User's Manual

AUTOSAR MCAL R4.0.3

User's Manual

PORT Driver Component Ver.1.0.4

Embedded User's Manual

Target Device:
RH850\P1x-C

Renesas Electronics
www.renesas.com

Rev.1.02 Jun 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
ANSI	American National Standards Institute
API	Application Programming Interface
ARXML	AutosaR eXtensible Mark-up Language
AUTOSAR	AUTomotive Open System ARchitecture
BUS	BUS Network
BSW	Basic SoftWare
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DIO	Digital Input Output
ECU	Electronic Control Unit
GNU	GNU is Not Unix
GPT	General Purpose Timer
HW	HardWare
ICU	Input Capture Unit
id/ID	Identifier
I/O	Input Output
ISR	Interrupt Service Routine
KB	Kilo Bytes
MCAL	Microcontroller Abstraction Layer
MCU	MicroController Unit
MHz	Mega Hertz
NA	Not Applicable
OS	Operating System
PDF	Parameter Definition File
PLL	Phase Locked Loop
PWM	Pulse Width Modulation
RAM	Random Access Memory
ROM	Read Only Memory
RTE	Runtime Environment
SWS	Software Requirements Specification
TAU	Timer Array Unit
WDT	Watchdog Timer

Definitions

Term	Represented by
PORT channel	Numeric identifier linked to a hardware PORT
PORT Idle State	The idle state represents the output state of the PORT channel after the call of Port_SetOutputToIdle or Port_DeInit.
PORT Output State	Defines the output state for a PORT signal. It could be: High Low
PORT period	Defines the period of the PORT signal.
PORT Polarity	Defines the starting output state of each PORT channel
Sl. No.	Serial Number

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

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

This document shall be used as reference by the users of PORT Driver Component for P1x-C Device. The information specific to P1x-C Device channel mapping, ISR handler, integration and build process for application along with the memory consumption and throughput information are provided.

The users of PORT Driver Component shall use this document as reference. This document describes the common features of PORT Driver Component.

This document is intended for the developers of ECU software using Application Programming Interfaces provided by AUTOSAR. The PORT Driver Component provides the following services:

- PORT Driver Component initialization
- Port Pin Direction Handling
- Port Pin Direction Refreshing
- Port Pin Mode Handling
- Port Set To Dio Mode
- Port Set To Alternate Mode
- Port Pin Set To Default Direction
- Port Pin Set To Default Mode
- Module Version Information

The following diagram shows the system overview of the AUTOSAR Architecture.

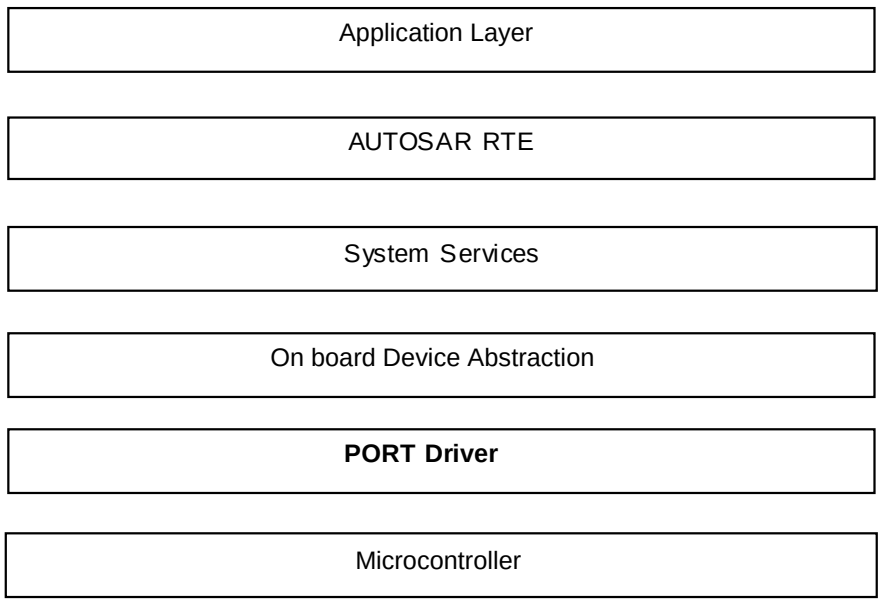

Figure 1-1 System Overview Of AUTOSAR Architecture

The PORT Driver Component comprises of two sections that is, embedded software and the configuration tool to achieve scalability and configurability. The PORT Driver Component Code Generation Tool is a command line tool that accepts ECU configuration description files as input and generates C Source and C Header files. The configuration description is an ARXML file that contains information about the configuration for PORT channels. The tool generates Port_Cfg.h, Port_Cbk.h, Port_Hardware.h, Port_Hardware.c and Port_PBcfg.c files.

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

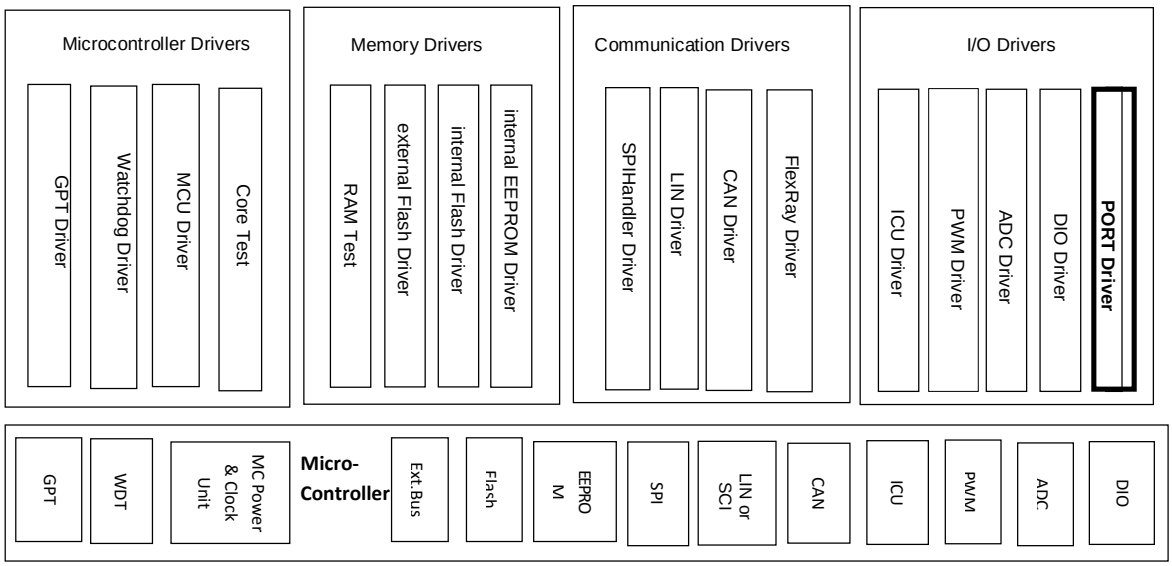

Figure 1-2 System Overview Of The PORT Driver In AUTOSAR MCAL Layer

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
Section 1 (Introduction)	This section provides an introduction and overview of PORT Driver Component.
Section 2 (Reference Documents)	This section lists the documents referred for developing this document.
Section 3 (Integration And Build Process)	This section explains the folder structure for PORT Driver Component along with a sample application.
Section 4 (Forethoughts)	This section provides brief information about the PORT Driver Component, the preconditions that should be known to the user before it is used, data consistency details and deviation list.
Section 5 (Architecture Details)	This section describes the layered architectural details of the PORT Driver Component.
Section 6 (Registers Details)	This section describes the register details of PORT Driver Component.
Section 7 (Interaction Between The User And PORT Driver Component)	This section describes interaction of the PORT Driver Component with the upper layers.
Section 8 (PORT Driver Component Header And Source File Description)	This section provides information about the PORT Driver Component source files is mentioned. This section also contains the brief note on the tool generated output file.
Section 9 (Generation Tool Guide)	This section provides information on the PORT Driver Component Code Generation Tool.
Section 10 (Application Programming Interface)	This section mentions all the APIs provided by the PORT Driver Component.
Section 11 (Development And Production Errors)	This section lists the DET and DEM errors.
Section 12 (Memory Organization)	This section provides the typical memory organization, which must be met for proper functioning of component.
Section 13 (P1x-C Specific Information)	This section describes P1x-C Sample Application with its folder structure and the information about RAM/ROM usage, stack depth and throughput details.
Section 14 (Release Details)	This section provides release details with version name and base version.

