User's Manual

AUTOSAR MCAL R4.0.3

User's Manual

WDG Driver Component Ver.1.0.2

Embedded User's Manual

Target Device:
RH850/P1x-C

Renesas Electronics
www.renesas.com

Rev1.00 Feb 2017

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
ANSI	American National Standards Institute
API	Application Programming Interface
AUTOSAR	AUTomotive Open System ARchitecture
DEM/Dem	Diagnostic Event Manager
DET/Det	Development Error Tracer
DIO	Digital Input Output
ECU	Electronic Control Unit
Id	Identifier
ISR	Interrupt Service Routine
MCAL	Microcontroller Abstraction Layer
MCU	MicroController Unit
PWM	Pulse Width Modulation
RAM	Random Access Memory
ROM	Read Only Memory
SPI	Serial Peripheral Interface
WDG/Wdg	WatchDog
WDT	WatchDog Timer
WDGIF	WatchDog Interface

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-C microcontrollers.

This document shall be used as reference by the users of WDG Driver Component. The system overview of complete AUTOSAR architecture is shown in the below Figure:

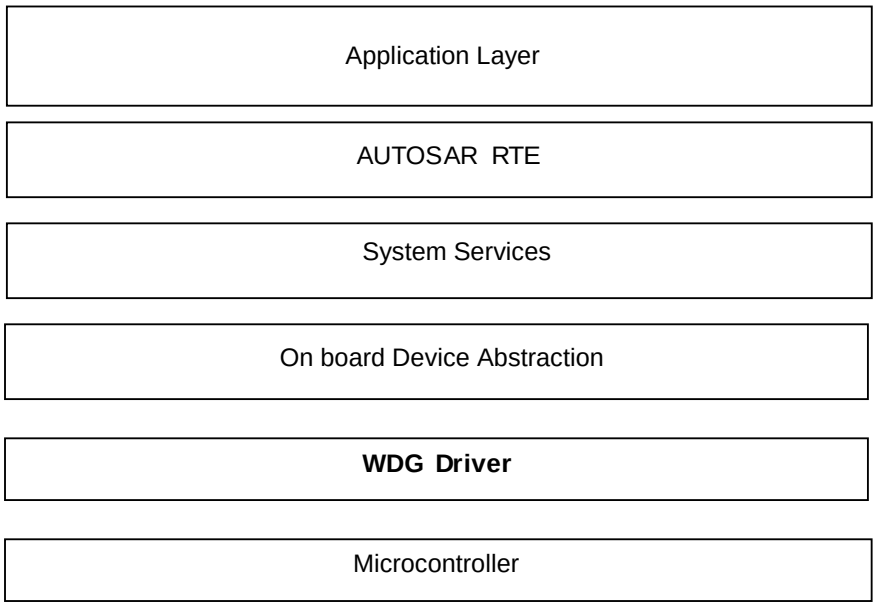

Figure 1-1 System Overview of AUTOSAR Architecture

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_Hardware.c, Wdg_59_DriverA_Cfg.h and Wdg_Hardware.h for Watchdog Driver A.

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

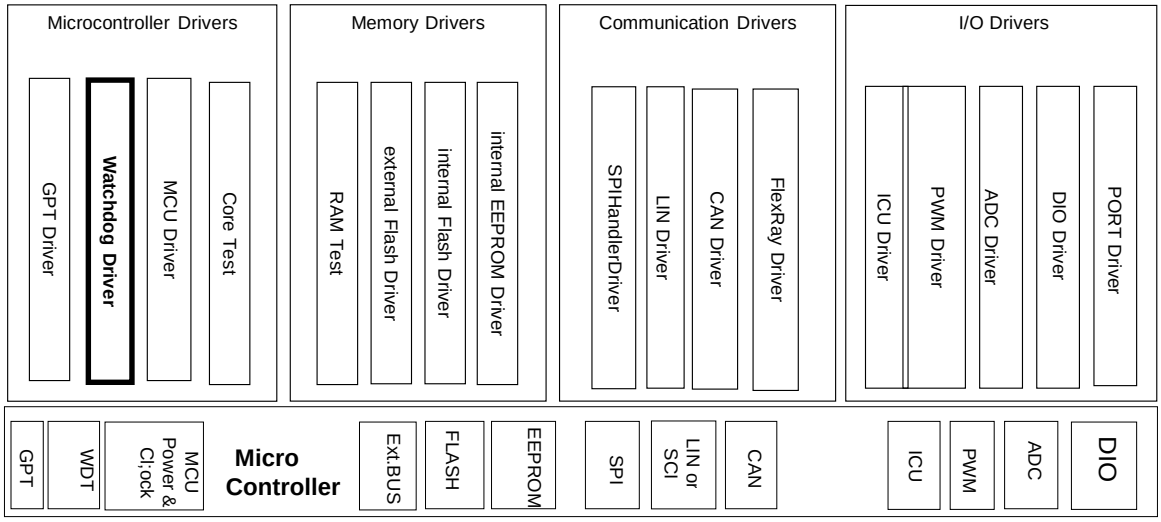

Figure 1-2 System Overview Of The WDG Driver In AUTOSAR MCAL Layer

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 and Support For Different Interrupt Categories.
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 (P1x-C Specific Information)	This section provides the P1x-C 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 Specification of WDG Driver (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.	RH850/P1H-C Document User's Manual: Hardware (r01uh0517ej0100_rh850p1x-c_Open.pdf)	1.00
4.	Specification of Compiler Abstraction (AUTOSAR_SWS_CompilerAbstraction.pdf)	3.2.0
5.	Specification of Memory Mapping (AUTOSAR_SWS_MemoryMapping.pdf)	1.4.0
6.	Specification of Platform Types (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 “R20UT3662EJ0100-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_Types.h
\Wdg_59_DriverA_Version.h
\Wdg_59_DriverA_RegWrite.h

