MICROSAR IOHWAB

Technical Reference

Version 2.02.02

Authors	Christoph Ederer
Status	Released

1 Document Information

1.1 History

Author	Date	Version	Remarks
Christian Marchl	2007-02-09	1.00.00	Initial version
Christian Marchl	2007-08-09	1.01.00	Typos corrected; Added description for component name field
Christian Marchl	2007-12-13	1.01.01	Version adapted according to new version scheme
Christoph Ederer	2008-05-21	2.00.00	Transfer of the document to new Technical Reference template; Adapted descriptions and screenshots to new software version
Christoph Ederer	2008-07-11	2.00.01	Update of document due to changes in DCM interface and RTE usage
Christoph Ederer	2009-01-14	2.01.00	Update of the naming of graphical elements in the configuration, Screenshots reworked, DCM subfunctions reworked, Added description of default value in configuration
Christoph Ederer	2009-03-23	2.01.01	Updated development error detection in GUI description, toolchain naming updated, hints added to chapter 4.1.2
Christoph Ederer	2009-07-21	2.02.00	Updated description of the generation process (user blocks, autom. SWC generation), updated AUTOSAR figure, added information on user defined signals
Christoph Ederer	2009-09-25	2.02.01	Reworked description of DCM interface
Christoph Ederer	2010-11-26	2.02.02	- Added chapter 4.4 Critical Sections - GUI description updated - Added information about necessary make process modifications to 4.1.2 (ESCAN00047210)

Table 1-1 History of the document

1.2 Reference Documents

No.	Title	Version
[1]	AUTOSAR_SWS_IO_HWAbstraction.pdf	V2.0.0
[2]	AUTOSAR_SWS_DET.pdf	V2.2.0
[3]	AUTOSAR_BasicSoftwareModules.pdf	V1.2.0

Table 1-2 Reference documents

Please note
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

2 Introduction

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

Supported AUTOSAR Release*:	3	
Supported Configuration Variants:	pre-compile	
Vendor ID:	IOHWAB_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	IOHWAB_MODULE_ID	254 decimal (according to ref. [3])

* For the precise AUTOSAR Release 3.x please see the release specific documentation.

The aim of the IOHWAB is to provide ECU hardware-independent data transition from driver modules up to the Software Components. To fulfill this task the IOHWAB generates a Software Component description from the user configuration. This Software Component description can be imported by DaVinci Developer.

Within the DaVinci modeling tool the IOHWAB is displayed like a normal Software Component (SW-C) with a set of server ports and runnables. Other SW-Cs can access driver data values through these ports.

This release of the IOHWAB currently supports a complete abstraction of the DIO driver, which means access functions will be generated automatically (except the direction switch feature of the Port driver).

For all other drivers the access function has to be coded separately by the user. The IOHWAB supports the user at this task by providing generated functions stubs, which can be filled with code.

2.1 Architecture Overview

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

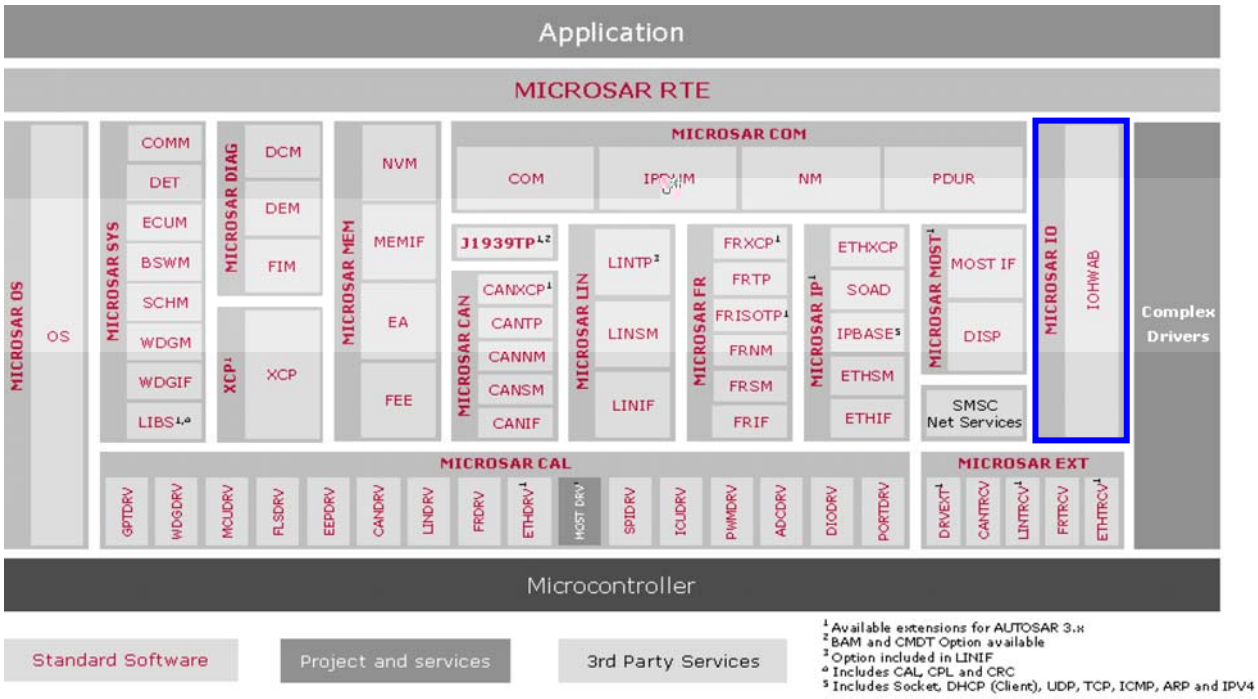


Figure 2-1 AUTOSAR architecture

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

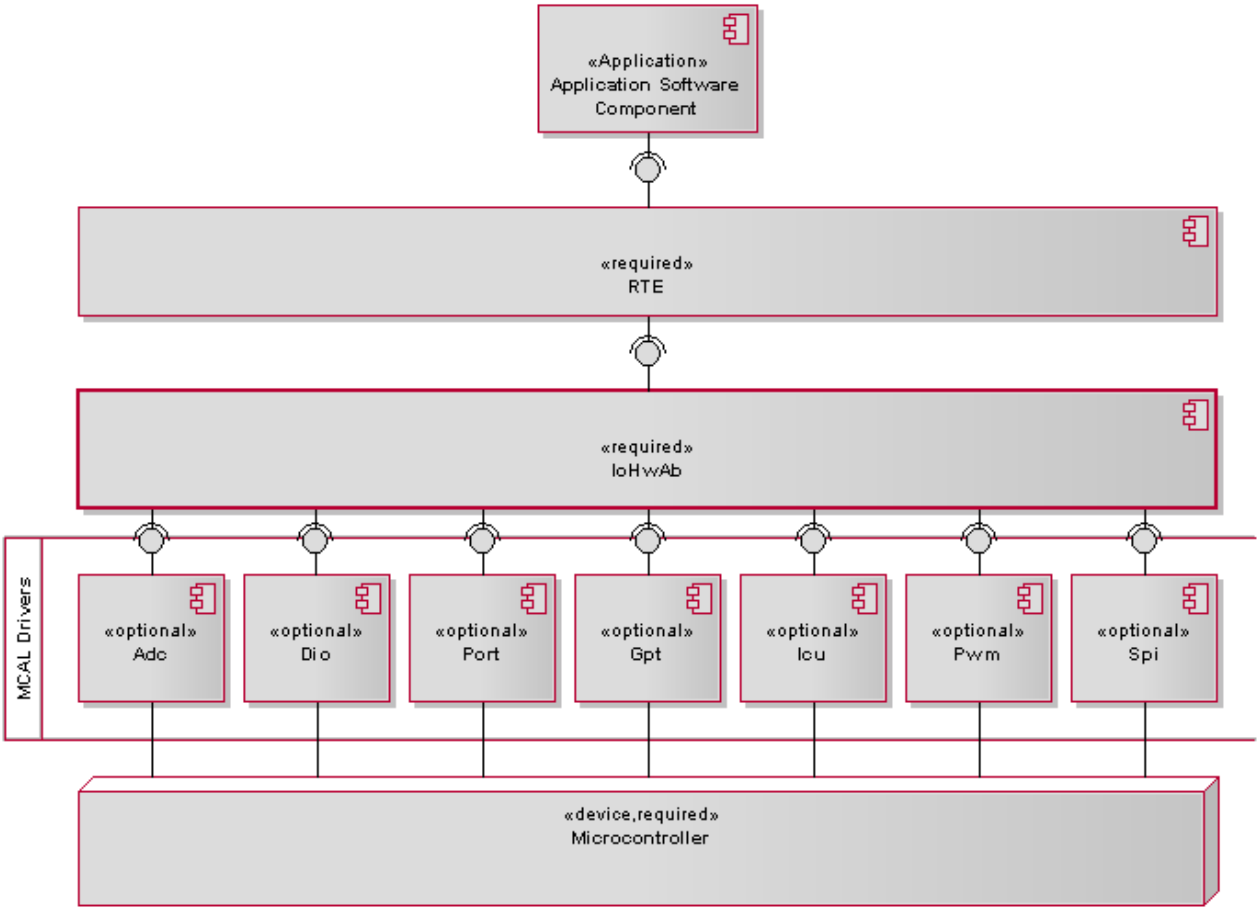


Figure 2-2 Interfaces to adjacent modules of the IOHWAB

Info

The current version of the IOHWAB only supports a complete abstraction of the DIO driver. The other MCAL drivers can be integrated manually via user-defined signals. See 6.1.2.2 ff. for further information about how to create a user-defined signal.

