Document Title	Specification of DIO Driver
Document Category	Software specification (SWS)
Document UID	20
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Removed GetVersionInfo API
2025-11-27	R25-11	AUTOSAR Release Management	• Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• Editorial changes
2023-11-23	R23-11	AUTOSAR Release Management	• Editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes
2021-11-25	R21-11	AUTOSAR Release Management	• Cleaned return codes
2020-11-30	R20-11	AUTOSAR Release Management	• Cleaned error classification
2019-11-28	R19-11	AUTOSAR Release Management	• MCALMulticoreDistribution (CONC_639) • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Introduced MaskedWritePort API
2017-12-08	4.3.1	AUTOSAR Release Management	• Removed unused artifacts • Editorial changes

▽

△

2016-11-30	4.3.0	AUTOSAR Release Management	• Removed [SWS_Dio_00065] • Replaced content of "7.6.2 Runtime Errors" by "There are no runtime errors." • Replaced content of "7.6.3 Transient Faults" by "There are no transient faults • Removed the definition of the "configuration variants" from 10.1.1 • Changed Figure 2: Include File Structure
2015-07-31	4.2.2	AUTOSAR Release Management	• DET Renaming and Extension Incorporation • Changed DioChannelId, DioPortId precompile configuration
2014-10-31	4.2.1	AUTOSAR Release Management	• DIO: ReadChannelGroup / WriteChannelGroup pointer parameters. Provided support for Link time only. • The generation of link-time parameters aggregated by a postBuildChangeable container may not be possible. Reference to SWS_BSW_00380 is removed.
2013-10-31	4.1.2	AUTOSAR Release Management	• Formalization of Service Interfaces • Revised return values of Service Interfaces • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Updated chapter 10 for 'Scope' values • Changed 'MemMap.h' to 'Dio_MemMap.h' • Added Subchapter 3.x • Requirement IDs changed to new format
2011-12-22	4.0.3	AUTOSAR Administration	• Removed Dem.h from [SWS_Dio_00171] and added new requirement [SWS_Dio_00194]

▽

△

2010-09-30	3.1.5	AUTOSAR Administration	• Added a new API "Dio_LevelType Dio_FlipChannel(Dio_ChannelType ChannelId)" to flip (change from 1 to 0 or from 0 to 1) the level of a channel and return the level of the channel after flip. • Removed requirement DIO174 and rephrased [SWS_Dio_00106]. • Added requirements [SWS_DIO_00188] and [SWS_Dio_00189], to report DET error DIO_E_PARAM_POINTER from Dio_GetVersionInfo.
2010-02-02	3.1.4	AUTOSAR Administration	• Clarification of [SWS_Dio_00014] • DioVersionInfoApi added to DIO071 • Clean up of configuration parameters and header file inclusion structure • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• Harmonized initialization with MCAL modules • Optional inclusion of the DEM header file • Added explanation on dependency between DIO_PORT_MASK and DIO_PORT_OFFSET • Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• File structure update • Removed BSW00324 • In the configuration where pre-compile and link time is possible the variant for pre-compile is now always "PC" and not "All variants". • Added Chapter 8.6 • Changes in referencing symbolic naming • Updated traceability matrix regarding SRS_BSW_00435 and SRS_BSW_00436

▽

△

			△ • Legal disclaimer revised • "Advice for users" revised • "Revision Information" added
2006-05-16	2.0	AUTOSAR Administration	• Major changes in chapter 10, Configuration specification • Structure of document changed partly • Readback support moved to PORT Driver
2005-05-31	1.0	AUTOSAR Administration	• Initial Release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 9
2 Acronyms and Abbreviations 11
3 Related documentation 12
3.1 Input documents & related standards and norms . . . 12
3.2 Related specification . . . 12
4 Constraints and assumptions 13
4.1 Limitations . . . 13
4.2 Applicability to car domains . . . 13
5 Dependencies to other modules 14
5.1 Port Driver Module . . . 14
5.2 Default Error Tracer . . . 14
6 Requirements Tracing 15
7 Functional specification 17
7.1 General Behaviour . . . 17
7.1.1 Background & Rationale . . . 17
7.1.2 Requirements . . . 17
7.2 Initialization . . . 19
7.2.1 Background & Rationale . . . 19
7.2.2 Requirements . . . 19
7.3 Runtime Reconfiguration . . . 20
7.3.1 Background & Rationale . . . 20
7.3.2 Requirements . . . 20
7.4 DIO Write Service . . . 20
7.4.1 Background & Rationale . . . 20
7.4.2 Requirements . . . 20
7.4.2.1 DIO Channel Write Service . . . 21
7.4.2.2 DIO Port Write Service . . . 21
7.4.2.3 DIO Channel Group Write Service . . . 21
7.4.2.4 DIO Masked Port Write Service . . . 21
7.5 DIO Read Service . . . 22
7.5.1 Background & Rationale . . . 22
7.5.2 Requirements . . . 22
7.5.2.1 DIO Channel Read Service . . . 22
7.5.2.2 DIO Port Read Service . . . 22
7.5.2.3 DIO Channel Group Read Service . . . 23
7.5.2.4 DIO Readback of Output Pins . . . 23
7.6 Error Classification . . . 23
7.6.1 Development Errors . . . 23

7.6.2 Runtime Errors . . . 24
7.6.3 Production Errors . . . 24
7.7 Security Events . . . 24
8 API specification 25
8.1 Imported types . . . 25
8.2 Type definitions . . . 25
8.2.1 Dio_ChannelType . . . 25
8.2.2 Dio_PortType . . . 26
8.2.3 Dio_ChannelGroupType . . . 26
8.2.4 Dio_LevelType . . . 27
8.2.5 Dio_PortLevelType . . . 27
8.3 Function definitions . . . 27
8.3.1 Dio_ReadChannel . . . 28
8.3.2 Dio_WriteChannel . . . 29
8.3.3 Dio_ReadPort . . . 30
8.3.4 Dio_WritePort . . . 31
8.3.5 Dio_ReadChannelGroup . . . 32
8.3.6 Dio_WriteChannelGroup . . . 33
8.3.7 Dio_FlipChannel . . . 34
8.3.8 Dio_MaskedWritePort . . . 35
8.4 Scheduled functions . . . 35
8.5 Expected interfaces . . . 36
8.5.1 Mandatory Interfaces . . . 36
8.5.2 Optional Interfaces . . . 36
9 Sequence diagrams 37
9.1 Read a value from a digital I/O . . . 37
9.2 Write a value to a digital I/O . . . 37
10 Configuration specification 38
10.1 Containers and configuration parameters . . . 38
10.1.1 Variants . . . 38
10.1.2 Dio . . . 39
10.1.3 DioGeneral . . . 39
10.1.4 DioPort . . . 41
10.1.5 DioChannel . . . 42
10.1.6 DioChannelGroup . . . 44
10.1.7 DioConfig . . . 46
10.2 Published Information . . . 46
A Not applicable requirements 47

B Change history of AUTOSAR traceable items 48
B.1 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 48
B.1.1 Added Specification Items in R24-11 . . . 48
B.1.2 Changed Specification Items in R24-11 . . . 48
B.1.3 Deleted Specification Items in R24-11 . . . 48
B.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 48
B.2.1 Added Specification Items in R25-11 . . . 48
B.2.2 Changed Specification Items in R25-11 . . . 48
B.2.3 Deleted Specification Items in R25-11 . . . 49
B.3 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 49
B.3.1 Added Specification Items in R26-11 . . . 49
B.3.2 Changed Specification Items in R26-11 . . . 49
B.3.3 Deleted Specification Items in R26-11 . . . 49

1 Introduction and functional overview

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

This specification is applicable to drivers only for on chip DIO pins and ports.

The DIO Driver provides services for reading and writing to / from

- DIO Channels (Pins)
- DIO Ports
- DIO Channel Groups

The behaviour of those services is synchronous.

This module works on pins and ports which are configured by the PORT driver for this purpose. For this reason, there is no configuration and initialization of this port structure in the DIO Driver.

The diagram below identifies the DIO Driver functions, and the structure of the PORT Driver and DIO Driver within the MCAL software layer.

