User's Manual

AUTOSAR Modules Overview

User's Manual

Version 1.0.3

Target Device:
RH850/X1x

Renesas Electronics
www.renesas.com

Rev.1.01 May 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
ADC	Analog to Digital Converter
API	Application Programming Interface
ATOM	ARU-connected Timer Output Module
AUTOSAR	AUTomotive Open System ARchitecture
CC	Communication Controller
CMU	Clock Management Unit
CORTST	Core Test
DEM	Diagnostic Event Manager
DET/Det	Development Error Tracer
DIO	Digital Input Output
ETH	Ethernet
FLS	FLaSh Driver
FLSTST	FLaSh Test
FR	FlexRay
FSL	Flash Self programming Library
GPT	General Purpose Timer
GTM	Generic Timer Module
ICU	Input Capture Unit
LIN	Local Interconnect Network
Lpdu	Data Link Protocol Datagram Unit
MCAL	MicroController Abstraction Layer
MCU	MicroController Unit
Nm	Network Management
POC	Protocol Operation Control
PWM	Pulse Width Modulation
RAMTST	Ram Test
Rx	Receiver
SPI	Serial Peripheral Interface
TIM	Timer Input Module
WDG	WatchDog driver
µC	Micro controller

Definitions

Term	Represented by
Sl. No.	Serial Number
<Autosar Version>	4.0.3 when tested for R4.0.3

Table of Contents

List of Figures

List of Tables

Chapter 1 INTRODUCTION

This document shall be used as reference by the users for module overview, module dependencies, source code dependencies and configuration parameter dependencies.

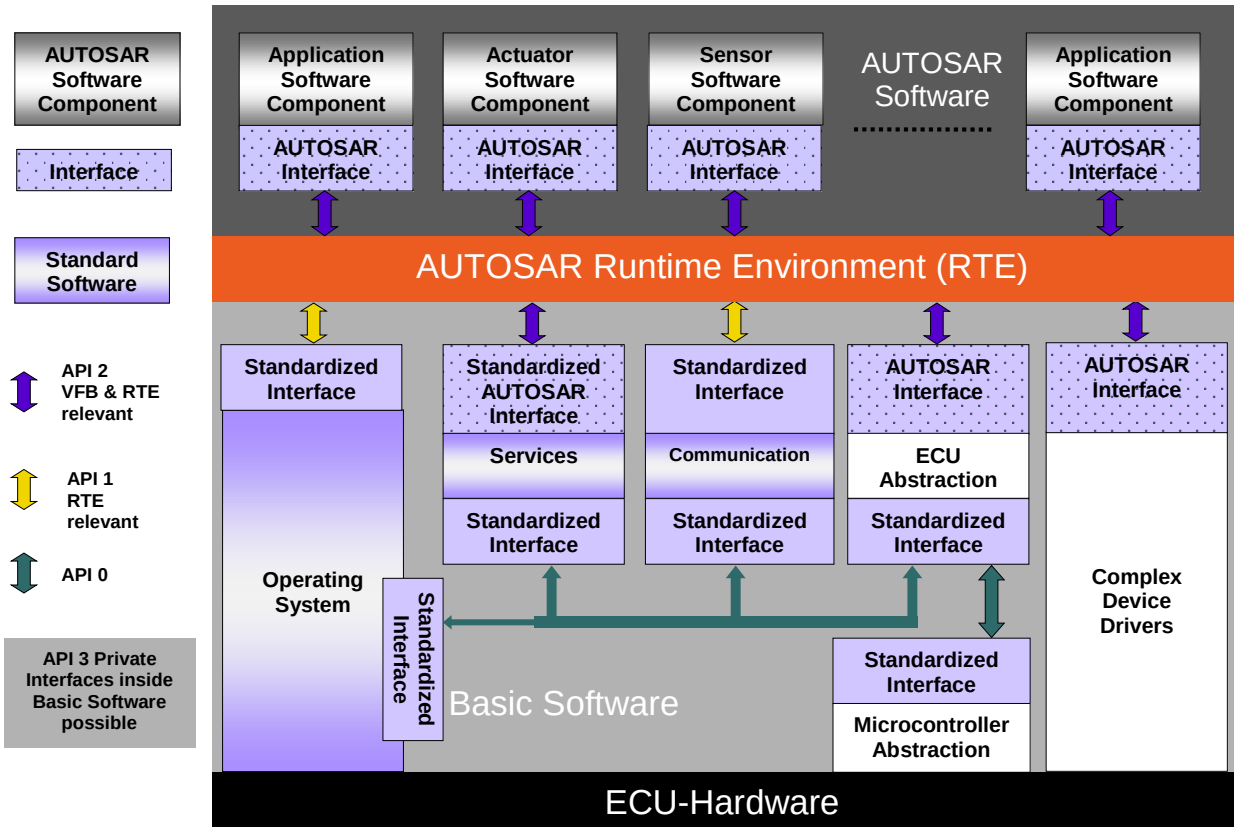

Figure 1-1 : System Overview of the AUTOSAR Architecture Layer

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
Section1 (Introduction)	Explains the purpose of this document.
Section2 (Reference Documents)	Lists the documents referred for developing this document.
Section3 (MCAL Modules)	Provides the list of modules developed in the MCAL layer. Brief information about the Module overview, Modules dependency, Configuration parameter dependency, source code dependency and stubs .

