Document Title	Specification of Port Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	40

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R22-11

Document Change History

Date	Release	Changed by	Description
2022-11-24	R22-11	AUTOSAR Release Management	• Proper implementation of TPS_STDT_00042
2021-11-25	R21-11	AUTOSAR Release Management	• Removed [SWS_Port_00072] and [SWS_Port_00073] and corresponding notes
2020-11-30	R20-11	AUTOSAR Release Management	• Added Enum values for `Port_PinDirectionType` • Editorial changes
2019-11-28	R19-11	AUTOSAR Release Management	• MCAL Multicore Distribution • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• MCAL Multicore Distribution (Draft)
2017-12-08	4.3.1	AUTOSAR Release Management	• minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
2016-11-30	4.3.0	AUTOSAR Release Management	• Removed remaining references to DEM • Removed section "Variants"

2015-07-31	4.2.2	AUTOSAR Release Management	• Rephrased [SWS_Port_00077], [SWS_Port_00087], [SWS_Port_00087], [SWS_Port_00223] • Editorial changes on Chapter 7 • Remove [SWS_Port_0105] • Replace PORT_E_PARAM_CONFIG by PORT_E_INIT_FAILED
2014-10-31	4.2.1	AUTOSAR Release Management	• Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• PORT_E_UNINIT removed from Port_GetVersionInfo API • Header file structure: MemMap.h changed to Port_MemMap.h and EcuM.h removed • Changed scope of configuration parameters from "module" to "local" • Removal of specification items covered by the new SWS BSW General • Formal rework
2011-12-22	4.0.3	AUTOSAR Administration	• Removed Port132 and updated Figure 1 • Rephrased [SWS_Port_00114] and [SWS_Port_00075] • Removed Port210 • Added Chapter 12
2010-09-30	3.1.5	AUTOSAR Administration	• A lot of requirements are changed to make them atomic or /and more clear. No technical content changes. Look Chapter 14 for the complete list. • Debugging Concept Inserted. • Insertion of the new configuration parameter to enable or disable the mode of the port pins at run time. • Checking of Port_GetVersionInfo. • Legal disclaimer revised

2009-02-04	3.1.2	AUTOSAR CM	• Updated document to new SWS macros and updated links to generated content
2009-07-24	3.1.3	AUTOSAR Administration	• Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Table formatting corrected
2008-02-01	3.0.2	AUTOSAR Administration	• Update to Chapter 10 configuration. • Inclusion of Port Container • Inclusion of new SRS general requirements • Removal of redundant function: Dem_ReportErrorEvent() • Development errors and error codes added • Rewording of requirements (as part of the SWS Improvements) • Renaming of configuration parameter (PORT_PIN_DIRECTION_CHANGES_ALLOWED -> PORT_SEP_PIN_DIRECTION_API) • Technical Office Improvements: wording improvements, alignment of API description. • Document meta information extended • Small layout adaptations made

2007-01-24	2.1.15	AUTOSAR Administration	• New API introduced: Port_SetPinMode • New Module type definition: Port_PinModeType • Updated to section 5.1.2: Inclusion of new file structure information • Inclusion of new pre-processor switch PortSetPinModeApi • New configurable parameter introduced: PortPinInitialMode • Rewording of requirement [SWS_Port_00105]. • Removal of redundant requirements PORT119 and PORT026 in requirements matrix. • Legal disclaimer revised • Release Notes added • "Advice for users" revised • "Revision Information" added
2006-05-16	2.0	AUTOSAR Administration	• Document structure adapted to common Release 2.0 SWS Template. • Major changes in chapter 10 • Structure of document changed partly • Other changes see chapter 11
2005-05-31	1.0	AUTOSAR Administration	• Initial Release

Contents

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module PORT Driver.

This driver specification is applicable for on-chip ports and port pins.

This module shall provide the service for initializing the whole PORT structure of the microcontroller. Many ports and port pins can be assigned to various functionalities, e.g.

- General purpose I/O
- ADC
- SPI
- SCI
- PWM
- CAN
- LIN
- etc

For this reason, there shall be an overall configuration and initialization of this port structure. The configuration and mode of these port pins is microcontroller and ECU dependent.

Port initialisation data shall be written to each port as efficiently as possible.

This PORT driver module shall complete the overall configuration and initialisation of the port structure which is used in the DIO driver module. Therefore, the DIO driver works on pins and ports which are configured by the PORT driver.

The PORT driver shall be initialised prior to use of the DIO functions. Otherwise DIO functions will exhibit undefined behaviour.

The diagram below identifies the PORT driver functions, and the structure of the PORT driver and DIO driver within the MCAL software layer [1].

Driver:	**Name for a Port Pin:**	**Name for Subset of Adjacent pins on one port:**	**Name for a whole port:**
DIO Driver	Channel	Channel Group	Port
PORT Driver	Port pin	–	Port

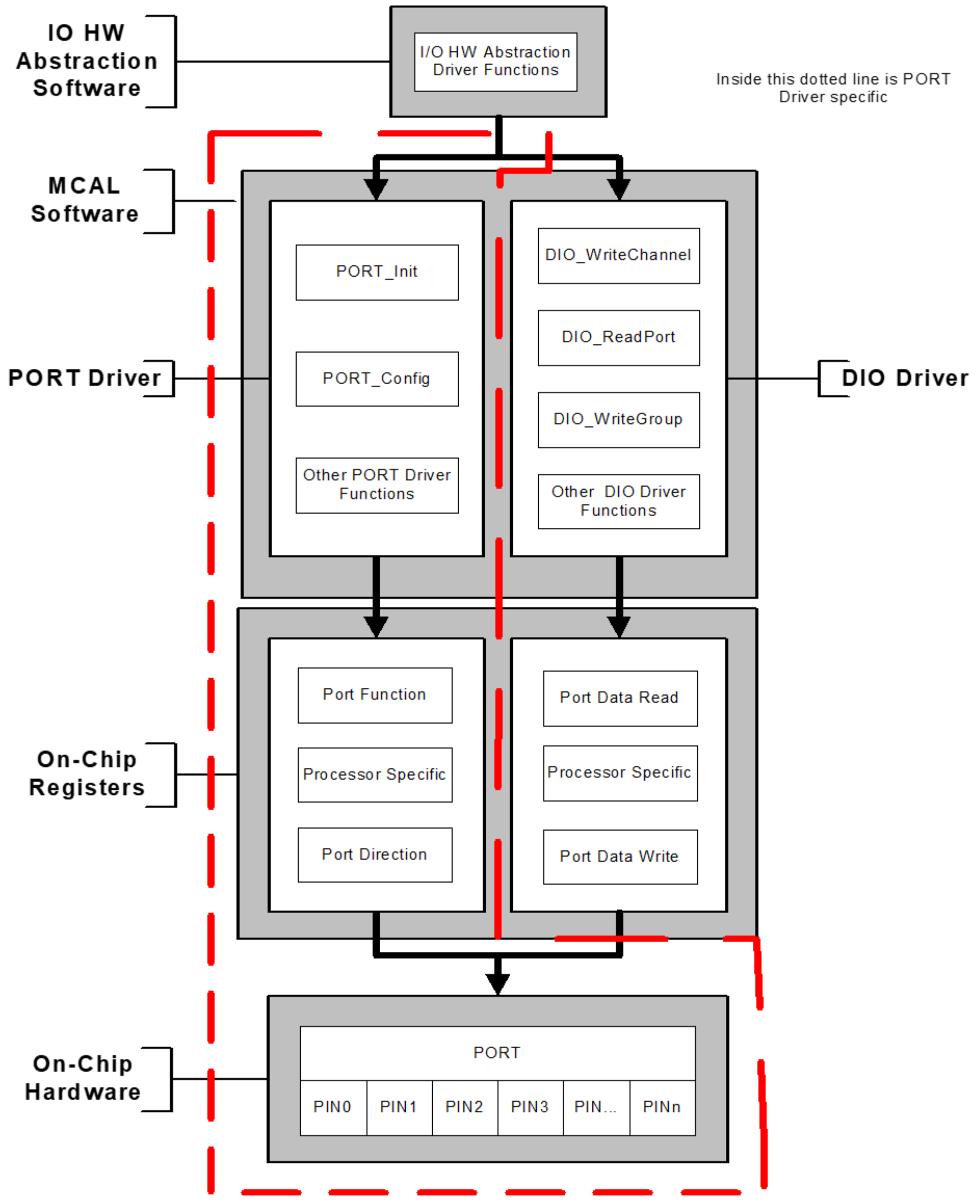
IO HW Abstraction Software
I/O HW Abstraction Driver Functions
Inside this dotted line is PORT Driver specific
MCAL Software
PORT Driver
PORT_Init
PORT_Config
Other PORT Driver Functions
DIO_WriteChannel
DIO_ReadPort
DIO_WriteGroup
Other DIO Driver Functions
DIO Driver
On-Chip Registers
Port Function
Processor Specific
Port Direction
Port Data Read
Processor Specific
Port Data Write
On-Chip Hardware
PORT
PIN0
PIN1
PIN2
PIN3
PIN...
PINn

Figure 1.1

2 Acronyms and abbreviations

The following table summarizes the expressions used within the PORT driver.

