MICROSAR PORT

Technical Reference

MCAL Emulation in VTT

Version 1.1.0

Authors	Peter Lang, Christian Leder
Status	Released

Document Information

History

Author	Date	Version	Remarks
Peter Lang	2013-09-07	1.00.00	Creation of document
Christian Leder	2015-02-17	1.01.00	> Global renaming of Vip to Vtt > Usage of template 5.11.0 for the Technical reference

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_PortDriver.pdf	V3.2.0
[2]	AUTOSAR	AUTOSAR_SWS_DevelopmentErrorTracer.pdf	V3.2.0
[3]	AUTOSAR	AUTOSAR_SWS_DiagnosticEventManager.pdf	V4.2.0
[4]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	V1.6.0

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
1.0.x	Initial version of the Vip PORT driver
2.0.x	Global renaming of Vip to Vtt

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module PORT as specified in [1].

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	PORT_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	PORT_MODULE_ID	124 decimal (according to ref. [4])

* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

The PORT driver centralizes the configuration and initialization of ports and pins of a microcontroller which are used by other driver modules like SPI or DIO. In case the MCAL is emulated in the VTT framework, the PORT driver does not provide any functionality since there are no physical ports or port pins emulated.

This module is provided only for compatibility reasons.

2.1 Architecture Overview

The following figure shows where the PORT is located in the AUTOSAR architecture.

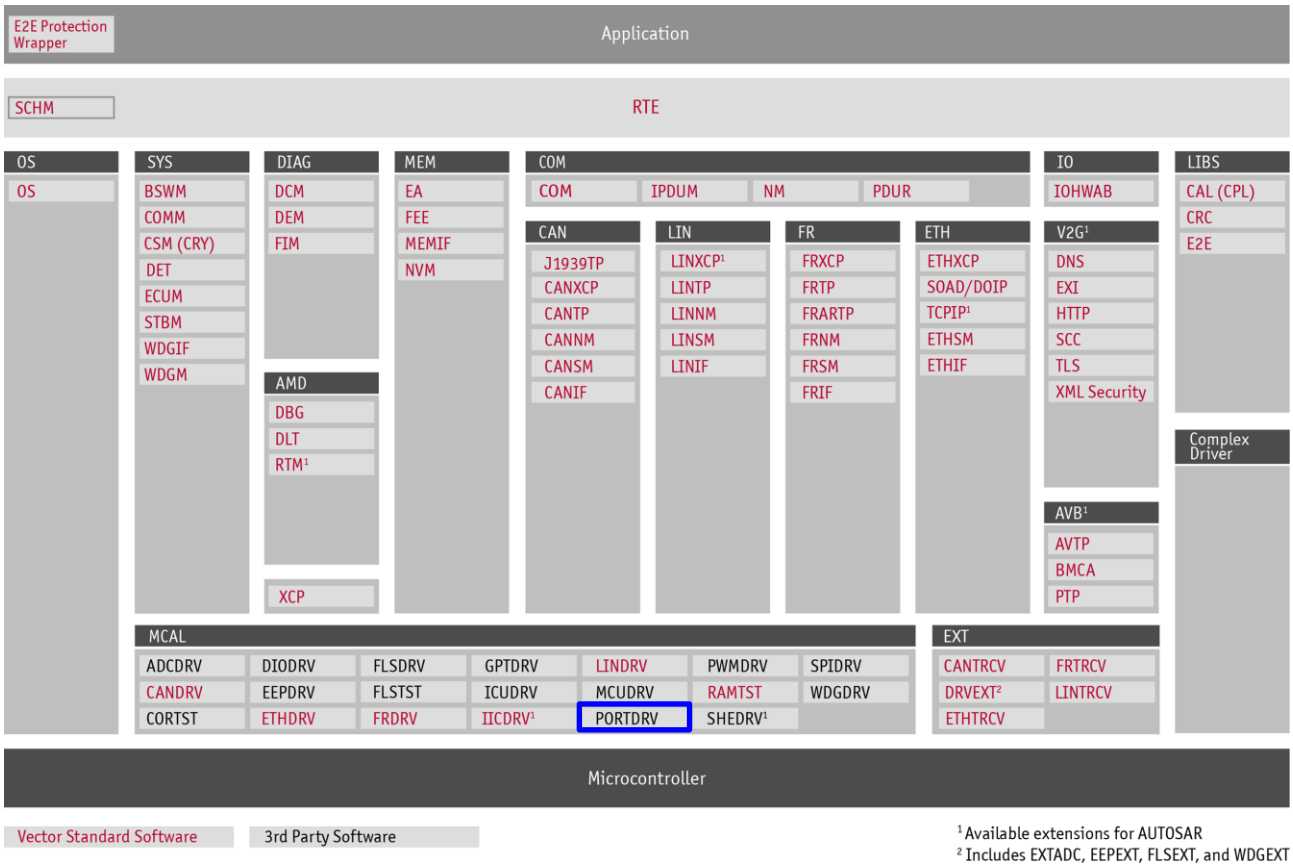

Figure 2-1 AUTOSAR 4.x Architecture Overview

The next figure shows the interfaces to adjacent modules of the PORT. These interfaces are described in chapter 5.

