

# AUTOSAR MCAL R4.0.3

## User's Manual

WDG Driver Component Ver.1.0.8

Embedded User's Manual

Target Device:  
RH850/P1x

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (<http://www.renesas.com>).



## Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.



## Abbreviations and Acronyms

Abbreviation / Acronym	Description
ADC	Analog Digital Converter
ANSI	American National Standards Institute
API	Application Programming Interface
AUTOSAR	Automotive Open System ARchitecture
ARXML	AUTOSAR Markup EXtensible Language
BSW	Basic SoftWare
CAN	Controller Area Network
DEM	Diagnostic Event Manager
DET/Det	Development Error Tracer
DIO	Digital Input And Output
ECU	Electronic Control Unit
EEPROM	Electrical Erasable Programmable Read Only Memory
FR	FlexRay
GHS	Green HillS
GNU	GNU is Not Unix
GPT	General Purpose Timer
HW	HardWare
ID/Id	Identifier
ISR	Interrupt Service Routine
IMR	Interrupt Mask Register
I/O	Input/Output
ICU	Input Capture Unit
INTWDTn	Interrupt Watch Dog Timer.
LIN	Local Interconnect Network
MCAL	Microcontroller Abstraction Layer
MCU	MicroController Unit
MHz	Mega Hertz
OS	Operating System
PWM	Pulse Width Modulation
PDF	Parameter Definition File
RAM	Random Access Memory
ROM	Read Only Memory
RTE	Run Time Environment
SCI	Serial Communication Interface
SPI	Serial Peripheral Interface
SCHM/SchM	Scheduler Manager
SV	Supervisor Mode
SW	SoftWare
WDG/wdg	WatchDog
WDT	WatchDog Timer
WDGIF	WatchDog Interface

WDTAnTRES	WatchDog Timer A Trigger Reset
WDTATCKI	WatchDog Timer A Trigger Clock Input
WDTAnMD	WatchDog Timer Mode
WDTAnREF	Watch Dog Timer A Reference

## Definitions

Term	Represented by
Sl. No.	Serial Number
WDTAEVAC	Watchdog Timer Enable Register for Varying Activation Code
WDTAMD	Watchdog Timer Mode Register
WDTAWDTE	Watchdog Timer Enable Register for Fixed Activation Code

## Table of Contents

<b>Chapter 1</b>	<b>Introduction .....</b>	<b>11</b>
1.1.	Document Overview .....	13
<b>Chapter 2</b>	<b>Reference Documents.....</b>	<b>15</b>
<b>Chapter 3</b>	<b>Integration And Build Process .....</b>	<b>17</b>
3.1.	WDG Driver Component Makefile .....	17
3.1.1.	Folder Structure.....	17
<b>Chapter 4</b>	<b>Forethoughts .....</b>	<b>19</b>
4.1.	General.....	19
4.2.	Preconditions .....	20
4.3.	Data Consistency.....	20
4.4.	WDG State Diagram .....	21
4.5.	WDTA 75% ISR Usage Details for R4.0.3.....	22
4.6.	Deviation List .....	24
4.7.	User mode and supervisor mode.....	25
4.8.	Register Write Verify .....	25
<b>Chapter 5</b>	<b>Architecture Details.....</b>	<b>27</b>
<b>Chapter 6</b>	<b>Registers Details .....</b>	<b>29</b>
<b>Chapter 7</b>	<b>Interaction Between The User And WDG Driver Component.....</b>	<b>31</b>
7.1.	Services Provided By WDG Driver Component To the User .....	31
<b>Chapter 8</b>	<b>WDG Driver Component Header And Source File Description .....</b>	<b>33</b>
<b>Chapter 9</b>	<b>Generation Tool Guide .....</b>	<b>37</b>
<b>Chapter 10</b>	<b>Application Programming Interface .....</b>	<b>39</b>
10.1.	Imported Types .....	39
10.1.1.	Standard Types .....	39
10.1.2.	Other Module Types.....	39
10.2.	Type Definitions .....	39
10.2.1.	Wdg_59_DriverA_ConfigType.....	39
10.3.	Function Definitions .....	40
10.3.1	Wdg_59_DriverA_Init .....	40
10.3.2	Wdg_59_DriverA_SetMode .....	41
10.3.3	Wdg_59_DriverA_SetTriggerCondition.....	41
10.3.4	Wdg_59_DriverA_GetVersionInfo.....	42

<b>Chapter 11 Development And Production Errors.....</b>	<b>43</b>
11.1. WDG Driver Component Development Errors .....	43
11.2. WDG Driver Component Production Errors.....	44
<b>Chapter 12 Memory Organization.....</b>	<b>45</b>
<b>Chapter 13 P1M Specific Information.....</b>	<b>47</b>
13.1. Interaction Between The User And WDG Driver Component.....	47
13.1.1. ISR Function Mapping Interrupt Vector Table .....	47
13.1.2. Translation Header File .....	47
13.1.3. Parameter Definition File.....	48
13.2. Sample Application.....	48
13.2.1 Sample Application Structure .....	48
13.2.2 Building Sample Application.....	50
13.2.2.1 Configuration Example .....	50
13.2.2.2 Debugging The Sample Application .....	50
13.3. Memory and Throughput for R4.0.3.....	51
13.3.1 ROM/RAM Usage .....	51
13.3.2 Stack Depth.....	52
13.3.3 Throughput Details .....	52
<b>Chapter 14 Release Details .....</b>	<b>53</b>



## List of Figures

Figure 1-1	System Overview Of AUTOSAR Architecture .....	11
Figure 1-2	System Overview Of The WDG Driver In AUTOSAR MCAL Layer .....	12
Figure 4-1	State Diagram of WDG .....	21
Figure 4-2	WDG behavior during Data exchange with hardware .....	22
Figure 4-3	WDG behavior when Wdg_SetTriggerCondition is called .....	23
Figure 5-1	Watch Driver And Watchdog Interface Architecture .....	27
Figure 5-2	Basic Architecture Of WDG Component .....	28
Figure 12-1	Memory Organization Of WDG Driver Component .....	45
Figure 13-1	Overview Of WDG Driver Sample Application .....	48

## List of Tables

Table 4-1	WDG Driver Deviation List .....	24
Table 4-2	AUTOSAR Deviation List .....	25
Table 4-3	Supervisor mode and User mode details .....	25
Table 6-1	Register Details .....	29
Table 8-1	Description Of The WDG Driver Component Files .....	34
Table 10-1	APIs provided by the WDG Driver Component .....	40
Table 11-1	DET Errors of WDG Driver Component .....	43
Table 11-2	DEM Errors of WDG Driver Component .....	44
Table 13-1	Interrupt Vector Table .....	47
Table 13-2	PDF information for P1M .....	48
Table 13-3	ROM/RAM Details Without DET .....	51
Table 13-4	ROM/RAM Details With DET .....	52
Table 13-5	Throughput Details Of The APIs .....	52



# Chapter 1 Introduction

The purpose of this document is to describe the information related to WDG Driver Component for Renesas P1x microcontrollers.

This document shall be used as reference by the users of WDG Driver Component for P1M Device. The information specific to P1M Device like, compiler, linker, assembler, integration and build process for application along with the memory consumption and throughput information are provided.

This document shall be used as reference by the users of WDG Driver Component. This document describes the common features of WDG Driver Component. This document is intended for the developers of ECU software using Application Programming Interfaces provided by AUTOSAR. The system overview of complete AUTOSAR architecture is shown in the below Figure:

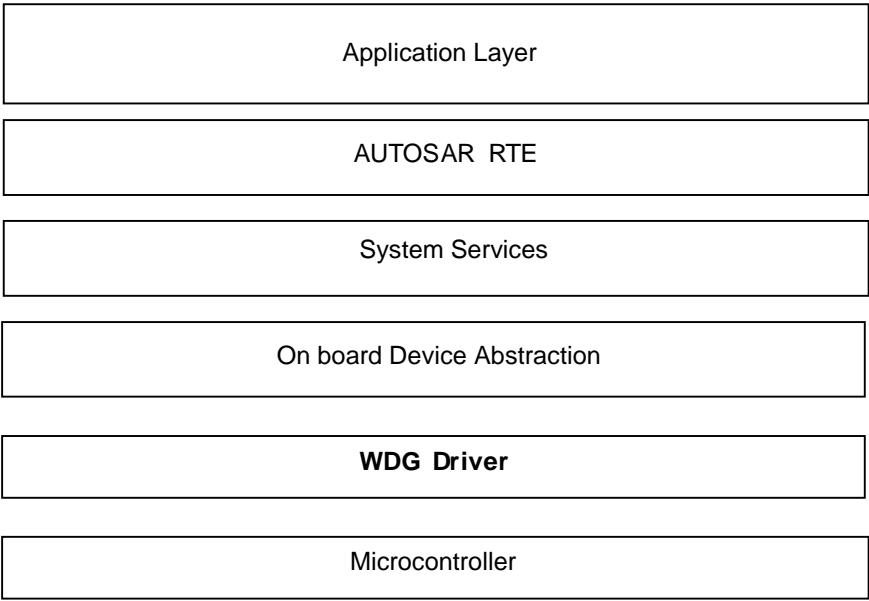
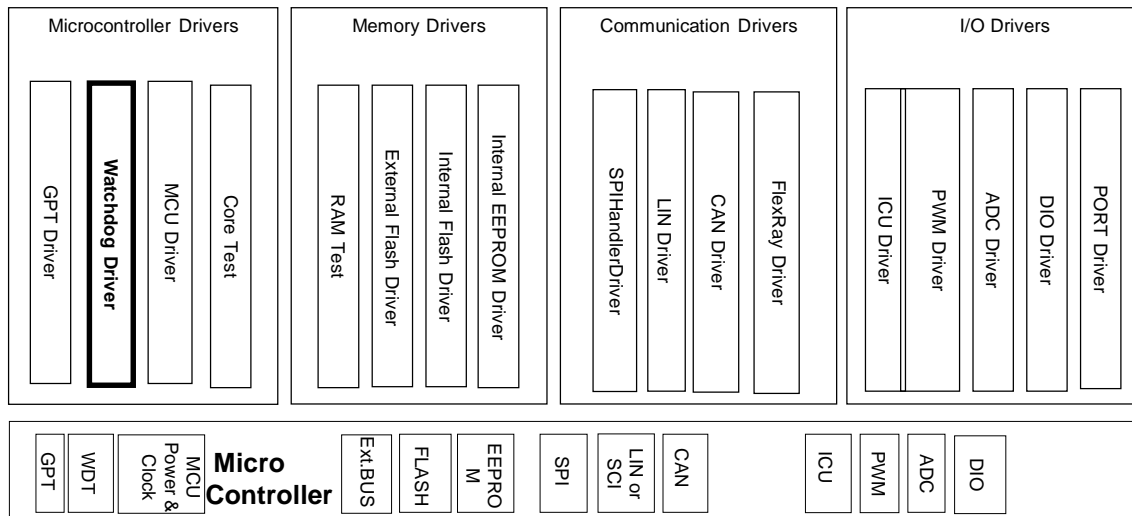


Figure 1-1 System Overview Of AUTOSAR Architecture

The WDG Component comprises embedded software and the Configuration Tool to achieve scalability and configurability.