Chapter 2 Reference Documents

Sl. No.	Title	Version
1.	Autosar R4.0 Specification of PORT Driver (AUTOSAR_SWS_PortDriver.pdf)	3.2.0
2.	AUTOSAR BUGZILLA (http://www.autosar.org/bugzilla) Note: AUTOSAR BUGZILLA is a database, which contains concerns raised against information present in AUTOSAR Specifications.	-
3.	RH850/P1x-C Group Document User's Manual: Hardware (r01uh0517ej00120_rh850p1x-c_Open.pdf)	Rev.1.20
4.	Specification of Compiler Abstraction (AUTOSAR_SWS_CompilerAbstraction.pdf)	3.2.0
5.	Specification of Memory Mapping (AUTOSAR_SWS_MemoryMapping.pdf)	1.4.0
6.	Specification of Platform Types (AUTOSAR_SWS_PlatformTypes.pdf)	2.5.0

Chapter 3 Integration And Build Process

In this section the folder structure of the PORT Driver Component is explained. Description of the Make files along with samples is provided in this section.

Remark The details about the C Source and C Header files that are generated by the PORT Driver Generation Tool are mentioned in the “R20UT3654EJ0102-AUTOSAR.pdf”.

3.1. PORT Driver Component Make file

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

3.1.1. Folder Structure

The files are organized in the following folders:

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

X1X\common_platform\modules\port\src
\Port.c
\Port_Ram.c
\Port_Version.c

X1X\common_platform\modules\port\include
\Port.h
\Port_PBTypes.h
\Port_Ram.h
\Port_Version.h
\Port_Debug.h
\Port_Types.h
\Port_RegWrite.h

X1X\P1x-C\modules\port\sample_application\make\ghs
App_Port_P1x-C_Sample.mak
App_Port_P1x-C_Sample.ld

X1X\P1x-C\modules\port\user_manual
(User manuals will be available in this folder)

X1X\P1x-C\modules\port\generator
\R403_PORT_P1x-C_BSWMDT.arxml

Note: < Sub-Variant> tag indicate device supported which is P1H-C, P1H-CE, and P1M-C.

Chapter 4 Forethoughts

4.1. General

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

- The PORT Driver Component does not enable or disable the ECU or Microcontroller power supply. The upper layer should handle this operation.
- Start-up code is not implemented by the PORT Driver Component.
- PORT Driver Component does not implement any callback notification functions.
- PORT Driver Component does not implement any scheduled functions.
- The PORT Driver Component is restricted to Post Build only.
- The authorization of the user for calling the software triggering of a hardware reset is not checked in the PORT Driver Component. This will be the responsibility of the upper layer.
- The PORT Driver Component supports setting of Analog and Digital Noise Elimination. To figure out the different port filter arrangements the device User Manual should be taken as reference. If no configuration of a certain port filter is done within this Port Module, the device specific default settings will take effect on this filter.
- The value of unused pins are set to defined state. i.e. Mode = DIO, Direction = Input, Pin Level Value = LOW
- All development errors will be reported to DET by using the API Det_ReportError provided by DET.
- All production errors will be reported to DEM by using the API Dem_ReportErrorStatus provided by DEM.
- The PORT Driver does not have the API support to read the status of Port pins or Port registers. Hence PORT Driver will not support ‘Read back’ feature.
- The file Interrupt_VectorTable.c provided is just a Demo and not all interrupts will be mapped in this file. So the user has to update the Interrupt_VectorTable.c as per his configuration.
- The parameter PortDriveStrengthControl has dependency on parameter PortUniversalCharacteristicCntrl while specifying the output driving abilities of port pins.
- Port_SetToDioMode and Port_SetPinDefaultMode Api shall not change or affect the level of the requested pin.
- The access to HW registers is possible only using AUTOSAR standard and vendor specific API functions described in this document (Chapter 10).
- The output level of each pin can be inverted by configuring the required value (true/false) through the configuration parameter PortOutputLevelInversion.
- The user shall take care of setting mode of a respective port pin as valid or not while calling Port_SetPinMode API.
- The value of unused pins are set to defined state. i.e. Mode = DIO, Direction = Input, Pin Level Value = LOW

4.2. Preconditions

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

- The Port_PBcfg.c, Port_Hardware.c, Port_Hardware.h Port_Cbk.h and Port_Cfg.h files generated by the PORT Driver Component Code Generation Tool must be compiled and linked along with PORT Driver Component source files.
- The application has to be rebuilt, if there is any change in the Port_Cfg.h file generated by the PORT Driver Component Generation Tool.
- File Port_PBcfg.c generated for single configuration set or multiple configuration sets using PORT Driver Component Code Generation Tool should be compiled and linked independently.
- Symbolic names for all Port Pins are generated in Port_Cfg.h file which can be used as parameters for passing to PORT Driver Component APIs.
- The PORT Driver Component needs to be initialized for all Port Pins before doing any operation on Port Pins. The Port_Init () API shall also be called after a reset in order to reconfigure the Port Pins of the microcontroller. If PORT Driver Component is not initialized properly, the behavior of Port Pins may be undetermined.
- The user should ensure that PORT Driver Component API requests are invoked with correct input arguments.
- The other modules depending on PORT Driver Component should ensure that the PORT Driver Component initialization is successful before doing any operation on Port Pins.
- Input parameters are validated only when the static configuration parameter PORT_DEV_ERROR_DETECT is enabled. Application should ensure that the right parameters are passed while invoking the APIs when PORT_DEV_ERROR_DETECT is disabled.
- Values for production code Event Id's should be assigned externally by the configuration of the DEM.
- A mismatch in the version numbers of header and the source files will result in a compilation error. User should ensure that the correct versions of the header and the source files are used.
- The PORT Driver Component APIs, except Port_GetVersionInfo API, which are intended to operate on Port Pins shall be called only after PORT Driver Component is initialized by invoking Port_Init() API. Otherwise Port Pin functions will exhibit undefined behavior.
- All Port Pins and their functions should be configured by the Port configuration tool. It is the User/Integrator responsibility to ensure that the same Port/Port Pin is not being accessed/configured in parallel by different entities in the same system.
- User have the responsibility to enable or disable the critical protection using the parameter PortCriticalSectionProtection. By enabling parameter PortCriticalSectionProtection, Microcontroller HW registers which suffer from concurrent access by multiple tasks, are protected.
- The same alternative function should not be assigned to two different pins at same time.
- The user shall configure the exact Module Short Name PORT in configurations as specified in config.xml file and the same shall be given in command line.

4.3. User Mode and Supervisor Mode

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

Table 4-1 Supervisor mode and User mode details

Sl.No	API Name	User Mode	Supervisor mode	Known limitation in User mode
1	Port_Init	x	x	-
2	Port_SetPinDirection	x	x	-
3	Port_RefreshPortDirection	x	x	-
4	Port_SetPinMode	x	x	-
5	Port_SetToDioMode	x	x	-
6	Port_SetToAlternateMode	x	x	-
7	Port_SetPinDefaultDirection	x	x	-
8	Port_SetPinDefaultMode	x	x	-
9	Port_GetVersionInfo	x	x	-

Note: Implementation of Critical Section is not dependent on MCAL. Hence Critical Section is not considered to the entries for User mode in the above table.
The user can switch between user mode and supervisor mode during Enter/Exit critical section functions, so that these functions will work properly even though critical section protection is ON.

4.4. Data Consistency

To support the re-entrance and interrupt services, the AUTOSAR PORT component will ensure the data consistency while accessing its own RAM storage or hardware registers. The PORT component will use SchM_Enter_Port_<Exclusive Area> and SchM_Exit_Port_<Exclusive Area> functions. The SchM_Enter_Port_<Exclusive Area> function is called before the data needs to be protected and SchM_Exit_Port_<Exclusive Area>function is called after the data is accessed.

