User's Manual

AUTOSAR MCAL R4.0.3

User's Manual

PORT Driver Component Ver.1.0.5

Embedded User's Manual

Target Device:
RH850\P1x

Renesas Electronics
www.renesas.com

Rev.0.02 Apr 2015

Notice

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

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

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

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

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority- owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Abbreviations and Acronyms

Abbreviation / Acronym	Description
ADC	Analog to Digital Converter
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
PLL	Phase Locked Loop
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
TAU	Timer Array Unit
WDT	Watchdog Timer

Definitions

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

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

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

This document shall be used as reference by the users of PORT Driver Component for P1M Device. The information specific to P1M Device channel mapping, ISR handler, 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

• De-initialization

• Reading the internal state of PORT Output signal

• Setting the PORT Output to Idle state

• Disabling/Enabling the PORT signal edge notification

• Synchronous start between the TAU units

The following diagram shows the system overview of the AUTOSAR Architecture. The PORT Driver initializes all the channels that are required for producing PORT outputs.

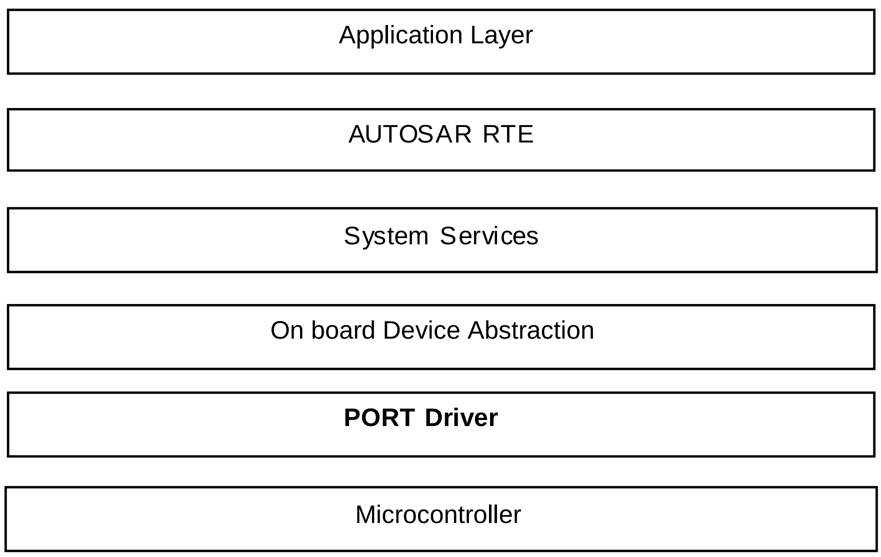

Figure 1-1 System Overview Of AUTOSAR Architecture

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

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

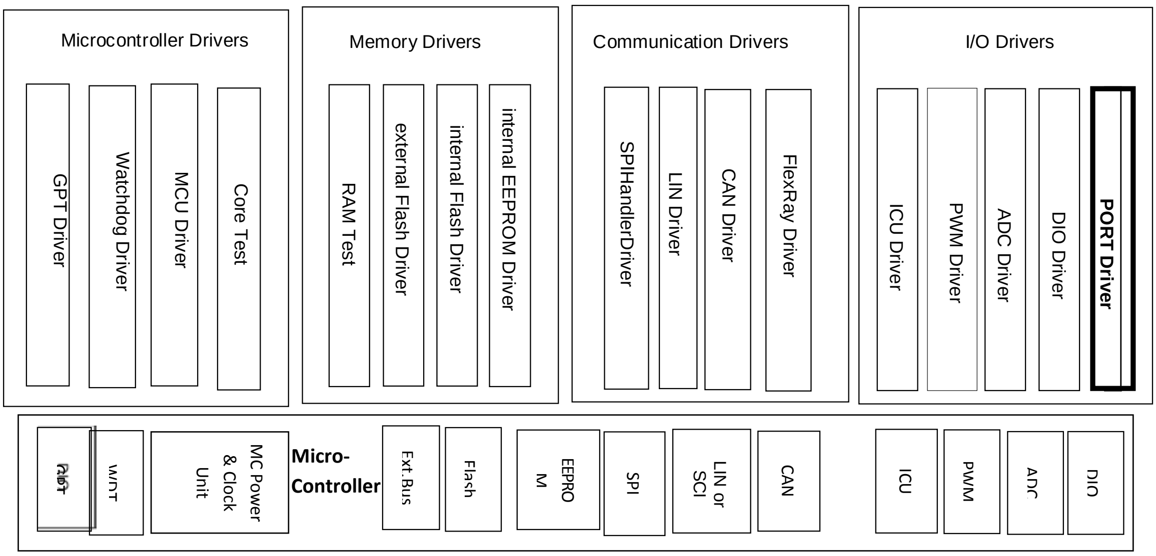

Figure 1-2 System Overview Of The PORT Driver In AUTOSAR MCAL Layer

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 is mentioned. This section also contains the brief note on the tool generated output file.
Section 9 (Generation Tool Guide)	This section provides information on the PORT Driver Component Code Generation Tool.
Section 10 (Application Programming Interface)	This section mentions all the APIs provided by the PORT Driver Component.
Section 11 (Development And Production Errors)	This section lists the DET and DEM errors.
Section 12 (Memory Organization)	This section provides the typical memory organization, which must be met for proper functioning of component.
Section 13 (P1M Specific Information)	This section describes the P1M specific information like channel mapping, the details of the P1M Sample Application and it's folder structure and the information about RAM/ROM usage, stack depth and throughput details.
Section 14 (Release Details)	This section provides release details with version name and base version.

Chapter 2 Reference Documents

Sl. No.	Title	Version
