Document Title	Specification of PORT Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	040
Document Classification	Standard

Document Version	3.2.2
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change Description
28.02.2014	3.2.2	AUTOSAR Release Management	• Editorial changes • Removed chapter(s) on change documentation
19.12.2013	3.2.1	AUTOSAR Administration	• Update outlines
27.04.2011	3.2.0	AUTOSAR Administration	• Update Chapter 8/10
29.01.2010	3.1.0	AUTOSAR Administration	• Range insertion for the parameter PortPinInitialMode (PortPin Container) in chapter 10 • Legal disclaimer revised
23.06.2008	3.0.2	AUTOSAR Administration	Legal disclaimer revised
23.01.2008	3.0.1	AUTOSAR Administration	Table formatting corrected

Document Change History

27.11.2007	3.0.0	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
31.01.2007	2.1.0	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 PORT105. • Removal of redundant requirements PORT119 and PORT026 in requirements matrix. • Legal disclaimer revised • Release Notes added • “Advice for users” revised • “Revision Information” added
28.04.2006	2.0.0	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
30.06.2005	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of 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.

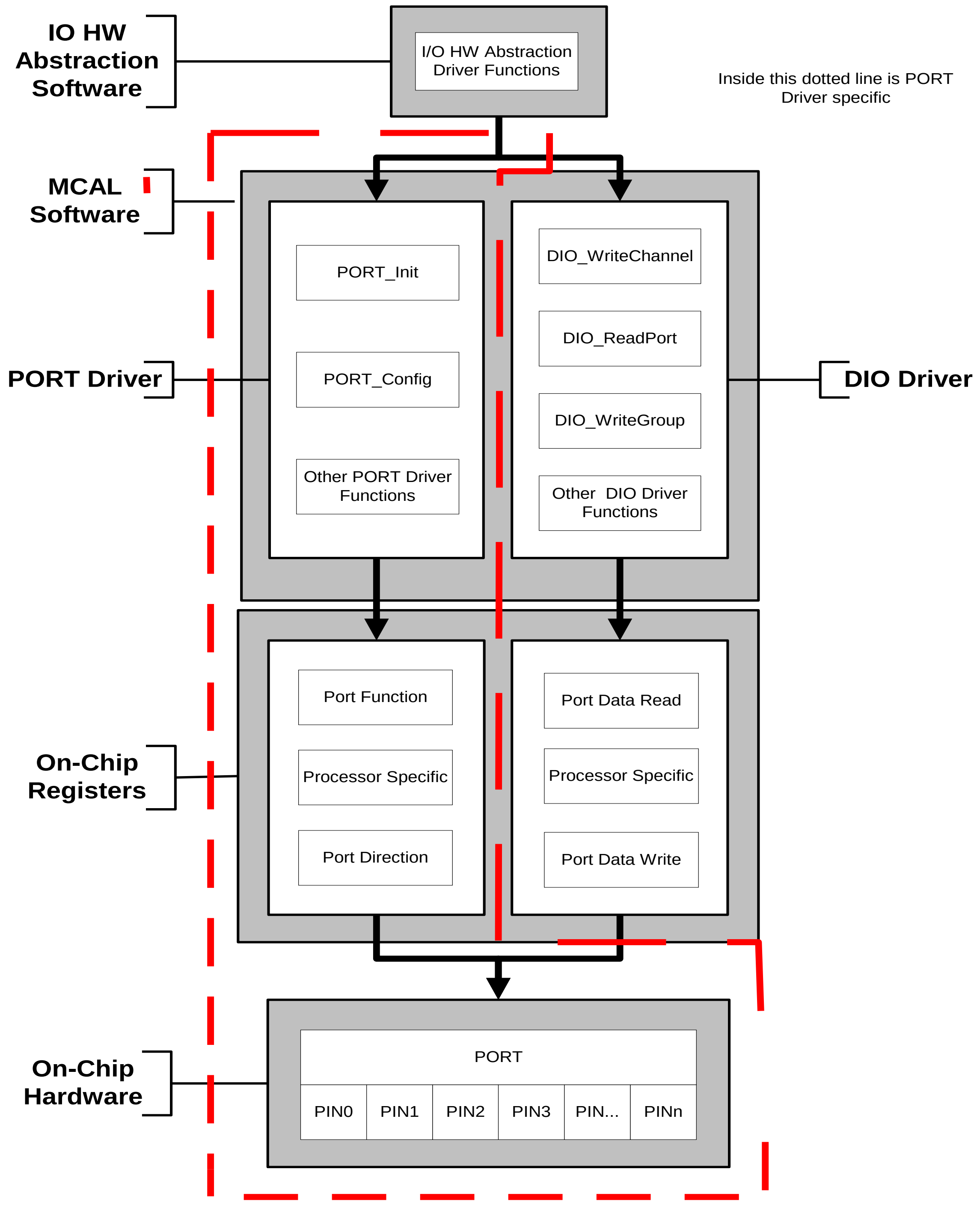
Port initialisation data shall be written to each port as efficiently as possible.

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.

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
DIO Driver
PORT_Init
PORT_Config
Other PORT Driver Functions
DIO_WriteChannel
DIO_ReadPort
DIO_WriteGroup
Other DIO Driver Functions
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

2 Acronyms and abbreviations

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

Abbreviation / Acronym:	***Description:***
DEM	Diagnostic Event Manager
DET	Development Error Tracer
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

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules,
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture,
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules,
AUTOSAR_SRS_General.pdf

[4] Specification of Development Error Tracer,
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[5] Specification of ECU Configuration,
AUTOSAR_ECU_Configuration.pdf

[6] Specification of Diagnostic Event Manager (DEM),
AUTOSAR_SWS_DEM.pdf

[7] Specification of ECU State Manager,
AUTOSAR_SWS_Ecu_StateManager.pdf

[8] General Requirements on SPAL,
AUTOSAR_SRS_SPAL_General.pdf

[9] Requirements on PORT driver,
AUTOSAR_SRS_PORT_Driver.pdf

[10] Specification of Standard Types,
AUTOSAR_SWS_StandardTypes.pdf

[11] AUTOSAR Basic Software Module Description Template,
AUTOSAR_BSW_Module_Description.pdf

3.2 Related standards and norms

[12] EC 7498-1 The Basic Model, IEC Norm, 1994

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

PORT108: The code file structure shall not be defined within this specification completely. At this point, it shall be pointed out that the code file structure shall include the following files named:

- `Port_Lcfg.c` – for link time configurable parameters and
- `Port_PBcfg.c` – for post build time configurable parameters.

