User's Manual

AUTOSAR Modules Overview

User's Manual

Version 1.0.9

Target Device:
RH850/P1x

Renesas Electronics
www.renesas.com

Rev.1.00 Jul 2016

Notice

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

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

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

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

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority- owned subsidiaries.
(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Abbreviations and Acronyms

Abbreviation / Acronym	Description
ADC	Analog to Digital Converter
API	Application Programming Interface
ANSI	American National Standards Institute
AUTOSAR	AUTomotive Open System ARchitecture
CAN	Controller Area Network
DEM	Diagnostic Event Manager
DET/Det	Development Error Tracer
DIO	Digital Input Output
FEE	Flash EEPROM Emulation
FLS	FLaSh Driver
FSL	Flash Self programming Library
FR	Flex-Ray
GPT	General Purpose Timer
ICU	Input Capture Unit
LIN	Local Interconnect Network
MCAL	MicroController Abstraction Layer
MCU	MicroController Unit
PWM	Pulse Width Modulation
SPI	Serial Peripheral Interface
TAU	Timer Array Unit
WDG	WatchDog driver

Definitions

Term	Represented by
Sl. No.	Serial Number
<Autosar Version>	3.2.2 when tested for R3.2.2
	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.

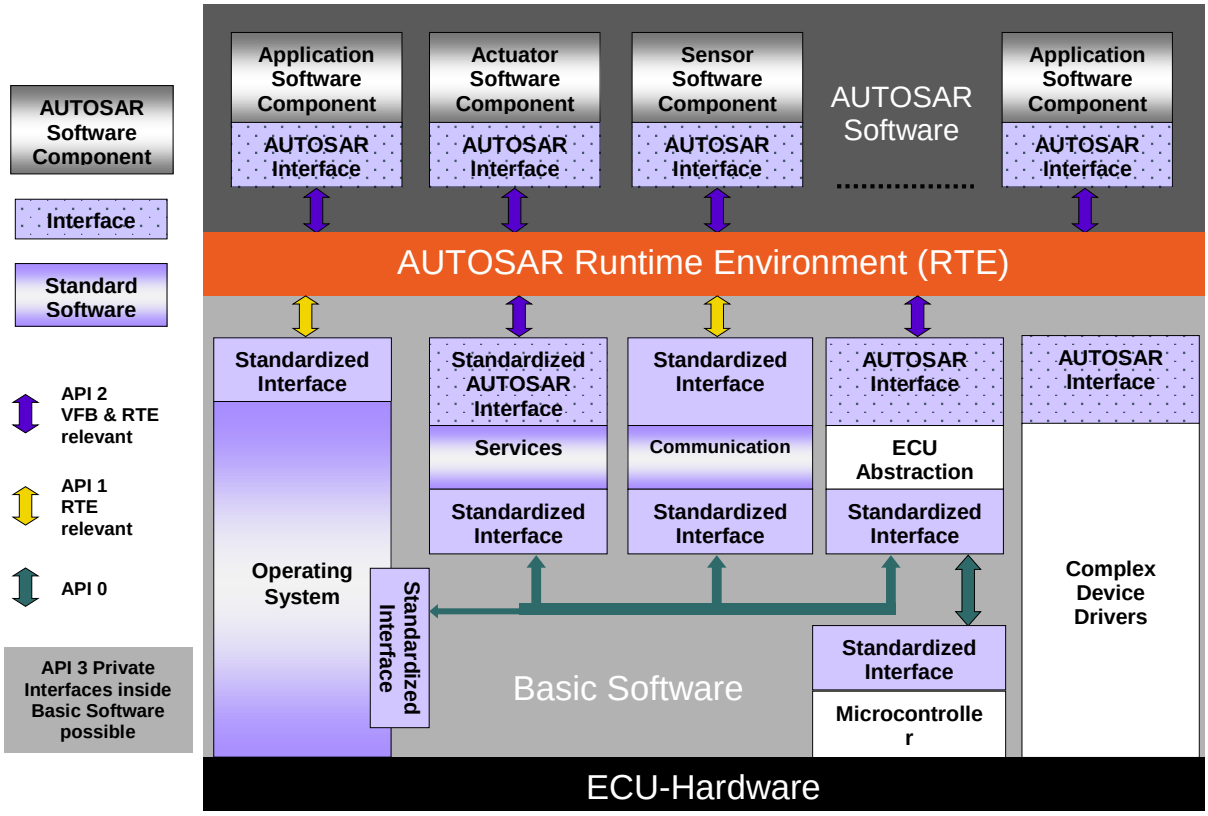

Figure 1-1 : System Overview of the AUTOSAR Architecture Layer

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 R3.2.2	Version
1.	Specification of ADC Driver (AUTOSAR_SWS_ADC_Driver.pdf)	3.0.3
2.	Specification of CAN Driver (AUTOSAR_SWS_CAN_Driver.pdf)	2.5.0
3.	Specification of PWM Driver (AUTOSAR_SWS_PWM_Driver.pdf)	2.3.0
4.	Specification of PORT Driver (AUTOSAR_SWS_Port_Driver.pdf)	3.2.0
5.	Specification of Flash EEPROM Emulation (AUTOSAR_SWS_Flash_EEPROMEmulation.pdf)	1.4.0
6.	Specification of DIO Driver (AUTOSAR_SWS_DIO_Driver.pdf)	2.4.0
7.	Specification of Module Flash Driver (AUTOSAR_SWS_Flash_Driver.pdf)	2.4.0
8.	Specification of SPI Handler/Driver (AUTOSAR_SWS_SPI_Handler_Driver.pdf)	2.4.0
9.	Specification of ICU Driver (AUTOSAR_SWS_ICU_Driver.pdf)	3.2.0
