Document Title	Specification of PORT Driver
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	2.1.0
Document Status	Draft
Part of Release	2.1
Revision	0014

Document Change History

Date	Version	Changed by	Change Description
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 • Section 10.2.2: Inclusion of new pre-processor switch PORT_SET_PIN_MODE_API • Section 10.2.3: New configurable parameter introduced: PORT_PIN_INITIAL_MODE • 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		Initial Release

Release Notes

Compatibility considerations with respect to current release

The changes to the PORT specification document from Release 2.0 are as follows:

1. Rewording of requirement PORT105.
   - The original wording of requirement PORT105 could not be fulfilled for the variant Precompile (PC). With the variant PC a NULL pointer is expected as the valid parameter. Therefore a correction to the wording of PORT105 was made.
2. Removal of requirements PORT119 and PORT026 from section 6 (Requirements Matrix). These requirements were not part of the PORT Specification document for Release 2.0 and therefore needed to be removed from the requirements matrix for Release 2.1.
3. New Module Type definition:
   - Port_PinModeType:
     - A limitation of the PORT driver resulted in the inability to change a pin feature during runtime (without generating a reset of the Electronic Control Unit).
     - Therefore a new feature was added to the PORT specification document to allow a number of port pin modes to be configurable on a port pin during runtime. This type was introduced to be used with the function call Port_SetPinMode().
     - New requirement PORT124 applies to the introduction of this new module type definition.
4. New API introduced:
   - Port_SetPinMode:
     - To enable the User to set the port pin mode during runtime.
5. New Configurable Parameter introduced to section 10.2.3:
   - PORT_PIN_INITIAL_MODE
     - Configurable parameter introduced for usage with Port_Init() function.
6. Inclusion of pre-processor switch in section 10.2.2
   - PORT_SET_PIN_MODE_API:
     - Pre-processor switch to enable / disable the use of the function Port_SetPinMode().
7. Update to section 5.1.2:
   - Inclusion of new file structure information.

Errata and known deficiencies

Please refer to list of outstanding issues below (Changes planned for next release).

Known and potential problems resulting from known deficiencies

Please refer to list of outstanding issues below (Changes planned for next release).

Changes planned for next release

The following changes are planned for the next releases:

Outstanding issues with the PORT module which may need introduction to future releases of the PORT specification document are:

1. PORT Container request:
   - The AUTOSAR Parameter Definition for the Port module contains a PortPin container (multiplicity 1:N) which should contain the parameters for configuring a single PortPin. For several reasons, it is not possible in a tool chain to map the configuration found in this structure to the right pins on different ports.
   - A suggestion for this issue would be to introduce a Port container (multiplicity 1:N) and to enter the PortPin as sub-container of this Port. This way a proper mapping of PortPins to Ports would be possible as well as entering Port-specific configurations at the 'right' location.
2. Missing SRS General requirements in Chapter 6:
   - In accordance with AUTOSAR_SWS_Template inclusion of the following requirements to Section 6 should be added:
     - **[BSW00435]** Header file Structure for the Basic Software Scheduler
     - **[BSW00436]** Module Header File Structure for the Basic Software Memory Mapping
3. Removal of function in section 8.6.2:
   - Within section 8.6.2 the obsolete function "Dem_ReportErrorEvent" should be removed.
4. Development Errors:
   - Due to the introduction of the new function Port_SetPinMode(), a request for a new development error was raised.
   - PORT_E_MODE_UNCHANGEABLE value Ox0D should be added in chapter 7.2.2
5. Removal of Non-existing requirements in Chapter 11:
   - PORT119 and PORT120 should be removed from Chapter 11 (as not applicable anymore).
6. Re-wording of requirement PORT075
   - Due to the introduction of the basic scheduler module, it may be required to reformulate the requirement PORT075. Suggested re-wording is: “The PORT driver shall provide atomic access to all ports and port pins by the use of either atomic instruction or the usage exclusive area of the basic software scheduler module.”

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents 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 Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, 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

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.