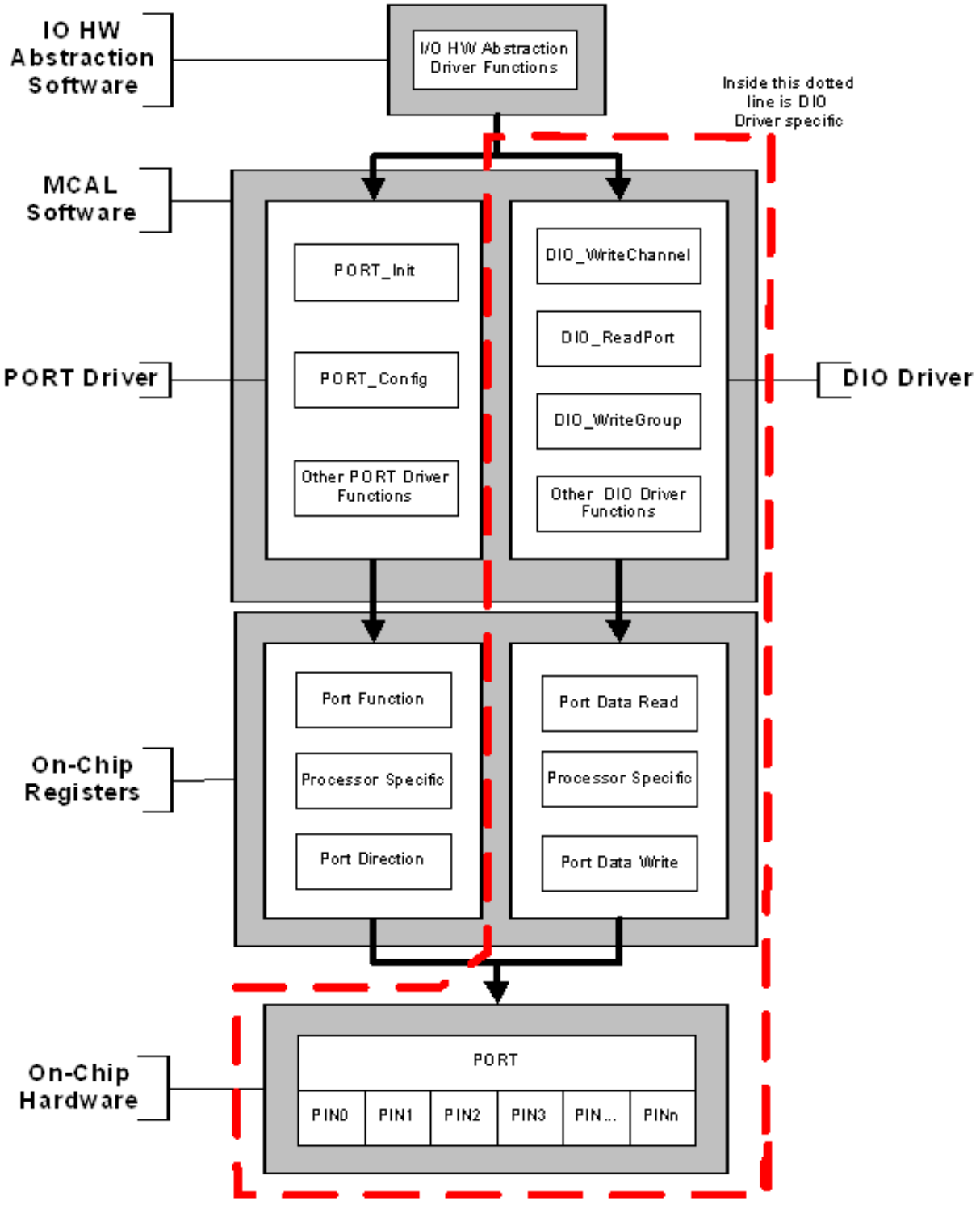


Figure 1.1: DIO Driver Structure and Integration

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the DIO Driver module that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym	Description
DIO channel	Represents a single general-purpose digital input / output pin
DIO port	Represents several DIO channels that are grouped by hardware (typically controlled by one hardware register). Example: Port A (8 bit) of Freescale HC08
DIO channel group	Represents several adjoining DIO channels represented by a logical group. A DIO channel group shall belong to one DIO port. Example: Port pins 2..6 of an 8 bit port addressing a multiplexer
Physical Level (Input)	Two states possible: LOW / HIGH. A bit value '0' represents a LOW, a bit value '1' represents a HIGH.
Physical Level (Output)	Two states possible: LOW / HIGH. A bit value '0' represents a LOW, a bit value '1' represents a HIGH.

Table 2.1: Acronyms and abbreviations used in the scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for DIO Driver.

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

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

5.1 Port Driver Module

Many ports and port pins are assigned by the PORT Driver Module to various functionalities as for example:

- General purpose I/O
- ADC
- SPI
- PWM

[SWS_Dio_00061] ⌈The Dio module shall not provide APIs for overall configuration and initialization of the port structure which is used in the Dio module. These actions are done by the PORT Driver Module.⌋

[SWS_Dio_00063] ⌈The Dio module shall adapt its configuration and usage to the microcontroller and ECU.⌋

[SWS_Dio_00102] ⌈The Dio module's user shall only use the Dio functions after the Port Driver has been initialized. Otherwise the Dio module will exhibit undefined behavior.⌋

5.2 Default Error Tracer

[SWS_Dio_00194]

Upstream requirements: SRS_BSW_00350

⌈Dio.c shall include Det.h if detection of development error (DET) is enabled.⌋

6 Requirements Tracing

This chapter refers to 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 DIO driver SWS document, that satisfy the input requirements. Only functional requirements are referenced.

Requirement	Description	Satisfied by
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_Dio_00074] [SWS_Dio_00075] [SWS_Dio_00114]
[SRS_BSW_00344]	BSW Modules shall support link-time configuration	[SWS_Dio_00001] [SWS_Dio_00002]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[SWS_Dio_00194]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_Dio_00074] [SWS_Dio_00075] [SWS_Dio_00114]
[SRS_BSW_00396]	The Basic Software Module specifications shall specify the supported configuration classes for changing values and multiplicities for each parameter/container	[SWS_Dio_00129]
[SRS_Dio_12003]	The DIO Driver shall provide a service that writes a data word to the assigned DIO port	[SWS_Dio_00004] [SWS_Dio_00007] [SWS_Dio_00034] [SWS_Dio_00035] [SWS_Dio_00051] [SWS_Dio_00053] [SWS_Dio_00089] [SWS_Dio_00105] [SWS_Dio_00108] [SWS_Dio_00200] [SWS_Dio_00201] [SWS_Dio_00202] [SWS_Dio_00203] [SWS_Dio_00204]
[SRS_Dio_12004]	The DIO Driver shall provide a service that writes a selectable number of adjoining bits to an assigned part of a DIO port	[SWS_Dio_00008] [SWS_Dio_00039] [SWS_Dio_00040] [SWS_Dio_00051] [SWS_Dio_00056] [SWS_Dio_00089] [SWS_Dio_00090] [SWS_Dio_00091] [SWS_Dio_00184]
[SRS_Dio_12005]	The DIO Driver shall provide a service for write access to single DIO channels	[SWS_Dio_00006] [SWS_Dio_00028] [SWS_Dio_00029] [SWS_Dio_00051] [SWS_Dio_00079] [SWS_Dio_00089] [SWS_Dio_00127] [SWS_Dio_00128]
[SRS_Dio_12006]	The DIO Driver shall provide a service for reading a data word from the assigned DIO port	[SWS_Dio_00013] [SWS_Dio_00031] [SWS_Dio_00051] [SWS_Dio_00053] [SWS_Dio_00089] [SWS_Dio_00104]
[SRS_Dio_12007]	The DIO Driver shall provide a service for reading a selectable number of adjoining bits from an assigned part of a DIO port	[SWS_Dio_00014] [SWS_Dio_00037] [SWS_Dio_00051] [SWS_Dio_00056] [SWS_Dio_00089] [SWS_Dio_00092] [SWS_Dio_00093] [SWS_Dio_00184]
[SRS_Dio_12008]	The DIO Driver shall provide a service for reading one bit of an assigned DIO channel	[SWS_Dio_00011] [SWS_Dio_00027] [SWS_Dio_00051] [SWS_Dio_00089] [SWS_Dio_00127] [SWS_Dio_00128]
[SRS_Dio_12352]	The DIO driver shall allow reading from and writing to DIO ports, channel groups and channels	[SWS_Dio_00012] [SWS_Dio_00064] [SWS_Dio_00070] [SWS_Dio_00083] [SWS_Dio_00084] [SWS_Dio_00109]