3 Functional Description

3.1 Features

The features listed in this chapter cover the complete functionality specified in [1].

The "supported" and "not supported" features are presented in the following two tables. For further information of not supported features, also see chapter 7.

The following features described in [1] are supported:

Feature
Abstraction of MCAL signals (the IOHWAB generates source code for accessing DIO channels for a specific signal, i.e. OP_GET, OP_SET and OP_DIAG)
Abstraction of user-defined signals (the IOHWAB generates source code templates that can be used for accessing any lower level software)
Provision of an interface to DCM, that allows to freeze or adjust measurement values for diagnostic purposes
Creation of a Software Component Description file that can be imported by DaVinci Developer

Table 3-1 Supported SWS features

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

Feature
Automatic abstraction of other MCAL drivers than DIO (calls to further drivers have to be implemented by the user)

Table 3-2 Not supported SWS features

3.2 Initialization

Initialization is not necessarily needed for the IOHWAB. Therefore, the provided service for initialization can be removed by a pre-compile option during the configuration phase.

Info
It is not intended to initialize drivers in the IOHWAB initialization function. Driver initialization has to be done by the ECUM.

3.3 States

The IOHWAB itself neither has certain module states nor does any handling of states of lower-level drivers. Caring about time flows and states is the user's responsibility.

3.4 Main Functions

The IOHWAB does not provide a main function.

3.5 Error Handling

3.5.1 Development Error Reporting

Detected development errors are by default reported to the DET using the service `Det_ReportError()`, (specified in [2]), if 'Development Mode' and 'Development Error Reporting' are enabled in the configuration tool.

The reported IOHWAB ID is 254.

The reported service IDs identify the services, which are described in 5.3. The following table presents the service IDs and the related services:

Service ID	Service
0x01	IoHwAb_Init()
0x10	IoHwAb_GetVersionInfo()
0x20	MCAL signal OP_GET
0x21	MCAL signal OP_SET
0x2A	MCAL DCM bypass read function
0x30	User-defined input operation (corresponds to OP_GET)
0x31	User-defined output operation (corresponds to OP_SET)
0x3A	User-defined DCM bypass read function

Table 3-3 Mapping of service IDs to services

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

Error Code		Description
0x10	IOHWAB_E_NULL_POINTER	API service called with "NULL pointer" parameter

Table 3-4 Errors reported to DET

3.5.1.1 Parameter Checking

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

Service \ Check	Null pointer check
MCAL signal OP_GET	■
MCAL DCM bypass read function	■
User define input operations	■
IoHwAb_GetVersionInfo()	■
User-defined DCM bypass read function	■

Table 3-5 Development Error Reporting: Assignment of checks to services

3.5.2 Production Code Error Reporting

As production error reporting is typically done on driver level, the IOHWAB does not provide any reporting mechanism. Nonetheless, production error reporting can be added manually to ‘IoHwAb.c’, if necessary.

4 Integration

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

4.1 Scope of Delivery

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

4.1.1 Static Files

File Name	Description
IoHwAb.h	Declares the interface of the MICROSAR component IOHWAB

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool.

File Name	Description
`IoHwAb.c`	Code template that contains the interfaces of the IOHWAB as well as the function bodies of the services for the user-defined operations and the corresponding DCM access functions
`IoHwAb_Dio.c`	Contains the implementation of the services for accessing MCAL signals as well as the corresponding DCM functions
`IoHwAb_Dcm.h`	Contains the prototypes of the DCM access functions
`IoHwAb_Cfg.h`	Contains the static configuration of the IOHWAB
`IoHwAb_Cbk.h`	Code template for prototypes of callback functions
`IoHwAb_Types.h`	Contains all the data types that are used in the IOHWAB (File is only used, if RTE usage is deactivated)

Table 4-2 Generated files

Info
The files ‘IoHwAb.c’ and ‘IoHwAb_Cbk.h’ are code templates, i.e. after generation they are not complete and require user modifications.

Caution

The file ‘IoHwAb.c’ is a generated code template. This file is not part of the AUTOSAR make files provided by the module, because the file may be copied to different locations in an integration package (e.g. a common folder for all editable code templates).

If you use ‘User Defined Operations’, please add the file ‘IoHwAb.c’ to your make process, manually. If the Vector make environment is used, add the file to the variable ‘APP_SOURCE_LST’ in the file ‘Makefile.project.part.defines’.

For successfully using the IOHWAB there are further files needed. These Files can be generated by DaVinci Developer/RTE Generator.

File Name	Description
`Rte_<IoHwAbServiceComponentName>.h`	Contains the prototypes of all the IOHWAB operation services
`Rte_Type.h`	Contains all the data types that are used in the IOHWAB

Table 4-3 DaVinci generated files

4.2 Include Structure

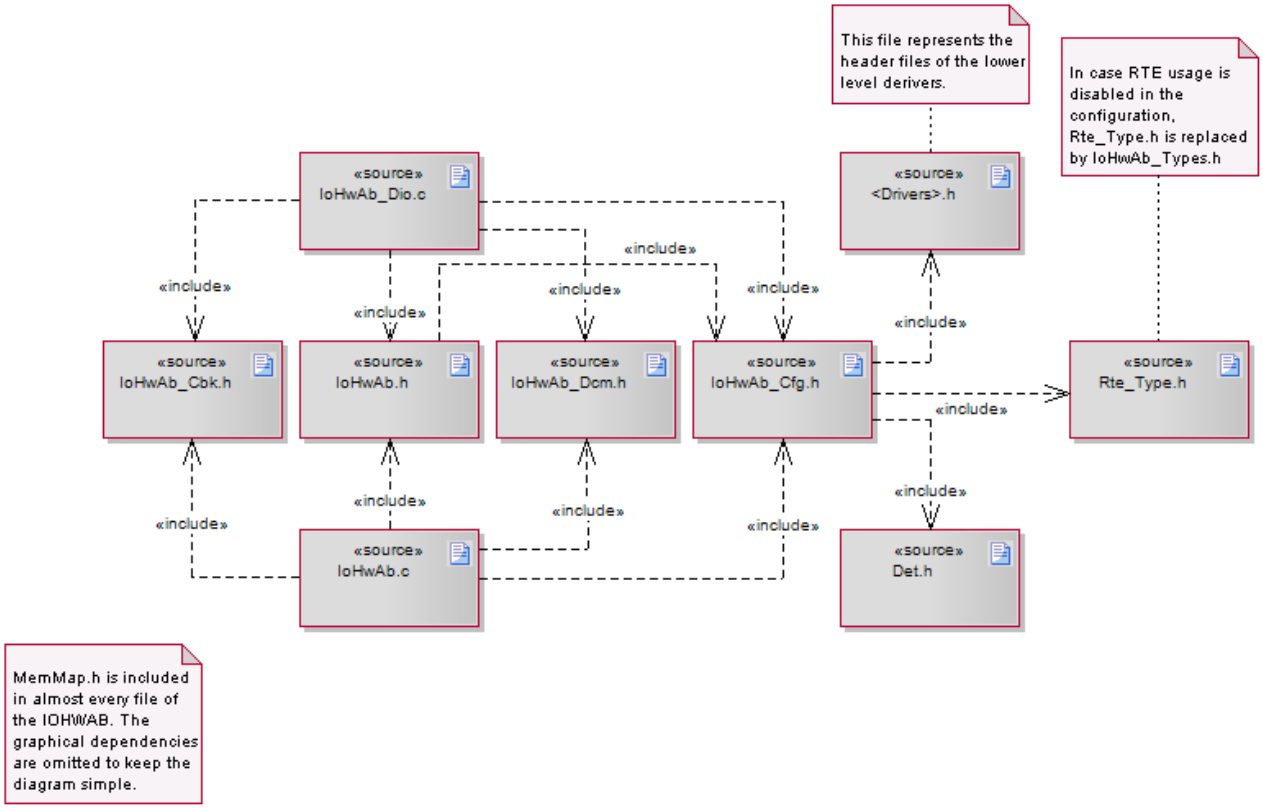


Figure 4-1 Include structure

4.3 Compiler Abstraction and Memory Mapping

The objects (e.g. variables, functions, constants) are declared by compiler independent definitions – the compiler abstraction definitions. Each compiler abstraction definition is assigned to a memory section.

