Document Title	Specification of DIO Driver
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	2.1.0
Document Status	Final
Part of Release	2.1
Revision	0014

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, Changes to Release 1
30.06.2005	1.0.0	AUTOSAR Administration	Initial Release

Page left intentionally blank

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).
Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

.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.

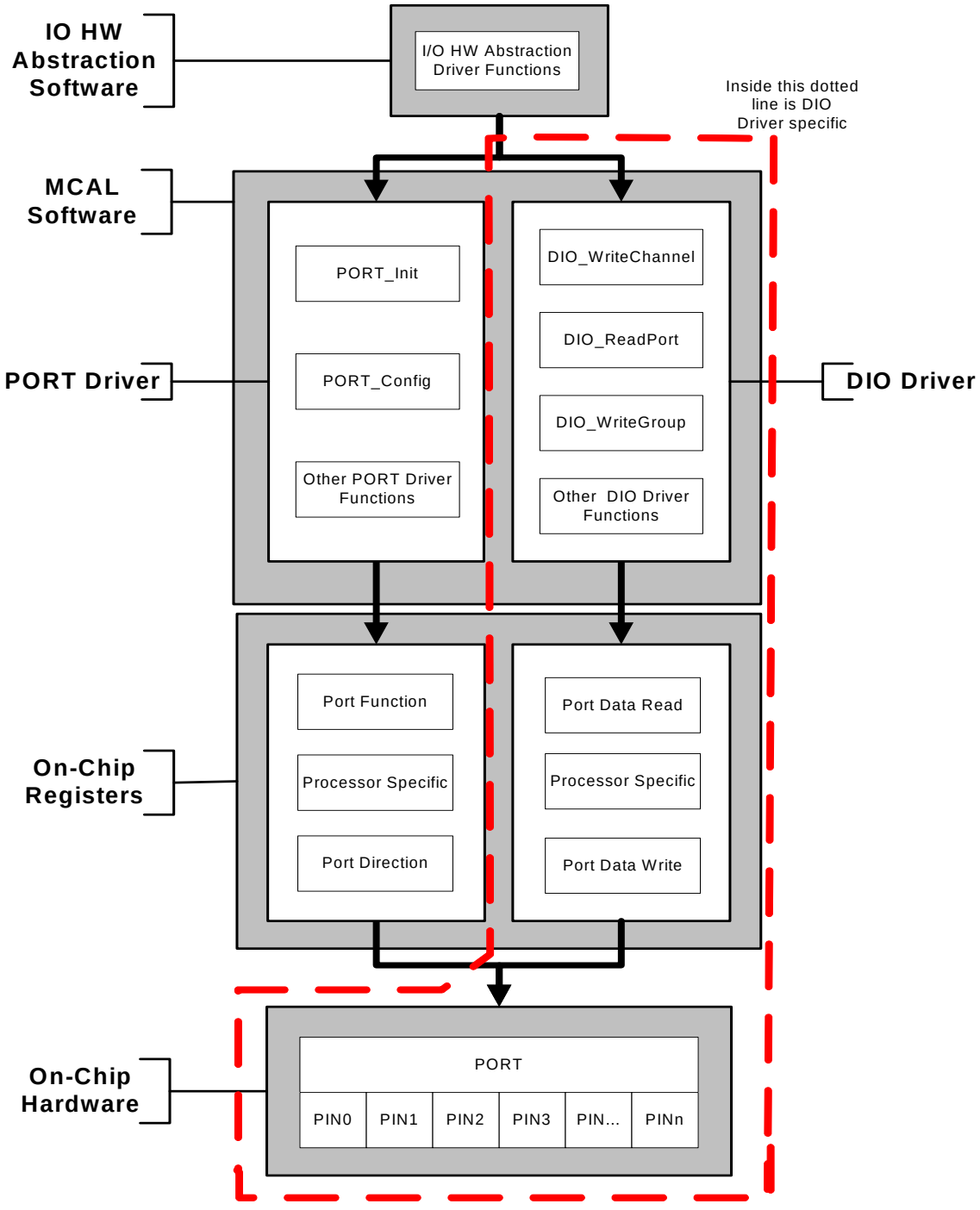

Figure 1: DIO Driver Structure and Integration

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

3 Related documentation

3.1 Deliverables of AUTOSAR

[1] Layered Software Architecture
https://svn.autosar.org/repos/10Releases/
AUTOSAR_LayeredSoftwareArchitecture.pdf