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

- PORT_SET_PIN_MODE_PROTECTION
- PORT_SET_PIN_DEFAULT_MODE_PROTECTION
- PORT_SET_PIN_DEFAULT_DIR_PROTECTION
- PORT_SET_PIN_DIR_PROTECTION
- PORT_SET_TO_DIO_ALT_PROTECTION
- PORT_REFRESHPORT_INTERNAL_PROTECTION

The functions SchM_Enter_Port_<Exclusive Area> and SchM_Exit_Port_<Exclusive Area> can be disabled by disabling the configuration parameter ‘PortCriticalSectionProtection’.

Table 4-2 PORT Driver Protected Resources List

API Name	Exclusive Area Type	Protected Resources
Port_SetPinDirection	PORT_SET_PIN_DIR_PROTECTION	HW registers: PSRn, JPSR0, PMSRn, PINVn and JPMSR0.
Port_RefreshPortDirecti	PORT_REFRESHPORT_INTERNAL_PROTECTION	HW registers: PMSRn and JPMSR0.

on		
Port_SetPin Mode	PORT_SET_PIN_MODE_PRO TECTION	HW registers: PIPCn, PMSRn, PMCSRn, PSRn, JPMSR0, JPMCSR0, JPSR0, PFCEn, PFCn and JPFCE0.
Port_SetTo DioMode	PORT_SET_TO_DIO_ALT_PR OTECTION	HW registers: PMCSRn, PIPCn and JPMCSR0
Port_SetTo AlternateMo de	PORT_SET_TO_DIO_ALT_PR OTECTION	HW registers: PMCSRn, PIPCn and JPMCSR0
Port_SetPin DefaultMod e	PORT_SET_PIN_DEFAULT_M ODE_PROTECTION	HW registers: PMCSRn, PMSRn, PIPCn, JPMCSR0, JPMSR0, PFCEn, PFCn, JPFCE0, PSRn and JPSR0.
Port_SetPin DefaultDire ction	PORT_SET_PIN_DEFAULT_DI R_PROTECTION	HW registers: PMSRn, JPMSR0, PSRn and JPSR0.
Port_GetVe rsionInfo	None	None

Note: The highest measured duration of a critical section is 2.512 micro seconds measured for Port_RefreshPortDirection API.

4.5. Deviation List

Table 4-3 PORT Driver Deviation List

Sl. No.	Description	AUTOSAR Bugzilla
1.	The Port Pin specific containers (PortPin0, PortPin1, PortPin2 and so on …) are added as sub containers of PortGroup<n> containers, having the parameters 'PortPinDirection', 'PortPinDirectionChangeable', 'PortPinLevelValue' and 'PortPinInitialMode' are added. AUTOSAR specified container 'PortPin' and all its parameters are considered as unused.	-
2.	PortPinMode configuration parameter is not used for implementation as all possible modes of a pin can be used in the Port_SetPinMode function.	-
3.	[ecuc_sws_2108] requirement is not applicable to port module since implementation of PORT module is vendor specific.	-
4.	Port Pin level inversion is implemented as per Renesas requirement which is violating AUTOSAR requirement PORT082	-

Chapter 5 Architecture Details

The PORT Driver Component accesses the microcontroller Port Pins that are located in the On-Chip hardware. The basic architecture of the PORT Driver Component is illustrated below:

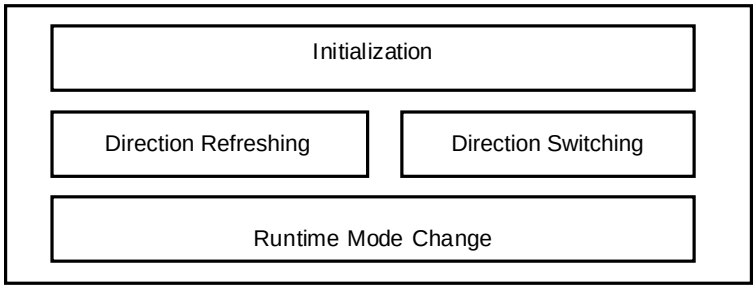

Figure 5-1 PORT Driver Architecture

The PORT Driver Component consists of the following sub modules based on the functionality:

- Port Initialization.
- Port Direction Refreshing.
- Port Pin Direction Switching.
- Port Pin Mode Change.
- Module Version Information

Port Initialization

This sub module provides the Port initialization functionality by providing the Port_Init() API. This API should be invoked before the usage of any other APIs of PORT Driver Component. Port Initialization includes initializing Port Pin mode, Port Pin direction, Port Pin Level value, Port Pin driven value (Normal / Open Drain), Activation of internal pull-ups and Port Filter configuration.

Port Direction Refreshing

This sub module provides the Port Direction Refreshing functionality by providing the Port_RefreshPortDirection() API. In this functionality the PORT Driver Component refreshes the direction of all configured Port Pins except those Port Pins that are configured as ‘Port Pin Direction Changeable during runtime’.

In this functionality only Direction of Port Pins is refreshed.

Port Pin Direction Switching

This sub module provides the Port Direction switching functionality at run time by providing the Port_SetPinDirection() API. In this functionality the PORT driver Component allows the user to change the direction of Port Pins during runtime.

Port Pin Mode changing
This sub module provides the Port Mode change functionality at run time by providing the Port_SetPinMode() API. In this functionality the PORT driver Component allows the user to change the mode of Port Pins during runtime.

This sub module provides the Port Mode change functionality at run time by providing the Port_SetToDioMode() API. In this functionality the PORT

driver Component allows the user to change the mode of Port Pin to DIO mode during runtime.

This sub module provides the Port Mode change functionality at run time by providing the Port_SetToAlternateMode() API. In this functionality the PORT driver Component allows the user to change the mode of Port Pin to alternate mode during runtime.

Module Version Information

The Api Port_GetVersionInfo is responsible for reading the version information of the PORT Driver Information. The version information includes Module ID, Vendor ID, and Version number of the PORT Driver software.

Chapter 6 Registers Details

This section describes the register details of PORT Driver Component.

Table 6-1 Register Details

API Name	Register Access 8/16/32 bits	Register Access r/w/rw	Registers	Configuration Parameter	Macro/Variable
Port_SetPinDirection	32 bit	rw	PSRn	PortPinLevelValue PortPinDirectionChangeable	usChangeableConfigVal
	32 bit	rw	JPSR0	PortPinLevelValue PortPinDirectionChangeable	usChangeableConfigVal
	32 bit	rw	PMSRn	PortPinDirection PortPinDirectionChangeable	usOrMaskVal
	32 bit	rw	JPMSR0	PortPinDirection PortPinDirectionChangeable	usOrMaskVal
	32 bit	w	PINVn	PortOutputLevelInversion PortPinDirectionChangeable PortPinDirection	usPortinversionVal
Port_RefreshPortDirection	32 bit	rw	PMSRn	PortPinDirection PortPinDirectionChangeable	ulMaskAndConfigValue
	32 bit	rw	JPMSR0	PortPinDirection PortPinDirectionChangeable	ulMaskAndConfigValue
Port_SetToDioMode	32 bit	rw	PMCSRn	PortPinDioAltModeChangeable PortPinInitialMode	usOrMask
	16 bit	rw	PIPCn	PortIpControl PortPinInitialMode PortPinDioAltModeChangeable	usOrMask
	32 bit	rw	JPMCSR0	PortPinDioAltModeChangeablePortPinInitialMode	usOrMask
Port_SetToAlternateMode	32 bit	rw	PMCSRn	PortPinDioAltModeChangeablePortPinInitialMode	usOrMask
	16 bit	rw	PIPCn	PortIpControl PortPinInitialMode PortPinDioAltModeChangeable	usOrMask
	32 bit	rw	JPMCSR0	PortPinDioAltModeChangeable PortPinInitialMode	usOrMask
Port_SetPinDefaultMode	32 bit	rw	PMCSRn	PortPinModeChangeable PortPinInitialMode PortPinDirection	usOrMask usInitModeRegVal
	32 bit	rw	PMSRn	PortPinModeChangeable PortPinInitialMode PortPinDirection	usOrMask usInitModeRegVal
	32 bit	rw	PSRn	PortPinModeChangeable PortPinLevelValue PortPinDirection	usOrMask usInitModeRegVal