The WDG Generation Tool is a command line tool that accepts ECU configuration description files as input and generates source and header files. The configuration description is an ARXML file that contains information about the configuration for Watchdog timer. The tool generates the Wdg\_59\_DriverA\_PBcfg.c, Wdg\_59\_DriverA\_Cfg.h and Wdg\_59\_DriverA\_Cbk.h for Watchdog Driver A.

The Figure in the following page depicts the WDG Driver as part of layered AUTOSAR MCAL Layer:



**Figure 1-2 System Overview Of The WDG Driver In AUTOSAR MCAL Layer**

Watchdog Driver module provides the services for initializing, changing the operation mode and triggering the watchdog.

## 1.1. Document Overview

The document has been segmented for easy reference. The table below provides user with an overview of the contents of each section:

Section	Contents
Section 1 (Introduction)	This section provides an introduction and overview of WDG Driver Component.
Section 2 (Reference Documents)	This section lists the documents referred for developing this document.
Section 3 (Integration and Build Process)	This section explains the folder structure, Makefile structure for WDG Driver Component. This section also explains about the Makefile descriptions, Integration of WDG Driver Component with other components, building the WDG Driver Component along with a sample application.
Section 4 (Forethoughts)	This section provides brief information about the WDG Driver Component, the preconditions that should be known to the user before it is used, data consistency details, WDG State Diagram, WDTA 75% ISR Usage Details, deviation list, Support For Different Interrupt Categories, user-mode and supervisor mode API support list, register read-back.
Section 5 (Architecture Details)	This section describes the layered architectural details of the WDG Driver Component.
Section 6 (Register Details)	This section describes the register details of WDG Driver Component.
Section 7 (Interaction Between The User And WDG Driver Component)	This section describes interaction of the WDG Driver Component with the upper layers.
Section 8 (WDG Driver Component Header And Source File Description)	This section provides information about the WDG Driver Component source files is mentioned. This section also contains the brief note on the tool generated output file.
Section 9 (Generation Tool Guide)	This section provides information on the WDG Driver Component Code Generation Tool.
Section 10 (Application Programming Interface)	This section explains all the APIs provided by the WDG Driver Component.
Section 11 (Development And Production Errors)	This section lists the DET and DEM errors.
Section 12 (Memory Organization)	This section provides the typical memory organization, which must be met for proper functioning of component.
Section 13 (P1M Specific Information)	This section provides the P1M Specific Information.
Section 14 (Release Details)	This section provides release details with version name and base version.



## Chapter 2    Reference Documents

Sl. No.	Title	Version
1.	Autosar R4.0 AUTOSAR_SWS_WatchdogDriver.pdf	2.5.0
2.	AUTOSAR BUGZILLA ( <a href="http://www.autosar.org/bugzilla">http://www.autosar.org/bugzilla</a> ) Note: AUTOSAR BUGZILLA is a database, which contains concerns raised against information present in AUTOSAR Specifications.	-
3.	r01uh0436ej0130_rh850p1x.pdf	1.30
4.	AUTOSAR_SWS_CompilerAbstraction.pdf	3.2.0
5.	AUTOSAR_SWS_MemoryMapping.pdf	1.4.0
6.	AUTOSAR_SWS_PlatformTypes.pdf	2.5.0





## Chapter 3      Integration And Build Process

In this section the folder structure of the WDG Driver Component is explained. Description of the Makefiles along with samples is provided in this section.

**Remark** The details about the C Source and Header files that are generated by the WDG Driver Generation Tool are mentioned in the “R20UT3729EJ0101-AUTOSAR.pdf”.

### 3.1. WDG Driver Component Makefile

The Makefile provided with the WDG Driver Component consists of the GNU Make compatible script to build the WDG Driver Component in case of any change in the configuration. This can be used in the upper level Makefile (of the application) to link and build the final application executable.

#### 3.1.1. Folder Structure

The files are organized in the following folders:

**Remark** Trailing slash ‘\’ at the end indicates a folder

```
X1X\common_platform\modules\wdg\src\Wdg_59_DriverA.c
                                \Wdg_59_DriverA_Irq.c
                                \Wdg_59_DriverA_Private.c
                                \Wdg_59_DriverA_Ram.c
                                \Wdg_59_DriverA_Version.c

X1X\common_platform\modules\wdg\include\Wdg_59_DriverA.h
                                \Wdg_59_DriverA_Debug.h
                                \Wdg_59_DriverA_Irq.h
                                \Wdg_59_DriverA_PBTypes.h
                                \Wdg_59_DriverA_Private.h
                                \Wdg_59_DriverA_Ram.h
                                \Wdg_59_DriverA_RegWrite.h
                                \Wdg_59_DriverA_Types.h
                                \Wdg_59_DriverA_Version.h

X1X\P1x\modules\wdg\sample_application\<SubVariant>\make\<Compiler>
                                \App_WDG_P1M_Sample.mak

X1X\P1x\modules\wdg\sample_application\<SubVariant>\obj\<Compiler>

(Note: For example, compiler can be ghs.)

X1X\P1x\modules\wdg\generator
```

\R403\_WDG\_P1x\_BSWMDT.arxml

X1X\common\_platform\modules\wdg\generator\Wdg\_X1x.dll

tools\RUCG\RUCG.exe

X1X\P1x\common\_family\generator

\Sample\_Application\_P1x.trxml

\P1x\_translation.h

X1X\P1x\modules\wdg\user\_manual

(User manuals will be available in this folder)

**Notes:**

1. <Compiler> can be ghs.
2. <SubVariant> can be P1M.
3. <AUTOSAR\_version> should be 4.0.3.

## Chapter 4 Forethoughts

### 4.1. General

Following information will aid the user to use the WDG Driver Component software efficiently:

- The WDG Component does not enable or disable the ECU or Microcontroller power supply. The upper layer should handle this operation.
- Option byte values required for the operation of watchdog will be flashed through Start up code.
- The WDG Component does not implement any scheduled functions.
- WDG Component Implements Call Back Notification function to notify WriteVerify Error.
- Example code mentioned in this document shall be taken only as a reference for implementation.
- The Watchdog hardware supports only Driver A. Hence, WDG Driver Component is implemented as Driver A. WDG\_DRIVER\_INSTANCE variable of Base Make file is updated for Driver A.
- All development errors will be reported to Det by using the API Det\_ReportError() provided by DET.
- All production errors will be reported to Dem by using the API Dem\_ReportErrorStatus() provided by DEM.
- It should be ensured that the respective clock source is switched ON before Watchdog is set to corresponding Clock Unit in Wdg\_59\_DriverA\_Init() API.
- The API Wdg\_59\_DriverA\_SetTriggerCondition() initializes the trigger counter global variable with timeout value divided by either slow or fast time Value generated by the configuration.
- For WDG Reset functionality in debug mode, unmask the reset in debug mode during debug session with GHS command "target pinmask k".
- The file Interrupt\_VectorTable.c provided is just a Demo and not all interrupts will be mapped in this file. So the user has to update the Interrupt\_VectorTable.c as per his configuration.
- The accesses to HW registers is possible only in the low level driver layer. The MCAL user does never write or read directly from any register, but uses the AUTOSAR standard API provided by the MCAL.
- The Wdg\_SetMode() function must only support mode change from WDGIF\_OFF\_MODE to WDGIF\_FAST\_MODE or WDGIF\_SLOW\_MODE. This is a limitation by the hardware and WDG cannot be stopped in run time.
- "WDG\_SETTINGS\_SLOW" and "WDG\_SETTINGS\_FAST" is configured from the list of clock selections (16 choices are possible) and depending on the mode configured for "WDG\_DEFAULT\_MODE", watchdog settings is initialized in the API Wdg\_59\_DriverA\_Init ().

## 4.2. Preconditions

Following preconditions have to be adhered by the user, for proper functioning of the WDG Driver Component:

- The user should ensure that WDG Component API requests are invoked in the correct and expected sequence along with correct input arguments.
- User should ensure that the appropriate option bytes are flashed for the configured mode in the watchdog driver module.
- Validation of input parameters are done only when the static configuration parameter `WDG_59_DRIVER_A_DEV_ERROR_DETECT` is enabled. Application should ensure that the right parameters are passed while invoking the APIs when `WDG_59_DRIVER_A_DEV_ERROR_DETECT` is disabled.
- A mismatch in the version numbers will result in compilation error. Ensure that the correct versions of the header and the source files are used.
- The files `Wdg_59_DriverA_Cfg.h`, `Wdg_59_DriverA_PBcfg.c` and `Wdg_59_DriverA_Cbk.h` generated using watchdog driver generation tool has to be linked along with WDG Component source files.
- File `Wdg_59_DriverA_PBcfg.c` generated for single configuration set using Watchdog Driver Generation Tool can be compiled and linked independently.
- The WDG Component needs to be initialized before accepting any API requests. `Wdg_59_DriverA_Init` should be called by the ECU State Manager Module to initialize WDG Component. It should not be called more than once.
- User have the responsibility to enable or disable the critical protection using the parameter `WdgCriticalSectionProtection`. By enabling parameter `WdgCriticalSectionProtection`, Microcontroller HW registers which suffer from concurrent access by multiple tasks are protected.