The following table contains the memory section names and the compiler abstraction defines for the IOHWAB and illustrates their assignment among each other.

Memory Mapping Sections / Compiler Abstraction Definitions	IOHWAB_CODE	IOHWAB_VAR	IOHWAB_APPL_DATA	IOHWAB_APPL_CODE	IOHWAB_CONST
IOHWAB_START_SEC_CODE IOHWAB_STOP_SEC_CODE	■		■	■	
IOHWAB_START_SEC_CONST_32BIT IOHWAB_STOP_SEC_CONST_32BIT					■
IOHWAB_START_SEC_VAR_ZERO_INIT_UNSPECIFIED IOHWAB_STOP_SEC_VAR_ZERO_INIT_UNSPECIFIED		■			

Table 4-4 Compiler abstraction and memory mapping

4.4 Critical Sections

The IOHWAB implements the following critical section:

- IOHWAB_EXCLUSIVE_AREA_0: This critical section is used to protect code passages that contain coherent operations. The critical section shall prevent task switches.

4.5 Generated Template Files

Info
This chapter is only applicable if DaVinci Configurator is used for I/O Hardware Abstraction generation.

A generated template file in this document is a file which:

- is generated by the generation tool at every generation process
- the user can modify this template for his needs
- the changes made by the user will not be overwritten at the next generation process

In order not to overwrite the changes made by the user, the template file contains special comments, where the user can insert his code in between. The comments have the following format:

```
/**********************************************************************************
 * DO NOT CHANGE THIS COMMENT! <USERBLOCK $Variable_Name> DO NOT CHANGE THIS COMMENT
**********************************************************************************/

/**********************************************************************************
 * DO NOT CHANGE THIS COMMENT! </USERBLOCK> DO NOT CHANGE THIS COMMENT
**********************************************************************************/
```

Caution
Do not modify or delete the comments.

Example:

The following example explains where code can be implemented:

```
FUNC(void, IOHWAB_CODE) IoHwAb_Init(void)
{
  /* A */
/*****************************************************************************
 * DO NOT CHANGE THIS COMMENT!  <USERBLOCK … >             DO NOT CHANGE THIS COMMENT!
 *****************************************************************************/
  /* B */
  return;
/*****************************************************************************
 * DO NOT CHANGE THIS COMMENT!  </USERBLOCK>               DO NOT CHANGE THIS COMMENT!
 *****************************************************************************/

  /* C */
} /* IoHwAb_Init() */
```

A and C: all modifications before and after will be deleted at the next generation process

B: all modifications will not be deleted

The I/O Hardware Abstraction provides the following generated template files:

- IoHwAb.c
- IoHwAb_Cbk.h

4.5.1 Generated services in template files

The I/O Hardware Abstraction has no fixed API, there are fix services (`IoHwAb_Init()`, `IoHwAb_GetVersionInfo()`) as well as services that are generated depending on how much and which kind of signals are configured. The latter contain parts that will not be overwritten during the generation process. They have no fixed name, but an UUID as Identifier:

```
/*****************************************************************************
 * DO NOT CHANGE THIS COMMENT! <USERBLOCK 81fe3f4b-eabd-4f43-8d9a-a0c182f0cc71> DO NOT
CHANGE THIS COMMENT
*****************************************************************************/

/*****************************************************************************
 * DO NOT CHANGE THIS COMMENT! </USERBLOCK> DO NOT CHANGE THIS COMMENT
*****************************************************************************/
```

If a signal is deleted in the configuration, possibly already implemented code would be lost, because the generated service is also deleted from the generated file. In this case, the 'IoHwAb.c' template file contains a section at the end, which receives the code from all services whose associated signals have been deleted:

```
/**********************************************************************************
* DO NOT CHANGE THIS COMMENT! <<USERBLOCK Start of removed code area>> DO NOT CHANGE THIS
COMMENT!
**********************************************************************************/

/**********************************************************************************
 * DO NOT CHANGE THIS COMMENT!       <</USERBLOCK>>     DO NOT CHANGE THIS COMMENT!
 **********************************************************************************/
```

Caution

This section only contains code from signals that have been deleted during the last generation process. During the next generation, the section will be cleared.

5 API Description

5.1 Interfaces Overview

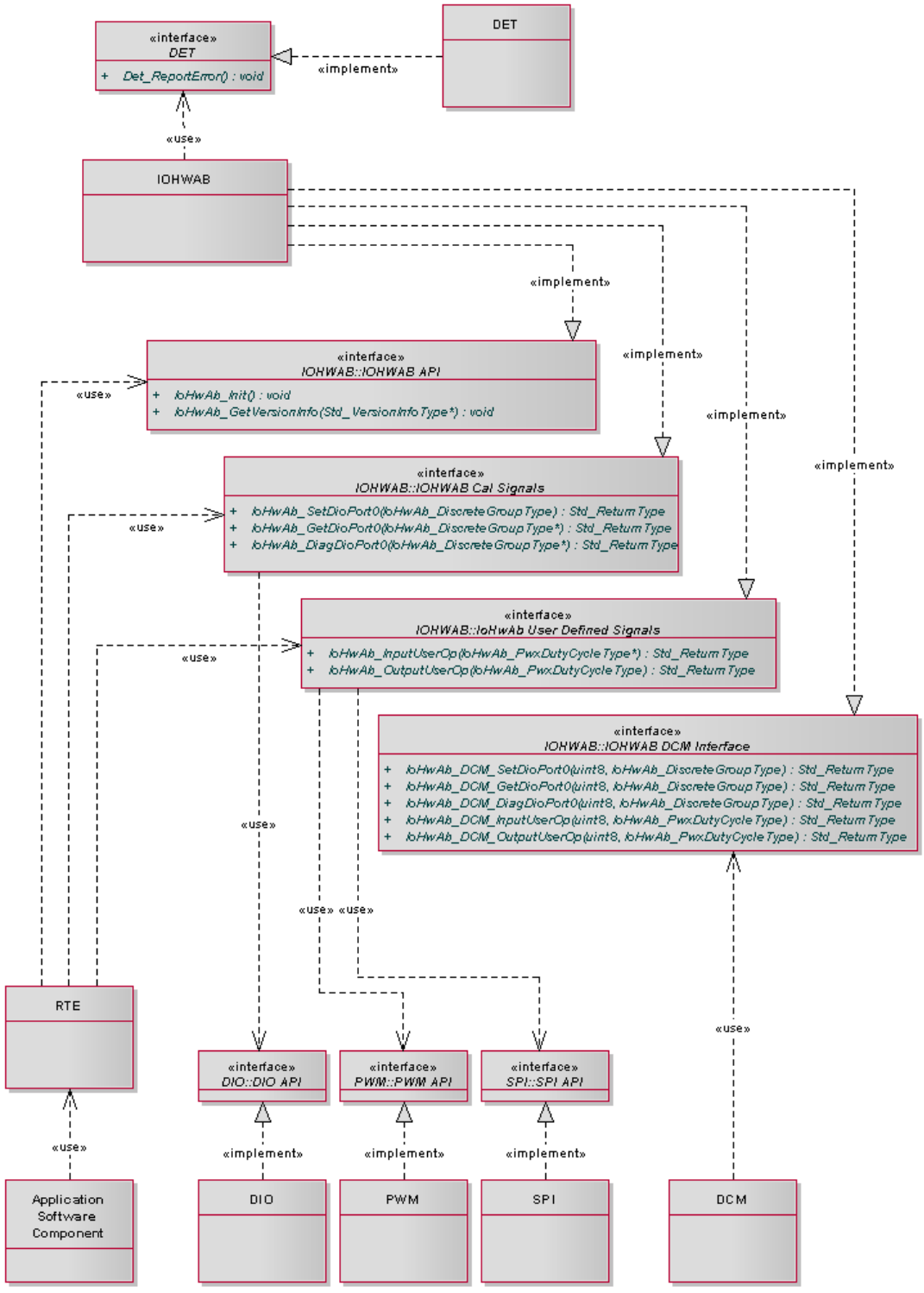


Figure 5-1 IOHWAB interactions with other BSW

5.2 Type Definitions

Type Name	C-Type	Description	Value Range
IoHwAb_BoolType	uint8	Value type for a MCAL signal that uses a DIO channel	`STD_LOW` `STD_HIGH`
IoHwAb_DiscreteGroupSize	uint8/ uint16/ uint32	Value type for a MCAL signal that uses either a DIO port or a DIO channel group	Depends on the configured data type
IoHwAb_CurrentType	uint16/ uint32	Value type for ECU Signals of the Analogue Class that depict current information	Depends on the configured data type
IoHwAb_VoltageType	uint16/ uint32	Value type for ECU Signals of the Analogue Class that depict voltage information	Depends on the configured data type
IoHwAb_ResistanceType	uint16/ uint32	Value type for ECU Signals of the Analogue Class that depict resistance information	Depends on the configured data type
IoHwAb_SignalDiagnosisType	uint16/ uint32	Value type for ECU Signals of the Diagnosis Class that depict the electrical failure state of an Output Signal	Depends on the configured data type
IoHwAb_PwxPeriodType	uint16/ uint32	Value type for ECU Signals of the “Pulse Width Modulation” Class That means this type is used either for modulation output or for demodulation input.	Depends on the configured data type
IoHwAb_PwxDutyCycleType	uint16/ uint32	Value type for ECU Signals of the “Pulse Width Modulation” Class That means this type is used either for modulation output or for demodulation input.	Depends on the configured data type