API Name	Register Access 8/16/32 bits	Register Access r/w/rw	Registers	Configuration Parameter	Macro/Variable
	16 bit	rw	PIPCn	PortPinModeChangeable PortIpControl	usOrMask usInitModeRegVal
	32 bit	rw	JPMCSR0	PortPinModeChangeable PortPinInitialMode PortPinDirection	usOrMask usInitModeRegVal
	32 bit	rw	JPMSR0	PortPinModeChangeable PortPinInitialMode PortPinDirection	usOrMask usInitModeRegVal
	32 bit	rw	JPSR0	PortPinModeChangeable PortPinLevelValue PortPinDirection	usOrMask usInitModeRegVal
	16 bit	rw	PFCEn	PortPinModeChangeable PortPinInitialMode	usOrMask usInitModeRegVal
	16 bit	rw	PFCn	PortPinModeChangeable PortPinInitialMode	usOrMask usInitModeRegVal
	8 bit	rw	JPFCE0	PortPinModeChangeable PortPinInitialMode	usOrMask usInitModeRegVal
Port_SetPinDefaultDirection	32 bit	rw	PMSRn	PortPinDirection PortPinDirectionChangeabl	usOrMaskVal
	32 bit	rw	PSRn	PortPinDirectionChangeable PortPinLevelValue	usOrMaskVal
	32 bit	rw	JPSR0	PortPinDirectionChangeable PortPinLevelValue	usOrMaskVal
	32 bit	rw	JPMSR0	PortPinDirection PortPinDirectionChangeable	usOrMaskVal
Port_SetPinMode	16 bit	rw	PIPCn	PortPinModeChangeable PortIpControl	usOrMask
	32 bit	rw	PMSRn	PortPinModeChangeable	usOrMask
	32 bit	rw	PMCSRn	PortPinModeChangeable	usOrMask
	32 bit	rw	PSRn	PortPinModeChangeable PortPinLevelValue	usInitModeRegVal
	32 bit	rw	JPMSR0	PortPinModeChangeable	usOrMask
	32 bit	rw	JPMCSR0	PortPinModeChangeable	usOrMask
	32 bit	rw	JPSR0	PortPinModeChangeable PortPinLevelValue	usInitModeRegVal
	16 bit	rw	PFCEn	PortPinModeChangeable	usOrMask
	16 bit	rw	PFCn	PortPinModeChangeable	usOrMask
	8 bit	rw	JPFCE0	PortPinModeChangeable	usOrMask
Port_Init	32 bit	rw	PSRn	PortPinLevelValue	usInitModeRegValPSR

API Name	Register Access 8/16/32 bits	Register Access r/w/rw	Registers	Configuration Parameter	Macro/Variable
	32 bit	rw	JPSR0	PortPinLevelValue	usInitModeRegValPSR
	32 bit	rw	PMSRn	PortPinDirection	usInitModeRegVal
	32 bit	rw	PMCSRn	PortPinInitialMode	usInitModeRegValPMCSR
	16 bit	rw	PISn	PortInputSelection	usInitModeRegValPIS
	8 bit	rw	JPIS0	PortInputSelection	usInitModeRegValPIS
	16 bit	rw	PIBCn	PortInputBufferControl	usInitModeRegValPIBC
	8 bit	rw	JPIBC0	PortInputBufferControl	usInitModeRegValPIBC
	16 bit	rw	PIPCn	PortIpControl	usInitModeRegValPIPC
	16 bit	rw	PUn	PullUpOption	usInitModeRegValPU
	8 bit	rw	JPU0	PullUpOption	usInitModeRegValPU
	16 bit	rw	PDn	PullDownOption	usInitModeRegValPD
	8 bit	rw	JPD0	PullDownOption	usInitModeRegValPD
	16 bit	rw	PBDCn	PortBiDirectionControl	usInitModeRegValPBDC
	8 bit	rw	JPBDC0	PortBiDirectionControl	usInitModeRegValPBDC
	8 bit	rw	DNFAnCTL	PortSameLevelSamples	ucDNFACTL
				PortSamplingClockFrequency	
	8 bit	rw	FCLAnCTLm	PortDigitalFilterEdgeControl	ucFCLACTL
	16 bit	rw	DNFAnEN	PortDigitalFilterEnableInput	usDNFAEN
	8 bit	rw	JPFCE0	PortPinInitialMode	usInitModeRegValPFCE
	32 bit	rw	JPMCSR0	PortPinInitialMode	usInitModeRegValPMCSR
	32 bit	rw	JPMSR0	PortPinDirection	usInitModeRegValPMSR
	16 bit	rw	PFCEn	PortPinInitialMode	usInitModeRegValPFCE
	16 bit	rw	PFCn	PortPinInitialMode	usInitModeRegValPFC
	32 bit	w	PODCn	PortOpenDrainControlExpansion	usInitModeRegValPODC
	32 bit	w	JPODC0	PortOpenDrainControlExpansion	usInitModeRegValPODC
	32 bit	w	PODCEn	PortOpenDrainControlExpansion	usInitModeRegValPODCE

API Name	Register Access 8/16/32 bits	Register Access r/w/rw	Registers	Configuration Parameter	Macro/Variable
	32 bit	w	PDSCn	PortDriveStrengthControl	usInitModeRegValPDSC
	32 bit	w	JPDSC0	PortDriveStrengthControl	usInitModeRegValPDSC
	32 bit	w	PUCCn	PortUnlimitedCurrentControl	usInitModeRegValPUCC
	32 bit	w	JPUCC0	PortUnlimitedCurrentControl	usInitModeRegValPUCC
	16 bit	w	PINVn	PortOutputLevelInversion	usInitModeRegValPINV
	16 bit	w	JPINV0	PortOutputLevelInversion	usInitModeRegValPINV
Port_GetVersionInfo	-	-	-	-	-

Chapter 7 Interaction Between The User And PORT Driver Component

The details of the services supported by the PORT Driver Component to the upper layers users and the mapping of the channels to the hardware units is provided in the following sections:

7.1. Services provided by PORT Driver Module to User

The PORT Driver provides following functionalities to the upper layers:

- To initialize the PORT pins.
- To change the direction of a PORT pin during runtime.
- To change the mode of a PORT pin during runtime.
- To refresh the direction of a PORT Pin.
- To read the version information of the PORT module.
- To change the direction of a PORT pin to default.
- To change the mode of a PORT pin to default.
- To change the mode of a PORT pin to DIO.
- To change the mode of a PORT pin to ALTERNATE

Chapter 8 PORT Driver Component Header And Source File Description

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

The C header file generated by PORT Driver Generation Tool:
- Port_Cfg.h
- Port_Cbk.h
- Port_Hardware.h

The C source file generated by PORT Driver Generation Tool:
- Port_PBcfg.c
- Port_Hardware.c

The PORT Driver Component C header files:
- Port.h
- Port_PBTypes.h
- Port_Ram.h
- Port_Version.h
- Port_Debug.h
- Port_Types.h
- Port_RegWrite.h

The PORT Driver Component source files:
- Port.c
- Port_Ram.c
- Port_Version.c

The Stub C header files:
- Compiler.h
- Compiler_Cfg.h
- MemMap.h
- Platform_Types.h
- Std_Types.h
- Dem.h
- Dem_Cfg.h
- Det.h
- Schm_Port.h