## 4.3. Data Consistency

To support the re-entrance and interrupt services, the AUTOSAR WDG component will ensure the data consistency while switching the watchdog mode and during the watchdog trigger routine. The WDG Driver component will use `SchM_Enter_Wdg` and `SchM_Exit_Wdg` functions. The `SchM_Enter_Wdg` function is called before the data needs to be protected and `SchM_Exit_Wdg` function is called after the data is accessed.

The following exclusive areas along with scheduler services are used to provide data integrity for shared resources:

- `TRIGG_PROTECTION`
- `MODE_SWITCH_PROTECTION`

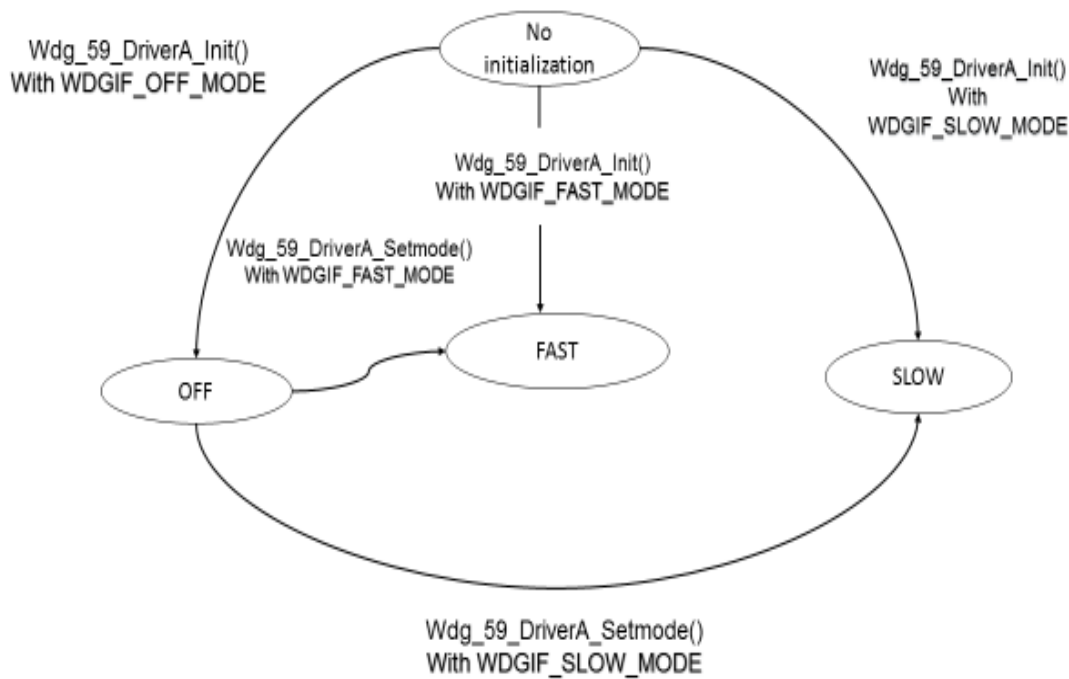
The protection areas `TRIGG_PROTECTION` and `MODE_SWITCH_PROTECTION` are used to protect the WDG triggering and WDG mode switching respectively.

The functions `SchM_Enter_Wdg` and `SchM_Exit_Wdg` can be disabled by disabling the configuration parameter '`WdgCriticalSectionProtection`'.

**Note:** The highest measured duration of a critical section is 1.575 micro seconds measured for Wdg\_59\_DriverA\_SetMode API.

#### 4.4. WDG State Diagram

The State diagram of WDG Driver is as shown below



**Figure 4-1 State Diagram of WDG**

WDG Driver supports following modes

- 1.WDGIF\_OFF\_MODE
- 2.WDGIF\_SLOW\_MODE
- 3.WDGIF\_FAST\_MODE

Like shown in the above figures when WDG Driver is initialized by the API Wdg\_59\_DriverA\_Init(), the WDG Driver gets into one of the modes based on the default value configured during configuration. When default mode is WDGIF\_OFF\_MODE, Wdg\_59\_DriverA\_Init() will not try to disable the WDG. It will just not start the WDG counter. Also the modes can be changed by the API Wdg\_59\_DriverA\_SetMode() only once after Wdg\_59\_DriverA\_Init(), if the current mode is WDGIF\_OFF\_MODE.

## 4.5. WDTA 75% ISR Usage Details for R4.0.3

WDG Driver using '75% interrupt output' feature services the Watchdog hardware to trigger watchdog hardware as long as the trigger condition is valid. If the trigger condition becomes invalid the Wdg Driver stops triggering and the watchdog expires.

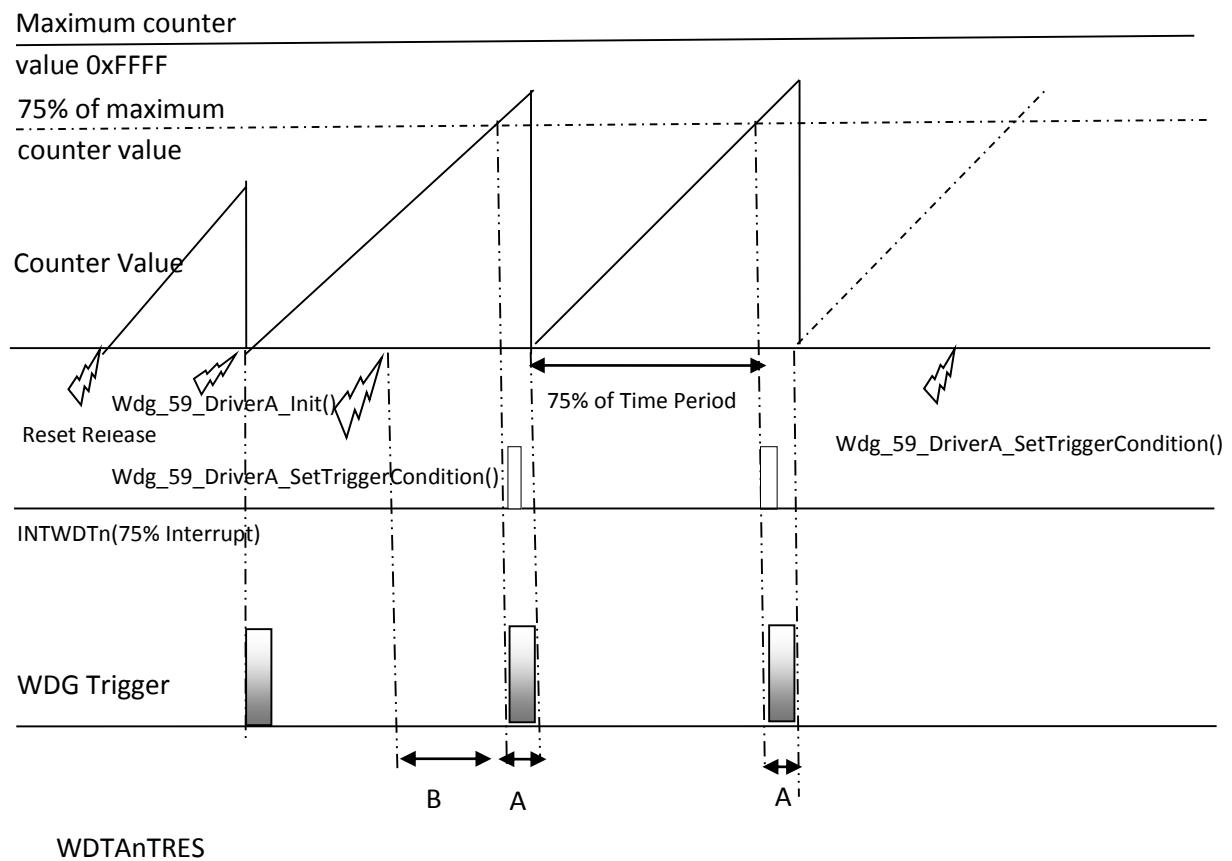


Figure 4-2 WDG behavior during Data exchange with hardware

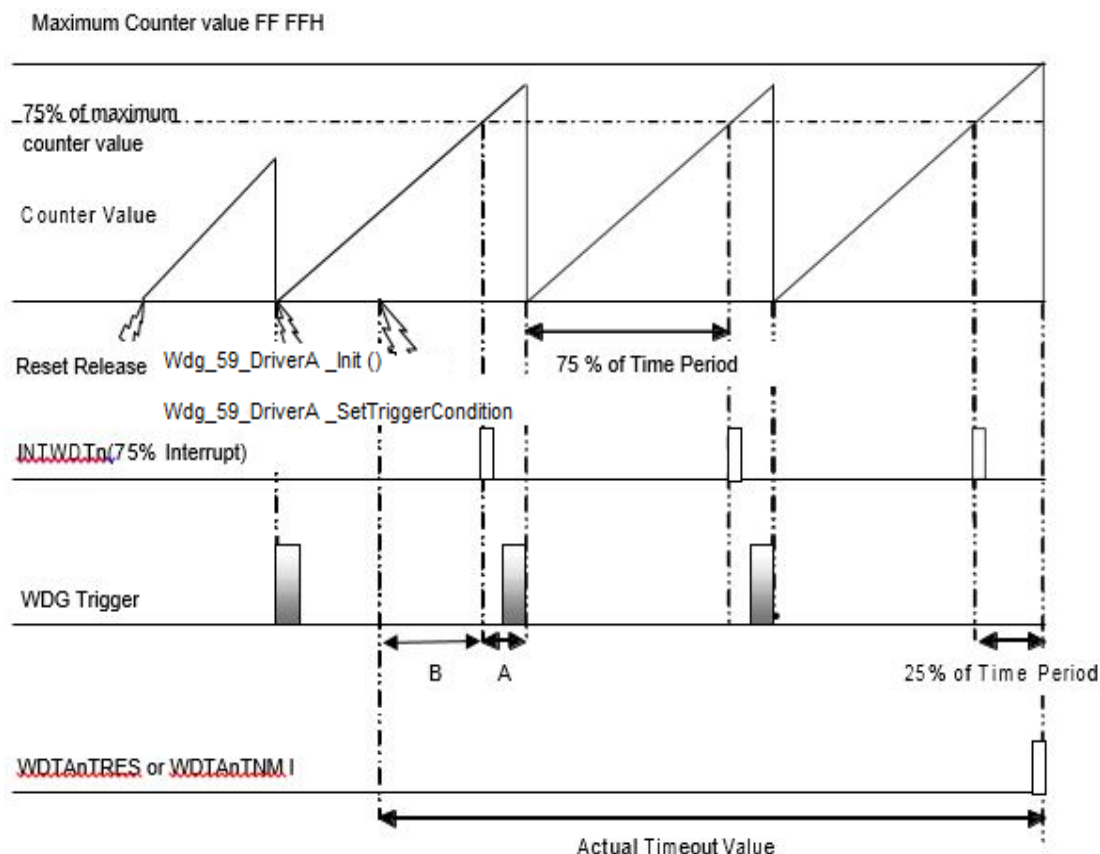


Figure 4-3 WDG behavior when Wdg\_SetTriggerCondition is called

**Note** User should adjust the Timeout value in such a way that the corrections of 'A' and 'B' are considered while passing the 'timeout' value to API 'Wdg\_59\_DriverA\_SetTriggerCondition'.