1.	AUTOSAR_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.	r01uh0436ej0070_rh850p1x.pdf	0.70
4.	AUTOSAR_SWS_CompilerAbstraction.pdf	2.2.0
5.	AUTOSAR_SWS_MemoryMapping.pdf	1.2.1
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 "AUTOSAR_PORT_Tool_UserManual.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

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.exe

X1X\P1x\common_family\generator
\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. <Complier> 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 in Port registers is undefined.
- 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.

4.2. Preconditions

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

- The Port_PBcfg.c and Port_Cfg.h files generated by the PORT Driver Component Code Generation Tool must be compiled and linked along with PORT Driver Component source files.
- The application has to be rebuilt, if there is any change in the Port_Cfg.h file generated by the PORT Driver Component Generation Tool.
- File Port_PBcfg.c generated for single configuration set or multiple configuration sets using PORT Driver Component Code Generation Tool should be compiled and linked independently.
- Symbolic names for all Port Pins are generated in Port_Cfg.h file which can be used as parameters for passing to PORT Driver Component APIs.
- The PORT Driver Component needs to be initialized for all Port Pins before doing any operation on Port Pins. The Port_Init () API shall also be called after a reset in order to reconfigure the Port Pins of the microcontroller. If PORT Driver Component is not initialized properly, the behavior of Port Pins may be undetermined.
- The user should ensure that PORT Driver Component API requests are invoked with correct input arguments.
- The other modules depending on PORT Driver Component should ensure that the PORT Driver Component initialization is successful before doing

any operation on Port Pins.

- Input parameters are validated only when the static configuration parameter PORT_DEV_ERROR_DETECT is enabled. Application should ensure that the right parameters are passed while invoking the APIs when PORT_DEV_ERROR_DETECT is disabled.
- Values for production code Event Id's should be assigned externally by the configuration of the DEM.
- A mismatch in the version numbers of header and the source files will result in a compilation error. User should ensure that the correct versions of the header and the source files are used.
- The PORT Driver Component APIs, except Port_GetVersionInfo API, which are intended to operate on Port Pins shall be called only after PORT Driver Component is initialized by invoking Port_Init() API. Otherwise Port Pin functions will exhibit undefined behavior.
- All Port Pins and their functions should be configured by the Port configuration tool. It is the User/Integrator responsibility to ensure that the same Port/Port Pin is not being accessed/configured in parallel by different entities in the same system.

