RENESAS

AUTOSAR MCAL R4.0.3
User's Manual

PORT Driver Component Ver.1.0.8

Embedded User's Manual

Target Device:
RH850/P1x

Renesas Electronics
www.renesas.com

Rev.1.00 Oct 2016

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

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 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.

Application Layer

AUTOSAR RTE

System Services

On board Device Abstraction

PORT Driver

Microcontroller
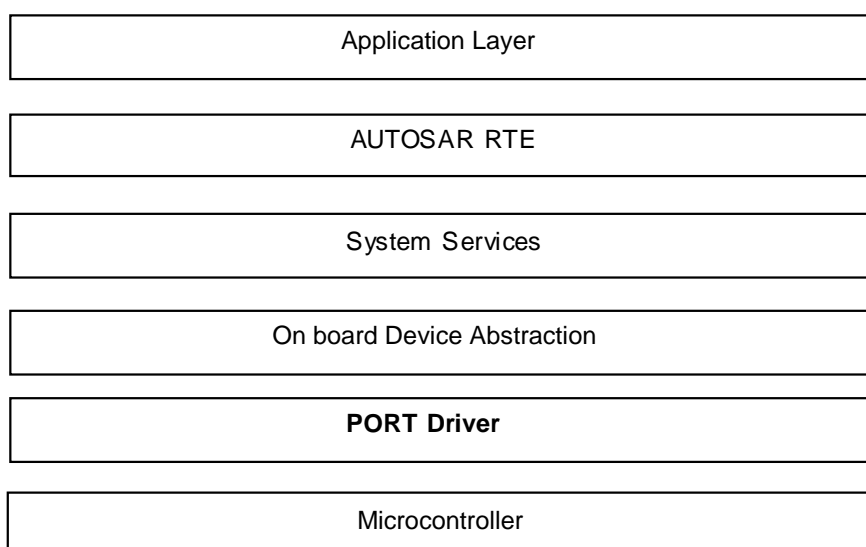
Figure 1-1   System Overview Of AUTOSAR Architecture

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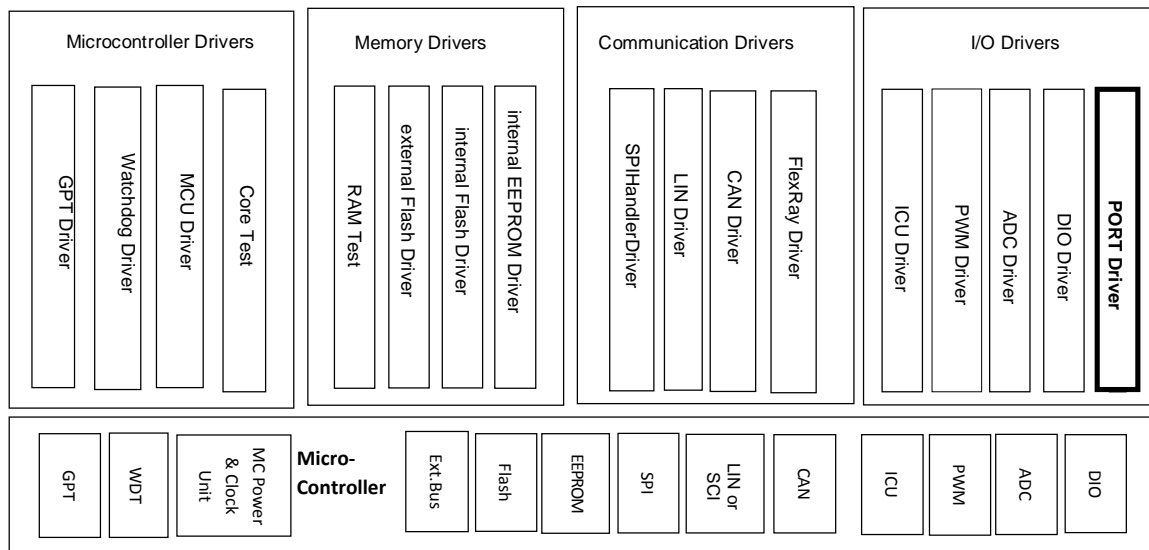
Figure 1-2   System Overview Of The PORT Driver In AUTOSAR MCAL Layer

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 2   Reference Documents

Sl. 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 3   Integration and Build Process

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:

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\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.