Chapter 2 REFERENCE DOCUMENTS

Sl. No.	Title For Autosar Version R4.0.3	Version
1.	Specification of ADC Driver (AUTOSAR_SWS_ADCDriver.pdf)	4.2.0
2.	Specification of PWM Driver (AUTOSAR_SWS_PWMDriver.pdf)	2.5.0
3.	Specification of PORT Driver (AUTOSAR_SWS_PortDriver.pdf)	3.2.0
4.	Specification of DIO Driver (AUTOSAR_SWS_DIODriver.pdf)	2.5.0
5.	Specification of Module Flash Driver (AUTOSAR_SWS_FlashDriver.pdf)	3.2.0
6.	Specification of SPI Handler/Driver (AUTOSAR_SWS_SPI_HandlerDriver.pdf)	3.2.0
7.	Specification of ICU Driver (AUTOSAR_SWS_ICUDriver.pdf)	4.2.0
8.	Specification of MCU Driver (AUTOSAR_SWS_MCUDriver.pdf)	3.2.0
9.	Specification of GPT Driver (AUTOSAR_SWS_GPTDriver.pdf)	3.2.0
10.	Specification of Watchdog Driver (AUTOSAR_SWS_WatchdogDriver.pdf)	2.5.0
11.	Specification of LIN Driver (AUTOSAR_SWS_LINDriver.pdf)	1.5.0
12.	Specification of FR Driver (AUTOSAR_SWS_FlexRayDriver.pdf)	2.5.0
13.	Specification of RAMTST Driver (AUTOSAR_SWS_RAMTest Driver.pdf)	1.5.0
14.	Specification of CORTST Driver (AUTOSAR_SWS_CoreTest.pdf)	1.2.0
15.	Specification of FLSTST Driver (AUTOSAR_SWS_FlashTest.pdf)	1.2.0
16.	Specification of ETH Driver (AUTOSAR_SWS_EthernetDriver.pdf)	1.2.0

Chapter 3 AUTOSAR MODULES

3.1 MCAL Module

The MicroController Abstraction layer is the lowest software layer of the Basic Software. It contains internal drivers, which are software modules with direct access to the µC internal peripherals and memory mapped µC external devices. Make higher software layers independent of µC.

The modules developed for MCAL layer are as follows:

ADC

PWM

PORT

DIO

FLS

SPI

ICU

MCU

GPT

WDG

LIN

FR

RAMTST

CORTST

FLSTST

ETH

3.1.1. ADC Driver Component

3.1.1.1. Module Overview

The ADC driver shall initialize and control the internal Analog Digital Converter unit of the microcontroller. The driver is equipped with a set of basic functionalities with single value result access mode and streaming access mode.

A One Shot conversion shall be started by a software trigger or a hardware event whereas a Continuous conversion shall be started by a software trigger only. The ADC conversion results shall be returned by an ADC read service. This service shall return the last converted result from an external result buffer.

The ADC Driver software component shall provide the following main features:

- Single value results access mode supports One-Shot conversion and Continuous conversion
- Streaming access mode supports linear buffer conversion and circular buffer conversion
- Various API services for functionalities like initialization, de-initialization, starting and stopping of ADC channels
- Notifications services for ADC channels

- Hardware Trigger services for ADC channels
- Channel group priority mechanism

3.1.1.2. Module Dependency

The dependency of ADC Driver on other modules and the required implementation is briefed as follows:

DET

In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

DEM

The Diagnostic Event manager (DEM) will be called whenever this module encounters a production relevant error.

PORT driver

Port pins used by the ADC Driver shall be configured using the PORT module. Both analog input pins and external trigger pins have to be considered.

IO Hardware Abstraction Layer

The ADC driver depends on the IO Hardware Abstraction Layer, which invokes the APIs and receives the callback notifications. If IO Hardware Abstraction Layer Module is not available, then the required functionality shall be stubbed.

RTE

The Run time Environment (RTE) module will be called whenever a critical section protection function is called.

OS

The ADC driver uses interrupts and therefore there is a dependency on the OS which configures the interrupt sources. If OS is not available, then the configuration of interrupt sources shall be stubbed.

3.1.1.3. Configuration Parameter Dependency

None

3.1.1.4. Source Code Dependency

The following are the common dependent used files by the ADC Driver module:

Det.h,

Dem.h,

MemMap.h,

Platform_Types.h,

Std_Types.h,

SchM_Adc.h

Rte.h

Os.h

rh850_Types.h

3.1.1.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>\”

The tables below will provide the common stubs to be used for ADC Driver component

Table 3-1 : ADC Driver Component Common Stubs

Common Stubs	Path
Det	X1X\common_platform\generic\stubs\<Autosar Version>\Det
SchM	X1X\common_platform\generic\stubs\<Autosar Version>\SchM
Os	X1X\common_platform\generic\stubs\<Autosar Version>\Os
Dem	X1X\common_platform\generic\stubs\<Autosar Version>\Dem

3.1.2. PWM Driver Component

3.1.2.1. Module Overview

The PWM Driver Component provides services for PWM Driver Component initialization, de-initialization, setting the period and duty cycle for a PWM channel, reading the internal state of PWM output signal and setting the PWM output to idle state and disabling or enabling the PWM signal edge notification. The PWM Driver Component is part of the Microcontroller Abstraction Layer (MCAL), the lowest layer of Basic Software in the AUTOSAR environment.

The PWM Driver Component is divided into PWM High Level Driver and PWM Low Level Driver to minimize the effort and to optimize the reuse of developed software on different platforms.

The PWM High Level Driver exports the APIs to the upper modules. All the references to specific microcontroller features and registers are provided in PWM Low Level Driver.

ATOM submodule of Generic Timer Module is used to generate variable PWM output.

The channel level notifications are provided for the rising edge, falling edge and both edges. Any of these notifications will be active only when these are configured for the corresponding channel and enabled by using PWM Driver Component APIs.

The PWM Driver component should provide following services based on the functions performed by the PWM Driver:

- Initialization
- De-Initialization

- Set the channel output to Idle
- Get the channel output state
- Set Duty Cycle
- Set Duty Cycle and Period
- Notification services (at the beginning, at the end and on both edged of a period)
- Get Version information

3.1.2.2. Module Dependency

The dependency of PWM Driver on other modules and the required implementation is briefed as follows:

DET

In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

DEM

The Diagnostic Event manager (DEM) will be called whenever this module encounters a production relevant error.

MCU Driver

The Microcontroller Unit Driver (MCU Driver) is primarily responsible for initializing the GTM CMU clock sources.

PORT driver

Port pins used by the PWM Driver shall be configured using the PORT module.

IO Hardware Abstraction Layer

The PWM driver depends on the IO Hardware Abstraction Layer, which invokes the APIs and receives the callback notifications. If IO Hardware Abstraction Layer Module is not available, then the required functionality shall be stubbed.

OS

The PWM driver uses interrupts and therefore there is a dependency on the OS which configures the interrupt sources. If OS is not available, then the configuration of interrupt sources shall be stubbed.