4.3. User Mode and Supervisor Mode

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

Table 4-1 Supervisor mode and User mode details

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

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:

- SET_PIN_MODE_PROTECTION

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

4.5. Deviation List

Table 4-2 PORT Driver Deviation List

Sl. No.	Description	AUTOSAR Bugzilla
1.	The Port Pin specific containers (PortPin0, PortPin1, PortPin2 and so on …) are added as sub containers of PortGroup<n> containers, having the parameters ‘PortPinDirection’, ‘PortPinDirectionChangeable’, ‘PortPinLevelValue’ and ‘PortPinInitialMode’ are added. AUTOSAR specified container ‘PortPin’ and all its parameters are considered as unused.	-
2.	PortPinMode configuration parameter is not used for implementation as all possible modes of a pin can be used in the Port_SetPinMode function.	-
3.	The Port_GetVersionInfo API is implemented as macro without DET error PORT_E_PARAM_POINTER	-
4.	[ecuc_sws_2108] requirement is not applicable to port module since implementation of PORT module is vendor specific.	-
5.	Digital Noise Filter to calculate time delay for following scenarios is not implemented in PORT driver. a. When digital filter output signal is input for alternative function b. When an event output signal of the digital filter is used as an interrupt	-

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:

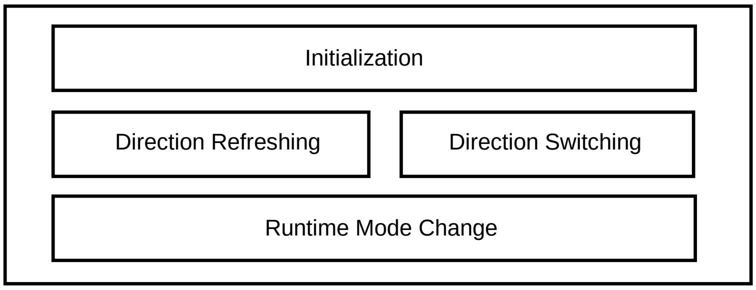

Figure 5-1 PORT Driver Architecture

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.

Port Initialization

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

Port Direction Refreshing

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

In this functionality only Direction of Port Pins is refreshed.

Port Pin Direction Switching

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

Port Pin Mode changing

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

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

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

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	Registers	Config Parameter	Macro/Variable
Port_Init	-	-	-	-
Port_SetPinDirection	32 bit	PSRn	PortPinLevelValue	usChangeableConfigV
	32 bit	PMSRn	-	usOrMask
	16 bit	PFCEn	-	usOrMask
	16 bit	PFCn	-	usOrMask
	16 bit	PFCAEn	-	usOrMask
	32 bit	PMCSRn	-	usOrMask
	16 bit	PINVn	PortOutputLevelInversion	usPortinversionVal
Port_RefreshPortDirection	-	-	-	-
Port_SetPinMode	32 bit	PSRn	PortPinLevelValue	usInitModeRegVal
	32 bit	PMSRn	-	usOrMask
	16 bit	PFCEn	-	usOrMask
	16 bit	PFCn	-	usOrMask
	16 bit	PFCAEn	-	usOrMask
	32 bit	PMCSRn	-	usOrMask
	16 bit	PIPCn	-	usOrMask
Port_SearchMode ChangeablePin	-	-	-	-
Port_InitConfig	32 bit	PSRn	PortPinLevelValue	usInitModeRegVal
	16 bit	PISn	PortInputSelection	usInitModeRegVal
	16 bit	PIBCn	PortInputBufferControl	usInitModeRegVal
	16 bit	PIPCn	PortIpControl	usInitModeRegVal
	16 bit	PUn	PullUpOption	usInitModeRegVal
	16 bit	PDn	PullDownOption	usInitModeRegVal
	16 bit	PBDCn	PortBiDirectionControl	usInitModeRegVal
	32 bit	PODCn	PortOpenDrainControl	usInitModeRegVal
	32 bit	PDSCn	PortDriveStrengthCont	usInitModeRegVal
	16 bit	PFCEn	PortPinInitialMode	usInitModeRegVal
	16 bit	PFCn	PortPinInitialMode	usInitModeRegVal
	16 bit	PFCAEn	PortPinInitialMode	usInitModeRegVal
	16 bit	PINVn	PortOutputLevelInversion	usInitModeRegVal
	16 bit	PODCEn	PortOpenDrainControl Expansion	usInitModeRegVal