10.	Specification of MCU Driver (AUTOSAR_SWS_MCU_Driver.pdf)	2.5.0
11.	Specification of GPT Driver (AUTOSAR_SWS_GPT_Driver.pdf)	2.2.2
12.	Specification of Watchdog Driver (AUTOSAR_SWS_Watchdog_Driver.pdf)	2.3.0
13.	Specification of LIN Driver (AUTOSAR_SWS_LIN_Driver.pdf)	1.5.0

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 CAN Driver (AUTOSAR_SWS_CANDriver.pdf)	4.0.0
3.	Specification of PWM Driver (AUTOSAR_SWS_PWMDriver.pdf)	2.5.0
4.	Specification of PORT Driver (AUTOSAR_SWS_PortDriver.pdf)	3.2.0
5.	Specification of Flash EEPROM Emulation (AUTOSAR_SWS_Flash_EEPROMEmulation.pdf)	2.0.0
6.	Specification of DIO Driver (AUTOSAR_SWS_DIODriver.pdf)	2.5.0
7.	Specification of Module Flash Driver (AUTOSAR_SWS_FlashDriver.pdf)	3.2.0
8.	Specification of SPI Handler/Driver (AUTOSAR_SWS_SPI_HandlerDriver.pdf)	3.2.0
9.	Specification of ICU Driver (AUTOSAR_SWS_ICUDriver.pdf)	4.2.0
10.	Specification of MCU Driver (AUTOSAR_SWS_MCUDriver.pdf)	3.2.0
11.	Specification of GPT Driver (AUTOSAR_SWS_GPTDriver.pdf)	3.2.0
12.	Specification of Watchdog Driver (AUTOSAR_SWS_WatchdogDriver.pdf)	2.5.0
13.	Specification of LIN Driver (AUTOSAR_SWS_LINDriver.pdf)	1.5.0

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

FEE

DIO

FLS

SPI

ICU

MCU

GPT

WDG

CAN

LIN

FR

FLSTST

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.

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

The ADC Driver Depends on the MCU Driver for clock value. Hence the parameter ‘AdcClockRef’ in the ‘AdcHwUnit’ container refers to the path “/ Renesas/Mcu0/McuModuleConfiguration0/McuClockSettingConfig0”.

3.1.1.4. Source Code Dependency

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

Det.h,

MemMap.h,

Platform_Types.h,

Std_Types.h,

SchM_Adc.h

Rte.h and

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 and port specific 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

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.

Timers TAUA, TAUB, TAUC and TAUJ are used in PWM Driver Component to generate variable PWM output. These timers can operate in Master mode as well as Slave mode depending on the configuration.

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.

