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.14.2 WdgSlowRefreshTime

Table 43 Specification for WdgSlowRefreshTime

Name	`WdgSlowRefreshTime`		
Description	Slow mode GTM/STM callback period in seconds for WDTx. It is used to calculate the value of the trigger counter used to service the WDT for slow mode. Following condition should be satisfied :- -- WdgSlowRefreshTime greater than WdgFastRefreshTime -- WdgSlowRefreshTime less than WdgSlowModeTimeoutValue -- For GTM, WdgSlowRefreshTime less than 10.73741824s, to ensure no overflow during the Wdg timeout calculation. -- For STM, WdgSlowRefreshTime less than 4.294s, to ensure no overflow during the Wdg timeout calculation. The default value can be modified by the user as per the needs of the application and it shall lie within the specified boundary value.		
Multiplicity	1..1	**Type**	EcucFloatParamDef
Range	0.001s - 10.73741824s		
Default value	0.015s		

(table continues...)

Table 43 (continued) Specification for WdgSlowRefreshTime

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.2 Functions - Type definitions

This section describes all the type definitions that are used by APIs.

For other generic datatypes refer AUTOSAR SWS.

1.3.2.1 Wdg_17_Scu_ConfigType

Table 44 Specification for Wdg_17_Scu_ConfigType

Syntax	`Wdg_17_Scu_ConfigType`	
Type	Structure	
File	`Wdg_17_Scu.h`	
Range	hardware dependent structure	The elements of the data structure are specific to the microcontroller.
Description	The elements of the data structure are specific to the microcontroller. This type definition will be detailed in Design based on data structure design. (Used for pointers to structures holding configuration data provided to the WDG module initialization routine for configuration of the module and watchdog hardware.)	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3 Functions - APIs

This section lists all the APIs for the WDG driver.

The names of the APIs are extended with VendorId and VendorApiInfix since the upper multiplicity of the WDG driver is greater than 1.

1.3.3.1 Wdg_17_Scu_Init

Table 45 Specification for `Wdg_17_Scu_Init` API

Syntax	`void Wdg_17_Scu_Init ( const Wdg_17_Scu_ConfigType * const ConfigPtr )`	
Service ID	0x00	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	This service is used to initialize each WDTx with configuration for the core from which this API invoked. To initialize WDTx, this API shall be called from CPUx (corresponding CPU core), with the pointer of corresponding core config data.	
Source	AUTOSAR	
Error handling	WDG_17_SCU_E_DISABLE_REJECTED, WDG_17_SCU_E_MODE_FAILED, WDG_17_SCU_E_PARAM_CONFIG, WDG_17_SCU_E_INIT_FAILED, WDG_17_SCU_E_BUSY	
Configuration dependencies	-	
User hints	- MCU should be in an initialized state before this function is called. Wdg_17_Scu_Init depends on MCU SPB clock. -The Wdg_17_Scu_Init API should be invoked by the user only from the initialization and should not be called anytime after the initialization procedure is completed.	

(table continues...)

Table 45 (continued) Specification for `Wdg_17_Scu_Init` API

SFR accessed	CPU_BIV(w), CPU_BTV(w), CPU_CORE_ID(r), CPU_DCON0(w), CPU_ISP(w), CPU_PCON0(w), CPU_PMA0(w), CPU_PMA1(w), CPU_SEGEN(w), GTM_ATOM_AGC_ENDIS_CTRL(w), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_OUTEN_CTRL(w), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_CM0(w), GTM_ATOM_CH_CM1(w), GTM_ATOM_CH_CN0(w), GTM_ATOM_CH_CTRL(w), GTM_ATOM_CH_IRQ_EN(w), GTM_ATOM_CH_IRQ_MODE(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_ATOM_CH_SR0(w), GTM_ATOM_CH_SR1(w), GTM_TOM_CH_CM0(w), GTM_TOM_CH_CM1(w), GTM_TOM_CH_CN0(w), GTM_TOM_CH_CTRL(w), GTM_TOM_CH_IRQ_EN(w), GTM_TOM_CH_IRQ_MODE(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_SR0(w), GTM_TOM_CH_SR1(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SCU_WDTCPU_CON0(rw), SCU_WDTCPU_CON1(rw), SCU_WDTCPU_SR(r), STM_CMCON(rw), STM_CMP(w), STM_ICR(rw), STM_ISCR(rw), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.2 Wdg_17_Scu_InitCheck

Table 46 Specification for `Wdg_17_Scu_InitCheck` API

Syntax	`Std_ReturnType  Wdg_17_Scu_InitCheck`
	`(`
	`  const Wdg_17_Scu_ConfigType * const ConfigPtr`
	`)`	

Service ID	0x06	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: if initialization was successful. E_NOT_OK: if initialization was unsuccessful.

