Document Title	# Specification of DIO Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	020
Document Classification	Standard

Document Version	2.4.1
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change Description
28.02.2014	2.4.1	AUTOSAR Release Management	• Editorial changes • Removed chapter(s) on change documentation
17.05.2012	2.4.0	AUTOSAR Administration	• Renamed sequence diagram names with prefix 'Dio'
27.04.2011	2.3.0	AUTOSAR Administration	• Added new requirements DIO152, DIO195, DIO196, DIO197, DIO198 and DIO199 • Update Chapter 8 and 10 • Legal disclaimer revised
23.06.2008	2.2.1	AUTOSAR Administration	Legal disclaimer revised
19.12.2007	2.2.0	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

Document Change History

Date	Version	Changed by	Change Description
24.01.2007	2.1.0	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 BSW00435 and BSW00436 • Legal disclaimer revised • “Advice for users” revised • “Revision Information” added
26.01.2006	2.0.0	AUTOSAR Administration	• Major changes in chapter 10, Configuration specification • Structure of document changed partly • Readback support moved to PORT Driver • Other changes see chapter 11
30.06.2005	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

.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.

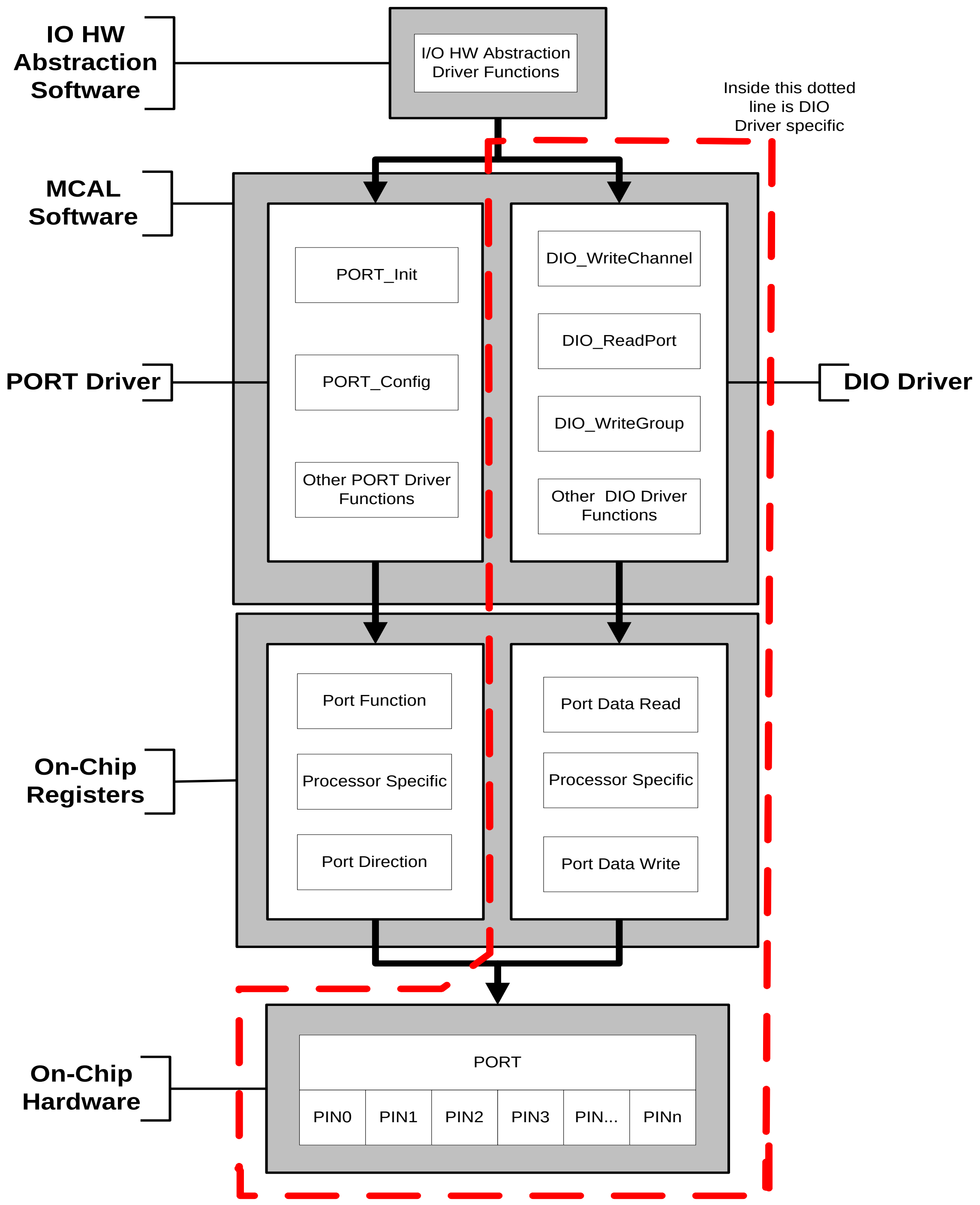

Figure 1: DIO Driver Structure and Integration

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	Development Error Tracer
DEM	Diagmostic Event Manager

3 Related documentation

3.1 Deliverables of AUTOSAR

[1] Layered Software Architecture
AUTOSAR_LayeredSoftwareArchitecture.pdf

[2] List of Basic Software Modules
AUTOSAR_BasicSoftwareModules.pdf

[3] General Requirements on SPAL
AUTOSAR_SRS_SPAL_General.pdf

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

[5] Specification of ECU Configuration
AUTOSAR_ECU_Configuration.pdf

[6] Specification of PORT Driver,
AUTOSAR_SWS_PORT_Driver.pdf

[7] Specification of Standard Types,
AUTOSAR_SWS_StandardTypes.pdf

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

3.2 Related standards and norms

[8] Specification I/O Drivers,
http://www.automotive-his.de/download/
API_IODriver_2_1_3.pdf

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

DIO061: 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.

DIO063: The Dio module shall adapt its configuration and usage to the microcontroller and ECU.

DIO102: 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.

