

# **AUTOSAR MCAL R4.0.3**

User's Manual

PORT Driver Component Ver.1.0.8 Embedded User's Manual

Target Device: RH850/P1x

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

#### 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.
- 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
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
CAN	Controller Area Network
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DIO	Digital Input Output
ECU	Electronic Control Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
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
LIN	Local Interconnect Network
MCAL	Microcontroller Abstraction Layer
MCU	MicroController Unit
MHz	Mega Hertz
NA	Not Applicable
OS	Operating System
PDF	Parameter Definition File
PWM	Pulse Width Modulation
RAM	Random Access Memory
ROM	Read Only Memory
RTE	Runtime Environment
SCI	Serial Communication Interface
SPI	Serial Peripheral Interface
SWS	Software Requirements Specification
WDT	Watchdog Timer

### **Definitions**

Term	Represented by	
PORT pin	Numeric identifier linked to a hardware PORT pin	
Sl. No.	Serial Number	
PORT	Represents a whole configurable port on a microcontroller device.	

# **Table of Contents**

Chapter 1		Introduction	11
1.1.	Docume	nt Overview	13
Chap	ter 2	Reference Documents	15
Chap	ter 3	Integration and Build Process	17
3.1.	PORT D	river Component Makefile	17
	3.1.1	Folder structure	17
Chap	ter 4	Forethoughts	19
4.1.	General		19
4.2.	Precond	litions	19
4.3.	User Mo	de and Supervisor Mode	20
4.4.	Data Co	nsistency	21
4.5.	Deviatio	n List	23
Chap	ter 5	Architecture Details	25
Chap	ter 6	Registers Details	27
•	ter 7 ponent	Interaction Between The User And PORT Driver 31	
7.1.	Services	s Provided By PORT Driver Component To User	31
-	ter 8 ription	PORT Driver Component Header And Source File 33	
Chap	ter 9	Generation Tool Guide	35
Chap	ter 10	Application Programming Interface	37
10.1.	Importe	d Types	
	10.1.1	Standard Types	37
	10.1.2	Other Module Types	37
10.2.	Type De	finitions	37
	10.2.1	Port_ConfigType	37
	10.2.2	Port_PinType	39
	10.2.3	Port_PinDirection Type	39
	10.2.4	Port_PinModeType	40
10.3.	Function	n Definitions	41
Chap	ter 11	Development And Production Errors	43
11.1.	PORT D	river Component Development Errors	43
11.2.	PORT D	river Component Production Errors	44

Chapter 12		Memory Organization	
Chap	ter 13	P1M Specific Information	47
13.1.	Interacti	ion between the User and PORT Driver Component	47
	13.1.1.	Translation Header File	47
	13.1.2.	Parameter Definition File	48
	13.1.3.	Services Provided By PORT Driver Component to the User	48
13.2.	Sample	Application	48
	13.2.1.	Sample Application Structure	48
	13.2.2.	Building Sample Application	50
		13.2.2.1. Configuration Example	50
		13.2.2.2. Debugging the Sample Application	50
13.3.	Memory	and Throughput	51
	13.3.1.	ROM/RAM Usage	51
	13.3.2.	Stack Depth	52
	13.3.3.	Throughput Details	52
Chap	ter 14	Release Details	53

# **List of Figures**

Figure 1-1	System Overview Of AUTOSAR Architecture	11
Figure 1-2	System Overview Of The PORT Driver In AUTOSAR MCAL Layer	12
Figure 5-1	PORT Driver Architecture	25
Figure 12-1	PORT Driver Component Memory Organization	45
Figure 13-1	Overview of PORT Driver Sample Application	48
	List of Tables	
Table 4-1	Supervisor mode and User mode details	20
Table 4-2	PORT Driver Protected Resources List	21
Table 4-3	PORT Driver Deviation List	23
Table 6-1	Register Details	27
Table 8-1	Description of the PORT Driver Component Files	34
Table 10-1	Function Definitions	41
Table 11-1	DET Errors of PORT Driver Component	43
Table 11-2	DEM Errors of PORT Driver Component	44
Table 13-1	PDF information for P1M	48
Table 13-2	ROM/RAM Details without DET	51
Table 13-3	ROM/RAM Details with DET	
Table 13-4	Throughput Details of the APIs	52

Introduction Chapter 1

# **Chapter 1** Introduction

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

This document shall be used as reference by the users of PORT Driver Component for P1M Device. The information specific to P1M Device like, compiler, linker, assembler, integration and build process for application along with the memory consumption and throughput information are provided.

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.

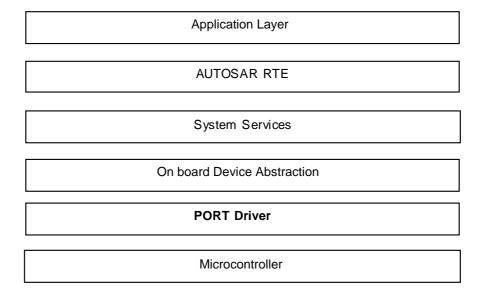


Figure 1-1 System Overview Of AUTOSAR Architecture

Chapter 1 Introduction

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 pins. The tool generates Port\_Cfg.h, Port\_Cbk.h and Port\_PBcfg.c files.