The above figure illustrates the scenario where Wdg\_59\_DriverA\_SetTriggerCondition API is called before the expiry of the Initial Timeout value.

#### The 75% duration calculation for one WDG trigger cycle in slow mode

WDTATCKI = 240 kHz

For example, considering current mode settings =  $WDTATCKI/2^{16}$

Period =  $2^{16}/240k = 0.273$  sec

Total window time = 273 msec

75% interrupt time = 204.7 msec

Generation tool will round off the 75% interrupt time "204.7 msec" to "205 msec" and rounded value is displayed on the command prompt. For the above example the information on command prompt for slow mode will be displayed as given below.

**The duration of 75% of one WDG trigger cycle for slow mode is <205 msec>**

If the timeout value passed by the API Wdg\_59\_DriverA\_Settriggercondition is 410 msec, then the counter value will be calculated in the WDG Driver as 2.

### The duration of 75% of one WDG trigger cycle calculation for fast mode

WDTATCKI = 240 kHz

For example, considering current mode settings =  $WDTATCKI/2^9$

Period =  $2^9/240k = 0.0021$  sec

Total window time = 2.1 msec

75% interrupt time = 1.5 msec

Generation tool will round off the 75% interrupt time “1.5 msec” to “2 msec” and rounded value is displayed on the command prompt. For the above example the information on command prompt for fast mode will be displayed as given below.

### The duration of 75% of one WDG trigger cycle for fast mode is <2 msec>

If the timeout value passed by the API

Wdg\_59\_DriverA\_Settriggercondition() is 50 msec, then the counter value will be calculated in the WDG Driver as 25.

The API Wdg\_59\_DriverA\_SetTriggerCondition() will not trigger the watchdog hardware it will only calculate the trigger counter value.

In General, the user should use the below formula while calculating the Timeout Period by considering the corrections of 75% duration round off, A and B values.

Timeout Period = (Trigger Count)\* (75% of Time Period + A)+B

where ‘A’ is the time required for the ISR to trigger the WDG hardware and ‘B’ is the time gap between Wdg\_59\_DriverA\_SetTriggerCondition() execution and next WDG trigger from 75% ISR.

## 4.6. Deviation List

Table 4-1 WDG Driver Deviation List

Sl. No.	Description	AUTOSAR Bugzilla
1.	'WDG116_Conf' 'WdgDisableAllowed' will have no effect if it is configured as TRUE due to Hardware limitation.	-
2.	'WDG025 ','WDG173' WDG HW unit does not allow itself to be disabled, this is a limitation. Therefore, if default mode given in the provided configuration set is WDGIF_OFF_MODE, Wdg_59_DriverA_Init function will not try to disable the WDG. It will just not start the WDG counter. Also Wdg_59_DriverA_Init shall not report production error WDG_E_MODE_FAILED.	-
3.	If the API Wdg_59_DriverA_SetTriggerCondition, is invoked with the timeout value "0" will not result in instantaneous watchdog reset of the ECU like mentioned in WDG140, instead the trigger counter will be set to "0" and watchdog reset will occur after the WatchDog counter value has reached its maximum value.	-



4.	'WDG163' Wdg triggering for servicing the hardware watchdog is handled by WDG_59_DRIVER_A_TRIGGERFUNCTION_ISR interrupt service routine. GPT callbacks are not used for servicing the Watchdog hardware.	-
----	--	---

**Table 4-2 AUTOSAR Deviation List**

SI.No.	Autosar ID	Description	Justification
1.	BSW00347	Naming separation of different instances of BSW drivers	Renesas WDG driver doesn't follow BSW00347 Autosar requirement regarding naming separation of different instances of BSW drivers.

## 4.7. User mode and supervisor mode

The below table specifies the APIs which can run in user mode, supervisor mode or both modes

**Table 4-3 Supervisor mode and User mode details**

SI.No	API Name	User Mode	Supervisor mode	Known limitation in User
1	Wdg_59_DriverA_Init	-	x	-
2	Wdg_59_DriverA_SetMode	x	x	-
3	Wdg_59_DriverA_SetTriggerCondition	x	x	-
4	Wdg_59_DriverA_GetVersionInfo	x	x	-

Note: Implementation of Critical Section is not dependent on MCAL. Hence Critical Section is not considered to the entries for User mode in the above table.

## 4.8. Register Write Verify

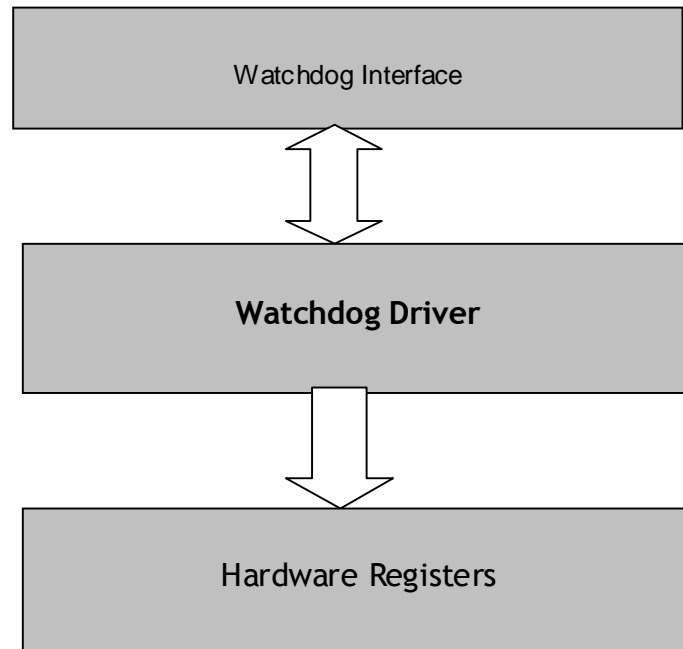
Register write-verify is a functional safety based implementation, where the verification of the control register's content after the write operation is carried out. After writing to control registers, content of the registers are read back and verified against the expected content to make sure that register content has been written correctly.

The purpose of this implementation is to detect random HW faults (transient / permanent) This can happen on the bus while writing to the configuration registers or because of faulty registers which will potentially lead to wrong configuration and wrong operation.



## Chapter 5 Architecture Details

The WDG Driver architecture is shown in the following figure. The WDG user shall directly use the APIs to configure and execute the WDG conversions:



**Figure 5-1 Watch Driver And Watchdog Interface Architecture**

Watchdog Interface invokes the corresponding Driver. The Driver APIs will access the hardware register of the Watchdog Timers for changing the mode and trigger the Watchdog Timer.

**Watchdog Driver component:**

The Watchdog Driver component is composed of following modules:

- Watchdog Driver Initialization module
- Watchdog Driver SetMode module
- Watchdog Driver SetTriggerCondition module
- Watchdog Driver VersionInfo module

The basic architecture of the Watchdog Driver component is illustrated in the following figure:

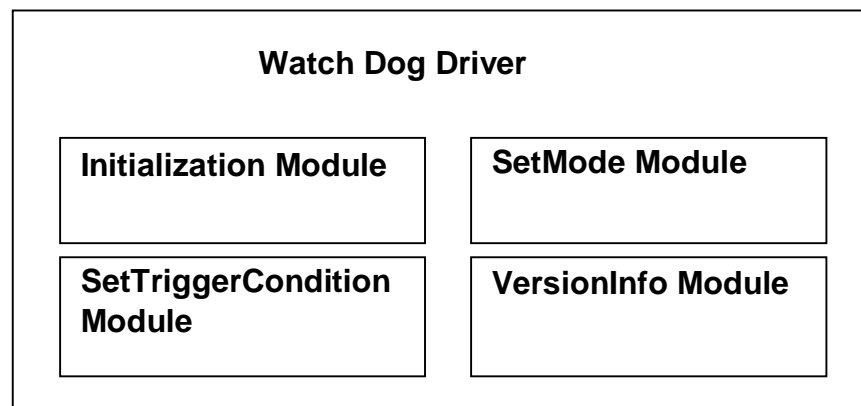


Figure 5-2 Basic Architecture Of WDG Component

#### Watchdog Driver Initialization module:

This module initializes the watchdog driver and watchdog hardware. It provides the API `Wdg_59_DriverA_Init()`. This API should be invoked before the usage of any other APIs of Watchdog Driver Module.

#### Watchdog Driver SetMode module:

This module will handle the functionality for setting the modes. It provides the API `Wdg_59_DriverA_SetMode()`. Following are the possible mode settings:

- `WDGIF_SLOW_MODE`
- `WDGIF_FAST_MODE`

**Remark** The above settings are configured using the `WDTAMD` register. `SetMode` will support mode switch as described in the *chapter 4.4 WDG State Diagram*.

`SetMode` API will set module's state to `WDG_BUSY` during execution and reset the module's state to `WDG_IDLE` before return.

#### Watchdog Driver SetTriggerCondition module:

This module will handle the functionality to reset the watchdog timeout counter according to the timeout value passed. It provides the API `Wdg_59_DriverA_SetTriggerCondition`.

There are two types of Activation codes to trigger the Watchdog. They are

- Fixed Activation Code.
- Varying Activation code.

Depending on the Activation code chosen, this function has to trigger the corresponding register.

- `WDTAWDTE` register will be used for Fixed Activation Code.
- `WDTAEVAC` register will be used for Varying Activation Code.