[2] List of Basic Software Modules
https://svn.autosar.org/repos/10Releases/
AUTOSAR_BasicSoftwareModules.pdf

[3] General Requirements on SPAL
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_SPAL_General.pdf

[4] General Requirements on Basic Software Modules
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_General.pdf

[5] Specification of ECU Configuration
https://svn.autosar.org/repos/10Releases/
AUTOSAR_ECU_Configuration.pdf

[6] Specification of PORT Driver,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_PORT_Driver.pdf

[7] Specification of Standard Types,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_StandardTypes.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 like

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

DIO061: The overall configuration and initialization of the port structure which is used in the DIO Driver Module will be done by the PORT Driver Module. DIO Driver module shall not provide APIs for these actions.

DIO063: The configuration and usage of the DIO Driver Module shall be adapted to the microcontroller and ECU.

DIO102: The Port Driver shall be initialized prior to the use of DIO functions. Otherwise DIO functions will exhibit undefined behavior.

5.1 File structure

DIO117: The file structure of the DIO driver shall be as follows.

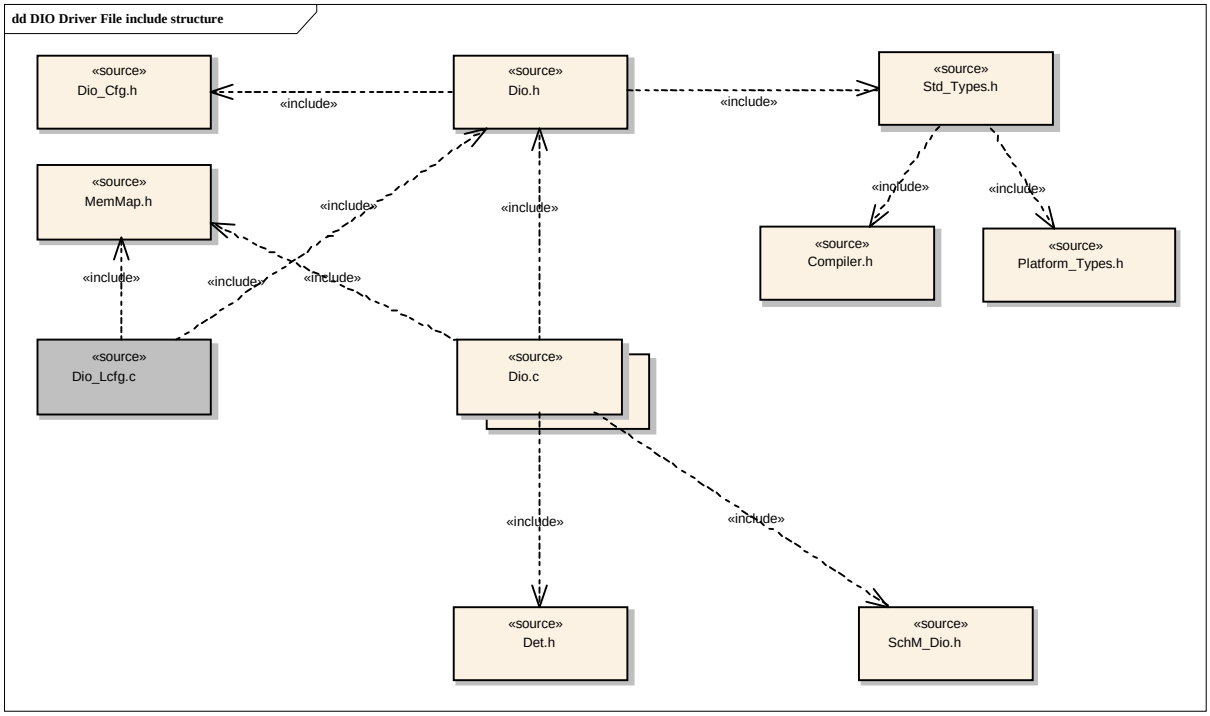

Figure 2: Include File Structure

`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 DIO_VERSION_INFO_API
Dio_GetVersionInfo(...)
#endif