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

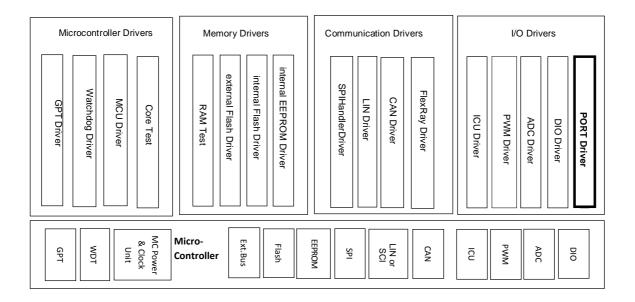


Figure 1-2 System Overview Of The PORT Driver In AUTOSAR MCAL Layer

Introduction Chapter 1

## 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 (Register 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. 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 handled by PORT Driver.
Section 12 (Memory Organization)	This section provides the typical memory organization, which must be met for proper functioning of component.
Section 13 (P1M Specific Information)	This section describes the details of the P1M Sample Application with its folder structure, the information about RAM/ROM usage, stack depth and throughput details.
Section 14 (Release Details)	This section provides release details with version name.

Chapter 1 Introduction

Reference Documents Chapter 2

# **Chapter 2** Reference Documents

SI. No.	Title	Version
1.	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.	r01uh0436ej111_rh850p1x.pdf	1.11
4.	AUTOSAR_SWS_CompilerAbstraction.pdf	3.2.0
5.	AUTOSAR_SWS_MemoryMapping.pdf	1.4.0
6.	AUTOSAR_SWS_PlatformTypes.pdf	2.5.0
7.	AUTOSAR_BSW_MakefileInterface.pdf	0.3

Chapter 2 Reference Documents

#### **Integration and Build Process** Chapter 3

In this section the folder structure of the PORT Driver Component is explained. Description of the Makefiles 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 "R20UT3723EJ0100-AUTOSAR.pdf".

#### 3.1. **PORT Driver Component Makefile**

The Makefile 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 Makefile (of the application) to link and build the final application executable.

#### 3.1.1 Folder structure

The files are organized in the following folders:

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\modules\port\sample application\<SubVariant>\make\<Compiler> \App\_PORT\_P1M\_Sample.mak

X1X\P1x\modules\port\sample\_application\<SubVariant>\obj\<Complier>

X1X\common\_platform\modules\port\generator\Port\_X1x.dll

X1X\P1x\common\_family\generator

\Global\_Application\_P1x.trxml \Sample\_Application\_P1x.trxml

\P1x translation.h

X1X\P1x\modules\port\user\_manual (User manuals will be available in this folder)

X1X\P1x\modules\port\generator

\R403\_PORT\_P1x\_BSWMDT.arxml

Note: 1. <Compiler> can be ghs.

- 2. <AUTOSAR\_version> should be 4.0.3.
- 3. <SubVariant> can be P1M.

Forethoughts Chapter 4

# **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 PortUnlimitedCurrentControl 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 functions/variables with "STATIC" as pilot tag, provides an
  indication to the compiler that the function/variable following this tag is a
  static type. So the scope of static functions/variables is restricted to the
  file where they are declared. User should take care of the tag name with
  respect to compiler used.

### 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\_Cfg.h and Port\_Cbk.h files generated by the PORT Driver Component Code Generation Tool must be compiled and Chapter 4 Forethoughts

- 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.

### 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
-----------	---------------------------------------

SI.No	API Name	User Mode	Supervisor mode	Known limitation in User mode
1	Port_Init	х	х	-
2	Port_SetPinDirection	х	х	-
3	Port_RefreshPortDirection	х	Х	-
4	Port_SetPinMode	х	Х	-

Forethoughts Chapter 4

SI.No	API Name	User Mode	Supervisor mode	Known limitation in User mode
5	Port_SetToDioMode	х	х	-
6	Port_SetToAlternateMode	х	х	-
7	Port_SetPinDefaultDirection	х	х	-
8	Port_SetPinDefaultMode	х	Х	-
9	Port_GetVersionInfo	х	х	-

Note: Implementation of critical section is not dependent on MCAL. Hence critical section is not considered to the entries for User mode in the above table.

### 4.4. 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 DIR PROTECTION
- PORT SET TO DIO ALT PROTECTION
- PORT\_INIT\_CONFIG\_PROTECTION
- PORT\_REFRESHPORT\_INTERNAL\_PROTECTION
- PORT\_SET\_PIN\_DEFAULT\_MODE\_PROTECTION
- PORT\_SET\_PIN\_DEFAULT\_DIR\_PROTECTION

The functions SchM\_Enter\_Port\_<Exclusive Area> and SchM\_Exit\_Port\_<Exclusive Area> can be disabled through the configuration parameter 'PortCriticalSectionProtection'.