RTE

The Run time Environment (RTE) module will be called whenever a critical section protection function is called.

3.1.2.3. Configuration Parameter Dependency

The PWM Driver Depends on MCU for the clock source configuration. Hence the parameter
'PwmGTMClockRef' in the 'PwmGeneral' container refers to the path "/Renesas/EcucDefs_Mcu/Mcu0/McuModuleConfiguration0/McuGTMClockSettingsConfig0"

3.1.2.4. Source Code Dependency

The following are the common dependent used files by the PWM Driver module:

Det.h,

Dem.h,

MemMap.h,

Platform_Types.h,

Std_Types.h,

SchM_Pwm.h

Rte.h

Os.h

rh850_Types.h

3.1.2.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>\”

The table below will provide the common stubs to be used for PWM Driver component

Table 3-2 : PWM Driver Component Common Stubs

Common Stubs	Path
Det	X1X\common_platform\generic\stubs\<Autosar Version>\Det
SchM	X1X\common_platform\generic\stubs\<Autosar Version>\SchM
Os	X1X\common_platform\generic\stubs\<Autosar Version>\Os
Dem	X1X\common_platform\generic\stubs\<Autosar Version>\Dem

3.1.3. PORT Driver Component

3.1.3.1. Module Overview

The PORT Driver Component access the hardware features directly. The upper layers call the functionalities provided by these components.

The PORT Driver Component provides services for:

- Initialization of every port pins to configured functionality.
- Changing the port pin direction during run time.
- Refreshing the port pin directions.
- Setting the port pin mode during runtime.
- Reading module version

3.1.3.2. Module Dependency

The dependency of PORT Driver on other modules and the required implementation is briefed as follows:

DET

In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

DEM

The Diagnostic Event manager (DEM) will be called whenever this module encounters a production relevant error.

RTE

The Run time Environment (RTE) module will be called whenever a critical section protection function is called.

3.1.3.3. Configuration Parameter Dependency

None.

3.1.3.4. Source Code Dependency

The following are the common dependent used files by the PORT Driver module:

Det.h,

Dem.h,

MemMap.h,

Platform_Types.h,

Std_Types.h,

SchM_Port.h

Rte.h and

3.1.3.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The table below will provide the common stubs to be used for PORT Driver component

Table 3-3 : PORT Driver Component Common Stubs

Common Stubs	Path
Det	X1X\common_platform\generic\stubs\<Autosar Version>\Det
SchM	X1X\common_platform\generic\stubs\<Autosar Version>\SchM
Dem	X1X\common_platform\generic\stubs\<Autosar Version>\Dem

3.1.4. DIO Driver Component

3.1.4.1. Module Overview

The DIO Driver Component access the hardware features directly. The upper layers call the functionalities provided by these components.

The DIO Driver Component provides services for:

- Reading from / writing to DIO Channel
- Reading from / writing to DIO Ports
- Reading from / writing to DIO Channel Groups
- Reading module version.

3.1.4.2. Module Dependency

The dependency of DIO Driver on other modules and the required implementation is briefed as follows:

DET

In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

DEM

The Diagnostic Event manager (DEM) will be called whenever this module encounters a production relevant error.

PORT driver

Port pins used by the DIO Driver shall be configured using the PORT module.

3.1.4.3. Configuration Parameter Dependency

None

3.1.4.4. Source Code Dependency

The following are the common dependent used files by the DIO Driver module:

Det.h,

Dem.h,

MemMap.h,

Platform_Types.h and

Std_Types.h

3.1.4.5. Stubs

The DIO driver uses Stubs which is categorized as common stubs and available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The table below provides the common stubs to be used for DIO Driver component:

Table 3-4 : DIO Driver Component Common Stubs

Common Stubs	Path
Det	X1X\common_platform\generic\stubs\<Autosar Version>\Det
Dem	X1X\common_platform\generic\stubs\<Autosar Version>\Dem

3.1.5. FLS Software Component

3.1.5.1. Module Overview

The FLS software component provides services for reading, writing, comparing and erasing flash memory.

The FLS Component conforms to the AUTOSAR standard and is implemented mapping to the AUTOSAR FLS Software Specification.

The FLS Driver Software Component provides services for:

- Initialization
- Erasing the flash memory
- Reading from the flash memory
- Writing to the flash memory
- Validating contents of flash memory
- Cancellation of Request
- Job result and status information
- Background job processing
- Module version information
- Job Processing

3.1.5.2. Module Dependency

The dependency of FLS software component on other modules and the required implementation is briefed as follows:

DET
In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

DEM

The Diagnostic Event manager (DEM) will be called whenever this module encounters a production relevant error.

RTE

The Run time Environment (RTE) module will be called whenever a critical section protection function is called.

3.1.5.3. Configuration Parameter Dependency

The FLS Driver Depends on the MCU Driver for clock value. Hence the parameter ‘FlsCpuFrequency’ in the ‘FlsDataFlash’ container refers to the path
/AUTOSAR/EcucDefs_Mcu/Mcu0/McuModuleConfiguration0/McuClockSett

ingConfig0/McuPLLClkSetting0

3.1.5.4. Source Code Dependency

The following are the common dependent used files by the FLS Software Component module:

Det.h,

Dem.h,

MemMap.h,

Platform_Types.h,

Std_Types.h,

SchM_Fls.h,

Rte.h

rh850_Types.h

3.1.5.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The tables below will provide the common stubs to be used for FLS Software component.

Table 3-5 : FLS Software Component Common Stubs

Common Stubs	Path
Det	X1X\common_platform\generic\stubs\<Autosar Version>\Det
Dem	X1X\common_platform\generic\stubs\<Autosar Version>\Dem
SchM	X1X\common_platform\generic\stubs\<Autosar Version>\SchM

3.1.6. SPI Driver Component

3.1.6.1. Module Overview

The SPI driver is split as High Level Driver and Low Level Driver. The High Level Driver exports the AUTOSAR API towards upper modules and it will be designed to allow the compilation for different platforms without or only slight modifications, i.e. that no reference to specific microcontroller features or registers will appear in the High Level Driver. All these references are moved inside a µC specific Low Level Driver. The Low Level Driver interface extends the High Level Driver types and methods in order to adapt it to the specific target microcontroller.