Table 5-1 Type definitions

.3 Services provided by IOHWAB

he IOHWAB API consists of services, which are realized by function calls.

.3.1 IoHwAb_Init

Prototype	
oid **IoHwAb_Init** (void)	
Parameter	
-	--
Return code	
oid	--
Functional Description	
his function stub can be used to implement any initialization activities and shall be called by the ECUM. It s not intended to initialize drivers in the IOHWAB initialization function. Driver initialization has to be done y the ECUM.	
Particularities and Limitations	
This service is synchronous.	
This service is non re-entrant.	
This service can be disabled using the preprocessor switch `IOHWAB_USE_INIT_FUNCTION`.	
Expected Caller Context	
Expected to be called by an application	

ble 5-2 IoHwAb_Init

.3.2 IoHwAb_GetVersionInfo

Prototype

Expected Caller Context
■ Expected to be called by an application.

Table 5-3 IoHwAb_GetVersionInfo

5.4 Generated API functions

The IOHWAB does not have a fixed API like other BSW modules; rather the API will be generated according to the used configuration.

Caution
The following functions can be invoked concurrently by the RTE. As software on the lower layer may not support concurrency, the application should avoid parallel calls of signal services.

5.4.1 IoHwAb_Set<CalSignalName>

Prototype	
`Std_ReturnType` **`IoHwAb_Set<CalSignalName>`** `( <ConfiguredType> signal )`	
Parameter	
signal	Signal value to be written to the DIO driver
Return code	
E_OK	Successfully executed
E_NOT_OK	An error occurred
Functional Description	
This service writes a given value to an existing DIO entity, i.e. it calls the DIO API (Dio_WriteChannel(), Dio_WriteChannelGroup(), Dio_WritePort()) and passes the parameter value through to the DIO driver.	
Particularities and Limitations	
■ This service is synchronous. ■ This service is non re-entrant. ■ This service will be generated, if 'IoHwAbCreateOpSet' is enabled for a MCAL signal	
Expected Caller Context	
■ Expected to be called by the RTE	

Table 5-4 IoHwAb_Set<CalSignalName>

Info
The upper service description does not refer to a certain API function, but to a function that will be generated by the IOHWAB. The service name, as well as the parameter format is influenced by the configuration settings.

5.4.2 IoHwAb_Get<CalSignalName>

Prototype	
`Std_ReturnType` **`IoHwAb_Get<CalSignalName>`** `( <ConfiguredType> *signal )`	
Parameter	
signal	Pointer to the variable, where the read value shall be stored
Return code	
E_OK	Successfully executed
E_NOT_OK	An error occurred
Functional Description	
This service reads a value from an existing DIO entity, i.e. it calls the DIO API (Dio_ReadChannel(), Dio_ReadChannelGroup(), Dio_ReadPort()) and stores the read value into the given address.	
Particularities and Limitations	
■ This service is synchronous. ■ This service is non re-entrant. ■ This service will be generated, if 'IoHwAbCreateOpGet' is enabled for a MCAL signal	
Expected Caller Context	
■ Expected to be called by the RTE	

Table 5-5 IoHwAb_Get<CalSignalName>

Info
The upper service description does not refer to a certain API function, but to a function that will be generated by the IOHWAB. The service name, as well as the parameter format is influenced by the configuration settings.

5.4.3 IoHwAb_DCM_<CalSignalName>

Prototype	
`Std_ReturnType IoHwAb_DCM_<CalSignalName> ( uint8 action, <ConfiguredType> signal )`	
Parameter	
action	Action to be executed
signal	Signal value to be written to the buffer
Return code	
E_OK	Successfully executed
E_NOT_OK	An error occurred
Functional Description	
This is the DCM access function for an MCAL signal. The state of the signal functions (OP_GET, OP_SET, OP_DIAG) can be changed by this service. The following states are available: ■ `IOHWAB_CONTROLTOECU` – The current value will be unlocked. The following read and write operations (by the SW-C) will work normally. Here, the parameter `signal` is ignored. ■ `IOHWAB_RESETTODEFAULT` – The signal will be locked and reset to the configured default value (for further information, see 6.1.2.1.1), i.e. the configured default value will be written to the hardware. The following read and write operations (by the SW-C) will have no effect of the hardware. ■ `IOHWAB_FREEZE` – The signal will be locked, i.e. the signal is read/written and locked. Further reads (by the SW-C) will return the locked value. Further writes (by the SW-C) will have no effect on the hardware. Here, the parameter `signal` is ignored. ■ `IOHWAB_ADJUSTMENT` – The signal will be locked, i.e. the parameter `signal` will be written into the buffer (input)/to the hardware (output). Further reads (by the SW-C) will always return this value. Further writes (by the SW-C) will have no effect on the hardware.	
Particularities and Limitations	
■ This service is synchronous. ■ This service is non re-entrant. ■ This service will be generated, if 'IoHwAbCreateOpGet', 'IoHwAbCreateOpSet' and 'IoHwAbDcmAccess' are enabled for a MCAL signal	
Expected Caller Context	
■ Expected to be called by the DCM	

Table 5-6 IoHwAb_DCM_<CalSignalName>

Info

The upper service description does not refer to a certain API function, but to a function that will be generated by the IOHWAB. The service name, as well as the parameter format is influenced by the configuration settings.

5.4.4 IoHwAb_DCM_Read<CalSignalName>

Prototype	
`Std_ReturnType` **`IoHwAb_DCM_Read<CalSignalName>`** `( <ConfiguredType> *signal )`	
Parameter	
signal	Pointer to the variable, where the read value shall be stored
Return code	
E_OK	Successfully executed
E_NOT_OK	An error occurred
Functional Description	
This method is a bypass read function that executes a normal OP_GET, reading the value from the hardware, **even if the signal is locked.**	
Particularities and Limitations	
■ This service is synchronous. ■ This service is non re-entrant. ■ This service will be generated, if 'IoHwAbCreateOpGet', 'IoHwAbCreateOpSet' and 'IoHwAbDcmAccess' are enabled for a MCAL signal	
Expected Caller Context	
■ Expected to be called by the DCM	

Table 5-7 IoHwAb_DCM_Read<CalSignalName>

Info
The upper service description does not refer to a certain API function, but to a function that will be generated by the IOHWAB. The service name, as well as the parameter format is influenced by the configuration settings.

5.4.5 IoHwAb_Diag<CalSignalName>

Prototype	
`Std_ReturnType` **`IoHwAb_Diag<CalSignalName>`** `( <ConfiguredType> *signal )`	
Parameter	
signal	Pointer to the variable, where the read value shall be stored
Return code	
E_OK	Successfully executed
E_NOT_OK	An error occurred
Functional Description	
This service reads a value from an existing DIO entity for diagnostic purposes, i.e. it calls the DIO API (Dio_ReadChannel(), Dio_ReadChannelGroup(), Dio_ReadPort()) and stores the read value into the given address.	
Particularities and Limitations	
■ This service is synchronous. ■ This service is non re-entrant. ■ This service will be generated, if 'IoHwAbCreateOpDiag' is enabled for a MCAL signal	
Expected Caller Context	
■ Expected to be called by the RTE	

Table 5-8 IoHwAb_Diag<CalSignalName>

Info
The upper service description does not refer to a certain API function, but to a function that will be generated by the IOHWAB. The service name, as well as the parameter format is influenced by the configuration settings.