MCU Driver

The Microcontroller Unit Driver (MCU Driver) is primarily responsible for initializing and controlling the chip's internal clock sources and clock pre-scalars.

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

None

3.1.2.4. Source Code Dependency

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

Det.h,

MemMap.h,

Platform_Types.h,

Std_Types.h,

SchM_Pwm.h

Rte.h and

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	Pat
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

3.1.3. 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 PORT 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,

MemMap.h,

Platform_Types.h,

Std_Types.h,

SchM_Port.h

Rte.h and

Dem.h

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	Pat
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. FEE Software Component

3.1.4.1. Module Overview

The FEE software component of the Memory Hardware Abstraction interface provides the emulation access to flash driver. The FEE software component layer provides the wrapper for the FEE EEPROM Emulation library, which comprises of EEPROM emulation layer, Data Flash Access layer and Flash control hardware. The FEE software component provides services for reading from and writing to flash memory, erasing and invalidating the flash memory.

The FEE Software Component provides services for:

- Initialization
- Reading and Writing to the memory
- Invalidating the memory
- Cancellation of request
- Reading status and result information
- Module version information

3.1.4.2. Module Dependency

The dependency of FEE 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 FEE module encounters a development error.

DEM

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

RTE

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

3.1.4.3. Configuration Parameter Dependency

None

3.1.4.4. Source Code Dependency

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

Det.h,

Dem.h,

MemIf.h,

NvM.h,

MemMap.h,

Platform_Types.h,

Std_Types.h,

SchM_Fee.h and

Rte.h

rh850_Types.h

3.1.4.5. 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 FEE Software component.

Table 3-4 : FEE 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
NvM	X1X\common_platform\generic\stubs\<Autosar Version>\NvM
MemIf	X1X\common_platform\generic\stubs\<Autosar Version>\MemIf

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

3.1.5. DIO Driver Component

3.1.5.1. 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.

3.1.5.2. 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.

PORT driver

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

3.1.5.3. Configuration Parameter Dependency

None

3.1.5.4. Source Code Dependency

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

Det.h,

MemMap.h,

Platform_Types.h and

Std_Types.h

3.1.5.5. 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 3-5 : DIO Driver Component Common Stubs

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

3.1.6. FLS Software Component

3.1.6.1. Module Overview

The FLS software component provides services for reading, writing, comparing and erasing flash memory. The FLS Component layer provides the wrapper for the Renesas Self Programming Library, which comprises of API for erase/write data to on-chip flash memory of the device. This means the FLS component makes use of the FSL, which is an underlying software library contains FSL functions to perform the activities like accessing and programming the on-chip flash hardware. FSL offers all functions and commands necessary to reprogram the application in a user friendly C language interface. The FSL basically consists of wrapper functions to the FLS routines.

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

3.1.6.2. 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.

3.1.6.3. Configuration Parameter Dependency

None

3.1.6.4. 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

3.1.6.5. 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 FLS Software component.

Table 3-6 : FLS Software Component Common Stubs

Common Stubs	Pa
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

3.1.7. SPI Driver Component

3.1.7.1. 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

3.1.7.2. 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.

MCU Driver

The configuration of SPI module for jobs contains the references to the MCU module for the input clock frequency for the SPI HW Unit. Hence, SPI baud rate depends on the frequency set in the MCU 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.

3.1.7.3. 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/Mcu0/McuModuleConfiguration0/McuClockSettingConfig0”.

3.1.7.4. 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 and

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

Table 3-7 : 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

Table 3-8 : SPI Driver Component Port Specific Stubs

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

3.1.8. ICU Driver Component

3.1.8.1. 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 period 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, timer operation modes and external interrupts have to be selected depending on ICU measurement mode. For the X1X microcontroller generation following concepts will be considered:

- Using TAU A and TAU B for Edge Counting Measurement mode
- Using TAU A, TAU B and TAU J for Time Stamping Measurement mode
- Using TAU A, TAU B and TAU J for Signal Measurement mode
- Using External Interrupts 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 not for any of the Timer channels. The remaining three measurement modes viz. Edge Counting, Time Stamping and Signal Measurement should be configured only for the timer 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
- Notification
- Signal Measurement
- Signal Activation and State Information
- Version Information

Various timers can be started at the same time by setting the related enable bits. The input signal can be split from one port pin to two consecutive TAU inputs, which allows the signal for period or duty cycle measurement to be fed into only one port pin.