Table 4-2 PORT Driver Protected Resources List

API Name	Exclusive Area Type	Protected Resources
Port_Init	PORT_INIT_CONFIG_PROTECTION	HW registers:PFCEn, PFCn, PFCAEn, PMCSRn,PMSRn,JPFCE0, JPMCSR0,JPMSR0,PSRn, JPSR0,PISAn,JPISA0,PIB Cn,JPIBC0,PIPCn,PUn,JP U0,PDn,JPD0,PBDCn,JPB DC0,PODCn,JPODC0,PO DCEn,PDSCn,PUCCn,JPD SC0,JPUCC0,PINVnDNFA nCTL,DNFAnENL,DNFCK SnC, FCLAnCTL
Port_SetPinDirection	PORT_SET_PIN_DIR_PROTECTION	HWregisters:PSRn,JPSR0, PMSRn,PINVn,JPMSR0

Chapter 4 Forethoughts

API Name	Exclusive Area Type	Protected Resources
Port_RefreshPortDire ction	PORT_REFRESHPORT_INTERNAL_PR OTECTION	HW registers:PMSRn, JPMSR0.
Port_SetPinMode	PORT_SET_PIN_MODE_PROTECTION	HWregisters:PIPCn,PMSRn ,PMCSRn,PSRn,JPMSRn,J PMCSR0,JPSR0,PFCEn,P FCn,PFCAEn,JPFCE0.
Port_SetToDioMode	PORT_SET_TO_DIO_ALT_PROTECTIO N	HW registers:PMCSRn, PIPCn, JPMCSR0
Port_SetToAlternateM ode	PORT_SET_TO_DIO_ALT_PROTECTIO N	HW registers:PMCSRn, PIPCn, JPMCSR0
Port_SetPinDefaultMo de	PORT_SET_PIN_DEFAULT_MODE_PRO TECTION	HW registers:PMCSRn, PMSRn, PIPCn, JPMCSR0,JPMSR0,PFCE n,PFCn,PFCAEn,JPFCE0.
Port_SetPinDefaultDir ection	PORT_SET_PIN_DEFAULT_DIR_PROTE CTION	HW registers:PMSRn, JPMSR0
Port_GetVersionInfo	None	None

**Note:** The highest measured duration of a critical section is 43.224 micro seconds measured for Port\_Init API.

Forethoughts Chapter 4

# 4.5. Deviation List

Table 4-3 PORT Driver Deviation List

SI. 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 containers 'PortContainer' and 'PortPin' with all its parameters are considered as unused.</n>	-
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	-
5.	Setting the parameter PortPinLevelValue to "PIN_LEVEL_HIGH" will not work for PortPin2 and PortPin4 of JTag PortGroup.	

Chapter 4 Forethoughts

Architecture Details Chapter 5

## **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:

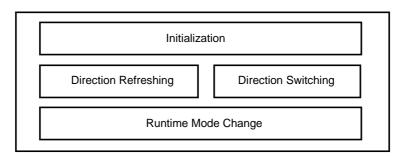


Figure 5-1 PORT Driver Architecture

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.

Chapter 5 Architecture Details

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

Registers Details Chapter 6

# **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	Register s	Config 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
	32 bit	r	PPROTSn	-	PORT_LONG_WORD_ON E
	32 bit	W	PPCMDn	-	PORT_WRITE_ERRO R_CLEAR_VAL
Port_RefreshPortDirecti on	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	PortlpControl PortPinInitialMode PortPinDioAltModeChangeable	usOrMask
	32 bit	rw	JPMCSR0	PortPinDioAltModeChangeable PortPinInitialMode	usOrMask
Port_SetToAlternateMo de	32 bit	rw	PMCSRn	PortPinDioAltModeChangeable PortPinInitialMode	usOrMask
	16 bit	rw	PIPCn	PortIpControl PortPinInitialMode PortPinDioAltModeChangeable	usOrMask
	32 bit	rw	JPMCSR0	PortPinDioAltModeChangeable PortPinInitialMode	usOrMask
Port_SetPinDefaultMod e	32 bit	rw	PMCSRn	PortPinModeChangeable, PortPinInitialMode PortPinDirection	usOrMask usInitModeRegVal

Chapter 6 Registers Details

API Name	Register Access 8/16/32 bits	Register Access r/w/rw	Register s	Config Parameter	Macro/Variable
	32 bit	rw	PMSRn	PortPinModeChangeable, PortPinInitialMode PortPinDirection	usOrMask usInitModeRegVal
	16 bit	rw	PIPCn	PortPinModeChangeable, PortlpControl	usOrMask usInitModeRegVal
	32 bit	rw	JPMCSR0	PortPinModeChangeable, PortPinInitialMode PortPinDirection	usOrMask usInitModeRegVal
	32 bit	rw	JPMSR0	PortPinModeChangeable, PortPinInitialMode PortPinDirection	usOrMask usInitModeRegVal
	16 bit	rw	PFCEn	PortPinModeChangeable, PortPinInitialMode	usOrMask usInitModeRegVal
	16 bit	rw	PFCn	PortPinModeChangeable, PortPinInitialMode	usOrMask usInitModeRegVal
	16 bit	rw	PFCAEn	PortPinModeChangeable, PortPinInitialMode	usOrMask usInitModeRegVal
	8 bit	rw	JPFCE0	PortPinModeChangeable, PortPinInitialMode	usOrMask usInitModeRegVal
Port_SetPinDefaultDire ction	32 bit	rw	PMSRn	PortPinDirection, PortPinDirectionChangeable	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
	16 bit	rw	PFCAEn	PortPinModeChangeable,	usOrMask
	8 bit	rw	JPFCE0	PortPinModeChangeable,	usOrMask
Port_Init	32 bit	rw	PSRn	PortPinLevelValue	usInitModeRegVal