These files shall contain all link time and post-build time configurable parameters.

5.1.2 Header file structure

PORT080: The include file structure of the Port Driver shall be as follows:

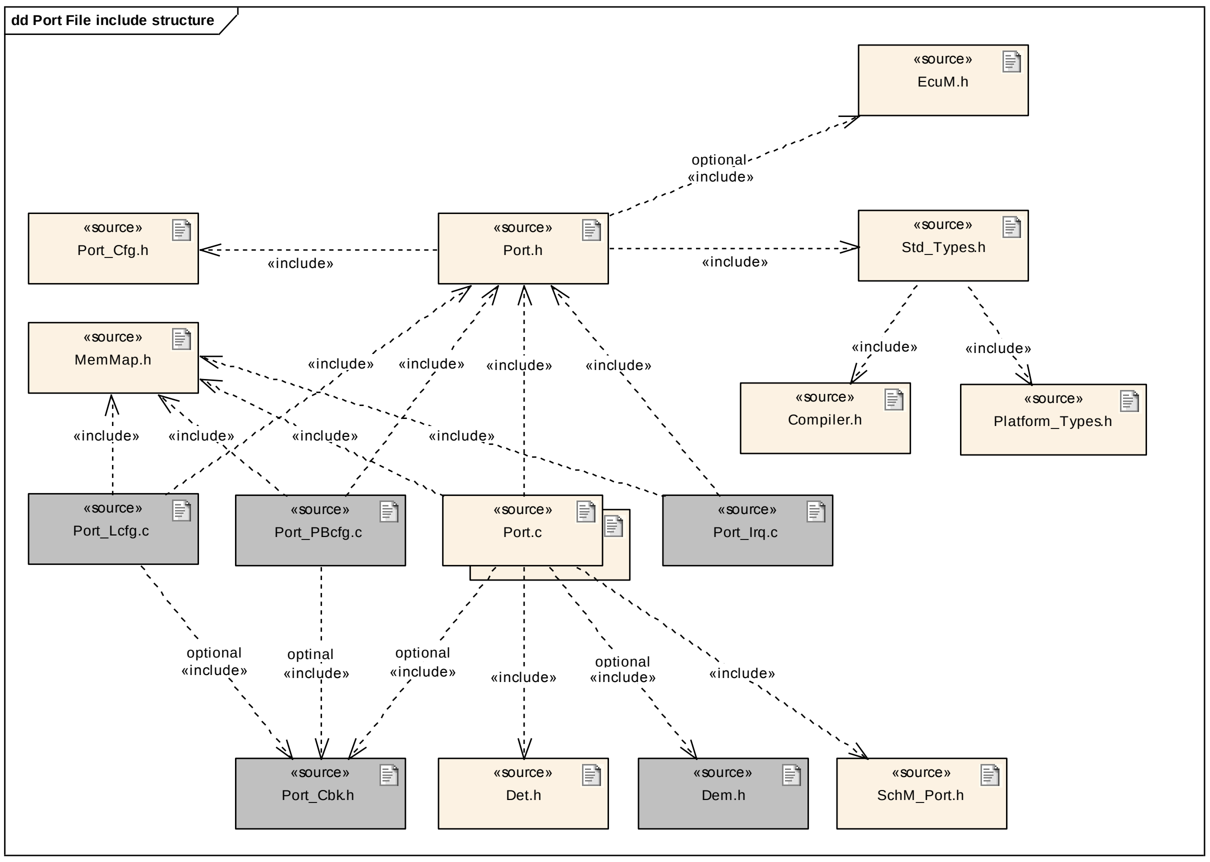


Figure 1 Header File Structure

The grey boxes are optional:

PORT130: `Port.h` shall include `Port_Cfg.h` for the API pre-compiler switches.

PORT131: `Port.c` shall include `Port.h` (PORT114).

`Port.c` has implicit access to the `Port_Cfg.h` file through the `Port.h` file.

PORT132: `Port_Irq.c` shall include `Port.h` for definition of the function to be called within the interrupt function.

PORT133: The type definitions for `Port_Lcfg.c` and `Port_PBcfg.c` shall be located in the file `Port_Cfg.h` or `Port.h`.

The implicit include of `Port_Cfg.h` via `Port.h` in the files `Port_Lcfg.c` and `Port_PBcfg.c` is necessary to solve the following construct:

```
Port.h
----------
#ifdef xxx_VERSION_INFO_API
xxx_GetVersionInfo(...)
#endif

Port_Cfg.h
---------------
#include "Port.h"
#define xxx_VERSION_INFO_API
```

Note: A separate file type is not required for the PORT driver as the Port Types depend on the platform and are not configurable.

6 Requirements traceability

This chapter refers to the input requirements specified in the SRS documents (Software Requirements Specifications) 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.

Document: AUTOSAR requirements on Basic Software, general [3]

Requirement	***Satisfied by***
[BSW003] Version identification	PORT106
[BSW004] Version check	PORT114, Chapter 7.1.3
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (Because architectural AUTOSAR concept is the basis for this concept)
[BSW006] Platform independency	Not applicable (because the module is not above the MCAL)
[BSW007] HIS MISRA C	Not applicable (Because it is requirement on implementation)
[BSW009] Module User Documentation	Section 3.1
[BSW010] Memory resource documentation	Not applicable (Because this is a requirement for the implementer)
[BSW101] Initialization interface	PORT001, PORT002, PORT041, PORT042
[BSW158] Separation of configuration from implementation	Figure 5.1: Header File
[BSW159] Tool-based configuration	PORT004
[BSW160] Human-readable configuration data	Not applicable (Because it only applies to the configuration. Requirement on implementation)
[BSW161] Microcontroller abstraction	Not applicable (Because architectural AUTOSAR concept is the basis for this concept)
[BSW162] ECU layout abstraction	Not applicable (Because architectural AUTOSAR concept is the basis for this concept)
[BSW164] Implementation of interrupt service routines	Not applicable (Because the PORT does not provide interrupt functionality
[BSW167] Static configuration checking	Not Applicable (Because this is a requirement for the configuration tool).
[BSW168] Diagnostic Interface of SW components	Not applicable (Because the PORT does not provide diagnostic capabilities
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable (Because it only affects the configuration)