△

Requirement	Description	Satisfied by
[SRS_Dio_12355]	Symbolic names shall be configured	[SWS_Dio_00017] [SWS_Dio_00020] [SWS_Dio_00022] [SWS_Dio_00026]
[SRS_Dio_12424]	Provide atomicity of DIO access	[SWS_Dio_00005] [SWS_Dio_00060]
[SRS_Dio_12900]	The DIO Driver shall provide a service to flip	[SWS_Dio_00191] [SWS_Dio_00192] [SWS_Dio_00193]
[SRS_SPAL_12064]	All driver modules shall raise an error if the change of the operation mode leads to degradation of running operations	[SWS_Dio_00001] [SWS_Dio_00002]
[SRS_SPAL_12263]	The implementation of all driver modules shall allow the configuration of specific module parameter types at link time	[SWS_Dio_00017] [SWS_Dio_00020] [SWS_Dio_00022]
[SRS_SPAL_12448]	All driver modules shall have a specific behavior after a development error detection	[SWS_Dio_00074] [SWS_Dio_00075] [SWS_Dio_00114] [SWS_Dio_00119]
[SRS_SPAL_12461]	Specific rules regarding initialization of controller registers shall apply to all driver implementations	[SWS_Dio_00001] [SWS_Dio_00002]
[SRS_SPAL_12462]	The register initialization settings shall be published	[SWS_Dio_00001] [SWS_Dio_00002]
[SRS_SPAL_12463]	The register initialization settings shall be combined and forwarded	[SWS_Dio_00001] [SWS_Dio_00002]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 General Behaviour

7.1.1 Background & Rationale

The DIO Driver abstracts the access to the microcontroller's hardware pins. Furthermore, it allows the grouping of those pins.

7.1.2 Requirements

The Dio SWS shall define functions allowing

- Port-
- Channel-
- Channel-group -

-based read and write access to the internal general purpose I/O ports.

[SWS_Dio_00051]
Upstream requirements: SRS_Dio_12003, SRS_Dio_12004, SRS_Dio_12005, SRS_Dio_12006, SRS_Dio_12007, SRS_Dio_12008

⌈The Dio module shall not buffer data when providing read and write services.

The Dio SWS shall define synchronous read / write services.⌋

[SWS_Dio_00005]
Upstream requirements: SRS_Dio_12424

⌈The Dio module's read and write services shall ensure for all services, that the data is consistent (Interruptible read-modify-write sequences are not allowed).⌋

[SWS_Dio_00089]
Upstream requirements: SRS_Dio_12003, SRS_Dio_12004, SRS_Dio_12005, SRS_Dio_12006, SRS_Dio_12007, SRS_Dio_12008

⌈Values used by the DIO Driver for the software level of Channels are either STD_HIGH or STD_LOW.⌋

[SWS_Dio_00128]
Upstream requirements: SRS_Dio_12005, SRS_Dio_12008

⌈A general-purpose digital IO pin represents a DIO channel.⌋

[SWS_Dio_00127]

Upstream requirements: SRS_Dio_12005, SRS_Dio_12008

⌈The Port module shall configure a DIO channel as input or output ([SWS_Dio_00001] and [SWS_Dio_00002]).⌋

[SWS_Dio_00053]

Upstream requirements: SRS_Dio_12003, SRS_Dio_12006

⌈In the DIO Driver, it shall be possible to group several DIO channels by hardware (typically controlled by one hardware register) to represent a DIO port.⌋

Note: The single DIO channel levels inside a DIO port represent a bit in the DIO port value, depending on their position inside the port.

[SWS_Dio_00056]

Upstream requirements: SRS_Dio_12004, SRS_Dio_12007

⌈A channel group is a formal logical combination of several adjoining DIO channels within a DIO port.⌋

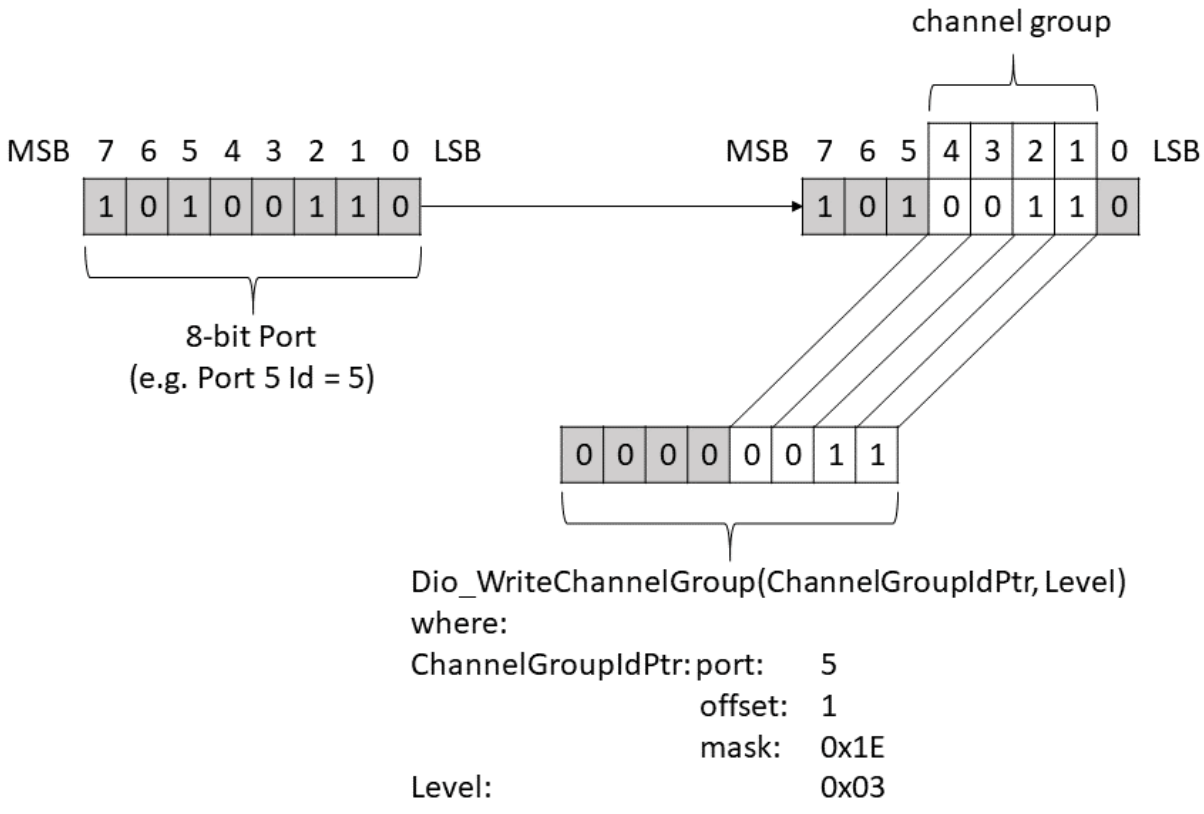


Figure 7.1: Schematic description of a ChannelGroup

The DIO Driver provides the following services:

- The Dio SWS shall define functions to modify the levels of output channels individually, for a port or for a channel group.

- The Dio SWS shall define functions to read the level of input and output (see [SWS_Dio_00083]) channels individually, for a port or for a channel group.

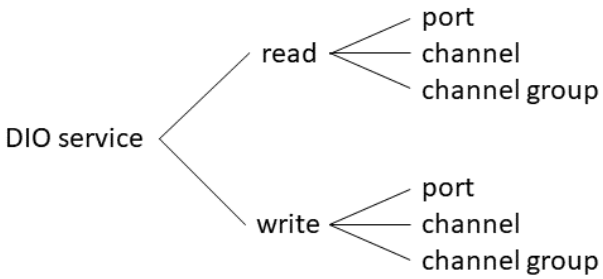


Figure 7.2: DIO Services

[SWS_Dio_00060]

Upstream requirements: SRS_Dio_12424

⌈All read and write functions of the Dio module shall be re-entrant.

Reason: The DIO Driver may be accessed by different upper layer handlers or drivers. These upper layer modules may access the driver concurrently.⌋

[SWS_Dio_00026]

Upstream requirements: SRS_Dio_12355

⌈The configuration process for Dio module shall provide symbolic names for each configured DIO channel, port and group.⌋

7.2 Initialization

7.2.1 Background & Rationale

Initialization of the hardware is done by the PORT Driver.

7.2.2 Requirements

[SWS_Dio_00001]

Upstream requirements: SRS_BSW_00344, SRS_SPAL_12064, SRS_SPAL_12461, SRS_SPAL_12462, SRS_SPAL_12463

⌈The Dio module shall not provide an interface for initialization of the hardware. The Port Driver performs this.⌋

7.3 Runtime Reconfiguration

7.3.1 Background & Rationale

Runtime reconfiguration is provided by the PORT Driver.