#### Watchdog Driver VersionInfo module:

This module will provide the current version of the Watchdog Driver Module. It contains the API `Wdg_59_DriverA_GetVersionInfo()`.

## Chapter 6 Registers Details

This section describes the register details of WDG Driver Component.

**Table 6-1 Register Details**

API Name	Registers	Config Parameter	Register Access r/w/rw	Macro/Variable
Wdg_59_DriverA_Init	IMRn	-	rw	WDG_59_DRIVER_A_INTWDTIMR_MASK
	WDTAnMD	WdgDefaultMode	rw	ucWdtamdDefaultValue.
	WDTAnEVAC	WdgVaryingActivationCodeMode	rw	WDG_59_DRIVER_A_RESTART, WDG_59_DRIVER_A_WDTAREF_ADDRESS
	WDTAnWDT E	WdgVaryingActivationCodeMode	rw	WDG_59_DRIVER_A_WDTAWDTE_MASK, WDG_59_DRIVER_A_RESTART
	WDTAnREF	-	r	WDG_59_DRIVER_A_WDTAREF_ADDRESS
Wdg_59_DriverA_Set Mode	WDTAnMD	-	rw	ucWdtamdSlowValue, ucWdtamdFastValue.
	WDTAnEVAC	WdgVaryingActivationCodeMode	rw	WDG_59_DRIVER_A_RESTART, WDG_59_DRIVER_A_WDTAREF
	WDTAnWDT E	WdgVaryingActivationCodeMode	rw	WDG_59_DRIVER_A_WDTAWDTE_MASK, WDG_59_DRIVER_A_RESTART
	WDTAnREF	-	r	WDG_59_DRIVER_A_WDTAREF_ADDRESS
Autosar R4.0: Wdg_59_DriverA_SetTriggerCondition	-	-	-	-
Wdg_59_DriverA_GetVersionInfo	-	-	-	-



## **Chapter 7      Interaction Between The User And WDG Driver Component**

The details of the services supported by the WDG Driver Component to the upper layer users are provided in the following sections:

### **7.1.      Services Provided By WDG Driver Component To the User**

The WDG Driver Component provides the following functions to upper layers:

- To Initialize Watchdog Timer
- To Set the Mode of the Watchdog Timer
- To handle the functionality of calculating the trigger counter value
- To Read the WDG Component Version Information.





## Chapter 8 WDG Driver Component Header And Source File Description

This section explains the WDG Driver Component's C Source and C Header files. These files have to be included in the project application while integrating with other modules.

The C header file generated by WDG Driver Generation Tool:

- Wdg\_59\_DriverA\_Cfg.h
- Wdg\_59\_DriverA\_Cbk.h

The C source file generated by WDG Driver Generation Tool:

- Wdg\_59\_DriverA\_PBcfg.c

The WDG Driver Component C header files:

- Wdg\_59\_DriverA.h
- Wdg\_59\_DriverA\_Debug.h
- Wdg\_59\_DriverA\_Irq.h
- Wdg\_59\_DriverA\_PBTypes.h
- Wdg\_59\_DriverA\_Private.h
- Wdg\_59\_DriverA\_Ram.h
- Wdg\_59\_DriverA\_Types.h
- Wdg\_59\_DriverA\_Version.h
- Wdg\_59\_DriverA\_RegWrite.h

The WDG Driver Component source files:

- Wdg\_59\_DriverA.c
- Wdg\_59\_DriverA\_Irq.c
- Wdg\_59\_DriverA\_Private.c
- Wdg\_59\_DriverA\_Ram.c
- Wdg\_59\_DriverA\_Version.c

The Stub C header files:

- Compiler.h
- Compiler\_Cfg.h
- MemMap.h
- Platform\_Types.h
- rh850\_Types.h
- Dem.h
- Dem\_Cfg.h
- Std\_Types.h
- Det.h

- Os.h
- Rte.h
- SchM\_Wdg\_59\_DriverA.h
- WdgLf\_Types.h

The description of the WDG Driver Component files is provided in the table below:

**Table 8-1 Description Of The WDG Driver Component Files**

File	Details
Wdg_59_DriverA_Cfg.h	This file is generated by the WDG Generation Tool for various WDG component pre-compile time parameters. Generated macros and the parameters will vary with respect to the configuration in the input ARXML file.
Wdg_59_DriverA_PBcfg.c	This file contains post-build configuration data. The structures related to WDG Initialization are provided in this file. Data structures will vary with respect to parameters configured.
Wdg_59_DriverA_Cbk.h	This file contains Prototype Declarations for WDG callback Notification Functions.
Wdg_59_DriverA.h	This file provides extern declarations for all the WDG Component APIs. This file provides service IDs of APIs, DET Error codes and type definitions for Watchdog Driver initialization structure. This header file shall be included in other modules to use the features of WDG Component.
Wdg_59_DriverA_Debug.h	This file provides Provision of global variables for debugging purpose.
Wdg_59_DriverA_Irq.h	It contains the external declaration for the interrupt functions used by WDG Driver component.
Wdg_59_DriverA_PBTtypes.h	This file contains the macros used internally by the WDG Component code and the structure declarations related to watchdog control registers.
Wdg_59_DriverA_Private.h	This file contains the declarations of the internally used functions.
Wdg_59_DriverA_Ram.h	This file contains the extern declarations for the global variables that are defined in Wdg_59_DriverA_Ram.c file and the version information of the file.
Wdg_59_DriverA_Types.h	This file contains the common macro definitions and the data types required internally by the WDG software component.
Wdg_59_DriverA_Version.h	This file contains the macros of AUTOSAR version numbers of all modules that are interfaced to WDG.
Wdg_59_DriverA_RegWrite.h	This file contains macro definitions for the registers write-verify functionality.
Wdg_59_DriverA.c	This file contains the implementation of all APIs.
Wdg_59_DriverA_Irq.c	This file contains the implementation of all the interrupt functions used by WDG Driver Component.
Wdg_59_DriverA_Private.c	This file contains the definition of the internal functions that access the hardware registers.
Wdg_59_DriverA_Ram.c	This file contains the global variables used by WDG Component.
Wdg_59_DriverA_Version.c	This file contains the code for checking version of all modules that are interfaced to WDG.
Compiler.h	Provides compiler specific (non-ANSI) keywords. All mappings of keywords, which are not standardized, and/or compiler specific are placed and organized in this compiler specific header.
Compiler_Cfg.h	This file contains the memory and pointer classes.
MemMap.h	This file allows to map variables, constants and code of modules to individual memory sections. Memory mapping can be modified as per ECU specific needs.
Platform_Types.h	This file provides provision for defining platform and compiler dependent types.

File	Details
rh850_Types.h	This file provides macros to perform supervisor mode (SV) write enabled Register ICxxx and IMR register writing using OR/AND/Direct operation.
Dem.h	This file is a stub for DEM component
Dem_Cfg.h	This file contains the stub values for Dem_Cfg.h
Std_Types.h	Provision for Standard types
Det.h	This file is a stub for DET component.
Rte.h	This file is a stub for Rte Component.
Os.h	This file is a stub for OS component
SchM_Wdg_59_DriverA.h	This file is a stub for SchM Component
WdgIf_Types.h	This file contains the common macro definitions and the Type definition for the current status and mode of Watchdog Driver.



## Chapter 9    Generation Tool Guide

For information on the WDG Driver Component Code Generation Tool, please refer “R20UT3729EJ0101-AUTOSAR.pdf” document.



## Chapter 10 Application Programming Interface

This section explains the Data types and APIs provided by the WDG Driver Component to the Upper layers.

### 10.1. Imported Types

This section explains the Data types imported by the WDG Driver Component and lists its dependency on other modules.

#### 10.1.1. Standard Types

In this section all types included from the Std\_Types.h are listed:

- Std\_ReturnType
- Std\_VersionInfoType

#### 10.1.2. Other Module Types

In this section all types included from the WdgIf\_Types.h and Dem.h are listed.

- WdgIf\_ModeType
- WdgIf\_Statustype
- Dem\_EventIdType
- Dem\_EventStatusType

### 10.2. Type Definitions

This section explains the type definitions of WDG Driver Component according to AUTOSAR Specification.

#### 10.2.1. Wdg\_59\_DriverA\_ConfigType

<b>Name:</b>	Wdg_59_DriverA_ConfigType		
<b>Type:</b>	Structure		
<b>Element:</b>	<b>Type</b>	<b>Name</b>	<b>Explanation</b>
	uint32	ulStartOfDbToc	Database start value
	uint32	ulInitTimerCountValue	Trigger counter value
	uint32	ulSlowTimeValue	SLOW mode value of WDTAMD register
	uint32	ulFastTimeValue	FAST mode value of WDTAMD register
	uint8	ucWdtamdSlowValue	WDTAnMD register value for the Slow Mode.

	uint8	ucWdtamdFastValue	WDTAnMD register value for the Fast Mode.
	uint8	ucWdtamdDefaultValue	Watchdog default mode
	WdgIf_ModeType	ddWdtamdDefaultMode	Default mode value configured by the user
<b>Description:</b>	This is the type of the data structure required for initializing the Watchdog Hardware unit.		

### 10.3. Function Definitions

This section explains the APIs provided by the WDG Driver Component.

**Table 10-1 APIs provided by the WDG Driver Component**

Sl.No	API's
1.	Wdg_59_DriverA_Init
2.	Wdg_59_DriverA_SetMode
3.	Wdg_59_DriverA_SetTriggerCondition
4.	Wdg_59_DriverA_GetVersionInfo

#### 10.3.1 Wdg\_59\_DriverA\_Init

Name:	Wdg_59_DriverA_Init		
Prototype:	FUNC(void, WDG_59_DRIVER_A_PUBLIC_CODE) Wdg_59_DriverA_Init (P2CONST(Wdg_59_DriverA_ConfigType, AUTOMATIC, WDG_59_DRIVER_A_APPL_CONST) ConfigPtr)		
Service ID:	0x00		
Sync/Async:	Synchronous		
Reentrancy:	Non-Reentrant		
Parameters In:	Type	Parameter	Value/Range
	Wdg_59_DriverA_ConfigType	ConfigPtr	NA
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	Type	Possible Return Values	
	None	NA	
Description:	This service initialize the Watchdog hardware.		
Configuration Dependency:	None		
Preconditions:	None		



### 10.3.2 Wdg\_59\_DriverA\_SetMode

Name:	Wdg_59_DriverA_SetMode		
Prototype:	FUNC(Std_ReturnType, WDG_59_DRIVER_A_PUBLIC_CODE) Wdg_59_DriverA_SetMode (WdgIf_ModeType Mode)		
Service ID:	0x01		
Sync/Async:	Synchronous		
Reentrancy:	Non-Reentrant		
Parameters In:	Type	Parameter	Value/Range
	WdgIf_ModeType	Mode	1. WDGIF_OFF_MODE 2. WDGIF_SLOW_MODE 3. WDGIF_FAST_MODE
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	Type	Possible Return Values	
	Std_ReturnType	E_OK or E_NOT_OK	
Description:	This service change the mode of the Watchdog timer.		
Configuration Dependency:	None		
Preconditions:	Wdg_59_DriverA_Init must be called before this function.		