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

Table 8-1 Description of the PORT Driver Component Files

File	Details
Port_Cfg.h	This file contains various PORT Driver Pre-compile time parameters, macro definitions for the ISRs, channel notifications used by PORT Driver, PORT channel handles.
Port_Cbk.h	This file contains the definition of error interface which will be invoked when the port register write-verify fails.
Port_PBcfg.c	This file contains the post-build configuration data. The structures related to PORT initialization, PORT Timer channel configuration and the timer related structures are also provided in this file.
Port_Hardware.h	This file is generated by the PORT Generation Tool which includes definition of hardware registers specific to P1x-C PORT.
Port_Hardware.c	This file is generated by the PORT Generation Tool which consists of Base address for each Port Register and Global variable definition of hardware registers specific to P1x-C PORT.
Port.h	This file provides extern declarations for all the PORT Driver Component APIs. This file provides service Ids of APIs, DET Error codes and type definitions for Port initialization structure. This header file shall be included in other modules to use the features of PORT Driver Component.
Port_PBTypes.h	This file contains the data structures related to Port initialization, Port Refresh, Direction changeable Pins at run time and Mode Changeable at run time.
Port_Types.h	This file provides data structure and type definitions for initialization of MCU Driver.
Port_Debug.h	This file is used for version check.
Port_RegWrite.h	This file is to have macro definitions for the registers write and verification.
Port_Ram.h	This file contains the extern declarations for the global variables defined in Port_Ram.c file.
Port_Version.h	This file contains the macros of AUTOSAR version numbers of all modules that are interfaced to PORT Driver.
Port.c	This file contains the implementation of all APIs.
Port_Ram.c	This file contains the global variables used by PORT Driver Component.
Port_Version.c	This file contains the code for checking version of all modules that are interfaced to PORT Driver.
Compiler.h	Provides compiler specific (non-ANSI) keywords. All mappings of keywords, which are not standardized, and/or compiler specific are placed and organized in this compiler specific header.
Compiler_Cfg.h	This file contains the memory and pointer classes.
MemMap.h	This file allows to map variables, constants and code of modules to individual memory sections. Memory mapping can be modified as per ECU specific needs.
Platform_Types.h	This file provides provision for defining platform and compiler dependent types.
Dem.h	This file is a stub for DEM component
Dem_Cfg.h	This file contains the stub values for Dem_Cfg.h
SchM_Port.h	This file is a stub for SchM Component
Std_Types.h	Provision for Standard types
Det.h	This file is a stub for DET component.

Chapter 9 Generation Tool Guide

For more information on the Code Generation, please refer “R20UT3654EJ0102-AUTOSAR.pdf” document.

Chapter 10 Application Programming Interface

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

10.1. Imported Types

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

10.1.1. Standard Types

In this section all types included from the Std_Types.h are listed:
Std_VersionInfoType
Std_ReturnType

10.1.2. Other Module Types

In this chapter all types included from the Dem_types.h are listed:
Dem_EventIdType

10.2. Type Definitions

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

10.2.1. Port_ConfigType

Name:	Port_ConfigType		
Type:	struct		
Element:	**Type**	**Name**	**Explanation**
	uint32	ulStartOfDbToc	Database start value.
	Port_Regs	pPortNumRegs	Pointer to the address of Numeric port registers configuration.
	Port_FuncCtrlRegs	pPortNumFuncCtrlRegs	Pointer to the address of the Numeric function control registers configuration.
	Port_PMSRRegs	pPortNumPMSRRegs	Pointer to the address of the Numeric PMSR registers configuration.
	Port_Regs	pPortJRegs	Pointer to the address of JTAG port registers configuration
	Port_FuncCtrlRegs	pPortJFuncCtrlRegs	Pointer to the address of JTAG function control registers configuration

	Port_PMSRRegs	pPortJPMSRRegs	Pointer to the address of JTAG PMSR registers configuration.
	Port_PinsDirChangeable	pPinDirChangeable	Pointer to the address of runtime direction changeable pins structure.
	Port_PinModeChangeableGroups	pPinModeChangeableGroups	Pointer to the address of runtime mode changeable pin group details structure.
	Port_PinDioAltChangeableDetails	pPinDioAltModeDetails	Pointer to the address of run time mode changeable pins structure.
	Port_PinModeChangeableDetails	pPinModeChangeableDetails	Pointer to the address of run time mode changeable pins structure.
	Port_DNFARegs	pPortDNFARegs	Pointer to the DNFA registers structure.
	Port_FCLARegs	pPortFCLARegs	Pointer to the FCLA registers structure.
	uint8	ucNoOfPinsDirChangeable	Total number of Pins configured for Direction Changeable at run time
	uint8	ucNoOfPinsModeChangeable	Total number of Pins configured for mode Changeable at run time
	uint8	ucNoOfPinsDioAltModeChangeable	Total number of Pins configured for mode Changeable at run time
	uint8	ucNoOfDNFARegs	The total number of DNFA noise elimination registers
	uint8	ucNoOfFCLARegs	The total number of FCLA noise elimination registers
Description:	This is the type of the external data structure containing the initialization data for the PORT Driver Component. The user shall use the symbolic names defined in the PORT Driver Configuration Tool. The configuration of each Port Pin is Microcontroller specific.		

10.2.2. Port_PinType

Name:	Port_PinType
Type:	uint16
Range:	0 to 65535
Description:	The user shall use the symbolic names defined in the PORT Driver Configuration Tool. The configuration of each Port Pin is Microcontroller specific.

10.2.3. Port_PinDirection Type

Name:	Port_PinDirectionlType	
Type:	Enumeration	
Range:	PORT_PIN_OUT	Output Direction
	PORT_PIN_IN	Input Direction
Description:	These are the possible directions; a port pin can have for both input and output.	

10.2.4. Port_PinModeType

Name:	Port_PinModeType		
Type:	uint8		
Range:	PIPC=0		
	0	PORT_DIO_OUT	(Port_PinModeType)0x00
	1	PORT_DIO_IN	(Port_PinModeType)0x01
	2	APP_ALT1_OUT	(Port_PinModeType)0x02
	3	APP_ALT1_IN	(Port_PinModeType)0x03
	4	APP_ALT2_OUT	(Port_PinModeType)0x04
	5	APP_ALT2_IN	(Port_PinModeType)0x05
	6	APP_ALT3_OUT	(Port_PinModeType)0x06
	7	APP_ALT3_IN	(Port_PinModeType)0x07
	8	APP_ALT4_OUT	(Port_PinModeType)0x08
	9	APP_ALT4_IN	(Port_PinModeType)0x09
Range:	PIPC=1		
	0	APP_ALT1_OUT_SET_PIPC	(Port_PinModeType)0x82
	1	APP_ALT1_IN_SET_PIPC	(Port_PinModeType)0x83
	2	APP_ALT2_OUT_SET_PIPC	(Port_PinModeType)0x84
	3	APP_ALT2_IN_SET_PIPC	(Port_PinModeType)0x85
	4	APP_ALT3_OUT_SET_PIPC	(Port_PinModeType)0x86
	5	APP_ALT3_IN_SET_PIPC	(Port_PinModeType)0x87
	6	APP_ALT4_OUT_SET_PIPC	(Port_PinModeType)0x88
	7	APP_ALT4_IN_SET_PIPC	(Port_PinModeType)0x89
Description:	These are the possible modes; a port pin can have for both input and output.		

10.3. Function Definitions

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

Table 10-1 AUTOSAR Specific APIs supported by the PORT Driver Component

SL.NO	API's	API's specific
1	Port_Init	-
2	Port_SetPInDirection	-
3	Port_RefreshPortDirection	-
4	Port_GetVersionInfo	-
5	Port_SetPinMode	-

Table 10-2 Non- AUTOSAR Specific APIs supported by the PORT Driver Component

SL. NO	API's
1	Port_SetToDioMode
2	Port_SetToAlternateMode
3	Port_SetPinDefaultMode
4	Port_SetPinDefaultDirection

10.3.1 Port_Init

Name:	Port_Init		
Prototype:	FUNC(void, PORT_PUBLIC_CODE) Port_Init (P2CONST (Port_ConfigType, AUTOMATIC, PORT_APPL_CONST) ConfigPtr)		
Service ID:	0x00		
Sync/Async:	Synchronous		
Reentrancy:	Non-Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Port_ConfigType	ConfigPtr	NA
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This service performs initialization of the PORT Driver components.		
Configuration Dependency:	None		
Preconditions:	None		

10.3.2 Port_SetPinDirection

Name:	Port_SetPinDirection		
Prototype:	FUNC (void, PORT_PUBLIC_CODE) Port_SetPinDirection (Port_PinType Pin, Port_PinDirectionType Direction)		
Service ID:	0x01		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Port_PinType	Pin	0-136
	Port_PinDirectionType	Direction	0,1
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This service sets the port pin direction during runtime		
Configuration Dependency:	None		
Preconditions:	Ports should be initialized by calling Port_Init().		