7.3.2 Requirements

[SWS_Dio_00002]

Upstream requirements: SRS_BSW_00344, SRS_SPAL_12064, SRS_SPAL_12461, SRS_SPAL_12462, SRS_SPAL_12463

⌈The PORT driver shall provide the reconfiguration of the port pin direction during runtime.⌋

7.4 DIO Write Service

7.4.1 Background & Rationale

The DIO Driver provides services to transfer data to the microcontroller's pins.

7.4.2 Requirements

[SWS_Dio_00064]

Upstream requirements: SRS_Dio_12352

⌈The Dio module's write functions shall work on input and output channels.⌋

[SWS_Dio_00070]

Upstream requirements: SRS_Dio_12352

⌈If a Dio write function is used on an input channel, it shall have no effect on the physical output level.⌋

[SWS_Dio_00109]

Upstream requirements: SRS_Dio_12352

⌈If supported by hardware, the Dio module shall set / clear the output data latch of an input channel so that the required level is output from the pin when the port driver configures the pin as a DIO output pin.⌋

[SWS_Dio_00119]

Upstream requirements: SRS_SPAL_12448

⌈If development errors are enabled and an error occurred, the Dio module's write functions shall NOT process the write command.⌋

7.4.2.1 DIO Channel Write Service

[SWS_Dio_00006]

Upstream requirements: SRS_Dio_12005

⌈The `Dio_WriteChannel` function shall set the level of a single DIO channel to STD_HIGH or STD_LOW.⌋

7.4.2.2 DIO Port Write Service

[SWS_Dio_00007]

Upstream requirements: SRS_Dio_12003

⌈The `Dio_WritePort` function shall simultaneously set the levels of all output channels. A bit value '0' sets the corresponding channel to physical STD_LOW, a bit value '1' sets the corresponding channel to physical STD_HIGH.⌋

[SWS_Dio_00004]

Upstream requirements: SRS_Dio_12003

⌈The `Dio_WritePort` function shall ensure that the functionality of the input channels of that port is not affected.⌋

7.4.2.3 DIO Channel Group Write Service

[SWS_Dio_00008]

Upstream requirements: SRS_Dio_12004

⌈The `Dio_WriteChannelGroup` function shall simultaneously set an adjoining subset of DIO channels (channel group). A bit value '0' sets the corresponding channel to physical STD_LOW, a bit value '1' sets the corresponding channel to physical STD_HIGH.⌋

7.4.2.4 DIO Masked Port Write Service

[SWS_Dio_00200]

Upstream requirements: SRS_Dio_12003

⌈The `Dio_MaskedWritePort` function shall simultaneously set the levels of the selected output channels.

Mask argument specifies which bits are selected: a bit value '0' means the corresponding channel is not selected, a bit value '1' means the corresponding channel is selected.

Level argument specifies the physical levels: a bit value '0' sets the corresponding channel to physical STD_LOW, a bit value '1' sets the corresponding channel to physical STD_HIGH.⌋

[SWS_Dio_00201]

Upstream requirements: SRS_Dio_12003

⌈The `Dio_MaskedWritePort` function shall ensure that the functionality of the input channels of that port is not affected.⌋

7.5 DIO Read Service

7.5.1 Background & Rationale

The DIO Driver provides services to transfer data from the microcontroller's pins.

7.5.2 Requirements

[SWS_Dio_00012]

Upstream requirements: SRS_Dio_12352

⌈The Dio module's read functions shall work on input and output channels.⌋

7.5.2.1 DIO Channel Read Service

[SWS_Dio_00011]

Upstream requirements: SRS_Dio_12008

⌈The `Dio_ReadChannel` function shall read the level of a single DIO channel.⌋

7.5.2.2 DIO Port Read Service

[SWS_Dio_00013]

Upstream requirements: SRS_Dio_12006

⌈The `Dio_ReadPort` function shall read the levels of all channels of one port. A bit value '0' indicates that the corresponding channel is physical STD_LOW, a bit value '1' indicates that the corresponding channel is physical STD_HIGH.⌋

7.5.2.3 DIO Channel Group Read Service

[SWS_Dio_00014]

Upstream requirements: SRS_Dio_12007

⌈The `Dio_ReadChannelGroup` function shall read the levels of a DIO channel group. A bit value '0' indicates that the corresponding channel is physical STD_LOW, a bit value '1' indicates that the corresponding channel is physical STD_HIGH.⌋

7.5.2.4 DIO Readback of Output Pins

[SWS_Dio_00083]

Upstream requirements: SRS_Dio_12352

⌈If the microcontroller supports the direct read-back of a pin value, the Dio module's read functions shall provide the real pin level, when they are used on a channel which is configured as an output channel.⌋

[SWS_Dio_00084]

Upstream requirements: SRS_Dio_12352

⌈If the microcontroller does not support the direct read-back of a pin value, the Dio module's read functions shall provide the value of the output register, when they are used on a channel which is configured as an output channel.⌋

7.6 Error Classification

Chapter [2, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.6.1 Development Errors

[SWS_Dio_91001] Definition of development errors in module Dio ⌈

Type of error	*Related error code*	*Error value*
Invalid channel requested	DIO_E_PARAM_INVALID_CHANNEL_ID	0x0A
Invalid port requested	DIO_E_PARAM_INVALID_PORT_ID	0x14
Invalid channel group requested	DIO_E_PARAM_INVALID_GROUP	0x1F

▽

△

Type of error	*Related error code*	*Error value*
API service called with a NULL pointer	DIO_E_PARAM_POINTER	0x20

⌋

7.6.2 Runtime Errors

There are no runtime errors.

7.6.3 Production Errors

There are no production errors.

7.7 Security Events

The module does not report security events.

8 API specification

8.1 Imported types

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

[SWS_Dio_00131] Definition of imported datatypes of module Dio ⌈

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

⌋

8.2 Type definitions

[SWS_Dio_00103] ⌈The port width within the types defined for the DIO Driver shall be the size of the largest port on the MCU which may be accessed by the DIO Driver.⌋

8.2.1 Dio_ChannelType

[SWS_Dio_00182] Definition of datatype Dio_ChannelType ⌈

Name	Dio_ChannelType		
Kind	Type		
Derived from	uint		
Range	This is implementation specific but not all values may be valid within the type.	–	Shall cover all available DIO channels
Description	Numeric ID of a DIO channel.		
Available via	Dio.h		

⌋

[SWS_Dio_00180] ⌈The mapping of the ID is implementation specific but not configurable.⌋

[SWS_Dio_00017]

Upstream requirements: SRS_SPAL_12263, SRS_Dio_12355

⌈For parameter values of type `Dio_ChannelType`, the Dio's user shall use the symbolic names provided by the configuration description.

Furthermore, [SWS_Dio_00103] applies to the type `Dio_ChannelType`.⌋

8.2.2 Dio_PortType

[SWS_Dio_00183] Definition of datatype Dio_PortType ⌈

Name	Dio_PortType		
Kind	Type		
Derived from	uint		
Range	0..<number of ports>	–	Shall cover all available DIO Ports.
Description	Numeric ID of a DIO port.		
Available via	Dio.h		

⌋

[SWS_Dio_00181] ⌈The mapping of ID is implementation specific but not configurable.⌋

[SWS_Dio_00020]

Upstream requirements: SRS_SPAL_12263, SRS_Dio_12355

⌈For parameter values of type `Dio_PortType`, the user shall use the symbolic names provided by the configuration description.

Furthermore, [SWS_Dio_00103] applies to the type `Dio_PortType`.⌋

8.2.3 Dio_ChannelGroupType

[SWS_Dio_00184] Definition of datatype Dio_ChannelGroupType

Upstream requirements: SRS_Dio_12004, SRS_Dio_12007

⌈

Name	Dio_ChannelGroupType	
Kind	Structure	
Elements	mask	
	Type	uint8/16/32
	Comment	This element mask which defines the positions of the channel group.
	offset	
	Type	uint8
	Comment	This element shall be the position of the Channel Group on the port, counted from the LSB.
	port	
	Type	Dio_PortType
	Comment	This shall be the port on which the Channel group is defined.
Description	Type for the definition of a channel group, which consists of several adjoining channels within a port.	
Available via	Dio.h	

⌋

[SWS_Dio_00022]

Upstream requirements: SRS_SPAL_12263, SRS_Dio_12355

⌈For parameter values of type `Dio_ChannelGroupType`, the user shall use the symbolic names provided by the configuration description.