### 10.3.3 Wdg\_59\_DriverA\_SetTriggerCondition

Name:	Wdg_59_DriverA_SetTriggerCondition		
Prototype:	FUNC(void, WDG_59_DRIVER_A_PUBLIC_CODE) Wdg_59_DriverA_SetTriggerCondition (uint16 timeout)		
Service ID:	0x03		
Sync/Async:	Synchronous		
Reentrancy:	Non-Reentrant		
Parameters In:	Type	Parameter	Value/Range
	uint16	timeout	NA
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	Type	Possible Return Values	
	None	NA	
Description:	This service is used to initialize the trigger counter.		
Configuration Dependency:	None		
Preconditions:	Wdg_59_DriverA_Init must be called before this function.		

### 10.3.4 Wdg\_59\_DriverA\_GetVersionInfo

Name:	Wdg_59_DriverA_GetVersionInfo		
Prototype:	FUNC(void, WDG_59_DRIVER_A_PUBLIC_CODE) Wdg_59_DriverA_GetVersionInfo(P2VAR (Std_VersionInfoType, AUTOMATIC, WDG_59_DRIVER_A_APPL_CONST) versioninfo)		
Service ID:	0x04		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	Type	Parameter	Value/Range
	None	NA	NA
Parameters InOut:	None	NA	NA
Parameters out:	Std_VersionInfoType	versioninfo	NA
Return Value:	Type	Possible Return Values	
	None	NA	
Description:	This service returns the version information of this module.		
Configuration Dependency:	None		
Preconditions:	None		

## Chapter 11 Development And Production Errors

In this section the development errors that are reported by the WDG Driver Component are tabulated. The development errors will be reported only when the pre compiler option WdgDevErrorDetect is enabled in the configuration.

### 11.1. WDG Driver Component Development Errors

The following table contains the DET errors that are reported by WDG Driver Component. These errors are reported to Development Error Tracer Module when the WDG Driver Component APIs are invoked with wrong input parameters or without initialization of the driver.

**Table 11-1 DET Errors of WDG Driver Component**

<b>Sl. No.</b>	<b>1</b>
Error Code	WDG_59_DRIVER_A_E_PARAM_CONFIG
Related API(s)	Wdg_59_DriverA_Init
Source of Error	When the API service is called with a configuration set which is not within the allowed boundaries.
<b>Sl. No.</b>	<b>2</b>
Error Code	WDG_59_DRIVER_A_E_PARAM_MODE
Related API(s)	Wdg_59_DriverA_SetMode
Source of Error	When the API service is called the Driver is not possible to change the mode.
<b>Sl. No.</b>	<b>3</b>
Error Code	WDG_59_DRIVER_A_E_DRIVER_STATE
Related API(s)	Wdg_59_DriverA_SetMode and Wdg_59_DriverA_SetTriggerCondition, WDG_59_DRIVER_A_TRIGGERFUNCTION_ISR.
Source of Error	If the API service is called when the driver state is not in idle state.
<b>Sl. No.</b>	<b>4</b>
Error Code	WDG_59_DRIVER_A_E_INVALID_DATABASE
Related API(s)	Wdg_59_DriverA_Init
Source of Error	When the API service is called with wrong database.
<b>Sl. No.</b>	<b>5</b>
Error Code	WDG_59_DRIVER_A_E_PARAM_TIMEOUT
Related API(s)	Wdg_59_DriverA_SetTriggerCondition
Source of Error	When the API service Wdg_59_DriverA_SetTriggerCondition is called with timeout value greater maximum timeout value (WdgMaxTimeout).
<b>Sl. No.</b>	<b>6</b>
Error Code	WDG_59_DRIVER_A_E_PARAM_POINTER
Related API(s)	Wdg_59_DriverA_Init, Wdg_59_DriverA_GetVersionInfo
Source of Error	API service is called with NULL Pointer.

## 11.2. WDG Driver Component Production Errors

The following table contains the DEM errors that are reported by WDG Component

**Table 11-2 DEM Errors of WDG Driver Component**

<b>Sl. No.</b>	<b>1</b>
Error Code	WDG_59_DRIVER_A_E_DISABLE_REJECTED
Related API(s)	Wdg_59_DriverA_SetMode
Source of Error	If error during mode switch failed, the above error is reported to DEM
<b>Sl. No.</b>	<b>2</b>
Error Code	WDG_59_DRIVER_A_E_MODE_FAILED
Related API(s)	Wdg_59_DriverA_SetMode
Source of Error	When switching between the modes is failed above error is reported to DEM.
<b>Sl. No.</b>	<b>3</b>
Error Code	WDG_59_DRIVER_A_E_REG_WRITE_VERIFY
Related API(s)	In Wdg_59_DriverA_Init and Wdg_59_DriverA_SetMode API write verify failure report DEM error.
Source of Error	Write verify failure is caused whenever a register is written and the register is not updated with the written value then this error is reported.
<b>Sl. No.</b>	<b>4</b>
Error Code	WDG_59_DRIVER_A_E_INT_INCONSISTENT
Related API(s)	WDG_59_DRIVER_A_TRIGGERFUNCTION_ISR API
Source of Error	When ISR is invoked from incorrect interrupt source, then this error is reported.

## Chapter 12 Memory Organization

Following picture depicts a typical memory organization, which must be met for proper functioning of WDG Component software.

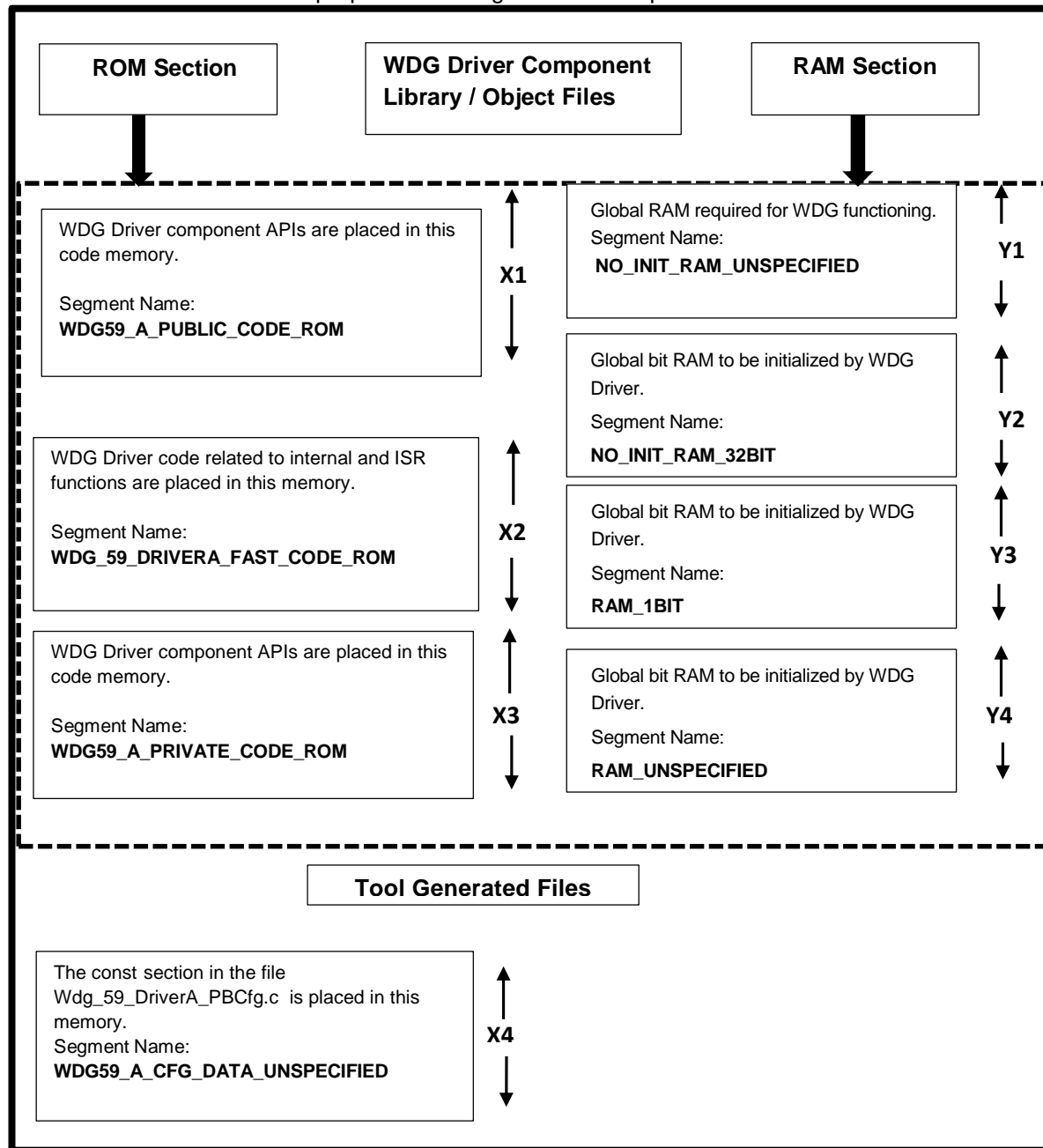


Figure 12-1 Memory Organization Of WDG Driver Component

**ROM Section (X1, X2, X3 and X4):**

**WDG59\_A\_PUBLIC\_CODE\_ROM (X1):** API(s) of WDG Driver Component, which can be located in code memory.

**WDG\_59\_DRIVER\_A\_FAST\_CODE\_ROM (X2):** Internal and ISR functions of WDG Driver Component code are placed in this code memory.