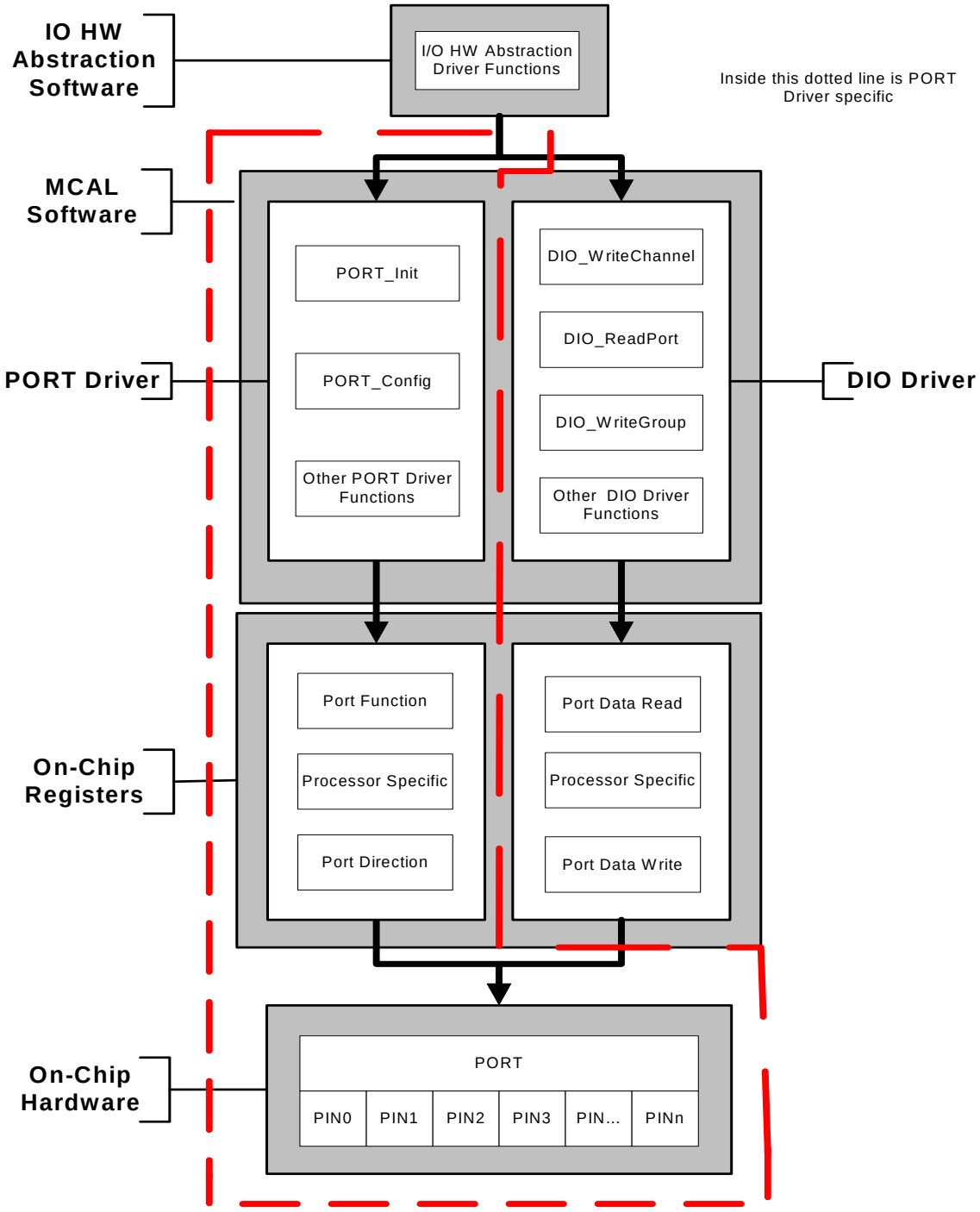
Port initialisation data shall be written to each port as efficiently as possible.