Abbreviation / Acronym:	Description:
DEM	Diagnostic Event Manager [2]
DET	Default Error Tracer [3]
MCU	MicroController Unit
Port Pin	Represents a single configurable input or output pin on an MCU device.
Port	Represents a whole configurable port on an MCU device.
Physical Level (Input)	Two states are possible: LOW/HIGH
Physical Level (Output)	Two states are possible: LOW/HIGH

Table 2.1: Acronyms and abbreviations

3 Related documentation

3.1 Input documents

[1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture

[2] Specification of Diagnostic Event Manager
AUTOSAR_SWS_DiagnosticEventManager

[3] Specification of Default Error Tracer
AUTOSAR_SWS_DefaultErrorTracer

[4] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral

[5] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral

[6] General Requirements on SPAL
AUTOSAR_SRS_SPALGeneral

[7] Requirements on Port Driver
AUTOSAR_SRS_PortDriver

[8] Specification of MCU Driver
AUTOSAR_SWS_MCUDriver

3.2 Related standards and norms

1. EC 7498-1 The Basic Model, IEC Norm, 1994

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [4, SWS BSW General], which is also valid for Port Driver.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Port Driver.

4 Constraints and assumptions

4.1 Limitations

Limitations for the PORT driver are specified as followed:

- It is the user's responsibility to ensure that the same Port/Port pin is not being accessed in parallel by different entities in the same system, e.g. by two tasks configuring the same port or two tasks configuring the same pin, or two tasks configuring different pins on the same port.

4.2 Applicability to car domains

No restrictions

5 Dependencies to other modules

Other driver modules may be dependent on the PORT driver depending on the available functionality of individual port pins on an MCU. For example, an MCU pin may be configurable as a DIO or SPI pin. Therefore, the DIO and/or the SPI driver modules may be dependent on the PORT module to configure the pin for the desired functionality.

5.1 File structure

5.1.1 Code file structure

For details refer to the chapter 5.1.6 "Code file structure" in [4].

6 Requirements traceability

This chapter refers to the input requirements specified in the SRS documents (Software Requirements Specifications [5], [6], [7]) that are applicable for this software module.

The table below lists the specification items of the PORT driver SWS document that satisfy the input requirements. Only functional requirements are referenced.

Requirement	Description	Satisfied by
[SRS_BSW_00005]	Modules of the μC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	[SWS_Port_NA_00227]
[SRS_BSW_00006]	The source code of software modules above the μC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	[SWS_Port_NA_00227]
[SRS_BSW_00007]	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	[SWS_Port_NA_00227]
[SRS_BSW_00010]	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	[SWS_Port_NA_00227]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_Port_00001] [SWS_Port_00002] [SWS_Port_00041] [SWS_Port_00042]
[SRS_BSW_00159]	All modules of the AUTOSAR Basic Software shall support a tool based configuration	[SWS_Port_00004]
[SRS_BSW_00160]	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	[SWS_Port_NA_00227]
[SRS_BSW_00161]	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	[SWS_Port_NA_00227]
[SRS_BSW_00162]	The AUTOSAR Basic Software shall provide a hardware abstraction layer	[SWS_Port_NA_00227]
[SRS_BSW_00164]	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	[SWS_Port_NA_00227]
[SRS_BSW_00167]	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	[SWS_Port_NA_00227]
[SRS_BSW_00168]	SW components shall be tested by a function defined in a common API in the Basis-SW	[SWS_Port_NA_00227]
[SRS_BSW_00170]	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	[SWS_Port_NA_00227]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00172]	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	[SWS_Port_NA_00227]
[SRS_BSW_00307]	Global variables naming convention	[SWS_Port_NA_00227]
[SRS_BSW_00308]	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	[SWS_Port_NA_00227]
[SRS_BSW_00309]	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	[SWS_Port_NA_00227]
[SRS_BSW_00321]	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	[SWS_Port_NA_00227]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_Port_00087]
[SRS_BSW_00325]	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	[SWS_Port_NA_00227]
[SRS_BSW_00327]	Error values naming convention	[SWS_Port_00051]
[SRS_BSW_00328]	All AUTOSAR Basic Software Modules shall avoid the duplication of code	[SWS_Port_NA_00227]
[SRS_BSW_00330]	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	[SWS_Port_NA_00227]
[SRS_BSW_00331]	All Basic Software Modules shall strictly separate error and status information	[SWS_Port_NA_00227]
[SRS_BSW_00333]	For each callback function it shall be specified if it is called from interrupt context or not	[SWS_Port_NA_00227]
[SRS_BSW_00334]	All Basic Software Modules shall provide an XML file that contains the meta data	[SWS_Port_NA_00227]
[SRS_BSW_00335]	Status values naming convention	[SWS_Port_NA_00227]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_Port_NA_00227]
[SRS_BSW_00337]	Classification of development errors	[SWS_Port_00051]
[SRS_BSW_00341]	Module documentation shall contains all needed informations	[SWS_Port_NA_00227]
[SRS_BSW_00342]	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	[SWS_Port_NA_00227]
[SRS_BSW_00343]	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	[SWS_Port_NA_00227]
[SRS_BSW_00344]	BSW Modules shall support link-time configuration	[SWS_Port_NA_00227]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00347]	A Naming seperation of different instances of BSW drivers shall be in place	[SWS_Port_NA_00227]
[SRS_BSW_00357]	For success/failure of an API call a standard return type shall be defined	[SWS_Port_NA_00227]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_Port_00140]
[SRS_BSW_00359]	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	[SWS_Port_NA_00227]
[SRS_BSW_00360]	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	[SWS_Port_NA_00227]
[SRS_BSW_00371]	No description	[SWS_Port_NA_00227]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[SWS_Port_NA_00227]
[SRS_BSW_00375]	Basic Software Modules shall report wake-up reasons	[SWS_Port_NA_00227]
[SRS_BSW_00377]	A Basic Software Module can return a module specific types	[SWS_Port_NA_00227]
[SRS_BSW_00378]	AUTOSAR shall provide a boolean type	[SWS_Port_NA_00227]
[SRS_BSW_00385]	List possible error notifications	[SWS_Port_00051]
[SRS_BSW_00395]	The Basic Software Module specifications shall list all configuration parameter dependencies	[SWS_Port_NA_00227]
[SRS_BSW_00398]	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	[SWS_Port_NA_00227]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_Port_00041]
[SRS_BSW_00406]	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	[SWS_Port_00051] [SWS_Port_00087]
[SRS_BSW_00413]	An index-based accessing of the instances of BSW modules shall be done	[SWS_Port_NA_00227]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_Port_00121]
[SRS_BSW_00416]	The sequence of modules to be initialized shall be configurable	[SWS_Port_NA_00227]
[SRS_BSW_00417]	Software which is not part of the SW-C shall report error events only after the Dem is fully operational.	[SWS_Port_NA_00227]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00419]	If a pre-compile time configuration parameter is implemented as `const` it should be placed into a separate c-file	[SWS_Port_NA_00227]
[SRS_BSW_00423]	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	[SWS_Port_NA_00227]
[SRS_BSW_00424]	BSW module main processing functions shall not be allowed to enter a wait state	[SWS_Port_NA_00227]
[SRS_BSW_00425]	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	[SWS_Port_NA_00227]
[SRS_BSW_00426]	BSW Modules shall ensure data consistency of data which is shared between BSW modules	[SWS_Port_NA_00227]
[SRS_BSW_00427]	ISR functions shall be defined and documented in the BSW module description template	[SWS_Port_NA_00227]
[SRS_BSW_00428]	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	[SWS_Port_NA_00227]
[SRS_BSW_00429]	Access to OS is restricted	[SWS_Port_NA_00227]
[SRS_BSW_00432]	Modules should have separate main processing functions for read/receive and write/transmit data path	[SWS_Port_NA_00227]
[SRS_BSW_00433]	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	[SWS_Port_NA_00227]
[SRS_BSW_00437]	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	[SWS_Port_NA_00227]
[SRS_BSW_00439]	Enable BSW modules to handle interrupts	[SWS_Port_NA_00227]
[SRS_BSW_00440]	The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via `Rte_Call` API	[SWS_Port_NA_00227]
[SRS_Port_12001]	The Port driver shall allow the static configuration of the following options for each port	[SWS_Port_00004] [SWS_Port_00079]
[SRS_Port_12300]	Ports and port pins that are not used shall be set to a defined state	[SWS_Port_00005]
[SRS_Port_12302]	The port driver shall allow the static configuration of the port pin names	[SWS_Port_00006]
[SRS_Port_12405]	The Port driver shall provide a service for setting the direction of port pins during runtime	[SWS_Port_00063] [SWS_Port_00086] [SWS_Port_00138]
[SRS_Port_12406]	The Port driver shall provide a service to refresh the direction of all configured ports	[SWS_Port_00060] [SWS_Port_00061]