10.3.3 Port_RefreshPortDirection

Name:	Port_RefreshPortDirection		
Prototype:	FUNC (void, PORT_PUBLIC_CODE) Port_RefreshPortDirection (void)		
Service ID:	0x02		
Sync/Async:	Synchronous		
Reentrancy:	Non-Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	None	NA	NA
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This service shall refresh the direction of all configured ports to the configured direction.		
Configuration Dependency:	None		
Preconditions:	Ports should be initialized by calling Port_init().		

10.3.4 Port_GetVersionInfo

Name:	Port_GetVersionInfo		
Prototype:	FUNC(void, PORT_PUBLIC_CODE) Port_GetVersionInfo (P2VAR(Std_VersionInfoType, AUTOMATIC, PORT_APPL_DATA)versioninfo)		
Service ID:	0x03		
Sync/Async:	Synchronous		
Reentrancy:	Non-Reentrant		
	Type	**Parameter**	**Value/Range**

Parameters In:	None	NA	NA
Parameters InOut:	None	NA	NA
Parameters out:	Std_VersionInfoType	versioninfo	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This API will return the version information of this Port Driver.		
Configuration Dependency:	None		
Preconditions:	None		

10.3.5 Port_SetPinMode

Name:	Port_SetPinMode		
Prototype:	FUNC (void, PORT_PUBLIC_CODE) Port_SetPinMode (Port_PinType Pin, Port_PinModeType Mode)		
Service ID:	0x04		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Port_PinType	Pin	0-136
	Port_PinModeType	Mode	2-9, 82-89
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This function used to set the mode of a port pin during runtime.		
Configuration Dependency:	None		
Preconditions:	Ports should be initialized by calling Port_init().		

10.3.6 Port_SetToDioMode

Name:	Port_SetToDioMode		
Prototype:	FUNC (void, PORT_PUBLIC_CODE) Port_SetToDioMode (Port_PinType Pin)		
Service ID:	0x05		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Port_PinType	Pin	0-136
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This function used to set the mode of a port pin to DIO mode during runtime.		

Configuration Dependency:	None
Preconditions:	Ports should be initialized by calling Port_init().

10.3.7 Port_SetToAlternateMode

Name:	Port_SetToAlternateMode		
Prototype:	FUNC (void, PORT_PUBLIC_CODE) Port_SetToAlternateMode (Port_PinType Pin)		
Service ID:	0x06		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Port_PinType	Pin	0-136
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This function used to set the mode of a port pin to alternate mode during runtime.		
Configuration Dependency:	None		
Preconditions:	Ports should be initialized by calling Port_init().		

10.3.8 Port_SetPinDefaultMode

Name:	Port_SetPinDefaultMode		
Prototype:	FUNC (void, PORT_PUBLIC_CODE) Port_SetPinDefaultMode (Port_PinType Pin)		
Service ID:	0x07		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Port_PinType	Pin	0-136
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This function used to set the mode of a port pin during runtime. The PORT Driver module allows changing the mode of the pin to default mode set by the configuration at the time of Port_Init().		
Configuration Dependency:	None		
Preconditions:	Ports should be initialized by calling Port_init().		

10.3.9 Port_SetPinDefaultDirection

Name:	Port_SetPinDefaultDirection		
Prototype:	FUNC (void, PORT_PUBLIC_CODE) Port_SetPinDefaultDirection (Port_PinType Pin)		
Service ID:	0x08		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Port_PinType	Pin	0-136
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This service sets the port pin direction during runtime. The PORT Driver module allows changing the mode of the pin to default mode set by the configuration at the time of Port_Init().		
Configuration Dependency:	None		
Preconditions:	Ports should be initialized by calling Port_Init().		

Chapter 11 Development And Production Errors

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

11.1. PORT Driver Component Development Errors

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

Table 11-1 DET Errors of PORT Driver Component

Sl. No.	**1**
Error Code	PORT_E_PARAM_CONFIG
Related API(s)	Port_Init
Source of Error	API is invoked with NULL Pointer
Sl. No.	**2**
Error Code	PORT_E_INVALID_DATABASE
Related API(s)	Port_Init
Source of Error	Invalid database is found
Sl. No.	**3**
Error Code	PORT_E_UNINIT
Related API(s)	Port_RefreshPortDirection, Port_SetPinDirection, Port_SetPinMode, Port_SetToDioMode, Port_SetToAlternateMode
Source of Error	APIs are invoked without the initialization of the PORT Driver Component.
Sl. No.	**4**
Error Code	PORT_E_PARAM_PIN
Related API(s)	Port_SetPinMode, Port_SetPinDirection, Port_SetToDioMode, Port_SetToAlternateMode
Source of Error	API is invoked with invalid Pin
Sl. No.	**5**
Error Code	PORT_E_PARAM_INVALID_MODE
Related API(s)	Port_SetPinMode
Source of Error	API is invoked with invalid mode
Sl. No.	**6**
Error Code	PORT_E_DIRECTION_UNCHANGEABLE
Related API(s)	Port_SetPinDirection
Source of Error	API is invoked with Pin which is not configured as ‘Direction Changeable during run time’.
Sl. No.	**7**
Error Code	PORT_E_MODE_UNCHANGEABLE
Related API(s)	Port_SetPinMode, Port_SetToDioMode, Port_SetToAlternateMode
Source of Error	API is invoked with Pin which is not configured as ‘Mode Changeable during run time’.

Sl. No.	8
Error Code	PORT_E_PARAM_POINTER
Related API(s)	Port_GetVersionInfo
Source of Error	GetVersionInfo is called with NULL pointer.

11.2. PORT Driver Component Production Errors

The following table contains the DEM errors that are reported by PORT software component.

Table 11-2 DEM Errors of PORT Driver Component

Sl. No.	1
Error Code	PORT_E_REG_WRITE_VERIFY
Related API(s)	Port_Init ,Port_SetPinDirection, Port_RefreshPortDirection, Port_SetPinMode, Port_SetToDioMode, Port_SetToAlternateMode, Port_SetPinDefaultMode, Port_SetPinDefaultDirection
Source of Error	When register write-verify fails.