The SPI Driver Component provides services for:

- Initialization and De-initialization
- Buffer Management

- Communication
- Status information
- Module version information
- Memory mapping
- Compiler abstraction

3.1.6.2. Module Dependency

The dependency of SPI Driver on other modules and the required implementation is briefed as follows:

DET

In development mode, the Development Error Tracer (DET) will be called whenever this module encounters a development error.

PORT

The CSIG HW Units uses port lines as external chip selects. In this case, the chip select is realized using microcontroller pins and hence the SPI module has a relationship with PORT module for initializing appropriate mode and direction of the port lines.

The basic SPI functionality for both CSIG and CSIH has to be configured as an alternate functionality by the PORT module.

IO Hardware Abstraction Layer

The IO Hardware Abstraction Layer invokes APIs of the SPI module and receives the callback notifications.

Memory Hardware Abstraction Layer

The Memory Hardware Abstraction Layer invokes APIs of the SPI module in case driver for any external memory devices (for example, external EEPROM) are implemented through the SPI module.

Onboard Device Abstraction Layer
The Onboard Device Abstraction Layer invokes APIs of the SPI module in case driver for any external devices (for example, external watchdog) are implemented through the SPI module.

RTE

The functions related to critical section protection area of the SPI module are invoked by the Run time Environment (RTE) module.

DEM

The SPI module uses the DEM module for getting the reference for all production errors.

3.1.6.3. Configuration Parameter Dependency

The SPI Driver Depends on the MCU Driver for clock value. Hence the parameter ‘SpiClockFrequencyRef’ in the ‘SpiExternalDevice’ container refers to the path /Renesas/EcucDefs_Mcu/Mcu/McuModuleConfiguration0/McuClockSettingConfig0/McuClockReferencePoint0/McuClockReferencePointFrequency

3.1.6.4. Source Code Dependency

The following are the common dependent used files by the SPI Driver module:

Det.h,

Dem.h,

MemMap.h,

Platform_Types.h,

Std_Types.h,

SchM_Spi.h

Rte.h

Os.h

rh850_Types.h

3.1.6.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The tables below will provide the common stubs to be used for SPI Driver component

Table 3-6 : SPI Driver Component Common Stubs

Common Stubs	Path
Det	X1X\common_platform\generic\stubs\<Autosar Version>\Det
Dem	X1X\common_platform\generic\stubs\<Autosar Version>\Dem
SchM	X1X\common_platform\generic\stubs\<Autosar Version>\SchM
Os	X1X\common_platform\generic\stubs\<Autosar Version>\Os

3.1.7. ICU Driver Component

3.1.7.1. Module Overview

The ICU Driver Component provides following services:

- Signal Edge detection and notification
- Services for Driver initialization and de-initialization
- Signal time measurement like Active Time, Period Time and Duty cycle
- Signal Edge time stamping and Edge counting
- Support Post-build configurations

The ICU Driver Component is part of the Microcontroller Abstraction Layer (MCAL), the lowest layer of Basic Software in the AUTOSAR environment. Different applications require different number of ICU channels in different modes. Therefore, the timer operation modes and external interrupts have to be selected depending on ICU measurement mode. For P1x-C microcontroller generation, following concepts are considered:

- Using TIM0/TIM1 channels for Edge Counting Measurement mode
- Using TIM0/TIM1 channels for Time Stamping Measurement mode
- Using TIM0/TIM1 channels for Signal Measurement mode
- Using TIM0/TIM1 and External Interrupts channels for Edge Detection mode

The ICU channel can be configured to either a timer channel or an external interrupt based on the required measurement mode. The configuration for Edge Detection measurement mode will be made only for an external interrupt channel and TIM0/TIM1 channels. The remaining three measurement modes viz. Edge Counting, Time Stamping and Signal Measurement should be configured only for the TIM0/TIM1 channels. The configuration of Timer in different operating modes will be taken care by the software itself.

The ICU Driver component can be divided into following sections based on the functions performed by the ICU Driver:

- Initialization
- De-Initialization
- Wakeup Services
- Notification Services
- Signal Measurement Services
- Signal Activation and State Information Services
- Version Information

3.1.7.2. Module Dependency

The dependency of ICU Driver on other modules and the required implementation is briefed as follows:

MCU Driver

The Microcontroller Unit Driver (MCU Driver) is primarily responsible for initializing the GTM CMU clock sources.

OS

The ICU driver uses interrupts and therefore there is a dependency on the OS which configures the interrupt sources. If OS is not available, then the configuration of interrupt sources shall be stubbed.

PORT Module

The configuration of port pins used for the ICU as inputs is done by the PORT driver. Hence the PORT driver has to be initialized prior to the use of ICU functions. If the PORT Driver is not available, then the configuration of port pins used for the ICU shall be stubbed.

In order to use the external interrupt functionality, port filter of respective

external interrupt needs to be enabled in PORT component. ICU can override edge detection settings and PORT can do as well. The registers FCLAxCTLx are used by ICU and PORT at the same time and the order of calling APIs is important.

EcuM Module

The ICU driver shall do the reporting of wakeup interrupts to the EcuM. If the EcuM is not available, and then the required functionality shall be stubbed.

DET Module

If the Development Error Tracer is not available, stubs need to be used to the interfaces for those modules.

IO Hardware Abstraction Layer Module

The ICU driver depends on the I/O Hardware Abstraction Layer which invokes the APIs and receives the call-back notifications. If I/O Hardware Abstraction Layer Module is not available, then the required functionality shall be stubbed.

RTE Module

The ICU driver shall perform data protection using SchM APIs. If the SchM is not available, then the required functionality shall be stubbed.

3.1.7.3. Configuration Parameter Dependency

The ICU Driver Depends on EcuM. Hence the parameter 'IcuChannelWakeupInfo' in the 'IcuWakeup' container of each channel refers to the path "/Renesas/EcucDefs_Icu/EcuM0/EcuMConfiguration0/EcuMCommonConfiguration0/EcuMWakeupSource_1".