(table continues...)

Table 46 (continued) Specification for `Wdg_17_Scu_InitCheck` API

Description	This API returns the status of the module initialization in context to the core from where the API is invoked. The API is available when safety is enabled or initialization check is explicitly enabled. This API does not check any running counter values of GTM, STM and global variables. *Note: Init check should be performed in the following sequence:* *- Call Wdg_17_Scu_Init from a core.* *- Call Wdg_17_Scu_InitCheck from the same core.*
Source	IFX
Error handling	-
Configuration dependencies	WdgInitCheckApi
User hints	None.
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_AGC_ENDIS_STAT(r), GTM_ATOM_CH_CM0(r), GTM_ATOM_CH_CM1(r), GTM_ATOM_CH_CN0(r), GTM_ATOM_CH_CTRL(r), GTM_ATOM_CH_IRQ_EN(r), GTM_ATOM_CH_IRQ_MODE(r), GTM_ATOM_CH_SR0(r), GTM_ATOM_CH_SR1(r), GTM_TOM_CH_CM0(r), GTM_TOM_CH_CM1(r), GTM_TOM_CH_CN0(r), GTM_TOM_CH_CTRL(r), GTM_TOM_CH_IRQ_EN(r), GTM_TOM_CH_IRQ_MODE(r), GTM_TOM_CH_SR0(r), GTM_TOM_CH_SR1(r), GTM_TOM_TGC0_ENDIS_STAT(r), GTM_TOM_TGC1_ENDIS_STAT(r), SCU_WDTCPU_CON0(r), SCU_WDTCPU_CON1(r), SCU_WDTCPU_SR(r), STM_CMCON(r), STM_ICR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.3 Wdg_17_Scu_SetMode

Table 47 Specification for `Wdg_17_Scu_SetMode` API

Syntax	`Std_ReturnType Wdg_17_Scu_SetMode ( const WdgIf_ModeType Mode )`	
Service ID	0x01	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	Mode	One of the following statically configured modes: 1. WDGIF_OFF_MODE 2. WDGIF_SLOW_MODE 3. WDGIF_FAST_MODE

(table continues...)

Table 47 (continued) Specification for `Wdg_17_Scu_SetMode` API

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: if mode change is successful E_NOT_OK: if mode change is unsucessful
Description	Switches the watchdog into the Mode Requested. To change mode for WDTx, the API shall be called from CPUx (corresponding CPU core). It services the watchdog for safety to ensure window does not expire in corner cases.	
Source	AUTOSAR	
Error handling	WDG_17_SCU_E_MODE_FAILED, WDG_17_SCU_E_DISABLE_REJECTED, WDG_17_SCU_E_PARAM_MODE, WDG_17_SCU_E_DRIVER_STATE	
Configuration dependencies	-	
User hints	WDG should be in an initialized state before this function is called.This function shall be called before watchdog timeout timer elapses.	
SFR accessed	CPU_BIV(w), CPU_BTV(w), CPU_CORE_ID(r), CPU_DCON0(w), CPU_ISP(w), CPU_PCON0(w), CPU_PMA0(w), CPU_PMA1(w), CPU_SEGEN(w), GTM_ATOM_AGC_ENDIS_CTRL(w), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_OUTEN_CTRL(w), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_CM0(w), GTM_ATOM_CH_CM1(w), GTM_ATOM_CH_CN0(w), GTM_ATOM_CH_CTRL(w), GTM_ATOM_CH_IRQ_EN(w), GTM_ATOM_CH_IRQ_MODE(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_ATOM_CH_SR0(w), GTM_ATOM_CH_SR1(w), GTM_TOM_CH_CM0(w), GTM_TOM_CH_CM1(w), GTM_TOM_CH_CN0(w), GTM_TOM_CH_CTRL(w), GTM_TOM_CH_IRQ_EN(w), GTM_TOM_CH_IRQ_MODE(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_SR0(w), GTM_TOM_CH_SR1(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SCU_WDTCPU_CON0(rw), SCU_WDTCPU_CON1(rw), SCU_WDTCPU_SR(r), STM_CMCON(rw), STM_CMP(w), STM_ICR(rw), STM_ISCR(rw), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.4 Wdg_17_Scu_SetTriggerCondition