X1X\ P1x-C \modules\wdg\sample_application\<SubVariant>\make\ghs App_Wdg_P1x-C_Sample.mak

X1X\ P1x-C \modules\wdg\sample_application\<SubVariant>\make\ghs App_Wdg_P1x-C_Sample.ld

X1X\ P1x-C \modules\wdg\sample_application\<SubVariant>\obj

X1X \P1x-C \modules\wdg\generator
\R403_WDG_P1x-C_BSWMDT.arxml

X1X\P1x-C\modules\wdg\user_manual

(User manuals will be available in this folder)

Note: 1. <AUTOSAR_version> should be 4.0.3.
2. <SubVariant> can be P1H-C.

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 does not implement any Call Back Notification functions.
- Example code mentioned in this document shall be taken only as a reference for implementation.
- The Watchdog hardware supports two instances. Hence, WDG Driver Component is implemented as two separate drivers Driver A and Driver B. The difference between these drivers are in the value of the parameter 'VendorApiInfix' in Parameter Definition File and the address of registers generated in respective configuration header file.
  WDG_DRIVER_INSTANCE variable of Base Make file also has to be updated for the selected driver.
- 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 Main Oscillator 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.
- 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.

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 is 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 and Wdg_59_DriverA_PBcfg.c 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.
- When using an emulator, please be sure to set a BOOTCTRL register. In SYNCHRONOUS debug mode it cannot be operated, because PE1 cannot start when PE2 is remaining in STOP mode. With this combination, you have to use the asynchronous debug mode, since only here two PEs are independently operated. Otherwise there is a permanent Fetch- stop. Synchronous debug shall be switched off with the command "target syncdebug off 1
- The user shall configure the exact Module Short Name Wdg in configurations as specified in config.xml file and the same shall be given in command line

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 WDG_59_DRIVERA_ENTER_CRITICAL_SECTION and WDG_59_DRIVERA_EXIT_CRITICAL_SECTION functions. The WDG_59_DRIVERA_ENTER_CRITICAL_SECTION function is called before the data needs to be protected and WDG_59_DRIVERA_EXIT_CRITICAL_SECTION 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:

WDG_59_DRIVERA_TRIGG_PROTECTION

WDG_59_DRIVERA_MODE_SWITCH_PROTECTION

The protection areas WDG_59_DRIVERA_TRIGG_PROTECTION and WDG_59_DRIVERA_MODE_SWITCH_PROTECTION are used to protect the WDG triggering and WDG mode switching respectively.

The functions WDG_59_DRIVERA_ENTER_CRITICAL_and WDG_59_DRIVERA_EXIT_CRITICAL_SECTION can be disabled by disabling the configuration parameter ‘WdgCriticalSectionProtection’.

Note: The above sequence is followed for DriverB also.

Table 4-1 WDG Driver Protected Resources List

API Name	Exclusive Area Type	Protected Resources
Wdg_59_DriverA_Init	WDG_59_DRIVERA_MODE_SWITCH_PROTECTION	Registers: IMR0, EIC8 WDTAAMD
	WDG_59_DRIVERA_TRIGG_PROTECTION	Registers: WDTAAWDTE WDTAAREF WDTAAEVAC
Wdg_59_DriverA_SetMode	WDG_59_DRIVERA_MODE_SWITCH_PROTECTION	Registers: WDTAAMD Global Data: Wdg_59_DriverA_GddCurrentMode
	WDG_59_DRIVERA_TRIGG_PROTECTION	Global Data: Wdg_59_DriverA_GpConfigPtr Wdg_59_DriverA_GusTiggerCounter Registers: WDTAAWDTE WDTAAREF WDTAAEVAC
Wdg_59_DriverA_SetTriggerCondition	WDG_59_DRIVERA_TRIGG_PROTECTION	Global Data: Wdg_59_DriverA_GpConfigPtr Wdg_59_DriverA_GusTiggerCounter
Wdg_59_DriverA_GetVersionInfo	-	-

Note1: The above is followed for DriverB also.