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

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

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	-

Sl.No	API Name	User Mode	Supervisor mode	Known limitation in User mode
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.

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

API Name	Exclusive Area Type	Protected Resources
Port_RefreshPortDirection	PORT_REFRESHPORT_INTERNAL_PROTECTION	HW registers:PMSRn, JPMSR0.
Port_SetPinMode	PORT_SET_PIN_MODE_PROTECTION	HWregisters:PIPCn,PMSRn,PMCSRn,PSRn,JPMSRn,JPMCSR0,JPSR0,PFCEn,PFCn,PFCAEn,JPFCE0.
Port_SetToDioMode	PORT_SET_TO_DIO_ALT_PROTECTION	HW registers:PMCSRn, PIPCn, JPMCSR0
Port_SetToAlternateMode	PORT_SET_TO_DIO_ALT_PROTECTION	HW registers:PMCSRn, PIPCn, JPMCSR0
Port_SetPinDefaultMode	PORT_SET_PIN_DEFAULT_MODE_PROTECTION	HW registers:PMCSRn, PMSRn, PIPCn, JPMCSR0,JPMSR0,PFCEn,PFCn,PFCAEn,JPFCE0.
Port_SetPinDefaultDirection	PORT_SET_PIN_DEFAULT_DIR_PROTECTION	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.

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 containers 'PortContainer' and 'PortPin' with 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	-
5.	Setting the parameter PortPinLevelValue to "PIN_LEVEL_HIGH" will not work for PortPin2 and PortPin4 of JTag PortGroup.	

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:

```
┌─────────────────────────────────────────────────────┐
│  ┌───────────────────────────────────────────────┐  │
│  │                 Initialization                │  │
│  └───────────────────────────────────────────────┘  │
│  ┌──────────────────────┐  ┌──────────────────────┐ │
│  │ Direction Refreshing │  │  Direction Switching │ │
│  └──────────────────────┘  └──────────────────────┘ │
│  ┌───────────────────────────────────────────────┐  │
│  │              Runtime Mode Change              │  │
│  └───────────────────────────────────────────────┘  │
└─────────────────────────────────────────────────────┘
```
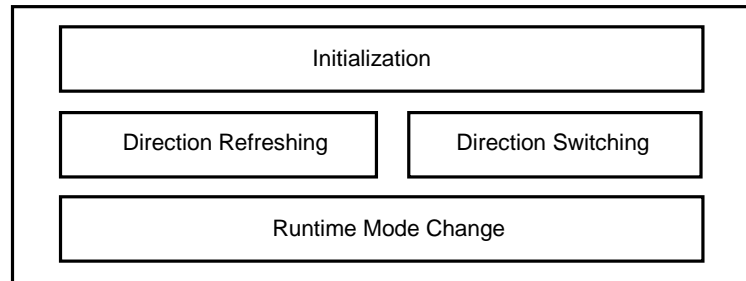
Figure 5-1   PORT Driver Architecture

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.

25

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.

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	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_ONE
	32 bit	w	PPCMDn	-	PORT_WRITE_ERROR_CLEAR_VAL
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	PortPinDioAltModeChangeable PortPinInitialMode	usOrMask
Port_SetToAlternateMode	32 bit	rw	PMCSRn	PortPinDioAltModeChangeable PortPinInitialMode	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

API Name	Register Access 8/16/32 bits	Register Access r/w/rw	Registers	Config Parameter	Macro/Variable
	32 bit	rw	PMSRn	PortPinModeChangeable, PortPinInitialMode PortPinDirection	usOrMask usInitModeRegVal
	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
	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_SetPinDefaultDirection	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

