Document Title	Specification of DIO Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	020
Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History			
Date	Release	Changed by	Change Description
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 precomplile 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.

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
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

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
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

Table of Contents

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.

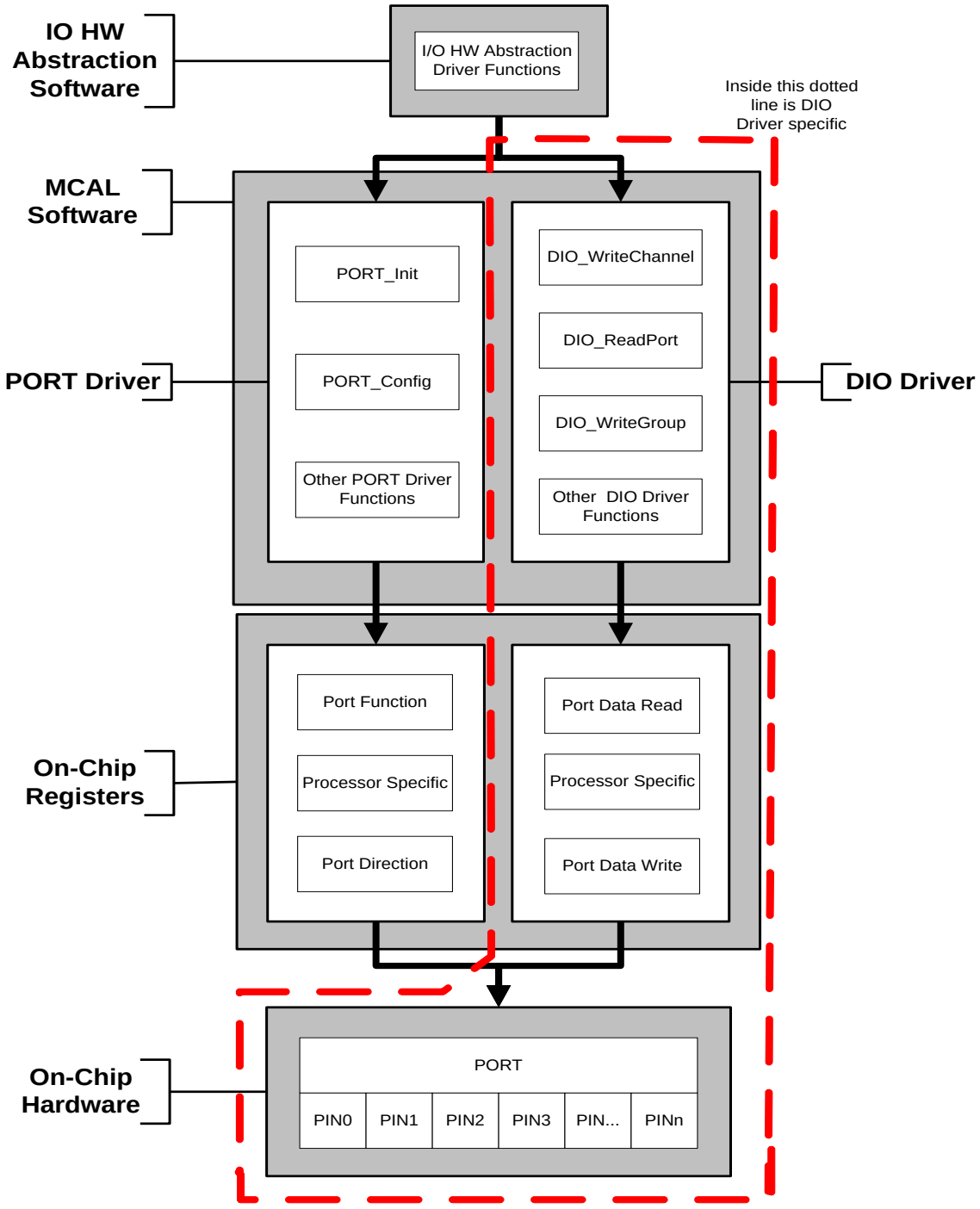

Figure 1: DIO Driver Structure and Integration

2 Acronyms and abbreviations

Acronyms and abbreviations that have a local scope are not contained in the AUTOSAR glossary. These must appear in a local 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.
LSB	Least Significant Bit
MSB	Most Significant Bit
DIO	Digital Input Output
ID	Identifier
ADC	Analog to Digital Converter
SPI	Serial Peripheral Interface
PWM	Pulse Width Modulation
ICU	Input Capture Unit
DET	Default Error Tracer
DEM	Diagnostic Event Manager

3 Related documentation

3.1 Deliverables of AUTOSAR

[1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[2] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[3] General Requirements on SPAL
AUTOSAR_SRS_SPALGeneral.pdf

[4] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[5] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[5] Specification of PORT Driver,
AUTOSAR_SWS_PortDriver.pdf

[6] Specification of Standard Types,
AUTOSAR_SWS_StandardTypes.pdf

[6] AUTOSAR Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[7] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [7] (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

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.⌋ ()