API Name	Register Access 8/16/32 bits	Registers	Config Parameter	Macro/Variable
	16 bit	PUCCn	PortUnlimitedCurrentControl	usInitModeRegVal
	32 bit	PMSRn	PortPinDirection	usInitModeRegVal
	32 bit	PMCSRn	PortPinInitialMode	usInitModeRegVal
	8 bit	PPROTSn	-	PORT_ONE
	32 bit	PPCMDn	-	PORT_WRITE_ERRO R_CLEAR_VAL
	32 bit	PWSn	-	PORT_IOHOLD_SET
	32 bit	JPSRn	PortPinLevelValue	usInitModeRegVal
	8 bit	JPISn	PortInputSelection	usInitModeRegVal
	8 bit	JPIBCn	PortInputBufferControl	usInitModeRegVal
	16 bit	JPIPCn	PortIpControl	usInitModeRegVal
	8 bit	JPUn	PullUpOption	usInitModeRegVal
	8 bit	JPDn	PullDownOption	usInitModeRegVal
	8 bit	JPBDCn	PortBiDirectionControl	usInitModeRegVal
	32 bit	JPODCn	PortOpenDrainControl	usInitModeRegVal
	32 bit	JPDSCn	PortDriveStrengthControl	usInitModeRegVal
	8 bit	JPFCEn	PortPinInitialMode	usInitModeRegVal
	8 bit	JPFCn	PortPinInitialMode	usInitModeRegVal
	32 bit	JPMCSRn	PortPinInitialMode	usInitModeRegVal
	32 bit	JPMSRn	PortPinDirection	usInitModeRegVal
	32 bit	JPPROTS	-	PORT_ONE
	16 bit	JPODCEn	PortOpenDrainControl Expansion	usInitModeRegVal
	16 bit	JPUCCn	PortUnlimitedCurrentControl	usInitModeRegVal
Port_RefreshPortInternal	32 bit	PMSRn	-	ulMaskAndConfigValue
	32 bit	JPMSRn	-	ulMaskAndConfigValue
Port_GetVersionInfo	-	-	-	Port_GetVersionInfo
Port_SetToDioMode	32 bit	PMCSRn	-	usOrMask
Port_SetToAlternateMode	32 bit	PMCSRn	-	usOrMask
Port_SetToDioOrAltMode	-	-	-	-
Port_SearchDirChangeablePin	-	-	-	-
Port_FilterConfig	8 bit	DNFAnCTL	PortSameLevelSamples	ucDNFACTL
			PortDigitalFilterEnable	
			PortSamplingClockFreQuency	

API Name	Register Access 8/16/32 bits	Registers	Config Parameter	Macro/Variable
	8 bit	FCLAnCTL	PortAnalogFilterBypass PortEdgeOrLevelControl	ucFCLACTL
	16 bit	DNFAnEN	PortDigitalFilterEnable Input	usDNFAEN
	32 bit	DNFCKSnC	PortClockSource	ulDNFCKS
	8 bit	DNFP02nEDCm	PortDigitalFilterEnableInput PortEdgeDetectControl	ucDNFEDC

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 channels through channel configuration.
- To De-initialize the PORT channels.
- To set the PORT channel output to its configured idle state.
- To read the output state of a PORT channel.
- To read the version information of the PORT module.
- To support the diagnostic functionality for PORT channel.