[BSW171] Configurability of optional functionality	PORT117, PORT118
[BSW172] Compatibility and documentation of scheduling strategy	Not applicable (Because PORT does not have any special scheduling requirements
[BSW00300] Module naming convention	Figure 5.1
[BSW00301] Limit imported information	Figure 5.1
[BSW00302] Limit exported information	Figure 5.1
[BSW00304] AUTOSAR integer data types	Section 8.2
[BSW00305] Self-defined data types naming convention	Section 8.2
[BSW00306] Avoid direct use of compiler and platform specific keywords	Figure 5.1
[BSW00307] Global variables naming convention	Not applicable (Because it is a requirement on implementation)
[BSW00308] Definition of global data	Not applicable (Because this is a requirement for the implementer)
[BSW00309] Global data with read-only constraint	Section 8.3.1
[BSW00310] API naming convention	Section 8.3
[BSW00312] Shared code shall be reentrant	Section 8.3.2
[BSW00314] Separation of interrupt frames and service routines	Figure 5.1
[BSW00318] Format of module version numbers	PORT106
[BSW00321] Enumeration of module version numbers	Not applicable (Because this is a requirement for the Implementer)
[BSW00323] API parameter checking	PORT031
[BSW00325] Runtime of interrupt service routines	Not applicable (Because the PORT driver does not provide interrupt capabilities. This is a requirement for implementation)
[BSW00326] Transition from ISRs to OS tasks	Not applicable (Because PORT does not provide interrupt capabilities. This is a requirement on implementation)
[BSW00327] Error values naming convention	PORT051
[BSW00328] Avoid duplication of code	Not applicable (Because this is a requirement for the implementer)
[BSW00329] Avoidance of generic interfaces	Not applicable (Because there are no generic interfaces specified within this SWS)
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable (Because this is a requirement for the implementer)
[BSW00331] Separation of error and status values	Not applicable (Because there are no status values specified within this SWS)

[BSW00333] Documentation of callback function context	Not applicable (Because it is a non functional requirement. There is no callback functionality in the PORT module)
[BSW00334] Provision of XML file	Not applicable (Because this is specified by WP4.1.1.2)
[BSW00335] Status values naming convention	Not applicable (Because there are no status values specified within the SWS)
[BSW00336] Shutdown interface	Not applicable (Because for the PORT driver there is no need for this requirement.
[BSW00337] Classification of errors	PORT051
[BSW00338] Detection and Reporting of development errors	PORT100
[BSW00339] Reporting of production relevant error status	PORT037
[BSW00341] Microcontroller compatibility documentation	Not applicable (Because this is a requirement for the Implementer)
[BSW00342] Usage of source code and object code	Not applicable (Because it is requirement on implementation)
[BSW00343] Specification and configuration of time	Not applicable (Because it is a non functional requirement. Time configuration is not a requirement of the PORT driver.
[BSW00344] Reference to Link-time configuration	PORT117
[BSW00345] Pre-compile-time configuration	Figure 5.1: Header File, PORT117
[BSW00346] Basic set of module files	Figure 5.1: Header File
[BSW00347] Naming separation of different instances of BSW drivers	Not applicable (Because it is a requirement on implementation)
[BSW00348] Standard type header	Figure 5.1
[BSW00350] Development error detection keyword	PORT117
[BSW00353] Platform specific type header	Figure 5.1
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable (Because no integer data types are redefined in this specification)
[BSW00357] Standard API return type	Not applicable (Because this type is not used within the SWS)
[BSW00358] Return type of init() functions	Section 8.3.1
[BSW00359] Return type of callback functions	Not applicable (Because the PORT module does not provide a callback mechanism)
[BSW00360] Parameters of callback functions	Not applicable (Because the PORT module does not provide a callback mechanism)
[BSW00361] Compiler specific language extension header	Figure 5.1
[BSW00369] Do not return development error codes via API	PORT037

[BSW00370] Separation of callback interface from API	Not applicable (Because the PORT driver does not provide a callback mechanism)
[BSW00371] Do not pass function pointers via API	Not applicable (Because no function pointers are passed via API in this SWS)
[BSW00373] Main processing function naming convention	Not applicable (Because it is a non functional requirement. There is no main processing function specified in the PORT driver)
[BSW00374] Module vendor identification	PORT106
[BSW00375] Notification of wake-up reason	Not applicable (Because the PORT driver does not provide a wake-up mechanism)
[BSW00376] Return type and parameters of main processing functions	Not applicable (Because there is no main processing function specified)
[BSW00377] Module specific API return types	Not applicable (Because this type is not used within the SWS)
[BSW00378] AUTOSAR boolean type	Section 10.2.3
[BSW00379] Module identification	PORT106
[BSW00380] Separate C-File for configuration parameters	Figure 5.1: Header File
[BSW00381] Separate configuration header file for pre-compile time parameters	Figure 5.1: Header File
[BSW00383] List dependencies of configuration files	PORT080
[BSW00384] List dependencies to other modules	PORT080
[BSW00385] List possible error notificatons	PORT051
[BSW00386] Configuration for detecting an error	Chapter 7.2.1
[BSW00387] Specify the configuration class of callback function	Not applicable (Because the PORT driver does not have any callback capability).
[BSW00388] Introduce containers	Chapter 10.2
[BSW00389] Containers shall have names	Chapter 10.2
[BSW00390] Parameter content shall be unique within the module	Chapter 8.3
[BSW00391] Parameter shall have unique names	Chapter 8.3
[BSW00392] Parameters shall have a type	Chapter 8.3
[BSW00393] Parameters shall have a range	Chapter 8.3
[BSW00394] Specify the scope of the parameters	Chapter 8.3
[BSW00395] List the required parameters (per parameter)	Not applicable (Because none of the parameters of the PORT driver are dependent on other modules)
[BSW00396] Configuration classes	Chapter 10.2
[BSW00397] Pre-compile-time parameters	PORT117
[BSW00398] Link-time parameters	Chapter 10.2
[BSW00399] Loadable Post-build time parameters	Chapter 10.2
[BSW00400] Selectable Post-build time parameters	Chapter 10.2
[BSW00401] Documentation of multiple instances of configuration parameters	Chapter 10.2
[BSW00402] Published information	Chapter 10.3
[BSW00404] Reference to post build time configuration	PORT041
[BSW00405] Reference to multiple configuration sets	Chapter 10.2.2
[BSW00406] Check module initialization	PORT041
[BSW00407] Function to read out published parameters	PORT102

