



MCAL WDG Module Software Design Document

Document Version : 82
Document Owner : Texas Instruments
Document Status : Published
Last Approval Date : Mar 11, 2022

TI Confidential - NDA Restrictions
Copyright ©2022 Texas Instruments Incorporated










- [Revision History](#)
- [Terms and Abbreviations](#)
- [Introduction](#)
 - [Overview](#)
 - [Purpose and Scope](#)
 - [Module Overview](#)
 - [Requirements](#)
 - [Features Supported](#)
 - [Features Not Supported / NON-Compliance](#)
 - [Assumptions](#)
 - [Constraints](#)
 - [Hardware and SW platforms](#)
 - [Dependencies](#)
 - [SBL](#)
 - [ESM](#)
 - [Stakeholders](#)
 - [References](#)
- [Design Description](#)
 - [WDG Configuration Sequence](#)
 - [Directory Structure](#)
 - [Configurator](#)
 - [NON-Standard configurable parameters](#)
 - [Variant Support](#)
 - [Windowed Watchdog](#)
 - [Error Classification](#)
 - [Development Errors](#)




- Error Detection
 - Error notification (DET)
 - Runtime Errors
 - Error notification (DEM)
- Implementation Details
 - Data structures and resources
 - Global Variables
 - Dynamic Behavior - Control Flow Diagram
 - Dynamic Behavior - Data Flow Diagram
 - Application Parameters
 - Safety Diagnostic Features
- Low Level Definitions
 - Driver API's
 - Wdg_Init
 - Wdg_SetTriggerCondition
 - Wdg_GetVersionInfo
 - Wdg_Trigger
 - Wdg_RegisterReadback
- Performance Objectives
 - Resource Consumption Objectives
 - Critical timing and Performance
 - Watchdog SOC Reset Functionality
 - Watchdog Service Routine
- Testing Guidelines
- Template Revision History



1 Revision History

Version	Date	Author	Document Status	Comments
0.1	 30 Nov 2018	Sunil M S	Draft	First version
0.2	 06 Dec 2018	Sunil M S	In Review	Addressed Review Comments
0.3	 20 Apr 2021	Murthy N	In Review	Format change as per ASPICE
0.4	 13 Aug 2021	Nikki S	In Review	Adding Design IDs
0.5	 19 Aug 2021	Murthy N	In Approval	Fixing Review Comments
1.0	 07 Sep 2021	Nikki S	Published	Upload to Galileo
1.1	 24 Jan 2022	Nikki S	In Review	JACINTOREQ-1870



Version	Date	Author	Document Status	Comments
v82	 04 Mar 2022	Nikki S	Published	Review Comments Addressed

2 Terms and Abbreviations

Abbreviation /Term	Meaning / Explanation
CS	Chip select
DIO	Digital Input Output
ECU	Electric Control Unit
ICU	Interrupt Capture Unit
MCAL	Microcontroller Abstraction Layer
MCU	Microcontroller Unit
MMU	Memory Management Unit
OS	Operating System

Abbreviation /Term	Meaning / Explanation
PWM	Pulse Width Modulation
SFR	Special Function Register
RTE	Runtime environment
DIP	Digital Input/ Output
DET	Default Error Tracer
DEM	Diagnostic Event Manager – module to handle diagnostic relevant events.
WDG	Watchdog (module specific prefix)
UNINIT	Uninitialized (= not initialized)

3 Introduction

This document describes the design of the AUTOSAR BSW module WDG.

- Supported AUTOSAR Release : **4.3.1**
- Supported Configuration Variants : **Pre-Compile & Link Time**
- Vendor ID : **WDG_VENDOR_ID (44)**
- Module ID : **WDG_MODULE_ID (102)**