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_Init
PORT_Config
Other PORT Driver Functions
DIO_WriteChannel
DIO_ReadPort
DIO_WriteGroup
Other DIO Driver Functions
PORT Driver
DIO Driver
Port Function
Processor Specific
Port Direction
Port Data Read
Processor Specific
Port Data Write
On-Chip Registers
PORT
PIN0
PIN1
PIN2
PIN3
PIN...
PINn
On-Chip Hardware

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 possible: LOW/HIGH
Physical Level (Output)	Two states possible: LOW/HIGH

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules,
https://svn.autosar.org/repos/10Releases
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture,
https://svn.autosar.org/repos/10Releases
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules,
https://svn.autosar.org/repos/10Releases
AUTOSAR_SRS_General.pdf

[4] Specification of Development Error Tracer,
https://svn.autosar.org/repos/10Releases
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[5] Specification of ECU Configuration,
https://svn.autosar.org/repos/10Releases
AUTOSAR_ECU_Configuration.pdf

[6] Specification of Diagnostic Event Manager (DEM),
https://svn.autosar.org/repos/10Releases
AUTOSAR_SWS_DEM.pdf

[7] Specification of ECU State Manager,
https://svn.autosar.org/repos/10Releases
AUTOSAR_SWS_Ecu_StateManager.pdf

[8] General Requirements on SPAL,
https://svn.autosar.org/repos/10Releases
AUTOSAR_SRS_SPAL_General.pdf

[9] Requirements on PORT driver,
https://svn.autosar.org/repos/10Releases
AUTOSAR_SRS_PORT_Driver.pdf

[10] Specification of Standard Types,
https://svn.autosar.org/repos/10Releases
AUTOSAR_SWS_StandardTypes.pdf

3.2 Related standards and norms

[11]IEC 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 the same Port/Port pin is not being accessed in parallel by different entities in the same system, e.g. 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

PORT083: Other driver modules may be dependent on the PORT driver depending on the available functionality of individual port pins on a 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.

PORT084: The PORT driver is dependent on the OS for masking of interrupts (see PORT075).

PORT107: In development mode the Error-hook function of module DET will be called.

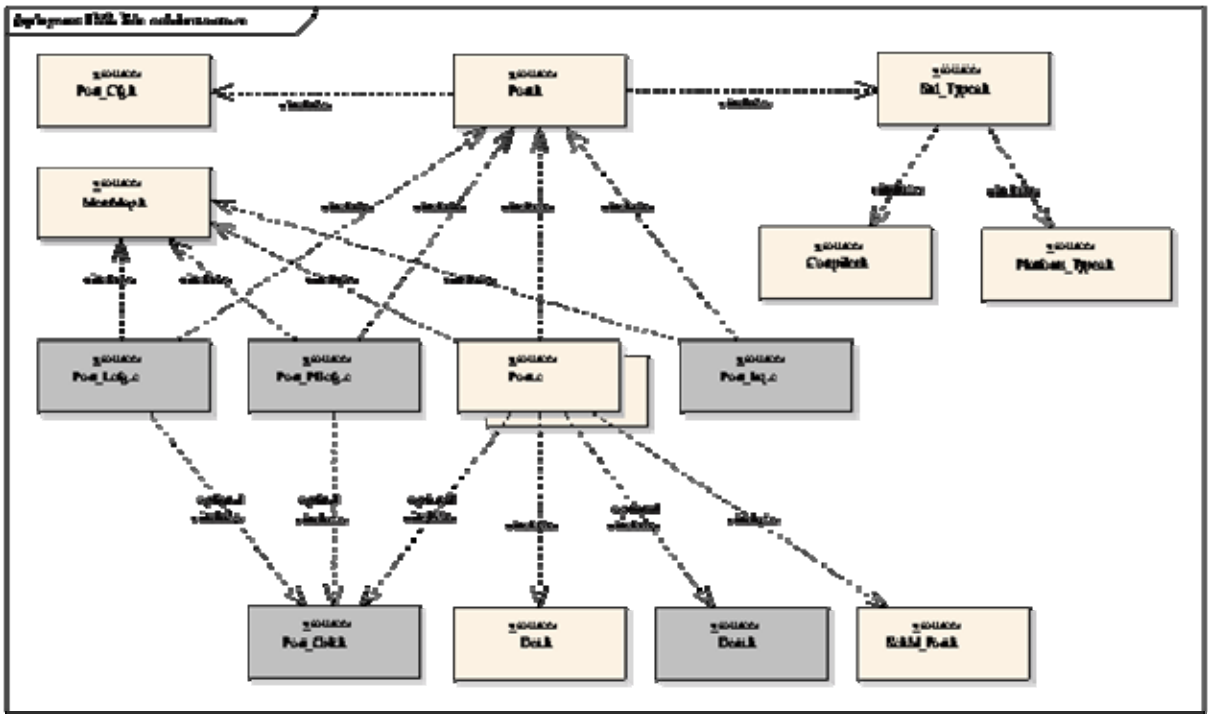
5.1 File structure

5.1.1 Code file structure

PORT108: The code file structure shall not be defined within this specification.