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

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

The PORT Driver Component source files:

- Port.c
- Port_Ram.c
- Port_Version.c

The port specific C header files:

- Compiler.h
- Compiler_Cfg.h
- MemMap.h
- Platform_Types.h

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

Table 8-1 Description of the PORT Driver Component Files

File	Details
Port_Cfg.h	This file contains various PORT Driver Pre-compile time parameters, macro definitions for the ISRs, channel notifications used by PORT Driver, PORT channel handles.
Port_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.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.

Chapter 9 Generation Tool Guide

For more information on the PORT Driver Component Generation Tool, please refer “AUTOSAR_PORT_Tool_UserManual.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

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	pPortAlphaRegs	Pointer to the address of the Alphabetic port registers configuration.

Name:	Port_ConfigType		
Type:	struct		
	Port_FuncCtrlRegs	pPortAlphaFuncCtrlRegs	Pointer to the address of Alphabetic function control registers configuration.
	Port_PMSRRegs	pPortAlphaPMSRRegs	Pointer to the address of the Alphabetic PMSR registers configuration.
	Port_Regs	pPortJRegs	Pointer to the address of JTAG port registers configuration
	Port_FuncCtrlRegs	pPortJFuncCtrlRegs	Pointer to the address of JTAG function control registers configuration
	Port_PMSRRegs	pPortJPMSRRegs	Pointer to the address of JTAG PMSR registers configuration.
	Port_PinsDirChangeable	pPinDirChangeable	Pointer to the address of runtime direction changeable pins structure.
	Port_PinModeChangeableGroups	pPinModeChangeableGroups	Pointer to the address of runtime mode changeable pin group details structure.
	Port_PinDioAltChangeableDetails	pPinDioAltModeDetails	Pointer to the address of run time mode changeable pins structure.
	Port_PinModeChangeableDetails	pPinModeChangeableDetails	Pointer to the address of run time mode changeable pins structure.
	Port_DNFARegs	pPortDNFARegs	Pointer to the DNFA registers structure.
	Port_FCLARegs	pPortFCLARegs	Pointer to the FCLA registers structure.
	Port_EDCRegs	pPortEDCRegs	Pointer to the EDC registers structure

Name:	Port_ConfigType		
Type:	struct		
	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
	uint8	ucNoOfDNFARegs	The total number of DNFA noise elimination registers
	uint8	ucNoOfEDCRegs	The total number of EDC registers
	uint8	ucNoOfFCLARegs	The total number of FCLA noise elimination registers
	uint8	ucNoOfNumRestoredRegs	The total number of Numeric Restored registers
	uint8	ucNoOfAlphaRestoredRegs	The total number of Alphabetic Restored registers
	uint8	ucNoOfAnalogRestoredRegs	The total number of Analog Restored 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
	E	APP_ALT7_OUT	(Port_PinModeType)0x0E
	F	APP_ALT7_IN	(Port_PinModeType)0x0F
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
	C	APP_ALT7_OUT_SET_PIPC	(Port_PinModeType)0x8E

	D	APP_ALT7_IN_SET_PIPC	(Port_PinModeType)0x8F
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

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

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.