[BSW00408] Configuration parameter naming convention	Chapter 10.2
[BSW00409] Header files for production code error IDs	Section 5.1
[BSW00410] Compiler switches shall have defined values	Chapter 10.2
[BSW00411] Get version info keyword	PORT103
[BSW00412] Separate H-File for configuration parameters	Figure 5.1: Header file
[BSW00413] Accessing instances of BSW modules	Not applicable (Because this is a requirement on implementation)
[BSW00414] Parameter of init function	PORT121
[BSW00415] User dependent include files	Figure 5.1
[BSW00416] Sequence of Initialization	Not applicable (Because this requirement describes the initialization of the whole SPAL layer).
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable (Because this driver is part of the basic software layer. This requirement applies only for non-BSW modules).
[BSW00419] Separate C-Files for pre-compile time configuration parameters	Figure 5.1
[BSW00420] Production relevant error event rate detection	Not applicable (Because it is a non functional requirement and applies only for DEM)
[BSW00421] Reporting of production relevant error events	PORT037
[BSW00422] Debouncing of production relevant error status	PORT037
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (Because it is a non-functional requirement. The PORT driver has no AUTOSAR interface)
[BSW00424] BSW main processing function task allocation	Not applicable (Because the PORT driver does not contain any main processing functions)
[BSW00425] Trigger conditions for schedulable objects	Not applicable (Because the PORT driver does not contain any schedulable objects/services. This is a requirement for the Implementer).
[BSW00426] Exclusive areas in BSW modules	Not applicable (Because applies only for the module descriptions template)
[BSW00427] ISR description for BSW modules	Not applicable (Because this is a requirement for the implementer)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (Because there is no main processing function specified).
[BSW00429] Restricted BSW OS functionality access	PORT084
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (Because this requirement is for an upper layer. There is no scheduling functionality in the PORT module).
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (Because the PORT driver does not contain any main processing functions).
[BSW00433] Calling of main processing functions	Not applicable (Because for the PORT driver there is no main processing function specified).
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (Because it is a non functional

	requirement. There is no scheduling functionality in the PORT driver)
[BSW00435] Header files Structure for the Basic Software Scheduler.	See Section 5.1.2
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping.	See Section 5.1.2
[BSW00437] NoInit—Area in RAM	Not applicable (Because this is a requirement for the implementer)
[BSW00438] Post build Configuration data Structure	See Section 10.

Document: AUTOSAR requirements on Basic Software, cluster SPAL, General [8]

Requirement	***Satisfied by***
[BSW12263] Object code compatible configuration concept	PORT041
[BSW12056] Configuration of notification mechanism	Not Applicable (Because the PORT driver does not include notification functionality).
[BSW12267] Configuration of wakeup sources	Not Applicable (Because there is no wake-up functionality associated to the PORT driver).
[BSW12057] Driver module initialisation	PORT041, PORT042, PORT043
[BSW12125] Initialization of hardware resources	PORT041, PORT042
[BSW12163] Driver module deinitialization	PORT003
[BSW12068] MCAL initialization sequence	Not applicable (Because this requirement describes the initialisation of the whole SPAL layer).
[BSW12069] Wake-up notification of ECU State Manager	Not applicable (Because the PORT driver has no wake-up functionality).
[BSW157] Notification mechanisms of drivers and handlers	Not applicable (Because there is no notification functionality associated to the PORT driver.
[BSW12169] Control of operation mode	Not applicable (Because there is no set mode functionality in the PORT driver).
[BSW12063] Raw value mode	Not applicable (Because there is no functionality in the PORT driver for the raw value mode).
[BSW12075] Use of application buffers	Not applicable (Because there is no random streaming capability)
[BSW12129] Resetting of interrupt flags	Not applicable (Because the interrupt functionality is not part of the PORT driver).
[BSW12064] Change of operation mode during running operation	Not applicable (Because this is a non-functional requirement concerning system design).
[BSW12067] Setting of wake-up conditions	Not applicable (Because the PORT driver has no wake-up conditions).
[BSW12448] Behaviour after development error detection	PORT037
[BSW12077] Non-blocking implementation	Not applicable (Because this is a requirement for the implementer)

[BSW12078] Runtime and memory efficiency	Not applicable (Because this is a requirement for the implementer)
[BSW12092] Access to drivers	Not applicable (Because this is a non-functional requirement concering the system design).
[BSW12265] Configuration data shall be kept constant	Not applicable (Because this is a requirement for the implementer)
[BSW12264] Specification of configuration items	Chapter 10 - Configuration specification.
[BSW12461] Responsibility for register initialisation	PORT113
[BSW12462] Provide settings for register initialisation.	Chapter 10.3 – Published Information

[BSW12463] Combine and forward settings for register initialisation.	Not applicable (Because this is a requirement for a configuration tool).

Document: Requirements on Basic Software, Module SPAL, PORT Driver

Requirement	***Satisfied by***
[BSW12001] Configuration of Port Pin Properties	PORT004, PORT079, PORT072
[BSW12302] Configuration of symbolic names	PORT006
[BSW12405] Set port pin direction	PORT063, PORT086, PORT138
[BSW12406] Refresh port direction	PORT060, PORT061
[BSW12300] Configuration of unused port pins	PORT005
[BSW12423] Provide atomicity of port access	PORT075

7 Functional specification

7.1 General Behaviour

7.1.1 Background & Rationale

PORT001: The PORT Driver module shall initialize the whole port structure of the microcontroller.

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

PORT004: The PORT Driver module shall allow the configuration of different functionality for each port and port pin, e.g. ADC, SPI, DIO etc. The configuration of the port (i.e. whole port or single port pin) is microcontroller dependent.

PORT079: 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.

PORT081: 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).

PORT082: The PORT Driver module shall not provide for the configuration of 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

PORT137: 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.

PORT138: 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.

7.1.2.3 Refresh port direction

PORT066: 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

PORT005: 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.