Registers Details Chapter 6

API Name	Register Access 8/16/32 bits	Register Access r/w/rw	Register s	Config Parameter	Macro/Variable
	32 bit	rw	JPSR0	PortPinLevelValue	usInitModeRegVal
	32 bit	rw	PMSRn	PortPinDirection,	usInitModeRegVal
	32 bit	rw	PMCSRn	PortPinInitialMode	usInitModeRegVal
	16 bit	rw	PISAn	PortInputSelection	usInitModeRegVal
	8 bit	rw	JPISA0	PortInputSelection	usInitModeRegVal
	16 bit	rw	PIBCn	PortInputBufferControl	usInitModeRegVal
	8 bit	rw	JPIBC0	PortInputBufferControl	usInitModeRegVal
	16 bit	rw	PIPCn	PortlpControl	usInitModeRegVal
	16 bit	rw	PUn	PullUpOption	usInitModeRegVal
	8 bit	rw	JPU0	PullUpOption	usInitModeRegVal
	16 bit	rw	PDn	PullDownOption	usInitModeRegVal
	8 bit	rw	JPD0	PullDownOption	usInitModeRegVal
	16 bit	rw	PBDCn	PortBiDirectionControl	usInitModeRegVal
	8 bit	rw	JPBDC0	PortBiDirectionControl	usInitModeRegVal
	8 bit	rw	DNFAnCTL	PortSameLevelSamples  PortSamplingClockFrequency	ucDNFACTL
	8 bit	rw	FCLAnCTL	PortDigitalFilterEdgeControl	ucFCLACTL
	8 bit	rw	m DNFAnENL	PortDigitalFilterEnableInput	ucDNFAENL
	32 bit	rw	DNFCKSnC	PortClockSource	ulDNFCKS
	8 bit	rw	JPFCE0	PortPinInitialMode	usInitModeRegVal
	32 bit	rw	JPMCSR0	PortPinInitialMode	usInitModeRegVal
	32 bit	rw	JPMSR0	PortPinDirection,	usInitModeRegVal
	16 bit	rw	PFCEn	PortPinInitialMode	usInitModeRegVal
	16 bit	rw	PFCn	PortPinInitialMode	usInitModeRegVal
	16 bit	rw	PFCAEn	PortPinInitialMode	usInitModeRegVal
	32 bit	W	PPCMDn	-	PORT_WRITE_ERROR_ CLEAR_VAL

Chapter 6 Registers Details

API Name	Register Access 8/16/32 bits	Register Access r/w/rw	Register s	Config Parameter	Macro/Variable
	32 bit	r	PPROTSn	-	PORT_LONG_WORD_ON E
	32 bit	W	JPPCMD0	-	PORT_WRITE_ERROR_ CLEAR_VAL
	32 bit	r	JPPROTS0	-	PORT_LONG_WORD_ON E
	32 bit	W	PODCn	PortOpenDrainControlExpansi on	usInitModeRegVal
	32 bit	W	JPODC0	PortOpenDrainControlExpansi on	usInitModeRegVal
	32 bit	W	PODCEn	PortOpenDrainControlExpansi on	usInitModeRegVal
	32 bit	w	PDSCn	PortDriveStrengthControl	usInitModeRegVal
	32 bit	w	JPDSC0	PortDriveStrengthControl	usInitModeRegVal
	32 bit	W	PUCCn	PortUnlimitedCurrentControl	usInitModeRegVal
	32 bit	w	JPUCC0	PortUnlimitedCurrentControl	usInitModeRegVal
	32 bit	W	PINVn	PortOutputLevelInversion	usInitModeRegVal
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 Component 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

The C source file generated by PORT Driver Generation Tool:

Port\_PBcfg.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
- Rte.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 pin 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.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_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_RegWrite.h	This file contains macro definitions for the registers write and Register write verification.			
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.			
Rte.h	This file is a stub for Rte Component.			

Generation Tool Guide Chapter 9

# **Chapter 9** Generation Tool Guide

For more information on the PORT Driver Component Generation Tool, please refer "R20UT3723EJ0100-AUTOSAR.pdf".