API Name	Register Access 8/16/32 bits	Register Access r/w/rw	Registers	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	PortIpControl	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	ucDNFACTL
				PortSamplingClockFrequency	
	8 bit	rw	FCLAnCTLm	PortDigitalFilterEdgeControl	ucFCLACTL
	8 bit	rw	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

API Name	Register Access 8/16/32 bits	Register Access r/w/rw	Registers	Config Parameter	Macro/Variable
	32 bit	r	PPROTSn	-	PORT_LONG_WORD_ONE
	32 bit	w	JPPCMD0	-	PORT_WRITE_ERROR_CLEAR_VAL
	32 bit	r	JPPROTS0	-	PORT_LONG_WORD_ONE
	32 bit	w	PODCn	PortOpenDrainControlExpansion	usInitModeRegVal
	32 bit	w	JPODC0	PortOpenDrainControlExpansion	usInitModeRegVal
	32 bit	w	PODCEn	PortOpenDrainControlExpansion	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.

Chapter 9   Generation Tool Guide

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

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

Type	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	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.
Port_DNFCKSRegs	pPortDNFCKSRegs	Pointer to the DNFCKS 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

	Type	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_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
	A	APP_ALT5_OUT	(Port_PinModeType)0x0A
	B	APP_ALT5_IN	(Port_PinModeType)0x0B
	C	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
	A	APP_ALT6_OUT_SET_PIPC	(Port_PinModeType)0x8C
	B	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'.
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_WRITE_TIMEOUT_FAILURE
Related API(s)	Port_Init ,Port_SetPinDirection
Source of Error	When writing to a write-protected register fails.
Sl. 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.
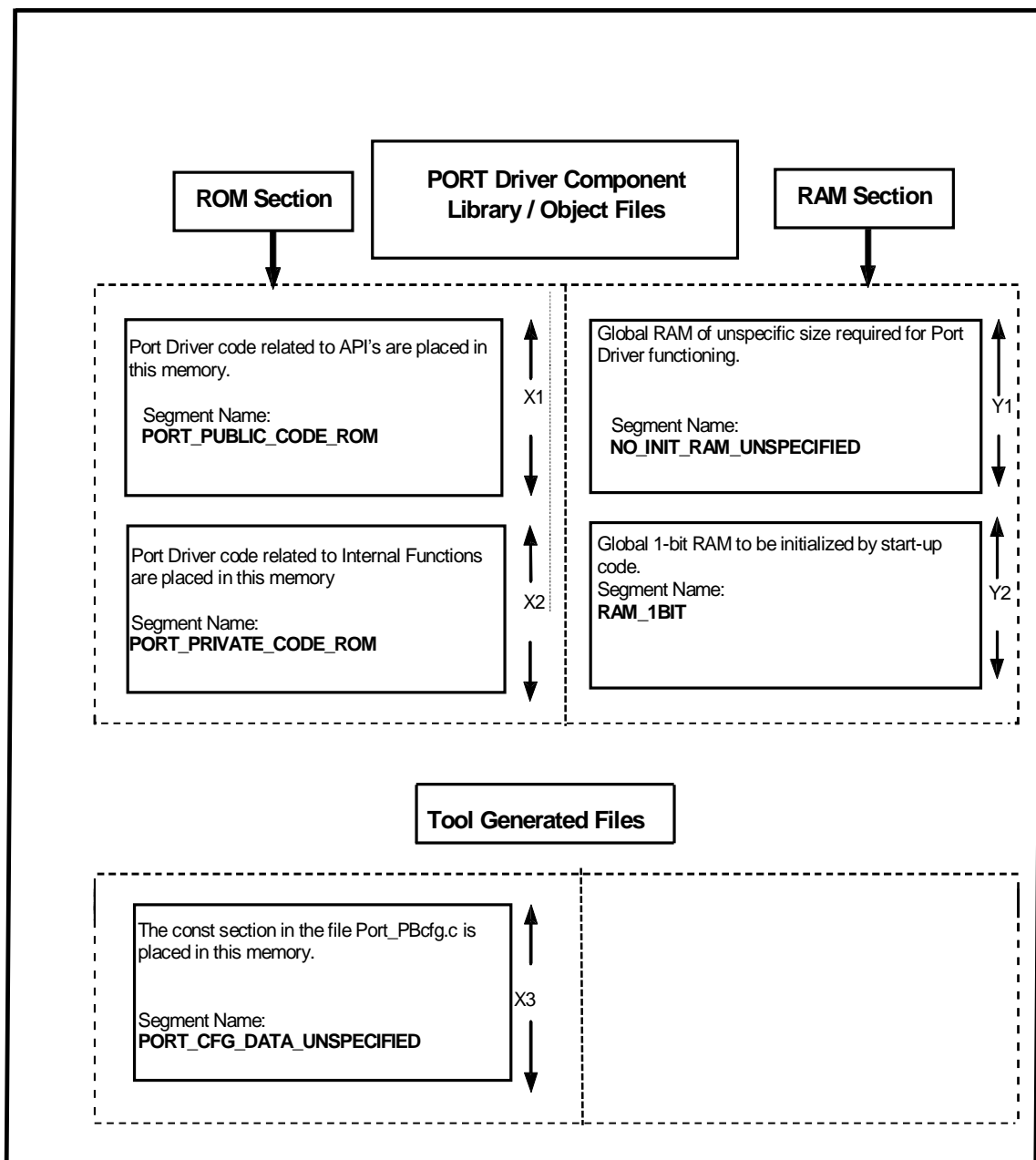
Figure 12-1   **PORT Driver Component Memory Organization**