[SWS_Dio_00194] ⌈Dio.c shall **i**nclude Det.h if detection of development error (DET) is enabled.⌋ ()

6 Requirements traceability

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_00005	Modules of the µC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_Dio_00195
SRS_BSW_00006	The source code of software modules above the µC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_Dio_00195
SRS_BSW_00007	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	SWS_Dio_00195
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_Dio_00195
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_Dio_00195
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_Dio_00195
SRS_BSW_00160	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	SWS_Dio_00195
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_Dio_00195
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_Dio_00195
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_Dio_00195
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_Dio_00195
SRS_BSW_00168	SW components shall be tested by a	SWS_Dio_00195

	function defined in a common API in the Basis-SW	
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_Dio_00195
SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	SWS_Dio_00195
SRS_BSW_00304	All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types	SWS_Dio_00195
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_Dio_00195
SRS_BSW_00307	Global variables naming convention	SWS_Dio_00195
SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_Dio_00195
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_Dio_00195
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_Dio_00195
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_Dio_00195
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_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_Dio_00195
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_Dio_00195
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_Dio_00195
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_Dio_00195
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_Dio_00195
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the	SWS_Dio_00195

	meta data	
SRS_BSW_00335	Status values naming convention	SWS_Dio_00195
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_Dio_00195
SRS_BSW_00339	Reporting of production relevant error status	SWS_Dio_00195
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_Dio_00195
SRS_BSW_00342	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	SWS_Dio_00195
SRS_BSW_00343	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	SWS_Dio_00195
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_Dio_00001, SWS_Dio_00002
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_Dio_00195
SRS_BSW_00357	For success/failure of an API call a standard return type shall be defined	SWS_Dio_00195
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_Dio_00195
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_Dio_00195
SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	SWS_Dio_00195
SRS_BSW_00371	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	SWS_Dio_00195
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_Dio_00195
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_Dio_00195
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_Dio_00195
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_Dio_00195
SRS_BSW_00384	The Basic Software Module specifications shall specify at least in the description which other modules	SWS_Dio_00195

	they require	
SRS_BSW_00399	Parameter-sets shall be located in a separate segment and shall be loaded after the code	SWS_Dio_00195
SRS_BSW_00400	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	SWS_Dio_00195
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_Dio_00195
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_Dio_00195
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_Dio_00195
SRS_BSW_00411	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	SWS_Dio_00139
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_Dio_00195
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_Dio_00195
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_Dio_00195
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_Dio_00195
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_Dio_00195
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_Dio_00195
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_Dio_00195
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_Dio_00195
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_Dio_00195
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_Dio_00195
SRS_BSW_00429	Access to OS is restricted	SWS_Dio_00195

SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_Dio_00195
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_Dio_00195
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_00089, SWS_Dio_00200, SWS_Dio_00201, SWS_Dio_00202, SWS_Dio_00203
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
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_00089
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
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
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
SRS_SPAL_00157	All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers	SWS_Dio_00195
SRS_SPAL_12057	All driver modules shall implement an interface for initialization	SWS_Dio_00195
SRS_SPAL_12063	All driver modules shall only support raw value mode	SWS_Dio_00195
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_12067	All driver modules shall set their wake-up conditions depending on the selected operation mode	SWS_Dio_00195
SRS_SPAL_12068	The modules of the MCAL shall be	SWS_Dio_00195

	initialized in a defined sequence	
SRS_SPAL_12069	All drivers of the SPAL that wake up from a wake-up interrupt shall report the wake-up reason	SWS_Dio_00195
SRS_SPAL_12075	All drivers with random streaming capabilities shall use application buffers	SWS_Dio_00195
SRS_SPAL_12077	All drivers shall provide a non blocking implementation	SWS_Dio_00195
SRS_SPAL_12078	The drivers shall be coded in a way that is most efficient in terms of memory and runtime resources	SWS_Dio_00195
SRS_SPAL_12092	The driver's API shall be accessed by its handler or manager	SWS_Dio_00195
SRS_SPAL_12125	All driver modules shall only initialize the configured resources	SWS_Dio_00195
SRS_SPAL_12129	The ISRs shall be responsible for resetting the interrupt flags and calling the according notification function	SWS_Dio_00195
SRS_SPAL_12163	All driver modules shall implement an interface for de-initialization	SWS_Dio_00195
SRS_SPAL_12169	All driver modules that provide different operation modes shall provide a service for mode selection	SWS_Dio_00195
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_12265	Configuration data shall be kept constant	SWS_Dio_00195
SRS_SPAL_12267	Wakeup sources shall be initialized by MCAL drivers and/or the MCU driver	SWS_Dio_00195
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_00118, 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

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

The Dio SWS shall define synchronous read/write services.⌋ (SRS_Dio_12003, SRS_Dio_12004, SRS_Dio_12005, SRS_Dio_12006, SRS_Dio_12007, SRS_Dio_12008)

[SWS_Dio_00005] ⌈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).⌋ (SRS_Dio_12424)