5.1 File structure

DIO117: The Dio module shall comply with the following file structure

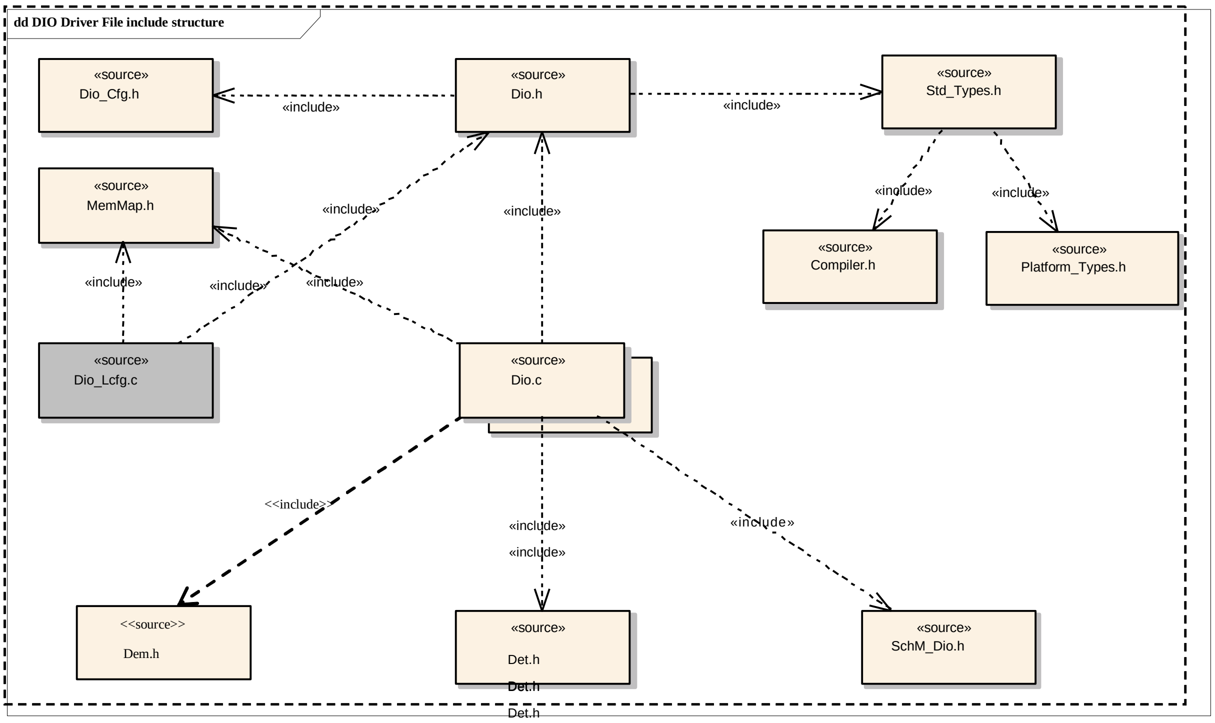

Figure 2: Include File Structure

`Dio.h` shall include `Dio _Cfg.h` for the API pre-compiler switches

`Dio.c` has access to the `Dio_Cfg.h` via the implicitly include through the `Dio.h` file.

The Type definitions for `Dio_Lcfg.c` are located in the file `Dio_Cfg.h`. or `Dio.h`.

The implicit include of `Dio_Cfg.h` via `Dio.h` in the files `Dio_Lcfg.c` is necessary to solve the following construct:

```
Dio.h
----------
#ifdef DioVersionInfoApi
Dio_GetVersionInfo(...)
#endif

Dio_Cfg.h
---------------
#include "Dio.h"
#define DioVersionInfoApi
```

`Dio.c` shall include `Dio_Cbk.h` for a pre-compile time configuration.

The module shall optionally include the Dem.h file if any production error will be issued by the implementation. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols, which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

6 Requirements traceability

This chapter refers to 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.

Document: AUTOSAR General Requirements on Basic Software Modules [4]

Requirement	***Satisfied by***
[BSW003] Version identification	DIO082
[BSW004] Version check	DIO082, DIO106
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (DIO does not use any other driver)
[BSW006] Platform independency	Not applicable (it is a non functional requirement)
[BSW007] HIS MISRA C	Not applicable (it is a non functional requirement)
[BSW009] Module User Documentation	Not applicable (it is a non functional requirement)
[BSW010] Memory resource documentation	Not applicable (it is a non functional requirement)
[BSW101] Initialization interface	DIO001, DIO002
[BSW158] Separation of configuration from implementation	See Figure 2; **DIO117**
[BSW159] Tool-based configuration	See Figure 2
[BSW160] Human-readable configuration data	Not applicable (it only applies to the configuration)
[BSW161] Microcontroller abstraction	Not applicable (architectural AUTOSAR concept is the basis for this driver)
[BSW162] ECU layout abstraction	Not applicable (architectural AUTOSAR concept is the basis for this driver)
[BSW164] Implementation of interrupt service routines	Not applicable (DIO does not provide interrupt functionality)
[BSW167] Static configuration checking	DIO071
[BSW168] Diagnostic Interface of SW components	Not applicable (DIO does not provide diagnostic capabilities)
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable (it only affects the configuration)
[BSW171] Configurability of optional functionality	DIO124, DIO125
[BSW172] Compatibility and documentation of scheduling strategy	Not applicable (DIO does not have any special scheduling requirements)
[BSW00300] Module naming convention	See Figure 2
[BSW00301] Limit imported information	DIO117 and Figure 2

[BSW00302] Limit exported information	DIO117 and Figure 2
[BSW00304] AUTOSAR integer data types	Not applicable (it is a non functional requirement)
[BSW00305] Self-defined data types naming convention	See Section 8.2
[BSW00306] Avoid direct use of compiler and platform specific keywords	Not applicable (requirement on implementation)
[BSW00307] Global variables naming convention	Not applicable (requirement on implementation)
[BSW00308] Definition of global data	Not applicable (requirement on implementation)
[BSW00309] Global data with read-only constraint	Not applicable (it is a non functional requirement)
[BSW00310] API naming convention	See section 8
[BSW00312] Shared code shall be reentrant	See section 8
[BSW00314] Separation of interrupt frames and service routines	Not applicable (DIO does not provide interrupt capabilities)
[BSW00318] Format of module version numbers	DIO082
[BSW00321] Enumeration of module version numbers	DIO082
[BSW00323] API parameter checking	DIO065, DIO074, DIO075, DIO114
[BSW00325] Runtime of interrupt service routines	Not applicable (DIO does not provide interrupt capabilities
[BSW00326] Transition from ISRs to OS tasks	Not applicable because DIO does not provide interrupt capabilities
[BSW00327] Error values naming convention	DIO067; DIO065, see chapter 7.6
[BSW00328] Avoid duplication of code	Not applicable (requirement for the implementer)
[BSW00329] Avoidance of generic interfaces	Not applicable (no generic interfaces specified within this SWS)
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable (requirement for the implementer)
[BSW00331] Separation of error and status values	Not applicable (no status values specified within this SWS)
[BSW00333] Documentation of callback function context	Not applicable (it is a non functional requirement)
[BSW00334] Provision of XML file	Not applicable (it is a non functional requirement)
[BSW00335] Status values naming convention	Not applicable (no status values specified within this SWS)
[BSW00336] Shutdown interface	Not applicable (for DIO there is no need for this)
[BSW00337] Classification of errors	DIO065
[BSW00338] Detection and Reporting of development errors	DIO066, DIO073, DIO067
[BSW00339] Reporting of production relevant errors and exceptions	Not applicable (DIO only provides development errors)
[BSW00341] Microcontroller compatibility documentation	Not applicable (requirement for the implementer)