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.
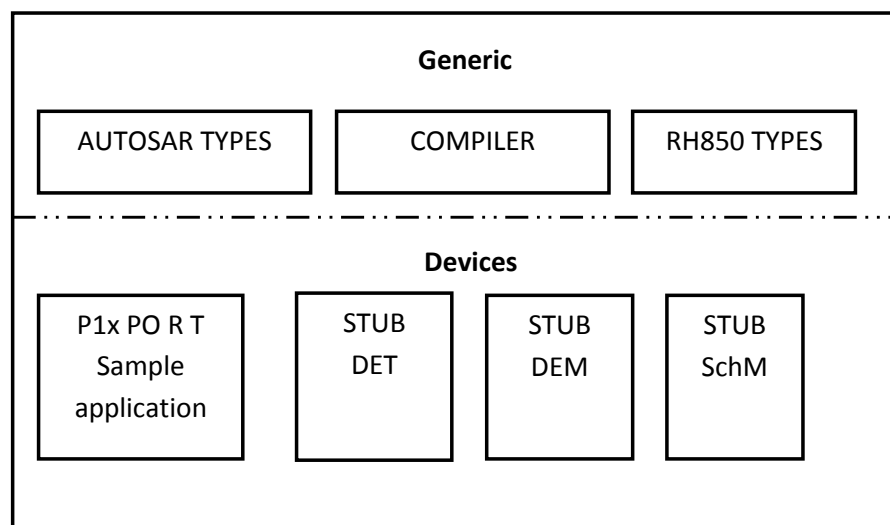

Figure 13-1   Overview of PORT Driver Sample Application

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>\
                                      \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_701314_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_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