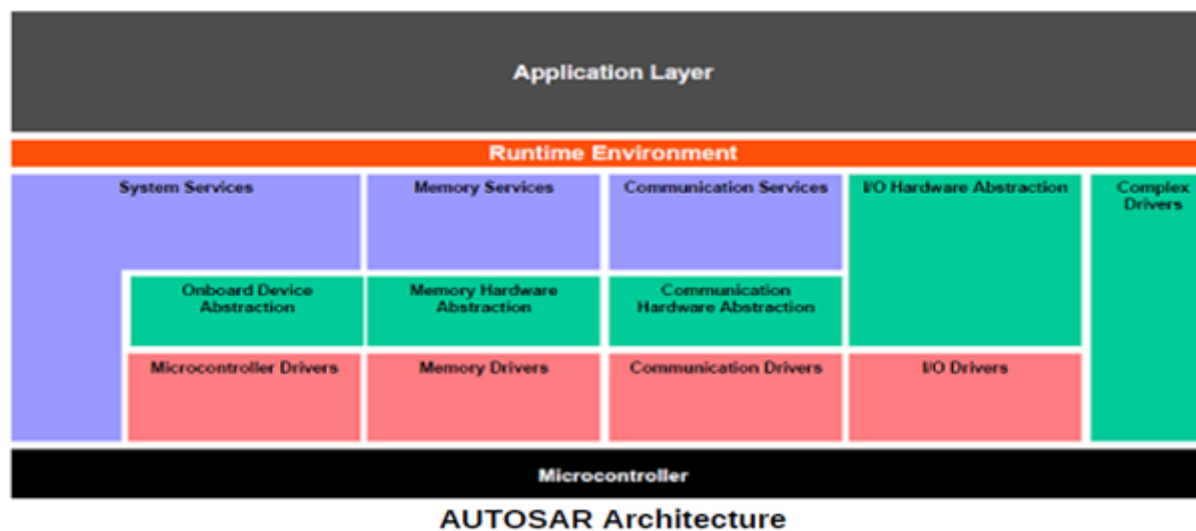
3.1 Overview

The figure below depicts the AUTOSAR layered architecture as 3 distinct layers,

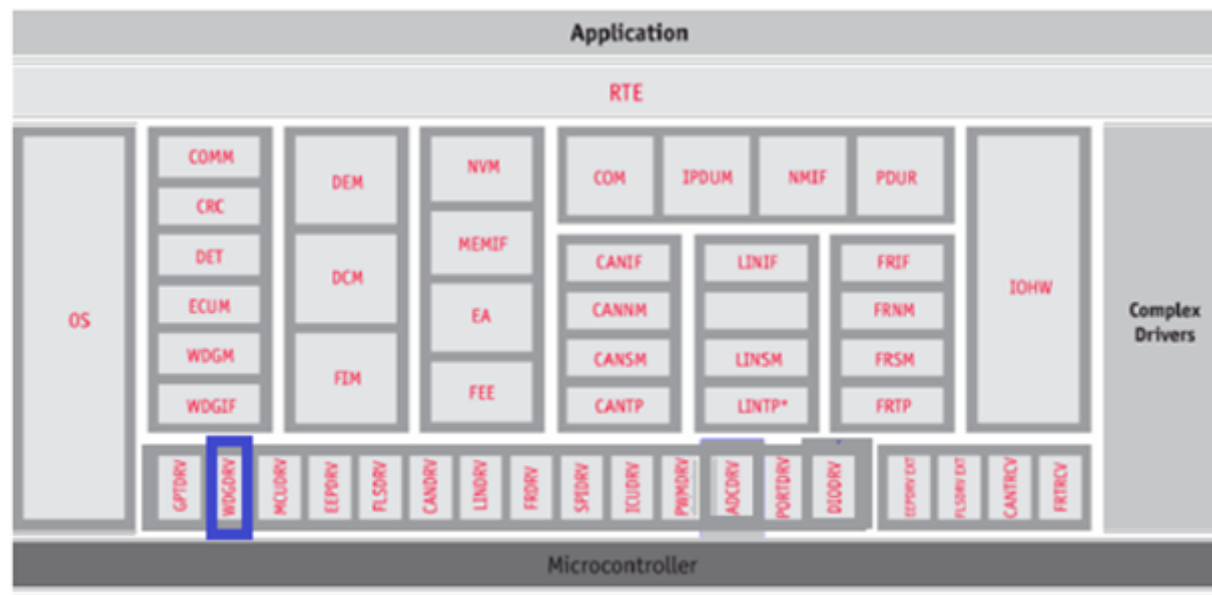
- Application
- Runtime Environment (RTE) and
- Basic Software (BSW).

The BSW is further divided into 4 layers:

- Services
- Electronic Control Unit Abstraction
- MicroController Abstraction (MCAL) and
- Complex Drivers.



MCAL is the lowest abstraction layer of the Basic Software. It contains software modules that interact with the Microcontroller and its internal peripherals directly. WDG driver is part of the Microcontroller Drivers (block, shown above). Below shows the position of the WDG driver in the AUTOSAR Architecture.



AUTOSAR Architecture – WDG MCAL

3.2 Purpose and Scope



This document specifies the implementation of MCAL driver for the module WDG. All requirements for implementing the driver are mapped in the document itself. Also, document holds the detailed information regarding the MACROs, structures and APIs for the implementation.

3.3 Module Overview

RTI (Real Time Interrupt using hardware IP "rti_10_rel.0.0.x") module supports three functional modes Counter mode, Capture mode & Windowed watchdog timer mode. Only Windowed watchdog timer mode is used to meet AUTOSAR WDG requirements.

The digital windowed watchdog generates an interrupt after a programmable period, if trigger is not serviced in the allowed timeframe. Windowed watchdog timer is such that key can be only written in the configured window programmed through software. The watchdog opens a configurable time window in which it must be serviced. Any attempt to service the watchdog outside this time window, or a failure to service the watchdog in this time window, will cause the watchdog to generate a NMI interrupt.

3.4 Requirements

The WGD driver shall implement as per requirements detailed in the Reference Documents. It is recommended to refer [Reference 1 - AUTOSAR 4.3.1](#) for clarification.

3.4.1 Features Supported

Below listed are some of the key features that are supported

- Initialization and configuration of WDG (configure window size, timeout value etc.).
- Setting default mode(FAST/SLOW).
- Service trigger via WDG Trigger API if called within the allowed time window.
- Supports all instances of RTI present in MCU domain.

- Supports additional configuration parameters, refer section (**Implementation specific parameters (computed)**) & (**WDG Register Readback**).

3.4.2 Features Not Supported / NON-Compliance

- **[NON-Compliance]** WDG_SetMode API is not supported. Due to hardware limitations, Mode and Timeout can't be modified if watchdog is already running i.e. only during initialization Mode and Timeout can be set.
- OFF-Mode is not supported.

3.5 Assumptions

Below listed are assumed to be valid for this design/implementation, exceptions and other deviations are listed for each explicitly. Care should be taken to ensure these assumptions are addressed.

1. The functional clock to the WDG module is expected to be enabled before calling any WDG module API.
2. The WDG driver as such doesn't perform any PRCM programming to get the functional clock.
3. The clock-source selection for WDG is not performed by the WDG driver, other entities such as SBL, MCAL module MCU shall perform the same.
4. Assumed that only one of the RTI instance is initiated per core at which driver is running.

Note that assumption 1 and 2 are specified by AUTOSAR WDG specification and 3 and 4 are device specific assumption.

3.6 Constraints

Some of the critical constraints of this design are listed below



In case where MCU module is not employed (supported) to configure the clock source for WDG module, refer Assumptions sub-item 3. Clock sources are listed in TRM, refer to SoC User Manual.

Also refer section features not supported. (Features Not Supported / NON-Compliance).

3.7 Hardware and SW platforms

Hardware Platforms

- Refer to specified SoC User Manual to check if ADC module is supported.

Software Platforms

- Bare metal

3.8 Dependencies

A WDG module for an internal (on-chip) watchdog accesses the microcontroller hardware directly and is located in the Microcontroller Abstraction layer.

A WDG module for an external watchdog uses other modules (e.g. SPI) to access the external watchdog device. Such a WDG module is located in the Onboard Device Abstraction Layer.