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

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

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	Not applicable (it is a non functional requirement)
[BSW159] Tool-based configuration	Not applicable (it is a non functional requirement)
[BSW160] Human-readable configuration data	Not applicable (it only applies to the configuration)
[BSW161] Microcontroller abstraction	Not applicable (it is a non functional requirement)
[BSW162] ECU layout abstraction	Not applicable (it is a non functional requirement)
[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	Not applicable (it is a non functional requirement)
[BSW00301] Limit imported information	DIO117
[BSW00302] Limit exported information	DIO117

[BSW00304] AUTOSAR integer data types	Not applicable (it is a non functional requirement)
[BSW00305] Self-defined data types naming convention	Not applicable (it is a non functional requirement)
[BSW00306] Avoid direct use of compiler and platform specific keywords	Not applicable (it is a non functional requirement)
[BSW00307] Global variables naming convention	Not applicable (it is a non functional requirement)
[BSW00308] Definition of global data	Not applicable (it is a non functional requirement)
[BSW00309] Global data with read-only constraint	Not applicable (it is a non functional requirement)
[BSW00310] API naming convention	Not applicable (it is a non functional requirement)
[BSW00312] Shared code shall be reentrant	Not applicable (it is a non functional requirement)
[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	DIO048, DIO065, DIO074, DIO075, DIO121
[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	Not applicable (it is a non functional requirement)
[BSW00328] Avoid duplication of code	Not applicable (it is a non functional requirement)
[BSW00329] Avoidance of generic interfaces	Not applicable (it is a non functional requirement)
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable (it is a non functional requirement)
[BSW00331] Separation of error and status values	Not applicable (it is a non functional requirement)
[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 (it is a non functional requirement)
[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, DIO048, DIO073, DIO067, DIO120, DIO121
[BSW00339] Reporting of production relevant errors and exceptions	Not applicable (DIO only provides development errors)
[BSW00341] Microcontroller compatibility documentation	Not applicable (it is a non functional requirement)
[BSW00342] Usage of source code and object code	Not applicable (it is a non functional requirement)

[BSW00343] Specification and configuration of time	Not applicable (it is a non functional requirement)
[BSW00344] Link-time configuration	DIO001, DIO002
[BSW00345] Pre-compile-time configuration	DIO071
[BSW00346] Basic set of module files	DIO117
[BSW00347] Naming separation of drivers	Not applicable (it is a non functional requirement)
[BSW00348] Standard type header	Not applicable (it is a non functional requirement)
[BSW00350] Development error detection keyword	DIO120
[BSW00353] Platform specific type header	Not applicable (it is a non functional requirement)
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable (it is a non functional requirement)
[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	Not applicable (it is a non functional requirement)
[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 (it is a non functional requirement)
[BSW00373] Main processing function naming convention	Not applicable (it is a non functional requirement)
[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 (it is a non functional requirement)
[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	Not applicable (it is a non functional requirement)

[BSW00384] List dependencies to other modules	Not applicable (it is a non functional requirement)
[BSW00385] List possible error notificatons	Not applicable (it is a non functional requirement)
[BSW00386] Configuration for detecting an error	Not applicable (it is a non functional requirement)
[BSW00387] Specify the configuration class of callback function	Not applicable (it is a non functional requirement)
[BSW00388] Introduce containers	Not applicable (it is a non functional requirement)
[BSW00389] Containers shall have names	Not applicable (it is a non functional requirement)
[BSW00390] Parameter content shall be unique within the module	Not applicable (it is a non functional requirement)
[BSW00391] Parameter shall have unique names	Not applicable (it is a non functional requirement)
[BSW00392] Parameters shall have a type	Not applicable (it is a non functional requirement)
[BSW00393] Parameters shall have a range	Not applicable (it is a non functional requirement)
[BSW00394] Specify the scope of the parameters	Not applicable (it is a non functional requirement)
[BSW00395] the required parameters (per parameter)	Not applicable (it is a non functional requirement)
[BSW00396] Configuration classes	Not applicable (it is a non functional requirement)
[BSW00397] Pre-compile-time parameters	Not applicable (it is a non functional requirement)
[BSW00398] Link-time parameters	Not applicable (it is a non functional requirement)
[BSW00399] Loadable Post-build time parameters	Not applicable (it is a non functional requirement)
[BSW00400] Selectable Post-build time parameters	Not applicable (it is a non functional requirement)
[BSW00401] Documentation of multiple instances of configuration parameters	Not applicable (it is a non functional requirement)
[BSW00402] Published information	Not applicable (it is a non functional requirement)
[BSW00404] Reference to post build time configuration	Not applicable (it is a non functional requirement)
[BSW00405] Reference to multiple configuration sets	Not applicable (it is a non functional requirement)
[BSW00406] Check module initialization	Not applicable (it is a non functional requirement)
[BSW00407] Function to read out published parameters	DIO123
[BSW00408] Configuration parameter naming convention	Not applicable (it is a non functional requirement)
[BSW00409] Header files for production code error IDs	Not applicable (it is a non functional requirement)
[BSW00410] Compiler switches shall have defined values	DIO124
[BSW00411] Get version info keyword	Not applicable (it is a non functional requirement)
[BSW00412] Separate H-File for configuration parameters	DIO117
[BSW00413] Accessing instances of BSW modules	Not applicable (it is a non functional requirement)
[BSW00414] Parameter of init function	Not applicable (it is a non functional requirement)