Table 48 Specification for `Wdg_17_Scu_SetTriggerCondition` API

Syntax	`void Wdg_17_Scu_SetTriggerCondition ( const uint16 timeout )`	
Service ID	0x03	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	timeout	Timeout value (milliseconds) for setting the trigger counter.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Sets the timeout value for the trigger counter. The API also services the Hardware watchdog timer to ensure the watchdog window does not expire in corner cases. Note: This is a deviation from the Autosar SWS_Wdg_00166 (which states that the watchdog servicing should happen only inside an ISR).	
Source	AUTOSAR	
Error handling	WDG_17_SCU_E_PARAM_TIMEOUT, WDG_17_SCU_E_DRIVER_STATE	
Configuration dependencies	-	
User hints	WDG should be in an initialized state before this function is called. This function shall be called before watchdog timeout timer elapses. To set the timeout value for WDTx, the API must be called from CPUx (corresponding CPU core).	
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_CH_CN0(w), GTM_TOM_CH_CN0(w), SCU_WDTCPU_CON0(rw), STM_CMCON(rw), STM_CMP(w), STM_ICR(rw), STM_ISCR(rw), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.5 Wdg_17_Scu_GetVersionInfo

Table 49 Specification for `Wdg_17_Scu_GetVersionInfo` API

Syntax	`void Wdg_17_Scu_GetVersionInfo ( Std_VersionInfoType * const versioninfo )`	
Service ID	0x04	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to where to store the version information of this module.
Parameters (in - out)	-	-
Return	void	Pointer to where to store the version information of this module.
Description	Returns the version information of the module.	
Source	AUTOSAR	
Error handling	WDG_17_SCU_E_PARAM_POINTER	
Configuration dependencies	WdgVersionInfoApi	
User hints	None.	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.4 Notifications and Callbacks

This section lists all the notification and callbacks of the WDG driver.

1.3.4.1 Wdg_17_Scu_Isr

Table 50 Specification for `Wdg_17_Scu_Isr` API

Syntax	`void Wdg_17_Scu_Isr ( const uint32 LogicalChId, const uint32 IsrStatus )`
Service ID	0x05
Sync/Async	Synchronous

(table continues...)

Table 50 (continued) Specification for `Wdg_17_Scu_Isr` API

Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	LogicalChId IsrStatus	channel ID This parameter gives information about which comparator cased the interrupt.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	ISR for GTM/STM interrupts	
Source	IFX	
Error handling	WDG_17_SCU_E_DRIVER_STATE, WDG_17_SCU_E_MODE_DISABLED, WDG_17_SCU_E_INVALID_INTERRPT_SOURCE, WDG_17_SCU_E_ACCESS, WDG_17_SCU_E_OVERFLOW	
Configuration dependencies	-	
User hints	This is a callback function invoked from Mcu_17_TimerIp.c when GTM/STM interrupt has occurred to service the watchdog driver. Timing constraint: This callback function shall be called before the watchdog timer overflows and resets the system.	
SFR accessed	CPU_CORE_ID(r), GTM_ATOM_AGC_ENDIS_CTRL(w), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_OUTEN_CTRL(w), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_TOM_TGC0_ENDIS_CTRL(w), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_OUTEN_CTRL(w), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(w), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_OUTEN_CTRL(w), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_OSCCON(r), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SCU_WDTCPU_CON0(rw), SCU_WDTCPU_SR(r), STM_CMCON(rw), STM_CMP(w), STM_ICR(rw), STM_ISCR(rw), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.5 Scheduled functions

The WDG driver does not provide any scheduled functions.

1.3.6 Interrupt service routines

The WDG driver does not provide any interrupt handlers.

1.3.7 Callout

The driver does not support any callout functions.

1.3.8 Errors Handling

This section describes the various errors reported by the WDG driver.