[BSW00342] Usage of source code and object code	Not applicable (requirement on implementation)
[BSW00343] Specification and configuration of time	Not applicable (DIO doesen't deal with time)
[BSW00344] Link-time configuration	DIO001, DIO002
[BSW00345] Pre-compile-time configuration	DIO071
[BSW00346] Basic set of module files	DIO117 and Figure 2
[BSW00347] Naming separation of drivers	Not applicable (requirement on implementation)
[BSW00348] Standard type header	See Figure 2
[BSW00350] Development error detection keyword	DIO066
[BSW00353] Platform specific type header	See Figure 2
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable (no integer data types redefined in this specification)
[BSW00357] Standard API return type	Not applicable (it is a non functional requirement)
[BSW00358] Return type of init() functions	Not applicable (there is no init() function for DIO)
[BSW00359] Return type of callback functions	Not applicable (DIO does not provide a callback mechanism)
[BSW00360] Parameters of callback functions	Not applicable (DIO does not provide a callback mechanism)
[BSW00361] Compiler specific language extension header	See Figure 2
[BSW00369] Do not return development error codes via API	Not applicable (it is a non functional requirement)
[BSW00370] Separation of callback interface from API	Not applicable (DIO does not provide a callback mechanism)
[BSW00371] Do not pass function pointers via API	Not applicable (no function pointers are passed via API to this module)
[BSW00373] Main processing function naming convention	Not applicable (no main processing function specified)
[BSW00374] Module vendor identification	DIO115
[BSW00375] Notification of wake-up reason	Not applicable (DIO does not provide a wake-up mechanism)
[BSW00376] Return type and parameters of main processing functions	Not applicable (no main processing function specified)
[BSW00377] Module specific API return types	Not applicable (it is a non functional requirement)
[BSW00378] AUTOSAR boolean type	Not applicable (it is a non functional requirement)
[BSW00379] Module identification	DIO082
[BSW00380] Separate C-File for configuration parameters	DIO117
[BSW00381] Separate configuration header file for pre-compile time parameters	DIO117

[BSW00382] Not-used configuration elements need to be listed	Not applicable (it is a non functional requirement)
[BSW00383] List dependencies of configuration files	See chapter 10.2
[BSW00384] List dependencies to other modules	Not applicable (it is a non functional requirement)
[BSW00385] List possible error notificatons	See Chapter 7.6
[BSW00386] Configuration for detecting an error	See Chapter 7.7
[BSW00387] Specify the configuration class of callback function	Not applicable (DIO doesen't provide a callback meachanism)
[BSW00388] Introduce containers	See chapter 10.1
[BSW00389] Containers shall have names	See chapter 10.1
[BSW00390] Parameter content shall be unique within the module	See chapter 10.1
[BSW00391] Parameter shall have unique names	See chapter 10.1
[BSW00392] Parameters shall have a type	See chapter 10.1
[BSW00393] Parameters shall have a range	See chapter 10.1
[BSW00394] Specify the scope of the parameters	See chapter 10.1
[BSW00395] List the required parameters (per parameter)	See chapter 10.1
[BSW00396] Configuration classes	See chapter 10.1
[BSW00397] Pre-compile-time parameters	See chapter 10.1
[BSW00398] Link-time parameters	See chapter 10.1
[BSW00399] Loadable Post-build time parameters	Not applicable (no post-build time configurable parameters specified)
[BSW00400] Selectable Post-build time parameters	(no post-build time configurable parameters specified)
[BSW00401] Documentation of multiple instances of configuration parameters	See chapter 10.1
[BSW00402] Published information	See chapter 10.4
[BSW00404] Reference to post build time configuration	Not applicable (no post-build time configurable parameters specified)
[BSW00405] Reference to multiple configuration sets	Not applicable (no post-build time configurable parameters specified)
[BSW00406] Check module initialization	Not applicable (Dio has no init function)
[BSW00407] Function to read out published parameters	DIO123
[BSW00408] Configuration parameter naming convention	See chapter 10.1
[BSW00409] Header files for production code error IDs	DIO117
[BSW00410] Compiler switches shall have defined values	DIO124
[BSW00411] Get version info keyword	DIO139
[BSW00412] Separate H-File for configuration parameters	DIO117
[BSW00413] Accessing instances of BSW modules	Not applicable (requirement on implementation)
[BSW00414] Parameter of init function	Not applicable (Dio has no init function)
[BSW00415] User dependent include files	See Figure2
[BSW00416] Sequence of Initialization	Nor applicable (DIO is not responsible for overall Autosar modules initialization)
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable (applies only for non BSW modules)
[BSW00419] Separate C-Files for pre-compile time configuration parameters	DIO117 and Figure2
[BSW00420] Production relevant error event rate detection	Not applicable (applies only for DEM)
[BSW00421] Reporting of production relevant error events	DIO066

[BSW00422] Debouncing of production relevant error status	Not applicable (applies only for DEM)
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (GPT driver has no Autosar Interface)
[BSW00424] BSW main processing function task allocation	Not applicable (no main processing function specified)
[BSW00425] Trigger conditions for schedulable objects	Not applicable (requirement for the implementer)
[BSW00426] Exclusive areas in BSW modules	Not applicable (applies only for the module description template)
[BSW00427] ISR description for BSW modules	Not applicable (applies only for the module description template)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (no main processing function specified)
[BSW00429] Restricted BSW OS functionality access	Not applicable (requirement for the implementer)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (no scheduling functionality in the DIO module
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (no main processing function specified)
[BSW00433] The Schedule Module shall provide an API for exclusive areas	Not applicable (no main processing function specified)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (no main processing function specified)
[BSW00435] Module Header File Structure for the Basic Software Scheduler	DIO117
[BSW00436] Module Header File Structure for the Memory Mapping	DIO117

Document: AUTOSAR Requirements on Basic Software, Module SPAL, general [3]

Requirement	***Satisfied by***
[BSW157] Notification mechanisms of drivers and handlers	Not applicable (DIO does not provide any notification mechanism)
[BSW12056] Configuration of notification mechanism	Dio has no callbacks
[BSW12057] Driver module initialization	DIO001, DIO002
[BSW12063] Raw value mode	Not applicable (DIO only provides digital values)
[BSW12064] Change of operation mode during running operation	DIO001, DIO002
[BSW12067] Setting of wake-up conditions	Not applicable (DIO does not provide any wake-up capability)
[BSW12068] MCAL initialization sequence	Not applicable (DIO does not need any initialization sequence)

[BSW12069] Wake-up notification of ECU State Manager	Not applicable (DIO does not provide any wake-up capability)
[BSW12075] Use of application buffers	Not applicable (DIO is no memory driver)
[BSW12077] Non-blocking implementation	Not applicable (it is a non functional requirement)
[BSW12078] Runtime and memory efficiency	Not applicable (it is a non functional requirement)
[BSW12092] Access to drivers [approved]	Not applicable (it is a non functional requirement)
[BSW12125] Initialization of hardware resources	DIO001, DIO002
[BSW12129] Resetting of interrupt flags	Not applicable (DIO does not provide any interrupt functionality)
[BSW12163] Driver module deinitialization	DIO001, DIO002
[BSW12169] Control of operation mode	Not applicable (DIO does not provide different operation modes)
[BSW12263] Object code compatible configuration concept	DIO017, DIO020, DIO022
[BSW12264] Specification of configuration items	Not applicable (it is a non functional requirement)
[BSW12265] Configuration data shall be kept constant	Not applicable (it is a non functional requirement)
[BSW12267] Configuration of wakeup sources	Not applicable (DIO does not provide any wake-up capability)
[BSW12448] Behaviour after development error detection	DIO074, DIO075,DIO080, DIO081 DIO114, DIO118, DIO119
[BSW12461] Responsibility for register initialization	DIO001, DIO002
[BSW12462] General initialization of overall registers	DIO001, DIO002
[BSW12463] Combine and forward settings for register initialization	DIO001, DIO002