[SWS_Dio_00089] ⌈Values used by the DIO Driver for the software level of Channels are either `STD_HIGH` or `STD_LOW`.⌋ (SRS_Dio_12003, SRS_Dio_12004, SRS_Dio_12005, SRS_Dio_12006, SRS_Dio_12007, SRS_Dio_12008)

[SWS_Dio_00128] ⌈A general-purpose digital IO pin represents a **DIO channel**.⌋ (SRS_Dio_12005, SRS_Dio_12008)

[SWS_Dio_00127] ⌈The Port module shall configure a DIO channel as input or output [SWS_Dio_00001 and SWS_Dio_00002].⌋ (SRS_Dio_12005, SRS_Dio_12008)

[SWS_Dio_00053] ⌈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] ⌈A channel group is a formal logical combination of several adjoining DIO channels within a DIO port.

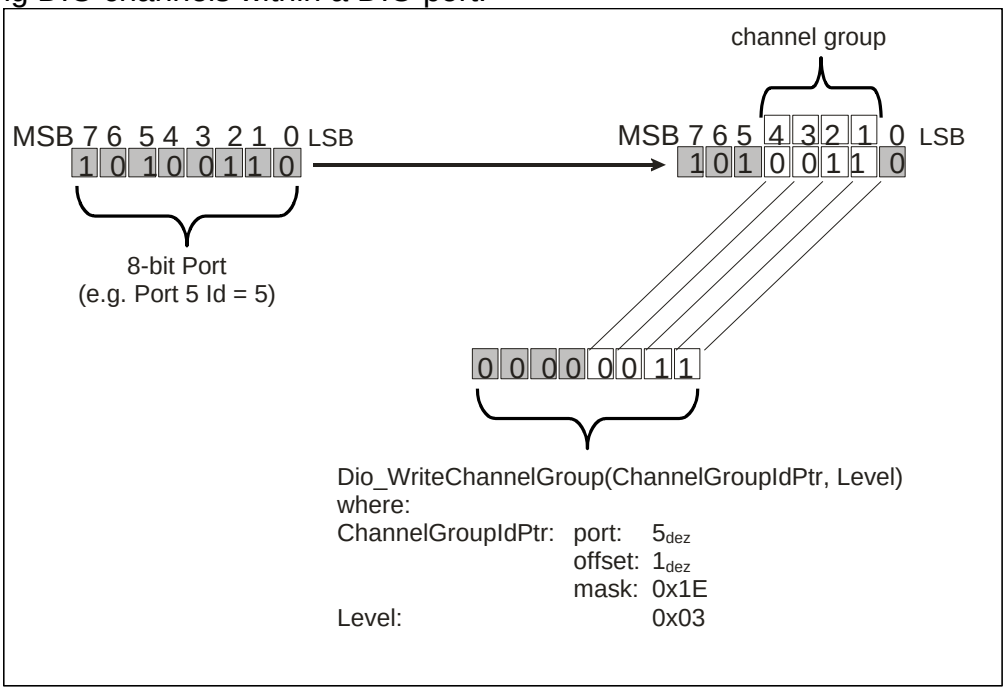

Figure 2: Schematic description of a ChannelGroup

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.

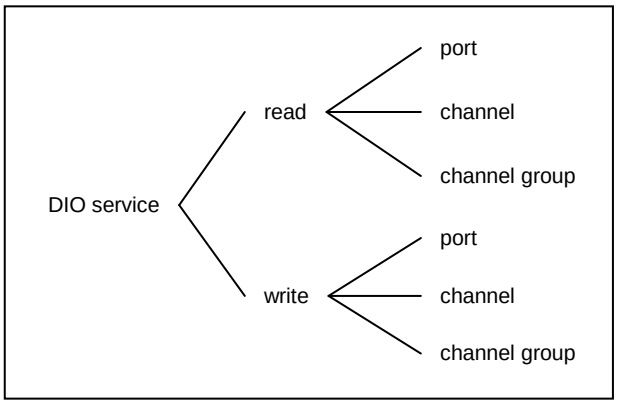

Figure 3: DIO Services

⌋(SRS_Dio_12004, SRS_Dio_12007)

[SWS_Dio_00060] ⌈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] ⌈The configuration process for Dio module shall provide symbolic names for each configured DIO channel, port and group.⌋(SRS_Dio_12355)

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] ⌈The Dio module shall not provide an interface for initialization of the hardware. The Port Driver performs this. ⌋(SRS_BSW_00344, SRS_SPAL_12064, SRS_SPAL_12461, SRS_SPAL_12462, SRS_SPAL_12463)

7.3 Runtime reconfiguration

7.3.1 Background & Rationale

Runtime reconfiguration is provided by the PORT Driver.