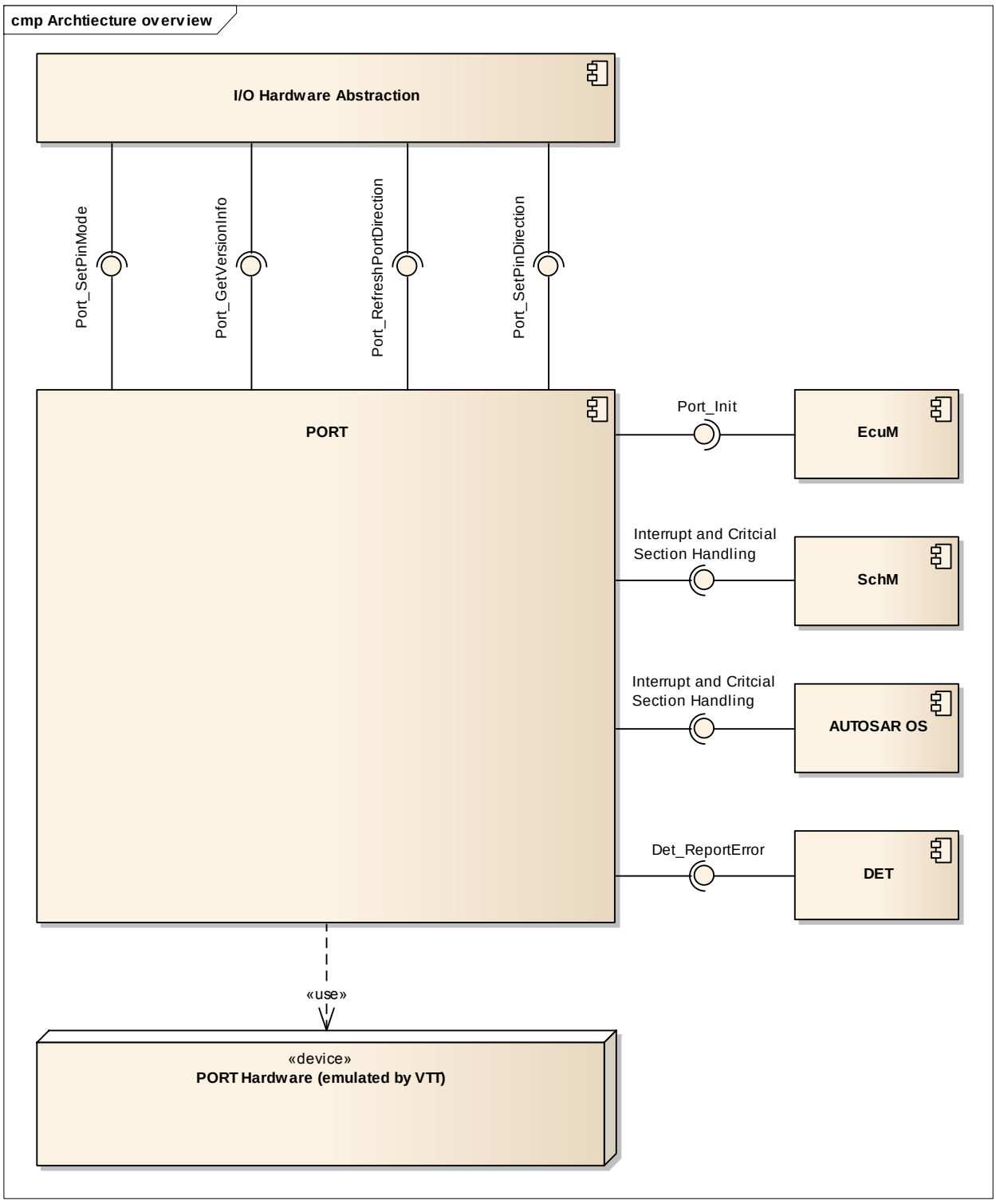

Figure 2-2 Interfaces to adjacent modules of the PORT

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for the PORT.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported AUTOSAR standard conform features

