User's Manual

AUTOSAR MCAL R4.0.3

User's Manual

WDG Driver Component Ver.1.0.7

Embedded User's Manual

Target Device:
RH850/P1x

Renesas Electronics
www.renesas.com

Rev.1.00 Oct 2016

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of 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. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may 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.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "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.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti- crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its 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 be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, 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. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be 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, or if you have any other inquiries.

(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
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
ID/Id	Identifier
ISR	Interrupt Service Routine
LIN	Local Interconnect Network
MCAL	Microcontroller Abstraction Layer
MCU	MicroController Unit
PWM	Pulse Width Modulation
RAM	Random Access Memory
ROM	Read Only Memory
SCI	Serial Communication Interface
SPI	Serial Peripheral Interface
WDG/wdg	WatchDog
WDT	WatchDog Timer
WDGIF	WatchDog Interface
CDD	Complex Device Driver

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

List of Figures

List of Tables

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:

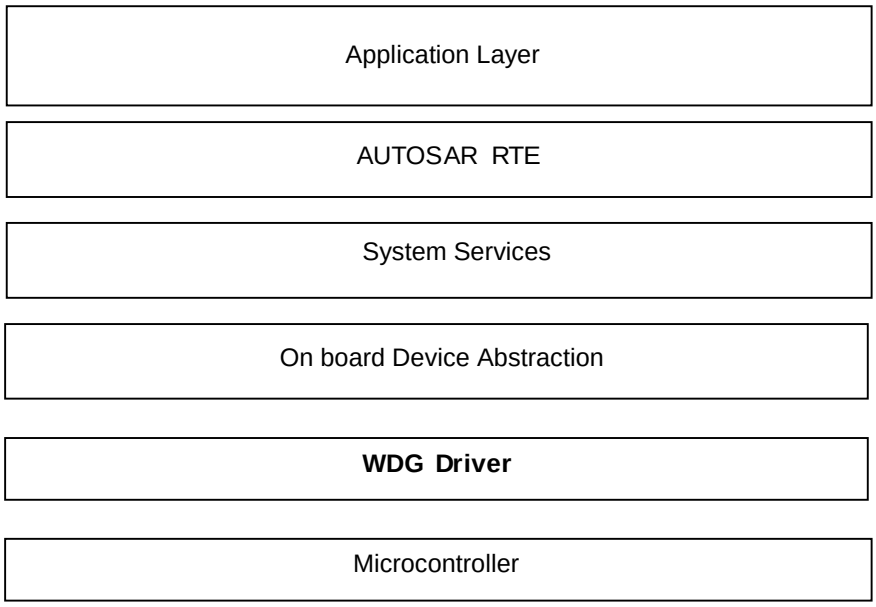

Figure 1-1 System Overview Of AUTOSAR Architecture

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:

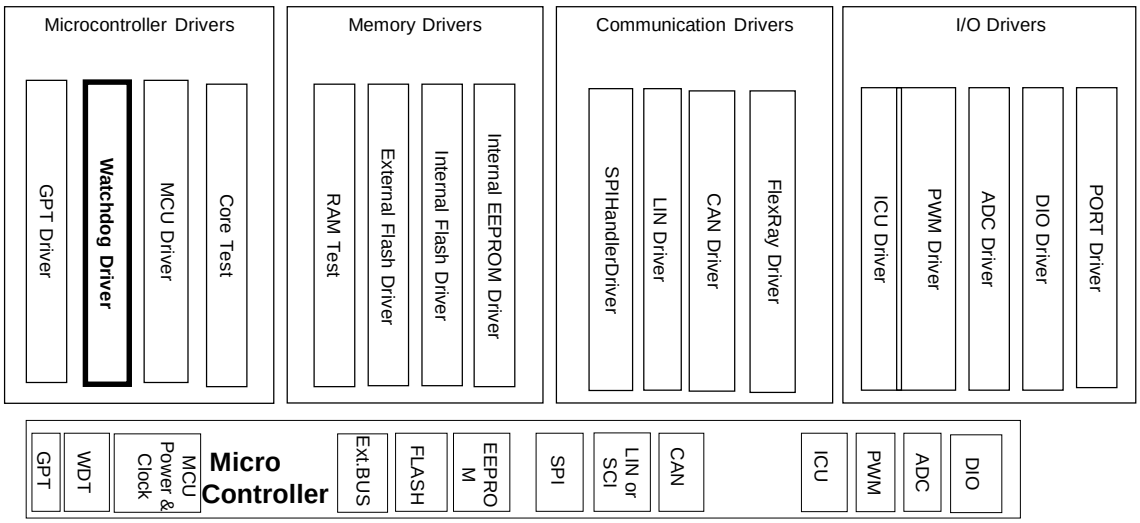

Figure 1-2 System Overview Of The WDG Driver In AUTOSAR MCAL Layer

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 (http://www.autosar.org/bugzilla) Note: AUTOSAR BUGZILLA is a database, which contains concerns raised against information present in AUTOSAR Specifications.	-
3.	r01uh0436ej0111_rh850p1x.pdf	1.11
4.	AUTOSAR_SWS_CompilerAbstraction.pdf	3.2.0
5.	AUTOSAR_SWS_MemoryMapping.pdf	1.4.0
6.	AUTOSAR_SWS_PlatformTypes.pdf	2.5.0
7.	AUTOSAR_BSW_MakefileInterface.pdf	0.3

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 "R20UT3729EJ0100-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\<Complier>
\App_WDG_P1M_Sample.mak

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

(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_DRIVERA_DEV_ERROR_DETECT is enabled. Application should ensure that the right parameters are passed while invoking the APIs when WDG_59_DRIVERA_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