The ICU Driver Depends on MCU for the clock source configuration. Hence the parameter 'IcuGTMClockRef' in the 'IcuGeneral' container refers to the path "/Renesas/EcucDefs_Msn/Mcu0/McuModuleConfiguration0/McuGTMClockSettingsConfig0"

3.1.7.4. Source Code Dependency

The following are the common dependent used files by the ICU Driver module:

Det.h,

MemMap.h,

Platform_Types.h,

Std_Types.h,

SchM_Icu.h,

Rte.h,

EcuM.h

EcuM_Cfg.h

EcuM_Cbk.h

Os.h

rh850_Types.h

3.1.7.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The table below will provide the common stubs to be used for ICU Driver component.

Table 3-7 : ICU Driver Component Common Stubs

Common Stubs	Path
Det	X1X\common_platform\generic\stubs\<Autosar Version>\Det
EcuM	X1X\common_platform\generic\stubs\<Autosar Version>\EcuM
Os	X1X\common_platform\generic\stubs\<Autosar Version>\Os

3.1.8. MCU Driver Component

3.1.8.1. Module Overview

The MCU Driver accesses the hardware features directly. The upper layers call the functionalities provided by the Driver. MCU component has functionalities related PLL Initialization, Clock Initialization & Distribution, RAM sections, Pre-Scaler Initializations, MCU Reduced Power Modes Activation and MCU Reset Activation & Reason.

The MCU Driver component is divided into the following sub modules based on the functionality required:

- Initialization
- Clock Initialization
- PLL Clock Distribution
- MCU Reduced Power Modes Activation
- RAM sections Initialization
- MCU Reset Activation & Reason
- Module Version Info

3.1.8.2. Module Dependency

DET

In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

DEM

Production errors will be reported to the Diagnostic Event Manager (DEM).

EcuM

The reference for the type of reset will be provided by the Mcu driver to the ECU State manager module.

OS

The MCU driver uses interrupts and therefore there is a dependency on the OS which configures the interrupt sources. If OS is not available, then the configuration of interrupt sources shall be stubbed.

RTE Module

The MCU driver shall perform data protection using SchM APIs. If the SchM is not available, then the required functionality shall be stubbed.

3.1.8.3. Configuration Parameter Dependency

None

3.1.8.4. Source Code Dependency

The following are the common dependent used files by the MCU Driver module:

Det.h,

Dem.h

MemMap.h,

Platform_Types.h,

Std_Types.h,

Rte.h,

SchM_Mcu.h

Os.h

rh850_Types.h

3.1.8.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The table below will provide the common stubs to be used for MCU Driver component.

Table 3-8 : MCU Driver Component Common Stubs

Common Stubs	Path
Det	X1X\common_platform\generic\stubs\<Autosar Version>\Det
Dem	X1X\common_platform\generic\stubs\<Autosar Version>\Dem
SchM	\X1X\common_platform\generic\stubs\<Autosar Version>\SchM
Os	X1X\common_platform\generic\stubs\<Autosar Version>\Os

3.1.9. GPT Driver Component

3.1.9.1. Module Overview

The GPT Driver Component provides services for GPT Driver Component Initialization, De-initialization, Setting starting and stopping a timer, getting elapsed and remaining time, setting GPT mode (one shot, continuous) and Disabling or Enabling the GPT notification. The GPT Driver Component is part of the Microcontroller Abstraction Layer (MCAL), the lowest layer of Basic Software in the AUTOSAR environment.

The GPT Driver Component is divided into GPT High Level Driver and GPT Low Level Driver to minimize the effort and to optimize the reuse of developed software on different platforms.

The GPT High Level Driver exports the APIs to the upper modules. All the references to specific microcontroller features and registers are provided in GPT Low Level Driver.

The GPT channel can be configured to either as continuous mode or one-shot mode. In continuous mode, the timers keep operating even after the target value is reached and it has multiple notifications (if enabled).

The ATOM sub modules in GTM consist of ATOM0, ATOM1 and ATOM2 are used in GPT Driver Component to generate timeout periods.

The GPT Driver component should provide following services based on the functions performed by the GPT Driver:

- Initialization: Provides the service to initialize the timer control registers and interrupt registers
- De-Initialization: Provides the service to reinitialize the timer registers and to stop the channels that are running
- Reading of timer values: Provides services for reading the elapsed time after the timer is started or Service for reading the remaining time before the next timeout
- Start/Stop timer: Provides the service to start/stop the requested timer channel
- Set mode for GPT(continuous, one shot): Provides services for the user to select the mode
- Notification services: Provides services for the user to enable or disable the notification for every timeout
- Wakeup Services: Provides services for the user to enable or disable the wakeup notification.
- Get version information: Provides the service for the user to read module version

3.1.9.2. Module Dependency

The dependency of GPT Driver on other modules and the required implementation is briefed as follows:

DET

In development mode the Development Error Tracer will be called whenever this module encounters a development error.

DEM

The Diagnostic Event manager (DEM) will be called whenever this module encounters a production relevant error.

MCU Driver

The Microcontroller Unit Driver (MCU Driver) is primarily responsible for initializing the GTM CMU clock sources.

EcuM

The GPT driver shall do the reporting of wakeup interrupts to the EcuM. If the EcuM is not available, then the required functionality shall be stubbed.

RTE

The Run time Environment (RTE) module will be called whenever a critical section protection function is called.

OS

The GPT driver uses interrupts and therefore there is a dependency on the OS which configures the interrupt sources. If OS is not available, then the configuration of interrupt sources shall be stubbed.

3.1.9.3. Configuration Parameter Dependency

The GPT Driver Depends on EcuM. Hence the parameter 'GptWakeupSourceRef' in the 'GptWakeupConfiguration' container of each channel refers to the path "/Renesas/EcucDefs_Gpt/EcuM0/EcuMConfiguration0/EcuMCommonConfiguration0/EcuMWakeupSource_1".

The GPT Driver Depends on the MCU Driver for clock source configuration. Hence the parameter GptGTMClockRef in the container GptDriverConfiguration refers to the path "/Renesas/EcucDefs_Msn/Mcu0/McuModuleConfiguration0/McuGTMClockSettingsConfig0".