7.3.2 Requirements

[SWS_Dio_00002] ⌈The PORT driver shall provide the reconfiguration of the port pin direction during runtime. ⌋ (SRS_BSW_00344, SRS_SPAL_12064, SRS_SPAL_12461, SRS_SPAL_12462, SRS_SPAL_12463)

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] ⌈The Dio module’s write functions shall work on input and output channels. ⌋ (SRS_Dio_12352)

[SWS_Dio_00070] ⌈If a Dio write function is used on an input channel, it shall have no effect on the physical output level. ⌋ (SRS_Dio_12352)

[SWS_Dio_00109] ⌈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] ⌈If development errors are enabled and an error ocurred, the Dio module's write functions shall NOT process the write command. ⌋ (SRS_SPAL_12448)

7.4.2.1 DIO channel write service

[SWS_Dio_00006] ⌈The `Dio_WriteChannel` function shall set the level of a single DIO channel to `STD_HIGH` or `STD_LOW`. ⌋ (SRS_Dio_12005)

7.4.2.2 DIO port write service

[SWS_Dio_00007] ⌈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`. ⌋ (SRS_Dio_12003)

[SWS_Dio_00004] ⌈The `Dio_WritePort` function shall ensure that the functionality of the input channels of that port is not affected. ⌋ (SRS_Dio_12003)

7.4.2.3 DIO channel group write service

[SWS_Dio_00008] ⌈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`. ⌋ (SRS_Dio_12004)

7.4.2.4 DIO masked port write service

[SWS_Dio_00200] ⌈ 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`.⌋ (SRS_Dio_12003)

[SWS_Dio_00201] ⌈ The Dio_MaskedWritePort function shall ensure that the functionality of the input channels of that port is not affected. ⌋ (SRS_Dio_12003)

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] ⌈The Dio module's read functions shall work on input and output channels. ⌋ (SRS_Dio_12352)

[SWS_Dio_00118] ⌈If development errors are enabled and an error ocurred the Dio module's read functions shall return with the value '0'. ⌋ (SRS_SPAL_12448)

7.5.2.1 DIO channel read Service

[SWS_Dio_00011] ⌈The `Dio_ReadChannel` function shall read the level of a single DIO channel. ⌋ (SRS_Dio_12008)

7.5.2.2 DIO port read service

[SWS_Dio_00013] ⌈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`. ⌋ (SRS_Dio_12006)

7.5.2.3 DIO channel group read service

[SWS_Dio_00014] ⌈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`. ⌋ (SRS_Dio_12007)

7.5.2.4 DIO readback of output pins

[SWS_Dio_00083] ⌈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.⌋ (SRS_Dio_12352)

[SWS_Dio_00084] ⌈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.⌋ (SRS_Dio_12352)

7.6 Error classification

7.6.1 Development Errors

[SWS_Dio_00175] ⌈Invalid channel requested.
Related error code: DIO_E_PARAM_INVALID_CHANNEL_ID
Value: [hex]: 0x0A ⌋ ()

[SWS_Dio_00177] ⌈Invalid port requested.
Related error code: DIO_E_PARAM_INVALID_PORT_ID
Value: [hex]: 0x14 ⌋ ( )

[SWS_Dio_00178] ⌈Invalid channel group requested.
Related error code: DIO_E_PARAM_INVALID_GROUP
Value: [hex]: 0x1F ⌋ ()

[SWS_Dio_00188] ⌈API service called with a NULL pointer.
Related error code: DIO_E_PARAM_POINTER
Value: [hex]: 0x20 ⌋ ()

7.6.2 Runtime Errors

There are no runtime errors.

7.6.3 Transient Faults

There are no transient faults.

7.6.4 Production Errors

This module does not specify any production errors.

7.7 Error detection

7.7.1 API Parameter checking

[SWS_Dio_00074] ⌈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. ⌋(SRS_BSW_00323, SRS_SPAL_12448)

[SWS_Dio_00075] ⌈ 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. ⌋(SRS_BSW_00323, SRS_SPAL_12448)

[SWS_Dio_00114] ⌈ 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. ⌋(SRS_BSW_00323, SRS_SPAL_12448)

8 API specification

8.1 Imported types

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

[SWS_Dio_00131] ⌈

Module	*Header File*	*Imported Type*
Std_Types	StandardTypes.h	Std_ReturnType
	StandardTypes.h	Std_VersionInfoType

⌋ ()

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] ⌈

Name:	`Dio_ChannelType`		
Type:	`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_00015] ⌈Parameters of type `Dio_ChannelType` contain the numeric ID of a DIO channel. ⌋ ()

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

[SWS_Dio_00017] ⌈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`. ⌋ (SRS_SPAL_12263, SRS_Dio_12355)

8.2.2 Dio_PortType

[SWS_Dio_00183] ⌈

Name:	`Dio_PortType`		
Type:	`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_00018] ⌈Parameters of type `Dio_PortType` contain the numeric ID of a DIO port. ⌋ ()

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

[SWS_Dio_00020] ⌈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`. ⌋ (SRS_SPAL_12263, SRS_Dio_12355)

8.2.3 Dio_ChannelGroupType

[SWS_Dio_00184] ⌈

Name:	`Dio_ChannelGroupType`		
Type:	`Structure`		
Element:	`uint8/16/32`	`mask`	This element mask which defines the positions of the channel group.
	`uint8`	`offset`	This element shall be the position of the Channel Group on the port, counted from the LSB.
	`Dio_PortType`	`port`	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_00021] ⌈`Dio_ChannelGroupType` is the type for the definition of a channel group, which consists of several adjoining channels within a port. ⌋ ()