5.4.6 IoHwAb_Input<UserDefSignalName>

Prototype	
`Std_ReturnType` **`IoHwAb_Input<UserDefSignalName>`** `( <ConfiguredType> *signal )`	
Parameter	
signal	Pointer to the variable, where the read value shall be stored
Return code	
E_OK	Successfully executed
E_NOT_OK	An error occurred
Functional Description	
This service is for user-defined input operations in general. Any lower level software can be accessed for reading values of the configured data type.	
Particularities and Limitations	
■ This service is synchronous. ■ This service is non re-entrant. ■ This service will be generated, if ‘IoHwAbAccess’ is set to ‘Input’ for a user-defined signal.	
Expected Caller Context	
■ Expected to be called by the RTE	

Table 5-9 IoHwAb_Input<UserDefSignalName>

Info
The upper service description does not refer to a certain API function, but to a function stub that will be generated by the IOHWAB. The service name, as well as the parameter format is influenced by the configuration settings.

5.4.7 IoHwAb_DCM_Input<UserDefSignalName>

Prototype	
`Std_ReturnType` **`IoHwAb_DCM_Input<UserDefSignalName>`** `( uint8 action, <ConfiguredType> signal )`	
Parameter	
action	Action to be executed
signal	Signal value to be stored to the buffer
Return code	
E_OK	Successfully executed
E_NOT_OK	An error occurred
Functional Description	
This is the DCM access function for a user-defined input function. The state of an input function can be changed by this service. The following states are available: ■ `IOHWAB_CONTROLTOECU` – The current value will be unlocked. The following read operations (by the SW-C) will work normally. Here, the parameter `signal` is ignored. ■ `IOHWAB_RESETTODEFAULT` – The signal will be locked and reset to the configured default value (for further information, see 6.1.2.2.1.1), i.e. the configured default value will be written to an internal buffer. Further reads (by the SW-C) will return the locked value. Here, the parameter `signal` is ignored. ■ `IOHWAB_FREEZE` – The signal will be locked, i.e. the signal is read and locked. Further reads (by the SW-C) will return the locked value. Here, the parameter `signal` is ignored. ■ `IOHWAB_ADJUSTMENT` – The signal will be locked, i.e. the parameter `signal` will be written into the internal buffer. Further reads (by the SW-C) will return the locked value.	
Particularities and Limitations	
■ This service is synchronous. ■ This service is non re-entrant. ■ This service will be generated, if 'IoHwAbAccess' of a user-defined signal is set to 'Input' and 'IoHwAbDcmAccess' is enabled.	
Expected Caller Context	
■ Expected to be called by the DCM	

Table 5-10 IoHwAb_DCM_Input<UserDefSignalName>

Info
The upper service description does not refer to a certain API function, but to a function that will be generated by the IOHWAB. The service name, as well as the parameter format is influenced by the configuration settings.

5.4.8 IoHwAb_DCM_Read<UserDefSignalName>

Prototype	
`Std_ReturnType` **`IoHwAb_DCM_Read<UserDefSignalName>`** `( <ConfiguredType> *signal )`	
Parameter	
signal	Pointer to the variable, where the read value shall be stored
Return code	
E_OK	Successfully executed
E_NOT_OK	An error occurred
Functional Description	
This method is a bypass read function that executes a normal input operation, reading the value from the hardware, **even if the signal is locked.**	
Particularities and Limitations	
■ This service is synchronous. ■ This service is non re-entrant. ■ This service will be generated, if 'IoHwAbAccess' of a user-defined signal is set to 'Input' and 'IoHwAbDcmAccess' is enabled.	
Expected Caller Context	
■ Expected to be called by the DCM	

Table 5-11 IoHwAb_DCM_Read<UserDefSignalName>

Info
The upper service description does not refer to a certain API function, but to a function that will be generated by the IOHWAB. The service name, as well as the parameter format is influenced by the configuration settings.

Caution
This service will be generated for both, input and output signals. For input signals it works similar to the CalSignals-implementation. For output signals, a read functionality has to be implemented manually.

5.4.9 IoHwAb_Output<UserDefSignalName>

Prototype	
`Std_ReturnType` **`IoHwAb_Output<UserDefSignalName>`** `( <ConfiguredType> signal )`	
Parameter	
signal	Signal value to be written
Return code	
E_OK	Successfully executed
E_NOT_OK	An error occurred
Functional Description	
This service is for user-defined output operations in general. Any lower level software can be accessed for writing values of the configured data type.	
Particularities and Limitations	
■ This service is synchronous. ■ This service is non re-entrant. ■ This service will be generated, if 'IoHwAbAccess' is set to 'Output' for a user-defined signal.	
Expected Caller Context	
■ Expected to be called by the RTE.	

Table 5-12 IoHwAb_Output<UserDefSignalName>

Info
The upper service description does not refer to a certain API function, but to a function that will be generated by the IOHWAB. The service name, as well as the parameter format is influenced by the configuration settings.

5.4.10 IoHwAb_DCM_Output<UserDefSignalName>

Prototype	
`Std_ReturnType` **`IoHwAb_DCM_Output<UserDefSignalName>`** `( uint8 action, <ConfiguredType> signal )`	
Parameter	
action	Action to be executed
signal	Signal value to be stored to the buffer
Return code	
E_OK	Successfully executed
E_NOT_OK	An error occurred
Functional Description	
This is the DCM access function for a user defined output function. The state of an output function can be changed by this service. The following states are available: ■ `IOHWAB_CONTROLTOECU` – The current value will be unlocked. The following write operations (by the SW-C) will work normally. Here, the parameter `signal` is ignored. ■ `IOHWAB_RESETTODEFAULT` – The signal will be locked and reset to the configured default value (for further information, see 6.1.2.2.1.1), i.e. the configured default value will be written to the hardware. Further writes (by the SW-C) will have no effect on the hardware. Here, the parameter `signal` is ignored. ■ `IOHWAB_FREEZE` – The signal will be locked, i.e. the signal is written and locked. Further writes (by the SW-C) will have no effect on the hardware. Here, the parameter `signal` is ignored. ■ `IOHWAB_ADJUSTMENT` – The signal will be locked, i.e. the parameter `signal` will be written to the hardware. Further writes (by the SW-C) will have no effect on the hardware.	
Particularities and Limitations	
■ This service is synchronous. ■ This service is non re-entrant. ■ This service will be generated, if 'IoHwAbAccess' of a user-defined signal is set to 'Output' and 'IoHwAbDcmAccess' is enabled.	
Expected Caller Context	
■ Expected to be called by the DCM	

Table 5-13 IoHwAb_DCM_Output<UserDefSignalName>

Info
The upper service description does not refer to a certain API function, but to a function that will be generated by the IOHWAB. The service name, as well as the parameter format is influenced by the configuration settings.

5.4.11 IoHwAb_<SignalHandlerName>

Prototype	
`void IoHwAb_<SignalHandlerName> ( void )`	
Parameter	
--	--
Return code	
void	--
Functional Description	
This function stub can be used to implement a special signal processing, e.g. debouncing.	
Particularities and Limitations	
■ This service is synchronous. ■ This service is non re-entrant.	
Expected Caller Context	
■ Expected to be called by the RTE.	

Table 5-14 IoHwAb_<SignalHandlerName>

Info
The upper service description does not refer to a certain API function, but to a function that will be generated by the IOHWAB. The service name is influenced by the configuration settings.

5.5 Services used by IOHWAB

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

Component	API
DET	Det_ReportError (optional)
DIO	Dio_WriteChannel (optional)
DIO	Dio_WriteChannelGroup (optional)
DIO	Dio_WritePort (optional)
DIO	Dio_ReadChannel (optional)
DIO	Dio_ReadChannelGroup (optional)
DIO	Dio_ReadPort (optional)

Table 5-15 Services used by the IOHWAB

5.6 Callback Functions

This release of the IOHWAB does not implement any callback functions. If necessary, the user may implement prototypes of callback functions in the file IoHwAb_Cbk.h.

Caution
It is not possible to pass an approaching callback function through the RTE to the software component. Actions for handling callbacks have to be implemented inside the IOHWAB.

5.7 Configurable Interfaces

5.7.1 Notifications

The IOHWAB does not provide notifications.

5.8 Service Ports

5.8.1 Client Server Interface

A client server interface is related to a Provide Port at the server side and a Require Port at client side.

5.8.1.1 Provide Ports

At the Provide Ports of the IOHWAB the API functions described in 5.3 are available as Runnable Entities. The Runnable Entities are invoked via Operations. The mapping from a SWC client call to an Operation is performed by the RTE. In this mapping, the RTE adds Port Defined Argument Values to the client call of the SWC, if configured.