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
WDG_17_SCU_E_ACCESS: If the Access Error Flag is Set while servicing an interrupt. (This bit is set when an illegal Password Access or Modify Access to register WDTxCON0 was attempted).	IFX	0xCB	SAFETY	0xCB	SAFETY
WDG_17_SCU_E_BUSY: If the state of the driver is BUSY when initialization is called asynchronously.	IFX	0xCC	SAFETY	0xCC	SAFETY
WDG_17_SCU_E_DISABLE_REJECTED: Initialization or watchdog mode switch failed because it would disable the watchdog though this is not allowed in this configuration . - FAILED: when disabling of the watchdog mode failed. - PASSED: when disabling of the watchdog mode not failed.	AUTOSAR	Assigned by DEM	Production error	Assigned by DEM	Production error
WDG_17_SCU_E_INIT_FAILED: Initialization called with NULL pointer.	AUTOSAR	0x15	DET_SAFETY	0x15	DET_SAFETY
WDG_17_SCU_E_INVALID_INTERRPT_SOURCE: WDG ISR is triggered if interrupt source is not CMP0 in case of GTM. *Note : This error check is not done in case of STM since it is expected that the interrupt when CMP0 is configured will always arrive at STMIR0 and CMP1 is configured it will always arrive at STMIR1.*	IFX	0xC9	SAFETY	0xC9	SAFETY
WDG_17_SCU_E_MODE_DISABLED: If ISR is triggered when WDG is disabled.	IFX	0xC8	SAFETY	0xC8	SAFETY

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
WDG_17_SCU_E_MODE_FAILED: Setting watchdog mode failed. - FAILED : Setting Watchdog mode failed. - PASSED : Setting Watchdog mode passed.	AUTOSAR	Assigned by DEM	Production error	Assigned by DEM	Production error
WDG_17_SCU_E_OVERFLOW: If the Overflow Error Flag is Set while servicing an interrupt. (This bit is set when the WDT overflows from FFFFH to 0000H.).	IFX	0xCA	SAFETY	0xCA	SAFETY
WDG_17_SCU_E_DRIVER_STATE: API service used in wrong context (for example, module not initialized).	AUTOSAR	0x10	DET_SAFETY	0x10	DET_SAFETY
WDG_17_SCU_E_PARAM_CONFIG: API service called with wrong / inconsistent parameter(s) . This DET will be reported if an API is called from a wrong core and if mode change was unsuccessful during initialization.	AUTOSAR	0x12	DET_SAFETY	0x12	DET_SAFETY
WDG_17_SCU_E_PARAM_MODE: API service called with wrong / inconsistent parameter(s).	AUTOSAR	0x11	DET_SAFETY	0x11	DET_SAFETY
WDG_17_SCU_E_PARAM_POINTER: API is called with wrong pointer value (for example, NULL pointer) other than initialization	AUTOSAR	0x14	DET_SAFETY	0x14	DET_SAFETY
WDG_17_SCU_E_PARAM_TIMEOUT: The passed timeout value is higher than the maximum timeout value.	AUTOSAR	0x13	DET_SAFETY	0x13	DET_SAFETY

1.3.9 Deviations and limitations

This section describes the deviations and limitations of the WDG driver.

1.3.9.1 Deviations

This section describes the deviations of the WDG driver.

1.3.9.1.1 Software specification deviations

This section describes the deviations from software specification.

Table 51 Known deviations

File structure	WdgIf.h is included instead of WdgIf_Types.h, which is deviation for Watchdog driver Header file structure requirement. WdgIf.h includes WdgIf_Types.h, hence no functional impact. *Note: Applicable for Autosar version 4.2.2 only.*
SWS_Wdg_00166	Besides the watchdog service ISR the watchdog is also serviced in Wdg_17_Scu_SetTriggerCondition() API, this is to ensure that the watchdog window does not expire in corner cases.
For all requirements related to Production errors	Reporting of Production error: Dem_ReportErrorStatus is done through Mcal_Wrapper_Dem_ReportErrorStatus interface for AUTOSAR 4.2.2 and Dem_SetEventStatus is done through Mcal_Wrapper_Dem_SetEventStatus interface for AUTOSAR 4.4.0. All production error related datatypes and modified interfaces inclusion shall be done via Mcal_Wrapper.h

1.3.9.1.2 AMDC Violations

The WDG driver does not have any AMDC violations.

1.3.9.1.3 VSMD Violations

This section describes the violations reported by the EB VSMD checker tool with respect to AUTOSAR.

Table 52 *Violations reported by VSMD checker tool for EB03*

Rule ID:	EB03

(table continues...)