Chapter 12 Memory Organization

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

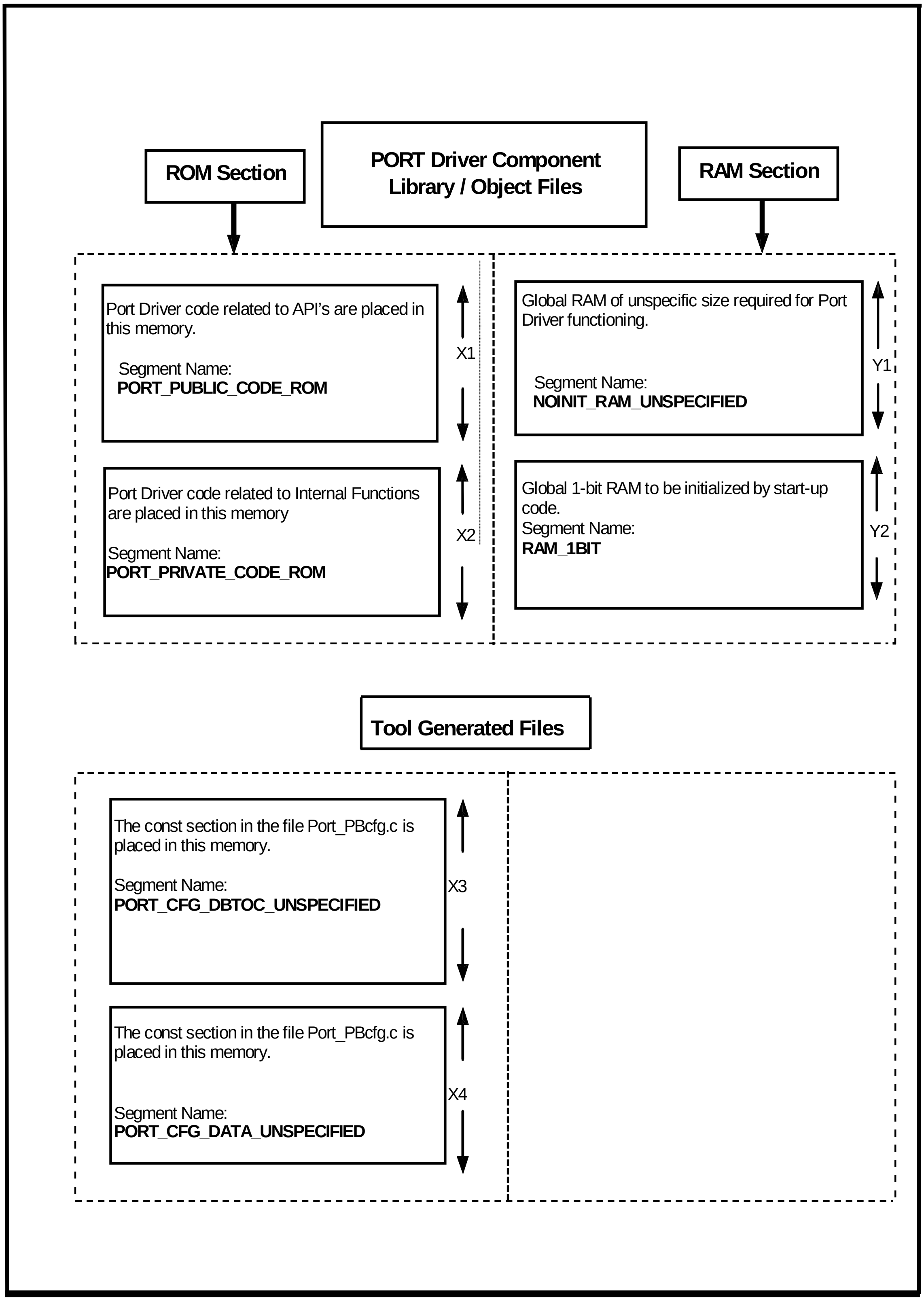

Figure 12-1 PORT Driver Component Memory Organization

<u>**ROM Section (X1, X2, X3 and X4):**</u>

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

PORT_CFG_DATA_UNSPECIFIED (X4): This section consists of PORT Driver Component constant configuration structures. This can be located in code memory.

<u>**RAM Section (Y1 and Y2):**</u>

NOINIT_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

13.1.1. 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

13.1.2. 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.