The following table presents the Provide Ports defined for the IOHWAB and the Operations defined for the Provide Ports, the API functions related to the Operations and the Port Defined Argument Values to be added by the RTE:

Provide Port	Operation	API Function	Port Defined Argument Values
<CalSignalName>	OP_GET	IoHwAb_Get<CalSignalName>	*IoHwAb_BoolType/ *IoHwAb_DiscreteGoupType
	OP_SET	IoHwAb_Set<CalSignalName>	IoHwAb_BoolType/ IoHwAb_DiscreteGoupType
	OP_DIAG	IoHwAb_Diag<CalSignalName>	IoHwAb_BoolType/ IoHwAb_DiscreteGoupType
<UserDefPortName>	<UserDefSignalName>	IoHwAb_Input<UserDefSignalName>	*<Configured data type>
<UserDefPortName>	<UserDefSignalName>	IoHwAb_Output<UserDefSignalName>	<Configured data type>

Table 5-16 Provide Ports on BSW module side

5.8.1.2 Require Ports

The current version of the IOHWAB does not use any Require Ports.

5.9 Software Component Template

5.9.1 Generation

The definition of the Provide Ports is described in an XML file. This file describes the IOHWAB as a software component with ports to which other applications can connect. This XML file is created, when the module is generated in DaVinci Configurator.

5.9.2 Import to the DaVinci modeling tool

For further processing, the generated software component template has to be imported into DaVinci Developer. In an existing DaVinci Developer project use ‘File->Import XML File…’ and choose the correct file.

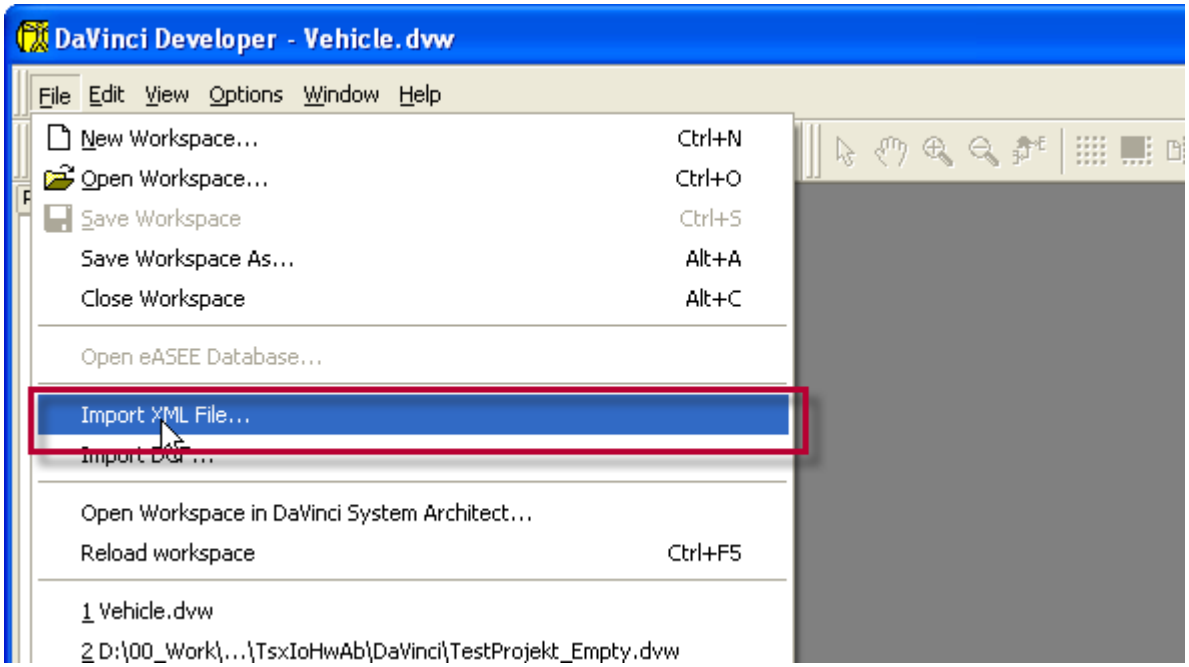


Figure 5-2 Import a new software component into DaVinci

After importing the IOHWAB as a software component, there is a new component type with all the configured ports available in the library view. You can open the software component by a double click.

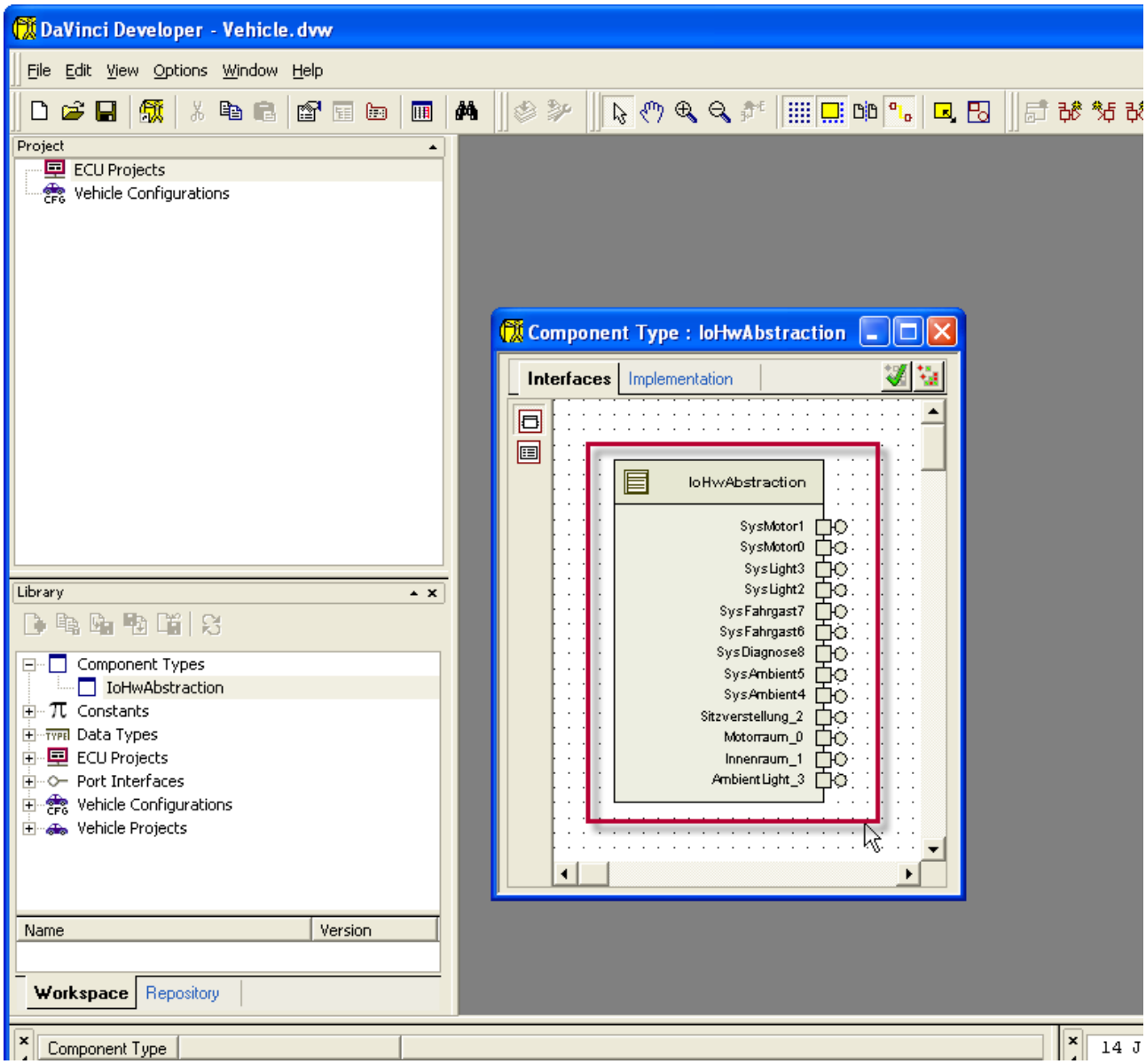


Figure 5-3 The imported software component

Caution

The ports in the DaVinci modeling tool contain the operations configured in the DaVinci Configurator. It may be irritating, that e.g. an output operation of a user-defined signal looks as follows:

UserOperation_1(In IoHwAbSignalDiagnosisType signal) [E_NOT_OK]

In this case, ‘In’ means that the operation’s parameter is an input parameter. To avoid confusion, mind the following:

1. Input operations (OP_GET, OP_DIAG, Input) always have output parameters (‘by reference’).

Output operations (OP_SET, Output) always have input parameters (‘by value’).

6 Configuration

In the IOHWAB, the attributes can be configured with the following methods:

- Configuration in DaVinci Configurator for a detailed description see 6.1
- Configuration with a standard AUTOSAR GCE – this option will not be described explicitly

6.1 Configuration of IOHWAB with DaVinci Configurator

The IOHWAB is configured with the help of the configuration tool DaVinci Configurator.