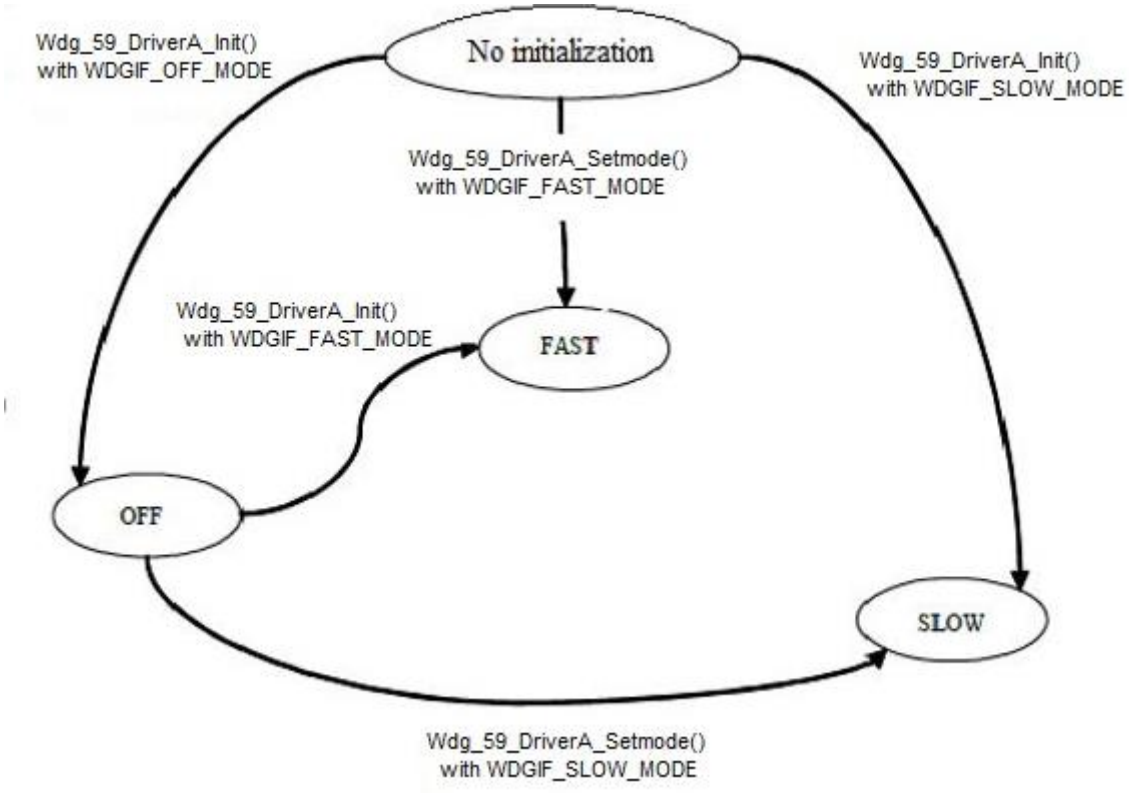

Figure 4-1 State Diagram of WDG

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.