In addition to dependencies listed in section 5 of [Reference 1 - AUTOSAR 4.3.1](#), WDG driver shall depend on these modules to meet the required functionality. WDG uses RTI module present in the device to meet the required functionality.



This peripheral also requires 2 different clocks to be operational, namely ICLK and FCLK.

3.8.1 SBL

- **ICLK:** Is interface clock required for internal read/write registers of the peripheral. This is not expected to change and typically programmed by SBL, please refer the device specific manual for details and valid value.
- **FCLK:** Is functional clock, used to operate the module. As detailed in section (**Constraints**), other entity would require selecting the right clock source for the peripheral.

3.8.2 ESM

There is no direct reset signal generation from watchdog when it expires instead it generates an ESM interrupt. ESM module will signal ESM pin error and then external controller resets the SOC.

Design Identifier	Description
MCAL-5597	[WDG example application] RTI interrupt routed to ESM

3.9 Stakeholders

- Developers
- Test Engineers



- Customer Integrator

3.10 References

	Specification	Comment/Link
1	AUTOSAR 4.3.1	AUTOSAR Specification for WDG Driver.
2	BSW General Requirements / Coding guidelines	Autosar and Coding guidelines for the Mcal drivers.
3	Software Product Specification (SPS)	Product Functional Requirements.
4	Software Architecture	Mcal Software Architecture.



4 Design Description

The Digital Watchdog Timer(DWT) generates reset after a programmable period, if not serviced within that period. In DWT, time-out boundary is configurable. In DWWD, along with configurable time-out boundary, the start time boundary is also configurable. The DWWD can generate Reset or Interrupt, if not serviced within window (Open Window) defined by start time and time-out boundary. Also the DWWD can generate Reset or Interrupt if serviced outside Open Window (i.e within Closed Window).Generation of Reset or Interrupt depends on the DWWD Reaction configuration.

DWWD Down Counter Overview :

Upper 12 bit part of the down counter is configurable and remaining 13 bit are always 1.

Minimum possible time-out value is 2^{13} RTI clock cycles.

Maximum possible time-out value is 2^{25} RTI clock cycles.

Example :

The expiration time of the DWD Down Counter can be determined with following equation

$$t_{exp} = (RTI_DWDPRLD + 1) \times 2^{13} / RTI_FCLK$$

where $RTI_DWDPRLD(12 \text{ bit}) = 0 \dots 4095$ and RTI_FCLK is RTI functional frequency.

$RTI_FCLK : 32\text{kHz}$

12 bit preload value : $0x004$

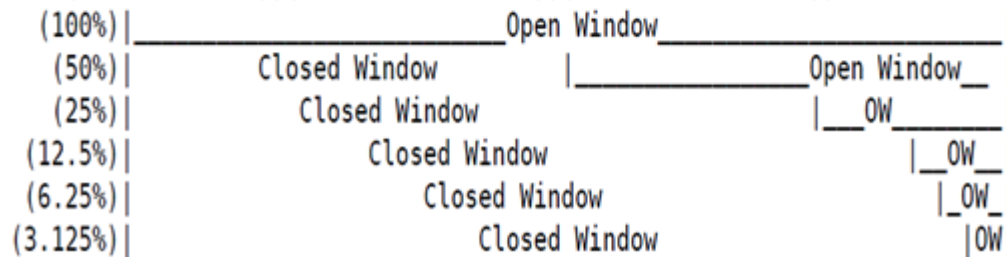
25 bit preload value : $0x0009FFF$

time-out value(in RTI clock cycles) : 40959.

time-out value(in seconds) : $(40959 + 1) / 32000 = 1.28 \text{ seconds}$.

DWWD Window Sizes Overview :

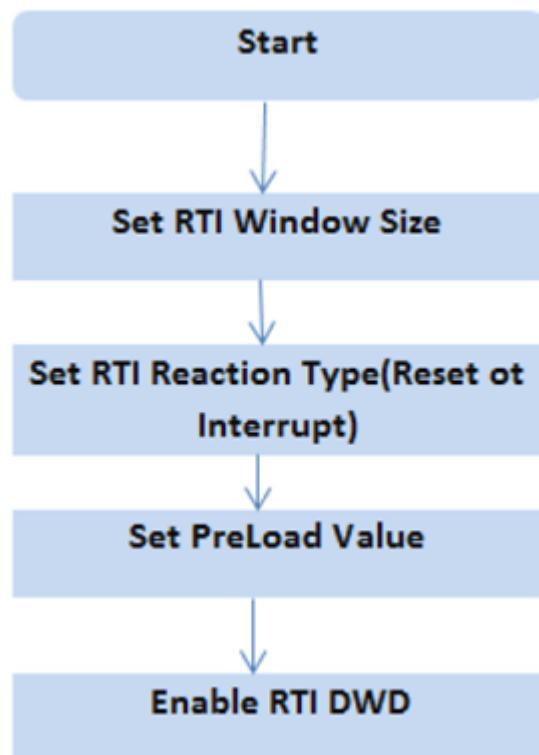
(timeout value), (timeout value - 1), (timeout value - 2), 3 2





4.1 WDG Configuration Sequence

The following diagram depicts the configuration sequence.