Chapter 9 Generation Tool Guide

## **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:	Туре	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

Туре	Name	Explanation
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	pPinModeChangeableGrou ps	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	pPinModeChangeableDetai Is	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.
Port_DNFCKSRegs		Pointer to the DNFCKS registers structure
uint8	ucNoOfPinsDirChangeable	Total number of Pins configured for Direction Changeable at run time
uint8	ucNoOfPinsModeChangea ble	Total number of Pins configured for mode Changeable at run time
uint8	ucNoOfPinsDioAltModeCha ngeable	Total number of Pins configured for mode Changeable at run time

	Туре	Name	Explanation
	uint8	ucNoOfDNFARegs	The total number of DNFA noise elimination registers
	uint8	ucNoOfFCLARegs	The total number of FCLA noise elimination registers
	uint8	ucNoOfDNFCKSRegs	The total number of DNFCKS 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_PinDirectionIType	
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_Pi	nModeType	
Туре:	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
	А	APP_ALT5_OUT	(Port_PinModeType)0x0A
	В	APP_ALT5_IN	(Port_PinModeType)0x0B
	С	APP_ALT6_OUT	(Port_PinModeType)0x0C
	D	APP_ALT6_IN	(Port_PinModeType)0x0D
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
	8	APP_ALT5_OUT_SET_PIPC	(Port_PinModeType)0x8A
	9	APP_ALT5_IN_SET_PIPC	(Port_PinModeType)0x8B
	Α	APP_ALT6_OUT_SET_PIPC	(Port_PinModeType)0x8C
	В	APP_ALT6_IN_SET_PIPC	(Port_PinModeType)0x8D
Description:	These are the possible modes; a port pin can have for both input and output.		

## 10.3. Function Definitions

**Table 10-1** Function Definitions

SI.No	API's	API's specific
1	Port_Init	-
2	Port_SetPInDirection	-
3	Port_RefreshPortDirection	-
4	Port_SetPinMode	-
5	Port_GetVersionInfo	-
6	Port_SetToDioMode	-
7	Port_SetToAlternateMode	-
8	Port_SetPinDefaultDirection	-
9	Port_SetPinDefaultMode	-

## **Chapter 11 Development And Production Errors**

In this section the development and production 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

SI. No.	1
Error Code	PORT E PARAM CONFIG
Related API(s)	Port Init
Source of Error	API is invoked with NULL Pointer
SI. No.	2
Error Code	PORT_E_INVALID_DATABASE
Related API(s)	Port Init
Source of Error	Invalid database is found
SI. No.	3
Error Code	PORT_E_UNINIT
Related API(s)	Port_RefreshPortDirection, Port_SetPinDirection, Port_SetPinMode, Port_SetToDioMode, Port_SetToAlternateMode, Port_SetPinDefaultMode, Port_SetPinDefaultDirection
Source of Error	APIs are invoked without the initialization of the PORT Driver Component.
SI. No.	4
Error Code	PORT_E_PARAM_PIN
Related API(s)	Port_SetPinMode, Port_SetPinDirection, Port_SetToDioMode, Port_SetToAlternateMode, Port_SetPinDefaultMode, ort_SetPinDefaultDirection
Source of Error	API is invoked with invalid Pin
SI. No.	5
Error Code	PORT_E_PARAM_INVALID_MODE
Related API(s)	Port_SetPinMode
Source of Error	API is invoked with invalid mode
SI. No.	6
Error Code	PORT_E_DIRECTION_UNCHANGEABLE
Related API(s)	Port_SetPinDirection, Port_SetPinDefaultDirection
Source of Error	API is invoked with Pin which is not configured as 'Direction Changeable during run time'.
SI. No.	7
Error Code	PORT_E_MODE_UNCHANGEABLE

Related API(s)	Port_SetPinMode, Port_SetToDioMode, Port_SetToAlternateMode, Port_SetPinDefaultMode
Source of Error	API is invoked with Pin which is not configured as 'Mode Changeable during run time'.
SI. 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

SI. No.	1
Error Code	PORT_E_WRITE_TIMEOUT_FAILURE
Related API(s)	Port_Init ,Port_SetPinDirection
Source of Error	When writing to a write-protected register fails.
SI. No.	2
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.

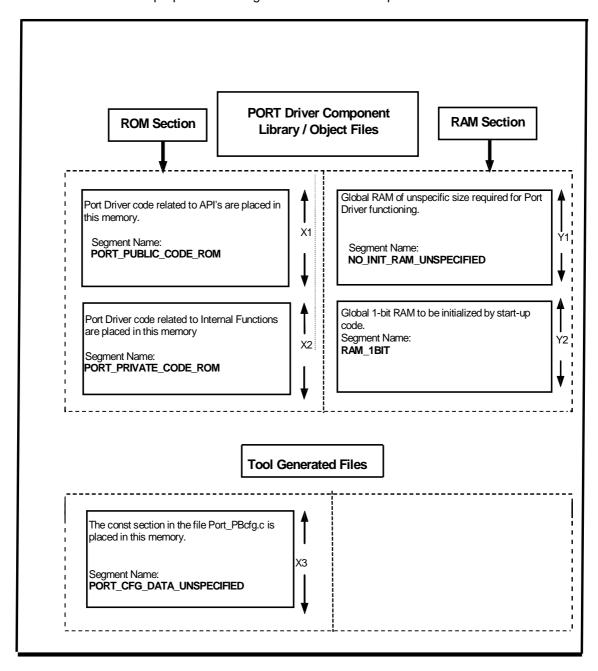


Figure 12-1 PORT Driver Component Memory Organization

#### ROM Section (X1. X2. X3):

**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.

#### RAM Section (Y1 and Y2):

**NO\_INIT\_RAM\_UNSPECIFIED (Y1):** This section consists of the global RAM 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 P1M Specific Information**