**WDG59\_PRIVATE\_CODE\_ROM (X3):** Internal functions of WDG Driver Component code that can be located in code memory.

**WDG59\_A\_CFG\_DATA\_UNSPECIFIED (X4):** This section consists of WDG Component database generated by the Watchdog Driver Generation Tool and the constant structures used in AUTOSAR Renesas WDG Driver Component. This can be located in code memory.

**RAM Section (Y1, Y2, Y3 and Y4):**

**NO\_INIT\_RAM\_UNSPECIFIED (Y1):** This section consists of the global RAM variables that are used internally by WDG Component and other software components. The specific sections of respective software components will be merged into this RAM section accordingly.

**NO\_INIT\_RAM\_32BIT (Y2):** This section consists of the global RAM variables of 32-bit size that are used internally by WDG Driver Component. This can be located in data memory.

**RAM\_1BIT (Y3):** This section consists of the global RAM variables of 1-bit size that are initialized by start-up code and used internally by WDG software component and other software components. The specific sections of respective software components will be merged into this RAM section accordingly.

**RAM\_UNSPECIFIED (Y4):** This section consists of the global RAM variables that are initialized by start-up code and used internally by WDG Component and other software components. This can be located in data memory.

- X1, X2, X3, Y1, Y2, Y3 and Y4 pertain to only WDG Component and do not include memory occupied by Wdg\_59\_DriverA\_PBCfg.c file generated by Watchdog Driver Generation Tool.
- User must ensure that none of the memory areas overlap with each other. Even 'debug' information should not overlap.

## Chapter 13 P1M Specific Information

P1M supports following devices:

- R7F701304
- R7F701305
- R7F701310
- R7F701311
- R7F701312
- R7F701313
- R7F701314
- R7F701315
- R7F701318
- R7F701319
- R7F701320
- R7F701321
- R7F701322
- R7F701323

### 13.1. Interaction Between The User And WDG Driver Component

#### 13.1.1. ISR Function Mapping Interrupt Vector Table

The table below provides the list of handler addresses corresponding to the hardware unit ISR(s) in WDG Driver Component. The user should configure the ISR functions mentioned below:

**Table 13-1 Interrupt Vector Table**

Interrupt Source	Name of the ISR Function
Autosar R4.0 only EI level mask able interrupt	WDG_59_DRIVER_A_TRIGGERFUNCTION_ISR
	WDG_59_DRIVER_A_TRIGGERFUNCTION_CAT2_ISR

**Note:** The functions with “INTERRUPT” as pilot tag, provides an indication to the compiler that the function following this tag is an interrupt function type. The tag name can vary according to the compiler. User should take care of the tag name with respect to compiler used.

#### 13.1.2. Translation Header File

P1x\_translation.h supports following devices:

- R7F701304
- R7F701305
- R7F701310
- R7F701311
- R7F701312
- R7F701313

- R7F701314
- R7F701315
- R7F701318
- R7F701319
- R7F701320
- R7F701321
- R7F701322
- R7F701323

### 13.1.3. Parameter Definition File

Parameter definition files support information for P1M

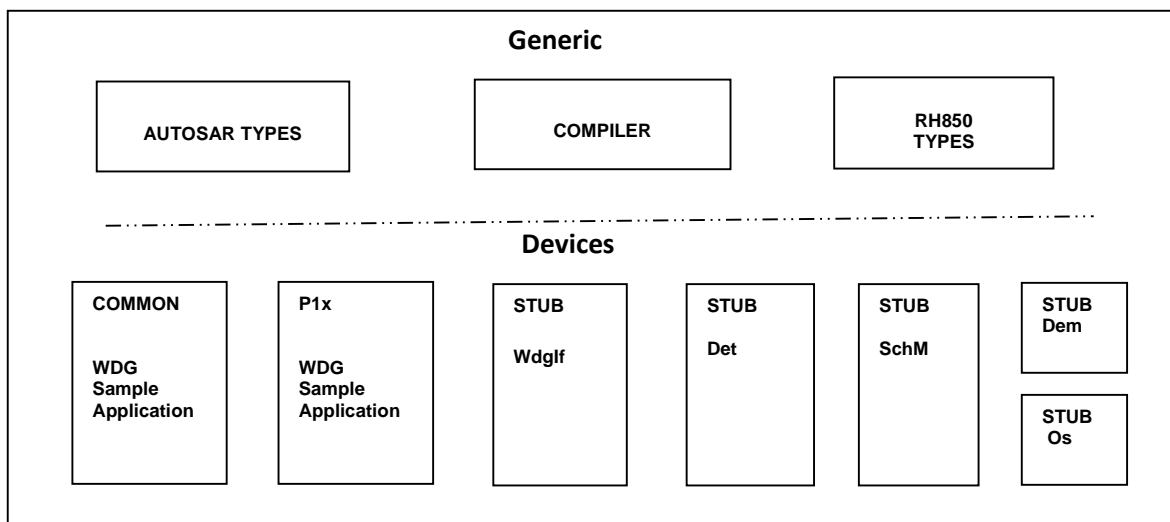
**Table 13-2 PDF information for P1M**

PDF files	Devices supported
R403_WDG_P1M_04_05_10_to_15_18_to_23.arxml	701304, 701305, 701310, 701311, 701312, 701313, 701314, 701315, 701318, 701319, 701320, 701321, 701322, 701323

## 13.2. Sample Application

### 13.2.1 Sample Application Structure

The Sample Application is provided as reference to the user to understand the method in which the WDG APIs can be invoked from the application.



**Figure 13-1 Overview Of WDG Driver Sample Application**

The Sample Application of the P1M is available in the path  
X1X\P1x\modules\wdg\sample\_application

The Sample Application consists of the following folder structure

X1X\P1x\modules\wdg\definition\<AUTOSAR\_version>\<SubVariant>



```

\ R403_WDG_P1M_04_05_10_to_15_18_to_23.arxml
X1X\P1x\modules\wdg\sample_application\<SubVariant>\<AUTOSAR_version>
    \src\Wdg_59_DriverA_PBcfg.c
    \inc\Wdg_59_DriverA_Cfg.h
    \inc\Wdg_59_DriverA_Cbk.h

\config\App_WDG_P1M_701304_Sample.arxml
\config\App_WDG_P1M_701305_Sample.arxml
\config\App_WDG_P1M_701310_Sample.arxml
\config\App_WDG_P1M_701311_Sample.arxml
\config\App_WDG_P1M_701312_Sample.arxml
\config\App_WDG_P1M_701313_Sample.arxml
\config\App_WDG_P1M_701314_Sample.arxml
\config\App_WDG_P1M_701315_Sample.arxml
\config\App_WDG_P1M_701318_Sample.arxml
\config\App_WDG_P1M_701319_Sample.arxml
\config\App_WDG_P1M_701320_Sample.arxml
\config\App_WDG_P1M_701321_Sample.arxml
\config\App_WDG_P1M_701322_Sample.arxml
\config\App_WDG_P1M_701323_Sample.arxml

```

In the Sample Application all the WDG APIs are invoked in the following sequence

When DriverA (WDTA0) is selected:

- The API Wdg\_59\_DriverA\_GetVersionInfo is invoked to get the version of the WDG Driver module with a variable of Std\_VersionInfoType, after the call of this API the passed parameter will get updated with the WDG Driver version details.
- The API Wdg\_59\_DriverA\_Init is invoked with a valid database address for the proper initialization of the WDG Driver, all the WDG Driver control registers and RAM variables will get initialized after this API is called.
- The API Wdg\_59\_DriverA\_SetMode is invoked with the mode which needs to be set, this API changes the mode of the Watchdog.

The API Wdg\_59\_DriverA\_SetTriggerCondition initializes the trigger counter global variable with timeout value divided by either usSlowTimeValue or usFastTimeValue based on the current mode of Watchdog'.

## 13.2.2 Building Sample Application

### 13.2.2.1 Configuration Example

This section contains the typical configuration which is used for measuring RAM/ROM consumption, stack depth and throughput details.

#### Configuration Details:

App\_WDG\_<SubVariant>\_<Device\_Number>\_Sample.arxml

The <Device Number> indicates the device to be compiled, which can be 701304, 701305, 701310, 701311, 701312, 701313, 701314, 701315, 701318, 701319, 701320, 701321, 701322 and 701323.

**Remark** In this typical configuration, all the conversion modes available for WDG Driver Component are configured so that each API's throughput analysis could be performed. Throughput is measured by toggling a port pin before invoking the API and again toggling the same port pin after the execution of the API.

Following Opbyte setting shall be followed:

If Variable activation code is enabled, Opbyte value = 0x71DF3FFB.

If Variable activation code is disabled, Opbyte value = 0x719F3FFB.

In debug mode unmask the reset using GHS command "target pinmask".

### 13.2.2.2 Debugging The Sample Application

GNU Make utility version 3.81 or above must be installed and available in the path as defined by the environment user variable "GNUMAKE" to complete the build process using the delivered sample files.

- Open a Command window and change the current working directory to "make" directory present as mentioned in below path:

"X1X\P1x\common\_family\make\<Compiler>"

Now execute the batch file SampleApp.bat with following parameters

SampleApp.bat Wdg < AUTOSAR\_version> <Device\_Name>

- After this, the tool output files will be generated with the configuration as mentioned in App\_WDG\_<SubVariant>\_<Device\_Number>\_Sample.arxml file available in the path:

"X1X\P1x\modules\wdg\sample\_application\<SubVariant>\<AUTOSAR\_version>\config".

- After this, all the object files, map file and the executable file App\_WDG\_P1M\_Sample.out will be available in the output folder

("X1X\P1x\modules\wdg\sample\_application\<SubVariant>\obj\<Compiler>").

(Note: For example compiler can be ghs.)

- The executable can be loaded into the debugger and the sample application can be executed.

**Remark** Executable files with "\*.out" extension can be downloaded into the target hardware with the help of Green Hills debugger.