Note2: The highest measured duration of a critical section is 1.2micro seconds measured for Wdg_59_DriverA_SetMode API

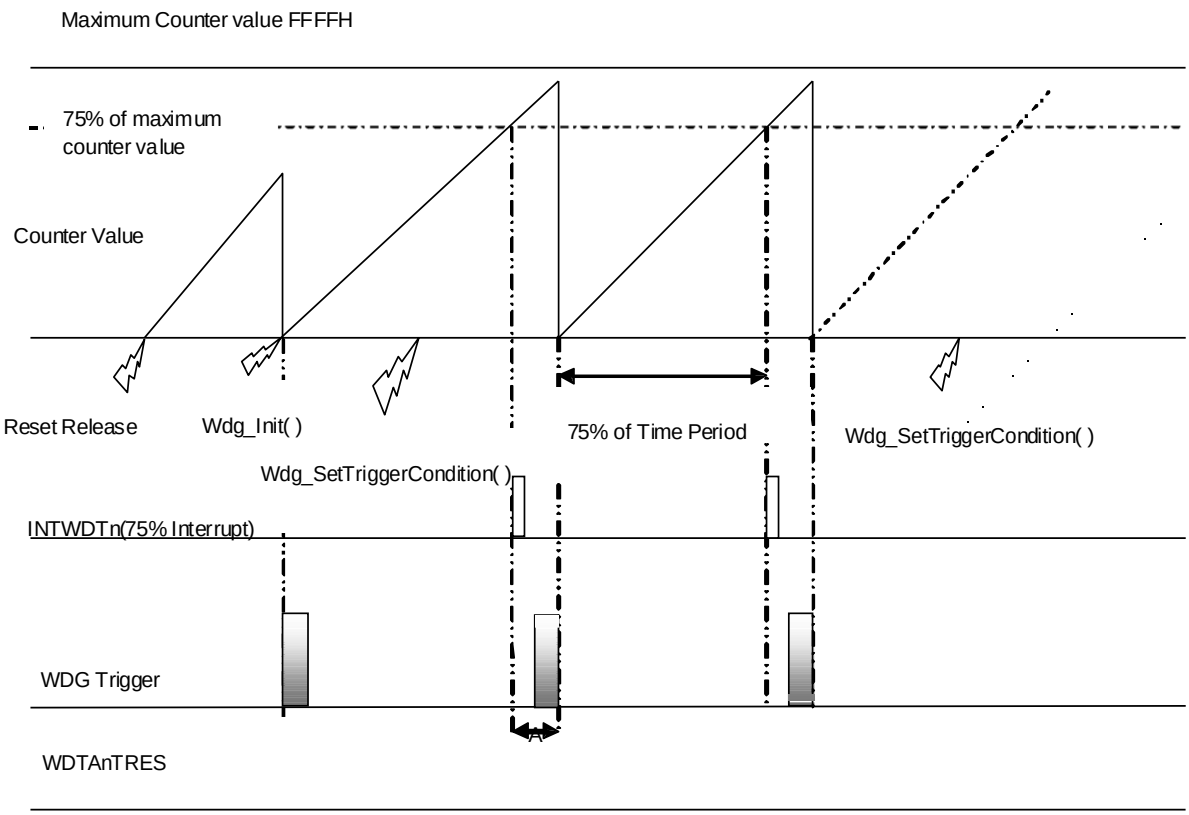

Figure 4-1 WDG behavior during Data exchange with hardware

4.4 WDG State Diagram

The State diagram of WDG Driver is as shown below

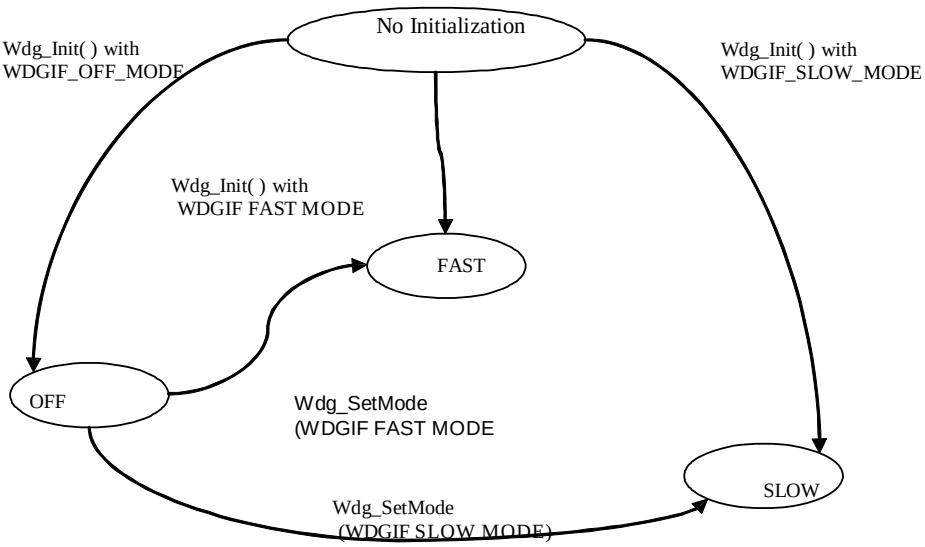

Figure 4-2 State Diagram of WDG when WdgDisableAllowed is true

WDG Driver supports following modes when configuration parameter WdgDisableAllowed is true.