Document: AUTOSAR Requirements on Basic Software, Module SPAL, DIO Driver [3]

Requirement	***Satisfied by***
[BSW12003] DIO port write service	DIO050, DIO051, DIO055, DIO089, DIO057, DIO004, DIO007, DIO034, DIO035
[BSW12004] DIO channel group write service	DIO050, DIO051, DIO055, DIO089, DIO056, DIO057, DIO008, DIO039, DIO040, DIO090, DIO091
[BSW12005] DIO channel write service	DIO050, DIO051, DIO055, DIO089, DIO127, DIO128, DIO057, DIO006, DIO028, DIO029, DIO079
[BSW12006] DIO port read service	DIO050, DIO051, DIO055, DIO089, DIO057, DIO013, DIO031
[BSW12007] DIO channel group read service	DIO050, DIO051, DIO055, DIO089, DIO056, DIO057, , DIO058, DIO014, DIO037, DIO092, DIO093
[BSW12008] DIO channel read service	DIO050, DIO051, DIO055, DIO089,

	DIO127, DIO128, DIO011, DIO027, DIO082
[BSW12352] General read/write behavior	DIO064, DIO070, DIO012, DIO083, DIO084
[BSW12355] Configuration of symbolic names	DIO026, DIO017, DIO020, DIO022, DIO113
[BSW12424] Provide atomicity of DIO access	DIO005

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

DIO050: The Dio SWS shall define functions allowing

- Port-
- Channel-
- Channel-group -

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

DIO051: The Dio module shall not buffer data when providing read and write services.

DIO055: The Dio SWS shall define synchronous read/write services.

DIO005: 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).

DIO089: Values used by the DIO Driver for the software level of Channels are either `STD_HIGH` or `STD_LOW`.

DIO128: A general-purpose digital IO pin represents a **DIO channel**.

DIO127: The Port module shall configure a DIO channel as input or output [DIO001 and DIO002].

DIO053: 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**. The single DIO channel levels inside a DIO port represent a bit in the DIO port value, depending on their position inside the port.

DIO056: A channel group is a formal logical combination of several adjoining DIO channels within a DIO port.

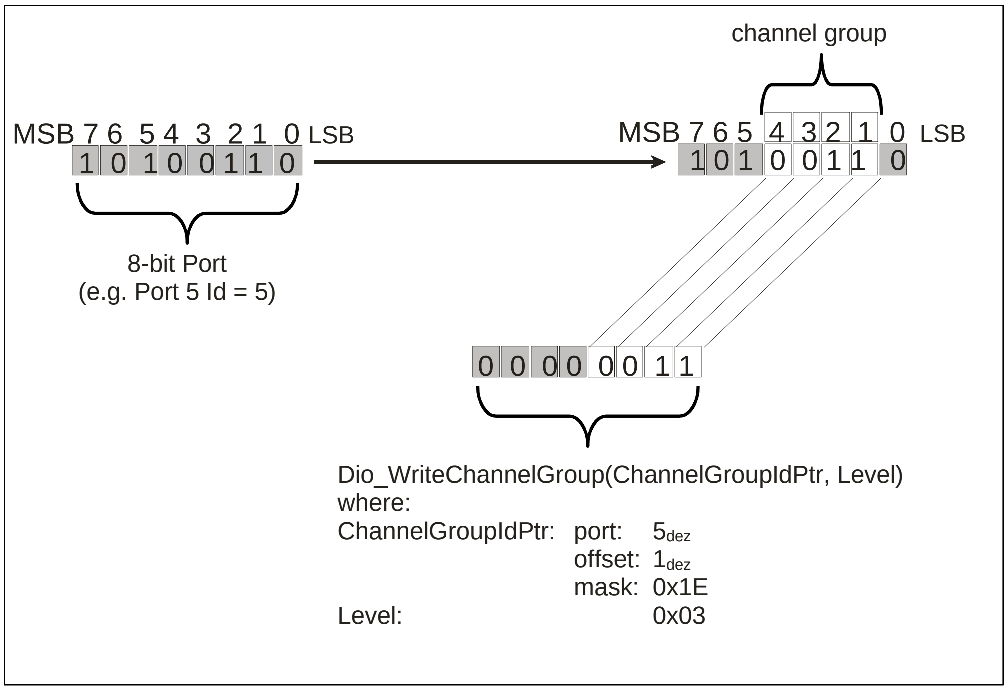

Figure 3: Schematic description of a ChannelGroup

The DIO Driver provides the following services:

- **DIO057:** The Dio SWS shall define functions to modify the levels of output channels individually, for a port or for a channel group.
- **DIO058:** The Dio SWS shall define functions to read the level of input and output (see DIO083) channels individually, for a port or for a channel group.

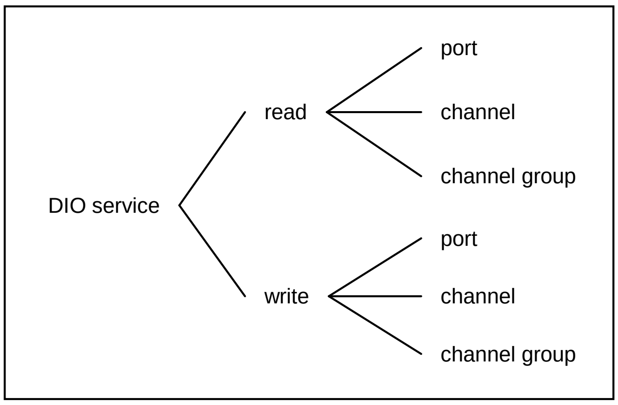

Figure 4: DIO Services

DIO060: The Dio SWS shall define that all read and write functions of the Dio module are 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.

DIO026: The configuration process for Dio module shall provide symbolic names for each configured DIO channel, port and group.

DIO113: The Dio module shall publish the symbolic names which have been created during the configuration process in the file “Dio_Cfg.h”.

7.1.3 Version check

7.1.3.1 Background & Rationale

The integration of incompatible files must be avoided. Minimum implementation is the version check of the header file inside the C file (version numbers of C and H file shall be identical)

7.1.3.2 Requirements

DIO106: The Dio module shall implement the following version checks of the header file:
For included header files:

- <MODULENAME>_AR_MAJOR_VERSION
- <MODULENAME>_AR_MINOR_VERSION

shall be identical.
For the module internal c and h files:

- <MODULENAME>_SW_MAJOR_VERSION
- <MODULENAME>_SW_MINOR_VERSION
- <MODULENAME>_AR_MAJOR_VERSION
- <MODULENAME>_AR_MINOR_VERSION
- <MODULENAME>_AR_PATCH_VERSION

shall be identical. [DIO082].

7.2 Initialization

7.2.1 Background & Rationale

Initialization is done by the PORT Driver.

7.2.2 Requirements

DIO001: The Dio module shall not provide an interface for initialization. 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

DIO002: 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

DIO064: The Dio module’s write functions shall work on input and output channels.

DIO070: If a Dio write function is used on an input channel, it shall have no effect on the physical output level.

DIO109: 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.

DIO119: If development errors are enabled and an error ocurred, the Dio module’s write functions shall NOT process the write command.