3.1.8.2. Module Dependency

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

MCU Driver

The ICU Driver depends on MCU for the setting of system clock and PLL and the length of the timer ticks depends on the clock settings made in MCU module. If MCU module is not available, the functionality of system clock and PLL settings shall be stubbed.

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.

3.1.8.3. Configuration Parameter Dependency

The ICU Driver Depends on EcuM. Hence the parameter 'IcuChannelWakeupInfo' in the 'IcuWakeup' container of each channel refers to the path "/AUTOSAR/Ecudefs_EcuM/EcuMConfiguration_1/ EcuMWakeupSource_1".

3.1.8.4. 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 and

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 to be used for ICU Driver component.

Table 3-9 : ICU Driver Component Common Stubs

Common Stubs	P
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

3.1.9. MCU Driver Component

3.1.9.1. 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

3.1.9.2. 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.

3.1.9.3. Configuration Parameter Dependency

None

3.1.9.4. 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

3.1.9.5. 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 MCU Driver component.

Table 3-10 : MCU Driver Component Common Stubs

Common Stubs	Pat
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

3.1.10. GPT Driver Component

3.1.10.1. 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).

Timers OSTM, TAUA TAUB, TAUC and TAUJ 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

3.1.10.2. 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.

IO Hardware Abstraction Layer

The GPT 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

MCU Driver

The GPT Driver component depends on MCU module for the setting of system clock, prescaler(s) and PLL. Thus any change in the system clock (For example, PLL On -> PLL Off) also affects the clock settings of GPT hardware. If MCU module is not available, the functionality of system clock prescaler(s) and PLL settings shall be stubbed.

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.

3.1.10.3. Configuration Parameter Dependency

The GPT Driver Depends on EcuM. Hence the parameter 'GptWakeupSourceRef' in the 'GptWakeupConfiguration' container of each channel refers to the path "/AUTOSAR/EcuDefs_EcuM/ EcuMConfiguration_1/ EcuMWakeupSource_1".

The GPT Driver Depends on the MCU Driver for clock value. Hence the parameter GptTauUnitClkRefPoint in the container GptTaUnit refers to the path "/Renesas/Mcu0/McuModuleConfiguration0/McuClockSettingConfig0".

3.1.10.4. Source Code Dependency

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

Det.h,

MemMap.h,

Platform_Types.h,

Std_Types.h,

SchM_Gpt.h,

Rte.h,

Os.h

EcuM_Cfg.h,

EcuM.h and

EcuM_Cbk.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 GPT Driver component.

Table 3-11 : 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

3.1.11. WDG Driver Component

3.1.11.1. Module Overview

To minimize the effort and to optimize the reuse of developed software, the Watchdog interface will invoke the corresponding drivers in case when multiple drivers exist.

In case of more than one Watchdog device and Watchdog Driver (both internal software Watchdog and external hardware Watchdog) is used on an ECU, Watchdog Interface module allows the upper layer to select the correct Watchdog Driver and Watchdog device while retaining the API and functionality of the underlying driver.

The Watchdog Driver architectural design is shown in the above Figure. 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

3.1.11.2. 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.

3.1.11.3. Configuration Parameter Dependency

None

3.1.11.4. 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

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

Table 3-12 : 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

3.1.12. CAN Driver Component

3.1.12.1. Module Overview

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

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

The CAN Driver Component fulfills requirements of upper layer communication components with respect to Initialization, Transmit confirmation, Receive indication, BusOff to CAN Interface layer and Wakeup

notification to ECU State Manager.

3.1.12.2. Module Dependency

The dependency of CAN 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.

MCU Driver

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

CAN Interface

The CAN Driver Component provides the following functionalities to the CAN Interface layer

- To change the operation mode of the controllers.
- To Enable/Disable the Controller Interrupts
- To process the L-PDU Transmission

ECU State Manager

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

OS

The CAN 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

The CAN Driver Depends on the MCU Driver for clock value. Hence the parameter ‘CanControllerClock’ in the ‘CanController’ container refers to the path “/Renesas/Mcu0/McuModuleConfiguration0/McuClockSettingConfig0”.