5.1.2 Header file structure

PORT080:The include file structure shall be as follows:



The grey boxes are optional:

- `Port.c` shall include `Port.h` (PORT114)
- `Port_Xcfg.c` where X is a placeholder for ‘L’ or ‘PB’

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

Port.c has access to the Port_Cfg.h via the implicitly include through the Port.h file.

Port_Irq.c shall include Port.h for the function, which shall be called in the interrupt function.

The Type definitions for Port_Lcfg.c and Port_PBcfg.c are located in the file Port_Cfg.h. or Port.h.

The implicitly 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
```

Port_Lcfg.c shall include Port_Cbk.h for a link time configuration, if the call back function is linked to the module via the ROM structure.

Port_PBcfg.c shall include Port_Cbk.h for a post build time configuration, if the call back function is linked to the module via the ROM structure.

Port.c shall include Port_Cbk.h for a pre-compile time configuration

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

PORT109: The module shall include the Dem.h file. 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.

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 links to 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)

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, PORT059, PORT086
[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: This module shall initialize the whole port structure of the microcontroller. The order in which the ports and port pins are configured is the task of the configuration tool.

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

PORT003: This function may also be used to initialize the driver software and reinitialize the ports and port pins to another configured state depending on the configuration set passed to this service.

PORT055: 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. If this is the case then the initialization service shall set the port pin output latch to a default level (defined during configuration) before setting the port pin direction to output, thus ensuring that the default level is immediately output on the port pin when it is set to an output port pin. For 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 initialization service shall ensure that the default level is set before switching the functionality of the port pin from input to output.

7.1.2 Requirements

7.1.2.1 Configuration of Port Pin Properties

PORT004: The PORT driver 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: Additional configurations for the MCU port/port pins provided by the PORT driver are:

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

PORT081: A number of optional configurations for the MCU ports and port pins (if supported by hardware) are:

- 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 shall not provide for the configuration of Level inversion. The default value shall be set (not inverted). The IO Hardware Abstraction layer shall carry out level Inversion.

7.1.2.2 Switch port pin direction

PORT065: The PORT driver shall allow the user to change the direction of port pins during runtime. This service shall only be available for those port pins that are configured as changeable using the configuration tool. If the MCU port control hardware provides an output latch for setting the output level on a port pin, this service 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: This 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 module configuration.

7.1.2.5 Configuration of symbolic names

PORT006: 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: Symbolic names shall be defined in the file `Port_Cfg.h` and published through file `Port.h`.

7.1.2.6 Atomicity of port access

PORT075: The PORT driver shall provide atomic access to all ports and port pins by the use of either atomic instruction or the masking of interrupts using OS interrupt masking functions.

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