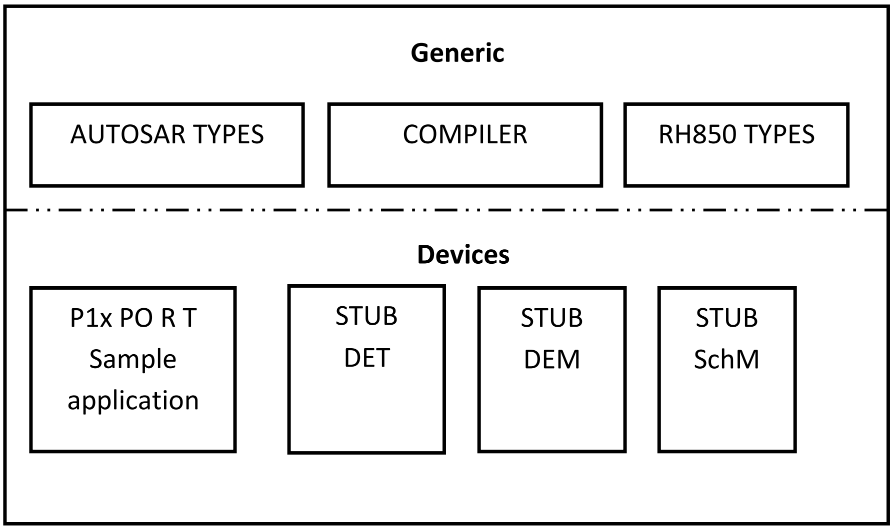

Figure 13-1 Overview of PORT Driver Sample Application

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

\config\App_PORT_P1M_701304_Sample.arxml
\config\App_PORT_P1M_701304_Sample.html
\config\App_PORT_P1M_701304_Sample.one

\config\App_PORT_P1M_701305_Sample.arxml
\config\App_PORT_P1M_701305_Sample.html
\config\App_PORT_P1M_701305_Sample.one

\config\App_PORT_P1M_701310_Sample.arxml
\config\App_PORT_P1M_701310_Sample.html
\config\App_PORT_P1M_701310_Sample.one

\config\App_PORT_P1M_701311_Sample.arxml
\config\App_PORT_P1M_701311_Sample.html
\config\App_PORT_P1M_701311_Sample.one

\config\App_PORT_P1M_701312_Sample.arxml
\config\App_PORT_P1M_701312_Sample.html
\config\App_PORT_P1M_701312_Sample.one

\config\App_PORT_P1M_701313_Sample.arxml
\config\App_PORT_P1M_701313_Sample.html
\config\App_PORT_P1M_701313_Sample.one

\config\App_PORT_P1M_701314_Sample.arxml
\config\App_PORT_P1M_701314_Sample.html
\config\App_PORT_P1M_701314_Sample.one

\config\App_PORT_P1M_701315_Sample.arxml
\config\App_PORT_P1M_701315_Sample.html
\config\App_PORT_P1M_701315_Sample.one

\config\App_PORT_P1M_701318_Sample.arxml
\config\App_PORT_P1M_701318_Sample.html
\config\App_PORT_P1M_701318_Sample.one

\config\App_PORT_P1M_701319_Sample.arxml
\config\App_PORT_P1M_701319_Sample.html
\config\App_PORT_P1M_701319_Sample.one

\config\App_PORT_P1M_701320_Sample.arxml
\config\App_PORT_P1M_701320_Sample.html
\config\App_PORT_P1M_701320_Sample.one

\config\App_PORT_P1M_701321_Sample.arxml
\config\App_PORT_P1M_701321_Sample.html
\config\App_PORT_P1M_701321_Sample.one

\config\App_PORT_P1M_701322_Sample.arxml
\config\App_PORT_P1M_701322_Sample.html
\config\App_PORT_P1M_701322_Sample.one

\config\App_PORT_P1M_701323_Sample.arxml
\config\App_PORT_P1M_701323_Sample.html
\config\App_PORT_P1M_701323_Sample.one

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 past parameter will get updated with the PORT Driver version details.