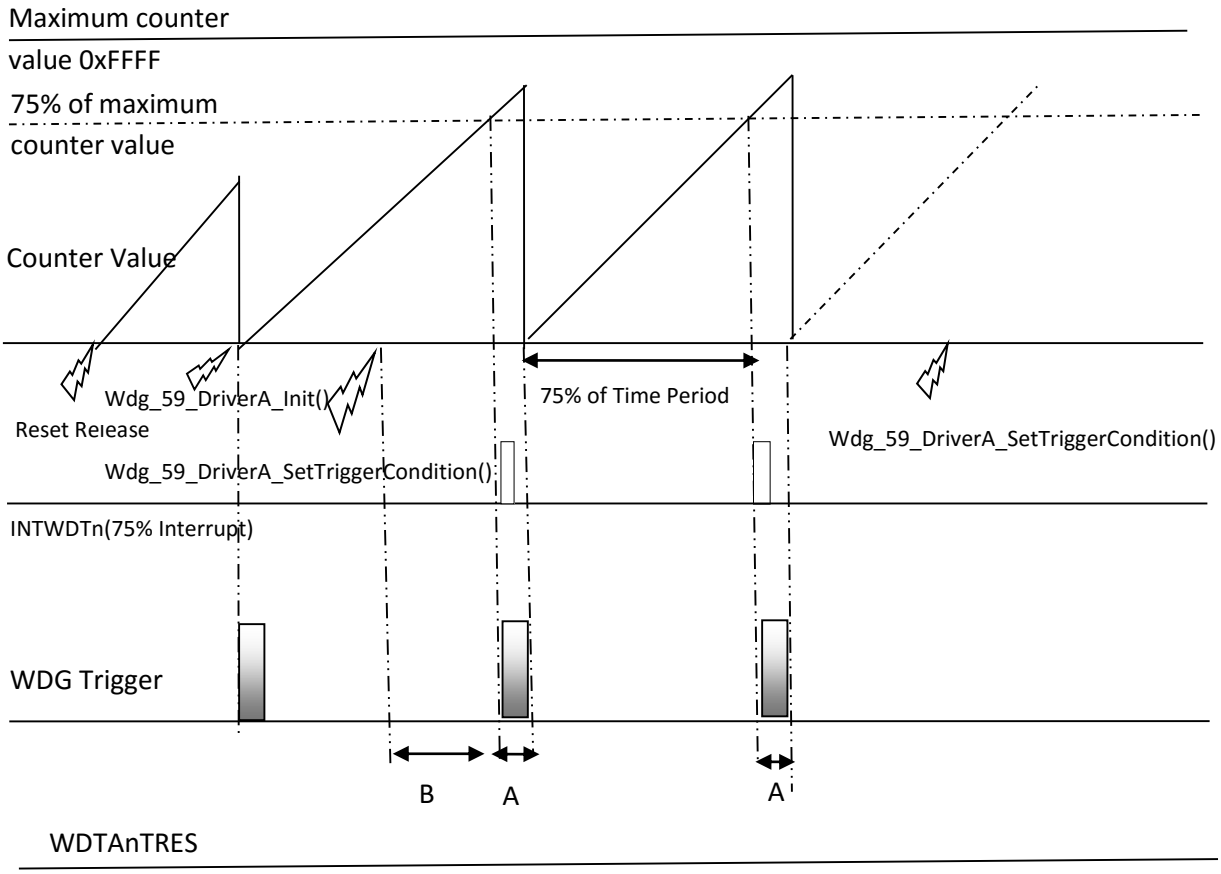

Figure 4-2 WDG behavior during Data exchange with hardware

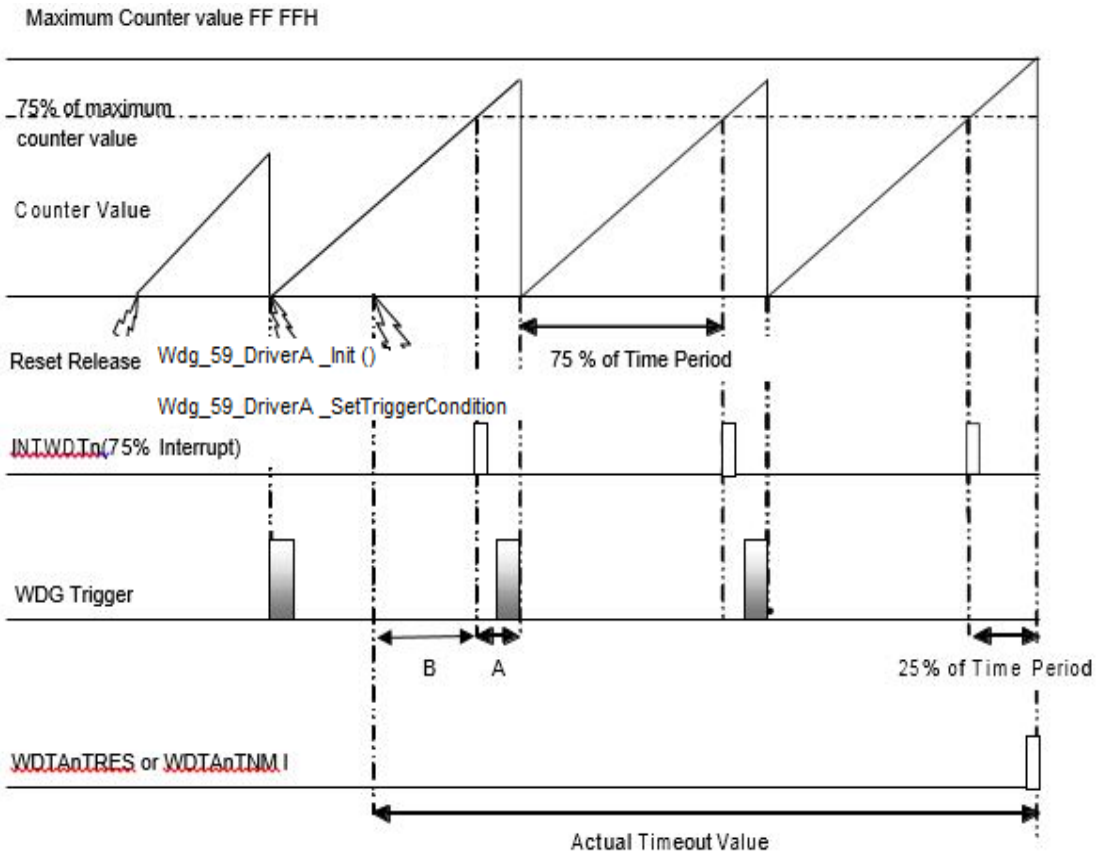

Figure 4-3 WDG behavior when Wdg_SetTriggerCondition is called

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.	-

Table 4-2 AUTOSAR Deviation List

Sl.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

Sl.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:

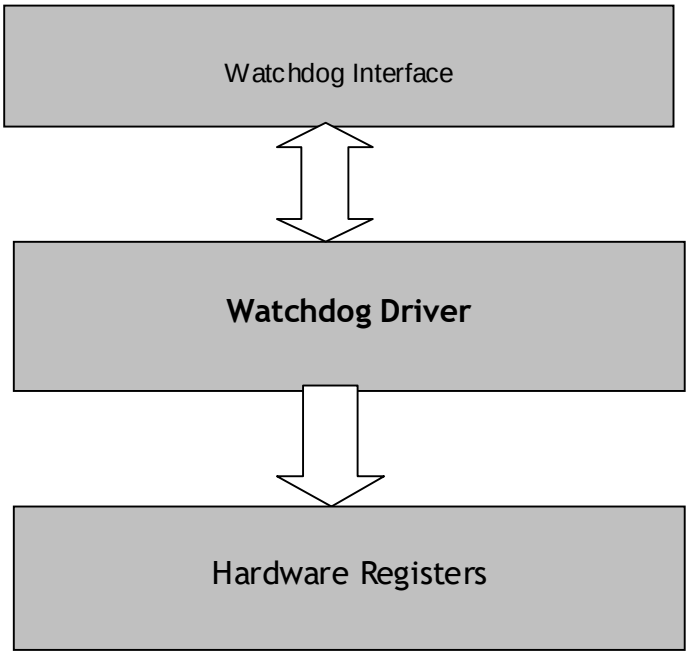

Figure 5-1 Watch Driver And Watchdog Interface Architecture

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:

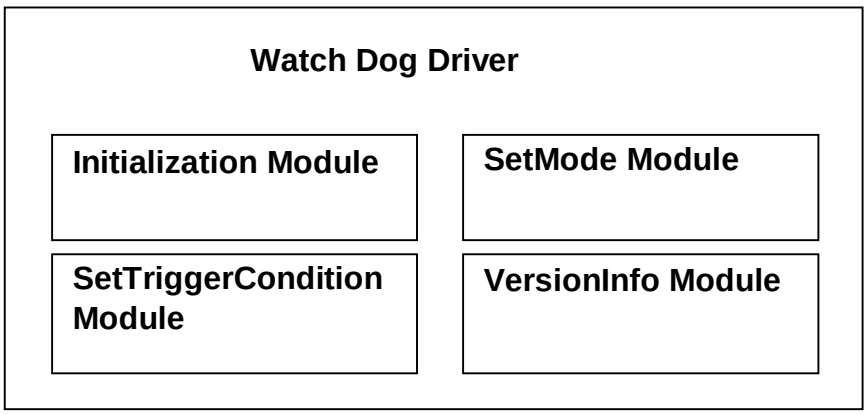

Figure 5-2 Basic Architecture Of WDG Component

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_DRIVERA_INTWDTIMR_MASK
	WDTAnMD	WdgDefaultMode	rw	ucWdtamdDefaultValue.
	WDTAnEVAC	WdgVaryingActivationCodeMode	rw	WDG_59_DRIVERA_RESTART, WDG_59_DRIVERA_WDTAREF_ADDRESS
	WDTAnWDTE	WdgVaryingActivationCodeMode	rw	WDG_59_DRIVERA_WDTAWDTE_MASK, WDG_59_DRIVERA_RESTART
	WDTAnREF	-	r	WDG_59_DRIVERA_WDTAREF_ADDRESS
Wdg_59_DriverA_SetMode	WDTAnMD	-	rw	ucWdtamdSlowValue, ucWdtamdFastValue.
	WDTAnEVAC	WdgVaryingActivationCodeMode	rw	WDG_59_DRIVERA_RESTART, WDG_59_DRIVERA_WDTAREF
	WDTAnWDTE	WdgVaryingActivationCodeMode	rw	WDG_59_DRIVERA_WDTAWDTE_MASK, WDG_59_DRIVERA_RESTART
	WDTAnREF	-	r	WDG_59_DRIVERA_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
- WdgIf_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_PBTypes.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 “R20UT3729EJ0100-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