7.4.2.1 DIO channel write service

DIO006: 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

DIO007: 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`.

DIO004: 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

DIO008: 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`.

7.5 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

DIO012: The Dio module’s read functions shall work on input and output channels.

DIO118: If development errors are enabled and an error ocurred the Dio module’s read functions shall return with the value '0'.

7.5.2.1 DIO channel read Service

DIO011: The `Dio_ReadChannel` function shall read the level of a single DIO channel.

7.5.2.2 DIO port read service

DIO013: 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

DIO014: The `Dio_ReadChannelGroup` function shall read the levels of a DIO channel group. A bit value '0' indicates that the corresponding pin 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

DIO083: 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.

DIO084: 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

DIO067: The Dio module shall report production errors to the Diagnostic Event Manager.

DIO065: The Dio module shall detect the following errors and exceptions depending on its build version (development/production mode).

Type or error	*Relevance*	*Related error code*	*Value*
Invalid channel name requested	Development	`DIO_E_PARAM_INVALID_CHANNEL_ID`	`10`
Invalid port name requested	Development	`DIO_E_PARAM_INVALID_PORT_ID`	`20`
Invalid ChannelGroup id passed	Development	`DIO_E_PARAM_INVALID_GROUP_ID`	`31`
--	Production	`No error code specified`	

7.7 Error detection

7.7.1 API Parameter checking

DIO074: If development error detection is enabled, the services `Dio_ReadChannel` and `Dio_WriteChannel` shall check the "Channels" parameter to be valid within the current configuration. If the "Channels" parameter is invalid, the functions shall report the error code `DIO_E_PARAM_INVALID_CHANNEL_ID` to the DET.

DIO075: If development error detection is enabled, the functions `Dio_ReadPort` and `Dio_WritePort` shall check the "Ports" parameter to be valid within the current configuration. If the "Ports" parameter is invalid, the functions shall report the error code `DIO_E_PARAM_INVALID_PORT_ID` to the DET.

DIO114: If development error detection is enabled, the functions `Dio_ReadChannelGroup` and `Dio_WriteChannelGroup` shall check the "ChannelGroupId" parameter to be valid within the current configuration. If the "ChannelGroupId" parameter is invalid, the functions shall report the error code `DIO_E_PARAM_INVALID_GROUP_ID` to the DET.

7.8 Error notification

DIO066: The detection of all development errors shall be configurable (on/off) with the preprocessor switch `DioDevErrorDetect`. The Dio module shall report detected development errors to the error hook of the Development Error Tracer (DET) if the preprocessor switch `DioDevErrorDetect` is set (see chapter 10).

DIO073: Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the DIO device specific implementation specification. The classification and enumeration shall be compatible to the errors listed above [DIO065].

8 API specification

8.1 Imported types

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

DIO131:

Module	***Imported Type***
Std_Types	Std_VersionInfoType

8.2 Type definitions

DIO103: 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

Name:	`Dio_ChannelType`		
Type:	`Unsigned Integer`		
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.		

DIO015: Parameters of type `Dio_ChannelType` contain the numeric ID of a DIO channel. The mapping of the ID is implementation specific but not configurable.

DIO017: For parameter values of type `Dio_ChannelType`, the Dio's user shall use the symbolic names provided by the configuration description.

Furthermore, DIO103 applies to the type `Dio_ChannelType`.

8.2.2 Dio_PortType

Name:	`Dio_PortType`		
Type:	`Unsigned Integer`		
Range:	`0..<number of ports>`	`--`	Shall cover all available DIO Ports.
Description:	Numeric ID of a DIO port.		

DIO018: Parameters of type `Dio_PortType` contain the numeric ID of a DIO port. The mapping of ID is implementation specific but not configurable.

DIO020: For parameter values of type `Dio_PortType`, the user shall use the symbolic names provided by the configuration description.

Furthermore, DIO103 applies to the type `Dio_PortType`.

8.2.3 Dio_ChannelGroupType

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.		

DIO021: `Dio_ChannelGroupType` is the type for the definition of a channel group, which consists of several adjoining channels within a port.

DIO022: For parameter values of type `Dio_ChannelGroupType`, the user shall use the symbolic names provided by the configuration description.

Furthermore, DIO056 applies to the type `Dio_ChannelGroupType`.

8.2.4 Dio_LevelType

Name:	`Dio_LevelType`		
Type:	`Unsigned Integer`		
Range:	`STD_LOW`	--	--
	`STD_HIGH`	--	--
Description:	These are the possible levels a DIO channel can have (input or output)		

DIO023: `Dio_LevelType` is the type for the possible levels that a DIO channel can have (input or output).

8.2.5 Dio_PortLevelType

Name:	`Dio_PortLevelType`		
Type:	`Unsigned Integer`		
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.		

DIO024: `Dio_PortLevelType` is the type for the value of a DIO port.

Furthermore, DIO103 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

DIO133:

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.	

DIO027: The `Dio_ReadChannel` function shall return the value of the specified DIO channel.

Regarding the return value of the `Dio_ReadChannel` function, the requirements [DIO083] and [DIO084] are applicable.

Furthermore, the requirements DIO005, DIO118 and DIO026 are applicable to the `Dio_ReadChannel` function.

8.3.2 Dio_WriteChannel

DIO134:

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	Level
Parameters (inout):	None	
Parameters (out):	None	

Return value:	None
Description:	Service to set a level of a channel.

DIO028: If the specified channel is configured as an output channel, the `Dio_WriteChannel` function shall set the specified Level for the specified channel.

DIO029: If the specified channel is configured as an input channel, the `Dio_WriteChannel` function shall have no influence on the physical output.

DIO079: 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 DIO005, DIO119 and DIO026 are applicable to the `Dio_WriteChannel` function.

8.3.3 Dio_ReadPort

DIO135:

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.	

DIO031: The `Dio_ReadPort` function shall return the level of all channels of that port.

DIO104: When reading a port which is smaller than the `Dio_PortType` using the `Dio_ReadPort` function (see [DIO103]), the function shall set the bits corresponding to undefined port pins to 0.

Furthermore, the requirements DIO005, DIO118 and DIO026 are applicable to the `Dio_ReadPort` function.

8.3.4 Dio_WritePort

DIO136:

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.	

DIO034: The `Dio_WritePort` function shall set the specified value for the specified port.

DIO035: When the `Dio_WritePort` function is called, DIO Channels that are configured as input shall remain unchanged.

DIO105: When writing a port which is smaller than the `Dio_PortType` using the `Dio_WritePort` function (see [DIO103]), the function shall ignore the MSB.

DIO108: The `Dio_WritePort` function shall have no effect on channels within this port which are configured as input channels.

Furthermore, the requirements DIO005, DIO119 and DIO026 are applicable to the `Dio_WritePort` function.

8.3.5 Dio_ReadChannelGroup

DIO137:

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	Dio_PortLevelType
Description:	This Service reads a subset of the adjoining bits of a port.	

DIO037: The `Dio_ReadChannelGroup` function shall read a subset of the adjoining bits of a port (channel group).

DIO092: The `Dio_ReadChannelGroup` function shall do the masking of the channel group.