- 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_SetPinMode: This service sets the Port Pin mode during runtime.

- Port_SetToDioMode: This function used to set the mode of a port pin to DIO mode during runtime.

- Port_SearchModeChangeablePin: This function searches the given PIN Id in the existing list of PIN IDs which are mode changeable in run time through Binary Search algorithm.

- 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.

- The API Port_GetOutputState is invoked to get the channel output state and provides the service to read the internal state of a PORT output signal of a channel.

- Port_InitConfig: This function initializes all ports and port pins with the configuration set pointed by ConfigPtr.

- Port_FilterConfig: This Function used to initialize all the registers of filter configuration.

- Port_SearchDirChangeablePin: This function searches the given PIN Id in the existing list of PIN Id's which are direction changeable in run time through Binary Search algorithm.

- Port_RefreshPortInternal: 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.

13.2.2. Building Sample Application

13.3.2.1. 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.html

13.3.2.2. 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.html file available in the path:

  "X1X\P1x\modules\port\sample_application\<SubVariant>\<AUTOSAR_version>\config\App_PORT_P1M_701310_Sample.html"

- 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 Set Pin Mode: This API will change the pin mode to the requested mode.

- Port_SetToDioMode: This API will set the mode of a pin to DIO mode.

- Port_SetToAlternateMode: This API will set the mode of a port pin to Alternate mode.

- Port SetPinDirection: This API will change the direction of the pin to the requested direction.

- Port RefreshPortDirection: This API will refresh all the port pins to the configured value except the pins that are configured as pin direction changeable during runtime.

Note: The <Device_name> indicates the device to be compiled, which can be 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 13.3.2.1 *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	-	1278
		PORT_PRIVATE_CODE_ROM	-	2142
		PORT_CFG_DATA_UNSPECIFIED	-	548
		PORT_CFG_DBTOC_UNSPECIFIED	-	48
2	RAM	RAM_1BIT	-	0
		NOINIT_RAM_UNSPECIFIED	-	4

The details of memory usage for the typical configuration, with DET enabled and all other configurations as provided in 13.3.2.1 *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	-	1828
		PORT_PRIVATE_CODE_ROM	-	2166
		PORT_CFG_DATA_UNSPECIFIED	-	532
		PORT_CFG_DBTOC_UNSPECIFIED	-	48
2	RAM	RAM_1BIT	-	1
		NOINIT_RAM_UNSPECIFIED	-	4

13.3.2. Stack Depth

The worst-case stack depth for PORT Driver Component for the typical configuration provided in Section 13.3.2.1 is 92 bytes.

13.3.3. Throughput Details

The throughput details of the APIs for the configuration mentioned in the Section 13.3.2.1 *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	Throughput in microseconds for 701312	Remarks
1	Port_Init	45.9	-	-
2	Port_RefreshPortDirection	2.52	-	-
3	Port_SetPindirection	3.24	-	-
4	Port_GetVersionInfo	0.45	-	-
5	Port_SetPinMode	3.69	-	-
6	Port_SetToDioMode	1.98	-	-
7	Port_SetToAlternateMode	1.62	-	-
8	Port_SetPinDefaultDirection	1.8	-	-
9	Port_SetPinDefaultMode	3.15	-	-

Chapter 14 Release Details

PORT Driver Software

Version: 1.5.0

Revision History

Sl.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

AUTOSAR MCAL R4.0.3 User's Manual
PORT Driver Component Ver.1.0.5
Embedded User’s Manual

Publication Date: Rev.0.02, April 29, 2015

Published by: Renesas Electronics Corporation

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

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

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

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

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

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

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

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

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

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

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

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

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

AUTOSAR MCAL R4.0.3

User's Manual

Renesas Electronics Corporation