Name:	Wdg_59_DriverA_ConfigType		
Type:	Structure		
Element:	**Type**	**Name**	**Explanation**
	unit32	ulStartOfDbToc	Database start value
	uint16	usInitTimerCountValue	Trigger counter value
	uint16	usSlowTimeValue	SLOW mode value of WDTAMD register
	uint16	usFastTimeValue	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
Description:	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

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

Sl. No.	**1**
Error Code	WDG_59_DRIVERA_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.
Sl. No.	**2**
Error Code	WDG_59_DRIVERA_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.
Sl. No.	**3**
Error Code	WDG_59_DRIVERA_E_DRIVER_STATE
Related API(s)	Wdg_59_DriverA_SetMode, Wdg_59_DriverA_Trigger and Wdg_59_DriverA_SetTriggerCondition, WDG_59_DRIVERA_TRIGGERFUNCTION_ISR.
Source of Error	If the API service is called when the driver state is not in idle state.
Sl. No.	**4**
Error Code	WDG_59_DRIVERA_E_INVALID_DATABASE
Related API(s)	Wdg_59_DriverA_Init
Source of Error	When the API service is called with wrong database.
Sl. No.	**5**
Error Code	WDG_59_DRIVERA_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).
Sl. No.	**6**
Error Code	WDG_59_DRIVERA_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

Sl. No.	**1**
Error Code	WDG_59_DRIVERA_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
Sl. No.	**2**
Error Code	WDG_59_DRIVERA_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.
Sl. No.	**3**
Error Code	WDG_59_DRIVERA_E_REG_WRITE_VERIFY
Related API(s)	In Wdg_59_DriverA_Init, Wdg_59_DriverA_SetMode and Wdg_59_DriverA_Trigger 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.
Sl. No.	**4**
Error Code	WDG_59_DRIVERA_E_INT_INCONSISTENT
Related API(s)	WDG_59_DRIVERA_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.