7.1.2.5 Configuration of symbolic names

PORT006: The user of the PORT Driver module shall configure the symbolic names of the port pins of the MCU. These symbolic names for the individual port pins (e.g. PORT_A_PIN_0) shall be defined in the configuration tool.

PORT076: The PORT Driver module's implementer shall define symbolic names in the file `Port_Cfg.h` and publish the symbolic names through the file `Port.h`.

7.1.2.6 Atomicity of port access

PORT075: The PORT Driver module shall provide atomic access to all ports and port pins by the use of either atomic instructions or the usage of an exclusive area 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

PORT114: The integration of incompatible files shall be avoided. Minimum implementation is the version check of the header file.
For included header files,

- PORT_AR_MAJOR_VERSION
- PORT_AR_MINOR_VERSION

shall be identical.

For the module internal c and h files,

- PORT_SW_MAJOR_VERSION
- PORT_SW_MINOR_VERSION
- PORT_AR_MAJOR_VERSION
- PORT_AR_MINOR_VERSION
- PORT_AR_PATCH_VERSION

shall be identical.

7.2 Error classification

PORT115: Values for production code Event Ids are assigned externally by the configuration of the DEM. They are published in the file `Dem_IntErrId.h` and included via `Dem.h`.

By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in `Dem_IntErrId.h`.

PORT116: Development error values are of type `uint8`.

PORT051: The following errors and exceptions shall be detectable by the PORT driver depending on its build version (development/production).

Type or error	***Relevance***	***Related error code***	***Value***
Invalid Port Pin ID requested	Development	`PORT_E_PARAM_PIN`	`0x0A`
Port Pin not configured as changeable	Development	`PORT_E_DIRECTION_UNCHANGEABLE`	`0x0B`
API Port_Init service called with wrong parameter.	Development	`PORT_E_PARAM_CONFIG`	`0x0C`
API Port_SetPinMode service called when mode is unchangeable.	Development	`PORT_E_PARAM_INVALID_MODE`	`0x0D`
		`PORT_E_MODE_UNCHANGEABLE`	`0x0E`
API service called without module initialization	Development	`PORT_E_UNINIT`	`0x0F`

7.3 Error detection

PORT100: The detection of development errors is configurable (ON/OFF) at pre-compile time. The switch `PortDevErrorDetect` (see Chapter 10) shall activate or deactivate the detection of all development errors.

PORT101: If the PortDevErrorDetect switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in Chapter 7.2 and Chapter 8.

PORT139: The detection of production code errors cannot be switched off.

7.4 Error notification

PORT037: Production errors shall be reported to the Diagnostic Event Manager [Ref.6].
PORT038: The PORT module's implementer shall add to the PORT device specific implementation specification additional errors that are to be detected because of specific implementation and/or specific hardware properties . The classification and enumeration shall be compatible to the errors listed above [PORT051].

PORT107: Detected development errors shall be reported to the `Det_ReportError` service of the Development Error Tracer (DET) if the pre-processor switch `PortDevErrorDetect` is set (see Chapter 10).

7.5 API Parameter checking

PORT031: If development error detection is enabled for the PORT driver, the following API parameter checking shall be performed according to the respective functions (see table below).

PORT077: 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.

PORT087: 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.

Function	***Error Condition***	***Related 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_PARAM_CONFIG`
`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_GetVersionInfo Port_RefreshPortDirection	API service called prior to module initialization	`PORT_E_UNINIT`

8 API specification

8.1 Imported types

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

PORT129:

Module	***Imported Type***
Dem	Dem_EventIdType
Std_Types	Std_VersionInfoType

8.2 Type definitions

8.2.1 Port_ConfigType

Name:	`Port_ConfigType`	
Type:	`Structure`	
Range:	`Hardware Dependent Structure`	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.	

PORT073: The type `Port_ConfigType` is a type for the external data structure containing the initialization data for the PORT Driver.

Note: The user shall use the symbolic names defined in the configuration tool.

Note: The configuration of each port pin is MCU specific. Therefore, it is not possible to include a complete list of different configurations in this specification.

PORT072: A list of possible port configurations for the structure `Port_ConfigType` is given below:

- Pin mode (e.g. DIO, ADC, SPI, …) – this port pin configuration is mandatory unless the port pin is configured for DIO.
- Pin direction (input, output) – this port pin configuration is mandatory when the port pin is to be used for DIO.
- Pin level init value (see PORT055) – this port pin configuration is mandatory when the port pin is used for DIO.
- Pin direction changeable during runtime (STD_ON/STD_OFF) – this port pin configuration is MCU dependent.
- Pin mode changeable during runtime (STD_ON/STD_OFF) – configuration is MCU dependent.

Optional parameters (if supported by hardware)

- Slew rate control.
- Activation of internal pull-ups.
- Microcontroller specific port pin properties.

8.2.2 Port_PinType

Name:	`Port_PinType`	
Type:	`Unsigned Integer`	
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.	

PORT013: The type `Port_PinType` shall be used for the symbolic name of a Port Pin.

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

Name:	`Port_PinDirectionType`	
Type:	`Enumeration`	
Range:	`PORT_PIN_IN`	Sets port pin as input.
	`PORT_PIN_OUT`	Sets port pin as output.
Description:	Possible directions of a port pin.	

PORT046: The type `Port_PinDirectionType` is a type for defining the direction of a Port Pin.
Port_PinModeType

Name:	`Port_PinModeType`	
Type:	`Unsigned Integer`	
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.	

PORT124: A port pin shall be configurable with a number of port pin modes (type `Port_PinModeType`). The type `Port_PinModeType` shall be used with the function call `Port_SetPinMode` (see Section 8.3.5).

8.3 Function definitions

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

8.3.1 Port_Init

PORT140:

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.

PORT041: The function `Port_Init` shall initialize ALL ports and port pins with the configuration set pointed to by the parameter `ConfigPtr`.

PORT078: The Port Driver module's environment shall call the function `Port_Init` first in order to initialize the port for use. If not called first, then no operation can occur on the MCU ports and port pins.

PORT042: The function `Port_Init` shall initialize all configured resources.

PORT113: The function `Port_Init` shall apply the following rules regarding initialisation of controller registers:

1. If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register.
2. If the register can affect several hardware modules and if it is an I/O register it shall be initialised by this PORT driver.
3. 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.
4. One-time writable registers that require initialisation directly after reset shall be initialised by the startup code.
5. All other registers shall be initialised by the start-up code.