Furthermore, [SWS_Dio_00056] applies to the type `Dio_ChannelGroupType`.⌋

8.2.4 Dio_LevelType

[SWS_Dio_00185] Definition of datatype Dio_LevelType ⌈

Name	Dio_LevelType
Kind	Type
Derived from	uint8
Description	These are the possible levels a DIO channel can have (input or output). The symbols STD_LOW (Physical state 0V) and STD_HIGH (Physical state 5V or 3.3V) shall be reused from Std_Types.h.
Available via	Dio.h

⌋

8.2.5 Dio_PortLevelType

[SWS_Dio_00186] Definition of datatype Dio_PortLevelType ⌈

Name	Dio_PortLevelType		
Kind	Type		
Derived from	uint		
Range	0...xxx	–	If the µC owns ports of different port widths (e.g. 4, 8,16...Bit) Dio_PortLevelType inherits the size of the largest port
Description	If the µC owns ports of different port widths (e.g. 4, 8,16...Bit) Dio_PortLevelType inherits the size of the largest port.		
Available via	Dio.h		

⌋

[SWS_Dio_00024] ⌈`Dio_PortLevelType` is the type for the value of a DIO port.

Furthermore, [SWS_Dio_00103] applies to the type `Dio_PortLevelType`.⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 Dio_ReadChannel

[SWS_Dio_00133] Definition of API function Dio_ReadChannel ⌈

Service Name	Dio_ReadChannel	
Syntax	`Dio_LevelType Dio_ReadChannel ( Dio_ChannelType ChannelId )`	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ChannelId	ID of DIO channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Dio_LevelType	`STD_HIGH` The physical level of the corresponding Pin is `STD_HIGH` `STD_LOW` The physical level of the corresponding Pin is `STD_LOW`
Description	Returns the value of the specified DIO channel.	
Available via	Dio.h	

⌋

[SWS_Dio_00027]

Upstream requirements: SRS_Dio_12008

⌈The `Dio_ReadChannel` function shall return the value of the specified DIO channel.

Regarding the return value of the `Dio_ReadChannel` function, the requirements [SWS_Dio_00083] and [SWS_Dio_00084] are applicable.

Furthermore, the requirements [SWS_Dio_00005] and [SWS_Dio_00026] are applicable to the `Dio_ReadChannel` function.⌋

[SWS_Dio_00074]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386, SRS_SPAL_12448

⌈If development error detection is enabled, the services `Dio_ReadChannel`, `Dio_WriteChannel` and `Dio_FlipChannel` shall check the "ChannelId" parameter to be valid within the current configuration. If the "ChannelId" parameter is invalid, the functions shall report the error code `DIO_E_PARAM_INVALID_CHANNEL_ID` to the DET.⌋

8.3.2 Dio_WriteChannel

[SWS_Dio_00134] Definition of API function Dio_WriteChannel ⌈

Service Name	Dio_WriteChannel	
Syntax	`void Dio_WriteChannel ( Dio_ChannelType ChannelId, Dio_LevelType Level )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ChannelId	ID of DIO channel
	Level	Value to be written
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service to set a level of a channel.	
Available via	Dio.h	

⌋

[SWS_Dio_00028]

Upstream requirements: SRS_Dio_12005

⌈If the specified channel is configured as an output channel, the `Dio_WriteChannel` function shall set the specified Level for the specified channel.⌋

[SWS_Dio_00029]

Upstream requirements: SRS_Dio_12005

⌈If the specified channel is configured as an input channel, the `Dio_WriteChannel` function shall have no influence on the physical output.⌋

[SWS_Dio_00079]

Upstream requirements: SRS_Dio_12005

⌈If the specified channel is configured as an input channel, the `Dio_WriteChannel` function shall have no influence on the result of the next Read-Service.

Furthermore, the requirements [SWS_Dio_00005], [SWS_Dio_00119] and [SWS_Dio_00026] are applicable to the `Dio_WriteChannel` function.⌋

8.3.3 Dio_ReadPort

[SWS_Dio_00135] Definition of API function Dio_ReadPort ⌈

Service Name	Dio_ReadPort	
Syntax	`Dio_PortLevelType Dio_ReadPort ( Dio_PortType PortId )`	
Service ID [hex]	0x02	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	PortId	ID of DIO Port
Parameters (inout)	None	
Parameters (out)	None	
Return value	Dio_PortLevelType	Level of all channels of that port
Description	Returns the level of all channels of that port.	
Available via	Dio.h	

⌋

[SWS_Dio_00031]

Upstream requirements: SRS_Dio_12006

⌈The `Dio_ReadPort` function shall return the level of all channels of that port.⌋

[SWS_Dio_00104]

Upstream requirements: SRS_Dio_12006

⌈When reading a port which is smaller than the `Dio_PortLevelType` using the `Dio_ReadPort` function (see [SWS_Dio_00103]), the function shall set the bits corresponding to undefined port pins to 0.

Furthermore, the requirements [SWS_Dio_00005] and [SWS_Dio_00026] are applicable to the `Dio_ReadPort` function.⌋

[SWS_Dio_00075]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386, SRS_SPAL_12448

⌈If development error detection is enabled, the functions `Dio_ReadPort`, `Dio_WritePort` and `Dio_MaskedWritePort` shall check the "PortId" parameter to be valid within the current configuration. If the "PortId" parameter is invalid, the functions shall report the error code `DIO_E_PARAM_INVALID_PORT_ID` to the DET.⌋

8.3.4 Dio_WritePort

[SWS_Dio_00136] Definition of API function Dio_WritePort ⌈

Service Name	Dio_WritePort	
Syntax	`void Dio_WritePort ( Dio_PortType PortId, Dio_PortLevelType Level )`	
Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	PortId	ID of DIO Port
	Level	Value to be written
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service to set a value of the port.	
Available via	Dio.h	

⌋

[SWS_Dio_00034]

Upstream requirements: SRS_Dio_12003

⌈The `Dio_WritePort` function shall set the specified value for the specified port.⌋

[SWS_Dio_00035]

Upstream requirements: SRS_Dio_12003

⌈When the `Dio_WritePort` function is called, DIO Channels that are configured as input shall remain unchanged.⌋

[SWS_Dio_00105]

Upstream requirements: SRS_Dio_12003

⌈When writing a port which is smaller than the `Dio_PortLevelType` using the `Dio_WritePort` function (see [SWS_Dio_00103]), the function shall ignore the MSB.⌋

[SWS_Dio_00108]

Upstream requirements: SRS_Dio_12003

⌈The `Dio_WritePort` function shall have no effect on channels within this port which are configured as input channels.

Furthermore, the requirements [SWS_Dio_00005], [SWS_Dio_00119] and [SWS_Dio_00026] are applicable to the `Dio_WritePort` function.⌋

8.3.5 Dio_ReadChannelGroup

[SWS_Dio_00137] Definition of API function Dio_ReadChannelGroup ⌈

Service Name	Dio_ReadChannelGroup	
Syntax	`Dio_PortLevelType Dio_ReadChannelGroup ( const Dio_ChannelGroupType* ChannelGroupIdPtr )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ChannelGroupIdPtr	Pointer to ChannelGroup
Parameters (inout)	None	
Parameters (out)	None	
Return value	Dio_PortLevelType	Level of a subset of the adjoining bits of a port
Description	This Service reads a subset of the adjoining bits of a port.	
Available via	Dio.h	

⌋

[SWS_Dio_00037]

Upstream requirements: SRS_Dio_12007

⌈The `Dio_ReadChannelGroup` function shall read a subset of the adjoining bits of a port (channel group).⌋

[SWS_Dio_00092]

Upstream requirements: SRS_Dio_12007

⌈The `Dio_ReadChannelGroup` function shall do the masking of the channel group.⌋

[SWS_Dio_00093]

Upstream requirements: SRS_Dio_12007

⌈The `Dio_ReadChannelGroup` function shall do the shifting so that the values read by the function are aligned to the LSB.