Info
In case of object delivery modifications are without effect, if a parameter is specified as a pre-compile value. Different object code is needed for different settings. Other object files can be obtained at Vector Informatik.

6.1.1 Start configuration of the IOHWAB

The component name of the IOHWAB in DaVinci Configurator is “IoHwAb”. In the “Architecture view” (initial page) of the DaVinci Configurator, the IOHWAB can be opened by its context menu to start its configuration. Optionally, the IOHWAB can be opened for configuration with the component list under the “System” tab located at the left side of the DaVinci Configurator.

Caution
Please save your project right after creation to ensure data consistency.

The configuration tool needs data from the saved file to calculate further configuration options as well as values of GUI elements.

6.1.2 Tab ‘IoHwAb Configuration’

6.1.2.1 Node ‘IoHwAbCalSignals’

This node contains the configuration of MCAL signals. This version of the IOHWAB abstracts the complete functionality of the DIO driver, i.e. Channels, Channel Groups and Ports can be read and written.

To add a MCAL signal, right-click on this node and choose ‘Append IoHwAbDiscrete’ in the context menu.

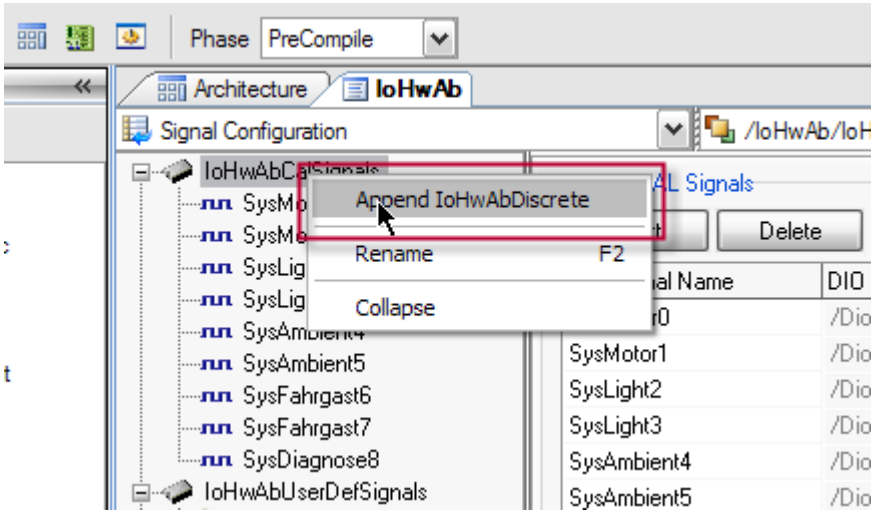


Figure 6-1 Adding a MCAL signal

6.1.2.1.1 Node ‘IoHwAbDiscrete’

This node contains the configuration of a single MCAL signal.

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Create OP_SET for Signal	pre-compile	Boolean	**ON** OFF	This switch enables/disables the generation of an OP_SET function for the current signal.
Create OP_GET for Signal	pre-compile	Boolean	ON **OFF**	This switch enables/disables the generation of an OP_GET function for the current signal.
Create OP_DIAG for Signal	pre-compile	Boolean	ON **OFF**	This switch enables/disables the generation of an OP_DIAG function for the current signal.
Used DIO Entity	pre-compile	Reference to a DIO entity	**--**	This dropdown field configured the reference to the DIO entity that is used by this signal.
Enable DCM Access functionality	pre-compile	Boolean	**ON** OFF	This switch enables/disables the generation of DCM access functions for the OP_SET and OP_GET function of the current signal.
Default Value	pre-compile	IoHwAb_BoolType /IoHwAb_Discrete GroupType	**0**	This field configures the default value that will be assigned to the signal if the associated DCM service is called with the attribute `IOHWAB_RESETTODEFAULT`.

Table 6-1 Node ‘IoHwAbDiscrete’

6.1.2.2 Node ‘IoHwAbUserDefSignals’

This node contains user-defined ports. These ports are similar to the MCAL signals, but do not contain the abstraction of a certain driver. Unlike the MCAL signals user-defined ports are not limited to predefined operations (OP_SET, OP_GET, OP_DIAG), but they can contain a various number of customizable operations.

To add a user-defined port right-click on this node and choose ‘Append IoHwAbUserDefPort’ in the context menu.

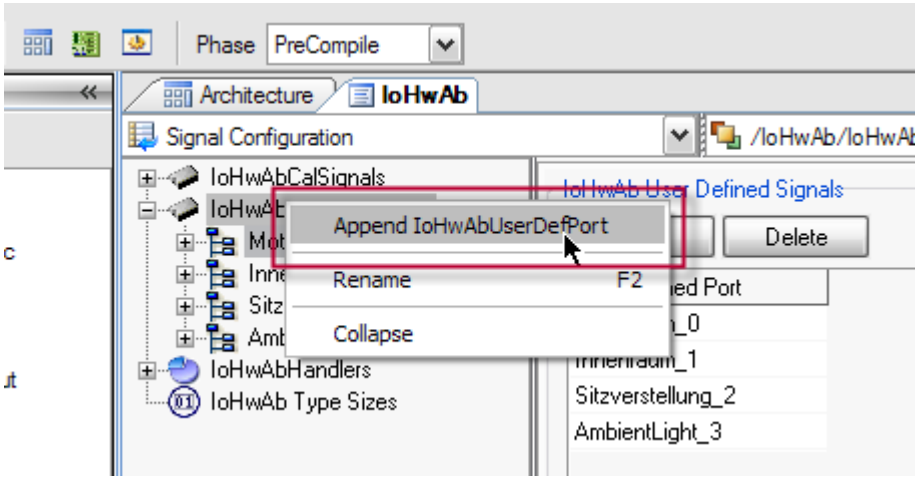


Figure 6-2 Adding a user-defined port

6.1.2.2.1 Node ‘IoHwAbUserDefPort’

This node contains user-defined signals.

To add a user-defined signal right-click on this node and choose ‘Append IoHwAbUserDefOp’ in the context menu.

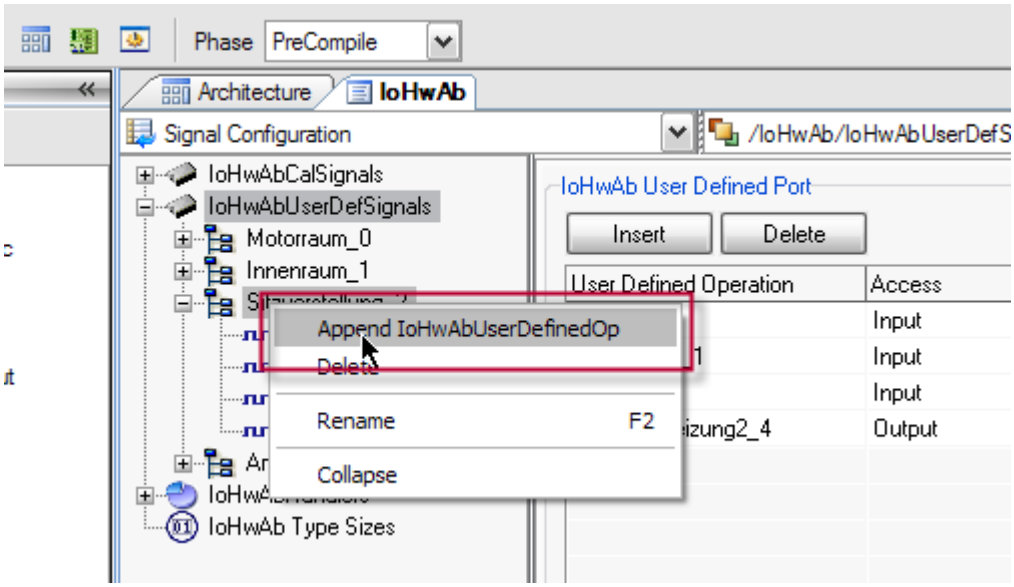


Figure 6-3 Adding a user-defined operation

6.1.2.2.1.1 Node ‘IoHwAbUserDefOp’

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Access	pre-compile	Boolean	**Output** Input	This switch configures the direction of the operation: input operations read from the driver layer, output operations write to the driver layer.
Data Type	pre-compile	String parameter	IoHwAb_BoolType IoHwAb_CurrentType IoHwAb_VoltageType IoHwAb_ResistanceType IoHwAb_SignalDiagnosisType IoHwAb_PwxPeriodType IoHwAb_PwxDutyCycleType sint8 **uint8** sint16 uint16 sint32 uint32 float32 float64	This field configures the data type of the operation’s parameter.
Unit	--	String parameter	mV V Ohm kOhm A mA µA °C Percent **n/a**	This field configures the unit of the operation. No output will be generated from this configuration element.
Range Min	--	Numeric value	**0**	This field contains the minimum value of the operation’s unit. No output will be generated from this configuration element.
Range Max	--	Numeric value	**--**	This field contains the maximum value of the operation’s unit. No output will be generated from this configuration element.