Table 52 (continued) *Violations reported by VSMD checker tool for EB03*

VSMD Node(s):	/AURIX2G/EcucDefs/Wdg/WdgDemEventParameterRefs /AURIX2G/EcucDefs/Wdg/WdgDemEventParameterRefs/WDG_E_DISABLE_REJECTED /AURIX2G/EcucDefs/Wdg/WdgDemEventParameterRefs/WDG_E_MODE_FAILED /AURIX2G/EcucDefs/Wdg/WdgGeneral/WdgEcucPartitionRef /AURIX2G/EcucDefs/Wdg/WdgSettingsConfig/WdgExternalConfiguration /AURIX2G/EcucDefs/Wdg/WdgSettingsConfig/WdgExternalConfiguration/WdgExternalContainerRef
Description:	The StMD node has LOWER-MULTIPLICITY=0 and UPPER-MULTIPLICITY=1. The VSMD-node shall get the OPTIONAL-attribute instead of creating a list!
Additional Information:	

Table 53 *Violations reported by VSMD checker tool for EB09*

Rule ID:	EB09
VSMD Node(s):	/AURIX2G/EcucDefs/Wdg
Description:	EB specific rule to check consistency of parameter postBuildVariantUsed.
Additional Information:	

Table 54 *Violations reported by VSMD checker tool for EcucSws_1014*

Rule ID:	EcucSws_1014
VSMD Node(s):	/AURIX2G/EcucDefs/Wdg /AURIX2G/EcucDefs/Wdg/WdgGeneral /AURIX2G/EcucDefs/Wdg/WdgSettingsConfig
Description:	Additional vendor specific parameter definitions (using ParameterTypes), container definitions and references shall be added to the VSMD according to the alphabetical order.
Additional Information:	

Table 55 *Violations reported by VSMD checker tool for EcucSws_1035*

Rule ID:	EcucSws_1035

(table continues...)

Table 55 (continued) *Violations reported by VSMD checker tool for EcucSws_1035*

VSMD Node(s):	/AURIX2G/EcucDefs/Wdg /AURIX2G/EcucDefs/Wdg/WdgDemEventParameterRefs /AURIX2G/EcucDefs/Wdg/WdgDemEventParameterRefs/WDG_E_DISABLE_REJECTED /AURIX2G/EcucDefs/Wdg/WdgDemEventParameterRefs/WDG_E_MODE_FAILED /AURIX2G/EcucDefs/Wdg/WdgGeneral /AURIX2G/EcucDefs/Wdg/WdgGeneral/WdgDevErrorDetect /AURIX2G/EcucDefs/Wdg/WdgGeneral/WdgDisableAllowed /AURIX2G/EcucDefs/Wdg/WdgGeneral/WdgIndex /AURIX2G/EcucDefs/Wdg/WdgGeneral/WdgInitialTimeout /AURIX2G/EcucDefs/Wdg/WdgGeneral/WdgMaxTimeout /AURIX2G/EcucDefs/Wdg/WdgGeneral/WdgRunArea /AURIX2G/EcucDefs/Wdg/WdgGeneral/WdgTriggerLocation /AURIX2G/EcucDefs/Wdg/WdgGeneral/WdgVersionInfoApi /AURIX2G/EcucDefs/Wdg/WdgPublishedInformation /AURIX2G/EcucDefs/Wdg/WdgPublishedInformation/WdgTriggerMode /AURIX2G/EcucDefs/Wdg/WdgSettingsConfig /AURIX2G/EcucDefs/Wdg/WdgSettingsConfig/WdgDefaultMode /AURIX2G/EcucDefs/Wdg/WdgSettingsConfig/WdgExternalConfiguration /AURIX2G/EcucDefs/Wdg/WdgSettingsConfig/WdgExternalConfiguration/WdgExternalContainerRef /AURIX2G/EcucDefs/Wdg/WdgSettingsConfig/WdgSettingsFast /AURIX2G/EcucDefs/Wdg/WdgSettingsConfig/WdgSettingsOff /AURIX2G/EcucDefs/Wdg/WdgSettingsConfig/WdgSettingsSlow
Description:	For Containers, Parameters and References elements UUID must be unique (also between StMD and VSMD).
Additional Information:	

Table 56 ***Violations reported by VSMD checker tool for EcucSws_2101***

Rule ID:	EcucSws_2101
VSMD Node(s):	/AURIX2G/EcucDefs/Wdg/ POST_BUILD_VARIANT_USED
Description:	For each ConfigurationVariant supported by the ModuleDef, there must be one ImplementationConfigClass element. In VSMD, the ImplementationConfigClass is mandatory.
Additional Information:	

Table 57 ***Violations reported by VSMD checker tool for EcucSws_6003***

Rule ID:	EcucSws_6003
VSMD Node(s):	/AURIX2G/EcucDefs/Wdg
Description:	The SHORT-NAME of the AR-PACKAGEs of StMD and VSMD must be different to ensure a unique SHORT-NAME-path.
Additional Information:	