▽

△

Requirement	Description	Satisfied by
[SRS_SPAL_00157]	All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers	[SWS_Port_NA_00227]
[SRS_SPAL_12056]	All driver modules shall allow the static configuration of notification mechanism	[SWS_Port_NA_00227]
[SRS_SPAL_12057]	All driver modules shall implement an interface for initialization	[SWS_Port_00041] [SWS_Port_00042] [SWS_Port_00043]
[SRS_SPAL_12063]	All driver modules shall only support raw value mode	[SWS_Port_NA_00227]
[SRS_SPAL_12064]	All driver modules shall raise an error if the change of the operation mode leads to degradation of running operations	[SWS_Port_NA_00227]
[SRS_SPAL_12067]	All driver modules shall set their wake-up conditions depending on the selected operation mode	[SWS_Port_NA_00227]
[SRS_SPAL_12068]	The modules of the MCAL shall be initialized in a defined sequence	[SWS_Port_NA_00227]
[SRS_SPAL_12069]	All drivers of the SPAL that wake up from a wake-up interrupt shall report the wake-up reason	[SWS_Port_NA_00227]
[SRS_SPAL_12075]	All drivers with random streaming capabilities shall use application buffers	[SWS_Port_NA_00227]
[SRS_SPAL_12077]	All drivers shall provide a non blocking implementation	[SWS_Port_NA_00227]
[SRS_SPAL_12078]	The drivers shall be coded in a way that is most efficient in terms of memory and runtime resources	[SWS_Port_NA_00227]
[SRS_SPAL_12092]	The driver's API shall be accessed by its handler or manager	[SWS_Port_NA_00227]
[SRS_SPAL_12125]	All driver modules shall only initialize the configured resources	[SWS_Port_00041] [SWS_Port_00042]
[SRS_SPAL_12129]	The ISRs shall be responsible for resetting the interrupt flags and calling the according notification function	[SWS_Port_NA_00227]
[SRS_SPAL_12163]	All driver modules shall implement an interface for de-initialization	[SWS_Port_00003]
[SRS_SPAL_12169]	All driver modules that provide different operation modes shall provide a service for mode selection	[SWS_Port_NA_00227]
[SRS_SPAL_12263]	The implementation of all driver modules shall allow the configuration of specific module parameter types at link time	[SWS_Port_00041]
[SRS_SPAL_12265]	Configuration data shall be kept constant	[SWS_Port_NA_00227]
[SRS_SPAL_12267]	Wakeup sources shall be initialized by MCAL drivers and/or the MCU driver	[SWS_Port_NA_00227]
[SRS_SPAL_12448]	All driver modules shall have a specific behavior after a development error detection	[SWS_Port_00077]

▽

△

Requirement	Description	Satisfied by
[SRS_SPAL_12461]	Specific rules regarding initialization of controller registers shall apply to all driver implementations	[SWS_Port_00113] [SWS_Port_00214] [SWS_Port_00215] [SWS_Port_00217] [SWS_Port_00218]
[SRS_SPAL_12462]	The register initialization settings shall be published	[SWS_Port_NA_00227]
[SRS_SPAL_12463]	The register initialization settings shall be combined and forwarded	[SWS_Port_NA_00227]

Table 6.1: RequirementsTracing

7 Functional specification

7.1 General Behaviour

7.1.1 Background & Rationale

[SWS_Port_00001] ⌈The PORT Driver module shall initialize the whole port structure of the microcontroller.⌋*(SRS_BSW_00101)*

Note: Defining the order in which the ports and port pins are configured is the task of the configuration tool.

7.1.2 Requirements

7.1.2.1 Configuration of Port Pin Properties

[SWS_Port_00004] ⌈The PORT Driver module shall allow the configuration of different functionality for each port and port pin, e.g. ADC, SPI, DIO etc.⌋*(SRS_BSW_00159, SRS_Port_12001)*

The configuration of the port (i.e. whole port or single port pin) is microcontroller dependent.

[SWS_Port_00079] ⌈The PORT Driver module shall provide additional configurations for the MCU port/port pins:

- Pin direction (input/output)
- Pin level initial value
- Pin direction changeable during runtime (yes/no).
- Port mode changeable during runtime.

⌋*(SRS_Port_12001)*

[SWS_Port_00081] ⌈The PORT Driver module shall provide a number of optional configurations for the MCU ports and port pins (if supported by hardware):

- Slew rate control
- Activation of internal pull-ups
- Input Thresholds
- Pin driven mode (push-pull / open drain).
- Type of Readback support (pin level, output register value).

⌋*()*

[SWS_Port_00082] ⌈The PORT Driver module shall not provide the facility to configure pin level inversion. The default value shall be set (i.e. not inverted).⌋*()*

Note: The IO Hardware Abstraction layer shall carry out level inversion.

7.1.2.2 Switch port pin direction

[SWS_Port_00137] ⌈For the port pins configured as changeable using the configuration tool, the PORT driver shall allow the user to change the direction of port pins during runtime.⌋*()*

[SWS_Port_00138] ⌈If the MCU port control hardware provides an output latch for setting the output level on a port pin, switching the port pin direction shall not alter the level set in this output latch.⌋*(SRS_Port_12405)*

7.1.2.3 Refresh port direction

[SWS_Port_00066] ⌈For refreshing of the port on the microcontroller, the PORT driver shall allow the user to refresh the direction of those port pins whose direction is set by configuration and cannot be changed dynamically.⌋*()*

7.1.2.4 Configuration of unused Ports and Port Pins

[SWS_Port_00005] ⌈The PORT Driver module shall configure all ports and port pins that are not used (neither as GPIO nor special purpose IO) to be set to a defined state by the PORT Driver module configuration.⌋*(SRS_Port_12300)*

7.1.2.5 Configuration of symbolic names

[SWS_Port_00006] ⌈The user of the PORT Driver module shall configure the symbolic names of the port pins of the MCU.⌋*(SRS_Port_12302)*

[SWS_Port_00207] ⌈These symbolic names for the individual port pins (e.g. PORT_A_PIN_0) shall be defined in the configuration tool.⌋*()*

[SWS_Port_00208] ⌈The PORT Driver module's implementer shall publish the symbolic names through the file Port.h⌋*()*

7.1.2.6 Atomicity of port access

[SWS_Port_00075] ⌈The PORT Driver module shall provide atomic access to all ports and port pins.⌋*()*

Note:

An atomic access is a non interruptible access to Microcontroller registers by the use of either atomic instructions or the usage of an exclusive area (interrupt disabling for example) provided by the basic software scheduler module.

7.1.3 Version Check

7.1.3.1 Background and Rationale

The integration of incompatible files shall be avoided. Minimum implementation is the version check of the header file inside the .c file (version numbers of .c and .h files shall be identical).

7.1.3.2 Requirements

The Port module shall avoid the integration of incompatible files by the following pre-processor checks:

For details refer to the chapter 5.1.8 "Version Check" in [4, SWS BSW General].