7.2.1 Background and Rationale

The error classification depends on the time of error occurrence according to product life cycle:

- Development Errors
  Those errors shall be detected and fixed during development phase. In most cases, those errors are software errors. The detection of errors that shall only occur during development can be switched off for production code (by static configuration namely pre-processor switches).

- Production
  Those errors are hardware errors and software exceptions that cannot be avoided and are also expected to occur in production code.

7.2.2 Requirements

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`.
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`

7.3 Error detection

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

PORT101: If the PORT_DEV_ERROR_DETECT 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.

PORT105: If development error detection is enabled, the parameter ConfigPtr shall be checked for being within the allowed range. The error shall be reported to the Development Error Tracer with error value PORT_E_PARAM_CONFIG.

7.4 Error notification

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

8 API specification

8.1 Imported types

8.1.1 Standard types

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

- Std_Types.h
-
- Std_VersionInfoType

8.1.2 Other Module types

- No types from other modules are imported.

8.2 Type definitions

8.2.1 Port_ConfigType

Type:	structure
Range:	Hardware Dependent Structure — The contents of the initialization data structure are specific to the microcontroller.
Description:	**PORT073**: This is the type of the external data structure containing the initialization data for the PORT driver. **PORT018**: The user shall use the symbolic names defined in the configuration tool. **PORT074**: The configuration of each port pin is MCU specific. Therefore a complete list of different configurations is not possible to include in this driver. **PORT072**: A list of possible port configurations 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 where the port pin is to be used for DIO. • pin level init value (see PORT055) – this port pin configuration is mandatory where the port pin is used for DIO. • pin direction changeable during runtime (yes/no) – this port pin 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

Type:	uint8…uint32	
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:	**PORT013**: Symbolic name of a Port pin. **PORT015**: The user shall use the symbolic names provided by the configuration tool.	

8.2.3 Port_PinDirectionType

Type:	enumeration	
Range:	PORT_PIN_IN:	Sets port pin as input.
	PORT_PIN_OUT:	Sets port pin as output.
Description:	**PORT046**: These are the possible directions a port pin can have.	

8.2.4 Port_PinModeType

Type:	uint8…uint32	
Range:	Implementation specific	As several port pin modes shall be configurable on one pin, the range shall be determined by the implementation.
Description:	PORT124: A number of port pin modes shall be configurable on a port pin. This type shall be used with the function call Port_SetPinMode (section 8.3.5).	

8.3 Function definitions

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

8.3.1 Port_Init

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 (out):	None	--
Return value:	None	--
Description:	**PORT041**: This initialisation function shall be called to initialise ALL ports and port	

	pins with the configuration set pointed to by ConfigPtr. **PORT078**: This function should be called first in order to initialise the port for use. If not called first then no operation can occur on the MCU ports and port pins. **PORT042**: This function shall initialise all configured resources. **PORT113**: The following rules regarding initialisation of controller registers shall apply to the PORT driver implementation: 1. If the hardware allows for only one usage of the register, the driver modules 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. 1. All other registers shall be initialised by the start-up code. **PORT043**: The initialisation shall avoid glitches and spikes on the affected port pins. **PORT071**: This function shall also be called after a reset, in order to reconfigure the ports and port pins of the MCU. Error Detection: PORT087