P1M supports following devices:

- R7F701304
- R7F701305
- R7F701310
- R7F701311
- R7F701312
- R7F701313
- R7F701314
- R7F701315
- R7F701318
- R7F701319
- R7F701320
- R7F701321
- R7F701322
- R7F701323

# 13.1. Interaction between the User and 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. Translation Header File

The translation header file supports following devices:

- R7F701304
- R7F701305
- R7F701310
- R7F701311
- R7F701312
- R7F701313
- R7F701314
- R7F701315
- R7F701318
- R7F701319
- R7F701320
- R7F701321
- R7F701322
- R7F701323

#### 13.1.2. Parameter Definition File

Parameter definition files support information for P1M

Table 13-1 PDF information for P1M

PDF Files	Devices Supported
R403_PORT_P1M_04_05	701304, 701305
R403_PORT_P1M_10_11_14_15	701310, 701311, 701314, 701315
R403_PORT_P1M_12_13	701312, 701313
R403_PORT_P1M_18_19_22_23	701318, 701319, 701322, 701323
R403_PORT_P1M_20_21	701320, 701321

#### 13.1.3. Services Provided By PORT Driver Component to the User

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.

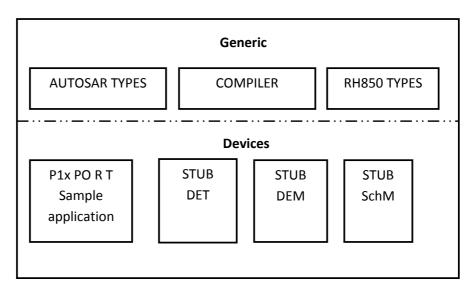


Figure 13-1 Overview of PORT Driver Sample Application

The Sample Application of the P1M is available in the path

X1X\P1x\modules\port\sample\_application

The Sample Application consists of the following folder structure:

X1X\P1x\modules\port\definition\<AUTOSAR\_version>\<SubVariant>\ R403\_PORT\_P1M\_04\_05.arxml R403\_PORT\_P1M\_10\_11\_14\_15.arxml R403\_PORT\_P1M\_12\_13.arxml R403\_PORT\_P1M\_18\_19\_22\_23.arxml R403\_PORT\_P1M\_20\_21.arxml

X1X\P1x\modules\port\sample\_application\<SubVariant>\ <AUTOSAR version>\

<AUTOSAR\_version> \src\Port\_PBcfg.c \include\Port\_Cfg.h \include\Port\_Cbk.h

\config\App\_PORT\_P1M\_701304\_Sample.arxml \config\App\_PORT\_P1M\_701305\_Sample.arxml \config\App\_PORT\_P1M\_701310\_Sample.arxml \config\App\_PORT\_P1M\_701311\_Sample.arxml \config\App\_PORT\_P1M\_701312\_Sample.arxml \config\App\_PORT\_P1M\_701313\_Sample.arxml \config\App\_PORT\_P1M\_701315\_Sample.arxml \config\App\_PORT\_P1M\_701318\_Sample.arxml \config\App\_PORT\_P1M\_701319\_Sample.arxml \config\App\_PORT\_P1M\_701320\_Sample.arxml \config\App\_PORT\_P1M\_701321\_Sample.arxml \config\App\_PORT\_P1M\_701322\_Sample.arxml \config\App\_PORT\_P1M\_701322\_Sample.arxml \config\App\_PORT\_P1M\_701323\_Sample.arxml \config\App\_PORT\_P1M\_701323\_Sample.arxml \config\App\_PORT\_P1M\_701323\_Sample.arxml \config\App\_PORT\_P1M\_701323\_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 passed 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 direction of Port Pin during runtime
- 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 will switch the port pin to the default direction during runtime

- 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 service sets the mode of Port Pin to default mode during runtime

#### 13.2.2. Building Sample Application

#### 13.2.2.1. Configuration Example

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

**Configuration Details**: App\_PORT\_P1M\_701310\_Sample.arxml

#### 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\common\_family\make\<Compiler>"

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

SampleApp.bat Port 4.0.3 < Device\_name >

 After this, the tool output files will be generated with the configuration as mentioned in App\_PORT\_P1M\_701310\_Sample.arxml file available in the path:

"X1X\P1x\modules\port\sample\_application\<SubVariant>\<AUTOSAR\_version>\config\App\_PORT\_P1M\_701310\_Sample.arxml"

- After this, all the object files, map file and the executable file App\_PORT\_P1M\_Sample.out will be available in the output folder: ("X1X\P1x\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\_SetPinMode: 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 701304 or 701305 or 701310 or 701311 or 701312 or 701313 or 701314 or 701315 or 701318 or 701319 or 701320 or 701321 or 701322 or 701323.