DIO093: The `Dio_ReadChannelGroup` function shall do the shifting so that the values read by the function are aligned to the LSB.

Furthermore, the requirements DIO005, DIO056, DIO083, DIO084, DIO118 and DIO026 are applicable to the `Dio_ReadChannelGroup` function.

8.3.6 Dio_WriteChannelGroup

DIO138:

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.	

DIO039: The `Dio_WriteChannelGroup` function shall set a subset of the adjoining bits of a port (channel group) to a specified level.

DIO040: The `Dio_WriteChannelGroup` shall not change the remaining channels of the port and channels which are configured as input.

DIO090: The `Dio_WriteChannelGroup` function shall do the masking of the channel group.

DIO091: The function `Dio_WriteChannelGroup` shall do the shifting so that the values written by the function are aligned to the LSB.

Furthermore, the requirements DIO005, DIO056, DIO119 and DIO026 are applicable for the `Dio_WriteChannelGroup` function.

8.3.7 Dio_GetVersionInfo

DIO139:

Service name:	Dio_GetVersionInfo	

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

Service ID[hex]:	0x12	
Sync/Async:	Synchronous	
Reentrancy:	Non 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.	

DIO123: The `Dio_GetVersionInfo` function shall return the version information of this module. The version information includes:

- Module Id (See Literature [2])
- Vendor Id
- Vendor specific version numbers (BSW00407).

DIO126: If source code for caller and callee is available, the module Dio should realize the function `Dio_GetVersionInfo` as a macro defined in the module's header file.

DIO124: The `Dio_GetVersionInfo` function shall be pre-compile time configurable (`On/Off`) by the configuration parameter `DioVersionInfoApi`.

See also Chapter 10.

8.3.8 Dio_MaskedWritePort

DIO195:

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.	

DIO196: The `Dio_MaskedWritePort` function shall set the specified value for the channels in the specified port if the corresponding bit in `Mask` is '1'.

DIO197: When the `Dio_MaskedWritePort` function is called, DIO Channels that are configured as input shall remain unchanged.

DIO198: When writing a port which is smaller than the `Dio_PortType` using the `Dio_MaskedWritePort` function (see [DIO103]), the function shall ignore the MSB.

DIO199: The `Dio_MaskedWritePort` function shall have no effect on channels within this port which are configured as input channels.

Furthermore, the requirements DIO005, DIO119 and DIO026 are applicable to the `Dio_MaskedWritePort` function.

See also Chapter 10.

8.4 Call-back notifications

This chaper 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 chaper 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.

DIO140:

API function	***Description***
Det_ReportError	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

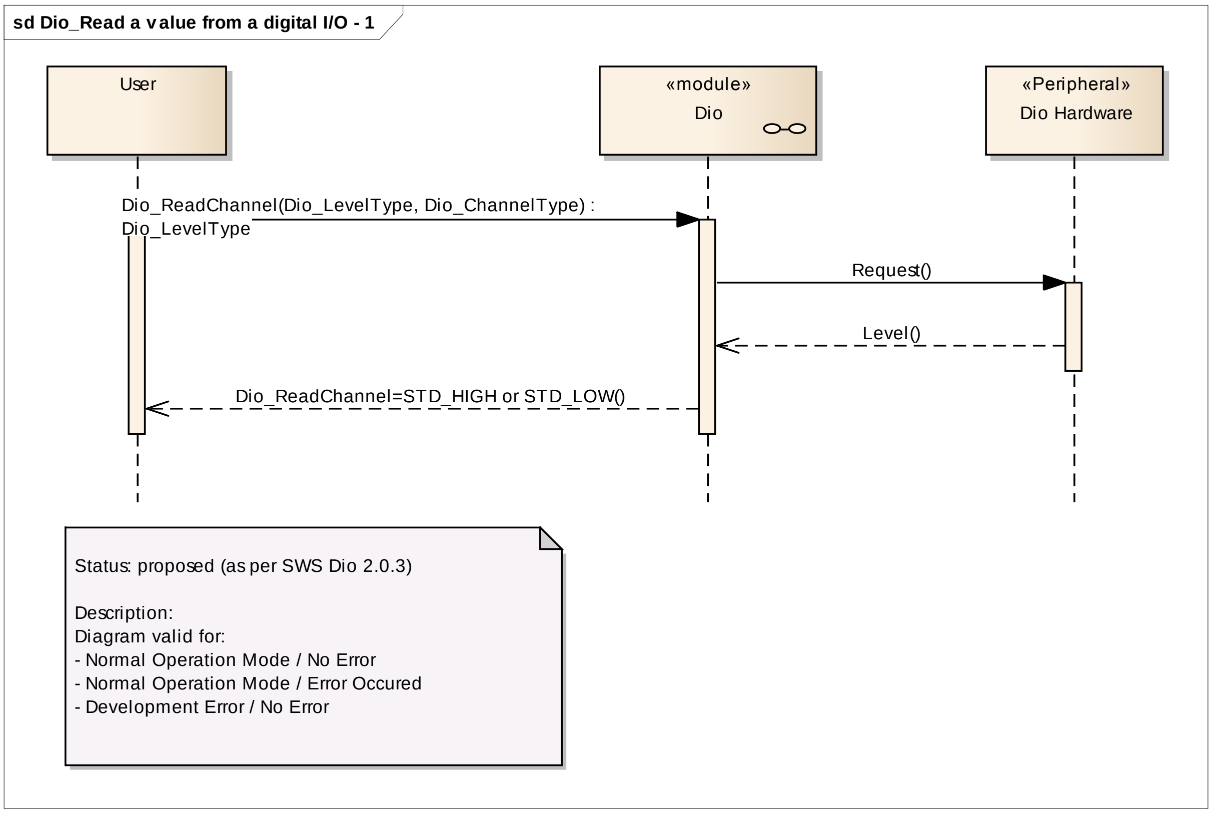
sd Dio_Read a value from a digital I/O - 1
User
«module»
Dio
«Peripheral»
Dio Hardware
Dio_ReadChannel(Dio_LevelType, Dio_ChannelType) :
Dio_LevelType
Request()
Level()
Dio_ReadChannel=STD_HIGH or STD_LOW()
Status: proposed (as per SWS Dio 2.0.3)
Description:
Diagram valid for:
- Normal Operation Mode / No Error
- Normal Operation Mode / Error Occured
- Development Error / No Error

Figure 5: Read Service Sequence Chart (normal operation mode)