Caveats:	This service should not be called during a running operation
Configuration:	All port pins and their functions, and alternate functions shall be configured by the configuration tool.

8.3.2 Port_SetPinDirection

Service name:	Port_SetPinDirection	
Syntax:	`void Port_SetPinDirection ( Port_PinType Pin, Port_PinDirectionType Direction )`	
Service ID [hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Re-entrant (Limited: see PORT054)	
Parameters (in):	`Pin`	Port Pin ID number
	`Direction`	Port Pin Direction
Parameters (out):	None	--
Return value:	None	--
Description:	**PORT063**: Sets the port pin direction during runtime. **PORT059**: The PORT driver shall only allow to change the direction of those port pins configured as changeable. Error Detection: PORT087	

Caveats:	**PORT054**: This function shall be re-entrant if accessing different pins, independent of port. **PORT086**: This function is only available if the runtime parameter PORT_PIN_DIRECTION_CHANGEABLE is set to TRUE. If set to FALSE, this function is not applicable . (see Section 10.2.2)
Configuration:	All ports and port pins shall be configured by the configuration tool. Reference PORT117

8.3.3 Port_RefreshPortDirection

Service name:	Port_RefreshPortDirection	

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

Service ID:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non re-entrant	
Parameters (in):	None	--
Parameters (out):	None	--
Return value:	--	--
	--	--
Description:	**PORT060**: This service shall refresh the direction of all configured ports to the configured direction. **PORT061**: The PORT driver shall exclude those port pins from refreshing that are configured as 'pin direction changeable during runtime'.	
Caveats:	--	
Configuration:	**PORT025**: names for each configured port pin shall be provided by the configuration tool.	

8.3.4 Port_GetVersionInfo

Service name:	Port_ GetVersionInfo	

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

Service ID [hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	non re-entrant	
Parameters (in):	none	--
Parameters (out):	`versioninfo`	Pointer to where to store the version information of this module.
Return value:	none	--
Description:	**PORT102**: This service returns the version information of this module. The	

	version information includes: - Module Id - Vendor Id - Vendor specific version numbers (BSW00407). **PORT103**: This function shall be pre compile time configurable `On/Off` by the configuration parameter: `PORT_VERSION_INFO_API` Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.
Caveats:	--
Configuration:	This service is only available if enabled by the pre-processor switch `PORT_VERSION_INFO_API`.

8.3.5 Port_SetPinMode

Service name:	Port_SetPinMode	

Syntax:	`void Port_SetPinMode`
	`(`
	`  Port_PinType  Pin,`
	`  Port_PinModeType  Mode`
	`)`	

Service ID [hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Re-entrant	
Parameters (in):	`Pin`	Port Pin ID number
	`Mode`	New Port Pin mode to be set on port pin.
Parameters (out):	None	--
Return value:	None	--
Description:	PORT125: Sets the port pin mode during runtime. Error Detection: PORT087	
Caveats:	-	
Configuration:	All ports and port pins shall be configured by the configuration tool. Reference PORT117	

8.4 Call-back notifications

PORT104: 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.

API function	`Det_ReportError`
Module	Development Error Tracer (DET, see section 3.1)
Description	Development error notification
Configuration parameter (description see chapter 10)	PORT_DEV_ERROR_DETECT

API function	`Dem_ReportErrorStatus`
Module	Diagnostic Event Manager (DEM, see section 3.1)