> Table 3-2 Not supported AUTOSAR standard conform features

Vector Informatik provides further PORT functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Configure all port pins (without functionality)
Initialize the configuration (without functionality)
Set pins to an initial default value (without functionality)
Refresh the direction configuration (without functionality)
Switch the port pin configuration (out / in) during runtime (without functionality)

Table 3-1 Supported AUTOSAR standard conform features

3.1.1 Deviations

The following features specified in [1] are not supported:

Not Supported AUTOSAR Standard Conform Features
API function `Port_SetPinMode()` does not provide development error checks `PORT_E_MODE_UNCHANGEABLE` and `PORT_E_PARAM_INVALID_MODE`.

Table 3-2 Not supported AUTOSAR standard conform features

3.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
None

Table 3-3 Features provided beyond the AUTOSAR standard

3.1.3 Limitations

3.1.3.1 Diagnostic Event Manager

Due to the fact that the PORT is emulated, reporting of hardware errors to the DEM is not supported. Because of compatibility reasons, the DEM has to be configured in DaVinci Configurator.

Emulation

This driver is an emulation of an PORT module.

Caution

Be careful using while loops in order to poll any status.

The user has to ensure, that the application does not block the emulation. So, within every while loop the following function call has to be called:

```
while(ANY_STATUS == temp_status)
{
  Schedule();
}
```

Use the function call Schedule() which is available once the header file of the module PORT is included.

3.2 Initialization

The PORT module is being initialized by calling `Port_Init(&PortConfigSet)`. All global variables are initialized by calling `Port_InitMemory()`. So, `Port_InitMemory()` has to be called prior to `Port_Init()`.

3.3 States

The PORT module does not implement a state machine.

3.4 Main Functions

The PORT module does not provide any cyclic main functions.

3.5 Error Handling

3.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `PORT_DEV_ERROR_DETECT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported PORT ID is 124.

The reported service IDs identify the services which are described in 5.2. The following table presents the service IDs and the related services:

Service ID	Service
0x00	Port_Init
0x01	Port_SetPinDirection
0x02	Port_RefreshPortDirection
0x03	Port_GetVersionInfo
0x04	Port_SetPinMode

Table 3-4 Service IDs

The errors reported to DET are described in the following table:

Error Code		Description
0x0A	PORT_E_PARAM_PIN	Invalid Port Pin ID requested
0x0B	PORT_E_DIRECTION_UNCHANGEABLE	Port_SetPinDirection called for a pin which direction is not changeable
0x0C	PORT_E_PARAM_CONFIG	Port_Init called with wrong parameter
0x0F	PORT_E_UNINIT	API service called while the driver is not initialized
0x20	PORT_E_PARAM_VINFO	Port_GetVersionInfo is called with wrong parameter

Table 3-5 Errors reported to DET

3.5.1.1 Parameter Checking

AUTOSAR requires that API functions check the validity of their parameters. The checks in Table 3-6 are internal parameter checks of the API functions. These checks are for development error reporting and can be en-/disabled.

The following table shows which parameter checks are performed on which services:

Service \ Check	PORT_E_PARAM_CONFIG	PORT_E_PARAM_PIN	PORT_E_UNINIT	[PORT_E_DIRECTION_UNCHANGEABLE	[PORT_E_PARAM_VINFO
Port_Init	■				
Port_SetPinDirection		■	■	■	
Port_RefreshPortDirection			■		
Port_GetVersionInfo					■
Port_SetPinMode		■	■		

Table 3-6 Development Error Reporting: Assignment of checks to services

3.5.2 Production Code Error Reporting

Info
Production errors are not supported in this emulation.

4 Integration

This chapter gives necessary information for the integration of the MICROSAR PORT into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the PORT contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Description
Port.h	The module header defines the interface of the PORT. This file must be included by upper layer software components
Port.c	This C-source contains the implementation of the module's functionalities
DrvPort_VttCanoe01Asr.jar	This jar-file contains the generator and the validator for the DaVinci Configurator
VTTPort_bswmd.arxml	Basic Software Module Description according to AUTOSAR for VTT Emulation
Port_bswmd.arxml	Optional Basic Software Module Description. Placeholder for real target (semiconductor manufacturer) in VTT only use case

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator.