PORT043: The function `Port_Init` shall avoid glitches and spikes on the affected port pins.

PORT071: 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.

PORT002: The function `Port_Init` shall initialize all variables used by the PORT driver module to an initial state.

PORT003: 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.

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.

PORT055: 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 PORT055 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 shall 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.

PORT105: If development error detection for the Port Driver module is enabled: In case the function Port_Init is called with a NULL ConfigPtr and if a variant containing postbuild multiple selectable configuration parameters is used (Variant PB), the function `Port_Init` shall raise the development error `PORT_E_PARAM_CONFIG` and return without any action.

PORT121: The function `Port_Init` shall always have a pointer as a parameter, even though for the configuration variant VariantPC, 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`.

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

PORT141:

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	

PORT063: The function `Port_SetPinDirection` shall set the port pin direction during runtime.

PORT054: The function `Port_SetPinDirection` shall be re-entrant if accessing different pins independent of a port.

PORT086: The function `Port_SetPinDirection` shall only be available to the userif the runtime parameter `PortPinDirectionChangeable` is set to TRUE. If set to FALSE, the function `Port_SetPinDirection` is not applicable. (see Section 10.2.2)

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

8.3.3 Port_RefreshPortDirection

PORT142:

Service name:	Port_RefreshPortDirection

Syntax:	`void Port_RefreshPortDirection(`
	
	`)`

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.

PORT060: The function `Port_RefreshPortDirection` shall refresh the direction of all configured ports to the configured direction (`PortPinDirection`).

PORT061: The function `Port_RefreshPortDirection` shall exclude those port pins from refreshing that are configured as ‘pin direction changeable during runtime‘.

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

8.3.4 Port_GetVersionInfo

PORT143:

Service name:	Port_GetVersionInfo	

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

Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Non 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.	

PORT102: The function `Port_GetVersionInfo` shall return the version information of this module. The version information includes:
- Module Id
- Vendor Id
- Vendor specific version numbers (BSW00407).

PORT103: The function `Port_GetVersionInfo` shall be pre compile time configurable On/Off by the configuration parameter `PortVersionInfoApi`.

PORT144: If source code for caller and callee of `Port_GetVersionInfo` is available, the PORT Driver module should realize `Port_GetVersionInfo` as a macro, defined in the module's header file.

8.3.5 Port_SetPinMode

PORT145:

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.	

PORT125: The function `Port_SetPinMode` shall set the port pin mode of the referenced pin during runtime.

PORT128: The function `Port_SetPinMode` shall be re-entrant if accessing different pins, independent of a port.

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.

PORT146:

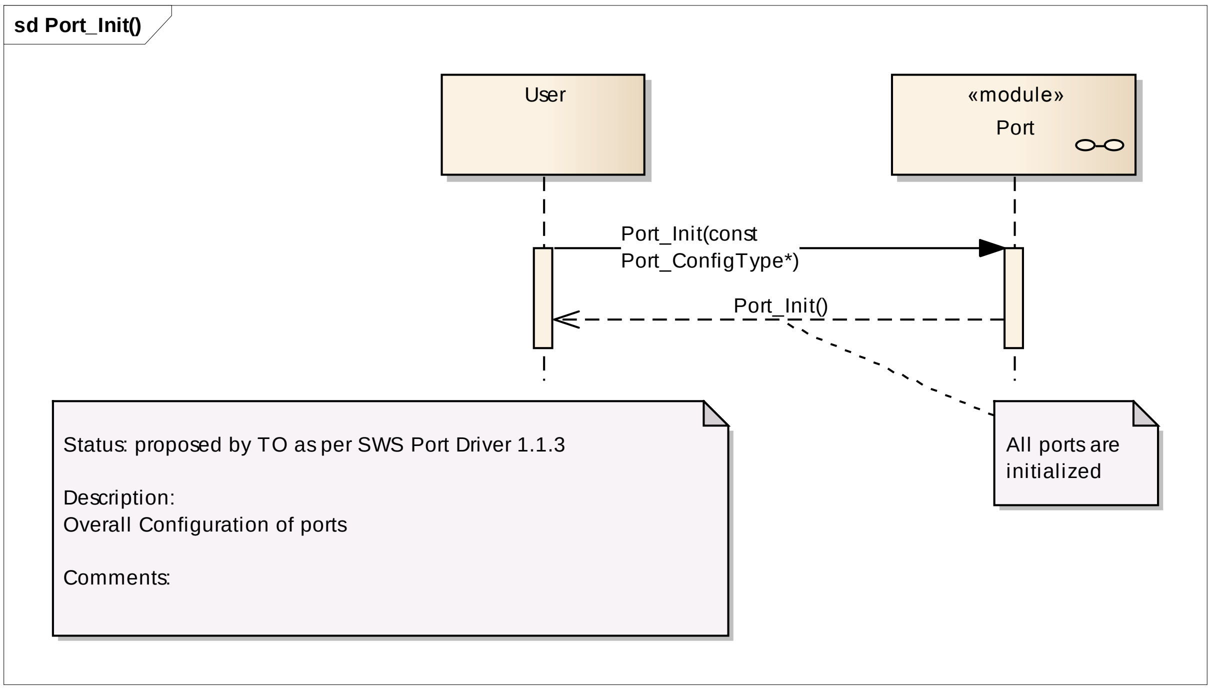
API function	***Description***
Dem_ReportErrorStatus	Reports errors to the DEM.
Det_ReportError	Service to report development errors.