Description	Reporting of production relevant error status

API function	`Dem_ReportErrorEvent`
Module	Diagnostic Event Manager (DEM, see section 3.1)
Description	Reporting of production relevant error events

8.6.3 Configurable Interfaces

None

8.7 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). Detected errors shall be reported to the Development Error Tracer (see PORT100).

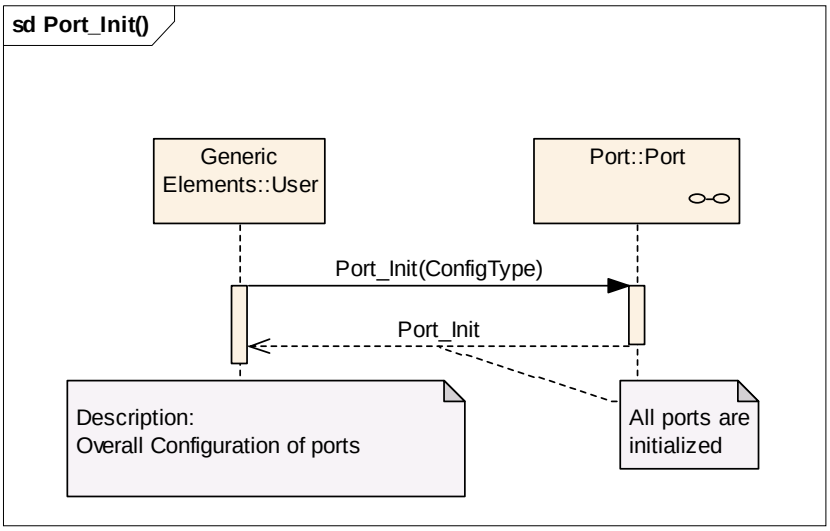
PORT077: The function parameters are checked in the order in which they are passed. For the function Port_SetPinDirection, the first parameter to be passed is the pin ID. This parameter shall identify the relevant port pin of the MCU's port. The second parameter passed, corresponds to the direction to change on the port pin.

PORT087: If development error detection is enabled and an error is detected, the desired functionality shall be skipped and the requested service is left without any action.

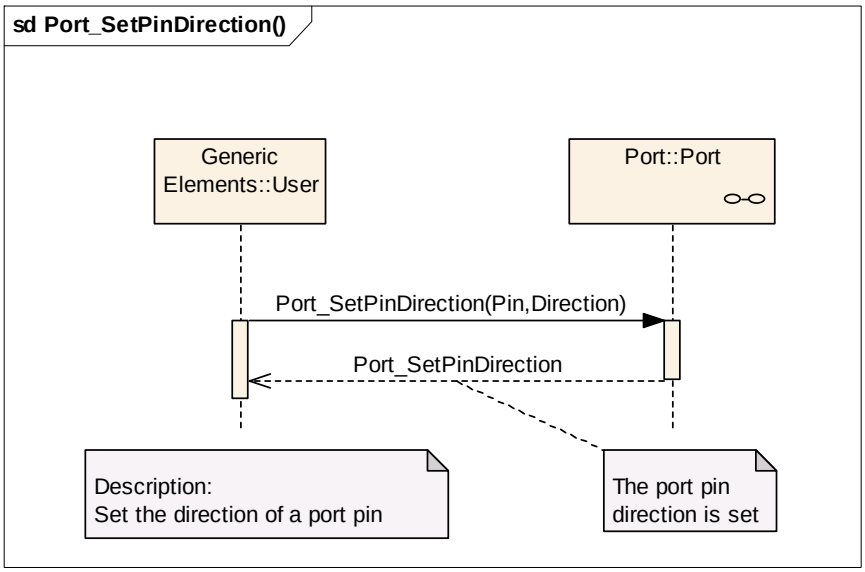
Function	***Error Condition***	***Related error value***
Port_SetPinDirection	Incorrect Port Pin ID passed	PORT_E_PARAM_PIN
	Port Pin not configured as changeable	PORT_E_DIRECTION_UNCHANGEABLE
Port_Init	Port_Init service called with wrong parameter.	PORT_E_PARAM_CONFIG
Port_SetPinMode	Incorrect Port Pin ID passed	PORT_E_PARAM_PIN
	Port Pin Mode passed not valid	PORT_E_PARAM_INVALID_MODE

9 Sequence diagrams

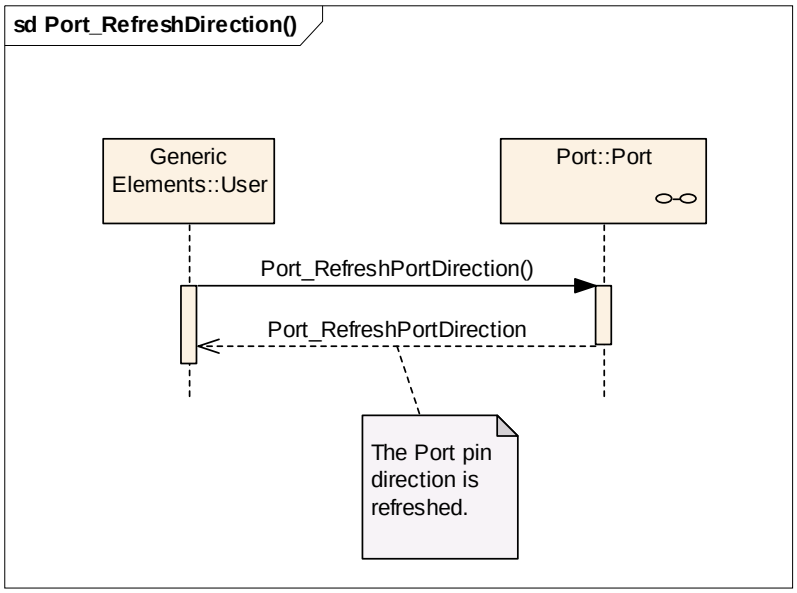
9.1 Overall Configuration of Ports



9.2 Set the direction of a Port Pin



9.3 Refresh the direction 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

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.

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.

PORT121: The initialization function of this module shall always have a pointer as a parameter, even though for VariantPC no configuration set shall be given. Instead a NULL pointer shall be passed to the initialization function. This means that, in contradiction to BSW00414 only one interface for initialization shall be implemented and it shall not depend on the modules configuration, which interface the calling software module shall use.

10.2.2 Port Driver Configuration

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

SWS Item	**PORT117**:
Container Name	PORT General configuration
Description	This container contains the configuration (parameters) of the PORT driver.
Configuration Parameters	

Name	PORT_DEV_ERROR_DETECT		
Description	Switches the Development Error Detection and Notification ON or OFF.		
Type	#define		
Unit	--		
Range	ON	Enabled	
	OFF	Disabled	
Configuration Class	***Pre-compile***	X	All Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PORT_VERSION_INFO_API

Description	Pre-processor switch to enable / disable the API to read out the modules version information.		
Type or Unit	#define		
Unit	--		
Range	ON	Version info API enabled.	
	OFF	Version info API disabled,	
Configuration Class	***Pre-compile***	X	All Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PORT_SET_PIN_MODE_API		
Description	Pre-processor switch to enable / disable the use of the function Port_SetPinMode().		
Type or Unit	#define		
Range	TRUE	Enabled - Function Port_SetPinMode() is available	
	FALSE	Disabled - Function Port_SetPinMode() is not available.	
Configuration Class	***Pre-compile***	X	All Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PORT_PIN_DIRECTION_CHANGES_ALLOWED		
Description	Pre-processor switch to enable / disable the use of the function Port_SetPinDirection().		
Type or Unit	#define		
Range	TRUE	Enabled - Function Port_SetPinDirection() is available	
	FALSE	Disabled - Function Port_SetPinDirection() is not available.	
Configuration Class	***Pre-compile***	X	All Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PortPin	1..*	--

10.2.3 Port Pin Container

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

Name	PORT_PIN_INITIAL_MODE		
Description	Port pin mode from mode list for use with Port_Init() function.		