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

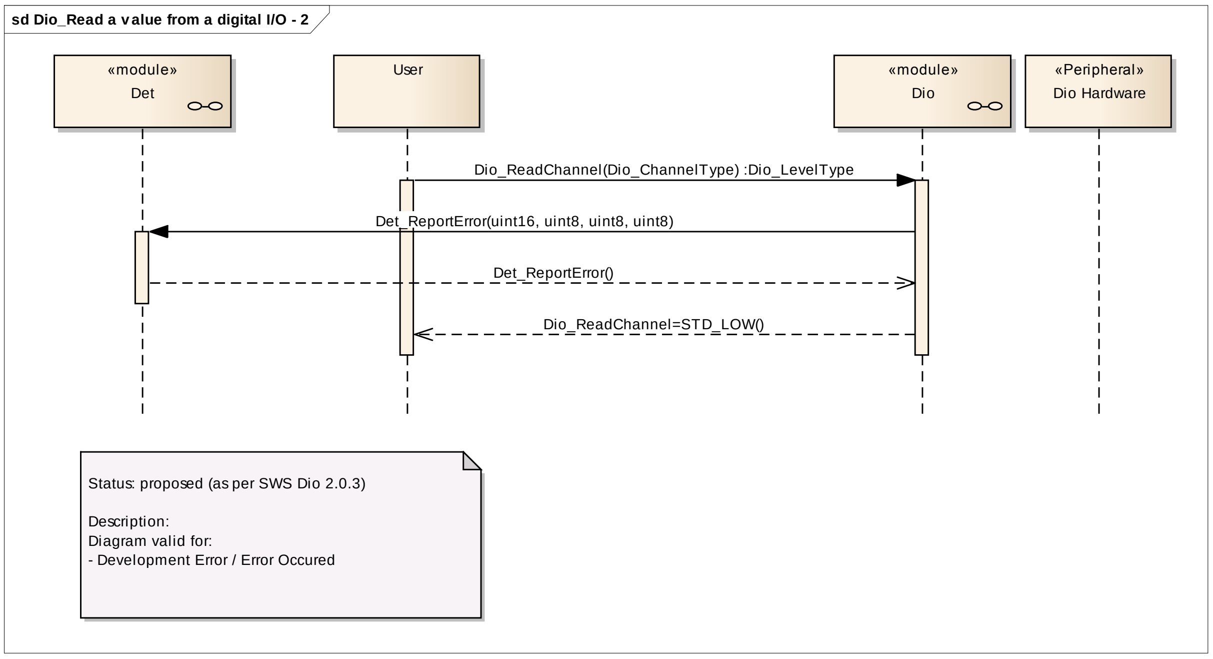
sd Dio_Read a value from a digital I/O - 2
«module»
Det
User
«module»
Dio
«Peripheral»
Dio Hardware
Dio_ReadChannel(Dio_ChannelType) :Dio_LevelType
Det_ReportError(uint16, uint8, uint8, uint8)
Det_ReportError()
Dio_ReadChannel=STD_LOW()
Status: proposed (as per SWS Dio 2.0.3)
Description:
Diagram valid for:
- Development Error / Error Occured

Figure 6: Read Service Sequence Chart (development error mode)

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

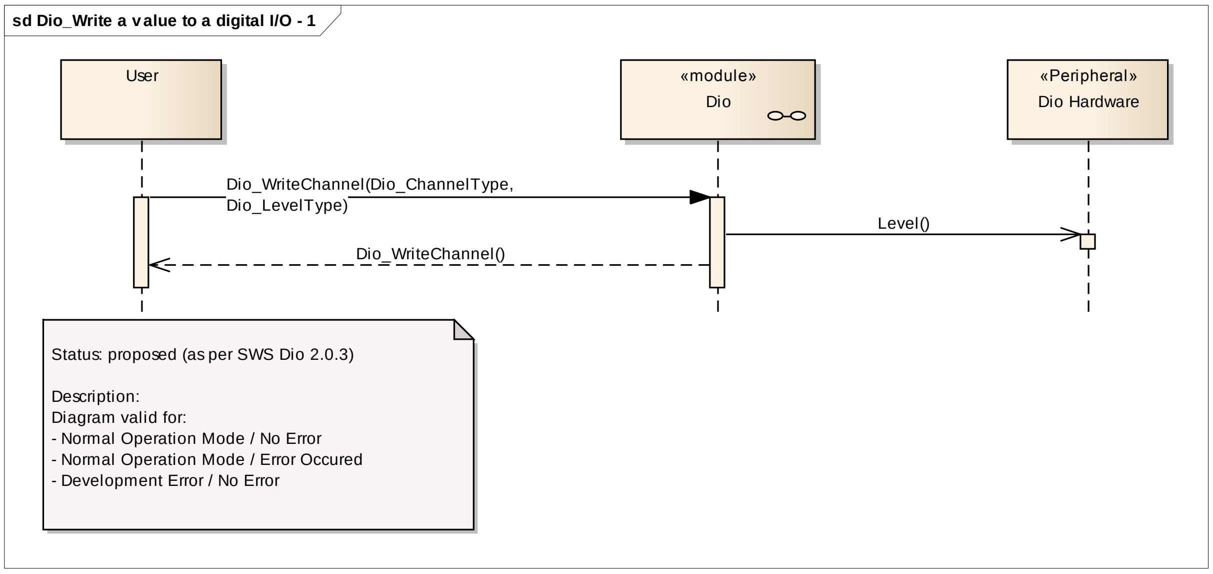
sd Dio_Write a value to a digital I/O - 1
User
«module»
Dio
«Peripheral»
Dio Hardware
Dio_WriteChannel(Dio_ChannelType, Dio_LevelType)
Level()
Dio_WriteChannel()
Status: proposed (as per SWS Dio 2.0.3)
Description:
Diagram valid for:
- Normal Operation Mode / No Error
- Normal Operation Mode / Error Occured
- Development Error / No Error

Figure 7: Write Service Sequence Chart (normal operation mode)

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

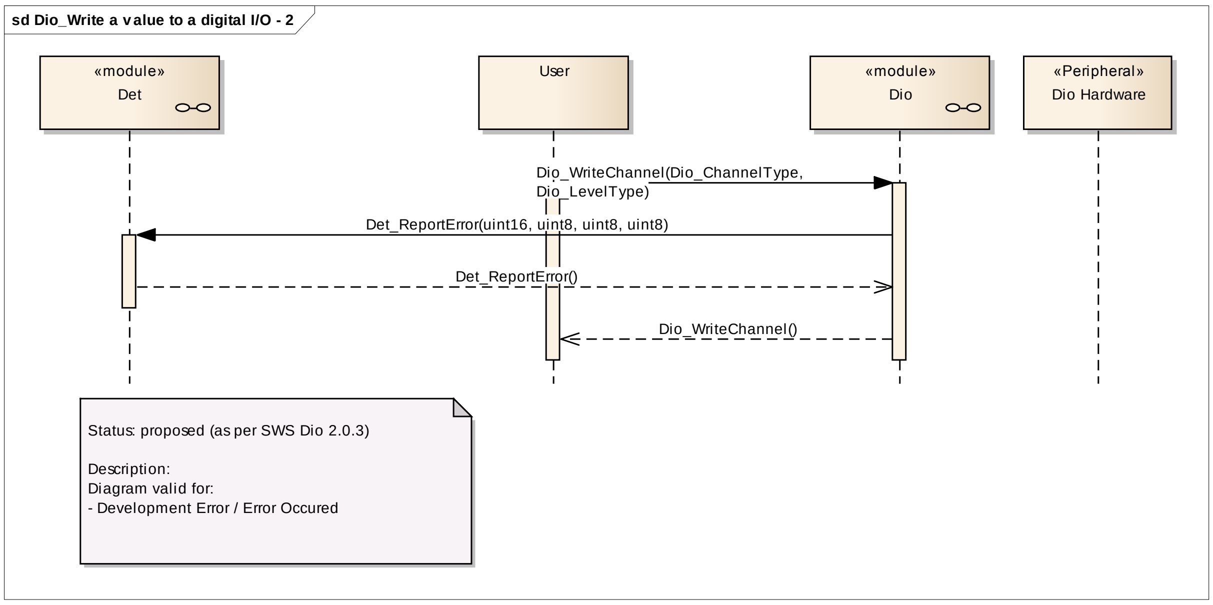
sd Dio_Write a value to a digital I/O - 2
«module»
Det
User
«module»
Dio
«Peripheral»
Dio Hardware
Dio_WriteChannel(Dio_ChannelType, Dio_LevelType)
Det_ReportError(uint16, uint8, uint8, uint8)
Det_ReportError()
Dio_WriteChannel()
Status: proposed (as per SWS Dio 2.0.3)
Description:
Diagram valid for:
- Development Error / Error Occured

Figure 8: 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.