3.1.9.4. Source Code Dependency

The following are the common dependent used files by the GPT Driver module:

Det.h,

Dem.h,

MemMap.h,

Platform_Types.h,

Std_Types.h,

SchM_Gpt.h,

Rte.h,

Os.h

EcuM.h

EcuM_Cbk.h

rh850_Types.h

3.1.9.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The table below will provide the common stubs to be used for GPT Driver component.

Table 3-9 : GPT Driver Component Common Stubs

Common Stubs	Path
Det	X1X\common_platform\generic\stubs\<Autosar Version>\Det
Dem	X1X\common_platform\generic\stubs\<Autosar Version>\Dem
EcuM	X1X\common_platform\generic\stubs\<Autosar Version>\EcuM
Os	X1X\common_platform\generic\stubs\<Autosar Version>\Os

3.1.10. WDG Driver Component

3.1.10.1. Module Overview

Watchdog Driver module provides the services for initializing, changing the operation mode and triggering the watchdog.
The Watchdog Driver accesses the microcontroller hardware directly and Interface communicates with the application.

The Watchdog Driver component is composed of following modules:

- Watchdog Driver Initialization module
- Watchdog Driver SetMode module
- Watchdog Driver Trigger module
- Watchdog Driver Version info module

3.1.10.2. Module Dependency

DET

In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

DEM

Production errors will be reported to the Diagnostic Event Manager (DEM).

RTE

The Run time Environment (RTE) module will be called whenever a critical section protection function is called.

MCU Driver

The count which indicates the number of times the watchdog should be triggered for a trigger condition’s timeout value depends on WDTATCLKI,

hence MCU reference path will be provided in the parameter definition file.

OS

The WDG driver uses interrupts and therefore there is a dependency on the OS which configures the interrupt sources. If OS is not available, then the configuration of interrupt sources shall be stubbed.

3.1.10.3. Configuration Parameter Dependency

The Watchdog Driver Depends on the MCU Driver for clock value. Hence the parameter 'WdgClockRef' in the 'WdgGeneral' container refers to the path

"/Renesas/EcucDefs_Msn/Mcu0/McuModuleConfiguration0/McuGTMClock SettingsConfig0"

3.1.10.4. Source Code Dependency

The following are the common dependent used files by the WDG Driver module:

Det.h,

Dem.h

WdgIf_Types.h

MemMap.h,

Platform_Types.h,

Rte.h

Std_Types.h

Os.h

rh850_Types.h

3.1.10.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

"X1X\common_platform\generic\stubs\<Autosar Version>"

The table below will provide the common stubs to be used for WDG Driver component.

Table 3-10 : WDG Driver Component Common Stubs

Common Stubs	Path
Det	X1X\common_platform\generic\stubs\<Autosar Version>\Det
Dem	X1X\common_platform\generic\stubs\<Autosar Version>\Dem
WdgIf	X1X\common_platform\generic\stubs\<Autosar Version>\WdgIf
Os	X1X\common_platform\generic\stubs\<Autosar Version>\Os

3.1.11. LIN Driver Component

3.1.11.1. Module Overview

The LIN driver is part of the microcontroller abstraction layer (MCAL), performs the hardware access and offers hardware independent API to the upper layer. Several LIN Controllers is controlled by the LIN Driver as long as they belong to the same LIN Hardware Unit.

The LIN Driver software component shall provide the following main features:

The LIN Driver Component fulfills requirements of upper layer communication components with respect to Initialization, Transmit and Receive confirmation and Wakeup notification to ECU State Manager.

3.1.11.2. Module Dependency

The dependency of LIN Driver on other modules and the required implementation is briefed as follows:
DET

In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

DEM

The Diagnostic Event manager (DEM) will be called whenever LIN module encounters a production relevant error.

MCU Driver

LIN driver depend on MCU Driver for the setting of channel clock.

ECU State Manager

If controller wake-up event is detected LIN Driver Component provides the call out notification functionality to the EcuM.

OS

The LIN driver uses interrupts and hence there is a dependency on the OS, which configures the interrupt sources. If OS is not available, then the configuration of interrupt sources shall be stubbed.

3.1.11.3. Configuration Parameter Dependency

The LIN Driver Depends on the MCU Driver for clock value. Hence the parameter ‘LinClockRef’ in the ‘LinChannel’ container refers to the path “/Renesas/EcucDefs_Mcu/Mcu/McuModuleConfiguration0/McuClockSettingConfig0/McuClockReferencePoint0”

3.1.11.4. Source Code Dependency

The following are the common dependent used files by the LIN Driver module:

Det.h,

Dem.h,

EcuM.h,

EcuM_Cfg.h,

EcuM_Cbk.h,

Dem.h

MemMap.h,

Platform_Types.h,

Std_Types.h,

Rte.h

SchM_Lin.h

rh850_Types.h

3.1.11.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The tables below will provide the common and port specific stubs to be used for LIN Driver component

Table 3-11 : LIN Driver Component Common Stubs

Common Stubs	Path
Det	\X1X\common_platform\generic\stubs\<Autosar Version>\Det
EcuM	\X1X\common_platform\generic\stubs\<Autosar Version>\EcuM
SchM	\X1X\common_platform\generic\stubs\<Autosar Version>\SchM
Dem	\X1X\common_platform\generic\stubs\<Autosar Version>\Dem
Os	\X1X\common_platform\generic\stubs\<Autosar Version>\Os

Table 3-12 : LIN Driver Component Specific Stubs

Lin Specific Stubs	Path
Mcu	\X1X\common_platform\generic\stubs\<Autosar Version>\Mcu
SchM	\X1X\common_platform\generic\stubs\<Autosar Version>\SchM
EcuM	X1X\common_platform\generic\stubs\<Autosar Version>\EcuM
Dem	X1X\common_platform\generic\stubs\<Autosar Version>\Dem

3.1.12. FR Driver Component

3.1.12.1. Module Overview

The FR driver provides services for FlexRay communication.