3.1.12.4. Source Code Dependency

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

Det.h,

CanIf_Cbk.h,

EcuM_Cfg.h,

EcuM_Cbk.h,

Dem.h

MemMap.h,

Platform_Types.h,

Std_Types.h,

Rte.h and

SchM_Can.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 and port specific stubs to be used for CAN Driver component

Table 3-13 : CAN 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
CanIf	\X1X\common_platform\generic\stubs\<Autosar Version>\CanIf
Os	\X1X\common_platform\generic\stubs\<Autosar Version>\Os

Table 3-14 : CAN Driver Component Port Specific Stubs

Port Specific Stubs	Path
Mcu	\X1X\common_platform\generic\stubs\<Autosar Version>\Mcu

3.1.13. LIN Driver Component

3.1.13.1. 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.

3.1.13.2. 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.

3.1.13.3. Configuration Parameter Dependency

The LIN Driver Depends on the MCU Driver for clock value. Hence the parameter ‘LinChannelClockRef’ in the ‘LinChannel’ container refers to the path

For RLIN2:
“/Renesas/EcucDefs_Mcu/Mcu0/McuModuleConfiguration0/McuClockSettingConfig0/McuIsoLin0”

For RLIN3:
“/Renesas/EcucDefs_Mcu/Mcu0/McuModuleConfiguration0/McuClockSettingConfig0/McuIsoLin30”

3.1.13.4. Source Code Dependency

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

Det.h,

EcuM.h,

EcuM_Cfg.h,

EcuM_Cbk.h,

EcuM_Types.h,

Dem.h

MemMap.h,

Platform_Types.h,

Std_Types.h,

Rte.h and

SchM_Lin.h

rh850_Types.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 and port specific stubs to be used for LIN Driver component

Table 3-15 : 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 3-16 : LIN Driver Component Port Specific Stubs

Port Specific Stubs	Path
Mcu	\X1X\common_platform\generic\stubs\<Autosar Version>\Mcu

3.1.14. FR Driver Component

3.1.14.1. 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.

3.1.14.2. 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.

3.1.14.3. Configuration Parameter Dependency

None

3.1.14.4. 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 and

SchM_Fr_59_Renesas.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 and port specific stubs to be used for FR Driver component

Table 3-17 : 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

3.1.15. 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 and

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 and port specific stubs to be used for FLSTST Driver component

Table 3-18 : 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

3.2 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 3-19 : Macros 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_ONL Y	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_ON LY	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. On front page F1L is replaced by X1x.	1.0.1	24-Jan-2014
3.	Following changes are made: 1. New section 3.1.13 is added for LIN driver component. 2. Alignment is done in throughout the file.	1.0.2	29-Jan-2014
4.	Following change is made: 1. Canif stub is removed from table 13-15 and accordingly description is updated in section 13.1.13.1.	1.0.3	30-Jan-2014
5.	Following changes are made: 1. R-number is updated for document. 2. Alignment is updated as per template.	1.0.4	08-Apr-2014
6.	Following changes are made: 1. Section 3.1 is updated for source code dependency files 2. New Section 3.2 is added for RH850 Macro definition	1.0.5	17-Jun-2014
7.	Following changes are made: 1. New Section 3.3 is added for adding information about ICxxx registers.	1.0.6	17-Jul-2014
8.	Following changes are made: 1. Copyright information is updated. 2. Document is updated as per template.	1.0.7	09-Aug-2014
9.	Following changes are made: 1. Copyright information is updated. 2. Added FR and FLSTST	1.0.8	31-Mar-2016
10	Following changes are made: 1. R-Number has been update. 2. Table 3-12 alignment is corrected.	1.0.9	14-Jul-2016

AUTOSAR Modules Overview User's Manual
Version 1.0.9

Publication Date: Rev.1.00, July 14, 2016

Published by: Renesas Electronics Corporation

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

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

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

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

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

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-65030, Fax: +49-211-6503-1327

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

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

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

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

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

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

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

Colophon 1.0

AUTOSAR Modules Overview

User's Manual

Renesas Electronics Corporation

R20UT3752EJ0100