Chapter 12 Memory Organization

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

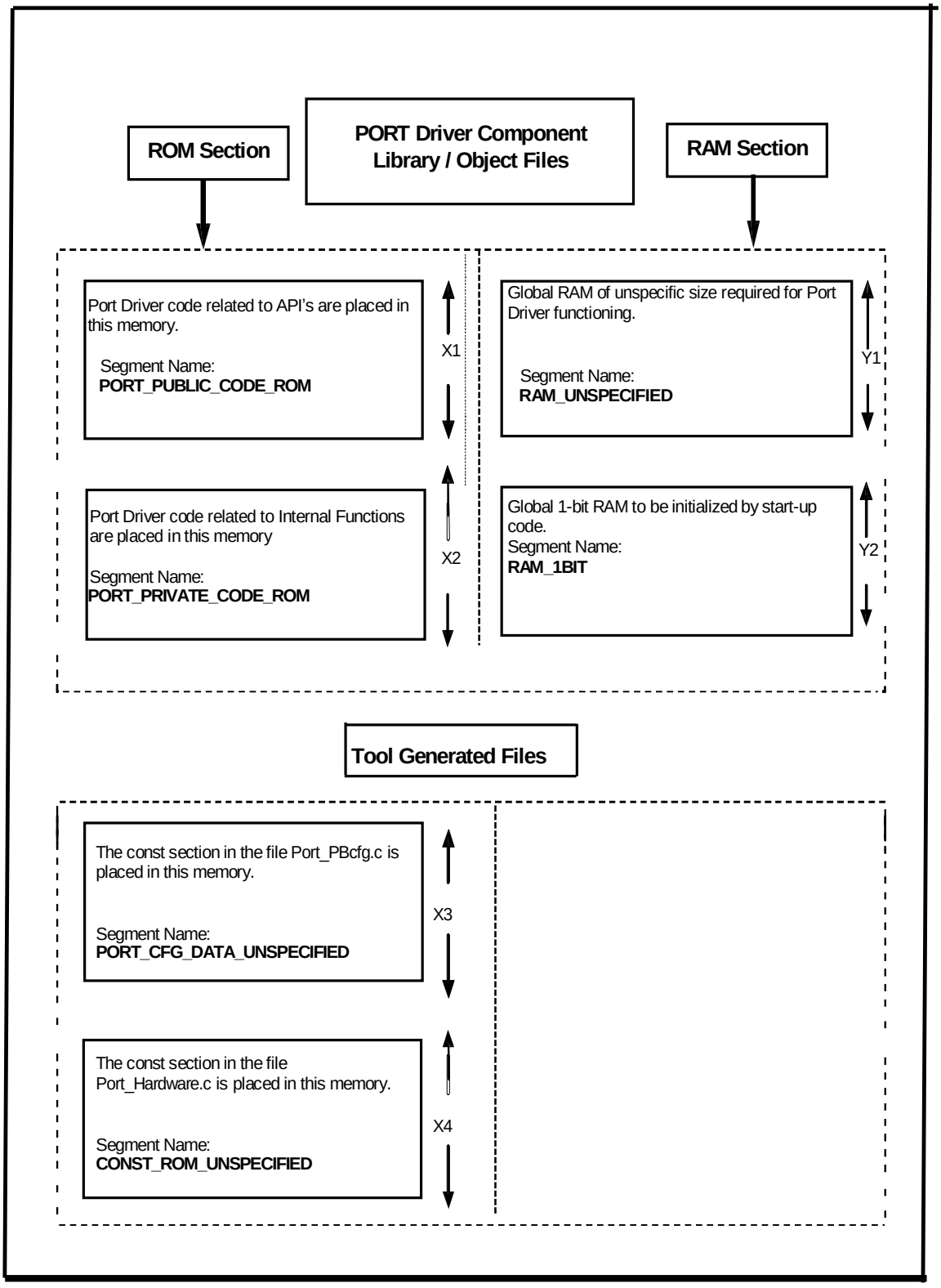

Figure 12-1 PORT Driver Component Memory Organization

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

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

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

PORT_CFG_DATA_UNSPECIFIED (X3): This section consists of PORT Driver Component constant configuration structures and database table of contents generated by the PORT Driver Component Generation Tool. This can be located in code memory.

CONST_ROM_UNSPECIFIED (X4): The constant section of PORT Driver Component code that can be located in code memory.

RAM Section (Y1 and Y2):

RAM_UNSPECIFIED (Y1): This section consists of the global RAM pointer variables that are used internally by PORT Driver Component. This can be located in data memory.

RAM_1BIT (Y2): This section consists of the global RAM variables of 1-bit size that are used internally by PORT Driver Component. This can be located in data memory.

Chapter 13 P1x-C Specific Information

P1x-C supports following devices:

- RF701370A(CPU1(PE1))
- RF701371(CPU1(PE1))
- RF701372(CPU1(PE1))
- RF701373
- RF701374

13.1. Interaction between the User and PORT Driver Component

The details of the services supported by the PORT Driver Component to the upper layers users and the mapping of the channels to the hardware units is provided in the following sections:

13.1.1. Parameter Definition File

Parameter definition files support information for P1x-C

Table 13-1 PDF information for P1x-C

PDF Files	Devices Supported
R403_PORT_P1X-C_70A_71_72.arxml	701370A(CPU1(PE1)), 701371(CPU1(PE1)), 701372(CPU1(PE1))
R403_PORT_P1X-C_73.arxml	701373
R403_PORT_P1X-C_74.arxml	701374

13.1.2. Services Provided By PORT Driver Component

The PORT Driver Component provides the following functionalities to the upper layers or users:

- To initialize the Port and set according Port filter functions.
- To refresh the direction of Port.
- To switch the Port pin direction at run time.
- To change the mode of a Port pin at run time.
- To read the PORT Driver Component version information.

13.2. Sample Application

13.2.1. Sample Application Structure

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

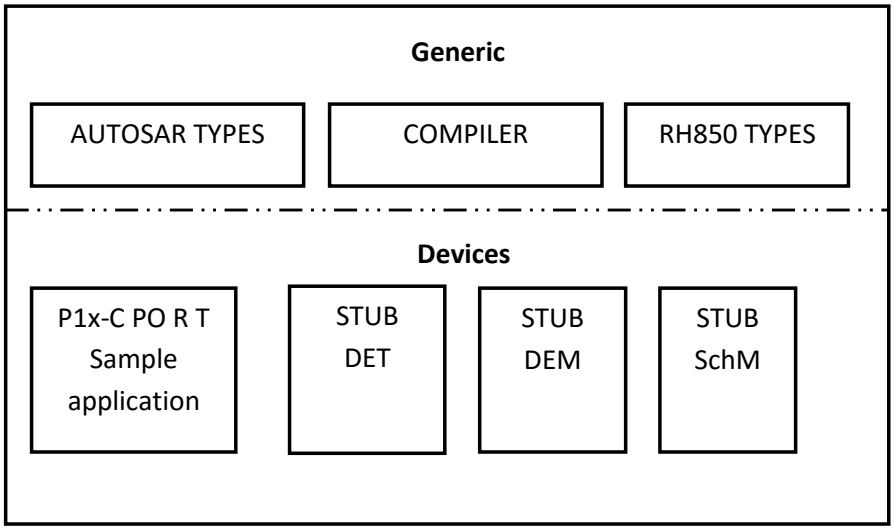

Figure 13-1 Overview of PORT Driver Sample Application

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

X1X\P1x-C\modules\port\sample_application

The Sample Application consists of the following folder structure:

X1X\P1x-C\modules\port\definition\4.0.3\P1H-C\
R403_PORT_P1X-C_70A_71_72.arxml

X1X\P1x-C\modules\port\definition\4.0.3\P1M-C\
R403_PORT_P1X-C_73.arxml
R403_PORT_P1X-C_74.arxml

X1X\P1x-C\modules\port\definition\4.0.3\ P1H-CE\
R403_PORT_P1X-C_70A_71_72.arxml

X1X\P1x-C\modules\port\sample_application\<SubVariant>\4.0.3
\src\Port_PBcfg.c
\src\Port_Hardware.c
\include\Port_Cfg.h
\include\Port_Hardware.h

X1X\P1x-C\modules\port\sample_application\P1H-CE\4.0.3

\config\ App_PORT_P1x-C_701370A_Sample.arxml

X1X\P1x-C\modules\port\sample_application\P1H-C\4.0.3

\config\ App_PORT_P1x-C_701371_Sample.arxml

\config\ App_PORT_P1x-C_701372_Sample.arxml

X1X\P1x-C\modules\port\sample_application\P1M-C\4.0.3

\config\ App_PORT_P1x-C_701373_Sample.arxml

\config\ App_PORT_P1x-C_701374_Sample.arxml

In the Sample Application all the PORT APIs are invoked in the following sequence:

- Port_GetVersionInfo: The API Port_GetVersionInfo is invoked to get the version of the PORT Driver module with a variable of Std_VersionInfoType after the call of this API the passing parameter will get updated with the PORT Driver version details.
- Port_Init: The API Port_Init is invoked with a valid database address for the proper initialization of the PORT Driver, all the PORT Driver control registers and RAM variables will get initialized after this API is called.
- Port_SetPinMode: This service sets the Port Pin mode during runtime.
- Port_SetPinDirection: This service sets the port pin direction during
- Port_RefreshPortDirection: The API refreshes the direction of all ports to the configured direction. It excludes those port pins from refreshing that are configured as 'pin direction changeable during runtime' by invoking internal API Port_RefreshPortInternal().
- Port_SetPinDefaultDirection: This service sets the port pin direction during runtime. The PORT Driver module allows changing the mode of the pin to default mode set by the configuration at the time of Port_Init().
- Port_SetToDioMode: This function used to set the mode of a port pin to DIO mode during runtime.
- Port_SetToAlternateMode: This function used to set the mode of a port pin to alternate mode during runtime.
- Port_SetPinDefaultMode: This function used to set the mode of a port pin during runtime. The PORT Driver module allows changing the mode of the pin to default mode set by the configuration at the time of Port_Init().