Watch Dog Timer Configuration Sequence

4.2 Directory Structure

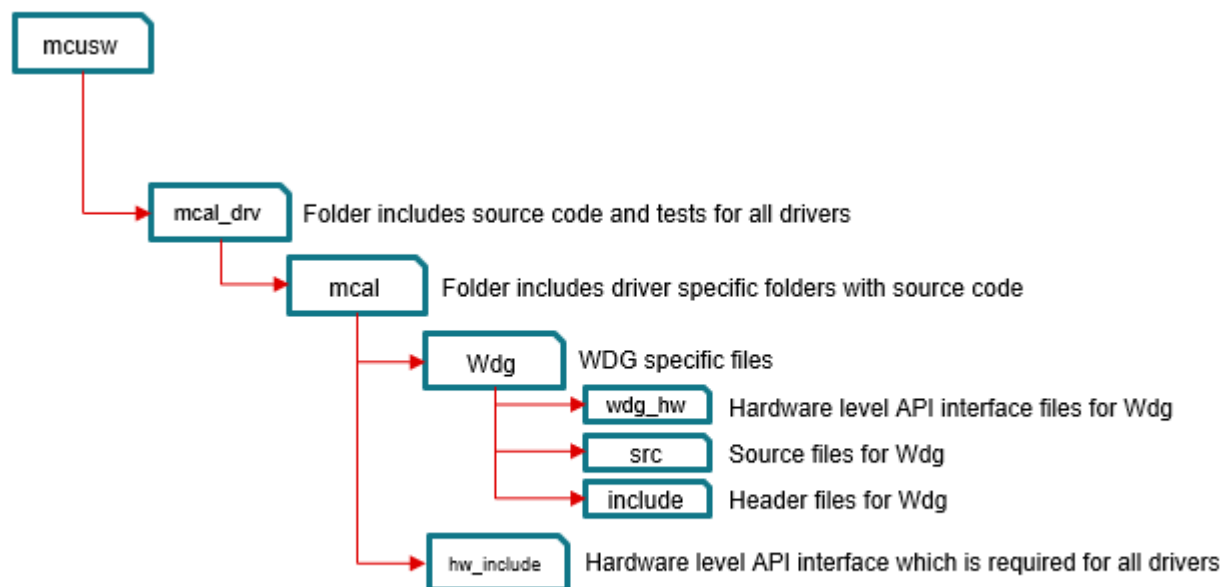
Design Identifier	Description
MCAL-5559	SWS_Wdg_00079 : Code File Structure

The directory structure is as depicted in figures below, the source files can be categorized under “Driver Implementation” and “Example Application”.

Driver Implemented by

All Mcal drivers share the same common directory structure as shown in the figure below (Common Driver Layout).

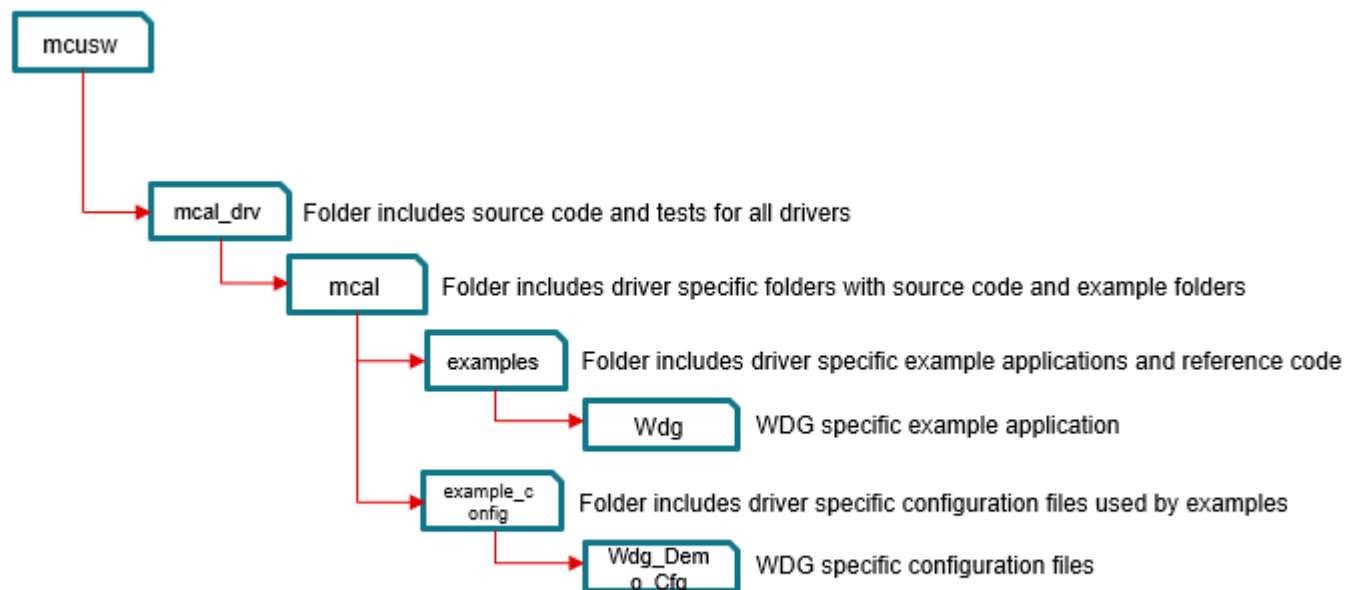
- Wdg.h and Wdg_Priv.h: Shall implement the interface provided by the AUTOSAR.
- Wdg_Priv.c and Wdg_Priv.h: Shall implement the driver functionality.
- Wdg_Dbg.h: Shall contain debug variable declarations of the driver.
- hw_rti.h: Shall include the SOC specific register definitions.





Example Application

- Wdg_Cfg.h and Wdg_Cfg.c: Shall implement the generated configuration for pre-compile variant.
- Wdg_Lcfg.c: Shall implement the generated configuration for link-time variant.
- Wdg_PBcfg.c: Shall implement the generated configuration for post-build variant.
- WdgApp.c and WdgApp.h: Shall implement the example application that demonstrates the use of the driver.



4.3 Configurator

The AUTOSAR WDG Driver Specification details mandatory parameters that shall be configurable via the configurator. Please refer section 10 of [Reference 1 - AUTOSAR 4.3.1](#).