1. WDGIF_OFF_MODE
2. WDGIF_SLOW_MODE
3. WDGIF_FAST_MODE

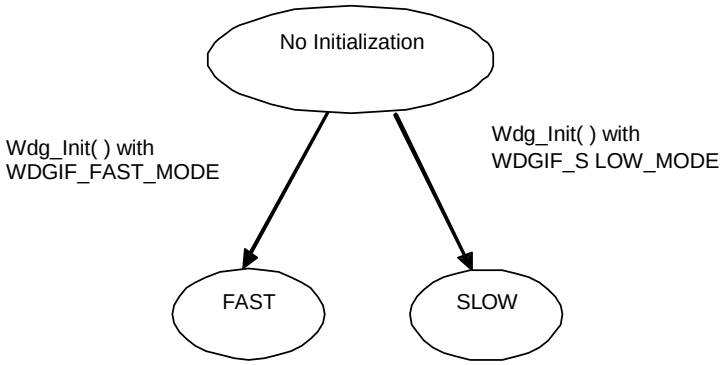

Figure 4-3 State Diagram of WDG when WdgDisableAllowed is false

WDG Driver supports following modes when configuration parameter WdgDisableAllowed is false

1. WDGIF_SLOW_MODE
2. WDGIF_FAST_MODE

Like shown in the above figures when WDG Driver is initialized by the API Wdg_Init(), the WDG Driver gets into one of the modes based on the default value configured during configuration. Also the modes can be changed by the API Wdg_SetMode() only once after Wdg_Init(), if the current mode is WDGIF_OFF_MODE.

4.5 WDTA 75% ISR Usage Details

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.

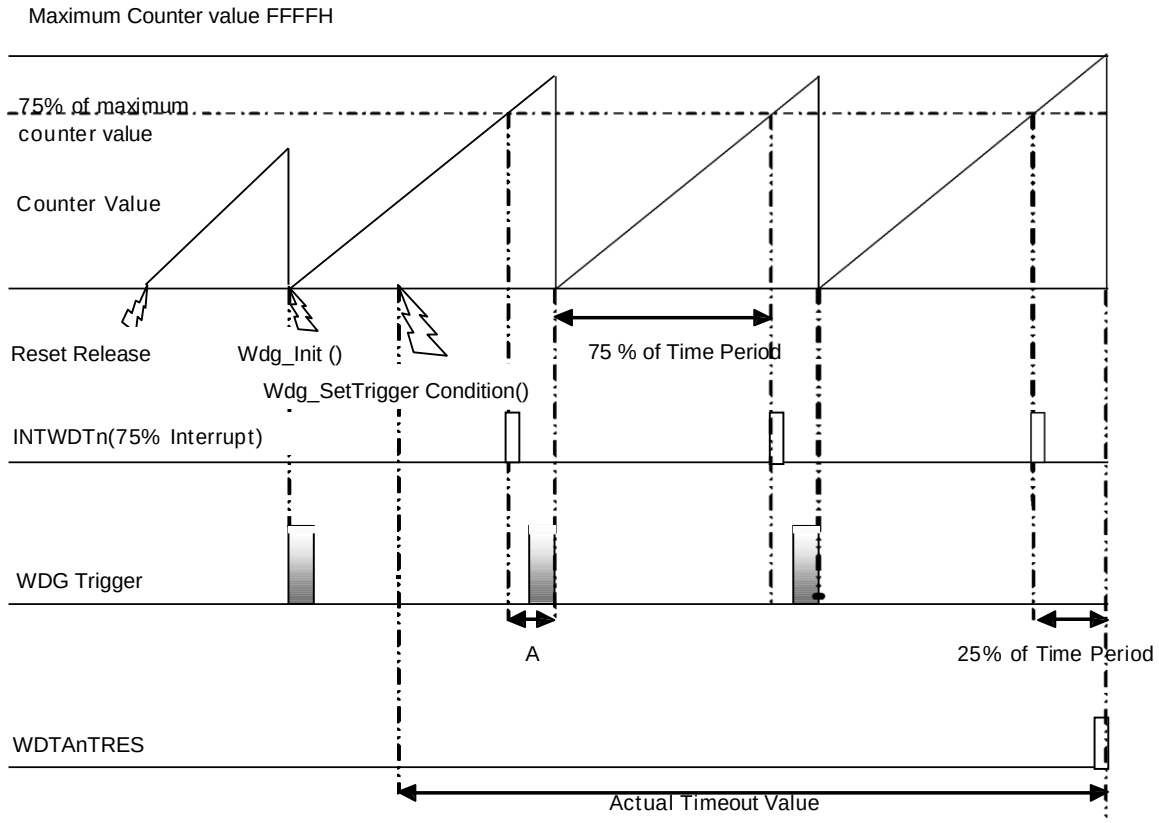

Figure 4-4 WDG behavior when Wdg_SetTriggerCondition is called

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_SetTriggerCondition'.

The above figure illustrates the scenario where Wdg_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 = 8MHz

For example considering current mode settings = WDTATCKI/2^16

Period = 2^16/8MHz = 0.008192 sec

Total window time = 8.192 msec

75% interrupt time = 6.144 msec

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 <6.143999 msec>

If the timeout value passed by the API Wdg_Settriggercondition is 100 msec, then the counter value will be calculated in the WDG Driver as 16.

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

WDTATCKI = 8MHz

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

Period = 2^9/240k = 0. 000064 sec

Total window time = 0.064 msec

75% interrupt time = 0.048 msec

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 <0.047999 msec>

If the timeout value passed by the API Wdg_Settriggercondition is 50 msec, then the counter value will be calculated in the WDG Driver as 50.