10.1.1 Variants

DIO129: Variant PC: all configuration parameters are pre-compile time parameter.
DIO130: Variant LT: mix of pre-compile and link time

10.1.2 Dio

Module Name	*Dio*
Module Description	Configuration of the Dio (Digital IO) module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DioGeneral	1	General DIO module configuration parameters.
DioPort	1..*	Configuration of individual DIO ports, consisting of channels and possible channel groups. Nothe 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.

10.1.3 DioGeneral

SWS Item	**DIO141 :**
Container Name	DioGeneral
Description	General DIO module configuration parameters.
Configuration Parameters	

SWS Item	**DIO142 :**		
Name	DioDevErrorDetect {DIO_DEV_ERROR_DETECT}		
Description	Switches the Development Error Detection and Notification ON or OFF		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**DIO152 :**
Name	DioMaskedWritePortApi {DIO_MASKED_WRITE_PORT_API}

Description	Adds / removes the service Dio_MaskedWritePort() from the code.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**DIO143 :**		
Name	DioVersionInfoApi {DIO_VERSION_INFO_API}		
Description	Adds / removes the service Dio_ GetVersionInfo() from the code.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.1.4 DioPort

SWS Item	**DIO144 :**
Container Name	DioPort
Description	Configuration of individual DIO ports, consisting of channels and possible channel groups. Nothe 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	**DIO145 :**		
Name	DioPortId {DIO_PORT_ID}		
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	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 ..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
DioChannel	0..*	Configuration of an individual DIO channel. Besides a HW specific channel name which is typically fixed for a specific micro controller, additional symbolic names can be defined per channel. Note hat 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.
DioChannelGroup	0..*	Definition and configuration of DIO channel groups. A channel group represents several adjoining DIO channels represented by a logical group. Note hat 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.

10.1.5 DioChannel

SWS Item	**DIO146 :**
Container Name	DioChannel
Description	Configuration of an individual DIO channel. Besides a HW specific channel name which is typically fixed for a specific micro controller, additional symbolic names can be defined per channel. Note hat 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.
Configuration Parameters	

SWS Item	**DIO147 :**		
Name	DioChannelId {DIO_CHANNEL_ID}		
Description	Channel Id of the DIO channel. This value will be assigned to the symbolic names.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 ..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.1.6 DioChannelGroup

SWS Item	**DIO148 :**
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 hat 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	**DIO149 :**
Name	DioChannelGroupIdentification {DIO_CHANNEL_GROUP_IDENTIFIKATION}
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	StringParamDef (Symbolic Name generated for this parameter)

Default value	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**DIO150 :**		
Name	DioPortMask {DIO_PORT_MASK}		
Description	This shall be the mask which defines the positions of the channel group. The data type depends on the port width		
Multiplicity	1		
Type	IntegerParamDef		
Range	..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**DIO151 :**		
Name	DioPortOffset {DIO_PORT_OFFSET}		
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	DerivedIntegerParamDef		
Range	..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2 Static configuration parameters

DIO071: The following table specifies parameters that shall be definable in the module's configuration header file Dio`_Cfg.h`.

Parameter name	***Type / Range (if known)***	***Parameter description***
`DioDevErrorDetect`	`#define`	Preprocessor switch for enabling the development error detection

Symbolic names should be placed in the file Dio_Cfg.h (DIO113).

10.3 Runtime configuration parameters

DIO049: The runtime configuration is handled by the Port Driver Module.

10.4 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

The standard common published information like

vendorId (<Module>_VENDOR_ID),
moduleId (<Module>_MODULE_ID),
arMajorVersion (<Module>_AR_MAJOR_VERSION),
arMinorVersion (<Module>_ AR_MINOR_VERSION),
arPatchVersion (<Module>_ AR_PATCH_VERSION),
swMajorVersion (<Module>_SW_MAJOR_VERSION),
swMinorVersion (<Module>_ SW_MINOR_VERSION),
swPatchVersion (<Module>_ SW_PATCH_VERSION),
vendorApiInfix (<Module>_VENDOR_API_INFIX)

is provided in the BSW Module Description Template (see [6] Figure 4.1 and Figure 7.1).

Additional published parameters are listed below if applicable for this module.

10.5 Configuration Example

This chapter shall provide a better understanding of how and where configuration parameters are defined and used.

Use Cases:

1. Configuration of a DIO channel
2. Configuration of a DIO port
3. Configuration of a DIO channel group

10.5.1 Generation of DIO configuration data

10.5.1.1 Configuration of a DIO channel

Each channel with index of type `Dio_ChannelType` shall be referenced via symbolic names through the file Dio_Cfg.h.
Example:

```
#define MOTOR_START_STOP (DIO_CHANNEL_A_5)
#define MOTOR_DIRECTION (DIO_CHANNEL_A_6)
```

Where `DIO_CHANNEL_A_5` and `DIO_CHANNEL_A_6` may be defined in a derivative or board specific header file.
The mapping shall be done implementation specific.

10.5.1.2 Configuration of a DIO port

Each port with index of type `Dio_PortType` shall be referenced via symbolic names through the file Dio_Cfg.h.

Example:

```
#define MOTOR_CTL_PORT (DIO_PORT_A)
#define MUX_SEL_PORT (DIO_PORT_B)
```

Where `DIO_PORT_A` and `DIO_PORT_B` may be defined in a derivative or board specific header file.
The mapping shall be done implementation specific.

10.5.1.3 Configuration of a DIO channel group

Each channel group which is of type `Dio_ChannelGroupType` shall be referenced via symbolic names through the file Dio_Cfg.h.

Example:

```
#define MOTOR_CTL_GRP_PTR (&DioConfigData[0])
#define MUX_SEL_GRP_PTR (&DioConfigData[1])
```

For description of `DioConfigData` see section 10.5.2.

10.5.2 Instantiation of DIO configuration data

The file that contains the instantiation (=definition) of the DIO configuration structure includes `Dio_Cfg.h` and uses the defined values for initialization of structure elements. The filename should be Dio_Lcfg.c (BSW00346).

Example:

```
const Dio_ChannelGroupType DioConfigData[2] =
{
   {
      port    = MOTOR_CTL_PORT,
      offset  = 5,
      mask    = 0x60,
   },
   {
      port    = MUX_SEL_PORT,
      offset  = 1,
      mask    = 0x1E,
   }
};
```