The FR driver component provides the following functionalities:

- To initialize the FlexRay communication controllers
- To start, halt or abort the communication
- To configure the channel for sending the wakeup pattern and to transmit the wakeup pattern on the configured FlexRay channel
- To get the current POC status of CC
- To get the synchronization state of CC and to adjust the global time of a FlexRay CC to an external clock source
- To transmit the frames on the FlexRay channels
- To receive the frames transmitted on the FlexRay channels
- To get the current cycle and macrotick offset value of CC
- To set the value for absolute timer interrupt and to stop the absolute timer
- To enable/disable the absolute timer interrupt. To reset the interrupt condition of absolute timer interrupt and to get the status of absolute timer interrupt
- To get the Channel status, Clock Correction, Number of startup frames, Clock Correction, Sync frame list and wakeup Rx status of CC
- To get the Nm Vector Information received on CC
- To send CC to ALLSLOTS and ALLOW_COLDSTART modes
- To reconfigure or disable an Lpdu in run time.

3.1.12.2. Module Dependency

The dependency of FR Driver on other modules and the required implementation is briefed as follows:

DET

In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

DEM

The Diagnostic Event manager (DEM) will be called whenever FR module encounters a production relevant error.

OS

The FR driver uses interrupts and hence there is a dependency on the OS, which configures the interrupt sources. If OS is not available, then the configuration of interrupt sources shall be stubbed.

3.1.12.3. Configuration Parameter Dependency

None

3.1.12.4. Source Code Dependency

The following are the common dependent used files by the FR Driver module:

Det.h,

Dem.h

MemMap.h,

Platform_Types.h,

Std_Types.h,

Rte.h

SchM_Fr_59_Renesas.h

rh850_Types.h

3.1.12.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The tables below will provide the common stubs to be used for FR Driver component

Table 3-13 : FR Driver Component Common Stubs

Common Stubs	Path
Det	\X1X\common_platform\generic\stubs\<Autosar Version>\Det
SchM	\X1X\common_platform\generic\stubs\<Autosar Version>\SchM
Dem	\X1X\common_platform\generic\stubs\<Autosar Version>\Dem
Os	\X1X\common_platform\generic\stubs\<Autosar Version>\Os

3.1.13. RAMTST Driver Component

3.1.13.1. Module Overview

The RAMTST driver is part of the microcontroller abstraction layer (MCAL), performs the hardware access and offers hardware independent API to the upper layer. RAMTST driver provides the feature to test the physical health of RAM cells with different algorithms. If any fault is detected, notifications are provided to upper layers to take necessary actions as well as Error Corrections which are possible are done. It is not intended to test the contents of the RAM. RAM used for registers is also tested.

A RAM Test may be called synchronously by the test environment (called “foreground test”) or may be called in a cyclic manner by an OS task or other cyclic calling method (called “background test”). The test environment may select test parameters, start and stop the test, and get status reports.

3.1.13.2. Module Dependency

The dependency of RAMTST Driver on other modules and the required implementation is briefed as follows:

DET

In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

DEM

The Diagnostic Event manager (DEM) will be called whenever RAMTST module encounters a production relevant error.

RTE Module

The RAMTST driver shall perform data protection using SchM APIs.

3.1.13.3. Configuration Parameter Dependency

None.

3.1.13.4. Source Code Dependency

The following are the common dependent used files by the RAMTST Driver module:

Det.h,

Dem.h

Dem_Cfg.h

MemMap.h,

Platform_Types.h,

Std_Types.h and

SchM_RamTst.h

3.1.13.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The tables below will provide the common stubs to be used for RAMTST Driver component

Table 3-14 : RAMTST Driver Component Common Stubs

Common Stubs	Path
Det	\X1X\common_platform\generic\stubs\<Autosar Version>\Det
SchM	\X1X\common_platform\generic\stubs\<Autosar Version>\SchM
Dem	\X1X\common_platform\generic\stubs\<Autosar Version>\Dem

3.1.14. CORTST Driver Component

3.1.14.1. Module Overview

The CORTST module provides services for configuring, starting, polling, terminating and notifying the application about Core Test results. It also provides services for returning test results in a predefined way. Furthermore it provides several tests to verify dedicated core functionality like e.g. general purpose registers or Arithmetical and Logical Unit (ALU).

It is up to the user of Core Test Driver API to choose suitable test combination and a scheduled execution order to fulfill the safety requirements of the system. The behavior of those services is asynchronous or synchronous.

The functional parameters of CORTST software components are statically configurable to fit as far as possible to the real needs of each ECU.

The CORTST Driver Component is divided into the following sub modules based on the functionality required:

- Initialization and De-Initialization
- Abort the core test operation
- Getting the execution status of the CORTST driver
- Getting Fore ground and Back ground Test result and Test Signature value
- Foreground Test and Background tests
- Module version information

3.1.14.2. Module Dependency

The dependency of CORTST Driver on other modules and the required implementation is briefed as follows:

DET

In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

DEM

The Diagnostic Event manager (DEM) will be called whenever CORTST module encounters a production relevant error.

RTE

The Run time Environment (RTE) module will be called whenever a critical section protection function is called.

OS

The CORTST driver uses interrupts and hence there is a dependency on the OS, which configures the interrupt sources. If OS is not available, then the configuration of interrupt sources shall be stubbed.

3.1.14.3. Configuration Parameter Dependency

None

3.1.14.4. Source Code Dependency

The following are the common dependent used files by the CORTST Driver module:

Det.h,

Dem.h

Os.h

MemMap.h,

Platform_Types.h,

Std_Types.h,

Rte.h

SchM_CorTst.h

rh850_Types.h

3.1.14.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The tables below will provide the common stubs to be used for CORTST Driver component

Table 3-15 : CORTST Driver Component Common Stubs

Common Stubs	Path
Det	\X1X\common_platform\generic\stubs\<Autosar Version>\Det
SchM	\X1X\common_platform\generic\stubs\<Autosar Version>\SchM
Dem	\X1X\common_platform\generic\stubs\<Autosar Version>\Dem
Os	\X1X\common_platform\generic\stubs\<Autosar Version>\Os

3.1.15. FLSTST Driver Component

3.1.15.1. Module Overview

The FLSTST Driver Component provides the following services:

- FLSTST Driver Component initialization
- De-initialization

• Reading the internal state of FLSTST Output signal

• Setting the FLSTST Output to Idle state

• Disabling/Enabling the FLSTST signal edge notification

3.1.15.2. Module Dependency

The dependency of FLSTST Driver on other modules and the required implementation is briefed as follows:

DET

In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

DEM

The Diagnostic Event manager (DEM) will be called whenever FLSTST module encounters a production relevant error.