The API Wdg_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_SetTriggerCondition execution and next

WDG trigger from 75% ISR.

4.6 Deviation List

Table 4-2 WDG Driver Deviation List

Sl. No.	Description	AUTOSAR Bugzilla
1.	"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_Init().	-
2.	The requirement 'WDG025' is handled in the generation tool itself by the error 'ERR_102_054'.	-
3.	If the API Wdg_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.	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 mode
1	Wdg_59_DriverA_Init	-	x	Interrupt register IMR cannot be accessed.
2	Wdg_59_DriverA_SetMode	x	x	-
3	Wdg_59_DriverA_SetTriggerCondition	x	x	-
4	Wdg_59_DriverA_GetVersionInfo	x	x	-

Note1: Above Sequence is followed for DriverB also.

Note2: 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.

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:

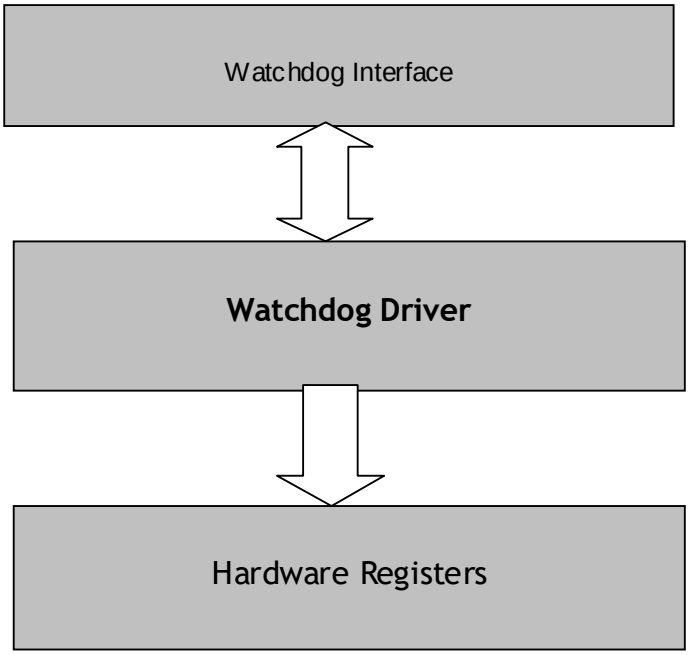

Figure 5-1 Watch Driver And Watchdog Interface Architecture

Watchdog Interface invoke 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:

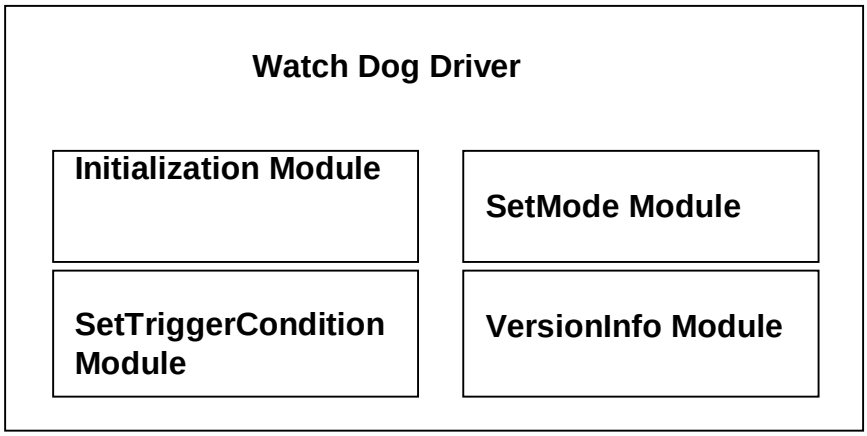

Figure 5-2 Basic Architecture Of WDG Component

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.

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	Macro/Variable
Wdg_59_DriverA_Init	IMRn	WdgErrorModeSetting	WDG_59_DRIVERA_INTWDTIMR_MASK
	WDTAnMD	WdgDefaultMode	ucWdtamdDefaultValue.
Wdg_59_DriverA_SetMode	WDTAnMD	-	ucWdtamdSlowValue, ucWdtamdFastValue
Wdg_59_DriverA_SetTriggerCondition	-	-	-
Wdg_59_DriverA_GetVersionInfo	-	-	-
Wdg_59_DriverA_TriggerFunc	WDTAnEVAC	-	WDG_59_DRIVERA_RESTART - WDG_59_DRIVERA_WDTAREF_ADDRESS
	WDTAnWDTE	-	WDG_59_DRIVERA_RESTART
	WDTAnREF	-	-

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
- Wdg_59_Hardware.h

The C source file generated by WDG Driver Generation Tool:

- Wdg_59_DriverA_PBcfg.c
- Wdg_59_Hardware.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

- Dem_IntErrId.h
- Det.h
- SchM_Wdg_59_DriverA.h
- SchM_Wdg_59_DriverB.h
- WdgIf_Types.h
- Os.h
- Rte.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 is generated by the WDG Driver Component Code Generation Tool for Prototype Declarations of WDG callback notification functions.
Wdg_59_Hardware.h	This file is generated by the WDG Generation Tool include definition of hardware registers specific to P1x-C WDG.
Wdg_59_Hardware.c	This file is generated by the WDG Generation Tool which consists of Global variable definition of hardware registers specific to P1x-C WDG.
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	This file contains the macro for the WDG Timer channels. It also 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 the macros functions for register write verify.
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.