[BSW00415] User dependent include files	Not applicable (it is a non functional requirement)
[BSW00416] Sequence of Initialization	Not applicable (it is a non functional requirement)
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable (it is a non functional requirement)
[BSW00419] Separate C-Files for pre-compile time configuration parameters	DIO117
[BSW00420] Production relevant error event rate detection	Not applicable (it is a non functional requirement)
[BSW00421] Reporting of production relevant error events	Not applicable (it is a non functional requirement)
[BSW00422] Debouncing of production relevant error status	Not applicable (it is a non functional requirement)
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (it is a non functional requirement)
[BSW00424] BSW main processing function task allocation	Not applicable (it is a non functional requirement)
[BSW00425] Trigger conditions for schedulable objects	Not applicable (it is a non functional requirement)
[BSW00426] Exclusive areas in BSW modules	Not applicable (it is a non functional requirement)
[BSW00427] ISR description for BSW modules	Not applicable (it is a non functional requirement)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (it is a non functional requirement)
[BSW00429] Restricted BSW OS functionality access	Not applicable (it is a non functional requirement)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (it is a non functional requirement)
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (it is a non functional requirement)
[BSW00433] The Schedule Module shall provide an API for exclusive areas	Not applicable (it is a non functional requirement)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (it is a non functional requirement)
[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	DIO101
[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	DIO048, 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, DIO052, 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, DIO052, DIO011, DIO027, DIO085, 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 shall abstract the access to the microcontroller's hardware pins. Furthermore, it shall allow the grouping of those pins.

7.1.2 Requirements

DIO050: The DIO Driver shall provide
- Port
- Channel
- Channel group

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

DIO051: All read and write services in the DIO Driver shall be un-buffered.

DIO055: The basic behaviour of this driver shall be synchronous.

DIO005: The DIO 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 shall be either STD_HIGH or STD_LOW.

DIO052: A general-purpose digital IO pin represents a **DIO channel**. A DIO channel may be configured as input or output by the PORT driver [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: Several adjoining DIO channels within a DIO port can be combined to a formal logical **channel group**.