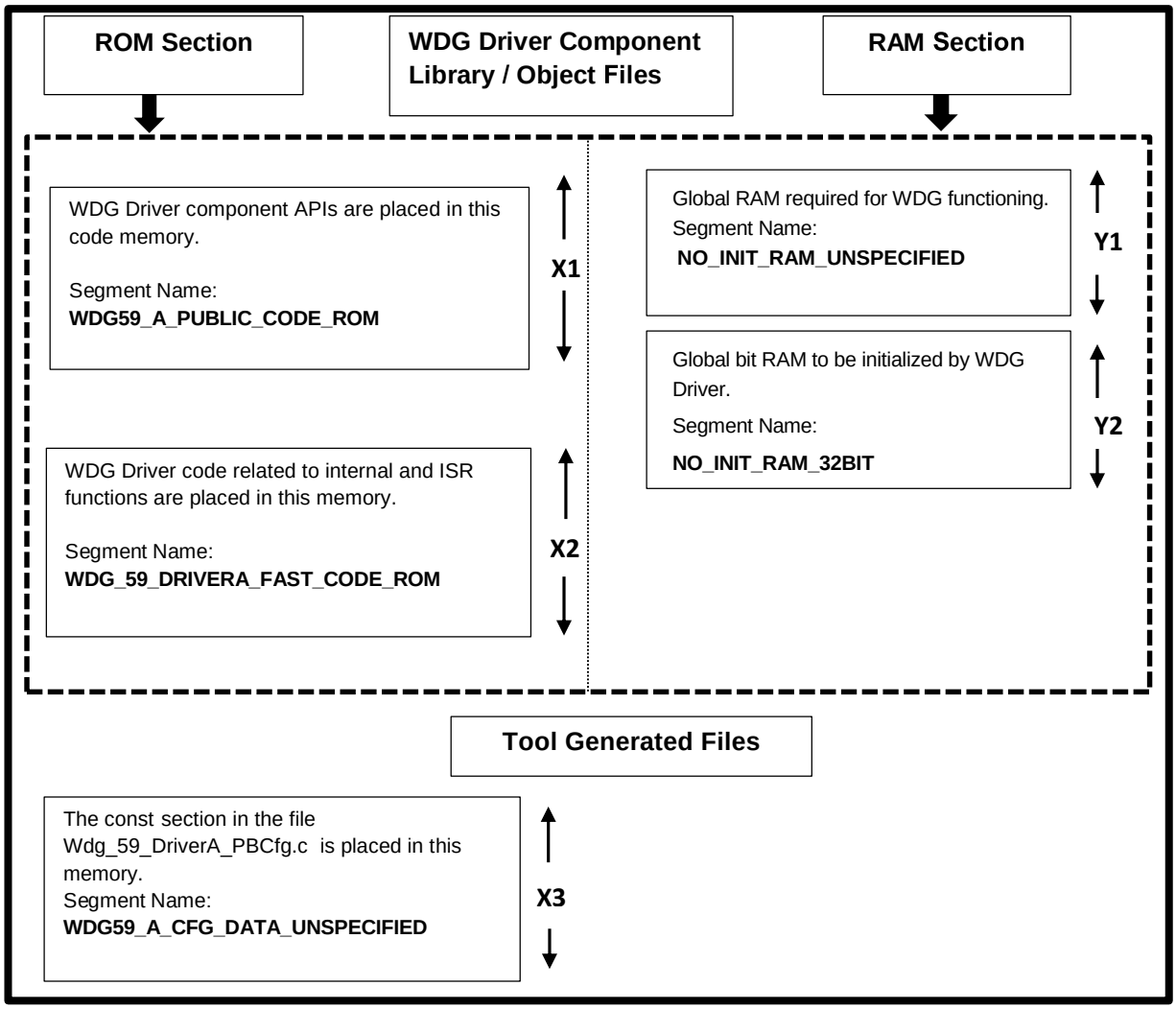

Figure 12-1 Memory Organization Of WDG Driver Component

ROM Section (X1, X2 and X3):

WDG59_A_PUBLIC_CODE_ROM (X1): API(s) of WDG Driver Component, which can be located in code memory.

WDG_59_DRIVERA_FAST_CODE_ROM (X2): Internal and ISR functions of WDG Driver Component code are placed in this code memory.

WDG59_A_CFG_DATA_UNSPECIFIED (X3): 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 and Y2):

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.

- X1, X2, X3, Y1 and Y2 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_DRIVERA_TRIGGERFUNCTION_ISR
	WDG_59_DRIVERA_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.