Design Identifier	Description
MCAL-5627	WDG : ECUC : WdgReaction
MCAL-5626	ECUC_Wdg_00148 : WdgDemEventParameterRefs
MCAL-5617	ECUC_Wdg_00150 : WDG_E_DISABLE_REJECTED
MCAL-5615	ECUC_Wdg_00120 : WdgDefaultMode
MCAL-5613	ECUC_Wdg_00115 : WdgDevErrorDetect
MCAL-5611	WDG : ECUC : WdgDeviceVariant
MCAL-5610	ECUC_Wdg_00119 : WdgVersionInfoApi
MCAL-5609	WDG : ECUC : WdgInstanceld
MCAL-5608	ECUC_Wdg_00147 : WdgRunArea

Design Identifier	Description
MCAL-5606	WDG : ECUC : WdgRtiFrequency
MCAL-5605	ECUC_Wdg_00121 : WdgSettingsFast
MCAL-5602	ECUC_Wdg_00149 : WDG_E_MODE_FAILED
MCAL-5601	ECUC_Wdg_00113 : WdgExternalContainerRef
MCAL-5600	ECUC_Wdg_00074 : WdgPublishedInformation
MCAL-5589	ECUC_Wdg_00116 : WdgDisableAllowed
MCAL-5588	ECUC_Wdg_00122 : WdgSettingsOff
MCAL-5585	ECUC_Wdg_00114 : WdgGeneral
MCAL-5581	ECUC_Wdg_00117 : WdgIndex
MCAL-5577	WDG : ECUC : WdgWindowSize
MCAL-5574	ECUC_Wdg_00123 : WdgSettingsSlow

Design Identifier	Description
MCAL-5570	WDG : ECUC : WdgTimeoutValue
MCAL-5566	ECUC_Wdg_00118 : WdgTriggerLocation
MCAL-5565	ECUC_Wdg_00127 : WdgTriggerMode
MCAL-5562	ECUC_Wdg_00082 : WdgSettingsConfig
MCAL-5561	ECUC_Wdg_00112 : WdgExternalConfiguration
MCAL-5556	ECUC_Wdg_00130 : WdgInitialTimeout
MCAL-5552	ECUC_Wdg_00131 : WdgMaxTimeout
MCAL-5541	WDG : ECUC : WdgRegisterReadbackApi
MCAL-5599	SWS_Wdg_00086 : Static Configuration Params Check
MCAL-5560	SWS_Wdg_00168 : Wdg Code Run Area

4.3.1 NON-Standard configurable parameters

Following lists this design's specific configurable parameters

Parameter	Usage comment
WdgReaction	Watchdog reaction for timer expiration or incorrect service.0x5 = This is the default value. The windowed watchdog will cause a reset if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all. 0xA = The windowed watchdog will generate a non-maskable interrupt to the CPU if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all. Writing any other value will cause a system reset if the watchdog is serviced outside the time window.
WdgWindowSize	Digital Windowed Watchdog Window Size. Selecting 100% enables standard watchdog (not windowed). WWDSIZE: 0x00000050 = 50%, WWDSIZE: 0x00000500 = 25%, WWDSIZE:0x00005000 = 12.5%, WWDSIZE: 0x00050000 = 6.25%, WWDSIZE: 0x00500000 = 3.125%,WWDSIZE: Any other value = 3.125%.
WdgTimeoutValue	Watchdog timeout period in mill seconds. Watchdog generates a non-maskable interrupt to the CPU if the watchdog is serviced after this timeout period.
WdgDeviceVariant	Select SOC variant.
WdgInstanceld	Selects Watchdog HW instance id.

WdgRtiFrequency	RTI Clock Frequency (Hz) used to calculate preload value during init time.
WdgRegisterReadbackApi	Compile switch to enable / disable the Critical Registers Readback API.

4.3.2 Variant Support

The driver shall support all variants VARIANT-PRE-COMPILE, VARIANT-LINK-TIME and VARIANT-POST-BUILD.

Design Identifier	Description
MCAL-5616	SWS_Wdg_00157 : VARIANT-PRE-COMPILE
MCAL-5624	SWS_Wdg_00159 : VARIANT-POST-BUILD
MCAL-5578	SWS_Wdg_00158 : VARIANT-LINK-TIME

4.3.3 Windowed Watchdog

The digital windowed watchdog generates an interrupt after a programmable period, if trigger is not serviced in the allowed time frame.

Design Identifier	Description
MCAL-5614	SWS_Wdg_00035 : Windowed Watchdog : DET WDG_E_DRIVER_STATE
MCAL-5612	SWS_Wdg_00094 : Windowed Watchdog : Trigger Cycle For Activation Code
MCAL-5598	SWS_Wdg_00052 : Windowed Watchdog : Module State WDG_IDLE/WDG_BUSY
MCAL-5594	SWS_Wdg_00135 : Windowed Watchdog : Trigger Counter Zero
MCAL-5569	SWS_Wdg_00093 : Windowed Watchdog : Handling Activation Code
MCAL-5551	SWS_Wdg_00095 : Windowed Watchdog : Initial Activation Code
MCAL-5543	SWS_Wdg_00134 : Windowed Watchdog : Trigger Counter Non Zero

4.4 Error Classification

Errors are classified in two categories, development error and runtime / production error.



Design Identifier	Description
MCAL-5625	SWS_Wdg_00183 : Extended production error : WDG_E_DISABLE_REJECTED : Case PASS
MCAL-5620	SWS_Wdg_00179 : Extended production error : WDG_E_DISABLE_REJECTED
MCAL-5613	ECUC_Wdg_00115 : WdgDevErrorDetect
MCAL-5591	SWS_Wdg_00010 : Error Classification
MCAL-5580	SWS_Wdg_00178 : Extended production error : WDG_E_MODE_FAILED
MCAL-5563	SWS_Wdg_00181 : Extended production error : WDG_E_MODE_FAILED : Case PASS
MCAL-5548	SWS_Wdg_00182 : Extended production error : WDG_E_DISABLE_REJECTED : Case FAIL
MCAL-5544	SWS_Wdg_00180 : Extended production error : WDG_E_MODE_FAILED : Case FAIL
MCAL-5614	SWS_Wdg_00035 : Windowed Watchdog : DET WDG_E_DRIVER_STATE

4.4.1 Development Errors

Type of Error	Related Error code	Value (Hex)
API service used in wrong context (e.g. module not initialized).	WDG_E_DRIVER_STATE	0x10
API service called with wrong / inconsistent parameter(s).	WDG_E_PARAM_MODE	0x11
API service called with wrong / inconsistent parameter(s).	WDG_E_PARAM_CONFIG	0x12
The passed timeout value is higher than the maximum timeout value.	WDG_E_PARAM_TIMEOUT	0x13
API is called with wrong pointer value (e.g. NULL pointer).	WDG_E_PARAM_POINTER	0x14
Invalid configuration set selection.	WDG_E_INIT_FAILED	0x15

4.4.2 Error Detection

The detection of development errors is configurable (ON / OFF) at pre-compile time. The switch WDG Dev Error Detect will enable or disable the detection of all development errors.