[SWS_Dio_00022] ⌈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`. ⌋ (SRS_SPAL_12263, SRS_Dio_12355)

8.2.4 Dio_LevelType

[SWS_Dio_00185] ⌈

Name:	`Dio_LevelType`		
Type:	`uint8`		
Range:	`STD_LOW`	`0x00`	Physical state 0V
	`STD_HIGH`	`0x01`	Physical state 5V or 3.3V
Description:	These are the possible levels a DIO channel can have (input or output)		
Available via:	`Dio.h`		

⌋ ()

[SWS_Dio_00023] ⌈`Dio_LevelType` is the type for the possible levels that a DIO channel can have (input or output). ⌋ ()

8.2.5 Dio_PortLevelType

[SWS_Dio_00186] ⌈

Name:	`Dio_PortLevelType`		
Type:	`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 is a list of functions provided for upper layer modules.

8.3.1 Dio_ReadChannel

[SWS_Dio_00133] ⌈

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] ⌈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, SWS_Dio_00118 and SWS_Dio_00026 are applicable to the `Dio_ReadChannel` function. ⌋ (SRS_Dio_12008)

8.3.2 Dio_WriteChannel

[SWS_Dio_00134] ⌈

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] ⌈If the specified channel is configured as an output channel, the `Dio_WriteChannel` function shall set the specified Level for the specified channel. ⌋ (SRS_Dio_12005)

[SWS_Dio_00029] ⌈If the specified channel is configured as an input channel, the `Dio_WriteChannel` function shall have no influence on the physical output. ⌋ (SRS_Dio_12005)

[SWS_Dio_00079] ⌈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. ⌋ (SRS_Dio_12005)

8.3.3 Dio_ReadPort

[SWS_Dio_00135] ⌈

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] ⌈The `Dio_ReadPort` function shall return the level of all channels of that port.⌋ (SRS_Dio_12006)

[SWS_Dio_00104] ⌈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, SWS_Dio_00118 and SWS_Dio_00026 are applicable to the `Dio_ReadPort` function.⌋ ()

8.3.4 Dio_WritePort

[SWS_Dio_00136] ⌈

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] ⌈The `Dio_WritePort` function shall set the specified value for the specified port.⌋ (SRS_Dio_12003)

[SWS_Dio_00035] ⌈When the `Dio_WritePort` function is called, DIO Channels that are configured as input shall remain unchanged.⌋ (SRS_Dio_12003)

[SWS_Dio_00105] ⌈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] ⌈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] ⌈

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] ⌈The `Dio_ReadChannelGroup` function shall read a subset of the adjoining bits of a port (channel group). ⌋ (SRS_Dio_12007)

[SWS_Dio_00092] ⌈The `Dio_ReadChannelGroup` function shall do the masking of the channel group. ⌋ (SRS_Dio_12007)

[SWS_Dio_00093] ⌈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, SWS_Dio_00118 and SWS_Dio_00026 are applicable to the `Dio_ReadChannelGroup` function. ⌋ (SRS_Dio_12007)

8.3.6 Dio_WriteChannelGroup

[SWS_Dio_00138] ⌈

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] ⌈The `Dio_WriteChannelGroup` function shall set a subset of the adjoining bits of a port (channel group) to a specified level. ⌋ (SRS_Dio_12004)

[SWS_Dio_00040] ⌈The `Dio_WriteChannelGroup` shall not change the remaining channels of the port and channels which are configured as input. ⌋ (SRS_Dio_12004)

[SWS_Dio_00090] ⌈The `Dio_WriteChannelGroup` function shall do the masking of the channel group.⌋ (SRS_Dio_12004)

[SWS_Dio_00091] ⌈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.⌋ (SRS_Dio_12004)

8.3.7 Dio_GetVersionInfo

[SWS_Dio_00139] ⌈

Service name:	Dio_GetVersionInfo	

Syntax:	`void Dio_GetVersionInfo(`
	`  Std_VersionInfoType* VersionInfo`
	`)`	

Service ID[hex]:	0x12	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	VersionInfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Service to get the version information of this module.	
Available via:	`Dio.h`	

⌋ (SRS_BSW_00411)

[SWS_Dio_00189] ⌈If DET is enabled for the DIO Driver module, the function Dio_GetVersionInfo shall raise DIO_E_PARAM_POINTER, if the argument is NULL pointer and return without any action.

See also Chapter 10. ⌋ ()

8.3.8 Dio_FlipChannel

[SWS_Dio_00190] ⌈

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] ⌈If the specified channel is configured as an output channel, the `Dio_FlipChannel` function shall read level of the channel (requirements [SWS_Dio_00083] & [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] ⌈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] ⌈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.