7.2 Error classification

[SWS_Port_00051] ⌈

Type of error	*Related error code*	*Error value*
Invalid Port Pin ID requested	PORT_E_PARAM_PIN	0x0A
Port Pin not configured as changeable	PORT_E_DIRECTION_UNCHANGEABLE	0x0B
API Port_Init service called with wrong parameter	PORT_E_INIT_FAILED	0x0C
API Port_SetPinMode service called when mode is unchangeable.	PORT_E_PARAM_INVALID_MODE	0x0D
API Port_SetPinMode service called when mode is unchangeable.	PORT_E_MODE_UNCHANGEABLE	0x0E
API service called without module initialization	PORT_E_UNINIT	0x0F
APIs called with a Null Pointer	PORT_E_PARAM_POINTER	0x10

⌋*(SRS_BSW_00327, SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00406)*

7.2.1 Runtime Errors

There are no runtime errors.

7.2.2 Transient Faults

There are no transcient faults.

7.2.3 Production Errors

There are no production errors.

7.2.4 Extended Production Errors

There are no extended production errors.

7.3 API Parameter checking

[SWS_Port_00077] ⌈If development error detection is enabled the Port Driver module shall check the function parameters in the order in which they are passed and skip further parameter checking if one check fails.

Example: For the function `Port_SetPinDirection`, the first parameter to be passed is the pin ID. This parameter shall identify the relevant port pin of the MCU's port. The second parameter passed corresponds to the direction to change on the port pin.⌋ *(SRS_SPAL_12448)*

[SWS_Port_00087] ⌈If development error detection is enabled and the Port Driver module has detected an error, the desired functionality shall be skipped and the requested service shall return without any action.⌋*(SRS_BSW_00323, SRS_BSW_-00406)*

See table below for a list of the Det errors reported by each function.

Function:	Error Condition:	Realted error value:
`Port_-SetPinDirection`	Incorrect Port Pin ID passed	`PORT_E_PARAM_PIN`
	Port Pin not configured as changeable	`PORT_E_DIRECTION_UNCHANGEABLE`
`Port_Init`	Port_Init service called with wrong parameter.	`PORT_E_INIT_FAILED`
`Port_SetPinMode`	Incorrect Port Pin ID passed	`PORT_E_PARAM_PIN`
	Port Pin Mode passed not valid	`PORT_E_PARAM_INVALID_MODE`
	Port_SetPinMode service called when the mode is unchangeable	`PORT_E_MODE_UNCHANGEABLE`
`Port_-SetPinDirection`, `Port_SetPinMode` `Port_RefreshPort-Direction`	API service called prior to module initialization	`PORT_E_UNINIT`
`Port_-GetVersionInfo`	Api called with a NULL Pointer Parameter	`PORT_E_PARAM_POINTER`

8 API specification

8.1 Imported types

In this chapter, all types included from the following modules are listed:

[SWS_Port_00129] ⌈

Module	***Header File***	***Imported Type***
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋*()*

8.2 Type definitions

8.2.1 Port_ConfigType

[SWS_Port_00228] ⌈

Name	Port_ConfigType	
Kind	Structure	
Elements	Hardware Dependent Structure	
	Type	–
	Comment	The contents of the initialization data structure are specific to the microcontroller.
Description	Type of the external data structure containing the initialization data for this module.	
Available via	Port.h	

⌋*()*

8.2.2 Port_PinType

[SWS_Port_00229] ⌈

Name	Port_PinType		
Kind	Type		
Derived from	uint		
Range	0 - <number of port pins:>	–	Shall cover all available port pins. The type should be chosen for the specific MCU platform (best performance).
Description	Data type for the symbolic name of a port pin.		
Available via	Port.h		

⌋*()*

[SWS_Port_00013] ⌈The type Port_PinType shall be used for the symbolic name of a Port Pin.⌋*()*

[SWS_Port_00219] ⌈The type Port_PinType shall be uint8, uint16 or uint32 based on the specific MCU platform.⌋*()*

Note: The user shall use the symbolic names provided by the configuration tool.

8.2.3 Port_PinDirectionType

[SWS_Port_00230] ⌈

Name	Port_PinDirectionType		
Kind	Enumeration		
Range	PORT_PIN_IN	0x00	Sets port pin as input.
	PORT_PIN_OUT	0x01	Sets port pin as output.
Description	Possible directions of a port pin.		
Available via	Port.h		

⌋*()*

[SWS_Port_00046] ⌈The type Port_PinDirectionType is a type for defining the direction of a Port Pin.⌋*()*

[SWS_Port_00220] ⌈The type Port_PinDirectionType shall be of enumeration type having range as PORT_PIN_IN and PORT_PIN_OUT.⌋*()*

8.2.4 Port_PinModeType

[SWS_Port_00231] ⌈

Name	Port_PinModeType		
Kind	Type		
Derived from	uint		
Range	Implementation specific	–	As several port pin modes shall be configurable on one pin, the range shall be determined by the implementation.
Description	Different port pin modes.		
Available via	Port.h		

⌋*()*

[SWS_Port_00124] ⌈A port pin shall be configurable with a number of port pin modes (type Port_PinModeType).⌋*()*

[SWS_Port_00212] ⌈The type Port_PinModeType shall be used with the function call Port_SetPinMode.⌋*()*

[SWS_Port_00221] ⌈The type Port_PinModeType shall be uint8, uint16 or uint32.⌋*()*

8.3 Function definitions

This is a list of functions provided for upper layer modules.

8.3.1 Port_Init

[SWS_Port_00140] ⌈

Service Name	Port_Init	
Syntax	`void Port_Init ( const Port_ConfigType* ConfigPtr )`	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to configuration set.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initializes the Port Driver module.	
Available via	Port.h	

⌋*(SRS_BSW_00358)*

[SWS_Port_00041] ⌈The function Port_Init shall initialize ALL ports and port pins with the configuration set pointed to by the parameter ConfigPtr.⌋*(SRS_BSW_00101, SRS_BSW_00404, SRS_SPAL_12263, SRS_SPAL_12057, SRS_SPAL_12125)*

[SWS_Port_00078] ⌈The Port Driver module's environment shall call the function Port_Init first in order to initialize the port for use.⌋*()*

[SWS_Port_00213] ⌈If Port_Init function is not called first, then no operation can occur on the MCU ports and port pins.⌋*()*

[SWS_Port_00042] ⌈The function Port_Init shall initialize all configured resources.⌋*(SRS_BSW_00101, SRS_SPAL_12057, SRS_SPAL_12125)*

The function Port_Init shall apply the following rules regarding initialisation of controller registers.

- **[SWS_Port_00113]** ⌈If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register.⌋*(SRS_SPAL_12461)*

- **[SWS_Port_00214]** ⌈If the register can affect several hardware modules and if it is an I/O register it shall be initialised by this PORT driver.⌋*(SRS_SPAL_12461)*
- **[SWS_Port_00215]** ⌈If the register can affect several hardware modules and if it is not an I/O register, it shall be initialised by the MCU driver [8].⌋*(SRS_SPAL_-12461)*
- **[SWS_Port_00217]** ⌈One-time writable registers that require initialisation directly after reset shall be initialised by the startup code.⌋*(SRS_SPAL_12461)*
- **[SWS_Port_00218]** ⌈All the other registers not mentioned before, shall be initialised by the start-up code.⌋*(SRS_SPAL_12461)*

[SWS_Port_00043] ⌈The function `Port_Init` shall avoid glitches and spikes on the affected port pins.⌋*(SRS_SPAL_12057)*

[SWS_Port_00071] ⌈The Port Driver module's environment shall call the function `Port_Init` after a reset in order to reconfigure the ports and port pins of the MCU.⌋*()*

[SWS_Port_00002] ⌈The function `Port_Init` shall initialize all variables used by the PORT driver module to an initial state.⌋*(SRS_BSW_00101)*

[SWS_Port_00003] ⌈The Port Driver module's environment may also uses the function `Port_Init` to initialize the driver software and reinitialize the ports and port pins to another configured state depending on the configuration set passed to this function.⌋*(SRS_SPAL_12163)*

Note: In some cases, MCU port control hardware provides an output latch for setting the output level on a port pin that may be used as a DIO port pin.

[SWS_Port_00055] ⌈The function `Port_Init` shall set the port pin output latch to a default level (defined during configuration) before setting the port pin direction to output.⌋*()*

Requirement [SWS_Port_00055] ensures that the default level is immediately output on the port pin when it is set to an output port pin.

Example: On some MCU's, after a power-on-reset, a DIO configurable port pin will be configured as an input pin. If the required configuration of the port pin is an output pin, then the function `Port_Init` shall ensure that the default level is set before switching the functionality of the port pin from input to output.