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

- If any configuration changes (only post-build) are made to the ECU Configuration Description files
  - "X1X\P1x\modules\port\sample\_application\<SubVariant> \<AUTOSAR\_version>\config\App\_PORT\_P1M\_701310\_Sample.arxml"
- The database alone can be generated by using the following commands.
   make –f App\_PORT\_P1M\_Sample.mak generate\_port\_config
   make –f App\_PORT\_P1M\_Sample.mak App\_PORT\_P1M\_Sample.s37
- After this, a flash able Motorola S-Record file App\_PORT\_P1M\_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 as provided in Section 13.2.2.1 *Configuration Example* are provided in this section.

Table 13-2	ROM/RAM Details without DET

SI. No.	ROM/RAM	Segment Name	Size in bytes for 701312	Size in bytes for 701310
1	ROM	PORT_PUBLIC_CODE_ROM	-	1140
		PORT_PRIVATE_CODE_ROM	-	3052
		PORT_CFG_DATA_UNSPECIFIED	-	610
2	RAM	RAM_1BIT	-	0
		NO_INIT_RAM_UNSPECIFIED	-	4

The details of memory usage for the typical configuration, with DET enabled and all other configurations as provided in 13.2.2.1 *Configuration Example* are provided in this section.

Table 13-3 ROM/RAM Details with DET

SI. No.	ROM/RAM	Segment Name	Size in bytes for 701312	Size in bytes for 701310
1	ROM	PORT_PUBLIC_CODE_ROM	-	1468
		PORT_PRIVATE_CODE_ROM	-	3374
		PORT_CFG_DATA_UNSPECIFIED	-	628
2	RAM	RAM_1BIT	-	1
		NO_INIT_RAM_UNSPECIFIED	-	4

#### 13.3.2. Stack Depth

The worst-case stack depth for PORT Driver Component for the typical configuration provided in Section 13.2.2.1 *Configuration Example* is 48 byte

#### 13.3.3. Throughput Details

The throughput details of the APIs for the configuration mentioned in the Section 13.2.2.1 *Configuration Example* will be provided in the next release. The clock frequency used to measure the throughput is 80 MHz for all APIs.

Table 13-4 Throughput Details of the APIs

SI. No.	API Name	Throughput in microseconds for 701310	Remarks
1	Port_Init	64.96	-
2	Port_RefreshPortDirection	3.99	-
3	Port_SetPindirection	3.20	-
4	Port_GetVersionInfo	0.18	-
5	Port_SetPinMode	9.43	-
6	Port_SetToDioMode	2.50	-
7	Port_SetToAlternateMode	1.99	-
8	Port_SetPinDefaultDirection	1.48	-
9	Port_SetPinDefaultMode	7.35	-

Release Details Chapter 14

## **Chapter 14** Release Details

**PORT Driver Software** 

Version: 1.5.3

Chapter 14 Release Details

#### **Revision History**

SI.No.	Description	Version	Date
1.	Initial Version	1.0.0	9-Oct-2013
2.	Following changes are made:  1. Sample application is regenerated for the change in parameter definition file.  2. Section 10.2.4 is updated for Port_PinModeType.	1.0.1	21-Nov-2013
3.	<ol> <li>Following changes are made:         <ol> <li>Chapter 2 is updated for referenced documents version.</li> <li>Section 13.1.1 is updated for adding the device names.</li> <li>Section 13.2 is updated for compiler, assembler and linker details.</li> </ol> </li> <li>Section 13.3 is updated to add parameter definition file and sample application configuration files for all P1M devices.</li> <li>Chapter 14 is updated for PORT driver component version information.</li> </ol>	1.0.2	31-Jan-2014
4.	<ol> <li>Following changes are made:         <ol> <li>Chapter 2 is updated for referenced documents version.</li> <li>Section 13.1.1 is updated for adding the device names.</li> <li>Section 13.2 is updated for compiler, assembler and linker details.</li> </ol> </li> <li>Section 13.3 is updated to add parameter definition file and sample application configuration files for all P1M devices.</li> <li>Chapter 14 is updated for PORT driver component version information.</li> <li>Deviation list is updated to add PORT_E_PARAM_POINTER error foe Port_GetVersioInfo API and AUTOSAR requirement.</li> <li>Memory and Throughput details are updated.</li> <li>Section 10.2.1 is updated to add new structure element.</li> </ol>	1.0.3	03-Sep-2014
5.	Following changes are made: 1. Section 13.4.3 updated for Throughput details. 2. Page alignment is updated. 3. Table of contents updated.		05-Sep-2014
6.	<ol> <li>Following changes are made:         <ol> <li>Section 1.1 Document Overview is updated.</li> <li>Chapter 2 Reference documents are updated for version change.</li> <li>Chapter 4 is updated for information regarding Interrupt vector table.</li> </ol> </li> <li>Chapter 6 Port_SetPinMode is updated.</li> <li>Section 10.2.4 Port_PinModeType is updated.</li> <li>Section 13.1.1 is updated for adding new devices.</li> <li>Section 13.2 Compiler, Linker and Assembler section is removed.</li> <li>Section 13.2 is updated for parameter definition file and sample application configuration files of all P1M devices.</li> <li>Section 13.3 Memory and Throughput details are updated.</li> </ol>	1.0.5	29-Apr-2015