See also Chapter 10.⌋ ()

8.3.9 Dio_MaskedWritePort

[SWS_Dio_00300] ⌈

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] ⌈ The Dio_MaskedWritePort function shall set the specified value for the channels in the specified port if the corresponding bit in Mask is '1'. ⌋ (SRS_Dio_12003)

[SWS_Dio_00203] ⌈When the Dio_MaskedWritePort function is called, DIO Channels that are configured as input shall remain unchanged. ⌋ (SRS_Dio_12003)

[SWS_Dio_00204] [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 Call-back notifications

This chapter lists all functions provided by the Dio module to lower layers.

The Dio module does not provide any callback notifications. Callbacks related to the functionality of the Dio module are implemented in another module (ICU Driver and/or complex drivers).

8.5 Scheduled functions

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

The Dio module has no scheduled functions.

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

None

8.6.2 Optional Interfaces

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

[SWS_Dio_00140] ⌈

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 and development mode with error condition. For development mode with no error the diagrams for normal operation mode are valid. 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 - 1

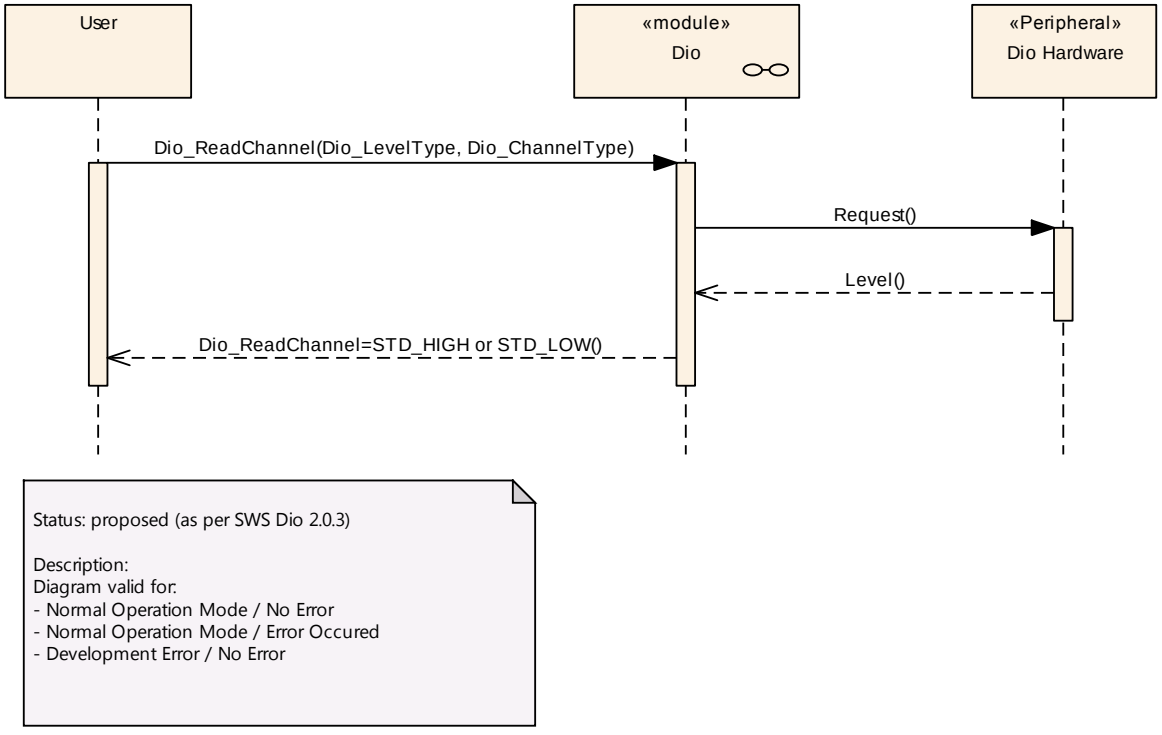

Figure 4: Read Service Sequence Chart (normal operation mode)