Furthermore, the requirements [SWS_Dio_00005], [SWS_Dio_00056], [SWS_Dio_00083], [SWS_Dio_00084] and [SWS_Dio_00026] are applicable to the `Dio_ReadChannelGroup` function.⌋

[SWS_Dio_00114]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386, SRS_SPAL_12448

⌈If development error detection is enabled, the functions `Dio_ReadChannelGroup` and `Dio_WriteChannelGroup` shall check the "ChannelGroupIdPtr" parameter to be valid within the current configuration. If the "ChannelGroupIdPtr" parameter is invalid, the functions shall report the error code `DIO_E_PARAM_INVALID_GROUP` to the DET.⌋

8.3.6 Dio_WriteChannelGroup

[SWS_Dio_00138] Definition of API function Dio_WriteChannelGroup ⌈

Service Name	Dio_WriteChannelGroup	
Syntax	`void Dio_WriteChannelGroup ( const Dio_ChannelGroupType* ChannelGroupIdPtr, Dio_PortLevelType Level )`	
Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ChannelGroupIdPtr	Pointer to ChannelGroup
	Level	Value to be written
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service to set a subset of the adjoining bits of a port to a specified level.	
Available via	Dio.h	

⌋

[SWS_Dio_00039]

Upstream requirements: SRS_Dio_12004

⌈The `Dio_WriteChannelGroup` function shall set a subset of the adjoining bits of a port (channel group) to a specified level.⌋

[SWS_Dio_00040]

Upstream requirements: SRS_Dio_12004

⌈The `Dio_WriteChannelGroup` shall not change the remaining channels of the port and channels which are configured as input.⌋

[SWS_Dio_00090]

Upstream requirements: SRS_Dio_12004

⌈The `Dio_WriteChannelGroup` function shall do the masking of the channel group.⌋

[SWS_Dio_00091]

Upstream requirements: SRS_Dio_12004

⌈The function `Dio_WriteChannelGroup` shall do the shifting so that the values written by the function are aligned to the LSB.

Furthermore, the requirements [SWS_Dio_00005], [SWS_Dio_00056], [SWS_Dio_00119] and [SWS_Dio_00026] are applicable for the `Dio_WriteChannelGroup` function.⌋

8.3.7 Dio_FlipChannel

[SWS_Dio_00190] Definition of API function Dio_FlipChannel ⌈

Service Name	Dio_FlipChannel	
Syntax	`Dio_LevelType Dio_FlipChannel ( Dio_ChannelType ChannelId )`	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	ChannelId	ID of DIO channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	Dio_LevelType	`STD_HIGH`: The physical level of the corresponding Pin is `STD_HIGH`. `STD_LOW`: The physical level of the corresponding Pin is `STD_LOW`.
Description	Service to flip (change from 1 to 0 or from 0 to 1) the level of a channel and return the level of the channel after flip.	
Available via	Dio.h	

⌋

[SWS_Dio_00191]

Upstream requirements: SRS_Dio_12900

⌈If the specified channel is configured as an output channel, the `Dio_FlipChannel` function shall read level of the channel (requirements [SWS_Dio_00083] and [SWS_Dio_00084] are applicable) and invert it, then write the inverted level to the channel. The return value shall be the inverted level of the specified channel.⌋

[SWS_Dio_00192]

Upstream requirements: SRS_Dio_12900

⌈If the specified channel is configured as an input channel, the `Dio_FlipChannel` function shall have no influence on the physical output.

The return value shall be the level of the specified channel.⌋

[SWS_Dio_00193]

Upstream requirements: SRS_Dio_12900

⌈If the specified channel is configured as an input channel, the `Dio_FlipChannel` function shall have no influence on the result of the next Read-Service.

Furthermore, the requirements [SWS_Dio_00005], [SWS_Dio_00119] and [SWS_Dio_00026] are applicable to the `Dio_FlipChannel` function.⌋

8.3.8 Dio_MaskedWritePort

[SWS_Dio_00300] Definition of API function Dio_MaskedWritePort ⌈

Service Name	Dio_MaskedWritePort	
Syntax	`void Dio_MaskedWritePort ( Dio_PortType PortId, Dio_PortLevelType Level, Dio_PortLevelType Mask )`	
Service ID [hex]	0x13	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	PortId	ID of DIO Port
	Level	Value to be written
	Mask	Channels to be masked in the port
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service to set the value of a given port with required mask.	
Available via	Dio.h	

⌋

[SWS_Dio_00202]

Upstream requirements: SRS_Dio_12003

⌈The `Dio_MaskedWritePort` function shall set the specified value for the channels in the specified port if the corresponding bit in Mask is '1'.⌋

[SWS_Dio_00203]

Upstream requirements: SRS_Dio_12003

⌈When the `Dio_MaskedWritePort` function is called, DIO Channels that are configured as input shall remain unchanged.⌋

[SWS_Dio_00204]

Upstream requirements: SRS_Dio_12003

⌈When writing a port which is smaller than the `Dio_PortLevelType` using the `Dio_MaskedWritePort` function (see [SWS_Dio_00103]), the function shall ignore the MSB.⌋

8.4 Scheduled functions

This chapter lists all functions called directly by the Basic Software Module Scheduler.

The Dio module has no scheduled functions.

8.5 Expected interfaces

This chapter lists all functions the Dio module requires from other modules.

8.5.1 Mandatory Interfaces

None

8.5.2 Optional Interfaces

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

[SWS_Dio_00140] Definition of optional interfaces requested by module Dio ⌈

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

⌋

9 Sequence diagrams

The diagrams below show the sequences when calling the `Dio_ReadChannel` and `Dio_WriteChannel` service, they show normal operation mode. Since all other services which are defined in Chapter 8.3 have exactly the same synchronous behavior concerning, there are intentionally no further sequence diagrams in this document.

9.1 Read a value from a digital I/O

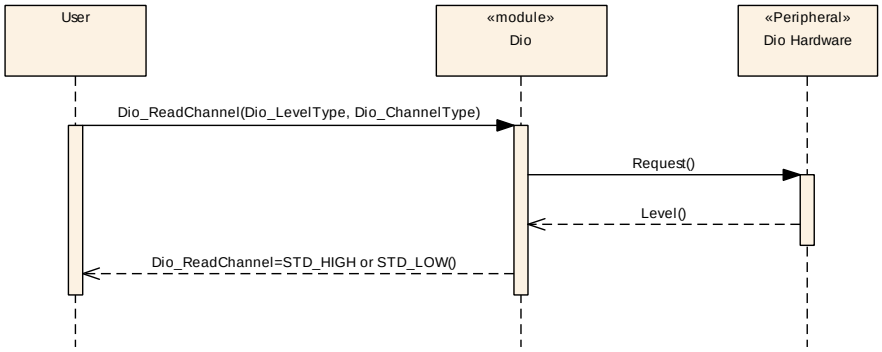


Figure 9.1: Read Service Sequence Chart (normal operation mode)

9.2 Write a value to a digital I/O

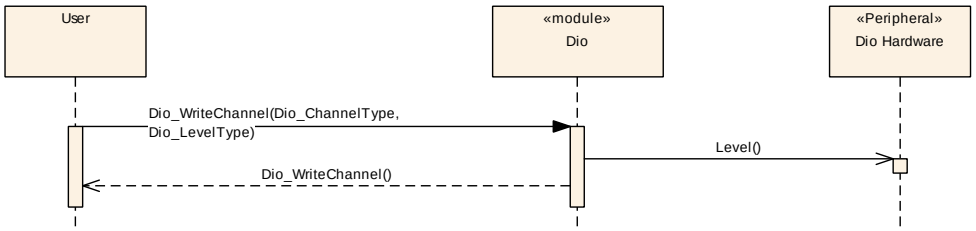


Figure 9.2: Write Service Sequence Chart (normal operation mode)

10 Configuration specification

This chapter defines configuration parameters and their clustering into containers.

10.1 Containers and configuration parameters

The following chapters summarize all configuration parameters.