4.4.3 Error notification (DET)

All detected development errors are reported via Det_ReportError service of the Development Error Tracer (DET).

4.4.4 Runtime Errors

The following runtime/production errors shall be detectable by WDG driver.

Type of Error	Related Error code	Value(Hex)
Setting a watchdog mode failed (during initialization or mode switch).	WDG_E_MODE_FAILED	Defined by integrator
Initialization or watchdog mode switch failed because it would disable the watchdog though this is not allowed in this configuration.	WDG_E_DISABLE_REJECTED	Defined By Integrator

4.4.5 Error notification (DEM)

All detected run time errors shall be reported via Dem_ReportErrorStatus() service of the Diagnostic Event Manager (DEM).

5 Implementation Details

5.1 Data structures and resources

5.1.1 MACROS, Data Types & Structures

The sections below list some of key data structures that shall be implemented and used in driver implementation.

Design Identifier	Description
MCAL-5587	SWS_Wdg_00171 : Wdg_ConfigType
MCAL-5557	SWS_Wdg_00105 : Imported Type
MCAL-5627	WDG : ECUC : WdgReaction
MCAL-5577	WDG : ECUC : WdgWindowSize
MCAL-5570	WDG : ECUC : WdgTimeoutValue
MCAL-5615	ECUC_Wdg_00120 : WdgDefaultMode
MCAL-5547	SWS_Wdg_00153 : Debugging: Internal Driver Timeout Counter

WDG_ModelInfoType

Used to define watchdog hardware specific parameters per instance and the values of these are expected to be populated by configurator

Type	Variable Name	comments
uint32	reaction	Reaction type: 0x5 - This value causes a reset if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all. 0xA - The windowed watchdog will generate a non-maskable interrupt to the CPU if the watchdog is serviced outside the time window defined by the configuration, or if the watchdog is not serviced at all.
uint32	windowSize	Digital Windowed Watchdog Window Size. Selecting 100% enables standard watchdog (not windowed). WWDSIZE: 0x00000050 = 50%, WWDSIZE: 0x00000500 = 25%, WWDSIZE: 0x00005000 = 12.5%, WWDSIZE: 0x00050000 = 6.25%, WWDSIZE: 0x00500000 = 3.125%, WWDSIZE: Any other value = 3.125%
uint32	timeOutVal	Watchdog timeout period in milli seconds. Watchdog generates a non-maskable interrupt or reset to the CPU if the watchdog is serviced after this timeout period.

Wdg_ConfigType



Refer section 8.2.1 of [Reference 1 - AUTOSAR 4.3.1](#). Used for pointers to structures holding configuration data provided to the WDG module initialization routine for configuration of the module and watchdog hardware.

Type	Variable Name	comments
WdgIf_ModeType	defaultMode	Default watchdog mode(WDGIF_FAST_MODE/WDGIF_SLOW_MODE).
Wdg_ModelInfoType	fastModeCfg	Contains fast mode hardware specific configuration. (Wdg_ModelInfoType).
Wdg_ModelInfoType	slowModeCfg	Contains slow mode hardware specific configuration. (Wdg_ModelInfoType).

Wdg_ConfigType_PC

Used to define Pre-Compile parameters populated by configurator.

Type	Variable Name	comments
uint16	Instance Id	Hardware Instance Id.
uint32	Initial Time Out	The initial timeout (seconds) for the trigger condition to be initialized during Init function. It shall be not larger than WDG Max Timeout.

Wdg_RegisterReadbackType

Name	Type	Range	comments
rtiDwdCtrl	uint32	0 to 0xFFFFFFFF	Digital Watchdog Control, To Enable DWD.
rtiWdStatus	uint32	0 to 0xFFFFFFFF	Digital Watchdog Status Register.
rtiWdKey	uint32	0 to 0xFFFFFFFF	Digital Watchdog Key Register.
rtiWwdRxnCtrl	uint32	0 to 0xFFFFFFFF	Digital Windowed Watchdog Reaction.
rtiWwdSizeCtrl	uint32	0 to 0xFFFFFFFF	Digital Windowed Watchdog Window Size.

5.1.2 Global Variables

This design expects that implementation require to use following global variables.

variable	Type	Description	Default value
----------	------	-------------	---------------

Wdg_DrvStatus	Wdg_StatusType	Initialization status of the driver is maintained.	WDG_UNINIT
Wdg_DrvObj	Wdg_DriverObjType	WDG driver object, local to the implementation and scope shall NOT be limited to Wdg.c.	Undefined