- If any configuration changes (only post-build) are made to the ECU Configuration Description files  
`"X1X\P1x\modules\wdg\<SubVariant>\<AUTOSAR_version>  
\config\App_WDG_<SubVariant>_<Device_name>_Sample.arxml"`
- The database alone can be generated by using the following commands.  
`make -f App_WDG_<SubVariant>_Sample.mak generate_wdg_config`  
`make -f App_WDG_<SubVariant>_Sample.mak`  
`App_WDG_<SubVariant>_Sample.s37`
- After this, a flashable Motorola S-Record file  
`App_WDG_<SubVariant>_Sample.s37` is available in the output folder.

**Notes:**

1. <Compiler> can be ghs.
2. <Device\_name> can be 701304, 701305, 701310, 701311, 701312, 701313, 701314, 701315, 701318, 701319, 701320, 701321, 701322 and 701323.
3. <AUTOSAR\_version> can be 4.0.3.
4. <SubVariant> can be P1M

## 13.3. Memory and Throughput for R4.0.3

### 13.3.1 ROM/RAM Usage

The details of memory usage for the typical configuration provided in Section 13.2.2.1 *Configuration Example* are provided in this section.

**Table 13-3 ROM/RAM Details Without DET**

Sl. No	ROM/RAM	Segment Name	Size in bytes for 701318
1.	ROM	WDG59_A_PUBLIC_CODE_ROM	374
		WDG_59_DRIVER_A_FAST_CODE_ROM	70
		WDG59_A_CFG_DATA_UNSPECIFIED	20
		WDG59_A_PRIVATE_CODE_ROM	0
2.	RAM	NO_INIT_RAM_UNSPECIFIED	5
		RAM_8BIT	3
		NO_INIT_RAM_32BIT	4

The details of memory usage for the typical configuration, with DET enabled and all other configurations as provided in 13.2.2.1 *Configuration Example* are provided in this section.

Table 13-4 ROM/RAM Details With DET

	ROM/RAM	Segment Name	Size in bytes for 701318
1.	ROM	WDG59_A_PUBLIC_CODE_ROM	642
		WDG_59_DRIVER_A_FAST_CODE_ROM	244
		WDG59_A_CFG_DATA_UNSPECIFIED	20
		WDG59_A_PRIVATE_CODE_ROM	136
2.	RAM	NO_INIT_RAM_UNSPECIFIED	5
		RAM_8BIT	3
		NO_INIT_RAM_32BIT	4

### 13.3.2 Stack Depth

The worst-case stack depth for WDG Driver Component is 20 bytes for the typical configuration provided in Section 13.2.2.1 *Configuration Example*.

### 13.3.3 Throughput Details

The throughput details of the APIs for the configuration mentioned in the Section 13.2.2.1 *Configuration Example* are listed here. The clock frequency used to measure the throughput is 80MHz for all APIs.

Table 13-5 Throughput Details Of The APIs

Sl. No.	API Name	Throughput in microsecond for 701318	Remarks
1.	Wdg_59_DriverA_Init	1.8	Timing is measured with default mode as WDGIF_SLOW_MODE
2.	Wdg_59_DriverA_SetMode	0.275	Timing is measured with default mode as WDGIF_SLOW_MODE
3.	Wdg_59_DriverA_SetTriggerCondition	0.65	-
4.	Wdg_59_DriverA_GetVersionInfo	0.137	-
5.	WDG_59_DRIVER_A_TRIGGER_FUNCTION_ISR	0.975	-

## Chapter 14 Release Details

WDG Driver Software R4.0.3  
Version: 1.0.12



## Revision History

Sl.No	Description	Version	Date
1.	Initial Version	1.0.0	18-Oct-2013
2	Following changes are made: 1. In chapter 2, Reference Documents and Device manual version changed. 2. 100 pin Device names are added. 3. Compiler version and options are changed. 4. ROM/RAM table is updated for 100 pin device. 5. Sample application folder path is updated for 100 pin device.	1.0.1	04-Feb-2014
3	Following changes are made: 1. In chapter 2, Reference Documents and Device manual version changed. 2. In chapter 13, translation header file that supports P1M devices are listed. 3. In chapter 13, sample application structure is modified according to P1M supporting devices. 4. In chapter 13, 13.1.2 ISR Function Mapping Interrupt Vector Table and 13.1.3 Parameter Definition File are added. 5. In chapter 13, Throughput details and ROM/RAM Usage are added.	1.0.2	26-Sep-2014
4.	Headers are corrected in chapter 10 and 11.	1.0.3	21-Nov-2014
5.	Following changes are made: 1. Updated Chapter 4.1 to add notes. 2. Updated Chapter 13 to add new device support. 3. Removed sections for Compiler, Linker and Assembler. 4. Updated Chapter 13.3 with memory and throughput details.	1.0.4	28-Apr-2015
6.	Following changes are made: 1. Updated Chapter 8 to change heading for stub files and addition of stub file details. 2. Updated Chapter 6 'Register Details' to add register access type. 3. Updated Section 4.6 'Deviation List' to add Autosar Deviation list. 4. Copyright information updated. 5. Updated chapter 4 'Forethoughts' to add details on register access. 6. Updated Chapter 13.3 with memory and throughput details. 7. Updated Chapter 12 to change memory section of tool generated file. 8. Updated Chapter 14 'Release Details' to update WDG Driver Software version. 9. Updated section 4.1 to add notes. 10. Updated Chapter 3 and Chapter 9 to add R-Numbered Tool User manual name. 11. Added R Number.	1.0.5	29-Mar-2016

SI.No	Description	Version	Date
7.	<p>Following changes are made:</p> <ol style="list-style-type: none"> <li>1. Section 3.1.1 is updated to add Wdg_59_DriverA_RegWrite.h and remove Wdg_59_DriverA_RegReadBack.h file</li> <li>2. Section 4.4 is updated for State diagram of WDG Driver.</li> <li>3. Section 4.6 Deviation List is updated.</li> <li>4. Section 4.8 Register Read-Back is changed to Register Write-verify.</li> <li>5. Chapter 8 is updated to add Wdg_59_DriverA_RegWrite.h and remove Wdg_59_DriverA_RegReadBack.h file. Also add Wdg_59_DriverA_Cbk.h under generated file.</li> <li>6. Table 8-1 is updated for Wdg_59_DriverA_RegWrite and Wdg_59_DriverA_Cbk.h and stub files.</li> <li>7. Section 10.2.1 is updated to remove element usDefaultTimeValue.</li> <li>8. Table 11.2 is updated to add new DEM errors.</li> <li>9. Removed *.html and *.one files from section 13.2.</li> <li>10. Chapter 6 Register Details is Updated.</li> <li>11. Chapter 12 Memory section is updated.</li> <li>12. Section 13.1.1 ISR Function Mapping Interrupt Vector Table is updated with note.</li> <li>13. Added a 'Note' below the table 'Supervisor mode and User mode details.</li> <li>14. Table 11.1 is updated to add the DET error WDG_59_DRIVERA_E_PARAM_POINTER</li> <li>15. Chapter 14 Release details is updated with WDG Driver.</li> <li>16. Chapter 3 and 9 are updated with R-Numbered Tool User manual name.</li> <li>17. Version of R-number is updated at the end of document.</li> <li>18. Section 13.3 Memory and throughput Values are updated.</li> <li>19. Section 4.2 and 4.3 is updated with critical section information.</li> <li>20. Section 4.6 General is updated.</li> </ol>	1.0.6	13-Jul-2016
8.	<p>Following changes are made</p> <ol style="list-style-type: none"> <li>1. Section 13.3 is updated for Memory and Throughput details.</li> <li>2. Chapter 12 is updated to remove NO_INIT_RAM_16BIT and add NO_INIT_RAM_32BIT.</li> </ol>	1.0.7	21-Oct-2016
9	<p>Following changes are made</p> <ol style="list-style-type: none"> <li>1. Abbreviation list is updated.</li> <li>2. Section 10.3 Function Definitions is updated.</li> <li>3. Chapter 14 release details version is updated.</li> <li>4. Chapter 3 and 9 are updated with R-Numbered Tool User manual name.</li> <li>5. Version of R-number is updated at the end of document.</li> <li>6. Notice and copyright information is updated.</li> <li>7. Table of contents updated.</li> <li>8. Table 4.1 WDG Driver Deviation List is updated to add not implemented requirement 'WDG163'.</li> <li>9. Chapter 10.2.1 Wdg_59_DriverA_ConfigType is updated.</li> <li>10. Chapter 12 Memory Organization is updated.</li> </ol>	1.0.8	15-Feb-2017





---

**AUTOSAR MCAL R4.0.3 User's Manual**  
**WDG Driver Component Ver.1.0.8**  
**Embedded User's Manual**

Publication Date: Rev.1.01, February 15, 2017

Published by: Renesas Electronics Corporation

---



Renesas Electronics Corporation

<http://www.renesas.com>

#### SALES OFFICES

Refer to "http://www.renesas.com/" for the latest and detailed information.

**Renesas Electronics America Inc.**

2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.  
Tel: +1-408-588-6000, Fax: +1-408-588-6130

**Renesas Electronics Canada Limited**

9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3  
Tel: +1-905-237-2004

**Renesas Electronics Europe Limited**

Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K.  
Tel: +44-1628-585-100, Fax: +44-1628-585-900

**Renesas Electronics Europe GmbH**

Arcadiastrasse 10, 40472 Düsseldorf, Germany  
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

**Renesas Electronics (China) Co., Ltd.**

Room 1709, Quantum Plaza, No.27 ZhichunLu Haidian District, Beijing 100191, P.R.China  
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

**Renesas Electronics (Shanghai) Co., Ltd.**

Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333  
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

**Renesas Electronics Hong Kong Limited**

Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong  
Tel: +852-2265-6688, Fax: +852-2886-9022

**Renesas Electronics Taiwan Co., Ltd.**

13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan  
Tel: +886-2-8175-9600, Fax: +886-2-8175-9670

**Renesas Electronics Singapore Pte. Ltd.**

80 Bendemeer Road, Unit #06-02 Hyllux Innovation Centre, Singapore 339949  
Tel: +65-6213-0200, Fax: +65-6213-0300

**Renesas Electronics Malaysia Sdn.Bhd.**

Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jin Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia  
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

**Renesas Electronics India Pvt. Ltd.**

No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India  
Tel: +91-80-67208700, Fax: +91-80-67208777

**Renesas Electronics Korea Co., Ltd.**

12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea  
Tel: +82-2-558-3737, Fax: +82-2-558-5141

# AUTOSAR MCAL R4.0.3 User's Manual