File	Details
Wdg_59_DriverA_Version.c	This file contains the code for checking version of all modules that are interfaced to WDG.
Compiler_Cfg.h	This file contains the memory and pointer classes.
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.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.
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.

Note : Above provided WDG Driver Component Files are followed for DriverB also.

Chapter 9 Generation Tool Guide

For information on the WDG Driver Component Code Generation Tool, please refer “R20UT3662EJ0100-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
	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.
	Uint32	usDefaultTimeValue	75% time value of either slow or fast mode in milliseconds
	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 Used in WDG module

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_POINTER
Related API(s)	Wdg_59_DriverA_Init, Wdg_59_DriverA_GetVersionInfo
Source of Error	When the API service is called with a configuration pointer as NULL_PTR.
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_SetTriggerCondition
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_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.	**5**
Error Code	WDG_59_DRIVERA_E_INVALID_DATABASE
Related API(s)	Wdg_59_DriverA_Init
Source of Error	When the API service Wdg_59_DriverA_Init is called with invalid database.
Sl. No.	**6**
Error Code	WDG_59_DRIVERA_E_PARAM_CONFIG
Related API(s)	Wdg_59_DriverA_Init
Source of Error	When the above mentioned API is invoked with NULL parameter.

Note: Above DET errors are applicable for DriverB also.

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_Init
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_Init
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_INT_INCONSISTENT
Related API(s)	WDG_59_DRIVERB_TRIGGERFUNCTION_ISR
Source of Error	When interrupt consistency fails the above error is reported to DEM
Sl. No.	**4**
Error Code	WDG_59_DRIVERA_E_REG_WRITE_VERIFY
Related API(s)	Wdg_59_DriverA_Init, Wdg_59_DriverA_SetMode
Source of Error	When register write-verify fails the above error is reported to DEM.