8.6.3 Configurable Interfaces

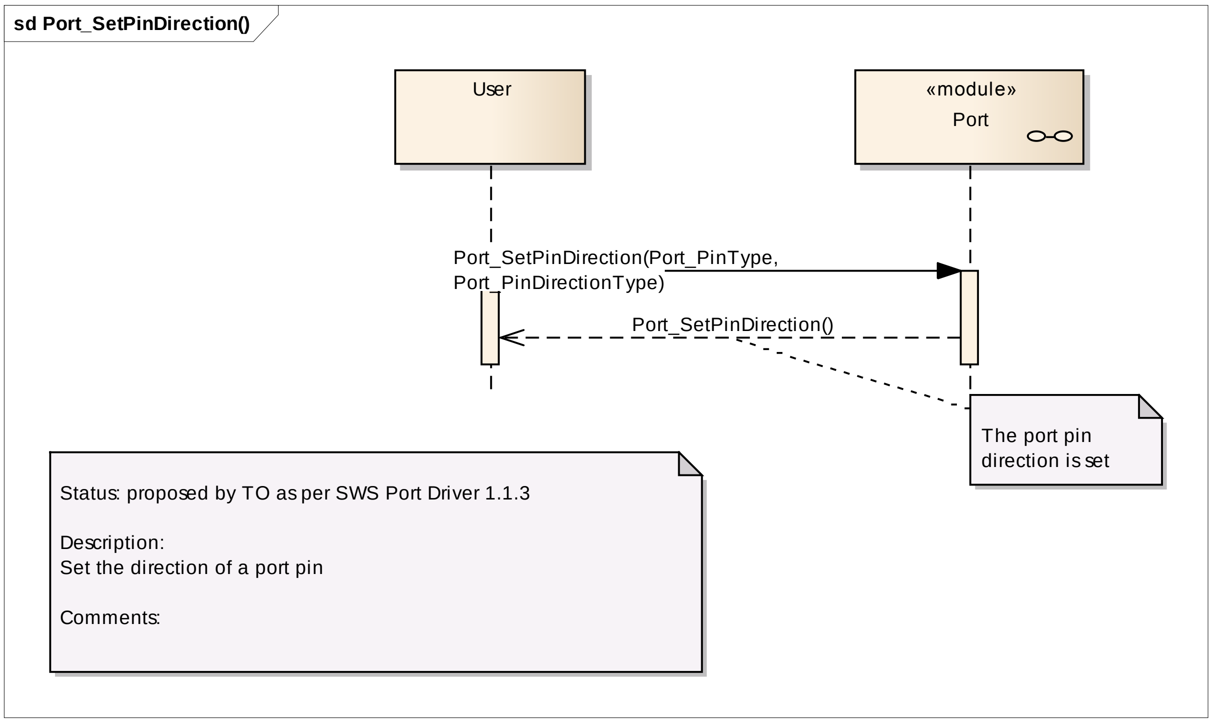
None

9 Sequence diagrams

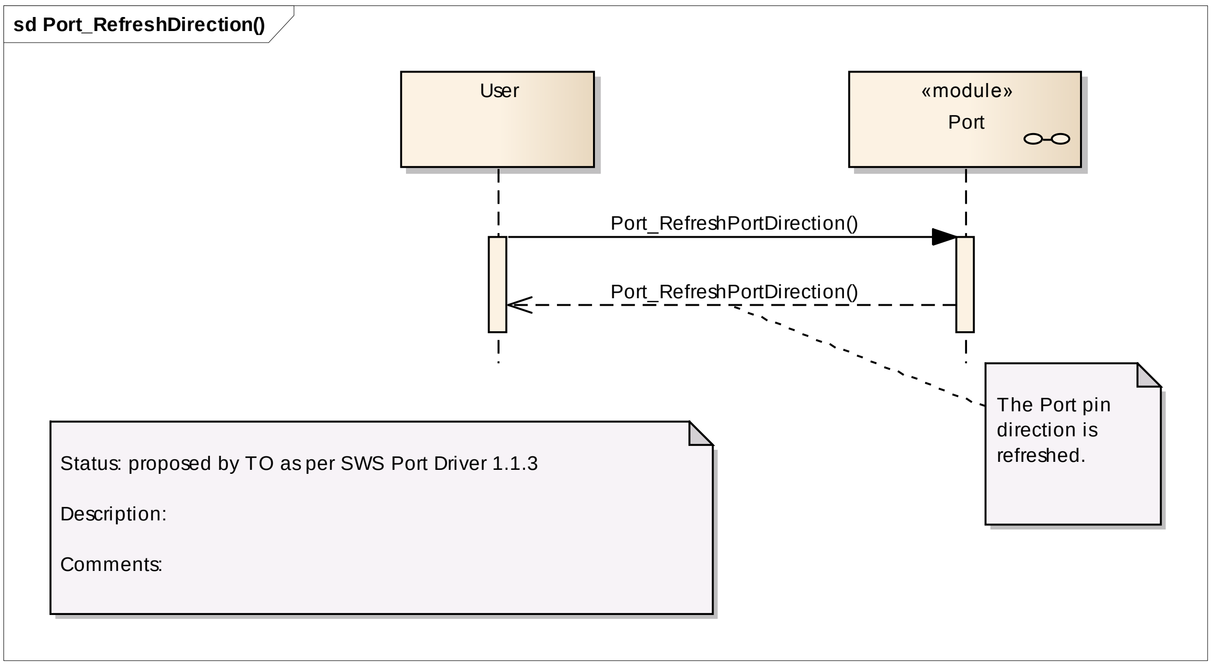
9.1 Overall Configuration of Ports



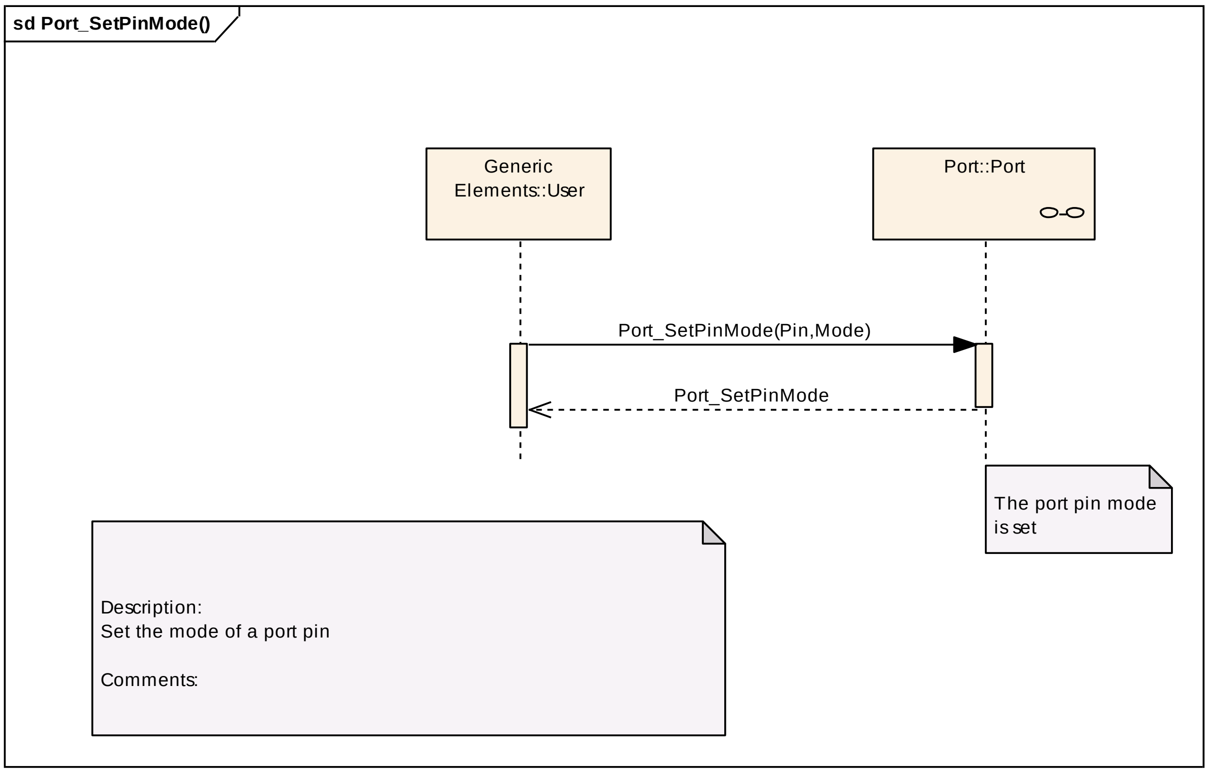
9.2 Set the direction of a Port Pin


sd Port_SetPinDirection()
User
«module»
Port
Port_SetPinDirection(Port_PinType,
Port_PinDirectionType)
Port_SetPinDirection()
Status: proposed by TO as per SWS Port Driver 1.1.3
Description:
Set the direction of a port pin
Comments:
The port pin
direction is set

9.3 Refresh the direction of all Port Pins


sd Port_RefreshDirection()
User
«module»
Port
Port_RefreshPortDirection()
Port_RefreshPortDirection()
The Port pin direction is refreshed.
Status: proposed by TO as per SWS Port Driver 1.1.3
Description:
Comments:

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.

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

Chapter 10.3 specifies published information of the module PORT.

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [2].
- AUTOSAR ECU Configuration Specification [5]
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.

(sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

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.

10.2.1 Variants

PORT147: VariantPC: This variant is limited to pre-compile-configuration parameters only. The intention of this variant is to optimize the parameters configuration for a source code delivery.

PORT148: VariantPB: This variant allows a mix of pre-compile time-, post build-time configuration parameters. The intention of this variant is to optimize the parameters configuration for a re-loadable binary.

10.2.2 Port

Module Name	*Port*
Module Description	Configuration of the Port module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PortConfigSet	1	This container is the base of a multiple configuration set
PortGeneral	1	Module wide configuration parameters of the PORT driver.

10.2.3 PortContainer

SWS Item	:
Container Name	PortContainer
Description	Container collecting the PortPins.
Configuration Parameters	

SWS Item	:		
Name	PortNumberOfPortPins		
Description	The number of specified PortPins in this PortContainer.		
Multiplicity	1		
Type	IntegerParamDef		
Range	..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

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

10.2.4 PortGeneral

SWS Item	**PORT117 :**
Container Name	PortGeneral{PORT General configuration}
Description	Module wide configuration parameters of the PORT driver.
Configuration Parameters	

SWS Item	:		
Name	PortDevErrorDetect {PORT_DEV_ERROR_DETECT}		
Description	Switches the Development Error Detection and Notification on or off. true: Enabled. false: Disabled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:		
Name	PortSetPinDirectionApi {PORT_SET_PIN_DIRECTION_API}		
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	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:		