[SWS_Port_00121] ⌈The function `Port_Init` shall always have a pointer as a parameter, even though for the configuration variant VARIANT-PRE-COMPILE, no configuration set shall be given. In this case, the Port Driver module's environment shall pass a NULL pointer to the function `Port_Init`.⌋*(SRS_BSW_00414)*

The Port Driver module's environment shall not call the function `Port_Init` during a running operation. This shall only apply if there is more than one caller of the PORT module.

Configuration of Port_Init: All port pins and their functions, and alternate functions shall be configured by the configuration tool.

8.3.2 Port_SetPinDirection

[SWS_Port_00141] ⌈

Service Name	Port_SetPinDirection	
Syntax	`void Port_SetPinDirection ( Port_PinType Pin, Port_PinDirectionType Direction )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Pin	Port Pin ID number
	Direction	Port Pin Direction
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Sets the port pin direction	
Available via	Port.h	

⌋*()*

[SWS_Port_00063] ⌈The function Port_SetPinDirection shall set the port pin direction during runtime.⌋*(SRS_Port_12405)*

[SWS_Port_00054] ⌈The function Port_SetPinDirection shall be re-entrant if accessing different pins independent of a port.⌋*()*

[SWS_Port_00086] ⌈The function Port_SetPinDirection shall only be available to the user if the pre-compile parameter PortSetPinDirectionApi is set to TRUE. If set to FALSE, the function Port_SetPinDirection is not available.⌋*(SRS_Port_12405)*

Configuration of Port_SetPinDirection: All ports and port pins shall be configured by the configuration tool. See PORT117.

8.3.3 Port_RefreshPortDirection

[SWS_Port_00142] ⌈

Service Name	Port_RefreshPortDirection

Syntax	`void Port_RefreshPortDirection (`
	`  void`
	`)`

Service ID [hex]	0x02
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Refreshes port direction.
Available via	Port.h

⌋*()*

[SWS_Port_00060] ⌈The function `Port_RefreshPortDirection` shall refresh the direction of all configured ports to the configured direction (`PortPinDirection`).⌋*(SRS_Port_12406)*

[SWS_Port_00061] ⌈The function `Port_RefreshPortDirection` shall exclude those port pins from refreshing that are configured as 'pin direction changeable during runtime'.⌋*(SRS_Port_12406)*

The configuration tool shall provide names for each configured port pin.

8.3.4 Port_GetVersionInfo

[SWS_Port_00143] ⌈

Service Name	Port_GetVersionInfo	

Syntax	`void Port_GetVersionInfo (`
	`  Std_VersionInfoType* versioninfo`
	`)`	

Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	versioninfo	Pointer to where to store the version information of this module.
Return value	None	
Description	Returns the version information of this module.	
Available via	Port.h	

⌋*()*

[SWS_Port_00225] ⌈if Det is enabled, the parameter versioninfo shall be checked for being NULL. The error PORT_E_PARAM_POINTER shall be reported in case the value is a NULL pointer.⌋*()*

8.3.5 Port_SetPinMode

[SWS_Port_00145] ⌈

Service Name	Port_SetPinMode	
Syntax	`void Port_SetPinMode ( Port_PinType Pin, Port_PinModeType Mode )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Pin	Port Pin ID number
	Mode	New Port Pin mode to be set on port pin.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Sets the port pin mode.	
Available via	Port.h	

⌋*()*

[SWS_Port_00125] ⌈The function Port_SetPinMode shall set the port pin mode of the referenced pin during runtime.⌋*()*

[SWS_Port_00128] ⌈The function Port_SetPinMode shall be re-entrant if accessing different pins, independent of a port.⌋*()*

[SWS_Port_00223] ⌈If Det is enabled, the function Port_SetPinMode shall report

PORT_E_MODE_UNCHANGEABLE error and return without any other action, if the parameter PortPinModeChangeable is set to FALSE.⌋*()*

Configuration of Port_SetPinMode: All ports and port pins shall be configured by the configuration tool. See PORT117.

8.4 Call-back notifications

There are no callback notifications from the PORT driver. The callback notifications are implemented in another module (ICU Driver and/or complex drivers).

8.5 Scheduled functions

There are no scheduled functions within the PORT Driver.

8.6 Expected Interfaces

In this chapter, all interfaces required from other modules are listed.

8.6.1 Mandatory Interfaces

None

8.6.2 Optional Interfaces

This chapter defines all interfaces which are required to fulfill an optional functionality of the module.

[SWS_Port_00146] ⌈

API Function	***Header File***	***Description***
Det_ReportError	Det.h	Service to report development errors.

⌋*()*

8.6.3 Configurable Interfaces

None

9 Sequence diagrams

9.1 Overall Configuration of Ports

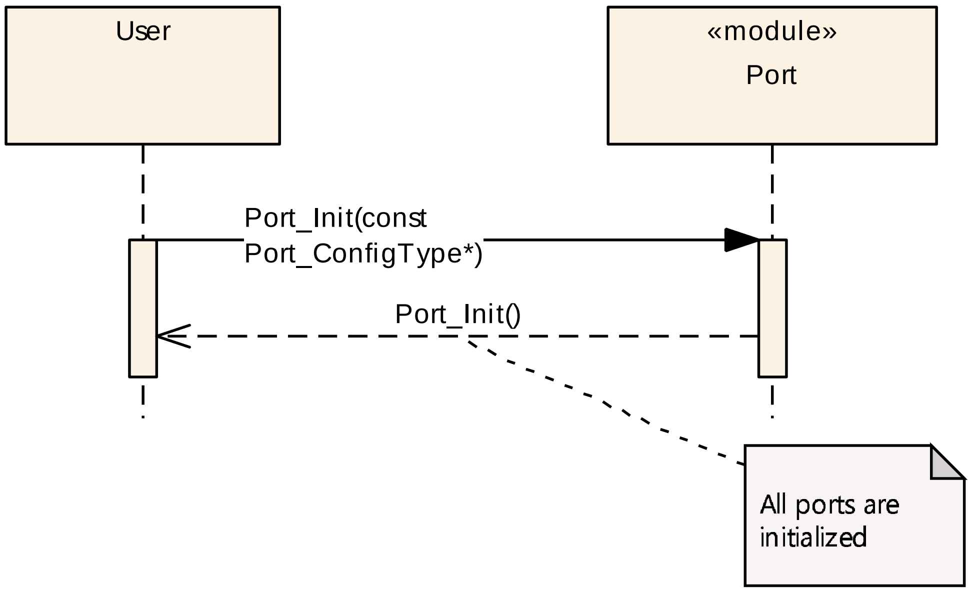

Figure 9.1: Overall Configuration of Ports

9.2 Set the direction of a Port Pin

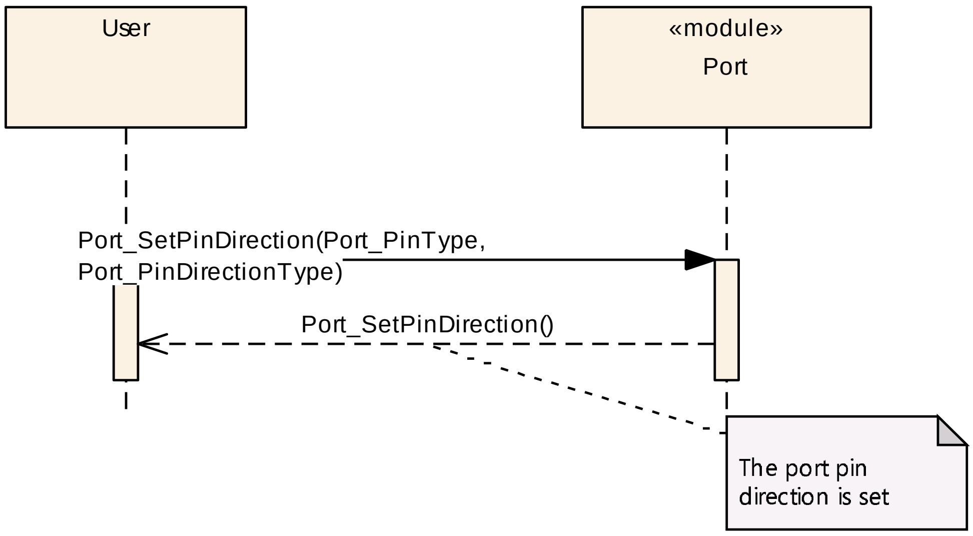
User
«module»
Port
Port_SetPinDirection(Port_PinType, Port_PinDirectionType)
Port_SetPinDirection()
The port pin direction is set