RTE

The Run time Environment (RTE) module will be called whenever a critical section protection function is called.

3.1.15.3. Configuration Parameter Dependency

None

3.1.15.4. Source Code Dependency

The following are the common dependent used files by the FLSTST Driver module:

Det.h,

Dem.h

MemMap.h,

Platform_Types.h,

Std_Types.h,

Rte.h

SchM_FlsTst.h

rh850_Types.h

3.1.15.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The tables below will provide the common stubs to be used for FLSTST Driver component

Table 3-16 : FLSTST Driver Component Common Stubs

Common Stubs	Path
Det	\X1X\common_platform\generic\stubs\<Autosar Version>\Det
SchM	\X1X\common_platform\generic\stubs\<Autosar Version>\SchM
Dem	\X1X\common_platform\generic\stubs\<Autosar Version>\Dem

3.1.16. ETH Driver Component

3.1.16.1. Module Overview

The ETH Driver component can be divided into following sub components based on the functions performed by the ETH Driver:

- Driver Initialization
- Controller Initialization
- Setting and getting the Controller Mode
- Getting the MAC Address of the Ethernet Controller
- Writing MII Interface register
- Reading MII Interface register
- Getting the Counter State
- Provide Transmit Buffer Access
- Transmit Functionality
- Receive Functionality
- Transmit confirmation
- Frame reception interrupt handling
- Frame Transmission Interrupt handling
- Module version information
- Address Filtering
- Magic Packet detection

3.1.16.2. Module Dependency

The dependency of ETH Driver on other modules and the required implementation is briefed as follows:

DET

In development mode the Development Error Tracer (DET) will be called whenever this module encounters a development error.

RTE

The Run time Environment (RTE) module will be called whenever a critical

section protection function is called.

3.1.16.3. Configuration Parameter Dependency

None

3.1.16.4. Source Code Dependency

The following are the common dependent used files by the ETH Driver module:

Det.h,

MemMap.h,

Platform_Types.h,

Std_Types.h,

Rte.h

SchM_Eth.h

Os.h

rh850_Types.h

3.1.16.5. Stubs

Stubs are categorized as common stub.

The common stubs are common for all the X1X family and are available in the path

“X1X\common_platform\generic\stubs\<Autosar Version>”

The table below will provide the common stubs to be used for ETH Driver component.

Table 3-17 : ETH Driver Component Common Stubs

Common Stubs	Path
Det	\X1X\common_platform\generic\stubs\<Autosar Version>\Det
SchM	\X1X\common_platform\generic\stubs\<Autosar Version>\SchM
Os	X1X\common_platform\generic\stubs\<Autosar Version>\Os

3.2 RH850 Macros Definition:

The driver supports both Supervisor mode and User mode.

To provide the provision to the user, to adapt the Driver to operate either in Supervisor/User Mode the IMRx/ICxxx register is moved to OS Module.

The macros provided in Table 3-17, available in rh850_types.h, should be used as mentioned below to switch modes.

- To operate the driver in User Mode: User must modify these macros.

- To operate the driver in Supervisor Mode: No modification is required.

Table 3-18 : Macro to perform write operation, on write enabled Register

Macro Name	Description	Input Parameter
RH850_SV_MODE_ICR_OR	This Macro performs supervisor mode (SV) write enabled Register ICxxx register writing which involves an OR operation.	SIZE : Register Access Size ADDR : Register address VAL : Value to be written to the register
RH850_SV_MODE_ICR_AND	This Macro performs supervisor mode(SV) write enabled Register ICxxx register writing which involves an AND operation.	SIZE : Register Access Size ADDR : Register address VAL : Value to be written to the register
RH850_SV_MODE_ICR_WRITE_ONLY	This Macro performs supervisor mode(SV) write enabled Register ICxxx register direct writing operation.	SIZE : Register Access Size ADDR : Register address VAL : Value to be written to the register
RH850_SV_MODE_IMR_OR	This Macro performs supervisor mode(SV) write enabled Register IMR register writing which involves an OR operation	SIZE : Register Access Size ADDR : Register address VAL : Value to be written to the register

RH850_SV_MODE_IMR_AND	This Macro performs supervisor mode(SV) write enabled Register IMR register writing which involves an AND operation	SIZE : Register Access Size ADDR : Register address VAL : Value to be written to the register
RH850_SV_MODE_IMR_WRITE_ONLY	This Macro performs supervisor mode (SV) write enabled Register IMR register direct writing operation.	SIZE : Register Access Size ADDR : Register address VAL : Value to be written to the register

3.3 ICxxx Registers Setting for TBxxx-Bit

- The ICxxx register's TBxxx-Bit is used to select the way to determine the interrupt vector.
  0: Direct jumping to an address determined from the level of priority
  1: Reference to a table.

- MCAL Driver does not set TBxxx bit. Hence user has to take care of setting TBxxx-Bit before initializing MCAL driver.

Revision History

Sl.No.	Description	Version	Date
1.	Initial Version	1.0.0	31-Jan-2013
2	Following changes are made: 1. Removed CAN and FEE driver components. 2. Updated GPT, ICU and PWM for GTM. 3. Updated Chapter 2 “REFERENCE DOCUMENTS”. 4. Added FR, RAMTST, CORTST, FLSTST and ETH Driver components in Chapter 3 “AUTOSAR MODULES” 5. Removed all the information related to Autosar version 3.2.2	1.0.1	26-Apr-2016
3	The following changes are made: 1. Updated section Configuration Parameter Dependency for GPT, ICU and PWM. 2. Added Dem for ADC, PWM, PORT, DIO, SPI, GPT. 3. Removed details regarding Dem from the section 3.1.16, ETH. 4. Updated R number	1.0.2	29-Nov-2016
4	The following changes are made: 1. Updated R number of the document 2. Notice and copyright information are updated.	1.0.3	05-May-2017

AUTOSAR Modules Overview User's Manual
Version 1.0.3

Publication Date: Rev.1.01, May 05, 2017

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.
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 Modules Overview

User's Manual

Renesas Electronics Corporation

R20UT3827EJ0101