SI.No.	Description	Version	Date
7	<ol> <li>Chapter 2 Reference documents are updated for version change.</li> <li>Definitions in Page 6 is updated for "PORT Idle State"</li> <li>Section 4.1 is updated for information "The parameter PortDriveStrengthControl has dependency on parameter PortUnlimitedCurrentControl while specifying the output driving abilities of port pins"</li> <li>Section 4.1 is updated for information "Port_SetToDioMode and Port_SetPinDefaultMode Api shall not change or affect the level of the requested pin" and information on accessing the Hardware registers.</li> <li>Section 4.4 is updated to add Exclusive areas.</li> <li>Chapter 6, Table 6-1, Register Details is updated for Apis Port_SetPinDirection, Port_SetPinMode, Port_InitConfig, Port_SetPinDefaultMode.</li> <li>Chapter 8 is updated to add Stub C header files</li> <li>Section 10.1.1 updated to add Std_ReturnType</li> <li>Chapter 8 is updated to remove PORT_CFG_DBTOC_UNSPECIFIED in ROM Section and in Figure 12-1</li> <li>Section 13.2.2 updated for correction in API names</li> <li>Updated Chapter 13.3 'Memory and Throughput'.</li> <li>Updated Chapter 14 'Release Details' to change the driver version.</li> <li>Updated Chapter 3 and Chapter 9 to add R-Numbered Tool User manual name.</li> <li>Added R number at end of the document.</li> <li>Copyright information is updated</li> </ol>	1.0.6	29-Mar-2016
8.	<ol> <li>Chapter 1 Introduction is updated.</li> <li>Information on unused pins and Port Pin level inversion is updated in Chapter 4.1 General.</li> <li>The point regarding digital filter time delay has been removed and the information on Port_GetVersionInfo is updated in 4.5 Deviation list.</li> <li>Chapter 5 Architecture Details is updated with the information on Port_GetVersionInfo.</li> <li>Chapter 7 is updated with the information on PORT Driver features.</li> <li>Chapter 8 is updated to add the information about Port_Cbk.h file.</li> <li>Table 11-2 is updated to add the DEM error PORT_E_REG_WRITE_VERIFY.</li> <li>Table 11-1 is updated to add the DET error PORT_E_PARAM_POINTER.</li> <li>Chapter 14 Release Details is updated with Port Driver Software version.</li> </ol>	1.0.7	06-Jun-2016

SI.No.	Descri	ption	Version	Date
		Section 4.3 is updated with note on critical section.		
	11.	Chapter 8 PORT Driver Component Header and Source File		
	12	Description is updated for adding stub files. Chapter 6 Register details is updated.		
		Section 13.2.1 and section 13.2.2 are updated.		
		Chapter 3 and 9 are updated with R-Numbered Tool User		
		manual name.		
		Version of R-number is updated at the end of document.		
	16.	Memory section NOINIT_RAM_UNSPECIFIED is updated to NO_INIT_RAM_UNSPECIFIED in Figure 12-1, Table 13-2 and 13-3.		
	17.	Added precondition items about critical protection and transient hardware faults in chapter 4.2 'Precondition' and worst case duration value is updated as Note in Chapter 4.4 Data Consistency.		
	18.	Section 10.2.1 Port_ConfigType is updated.		
		Port_RegWrite.h added for the section 3.1.1, chapter 8 and Table 8-1.		
	20.	Table 4-2 PORT Driver Protected Resources List added in		
		the section 4.4.		
	21.	Reference document updated in chapter 2.		
	22.	In Page 6 definitions section is updated.		
9.	Followin	Following changes are made:		24-Oct-2016
	1.	Definitions section is updated.		
	2.	Chapter 14 is updated for PORT driver component version information.		
	3.	Chapter 13.3 updated for ROM/RAM Usage, Stack Depth and Throughput Details.		
	4.			
	5.			
	6.	Port Pin level inversion information is updated in 4.5 Deviation list.		
	7.	Section 4.1.General is updated with the information on keyword STATIC.		
	8.			

AUTOSAR MCAL R4.0.3 User's Manual PORT Driver Component Ver.1.0.8 Embedded User's Manual

Publication Date: Rev.1.00, October 24, 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

Tel: +1-9U0-998-941, Fax: +1-9U0-998-3220
Renesas Electronics Europe Limited
Dukes Meadow, Milboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-858-10,0 Fax: +44-1628-585-900
Renesas Electronics Europe GmbH
Arcadiastrase 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: +861-08235-7155, Fax-88610-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China Teit +86-21-6877-1818, Fax: +86-21-6877-7889

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16fr., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tei: +852-2886-9318, Fax: +852-2886-9022/9044
Renesas Electronics Tailwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Tailwan
Tei: +886-2-6175-9600, Fax: +886-2-4175-9670

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

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

Renesas Electronics Korea Co., Ltd.
11F., Samik Lavied or Bildy, 7202 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

© 2016 Renesas Electronics Corporation. All rights reserved.

# AUTOSAR MCAL R4.0.3 User's Manual