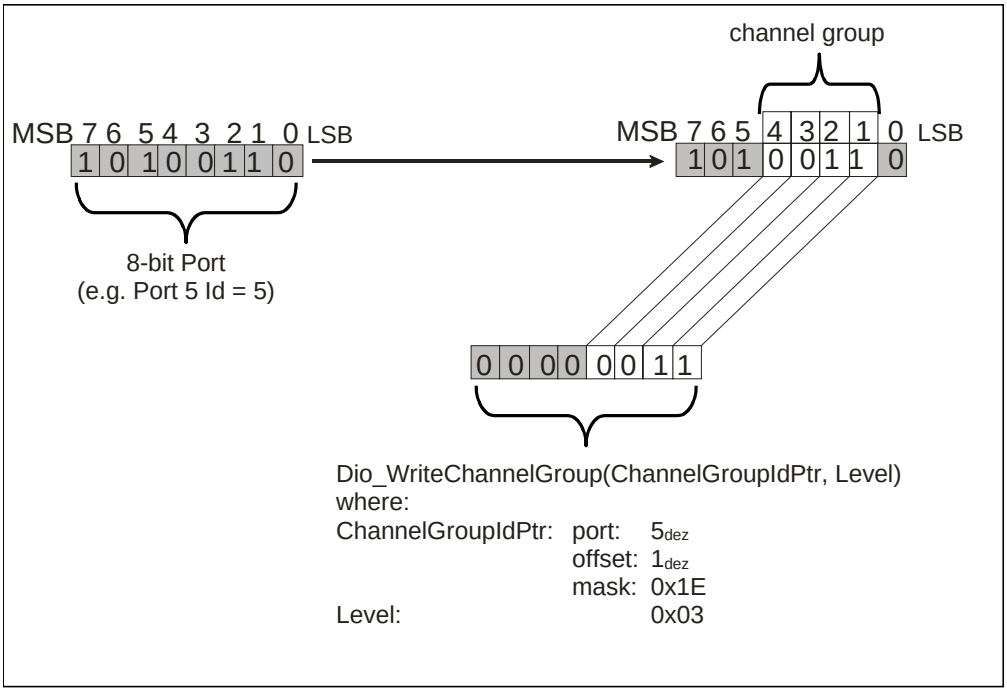

Figure 3: Schematic description of a ChannelGroup

The DIO Driver provides the following services:

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

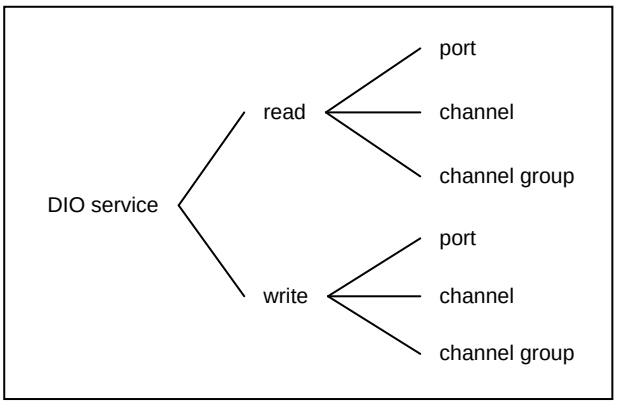

Figure 4: DIO Services

DIO060: All read and write services in the DIO Driver 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.

DIO026: Symbolic names for each configured DIO channel, port and group shall be provided by the configuration process.

DIO113: Symbolic Names shall be published in file “Dio_Cfg.h” which is created from the configuration process.

7.1.3 Version check

7.1.3.1 Background & Rationale

The integration of incompatible files shall 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 integration of incompatible files shall be avoided. Minimum implementation is the version check 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 shall be done by the PORT Driver.

7.2.2 Requirements

DIO001: The DIO shall not provide an interface for initialization. The Port Driver performs this.

7.3 Runtime reconfiguration

7.3.1 Background & Rationale

Runtime reconfiguration shall be provided by the PORT Driver.

7.3.2 Requirements

DIO002: The reconfiguration of the port pin direction during runtime is provided by PORT driver.

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 write services shall work on input and output channels.

DIO070: DIO write services operated on an input channel shall have no effect on the physical output level.

DIO109: If supported by hardware, the output data latch of an input channel will be set/cleared 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 write services shall NOT process the write command.

7.4.2.1 DIO channel write service

DIO006: The level of a single DIO channel shall be set STD_HIGH or STD_LOW with the service Dio_WriteChannel.

7.4.2.2 DIO port write service

DIO007: The levels of all output channels of a port shall be set simultaneously with the service Dio_WritePort. 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 port write service shall ensure that the functionality of the input channels of that port are not affected.

7.4.2.3 DIO channel group write service

DIO008: An adjoining subset of DIO channels (channel group) of a port shall be set simultaneously with the function Dio_WriteChannelGroup. 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 read services shall work on input and output channels.

DIO118: If development errors are enabled and an error ocurred the read services shall return with the value '0'.

7.5.2.1 DIO channel read Service

DIO011: The level of a single DIO channel shall be read with the service Dio_ReadChannel.

7.5.2.2 DIO port read service

DIO013: The levels of all channels of one port shall be read with Dio_ReadPort. A bit value '0' indicates the corresponding channel to physical STD_LOW, a bit value '1' indicates the corresponding channel to physical STD_HIGH.

7.5.2.3 DIO channel group read service

DIO014: The levels of a DIO channel group are read with the function Dio_ReadChannelGroup. A bit value '0' indicates the corresponding pin to physical STD_LOW, a bit value '1' indicates the corresponding channel to 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 read services 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, DIO read services 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: Production errors shall be reported to the Diagnostic Event Manager.