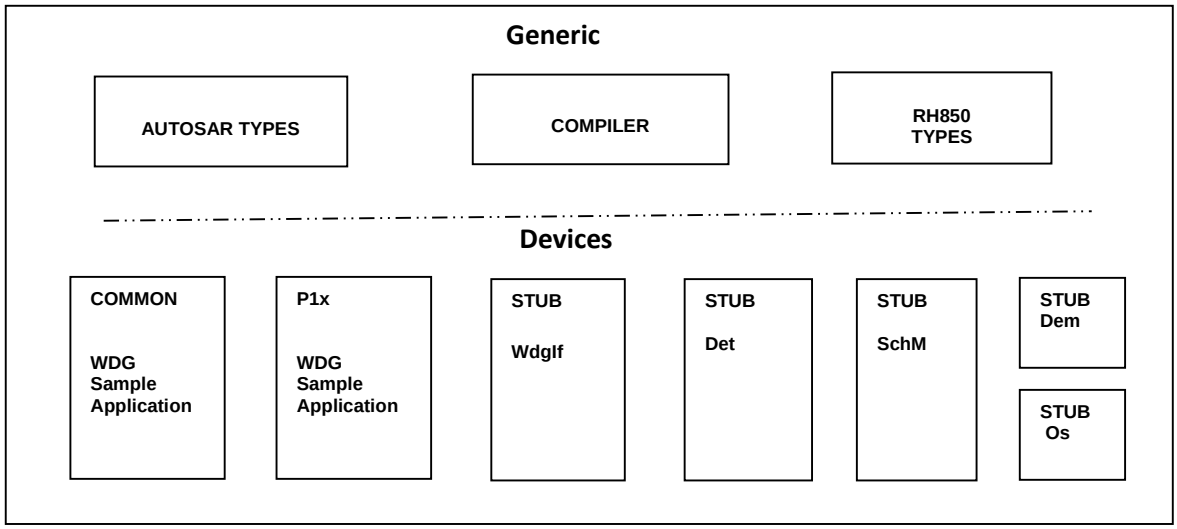

Figure 13-1 Overview Of WDG Driver Sample Application

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\<Complier>"

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\<Complier>").
  (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 701310
1.	ROM	WDG59_A_PUBLIC_CODE_ROM	374
		WDG_59_DRIVERA_FAST_CODE_ROM	70
		WDG59_A_CFG_ DATA_UNSPECIFIED	20
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 701310
1.	ROM	WDG59_A_PUBLIC_CODE_ROM	642
		WDG_59_DRIVERA_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 Section13.2.2.1 *Configuration Example*.

13.3.3 Throughput Details

The throughput details of the APIs for the configuration mentioned in the Section13.2.2.1 *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 701310	Remarks
1.	Wdg_59_DriverA_Init	1.775	Timing is measured with default mode as WDGIF_SLOW_MODE
2.	Wdg_59_DriverA_SetMode	0.262	Timing is measured with default mode as WDGIF_SLOW_MODE
3.	Wdg_59_DriverA_SetTriggerCon dition	0.637	-
4.	Wdg_59_DriverA_GetVersionInfo	0.125	-
5.	WDG_59_DRIVERA_TRIGGERFUNCTION_ISR	0.950	-

Chapter 14 Release Details

WDG Driver Software R4.0.3
Version: 1.0.11

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 chaper10 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

Sl.No	Description	Version	Date
7.	Following changes are made: 1. Section 3.1.1 is updated to add Wdg_59_DriverA_RegWrite.h and remove Wdg_59_DriverA_RegReadBack.h file 2. Section 4.4 is updated for State diagram of WDG Driver. 3. Section 4.6 Deviation List is updated. 4. Section 4.8 Register Read-Back is changed to Register Write-verify. 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. 6. Table 8-1 is updated for Wdg_59_DriverA_RegWrite and Wdg_59_DriverA_Cbk.h and stub files. 7. Section 10.2.1 is updated to remove element usDefaultTimeValue. 8. Table 11.2 is updated to add new DEM errors. 9. Removed *.html and *.one files from section 13.2. 10. Chapter 6 Register Details is Updated. 11. Chapter 12 Memory section is updated. 12. Section 13.1.1 ISR Function Mapping Interrupt Vector Table is updated with note. 13. Added a 'Note' below the table 'Supervisor mode and User mode details. 14. Table 11.1 is updated to add the DET error WDG_59_DRIVERA_E_PARAM_POINTER 15. Chapter 14 Release details is updated with WDG Driver. 16. Chapter 3 and 9 are updated with R-Numbered Tool User manual name. 17. Version of R-number is updated at the end of document. 18. Section 13.3 Memory and throughput Values are updated. 19. Section 4.2 and 4.3 is updated with critical section information. 20. Section 4.6 General is updated.	1.0.6	13-Jul-2016
8.	Following changes are made 1. Section 13.3 is updated for Memory and Throughput details. 2. Chapter 12 is updated to remove NO_INIT_RAM_16BIT and add NO_INIT_RAM_32BIT.	1.0.7	21-Oct-2016

AUTOSAR MCAL R4.0.3 User's Manual
WDG Driver Component Ver.1.0.7
Embedded User's Manual

Publication Date: Rev.1.00, October 21, 2016

Published by: Renesas Electronics Corporation

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

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

Renesas Electronics America Inc.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

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: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 1.0

AUTOSAR MCAL R4.0.3

User's Manual

Renesas Electronics Corporation

R20UT3728EJ0100