5.2 Dynamic Behavior - Control Flow Diagram

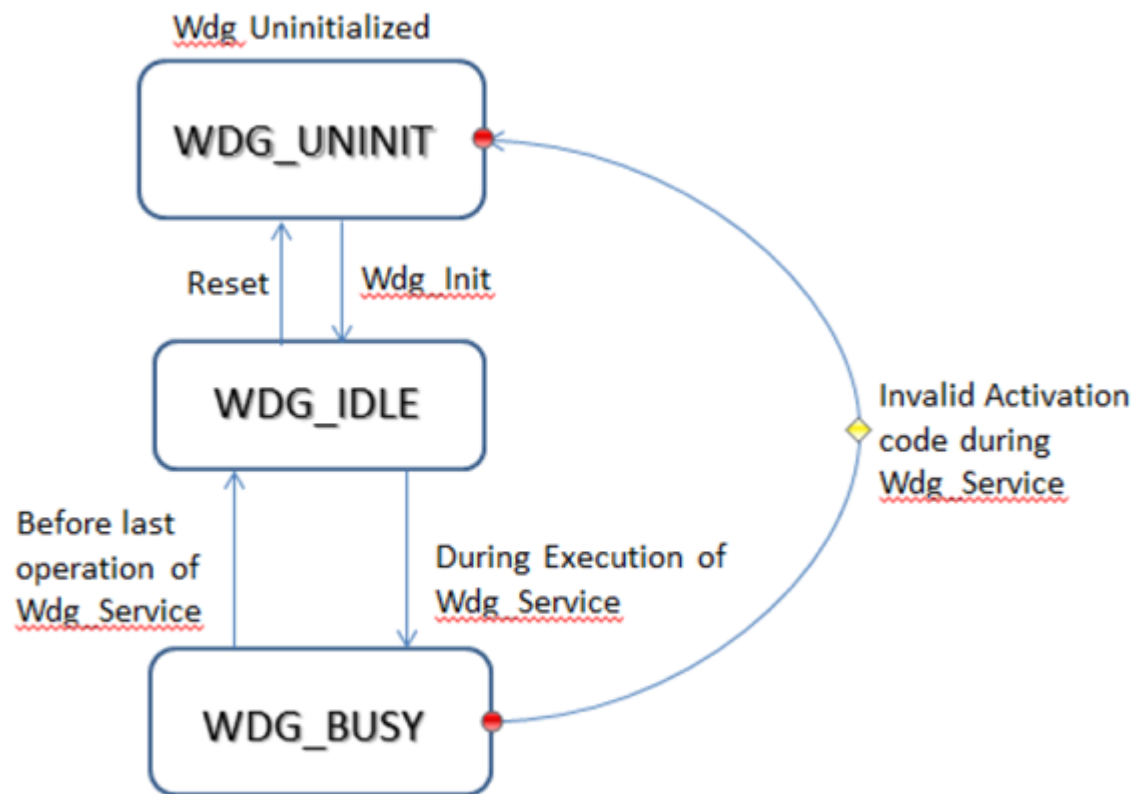
States

As detailed in specification of [Reference 1 - AUTOSAR 4.3.1](#), Driver will be in one of the following states.

WDG_UNINIT : Default state indicating a non-initialized module.

WDG_IDLE : Indicating initialization is successful.

WDG_BUSY : Indicating module is busy(during execution).



Driver States

Design Identifier	Description
MCAL-5622	SWS_Wdg_00152 : Debugging: Internal Driver State

5.3 Dynamic Behavior - Data Flow Diagram

Note Applicable

5.4 Application Parameters

Wdg_GetVersionInfo

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
versioninfo	Pointer to where to store the version information of this module (output parameter).	0 to 0xFFFFFFFF	-	-	N.A

Wdg_Init

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
-----------	-------------	-----------------------	---------------	---------------	---------

CfgPtr	Pointer to WDG driver configuration set(input parameter)..	0 to 0xFFFFFFFF	-	-	N.A
--------	--	-----------------	---	---	-----

Wdg_SetTriggerCondition

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
timeout	Timeout value (milliseconds) for setting the trigger counter (input parameter).	0 to 0xFFFF	ms	-	N.A

5.5 Safety Diagnostic Features

CLK5C - External Watchdog

This is for use of an external watchdog. Software necessary is defined by the External Watchdog selected by the System Integrator. This is provided by PMIC in TI solution. PMIC driver supports this.

Software Readback of Static Configuration Registers

Periodic readback of configuration registers can provide a diagnostic for inadvertent writes to these registers.

The WDG MCAL driver provides the API - **Wdg_RegisterReadback** to readback static and written configuration registers to implement this diagnostic feature.

6 Low Level Definitions

6.1 Driver API's

Refer to section 8.3 of the WDG AutoSar Specification as listed in [Reference 1 - AUTOSAR 4.3.1](#).

6.1.1 Wdg_Init

Initializes the module.

Design Identifier	Description
MCAL-5621	SWS_Wdg_00100 : Wdg Initialization Global Variables
MCAL-5618	SWS_Wdg_00173 : Wdg Initialization DEM WDG_E_MODE_FAILED
MCAL-5586	SWS_Wdg_00101 : Wdg Initialization Controller Registers
MCAL-5579	SWS_Wdg_00090 : Wdg Initialization DET WDG_E_PARAM_CONFIG
MCAL-5572	SWS_Wdg_00019 : Wdg Initialization DET WDG_UNINIT
MCAL-5550	SWS_Wdg_00025 : Wdg Initialization DEM WDG_E_DISABLE_REJECTED

Design Identifier	Description
MCAL-5542	SWS_Wdg_00001 : Wdg Initialization Hardware
MCAL-5558	SWS_Wdg_00031 : De-initialization Interface

6.1.2 Wdg_SetTriggerCondition

Sets the timeout value for the trigger counter.

Design Identifier	Description
MCAL-5592	SWS_Wdg_00136 : Wdg Set Trigger Condition Reset Timeout Counter
MCAL-5555	SWS_Wdg_00146 : Wdg Set Trigger Condition DET WDG_E_PARAM_TIMEOUT
MCAL-5593	SWS_Wdg_00138 : Wdg Set Trigger Condition Timeout Value Interpretation
MCAL-5592	SWS_Wdg_00136 : Wdg Set Trigger Condition Reset Timeout Counter
MCAL-5590	SWS_Wdg_00139 : Wdg Set Trigger Condition Current Watchdog Mode
MCAL-5571	SWS_Wdg_00140 : Wdg Set Trigger Condition Timeout Value Zero

6.1.3 Wdg_GetVersionInfo

Returns the version information of the module.

Design Identifier	Description
MCAL-5575	SWS_Wdg_00174 : Wdg Get Version Info WDG_E_PARAM_POINTER

6.1.4 Wdg_Trigger

Trigger routine which should be called by application periodically.