9.2 Read a value from a digital I/O - 2

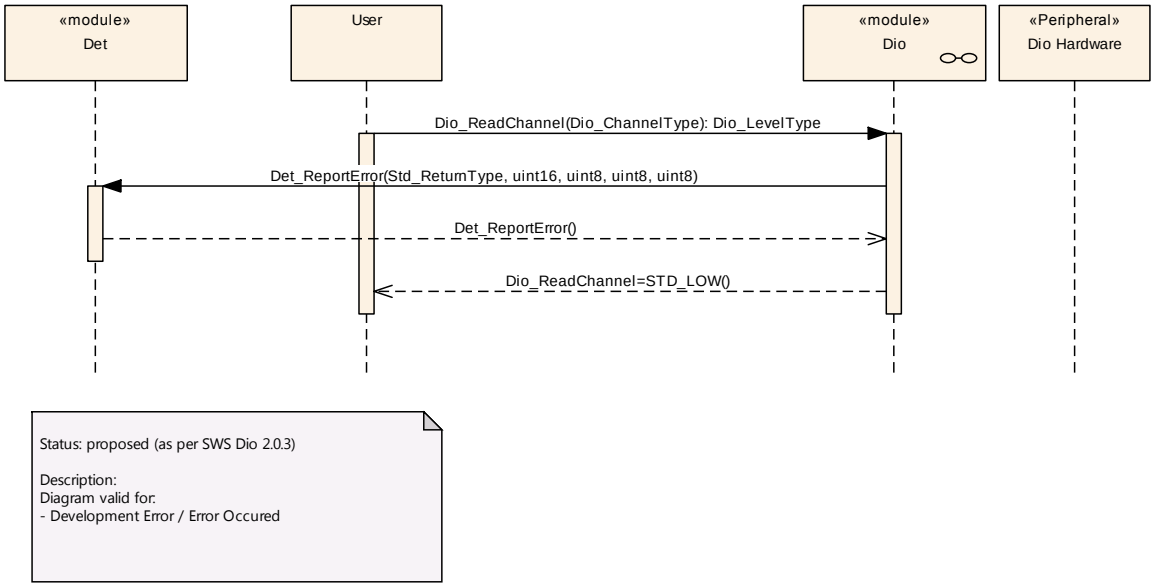

Figure 5: Read Service Sequence Chart (development error mode)

9.3 Write a value to a digital I/O - 1

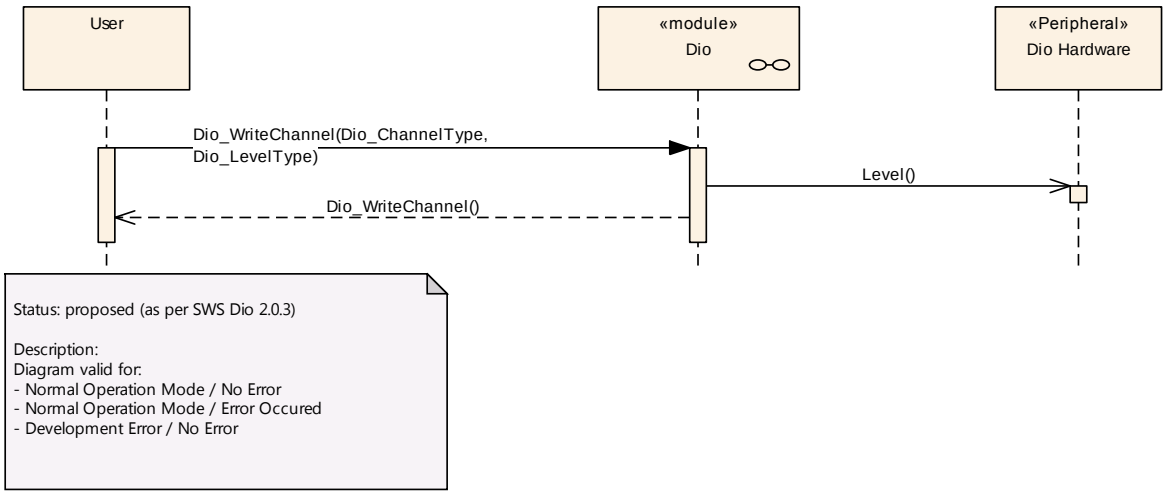

Figure 6: Write Service Sequence Chart (normal operation mode)

9.4 Write a value to a digital I/O - 2

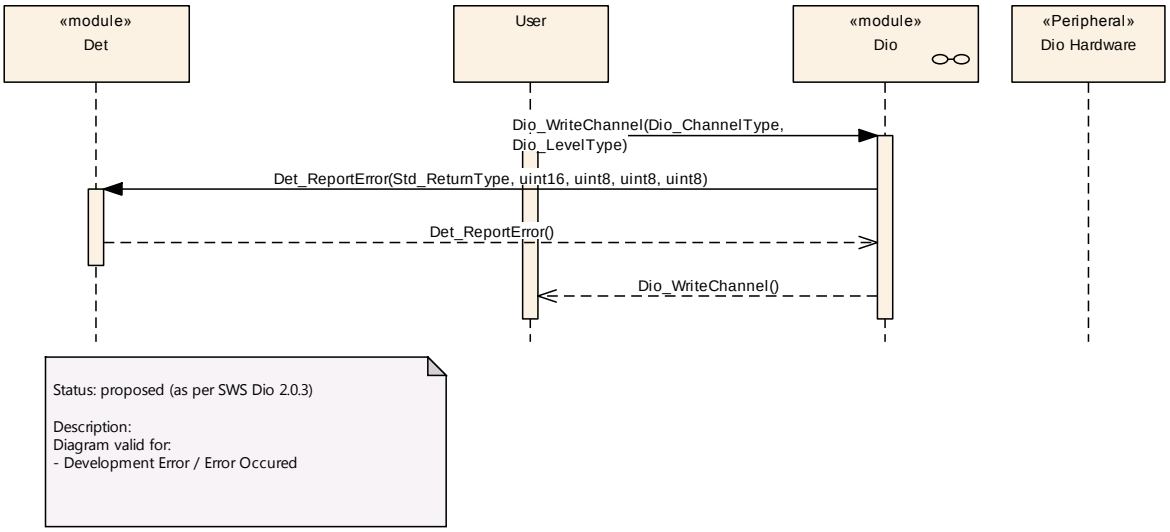

Figure 7: Write Service Sequence Chart (development error mode)

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. The detailed meanings of the parameters describe Chapters 7 and Chapter 8.