Type	Implementation Specific		
Unit	--		
Range	--	--	
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	--	--
	Post Build	M	VariantPB
Scope	Module		
Dependency	None		

Name	PORT_PIN_MODE		
Description	Port pin mode from mode list for use with Port_SetPinMode() function.		
Type	Implementation Specific		
Unit	--		
Range	--	--	
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	--	--
	Post Build	M	VariantPB
Scope	Module		
Dependency	None		

Name	PORT_PIN_DIRECTION		
Description	Port Pin Direction from Port Pin Direction list.		
Type	Port_PinDirectionType		
Unit	--		
Range	IN	Port Pin direction set as input	
	OUT	Port Pin direction set as output	
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	--	--
	Post Build	M	VariantPB
Scope	Module		
Dependency	None		

Name	PORT_PIN_LEVEL_VALUE		
Description	Port Pin Level value from Port pin list.		
Type	Implementation Specific		
Unit	--		
Range	HIGH	Port Pin level is High	
	LOW	Port Pin level is LOW	
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	--	--
	Post Build	M	VariantPB
Scope	Module		
Dependency	None		

Name	PORT_PIN_DIRECTION_CHANGEABLE		
Description	Parameter to indicate if the direction is changeable on a port pin during runtime.		
Type	Implementation Specific		
Unit	--		
Range	TRUE	Port Pin direction changeable enabled	
	FALSE	Port Pin direction changeable disabled.	
Configuration Class	***Pre-compile***	X	VariantPC
	Link time	--	--
	Post Build	M	VariantPB
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

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.

SWS Item	**PORT106**:	
Information elements		
Information element name	***Type / Range***	***Information element description***
PORT_VENDOR_ID	#define /uint16	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.
PORT_MODULE_ID	#define /uint8	Module ID of this module from Module List
PORT_AR_MAJOR_VERSION	#define /uint8	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
PORT_AR_MINOR_VERSION	#define /uint8	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
PORT_AR_PATCH_VERSION	#define /uint8	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
PORT_SW_MAJOR_VERSION	#define /uint8	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
PORT_SW_MINOR_VERSION	#define /uint8	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
PORT_SW_PATCH_VERSION	#define /uint8	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

11 Changes to Release 1

11.1 Deleted SWS Items

SWS Item	*Rationale*
PORT032	Removed due to template change for Section 10.
PORT034	Removed due to template change for Section 10.
PORT035	Removed due to template change for Section 10.
PORT036	Removed due to template change for Section 10.

11.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
PORT070	PORT114	New template structure.
PORT026	PORT100	New template structure

11.3 Changed SWS Items

SWS Item	*Rationale*
PORT076	Reworded text following Bugzilla request.
PORT054	Reworded text following Bugzilla request.

11.4 Added SWS Items

SWS Item	*Rationale*
PORT100	Inclusion of switch for detection of development errors.
PORT101	Inclusion of PORT_DEV_ERROR_DETECT switch
PORT102	New API service Port_GetVersionInfo()
PORT103	Pre-compile switch for version info, PORT_VERSION_INFO_API
PORT104	New requirement of this template.
PORT105	Parameter check for NULL pointer.
PORT106	New requirement of this template.
PORT107	New requirement of this template.
PORT108	New requirement of this template.
PORT109	New requirement of this template.
PORT113	New requirement [BSW12461] introduced in SPAL_General.SRS [3]
PORT114	New requirement of this template.
PORT115	New requirement of this template.
PORT116	New requirement of this template.
PORT117	New requirement of this template.
PORT118	New requirement of this template.
PORT119	New requirement of this template.
PORT120	New requirement [BSW12462] introduced in SPAL_General.SRS [3]
PORT121	Additional requirement introduced in line with BSW00414 (SPAL decision, 42nd meeting, minutes day2, issue 5)