Note: <SubVariant> indicate P1H-CE, P1H-C, P1M-C.

13.2.2. Building Sample Application

13.2.2.1 Configuration Example

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

13.2.2.2 Debugging the Sample Application

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

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

“X1X\P1x-C\common_family\make\<Compiler>”

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

SampleApp.bat Port <Device_name>

- After this, all the object files, map file and the executable file App_PORT_P1x-C_Sample.out will be available in the output folder: (“X1X\P1x-C\modules\port\sample_application\<SubVariant> \obj\<Compiler>”)
- The executable can be loaded into the debugger and the sample application can be executed.
- The initialization function initializes all ports and port pins with the configuration set pointed by ConfigPtr by invoking internal API Port_InitConfig(). This function should be called first in order to initialize the port for use otherwise no operation can occur on the MCU ports and port pins. This function is also called after reset, in order to reconfigure the ports and port pins of the MCU.
- Port Set Pin Mode: This API will change the pin mode to the requested mode.
- Port_SetToDioMode: This API will set the mode of a pin to DIO mode.
- Port_SetToAlternateMode: This API will set the mode of a port pin to Alternate mode.
- Port SetPinDirection: This API will change the direction of the pin to the requested direction.
- Port RefreshPortDirection: This API will refresh all the port pins to the configured value except the pins that are configured as pin direction changeable during runtime.

Note: The <Device_name> indicates the device to be compiled, which can be 701370A (CPU1(PE1)), 701371(CPU1(PE1)), 701372(CPU1(PE1)), 701373, 701374 , <Compiler> indicate, comp_201517, <AUTOSAR_version> indicates 4.0.3 and <SubVariant> indicate P1H-CE, P1H-C, P1H-M.

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

- If any configuration changes (only post-build) are made to the ECU Configuration Description files

  “X1X\P1x-C\modules\port\sample_application\<SubVariant> \<AUTOSAR_version>\config\App_PORT_P1x-C_701370A_Sample.arxml”

\App_PORT_P1x-C_701371_Sample.arxml”
\App_PORT_P1x-C_701372_Sample.arxml”
\App_PORT_P1x-C_701373_Sample.arxml”
\App_PORT_P1x-C_701374_Sample.arxml”

- The database alone can be generated by using the following commands.
  make –f App_PORT_P1x-C_Sample.mak generate_port_config
  make –f App_PORT_P1x-C_Sample.mak App_PORT_P1x-C_Sample.s37

- After this, a flash able Motorola S-Record file App_PORT_P1x-_Sample.s37 is available in the output folder.

13.3. Memory and Throughput

13.3.1. ROM/RAM Usage

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

Typical PORT configuration

DET OFF
All other Pre-Compile switches ON

Table 13-2 ROM/RAM Details without DET

Sl. No.	ROM/RAM	Segment Name	Size in bytes
1	ROM	PORT_CFG_DATA_UNSPECIFIED	1322
		CONST_ROM_UNSPECIFIED	96
		PORT_PUBLIC_CODE_ROM	1252
		PORT_PRIVATE_CODE_ROM	2774
2	RAM	RAM_UNSPECIFIED	4
		RAM_1BIT	0

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

Table 13-3 ROM/RAM Details with DET

Sl. No.	ROM/RAM	Segment Name	Size in bytes
1	ROM	PORT_CFG_DATA_UNSPECIFIED	1322
		CONST_ROM_UNSPECIFIED	96
		PORT_PUBLIC_CODE_ROM	1494
		PORT_PRIVATE_CODE_ROM	3168
2	RAM	RAM_UNSPECIFIED	4
		RAM_1BIT	1

13.3.2. Stack Depth

The worst-case stack depth for PORT Driver Component for the typical configuration is 104 bytes.

13.3.3. Throughput Details

The throughput details of the APIs shall be as following: The clock frequency used to measure the throughput is 160 MHz for all APIs.

Table 13-4 Throughput Details of the APIs

Sl. No.	API Name	Throughput in microseconds	Remarks
1	Port_Init	38.450	-
2	Port_SetPinDirection	2.175	-
3	Port_RefreshPortDirection	3.212	-
4	Port_GetVersionInfo	0.100	-
5	Port_SetPinMode	5.762	-
6	Port_SetToDioMode	1.550	-
7	Port_SetToAlternateMode	1.587	-
8	Port_SetPinDefaultDirection	1.275	-
9	Port_SetPinDefaultMode	1.850	-

13.4. Critical Section Details

The critical section throughput details are listed below. The clock frequency used to measure the throughput is 160MHz for all APIs.

Table 13-5 Critical Section Throughput Details of the APIs

Sl. No.	API Name	Critical section throughput in microseconds in GHS for 701372 (CPU1(PE1))	Remarks
1	Port_Init	NA	-
2	Port_SetPinDirection	0.950	-
3	Port_RefreshPortDirection	2.849	-
4	Port_GetVersionInfo	NA	-
5	Port_SetPinMode	1.862	-
6	Port_SetToDioMode	0.687	-
7	Port_SetToAlternateMode	0.725	-
8	Port_SetPinDefaultDirection	0.312	-
9	Port_SetPinDefaultMode	0.737	-

Chapter 14 Release Details

PORT Driver Software R4.0.3

Version: 1.0.4

Revision History

Sl.No.	Description	Version	Date
1.	Initial Version	1.0.0	17-Aug-2015
2.	The following changes are made 1. Chapter-2 Reference Documents section updated. 2. Section 4.2 Preconditions updated. 3. Section 4.6 Data Consistency has updated. 4. Chapter-13 P1x-C specific information updated for device support. 5. In Chapter-13, Section- 13.4.4 Sample Application Structure updated. 6. In Chapter-13, Section-13.4 Memory and Throughput, updated the ROM/RAM details, and Throughput Details. 7. Chapter-14 Driver Software version is updated. 8. Added R Number in last page	1.0.1	04-Apr-2016
3.	The following changes are made : 1. Removed the section 13.2. Compiler, Linker and Assembler. 2. Updated section 4.3 by adding a note. 3. Updated section 4.1 by adding a statement. 4. Chapter 8 updated for sub section heading change and missing stub files inclusion. 5. Section 4.4 updated for critical section protection 6. Chapter 6 Registers Details updated. 7. In Chapter 8, Port_Cbk.h file detail is updated. 8. Chapter 11, Section 11.1 updated for Port_GetVersionInfo 9. Section 11.2 added in the chapter Chapter 11 10. Removed PORT_CFG_DBTOC_UNSPECIFIED details in Chapter 12 11. Table 13-1 PDF information for P1x-C added in the Chapter 13 12. 13.2.1.Sample Application Structure updated for Dem stub 13. Device name updated. 14. User’s name changed to User's in the title.	1.0.2	10-Feb-2017
4.	The following changes are made 1. Subsections are added to Section 10.3 2. In Section 4.3 the Note for Table 4-4 is updated 3. Section 4.1 is updated with information about initialization of unused Port pins 4. Notice and copyright are updated 5. Description about Inverting the output level of a pin is added in section 4.1 6. Table 4-2 updated and Note in section 4.4 is corrected. 7. .one and .html files are removed from section 3.1 and 13.2 8. R-Number is updated	1.0.3	27-Apr-2017
5.	Following changes are made 1. Memory and Throughput details updated in chapter 13. 2. R-Number updated.	1.0.4	16-Jun-2017

AUTOSAR MCAL R4.0.3 User's Manual
PORT Driver Component Ver.1.0.4
Embedded User's Manual

Publication Date: Rev. 1.02, June 16, 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 MCAL R4.0.3

User's Manual

Renesas Electronics Corporation

R20UT3653EJ0102