Figure 9.2: Set the direction of a Port Pin

9.3 Refresh the direction of all Port Pins

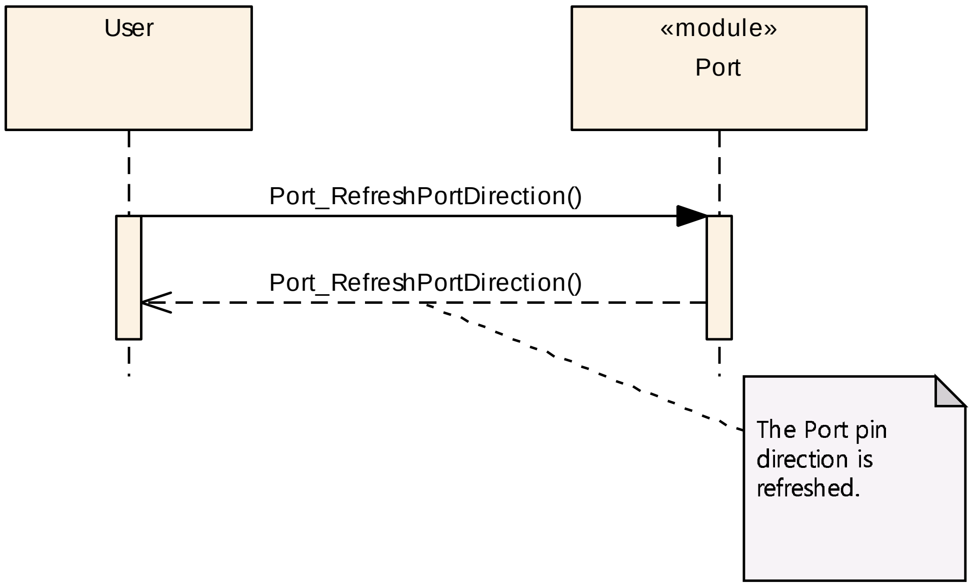
User
«module»
Port
Port_RefreshPortDirection()
Port_RefreshPortDirection()
The Port pin direction is refreshed.

Figure 9.3: Refresh the direction of all Port Pins

9.4 Change the mode of a Port Pin

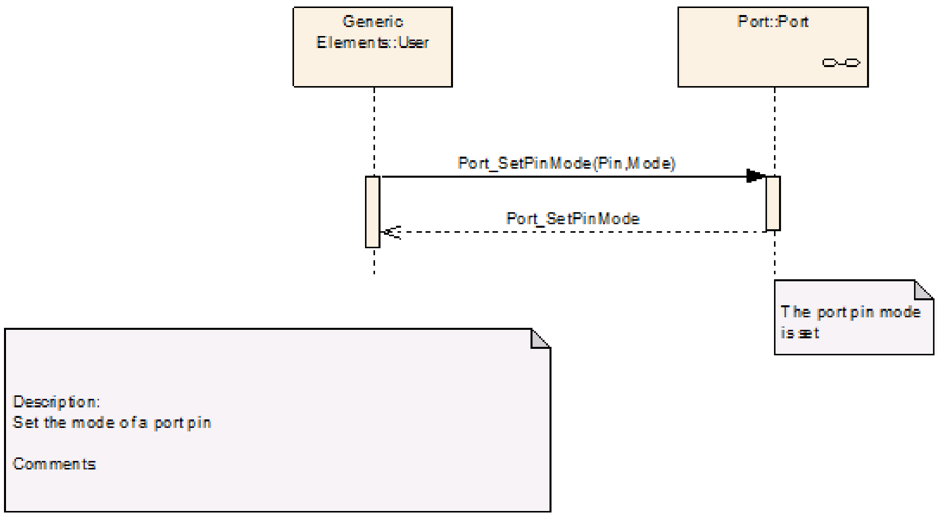

Figure 9.4: Change the mode of a Port Pin

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Section 10.2 specifies the structure (containers) and the parameters of the module PORT

Section 10.3 specifies published information of the module PORT.

10.1 How to read this chapter

For details refer to the Chapter 10.1 "Introduction to configuration specification" in [4].

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 8.

[SWS_Port_00232] ⌈The PORT module shall reject configurations with partition mappings which are not supported by the implementation.⌋*()*

10.2.1 Port

SWS Item	**[ECUC_Port_00135]**
Module Name	Port
Description	Configuration of the Port module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
PortConfigSet	1	This container contains the configuration parameters and sub containers of the AUTOSAR Port module.
PortGeneral	1	Module wide configuration parameters of the PORT driver.

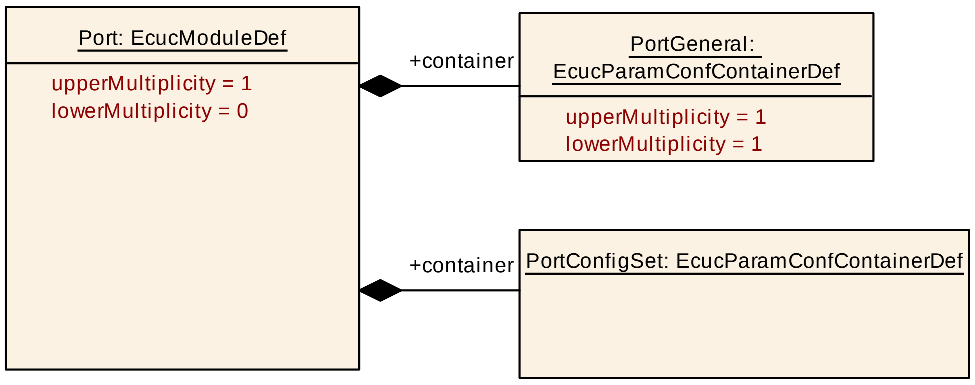

Figure 10.1: Port Configuration Overview

10.2.2 PortContainer

SWS Item	**[ECUC_Port_00122]**
Container Name	PortContainer
Parent Container	PortConfigSet
Description	Container collecting the PortPins.
Configuration Parameters	

SWS Item	**[ECUC_Port_00124]**		
Parameter Name	PortNumberOfPortPins		
Parent Container	PortContainer		
Description	The number of specified PortPins in this PortContainer.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
PortPin	1..*	Configuration of the individual port pins.