[SWS_Dio_00205] ⌈The DIO module shall reject configurations with partition mappings which are not supported by the implementation.⌋

10.1.1 Variants

[SWS_Dio_00129]

Upstream requirements: SRS_BSW_00396

⌈At least one of the following variants has to be supported by implementation: VARIANT-PRE-COMPILE, VARIANT-LINK-TIME.⌋

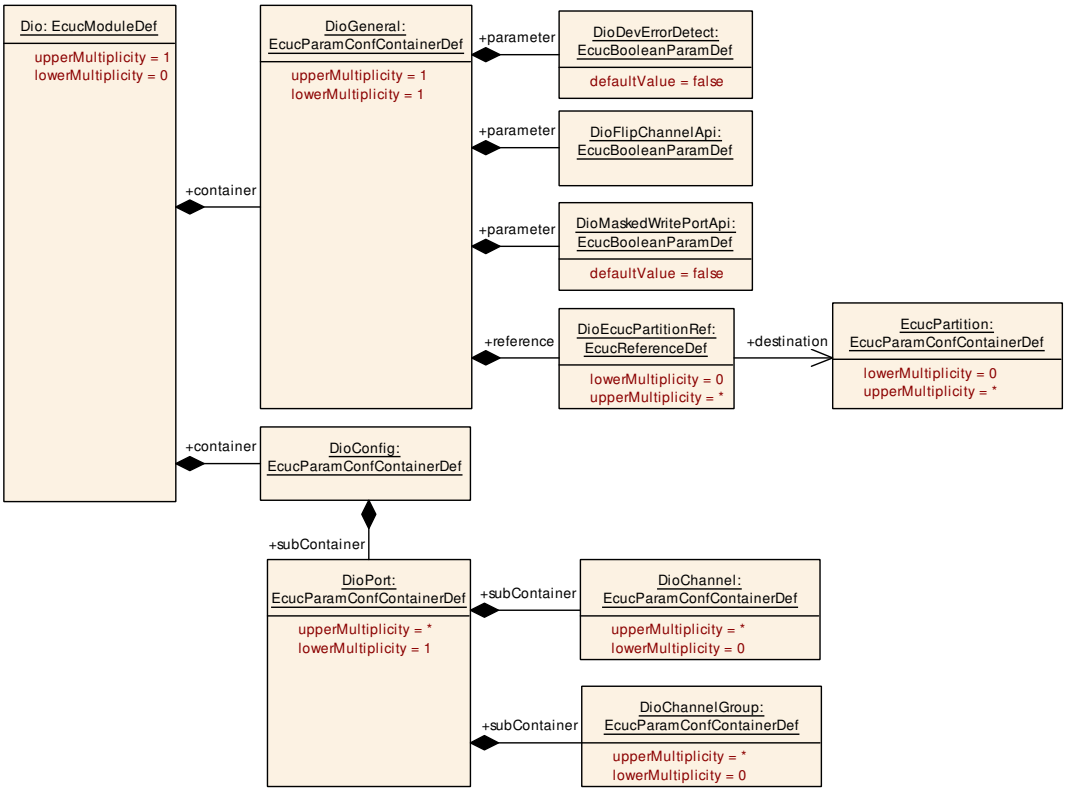


Figure 10.1: DIO Module Configuration

10.1.2 Dio

[ECUC_Dio_00154] Definition of EcucModuleDef Dio ⌈

Module Name	Dio
Description	Configuration of the Dio (Digital IO) module.
Post-Build Selectable Support	false
Post-Build Loadable Support	false

Included Containers		
Container Name	**Multiplicity**	**Description**
DioConfig	1	This container contains the configuration parameters and sub containers of the AUTOSAR DIO module.
DioGeneral	1	General DIO module configuration parameters.

⌋

10.1.3 DioGeneral

[ECUC_Dio_00141] Definition of EcucParamConfContainerDef DioGeneral ⌈

Container Name	DioGeneral
Parent Container	Dio
Upstream mapping defined	–
Description	General DIO module configuration parameters.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DioDevErrorDetect	1	[ECUC_Dio_00142]
DioFlipChannelApi	1	[ECUC_Dio_00153]
DioMaskedWritePortApi	1	[ECUC_Dio_00155]
DioEcucPartitionRef	0..*	[ECUC_Dio_00156]

No Included Containers

⌋

[ECUC_Dio_00142] Definition of EcucBooleanParamDef DioDevErrorDetect ⌈

Parameter Name	DioDevErrorDetect
Parent Container	DioGeneral
Upstream mapping defined	–

▽

△

Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_Dio_00153] Definition of EcucBooleanParamDef DioFlipChannelApi ⌈

Parameter Name	DioFlipChannelApi
Parent Container	DioGeneral
Upstream mapping defined	–
Description	Adds / removes the service Dio_FlipChannel() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Dio_00155] Definition of EcucBooleanParamDef DioMaskedWritePortApi ⌈

Parameter Name	DioMaskedWritePortApi
Parent Container	DioGeneral
Upstream mapping defined	–
Description	Adds / removes the service Dio_MaskedWritePort() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_Dio_00156] Definition of EcucReferenceDef DioEcucPartitionRef ⌈

Parameter Name	DioEcucPartitionRef
Parent Container	DioGeneral
Upstream mapping defined	–
Description	Maps the DIO driver to zero or multiple ECUC partitions to make the modules API available in this partition.
Multiplicity	0..*
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

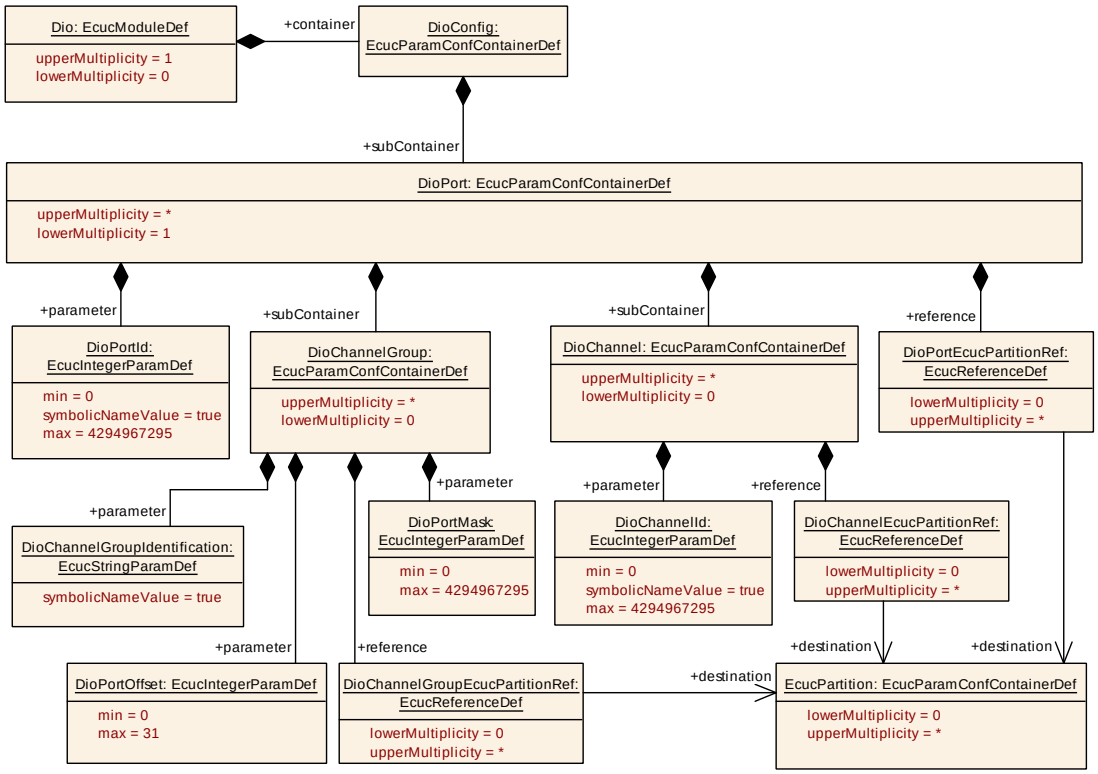


Figure 10.2: DioPort Configuration Container

10.1.4 DioPort

[ECUC_Dio_00144] Definition of EcucParamConfContainerDef DioPort ⌈

Container Name	DioPort
Parent Container	DioConfig
Upstream mapping defined	–
Description	Configuration of individual DIO ports, consisting of channels and possible channel groups. Note that this container definition does not explicitly define a symbolic name parameter. Instead, the container's short name will be used in the Ecu Configuration Description to specify the symbolic name of the port.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DioPortId	1	[ECUC_Dio_00145]
DioPortEcucPartitionRef	0..*	[ECUC_Dio_00157]

Included Containers		
Container Name	**Multiplicity**	**Description**
DioChannel	0..*	Configuration of an individual DIO channel.
DioChannelGroup	0..*	Definition and configuration of DIO channel groups. A channel group represents several adjoining DIO channels represented by a logical group. Note that this container definition does not explicitly define a symbolic name parameter. Instead, the container's short name will be used in the Ecu Configuration Description to specify the symbolic name of the channel group.