Table 58 ***Violations reported by VSMD checker tool for EcucSws_6008***

Rule ID:	EcucSws_6008
VSMD Node(s):	/AURIX2G/EcucDefs/Wdg/WdgSettingsConfig
Description:	The LOWER-MULTIPLICITY of an element in the VSMD must be bigger or equal and the UPPER-MULTIPLICITY must be equal or less than in the StMD
Additional Information:	Since the container multiplicity is dependent on the number of watchdog timers/number of cores for a device. The multiplicity will be from 0 to 5. (According to AUTOSAR it is 1)

Table 59 ***Violations reported by VSMD checker tool for TpsEcuc_06051_ASR41***

Rule ID:	TpsEcuc_06051_ASR41
VSMD Node(s):	/AURIX2G/EcucDefs/Wdg/ POST_BUILD_VARIANT_USED
Description:	The implementationConfigClass of an EcucParameterDef or EcucAbstractReferenceDef in VSMD shall be the same or higher (where PreCompile configuration class is considered to be the lowest and PostBuild the highest) as in StMD with respect to the selected subset defined by the actually implemented supportedConfigVariant.
Additional Information:	

1.3.9.2 Limitations

This section describes the limitations of the WDG driver.

Table 60 Known limitations

Reference	Limitation
WDT in suspend mode off	Wdg_17_Scu_InitCheck shall not be invoked if debugger is connected and suspend mode is off, since the reload value will be fixed with 0xFFFC (default timeout mode reload value) and will not match with the configured reload value.

Revision history

Table 61 Revision History

Date	Version	Description
2024-08-07	7.0	Released
2024-07-29	6.1	- 1.1.3.1 C file structure section Figure 2, inclusion link updated for Mcal_Wrapper.h and Det.h. - 1.3.1.10.7 WdgSystemClockRef section, McuClockReferencePoint dependency is removed. - 1.3.8 Errors Handling section, updated Error ID of error "WDG_17_SCU_E_INIT_FAILED" for AS422.
2023-07-04	6.0	Released
2023-06-14	5.1	- In section 1.1.4.1, DEM module has been removed and is replaced with Mcal_Wrapper module. - DEM has been modified to Production error where applicable. - Updated Figure 1, DEM Module is removed and Mcal_Wrapper Module is added. - The Assumption of Use description is updated to reflect the usage of Wdg driver upto the highest safety level captured in the release notes on the relevant applications - Updated section 1.1.3.1 to include Mcal_Wrapper.h and removed Dem.h. - Updated Description of section 1.3.1.6.1 and 1.3.1.6.2 - ASIL Level has been updated to Safety level and the description of the safety level is updated in Section 1.3.3 and 1.3.4 - Updated the Description of WdgDemEventParameterRefs container - Updated the section 1.3.9.1.1: Software Specification Deviations for Autosar requirements. Updated Reference from "SWS_Wdg_00105: Rte_Dem_Type.h" to "For all requirements related to Production errors". Updated Description of "SWS_Wdg_00105: Rte_Dem_Type.h" to add Mcal_Wrapper Module Information. - Deviation for SWS_Wdg_00166 Specification is added in section 1.3.9.1.1
2021-11-08	5.0	Released
2021-11-08	4.1	‘Mapping of hardware-software interfaces’ figure is corrected
2021-10-25	4.0	Released
2021-10-25	3.1	Config variant attribute table information is removed and added this information in 'Configuration interfaces' section.
2021-02-18	3.0	Released
2021-02-05	2.1	Deviation added for file structure AUTOSAR version 4.2.2 WdgIf.h inclusion.
2020-11-14	2.0	Released
2020-11-14	1.1	Updated the limitation of suspend mode
2020-07-31	1.0	Released

(table continues...)

Table 61 (continued) Revision History

2020-07-30	0.1	- Initial Version - Wdg_17_Scu driver chapter moved from MC-ISAR_TC3xx_UM_Basic to this document - LogicalChId of Wdg_17_Scu_Isr made as uint32 - System WDT replaced as Safety WDT in SCU primary hardware peripheral section - Updated the range, description and variant value of Configuration Parameters WdgFastModeTimeoutValue, WdgSlowModeTimeoutValue, WdgFastRefreshTime, WdgSlowRefreshTime, WdgInitialTimeout and WdgMaxTimeout

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2024-08-07

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-ocr1484806431059

Important notice

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.