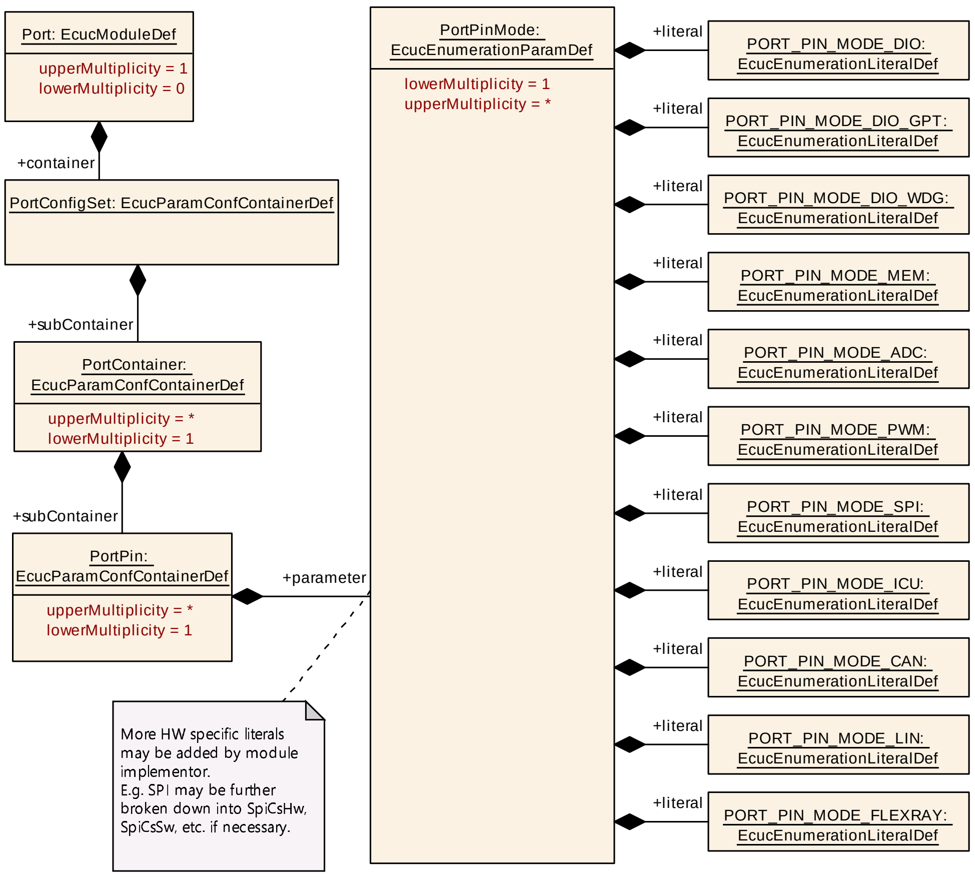
Port: EcucModuleDef
upperMultiplicity = 1
lowerMultiplicity = 0
+container
PortConfigSet: EcucParamConfContainerDef
+subContainer
PortContainer: EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
+subContainer
PortPin: EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
+parameter
PortPinMode: EcucEnumerationParamDef
lowerMultiplicity = 1
upperMultiplicity = *
More HW specific literals may be added by module implementor.
E.g. SPI may be further broken down into SpiCsHw, SpiCsSw, etc. if necessary.
+literal
PORT_PIN_MODE_DIO: EcucEnumerationLiteralDef
PORT_PIN_MODE_DIO_GPT: EcucEnumerationLiteralDef
PORT_PIN_MODE_DIO_WDG: EcucEnumerationLiteralDef
PORT_PIN_MODE_MEM: EcucEnumerationLiteralDef
PORT_PIN_MODE_ADC: EcucEnumerationLiteralDef
PORT_PIN_MODE_PWM: EcucEnumerationLiteralDef
PORT_PIN_MODE_SPI: EcucEnumerationLiteralDef
PORT_PIN_MODE_ICU: EcucEnumerationLiteralDef
PORT_PIN_MODE_CAN: EcucEnumerationLiteralDef
PORT_PIN_MODE_LIN: EcucEnumerationLiteralDef
PORT_PIN_MODE_FLEXRAY: EcucEnumerationLiteralDef

Figure 10.2: PortContainer Parameters Overview

10.2.3 PortGeneral

SWS Item	**[ECUC_Port_00117]**
Container Name	PortGeneral
Parent Container	Port
Description	Module wide configuration parameters of the PORT driver.
Configuration Parameters	

SWS Item	**[ECUC_Port_00123]**		
Parameter Name	PortDevErrorDetect		
Parent Container	PortGeneral		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants

▽

△

	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Port_00131]**		
Parameter Name	PortSetPinDirectionApi		
Parent Container	PortGeneral		
Description	Pre-processor switch to enable / disable the use of the function Port_SetPinDirection(). TRUE: Enabled - Function Port_SetPinDirection() is available. FALSE: Disabled - Function Port_SetPinDirection() is not available.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Port_00132]**		
Parameter Name	PortSetPinModeApi		
Parent Container	PortGeneral		
Description	Pre-processor switch to enable / disable the use of the function Port_SetPinMode(). true: Enabled - Function Port_SetPinMode() is available. false: Disabled - Function Port_SetPinMode() is not available.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Port_00133]**		
Parameter Name	PortVersionInfoApi		
Parent Container	PortGeneral		
Description	Pre-processor switch to enable / disable the API to read out the modules version information. true: Version info API enabled. false: Version info API disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_Port_00136]		
Parameter Name	PortEcucPartitionRef		
Parent Container	PortGeneral		
Description	Maps the Port driver to zero a multiple ECUC partitions to make the modules API available in this partition.		
Multiplicity	0..*		
Type	Reference to EcucPartition		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

The top level Port Driver container holds parameters that apply to the PORT configuration.

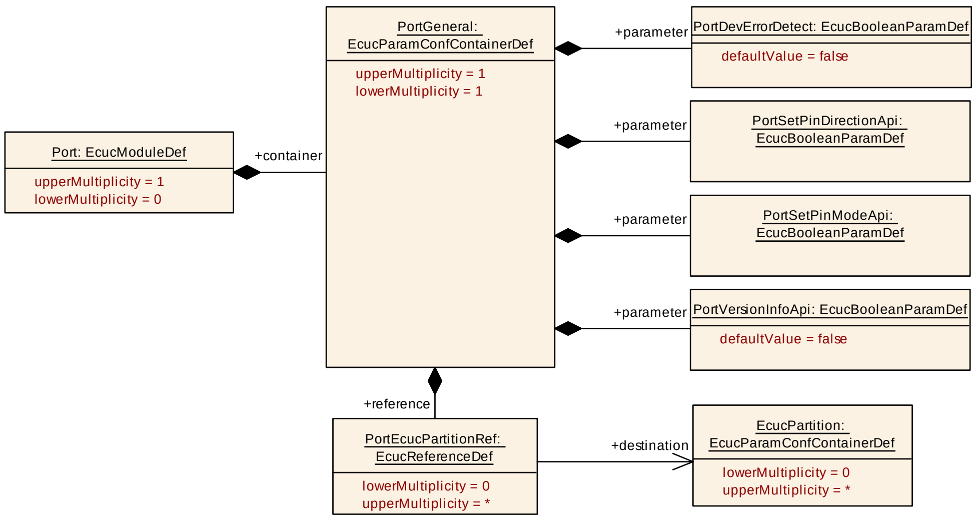

Figure 10.3: PortGeneral Parameters Overview

10.2.4 PortPin

SWS Item	[ECUC_Port_00118]
Container Name	PortPin
Parent Container	PortContainer
Description	Configuration of the individual port pins.
Configuration Parameters	

SWS Item	[ECUC_Port_00125]		
Parameter Name	PortPinDirection		
Parent Container	PortPin		
Description	The initial direction of the pin (IN or OUT). If the direction is not changeable, the value configured here is fixed. The direction must match the pin mode. E.g. a pin used for an ADC must be configured to be an in port. Implementation Type: Port_PinDirectionType		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	PORT_PIN_IN	Port Pin direction set as input	
	PORT_PIN_OUT	Port Pin direction set as output	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_Port_00126]		
Parameter Name	PortPinDirectionChangeable		
Parent Container	PortPin		
Description	Parameter to indicate if the direction is changeable on a port pin during runtime. true: Port Pin direction changeable enabled. false: Port Pin direction changeable disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_Port_00127]		
Parameter Name	PortPinId		
Parent Container	PortPin		
Description	Pin Id of the port pin. This value will be assigned to the symbolic name derived from the port pin container short name.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	1 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Port_00128]**		
Parameter Name	PortPinInitialMode		
Parent Container	PortPin		
Description	Port pin mode from mode list for use with Port_Init() function.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	PORT_PIN_MODE_ADC	Port Pin used by ADC	
	PORT_PIN_MODE_CAN	Port Pin used for CAN	
	PORT_PIN_MODE_DIO	Port Pin configured for DIO. It shall be used under control of the DIO driver.	
	PORT_PIN_MODE_DIO_GPT	Port Pin configured for DIO. It shall be used under control of the general purpose timer driver.	
	PORT_PIN_MODE_DIO_WDG	Port Pin configured for DIO. It shall be used under control of the watchdog driver.	
	PORT_PIN_MODE_FLEXRAY	Port Pin used for FlexRay	
	PORT_PIN_MODE_ICU	Port Pin used by ICU	
	PORT_PIN_MODE_LIN	Port Pin used for LIN	
	PORT_PIN_MODE_MEM	Port Pin used for external memory under control of a memory driver.	
	PORT_PIN_MODE_PWM	Port Pin used by PWM	
	PORT_PIN_MODE_SPI	Port Pin used by SPI	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Port_00129]**		
Parameter Name	PortPinLevelValue		
Parent Container	PortPin		
Description	Port Pin Level value from Port pin list.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	PORT_PIN_LEVEL_HIGH	Port Pin level is High	
	PORT_PIN_LEVEL_LOW	Port Pin level is LOW	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Port_00130]**
Parameter Name	PortPinMode
Parent Container	PortPin
Description	Port pin mode from mode list. Note that more than one mode is allowed by default. That way it is e.g. possible to combine DIO with another mode such as ICU.
Multiplicity	1..*
Type	EcucEnumerationParamDef