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Resolution	--	Numeric value	**--**	This field contains the hardware resolution of the signal. No output will be generated from this configuration element.
Enable DCM Access Functionality	pre-compile	Boolean	ON **OFF**	This switch enables/disables the generation of DCM access functions for the current operation.
Default Value	pre-compile	Depends on the data type of the signal	**--**	This field contains the default value, that will be assigned to the signal if the associated DCM function is called with the parameter `IOHWAB_RESETTODEFAULT`.

6.1.2.3 Node ‘IoHwAbHandlers’

This node contains all the signal handler functions. Signal handlers are called cyclically and can be used for special signal processing.

To add a handler function right-click on this node and choose ‘Append IoHwAbHandler’ in the context menu.

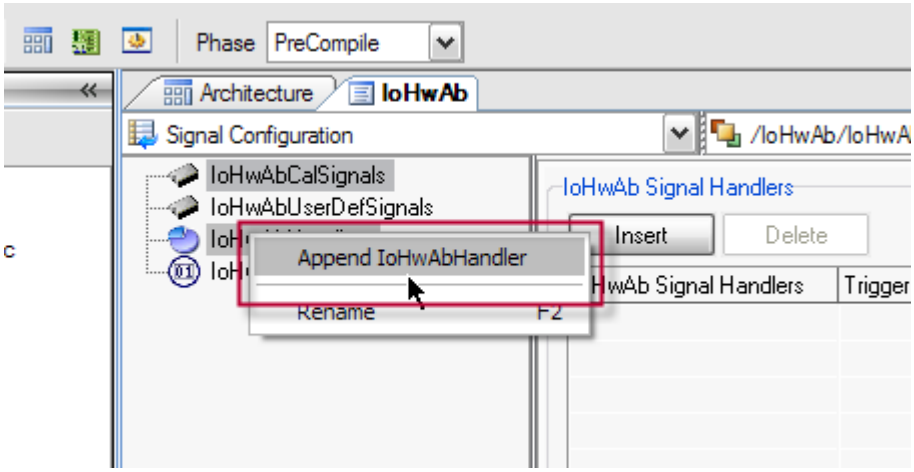


Figure 6-4 Adding a handler function

6.1.2.3.1 Node ‘IoHwAbHandler’

This node contains the configuration of a signal handler function.

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Trigger Period	pre-compile	Numeric Value	**10**	This field contains the trigger period for the signal handler function.
Timebase	pre-compile	String parameter	sec **msec**	This switch configures the time base of the handler’s trigger period.

6.1.2.4 Node IoHwAbDataTypeSizes

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Size of IoHwAb_DiscreteGroupSize	pre-compile	String parameter	**uint8** uint16 uint32	This field determines the size of the data type `IoHwAb_DiscreteGroupType`.
Size of IoHwAb_CurrentType	pre-compile	String parameter	**uint16** uint32	This field determines the size of the data type `IoHwAb_CurrentType`.
Size of IoHwAb_VoltageType	pre-compile	String parameter	**uint16** uint32	This field determines the size of the data type `IoHwAb_VoltageType`.
Size of IoHwAb_ResistanceType	pre-compile	String parameter	**uint16** uint32	This field determines the size of the data type `IoHwAb_ResistanceType`.
Size of IoHwAb_SignalDiagnosisType	pre-compile	String parameter	**uint16** uint32	This field determines the size of the data type `IoHwAb_SignalDiagnosisType`.
Size of IoHwAb_PwxPeriodType	pre-compile	String parameter	**uint16** uint32	This field determines the size of the data type `IoHwAb_PwxPeriodType`.
Size of IoHwAb_PwxDutyCycleType	pre-compile	String parameter	**uint16** uint32	This field determines the size of the data type `IoHwAb_PwxDutyCycleType`.

6.1.3 Tab ‘General Settings’

This tab contains the general configuration of the IOHWAB.

6.1.3.1 Area ‘Error Detection – Development Mode’

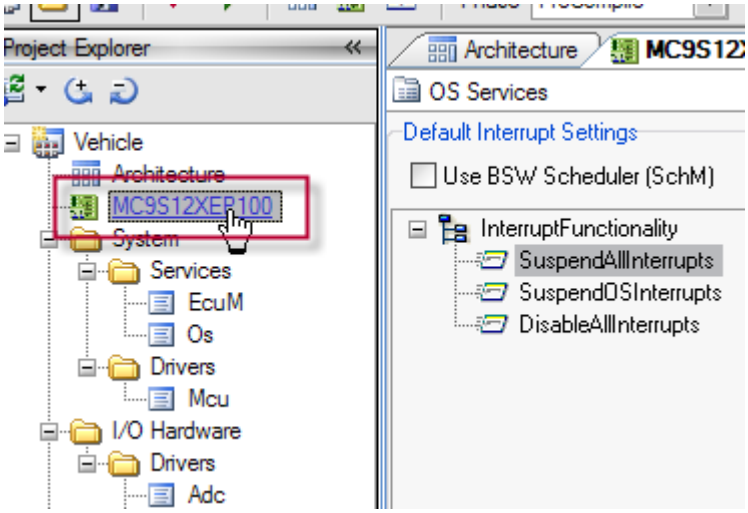
Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Development Mode	pre-compile	Boolean	**ON** OFF	This switch enables/disables the development error

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
				detection of the IOHWAB.
Check Parameter Pointer	pre-compile	Boolean	**ON** OFF	This switch enables/disables the pointer parameter check of the generated API functions. The parameter will be checked for not being a NULL_PTR.
Development Error Reporting	pre-compile	Boolean	**ON** OFF	This switch enables/disables the reporting of detected development errors to the error tracer module (by default: DET).
Errorhook Function	pre-compile	C-function identifier	**Det_ReportError**	This field contains the function name of the development error reporting function.
Include File	pre-compile	Header file	**Det.h**	This FileEdit is for inclusion of the header file that contains the development error reporting function.

6.1.3.2 Area ‘Interrupt Services’

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Critical Section Handling	pre-compile	--	**UseSuspendFunctions** UseOSFunctions UseEnableFunctions	This field configures the critical section handling of the IOHWAB.

Info
The values in this list can be configured in the platform specific configuration:



6.1.3.3 Area ‘Common Settings’

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Use RTE	pre-compile	Boolean	**ON** OFF	This switch enables/disables RTE usage. If RTE usage is on, IoHwAb types and function prototypes are generated by the RTE generator. Otherwise IoHwAb types will be generated into IoHwAb_types.h and function prototypes will be added to IoHwAb_Cfg.h
Export As Service Component	pre-compile	Boolean	ON **OFF**	This parameter enables/disables the generation of the I/O Hardware Abstraction as a Service Component. By Default, the I/O Hardware Abstraction is an Application Component. By enabling this option, the module can be declared as a Service Component.
Component Name	pre-compile	String parameter	**IoHwAbstraction**	This field contains the component name of the IOHWAB. After importing the IOHWAB to a DaVinci Developer project, the service

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
				component will have this name.

6.1.3.4 Area ‘Include List’

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
IoHwAbCfgIncludeList	pre-compile	String parameter	**Dio.h**	This field contains all necessary includes of lower level drivers.

6.1.4 Tab ‘Module API’

6.1.4.1 Area ‘API Optimization’

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Use IoHwAb_Init	pre-compile	Boolean	**ON** OFF	This switch enables/disables the service `IoHwAb_Init()`.
Use IoHwAb_GetVersionInfo	pre-compile	Boolean	**ON** OFF	This switch enables/disables the service `IoHwAb_GetVersionInfo()`.

7 AUTOSAR Standard Compliance

The current version of the IOHWAB has only full support of the DIO driver implemented, i.e. the access to the DIO driver's API is generated completely and needs no further modification. Abstraction of further AUTOSAR drivers as well as custom drivers has to be done manually by using the generated function stubs.

8 Glossary and Abbreviations

8.1 Glossary

Term	Description
DaVinci Configurator	Configuration and generation tool for MICROSAR BSW components

Table 8-1 Glossary

8.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DCM	Diagnostic Communication Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
GCE	Generic Configuration Editor
HIS	Hersteller Initiative Software
IOHWAB	Input/Output Hardware Abstraction
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
PPort	Provide Port
RPort	Require Port
RTE	Runtime Environment
SRS	Software Requirement Specification
SW-C	Software Component
SWS	Software Specification

Table 8-2 Abbreviations

9 Contact

Visit our website for more information on

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

www.vector-informatik.com