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

10.1.1 Variants

[SWS_Dio_00129] ⌈At least one of the following variants has to be supported by implementation:
- VARIANT-PRE-COMPILE
- VARIANT-LINK-TIME⌋ ()

10.1.2 Dio

SWS Item	**ECUC_Dio_00154 :**
Module Name	*Dio*
Module Description	Configuration of the Dio (Digital IO) module.
Post-Build Variant Support	false
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-PRE-COMPILE

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
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

SWS Item	**ECUC_Dio_00141 :**
Container Name	DioGeneral
Description	General DIO module configuration parameters.
Configuration Parameters	

SWS Item	**ECUC_Dio_00142 :**		
Name	DioDevErrorDetect		
Parent Container	DioGeneral		
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		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dio_00153 :**		
Name	DioFlipChannelApi		
Parent Container	DioGeneral		
Description	Adds / removes the service Dio_FlipChannel() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dio_00155 :**		
Name	DioMaskedWritePortApi		
Parent Container	DioGeneral		
Description	Adds / removes the service Dio_MaskedWritePort() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dio_00143 :**		
Name	DioVersionInfoApi		
Parent Container	DioGeneral		
Description	Adds / removes the service Dio_ GetVersionInfo() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dio_00156 :**		
Name	DioEcucPartitionRef		
Parent Container	DioGeneral		
Description	Maps the DIO driver to zero or multiple ECUC partitions to make the modules API available in this partition. **Tags:** atp.Status=draft		
Multiplicity	0..*		
Type	Reference to [EcucPartition]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.1.4 DioPort

SWS Item	**ECUC_Dio_00144 :**
Container Name	DioPort
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.
Configuration Parameters	

SWS Item	**ECUC_Dio_00145 :**		
Name	DioPortId		
Parent Container	DioPort		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dio_00157 :**

Name	DioPortEcucPartitionRef		
Parent Container	DioPort		
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. **Tags:** atp.Status=draft		
Multiplicity	0..*		
Type	Reference to [EcucPartition]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
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.

[SWS_Dio_00206] DRAFT [The ECUC partitions referenced by DioPortEcucPartitionRef shall be a subset of the ECUC partitions referenced by DioEcucPartitionRef.]()

10.1.5 DioChannel

SWS Item	**ECUC_Dio_00146 :**
Container Name	DioChannel
Description	Configuration of an individual DIO channel.
Configuration Parameters	

SWS Item	**ECUC_Dio_00147 :**		
Name	DioChannelId		
Parent Container	DioChannel		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dio_00158 :**		
Name	DioChannelEcucPartitionRef		
Parent Container	DioChannel		
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. **Tags:** atp.Status=draft		
Multiplicity	0..*		
Type	Reference to [EcucPartition]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

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

10.1.6 DioChannelGroup

SWS Item	**ECUC_Dio_00148 :**
Container Name	DioChannelGroup
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.
Configuration Parameters	

SWS Item	**ECUC_Dio_00149 :**
Name	DioChannelGroupIdentification
Parent Container	DioChannelGroup
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	--

maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_Dio_00150 :**		
Name	DioPortMask		
Parent Container	DioChannelGroup		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dio_00151 :**		
Name	DioPortOffset		
Parent Container	DioChannelGroup		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Dio_00159 :**
Name	DioChannelGroupEcucPartitionRef
Parent Container	DioChannelGroup
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. **Tags:** atp.Status=draft
Multiplicity	0..*
Type	Reference to [EcucPartition]
Post-Build Variant Multiplicity	false

Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

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

10.1.7 DioConfig

SWS Item	**ECUC_Dio_00152 :**
Container Name	DioConfig
Description	This container contains the configuration parameters and sub containers of the AUTOSAR DIO module.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
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 the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*

11 Not applicable requirements

[SWS_Dio_00195] ⌈These requirements are not applicable to this specification.⌋ (SRS_BSW_00101, SRS_BSW_00005, SRS_BSW_00006, SRS_BSW_00007, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00160, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00164, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00172, SRS_BSW_00304, SRS_BSW_00306, SRS_BSW_00307, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00314, SRS_BSW_00321, SRS_BSW_00325, SRS_BSW_00328, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00333, SRS_BSW_00334, SRS_BSW_00335, SRS_BSW_00336, SRS_BSW_00339, SRS_BSW_00341, SRS_BSW_00342, SRS_BSW_00343, SRS_BSW_00347, SRS_BSW_00357, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00369, SRS_BSW_00371, SRS_BSW_00373, SRS_BSW_00375, SRS_BSW_00377, SRS_BSW_00378, SRS_BSW_00384, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00406, SRS_BSW_00413, SRS_BSW_00416, SRS_BSW_00417, 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_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)