Note: Above DEM errors are applicable for DriverB also

Chapter 12 Memory Organization

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

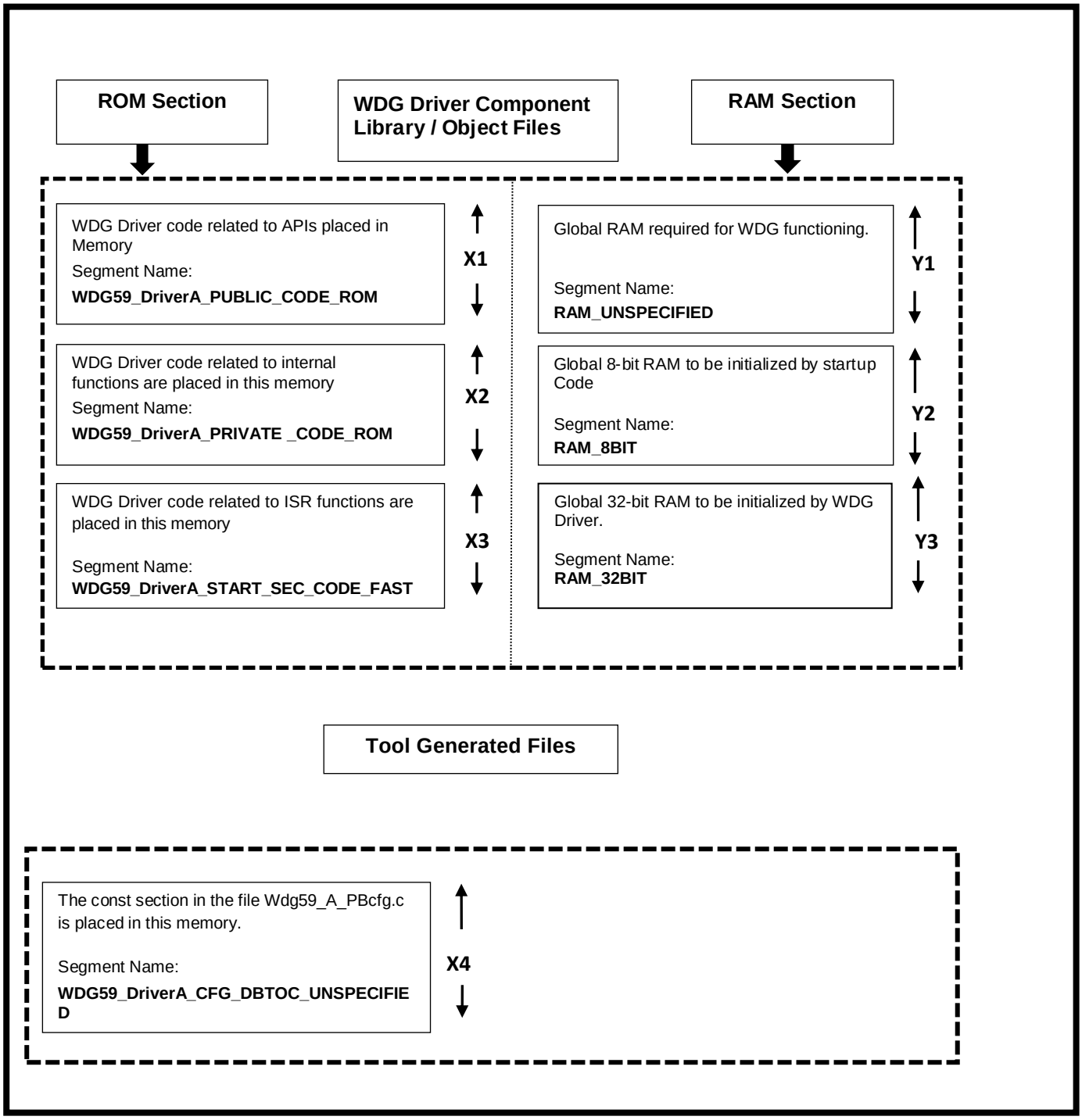

Figure 12-1 Memory Organization Of WDG Driver Component

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

WDG59_DriverA_PUBLIC_CODE_ROM (X1): WDG Driver Component APIs, which can be located in code memory.

WDG59_DriverA_PRIVATE_CODE_ROM (X2): Internal functions of WDG Driver Component code that can be located in code memory.

WDG59_DriverA_START_SEC_CODE_FAST(X3): Interrupt functions of WDG Driver Component code that can be located in code memory.

WDG59_DriverA_CFG_DBTOC_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 and Y3):

RAM_UNSPECIFIED (Y1): This section consists of the global RAM pointer 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.

RAM_8BIT (Y2): This section consists of the global RAM variables of 8-bit size that are initialized by start-up code and used internally by WDG Driver Component and other software components. The specific sections of respective software components will be merged into this RAM section accordingly.

RAM_32BIT (Y3): 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, Y1, Y2, and Y3 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 P1x-C Specific Information

P1x-C supports following devices:

- R7F701370A(CPU1(PE1)), R7F701371(CPU1(PE1)), R7F701372(CPU1(PE1)), R7F701373, R7F701374

13.1 Sample Application

13.1.1 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.

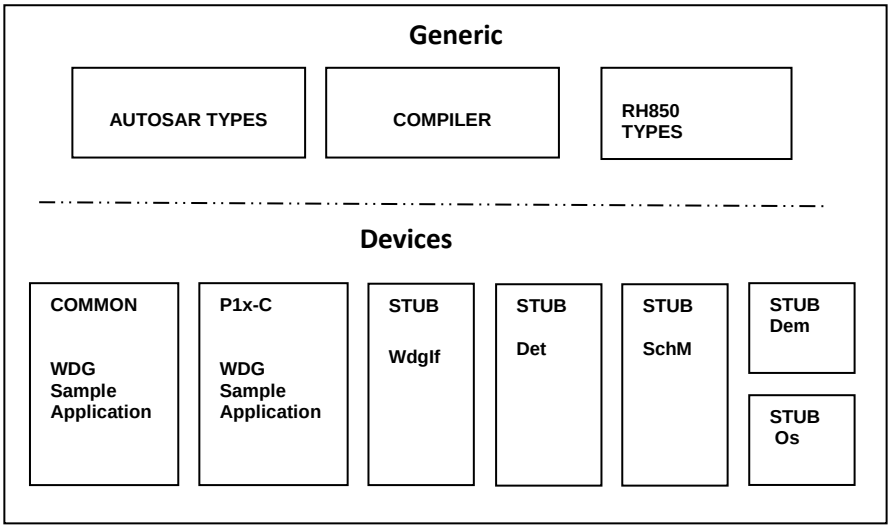

Figure 13-1 Overview Of WDG Driver Sample Application

The Sample Application of the P1x-C is available in the path

X1X\P1x-C\modules\wdg\sample_application

The Sample Application consists of the following folder structure

X1X\P1x-C\modules\wdg\definition\<AUTOSAR_version>\<SubVariant> \

R403_WDG_DriverA_P1X-C.arxml

X1X\P1x-C\modules\wdg\sample_application\<SubVariant>\
<AUTOSAR_version>

\src\WDG_59_DriverA_PBcfg.c

\include\WDG_59_DriverA_cfg.h

\config\ App_WDG_P1x-C_701370A_Sample.arxml

\config\ App_WDG_P1x-C_701371_Sample.arxml

\config\ App_WDG_P1x-C_701372_Sample.arxml

\config\ App_WDG_P1x-C_701373_Sample.arxml

\config\ App_WDG_P1x-C_701374_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.

13.1.2 Building Sample Application

13.1.2.1 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_P1x-C_701372_Sample.arxml

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, Opbyte0 value = 0x7FBFFFFF.

If Variable activation code is disabled, Opbyte0 value =

0x7F3FFFFF.

13.1.2.2 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-C\common_family\make\<Complier>”

Now execute the batch file SampleApp.bat with following parameter:

SampleApp.bat Wdg <Device_name>

- After this, the tool output files will be generated with the configuration as mentioned in App_WDG_P1x-C_701370A_Sample.arxml file available in the path:

  “X1X\P1x-C\modules\wdg\sample_application\<SubVariant>\

  4.0.3\config\App_WDG_P1x-C_701372_Sample.arxml”

- After this, all the object files, map file and the executable file App_WDG_P1x-C_Sample.out will be available in the output folder

  (“X1X\P1x-C\modules\wdg\sample_application\<SubVariant>\obj\”).