Design Identifier	Description
MCAL-5604	WDG: Wdg_Trigger API Service ID
MCAL-5603	SWS_Wdg_00034 : Wdg Trigger Routine Start Address

6.1.5 Wdg_RegisterReadback

As noted from previous implementation, the WDG configuration registers could be potentially corrupted by other entities (s/w or h/w). One of the recommended detection methods would be to periodically read-back the configuration and confirm configuration is consistent. The service API defined below shall be implemented to enable this detection. Constraint: Should be called only after module initialization.



The critical register listed is a recommendation and implementation shall determine appropriate registers.

This service could potentially be turned OFF in the configurator.

	Description	Comments
Service Name	Wdg_RegisterReadback	Can potentially be turned OFF.
Design ID	MCAL-5553	
Syntax	Std_ReturnTypeWdg_RegisterReadback(P2VAR(Wdg_RegisterReadbackType ,AUTOMATIC, WDG_APPL_DATA) regRbPtr)	Wdg_RegisterReadbackType Defines the type, that holds critical values, refer below.
Service ID	0x06	
Sync / Async	Sync	
Reentrancy	Non Reentrant	



Parameter in	None	None
Parameters out	regRbPtr	A pointer of type Wdg_RegisterReadbackType , which holds the read back values.
Return Value	Standard return type	E_OK or E_NOT_OK in case of DET error.
Design Identifier		Description
MCAL-5553		WDG : Register Readback : service API
MCAL-5619		WDG: Safety Diagnostic: Reference and Example for external WDG

7 Performance Objectives

7.1 Resource Consumption Objectives

ROM - Program(KB)	ROM - Data(KB)	RAM - Program(KB)	RAM - Data(KB)	Stack Size (KB)	EEPROM (KB)	% CPU Utilization
30	NA	NA	2	2	NA	NA

7.2 Critical timing and Performance

Not Applicable



8 Decision Analysis & Resolution (DAR)

Sections below list some of the important design decisions and rationale behind those decisions.

8.1 Watchdog SOC Reset Functionality

The watchdog hardware generates a violation interrupt or ESM interrupt after a programmable period, if no correct key sequence is written to the RTI watchdog key register.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
1	Guarantee reset on watchdog expiry and minimal software load on CPU.	<p>CPU Interrupt Mode: RTI expires (AUTOSAR Core) -> R5F Interrupt -> Safety R5F -> DMSC -> Reset Core Available Alternatives.</p> <p>Advantages:</p> <ul style="list-style-type: none"> • Simple to implement. • No dependency on external modules. <p>Disadvantages:</p> <ul style="list-style-type: none"> • If CPU is not able to execute the ISR (eg DDR,OCMC Failure) reset may not occur. 	ESM interrupt mode.	<p>ESM will make sure to signal the severe device failure if interrupt occurs.</p> <p>.</p>	Dependency on external module to reset the core.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		ESM Interrupt Mode: RTI expires (AUTOSAR Core) -> ESM Interrupt -> ESM pin Error -> External controller resets the whole SoC. Advantages: <ul style="list-style-type: none"> • No overhead to CPU. Disadvantages: <ul style="list-style-type: none"> • Dependency on external module to reset the core. 			

8.2 Watchdog Service Routine

The routine servicing a watchdog shall be implemented as an interrupt routine driven by a hardware timer/GPT. Refer SWS_Wdg_00166 AUTOSAR WDG specification in [Reference 1 - AUTOSAR 4.3.1](#).

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
1	There should be no dependency on external module.	Driven by Hardware Timer Advantages: <ul style="list-style-type: none"> Guarantees timing constraints required for windowed watchdog conditions. Disadvantages: <ul style="list-style-type: none"> Dependency on external module. 	Driven by Application Periodically.	To avoid dependency on the external module, Wdg_Trigger API is provided which should be called by application periodically.	Application need to take care of the latency by calling service API within the time window. .

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		Driven by Application Periodically Advantages: <ul style="list-style-type: none"> No dependency on external module. Disadvantages: <ul style="list-style-type: none"> Application need to take care of the latency by calling service API within the time window. 			

9 Testing Guidelines

The sections below identify some of the aspects of design that would require emphasis during testing of this design implementation.

Timeout

Test cases shall ensure watchdog generates ESM interrupts and thereby reset occurs for the configured timeout value.

Also test with different set of timeout values (Equivalence partition test).

State Transitions

Test cases shall exercise all state transitions as detailed in section (States).

Modes

Test cases shall ensure watchdog support both fast/slow modes.

Trigger Condition

Test cases shall ensure driver test with different set of trigger condition timeout values (Equivalence partition test).

Window Sizes

Test cases shall ensure watchdog operation test with all window sizes that hardware supports.

Window Violation Test

Test cases shall ensure watchdog start time violation test.

Test for all instances








Test cases shall ensure watchdog operation for all the RTI instances supported.





Test for different clock sources

Test cases shall ensure watchdog operation for all the RTI clock sources supported.

10 Template Revision History

Author Name	Description	Version	Date
Yaniv Machani	Initial version	0.1	 03 Oct 2018
Yaniv Machani	Updated to include EP views	0.4	 02 Nov 2018
Yaniv Weizman	Restructuring and editing to further meet the A-SPICE and EP requirements	0.5	 27 Dec 2018
Yaniv Weizman	Adding link to Architecture review template	0.6	 22 Oct 2019
Yaniv Weizman	Adding requirement type column for requirements table (Functional/Non-Functional). Adding DAR table	0.65	 13 Nov 2019



Author Name	Description	Version	Date
Yaniv Weizman	Adding tables for Testing guidelines	0.7	 18 Nov 2019
Krishna	Updated based on ASPICE requirements	0.8	 20 Aug 2020
Krishna	Updated based on the feedback from Jon N	0.9	 09 Oct 2020
Krishna	Updated the traceability scheme	1.0	 17 Dec 2020