Name	PortSetPinModeApi {PORT_SET_PIN_MODE_API}		
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	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:
Name	PortVersionInfoApi {PORT_VERSION_INFO_API}

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	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

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

10.2.5 PortPin

SWS Item	**PORT118 :**
Container Name	PortPin
Description	Configuration of the individual port pins.
Configuration Parameters	

SWS Item	**:**		
Name	PortPinDirection {PORT_PIN_DIRECTION}		
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	EnumerationParamDef		
Range	PORT_PIN_IN	Port Pin direction set as input	
	PORT_PIN_OUT	Port Pin direction set as output	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**:**		
Name	PortPinDirectionChangeable {PORT_PIN_DIRECTION_CHANGEABLE}		
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	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**:**

Name	PortPinId		
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	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 ..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:		
Name	PortPinInitialMode {PORT_PIN_INITIAL_MODE}		
Description	Port pin mode from mode list for use with Port_Init() function.		
Multiplicity	1		
Type	EnumerationParamDef		
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	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	:		
Name	PortPinLevelValue {PORT_PIN_LEVEL_VALUE}		
Description	Port Pin Level value from Port pin list.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	PORT_PIN_LEVEL_HIGH	Port Pin level is High	
	PORT_PIN_LEVEL_LOW	Port Pin level is LOW	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	:
Name	PortPinMode {PORT_PIN_MODE}

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	EnumerationParamDef		
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	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.6 PortConfigSet

SWS Item	:
Container Name	PortConfigSet [Multi Config Container]
Description	This container is the base of a multiple configuration set
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PortContainer	1..*	Container collecting the PortPins.

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

The standard common published information like

vendorId (<Module>_VENDOR_ID),
moduleId (<Module>_MODULE_ID),
arMajorVersion (<Module>_AR_MAJOR_VERSION),
arMinorVersion (<Module>_ AR_MINOR_VERSION),
arPatchVersion (<Module>_ AR_PATCH_VERSION),
swMajorVersion (<Module>_SW_MAJOR_VERSION),
swMinorVersion (<Module>_ SW_MINOR_VERSION),
swPatchVersion (<Module>_ SW_PATCH_VERSION),
vendorApiInfix (<Module>_VENDOR_API_INFIX)

is provided in the BSW Module Description Template (see [11] Figure 4.1 and Figure 7.1).

Additional published parameters are listed below if applicable for this module.