▽

△

Range	PORT_PIN_MODE_ADC	Port Pin used by ADC	
	PORT_PIN_MODE_CAN	Port Pin used for CAN	
	PORT_PIN_MODE_DIO	Port Pin configured for DIO. It shall be used under control of the DIO driver.	
	PORT_PIN_MODE_DIO_GPT	Port Pin configured for DIO. It shall be used under control of the general purpose timer driver.	
	PORT_PIN_MODE_DIO_WDG	Port Pin configured for DIO. It shall be used under control of the watchdog driver.	
	PORT_PIN_MODE_FLEXRAY	Port Pin used for FlexRay	
	PORT_PIN_MODE_ICU	Port Pin used by ICU	
	PORT_PIN_MODE_LIN	Port Pin used for LIN	
	PORT_PIN_MODE_MEM	Port Pin used for external memory under control of a memory driver.	
	PORT_PIN_MODE_PWM	Port Pin used by PWM	
	PORT_PIN_MODE_SPI	Port Pin used by SPI	
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Port_00134]**		
Parameter Name	PortPinModeChangeable		
Parent Container	PortPin		
Description	Parameter to indicate if the mode is changeable on a port pin during runtime. True: Port Pin mode changeable allowed. False: Port Pin mode changeable not permitted.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**[ECUC_Port_00137]**
Parameter Name	PortPinEcucPartitionRef
Parent Container	PortPin
Description	Maps the Port pin to zero a multiple ECUC partitions. The ECUC partitions referenced are a subset of the ECUC partitions where the Port driver is mapped to.
Multiplicity	0..*
Type	Reference to EcucPartition
Post-Build Variant Multiplicity	true
Post-Build Variant Value	true

▽

△

Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

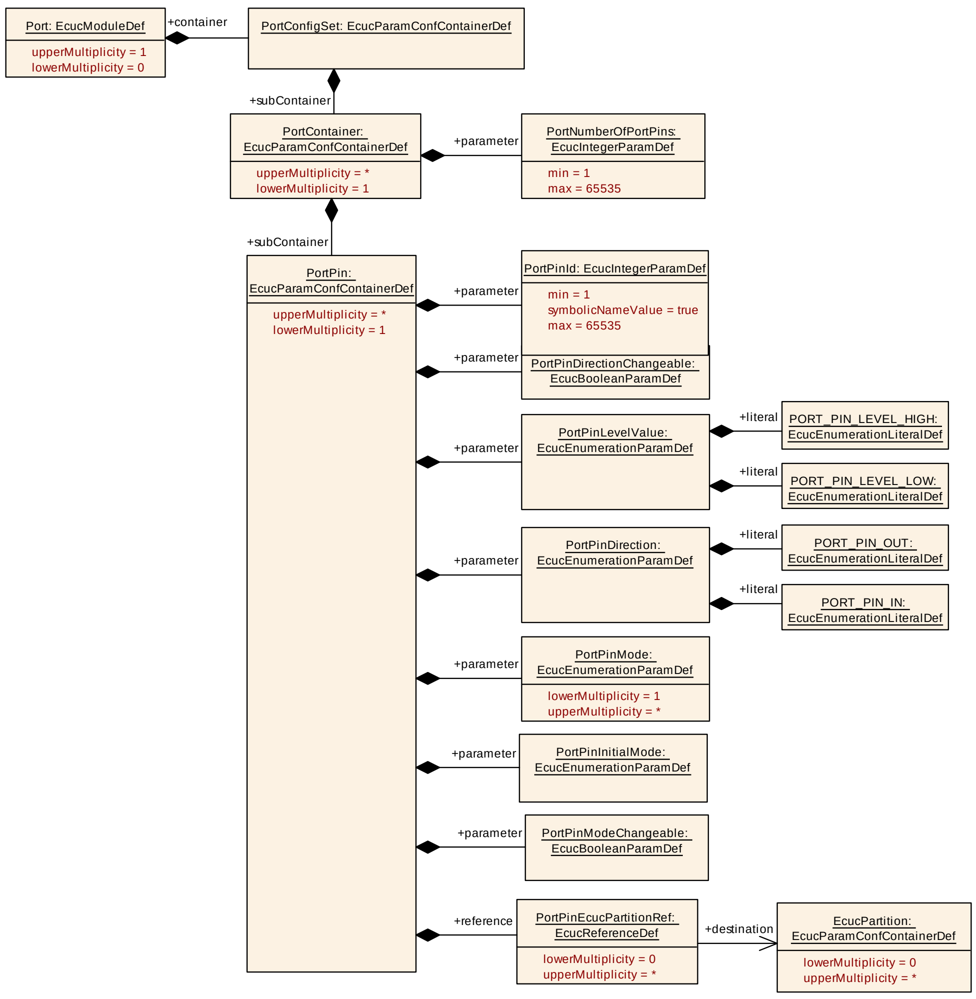
Port: EcucModuleDef
upperMultiplicity = 1
lowerMultiplicity = 0
+container
PortConfigSet: EcucParamConfContainerDef
+subContainer
PortContainer: EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
+parameter
PortNumberOfPortPins: EcucIntegerParamDef
min = 1
max = 65535
+subContainer
PortPin: EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
+parameter
PortPinId: EcucIntegerParamDef
min = 1
symbolicNameValue = true
max = 65535
+parameter
PortPinDirectionChangeable: EcucBooleanParamDef
+parameter
PortPinLevelValue: EcucEnumerationParamDef
+literal
PORT_PIN_LEVEL_HIGH: EcucEnumerationLiteralDef
+literal
PORT_PIN_LEVEL_LOW: EcucEnumerationLiteralDef
+parameter
PortPinDirection: EcucEnumerationParamDef
+literal
PORT_PIN_OUT: EcucEnumerationLiteralDef
+literal
PORT_PIN_IN: EcucEnumerationLiteralDef
+parameter
PortPinMode: EcucEnumerationParamDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
PortPinInitialMode: EcucEnumerationParamDef
+parameter
PortPinModeChangeable: EcucBooleanParamDef
+reference
PortPinEcucPartitionRef: EcucReferenceDef
lowerMultiplicity = 0
upperMultiplicity = *
+destination
EcucPartition: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = *

Figure 10.4: PortPin Parameters Overview

10.3 Constraints

[SWS_Port_CONSTR_00233] ⌈The ECUC partitions referenced by `PortPinEcucPartitionRef` shall be a subset of the ECUC partitions referenced by `PortEcucPartitionRef`.⌋*()*

[SWS_Port_CONSTR_00234] ⌈If `PortEcucPartitionRef` references one or more ECUC partitions, `PortPinEcucPartitionRef` shall have a multiplicity of greater than zero and reference one or several of these ECUC partitions as well.⌋*()*

10.4 Published Information

For details refer to the Chapter 10.3 "Published Information" in [4].

A Not applicable requirements

[SWS_Port_NA_00227] ⌈These requirements are not applicable to this specification.⌋*(SRS_BSW_00005, SRS_BSW_00006, SRS_BSW_00007, SRS_BSW_00010, SRS_BSW_00160, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00164, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00172, SRS_BSW_00307, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00321, SRS_BSW_00325, SRS_BSW_00328, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00333, SRS_BSW_00334, SRS_BSW_00335, SRS_BSW_00336, SRS_BSW_00341, SRS_BSW_00342, SRS_BSW_00343, SRS_BSW_00344, SRS_BSW_00347, SRS_BSW_00357, SRS_BSW_00359, SRS_BSW_00360, SRS_SPAL_12463, SRS_SPAL_12462, SRS_SPAL_12265, SRS_SPAL_12092, SRS_SPAL_12078, SRS_SPAL_12077, SRS_SPAL_12067, SRS_SPAL_12064, SRS_SPAL_12129, SRS_SPAL_12075, SRS_SPAL_12063, SRS_SPAL_12169, SRS_SPAL_00157, SRS_SPAL_12069, SRS_SPAL_12068, SRS_SPAL_12267, SRS_SPAL_12056, SRS_BSW_00440, SRS_BSW_00439, SRS_BSW_00437, SRS_BSW_00433, SRS_BSW_00432, SRS_BSW_00429, SRS_BSW_00428, SRS_BSW_00427, SRS_BSW_00426, SRS_BSW_00425, SRS_BSW_00424, SRS_BSW_00423, SRS_BSW_00419, SRS_BSW_00417, SRS_BSW_00416, SRS_BSW_00413, SRS_BSW_00398, SRS_BSW_00395, SRS_BSW_00378, SRS_BSW_00377, SRS_BSW_00375, SRS_BSW_00373, SRS_BSW_00371)*