File Name	Description
Port_Cfg.h	The configuration-header contains the static configuration part of this module
Port_PBcfg.c	The configuration-source contains the object independent part of the runtime configuration

Table 4-2 Generated files

4.2 Include Structure

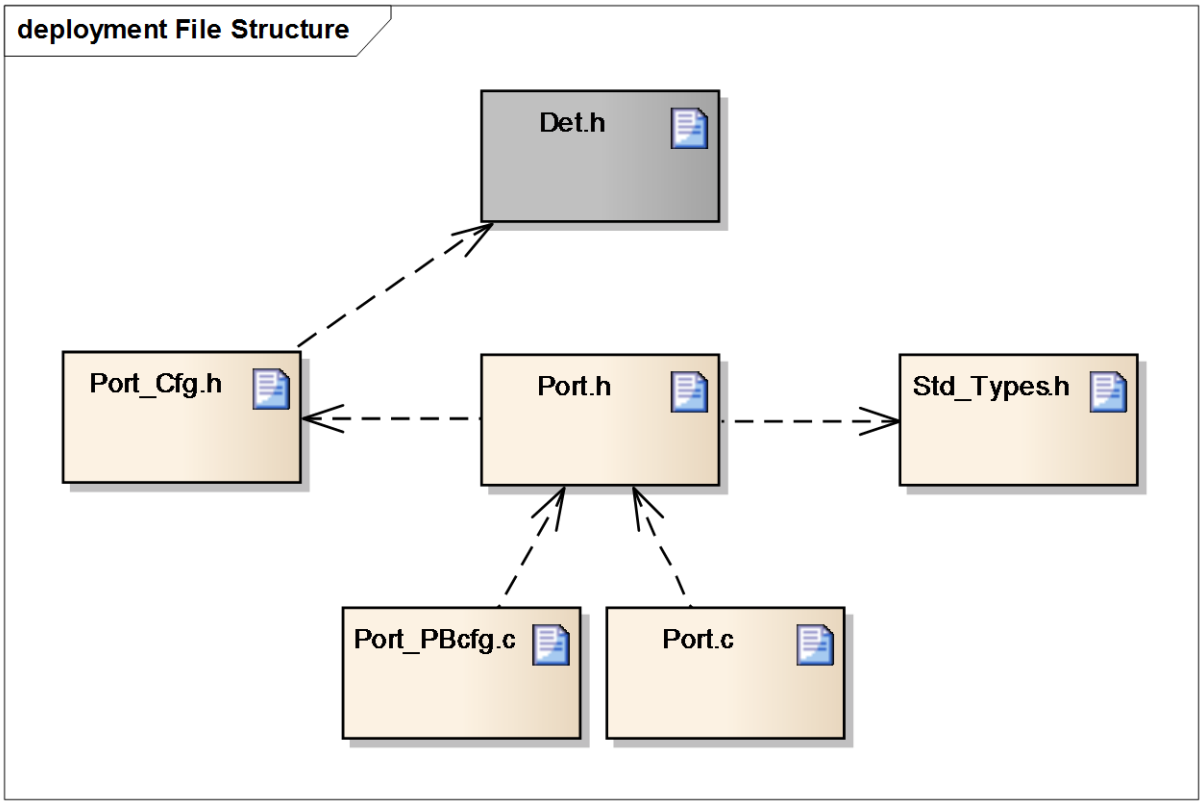

Figure 4-1 Include Structure

4.3 Dependencies on SW modules

4.3.1 AUTOSAR OS (optional)

This module depends on the AUTOSAR OS, enabling or disabling global interrupts.

4.3.2 DET (optional)

The MCU depends on the DET. This module can be used in Development Mode. The PORT module reports all development errors to DET. The usage of the DET can be disabled by the switch “Development mode” in DaVinci Configurator.

4.3.3 EcuM

The EcuM cares for the initialization of the module PORT.

4.3.4 SchM (Optional)

Beside the AUTOSAR OS the Schedule Manager provides functions that module PORT calls at begin and end of critical sections.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

The types defined by the PORT are described in this chapter.