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

Note: Sample App is tested with Variable activation code disabled, Opbyte value = 0x7F3FFFFF.

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 file

  “X1X\P1x-

  C\modules\wdg\sample_application\<SubVariant>\4.0.3\config\App_W

  DG_P1x-C_701372_Sample.arxml”

- The database alone can be generated by using the following commands.

  make –f App_WDG_P1x-C_Sample.mak generate_wdg_config

  make –f App_WDG_P1x-C_Sample.mak App_WDG_P1x-C_Sample.s37

- After this, a flashable Motorola S-Record file App_WDG_P1x-C_Sample.s37 is available in the output folder.

Note: The <Device_name> indicates the device to be compiled, which can be 701372 and <SubVariant> can be P1H-C.

13.2 Memory And Throughput

13.2.1 ROM/RAM Usage

The details of memory usage for the typical configuration is provided in this section.

Only 1 WDG driver instance configured

WdgDevErrorDetect: false
WdgDisableAllowed: true
WdgVersionInfoApi: true
WdgInitialTimeout: 1

WdgDefaultMode: WDGIF_SLOW_MODE
WdgClkSettingsFast: TWO_POWOF_9_DIVBY_ WDTATCKI
WdgClkSettingsSlow: TWO_POWOF_16_DIVBY_ WDTATCKI

The details of memory usage for the typical configuration is provided in the section below.

Table 13-1 ROM/RAM Details without DET

Sl. No.	ROM/RAM	Segment Name	Size in bytes
1.	ROM	DEFAULT_CODE_ROM	638
2.	RAM	RAM_UNSPECIFIED	4
		RAM_32BIT	4
		RAM_8BIT	1

The details of memory usage for the typical configuration, with DET enabled is provided in the section below.

Table 13-2 ROM/RAM Details with DET

Sl. No.	ROM/RAM	Segment Name	Size in bytes
1.	ROM	DEFAULT_CODE_ROM	922
2.	RAM	RAM_UNSPECIFIED	4
		RAM_32BIT	4
		RAM_8BIT	2

13.2.2 Stack Depth

The worst-case stack depth for WDG Driver Component is 76 bytes for the typical configuration.

13.2.3 Throughput Details

The throughput details of the APIs for the Typical configuration is as follows:

The clock frequency used to measure the throughput is 240MHz for all APIs.

Table 13-3 Throughput Details of the APIs

Sl. No.	API Name	Throughput in microseconds	Remarks
1.	Wdg_59_DriverA_Init	0.862	Timing is measured with default mode as WDGIF_SLOW_MODE
2.	Wdg_59_DriverA_SetMode	0.287	Timing is measured with passing parameter WDGIF_SLOW_MODE
3.	Wdg_59_DriverA_SetTriggerCondition	0.325	-
4.	Wdg_59_DriverA_GetVersionInfo	0.620	-
5.	WDG_59_DRIVERA_TRIGGERFUNCTION_ISR	0.425	-

13.2.4 Precautions

If the critical section is disabled, there could be possibility that API is called concurrently and executed based on task priority in which it is called.Because the data inconsistency described above may happen, Critical Section Protection is required to avoid this mal-functioning.

Chapter 14 Release Details

WDG Driver Software

Version: 1.0.2

Revision History

Sl.No	Description	Version	Date
1.	Initial Version	1.0.0	10-Aug-2015
2.	The following changes are made: 1. Deviation list updated for requirement BSW00347. 2. Updated section 4.1 General. 3. R-number added to the document. 4. Added DET error for Wdg_59_DriverA_GetVersionInfo in Chapter 11. 5. Section 13.3 Memory and Throughput updated. 6. Updated section 4.2 Preconditions.	1.0.1	23-Feb-2016
3.	1. The following changes are made: 2. Section 13.2.4 Precautions is added. 3. Chapter 8 stub C header file is replaced with The port specific C header files and added the stub files 4. Added a ‘Note’ and updated the table 'Supervisor mode And User mode’ details. 5. Section 13.2.1, naming convention of WdgClkSettingsFast and WdgClkSettingsSlow values are changed. 6. Section 13.2 Memory and Throughput updated. 7. R-number updated for the document. 8. Device name updated. 9. Section 13.1 Compiler, Linker and Assembler removed. 10. Updated Section 4.3 for Critical section details and added table WDG Driver Protected Resources List. 11. Reference for .one and .html are removed. 12. In Section 4.7 updated Note to Note2 and added Note1 13. Added new errors in Table 11-2 DEM Errors Of WDG Driver Component and added Notes to Table 11-2 and Table 11-1. 14. Updated Section 8 with a Note. 15. Updated Content of Section 13.2.4	1.0.2	27-Feb-2017

AUTOSAR MCAL R4.0.3 User's Manual
WDG Driver Component Ver. 1.0.2
Embedded User's Manual

Publication Date: Rev1.00, February 27, 2017

Published by: Renesas Electronics Corporation

Renesas Electronics Corporation

SALES OFFICES http://www.renesas.com

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 Hyflux 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, Jln 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

Colophon 4.1

AUTOSAR MCAL R4.0.3

User's Manual

Renesas Electronics Corporation

R20UT3661EJ0100