Sl. 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

Sl. 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

Sl. 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	-

Chapter 14   Release Details

PORT Driver Software

Version: 1.5.3

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.	Following changes are made: 1. Chapter 2 is updated for referenced documents version. 2. Section 13.1.1 is updated for adding the device names. 3. Section 13.2 is updated for compiler, assembler and linker details. 4. Section 13.3 is updated to add parameter definition file and sample application configuration files for all P1M devices. 5. Chapter 14 is updated for PORT driver component version information.	1.0.2	31-Jan-2014
4.	Following changes are made: 1. Chapter 2 is updated for referenced documents version. 2. Section 13.1.1 is updated for adding the device names. 3. Section 13.2 is updated for compiler, assembler and linker details. 4. Section 13.3 is updated to add parameter definition file and sample application configuration files for all P1M devices. 5. Chapter 14 is updated for PORT driver component version information. 6. Deviation list is updated to add PORT_E_PARAM_POINTER error foe Port_GetVersioInfo API and AUTOSAR requirement. 7. Memory and Throughput details are updated. 8. Section 10.2.1 is updated to add new structure element.	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.	1.0.4	05-Sep-2014
6.	Following changes are made: 1. Section 1.1 Document Overview is updated. 2. Chapter 2 Reference documents are updated for version change. 3. Chapter 4 is updated for information regarding Interrupt vector table. 4. Chapter 6 Port_SetPinMode is updated. 5. Section 10.2.4 Port_PinModeType is updated. 6. Section 13.1.1 is updated for adding new devices. 7. Section 13.2 Compiler, Linker and Assembler section is removed. 8. Section 13.2 is updated for parameter definition file and sample application configuration files of all P1M devices. 9. Section 13.3 Memory and Throughput details are updated.	1.0.5	29-Apr-2015

SI.No.	Description	Version	Date
7	Following changes are made: 1. Chapter 2 Reference documents are updated for version change. 2. Definitions in Page 6 is updated for "PORT Idle State" 3. Section 4.1 is updated for information "The parameter PortDriveStrengthControl has dependency on parameter PortUnlimitedCurrentControl while specifying the output driving abilities of port pins" 4. 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. 5. Section 4.4 is updated to add Exclusive areas. 6. Chapter 6, Table 6-1, Register Details is updated for Apis Port_SetPinDirection, Port_SetPinMode, Port_InitConfig, Port_SetToDioOrAltMode, Port_SetPinDefaultDirection and Port_SetPinDefaultMode. 7. Chapter 8 is updated to add Stub C header files 8. Section 10.1.1 updated to add Std_ReturnType 9. Chapter 12 is updated to remove PORT_CFG_DBTOC_UNSPECIFIED in ROM Section and in Figure 12-1 10. Section 13.2.2 updated for correction in API names 11. Updated Chapter 13.3 'Memory and Throughput'. 12. Updated Chapter 14 'Release Details' to change the driver version. 13. Updated Chapter 3 and Chapter 9 to add R-Numbered Tool User manual name. 14. Added R number at end of the document. 15. Copyright information is updated	1.0.6	29-Mar-2016
8.	Following changes are made: 1. Chapter 1 Introduction is updated. 2. Information on unused pins and Port Pin level inversion is updated in Chapter 4.1 General. 3. The point regarding digital filter time delay has been removed and the information on Port_GetVersionInfo is updated in 4.5 Deviation list. 4. Chapter 5 Architecture Details is updated with the information on Port_GetVersionInfo. 5. Chapter 7 is updated with the information on PORT Driver features. 6. Chapter 8 is updated to add the information about Port_Cbk.h file. 7. Table 11-2 is updated to add the DEM error PORT_E_REG_WRITE_VERIFY. 8. Table 11-1 is updated to add the DET error PORT_E_PARAM_POINTER. 9. Chapter 14 Release Details is updated with Port Driver Software version.	1.0.7	06-Jun-2016

SI.No.	Description	Version	Date
	10. Section 4.3 is updated with note on critical section. 11. Chapter 8 PORT Driver Component Header and Source File Description is updated for adding stub files. 12. Chapter 6 Register details is updated. 13. Section 13.2.1 and section 13.2.2 are updated. 14. Chapter 3 and 9 are updated with R-Numbered Tool User manual name. 15. 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. 19. 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.	Following changes are made: 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. Critical section value updated in the Table 4-2. 5. Preconditions updated in section 4.2. 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. Deviation regarding the parameter PortPinlevelValue of JTag PortPin is added in the section 4.5.Deviation List	1.0.8	24-Oct-2016

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

RENESAS

Renesas Electronics Corporation

SALES OFFICES

http://www.renesas.com

AUTOSAR MCAL R4.0.3

User's Manual

RENESAS

Renesas Electronics Corporation