⌋

[ECUC_Dio_00145] Definition of EcucIntegerParamDef DioPortId ⌈

Parameter Name	DioPortId
Parent Container	DioPort
Upstream mapping defined	–
Description	Numeric identifier of the DIO port. Not all MCU ports may be used for DIO, thus there may be "gaps" in the list of all IDs. This value will be assigned to the DIO port symbolic name (i.e. the SHORT-NAME of the DioPort container).
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 4294967295
Default value	–
Pre-Compile only	true

⌋

[ECUC_Dio_00157] Definition of EcucReferenceDef DioPortEcucPartitionRef ⌈

Parameter Name	DioPortEcucPartitionRef
Parent Container	DioPort
Upstream mapping defined	–
Description	Maps the DIO ports to zero or multiple ECUC partitions. The ECUC partitions referenced are a subset of the ECUC partitions where the DIO driver is mapped to.
Multiplicity	0..*
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[SWS_Dio_00206] ⌈The ECUC partitions referenced by `DioPortEcucPartitionRef` shall be a subset of the ECUC partitions referenced by `DioEcucPartitionRef`.⌋

[SWS_Dio_00209] ⌈If `DioEcucPartitionRef` references one or more ECUC partitions, `DioPortEcucPartitionRef` shall have a multiplicity of greater than zero and reference one or several of these ECUC partitions as well.⌋

10.1.5 DioChannel

[ECUC_Dio_00146] Definition of EcucParamConfContainerDef DioChannel ⌈

Container Name	DioChannel
Parent Container	DioPort
Upstream mapping defined	–
Description	Configuration of an individual DIO channel.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
DioChannelId	1	[ECUC_Dio_00147]
DioChannelEcucPartitionRef	0..*	[ECUC_Dio_00158]

No Included Containers

⌋

[ECUC_Dio_00147] Definition of EcucIntegerParamDef DioChannelId ⌈

Parameter Name	DioChannelId	
Parent Container	DioChannel	
Upstream mapping defined	–	
Description	Channel Id of the DIO channel. This value will be assigned to the symbolic names.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dio_00158] Definition of EcucReferenceDef DioChannelEcucPartitionRef ⌈

Parameter Name	DioChannelEcucPartitionRef
Parent Container	DioChannel
Upstream mapping defined	–
Description	Maps a DIO channel to zero or multiple ECUC partitions. The ECUC partitions referenced are a subset of the ECUC partitions where the related DIO port is mapped to.
Multiplicity	0..*
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[SWS_Dio_00207] ⌈The ECUC partitions referenced by `DioChannelEcucPartitionRef` shall be a subset of the ECUC partitions referenced by `DioPortEcucPartitionRef`.⌋

[SWS_Dio_00210] ⌈If DioEcucPartitionRef references one or more ECUC partitions, DioChannelEcucPartitionRef shall have a multiplicity of greater than zero and reference one or several of these ECUC partitions as well.⌋

10.1.6 DioChannelGroup

[ECUC_Dio_00148] Definition of EcucParamConfContainerDef DioChannelGroup ⌈

Container Name	DioChannelGroup
Parent Container	DioPort
Upstream mapping defined	–
Description	Definition and configuration of DIO channel groups. A channel group represents several adjoining DIO channels represented by a logical group. Note that this container definition does not explicitly define a symbolic name parameter. Instead, the container's short name will be used in the Ecu Configuration Description to specify the symbolic name of the channel group.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
DioChannelGroupIdentification	1	[ECUC_Dio_00149]
DioPortMask	1	[ECUC_Dio_00150]
DioPortOffset	1	[ECUC_Dio_00151]
DioChannelGroupEcucPartitionRef	0..*	[ECUC_Dio_00159]

No Included Containers

⌋

[ECUC_Dio_00149] Definition of EcucStringParamDef DioChannelGroupIdentification ⌈

Parameter Name	DioChannelGroupIdentification
Parent Container	DioChannelGroup
Upstream mapping defined	–
Description	The DIO channel group is identified in DIO API by a pointer to a data structure (of type Dio_ChannelGroupType). That data structure contains the channel group information. This parameter contains the code fragment that has to be inserted in the API call of the calling module to get the address of the variable in memory which holds the channel group information. Example values are "&MyDioGroup1" or "&MyDioGroupArray[0]"
Multiplicity	1
Type	EcucStringParamDef (Symbolic Name generated for this parameter)
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_Dio_00150] Definition of EcucIntegerParamDef DioPortMask ⌈

Parameter Name	DioPortMask	
Parent Container	DioChannelGroup	
Upstream mapping defined	–	
Description	This shall be the mask which defines the positions of the channel group. The channels shall consist of adjoining bits in the same port. The data type depends on the port width.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dio_00151] Definition of EcucIntegerParamDef DioPortOffset ⌈

Parameter Name	DioPortOffset	
Parent Container	DioChannelGroup	
Upstream mapping defined	–	
Description	The position of the Channel Group on the port, counted from the LSB. This value can be derived from DioPortMask. calculationFormula = Position of the first bit of DioPortMask which is set to '1' counted from LSB	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 31	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Dio_00159] Definition of EcucReferenceDef DioChannelGroupEcucPartitionRef ⌈

Parameter Name	DioChannelGroupEcucPartitionRef
Parent Container	DioChannelGroup
Upstream mapping defined	–
Description	Maps a DIO channel group to zero or multiple ECUC partitions. The ECUC partitions referenced are a subset of the ECUC partitions where the related DIO port is mapped to.
Multiplicity	0..*
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[SWS_Dio_00208] ⌈The ECUC partitions referenced by `DioChannelGroupEcucPartitionRef` shall be a subset of the ECUC partitions referenced by `DioPortEcucPartitionRef`.⌋

[SWS_Dio_00211] ⌈If `DioPortEcucPartitionRef` references one or more ECUC partitions, `DioChannelGroupEcucPartitionRef` shall have a multiplicity of greater than zero and reference one or several of these ECUC partitions as well.⌋

10.1.7 DioConfig

[ECUC_Dio_00152] Definition of EcucParamConfContainerDef DioConfig ⌈

Container Name	DioConfig
Parent Container	Dio
Upstream mapping defined	–
Description	This container contains the configuration parameters and sub containers of the AUTOSAR DIO module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
DioPort	1..*	Configuration of individual DIO ports, consisting of channels and possible channel groups. Note that this container definition does not explicitly define a symbolic name parameter. Instead, the container's short name will be used in the Ecu Configuration Description to specify the symbolic name of the port.

⌋

10.2 Published Information

For details refer to [2] Chapter 10.3 *"Published Information"*.

A Not applicable requirements

[SWS_Dio_NA_00195]

Upstream requirements: SRS_BSW_00004, SRS_BSW_00101, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00336, SRS_BSW_00339, SRS_BSW_00369, SRS_BSW_00375, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00403, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00406, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00461, SRS_BSW_00469, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00478, SRS_BSW_00490, SRS_BSW_00491, SRS_SPAL_00157, SRS_SPAL_12057, SRS_SPAL_12063, SRS_SPAL_12067, SRS_SPAL_12068, SRS_SPAL_12069, SRS_SPAL_12075, SRS_SPAL_12077, SRS_SPAL_12078, SRS_SPAL_12092, SRS_SPAL_12125, SRS_SPAL_12129, SRS_SPAL_12163, SRS_SPAL_12169, SRS_SPAL_12265, SRS_SPAL_12267

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R24-11

B.1.1 Added Specification Items in R24-11

none

B.1.2 Changed Specification Items in R24-11

none

B.1.3 Deleted Specification Items in R24-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R25-11

B.2.1 Added Specification Items in R25-11

none

B.2.2 Changed Specification Items in R25-11

Number	Heading
[SWS_Dio_00185]	Definition of datatype Dio_LevelType

Table B.1: Changed Specification Items in R25-11

B.2.3 Deleted Specification Items in R25-11

Number	Heading
[SWS_Dio_00015]	
[SWS_Dio_00018]	
[SWS_Dio_00021]	
[SWS_Dio_00023]	

Table B.2: Deleted Specification Items in R25-11

B.3 Traceable item history of this document according to AUTOSAR Release R26-11

B.3.1 Added Specification Items in R26-11

none

B.3.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_Dio_00141]	Definition of EcucParamConfContainerDef DioGeneral
[SWS_Dio_00131]	Definition of imported datatypes of module Dio

Table B.3: Changed Specification Items in R26-11

B.3.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_Dio_00143]	Definition of EcucBooleanParamDef DioVersionInfoApi
[SWS_Dio_00139]	Definition of API function Dio_GetVersionInfo
[SWS_Dio_00189]	

Table B.4: Deleted Specification Items in R26-11