Type Name	C-Type	Description	Value Range
Port_PinDirectionType	enum	These are the possible directions a port pin can have.	`PORT_PIN_IN` `PORT_PIN_OUT`
Port_PinType	uint8	This is the numeric representative of the symbolic name of a port pin.	0..<number of pins> Covers all available port pins.
Port_PinModeType	uint8	Pin mode type is not implemented but provided for compatibility reasons	`0..255`

Table 5-1 Type definitions

5.2 Services provided by PORT

5.2.1 Port_InitMemory

Prototype	
`void` **`Port_InitMemory`** `(void)`	
Parameter	
-	-
Return code	
-	-
Functional Description	
This service initializes the global variables in case the startup code does not work	
Particularities and Limitations	
> This function is synchronous. > This function is non reentrant. > Module must not be initialized.	
Expected Caller Context	
> Called during startup	

Table 5-2 Port_InitMemory

5.2.2 Port_Init

Prototype	
`void` **`Port_Init`** `(P2CONST(Port_ConfigType, AUTOMATIC, PORT_APPL_CONST) ConfigPtr)`	
Parameter	
`ConfigPtr`	Pointer to configuration set
Return code	
-	-
Functional Description	
This function has to be called to initialize all ports and port pins with the configuration set pointed to by `ConfigPtr` (even if there are no ports and port pins emulated in CANoe emulation). This function has to be called first in order to initialize the PORT for use. This function also has to be called after a reset, in order to reconfigure the ports and port pins of the microcontroller.	
Particularities and Limitations	
> This function is synchronous. > This function is non reentrant. > Module must not be initialized.	
Expected Caller Context	
> Expected to be called in application context, mostly during initialization phase.	

Table 5-3 Port_Init

5.2.3 Port_SetPinDirection

Prototype	
`void` **`Port_SetPinDirection`** `(Port_PinType Pin, Port_PinDirectionType Direction)`	
Parameter	
`Pin`	Symbolic Pin name
`Direction`	Port Pin direction
Return code	
-	-
Functional Description	
This function sets the direction of port pins during runtime. In this emulation, it has no functionality (except checking development errors) and is only provided for compatibility reasons.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different port pins.	
Expected Caller Context	
> Expected to be called in application context.	

Table 5-4 Port_SetPinDirection

5.2.4 Port_RefreshPortDirection

Prototype	
`void Port_RefreshPortDirection (void)`	
Parameter	
-	-
Return code	
-	-
Functional Description	
This function sets the direction of all port pins that are not configured as “pin direction changeable during runtime” to the initial direction. In this emulation, it has no functionality (except checking development errors) and is only provided for compatibility reasons.	
Particularities and Limitations	
> This function is synchronous. > This function is non reentrant.	
Expected Caller Context	
> Expected to be called in application context.	

Table 5-5 Port_RefreshPortDirection

5.2.5 Port_SetPinMode

Prototype	
`void Port_SetPinMode (Port_PinType Pin, Port_PinModeType Mode)`	
Parameter	
`Pin`	Symbolic Pin name
`Mode`	Port Pin mode
Return code	
-	-
Functional Description	
Sets the mode of the specified ‘`Pin`‘ to operation mode ‘`Mode`‘ In this emulation, it has no functionality (except checking development errors) and is only provided for compatibility reasons.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different port pins.	
Expected Caller Context	
> Expected to be called in application context.	

Table 5-6 Port_SetPinMode

5.2.6 Port_GetVersionInfo

Prototype
`void Port_GetVersionInfo ( P2VAR(Std_VersionInfoType, AUTOMATIC, PORT_APPL_DATA) versioninfo )`
Parameter
`versioninfo` — Pointer where to store the version information of this module.
Return code
- — -
Functional Description
This function returns the version information of the module. The version information includes: > Module Id > Vendor Id > Software version numbers
Particularities and Limitations
> This function is synchronous. > This function is reentrant.
Expected Caller Context
> Expected to be called in application context.

Table 5-7 Port_GetVersionInfo

5.3 Services used by PORT

In the following table services provided by other components, which are used by the PORT are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET	Det_ReportError

Table 5-8 Services used by the PORT

6 Configuration

6.1 Configuration Variants

The PORT supports the configuration variants

> VARIANT-PRE-COMPILE

The configuration classes of the PORT parameters depend on the supported configuration variants. For their definitions please see the VTTPort_bswmd.arxml file.

6.2 Configuration with DaVinci Configurator 5

The PORT module is configured with the help of the configuration tool DaVinci Configurator 5 (CFG5). The definition of each parameter is given in the corresponding BSWMD file.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
CANoe	Tool for simulation and testing of networks and electronic control units.
DaVinci Configurator	Configuration and generation tool for MICROSAR components

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
BSWMD	Basic Software Module Description
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DIO	Digital Input Output
ECU	Electronic Control Unit
EcuM	ECU State Manager
IoHwAb	BSW Module I/O Hardware Abstraction (Connection to RTE)
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
RTE	Runtime Environment
SchM	BSW Module Scheduler
SPI	Serial Peripheral Interface
SWS	Software Specification
VTT	vVIRTUALtarget

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com