DIO065: The following errors and exceptions shall be detectable by the DIO Driver 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

DIO120: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time.
The switch `DIO_DEV_ERROR_DETECT` (see chapter 10) shall activate or deactivate the detection of all development errors.

DIO121: If the `DIO_DEV_ERROR_DETECT` switch is enabled, API parameter checking is enabled.

7.7.1 API Parameter checking

DIO048: If the development error detection is enabled for this module, the following API parameter checks shall be performed within the services `Dio_ReadChannel()`, `Dio_WriteChannel()`, `Dio_ReadPort()`, `Dio_WritePort()`, `Dio_ReadChannelGroup()` and `Dio_WriteChannelGroup()`. Detected errors shall be reported to the Development Error Tracer (see DIO065).

DIO074: For `Dio_ReadChannel` and `Dio_WriteChannel`: Channels shall be valid within the current configuration. Related error value:
DIO_E_PARAM_INVALID_CHANNEL_ID

DIO075: For `Dio_ReadPort` and `Dio_WritePort`: Ports shall be valid within the current configuration. Related error value:

DIO_E_PARAM_INVALID_PORT_ID

DIO114: For `Dio_ReadChannelGroup` and `Dio_WriteChannelGroup:` Passing of invalid ChannelGroupId parameters. Related error value: DIO_E_PARAM_INVALID_GROUP_ID.

7.8 Error notification

DIO066: The detection of all development errors shall be configurable (on/off) with the preprocessor switch `DIO_DEV_ERROR_DETECT.` Detected development errors shall be reported to the error hook of the Development Error Tracer (DET) if the preprocessor switch `DIO_DEV_ERROR_DETECT` 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

8.1.1 Standard types

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

- Std_Types.h

The following types are imported:

- `STD_LOW`
- `STD_HIGH`
- `Std_VersionInfoType`

8.1.2 Other Module types

- no types from other modules are imported.

8.2 Type definitions

DIO103: The port width within these types shall be the size of the largest port on the MCU which may be accessed by this driver.

8.2.1 Dio_ChannelType

Type:	`uint8..uint32`
Range:	This is implementation specific but not all values may be valid within the type. Shall cover all available DIO channels
Description:	**DIO015:** Numeric ID of a DIO channel. The mapping of ID is implementation specific but not configurable. **DIO017:** The user shall use the symbolic names provided by configuration description. DIO103

8.2.2 Dio_PortType

Type:	`uint8..uint32`
Range:	0..<number of ports> Shall cover all available DIO Ports
Description:	**DIO018:** Numeric ID of a DIO port. The mapping of ID is implementation specific but not configurable. **DIO020:** The user shall use the symbolic names provided by configuration description. DIO103

8.2.3 Dio_ChannelGroupType

Type:	`struct`		
Element:	`Dio_PortType`	`port`	This shall be the port on which the Channel group is defined.
Element:	`uint8`	`offset`	This shall be the position of the Channel Group on the port, counted from the LSB.
Element:	`uint8..uint32`	`mask`	This shall be the mask which defines the positions of the channel group. The data type depends on the port width.
Description:	**DIO021:** Type for the definition of a channel group, which consists of several adjoining channels within a port. **DIO022:** The user shall use the symbolic names provided by configuration description. DIO056		

8.2.4 Dio_LevelType

Type:	`uint8..uint32`	
Range:	`STD_LOW`	as defined in [7]
	`STD_HIGH`	as defined in [7]
Description:	**DIO023:** These are the possible levels a DIO channel can have (input or output)	

8.2.5 Dio_PortLevelType

Type:	`uint8..uint32`	
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:	**DIO024:** Value of a DIO port. DIO103	

8.3 Function definitions

This is a list of functions provided for upper layer modules.

8.3.1 Dio_ReadChannel

Service name:	Dio_ReadChannel
Syntax:	`Dio_LevelType Dio_ReadChannel ( Dio_ChannelType ChannelId )`
Service ID:	0
Sync/Async.:	Synchronous
Reentrancy:	Re-entrant

Parameters (in):	ChannelId	ID of DIO channel
Parameters (out):	None	
Return value:	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:	**DIO027:** Returns the value of the specified DIO channel. **DIO085:** The return value depends on [DIO083] and [DIO084]. DIO005, DIO118	
Caveats:	-	
Configuration:	DIO026	

8.3.2 Dio_WriteChannel

Service name:	Dio_WriteChannel

Syntax:	`void Dio_WriteChannel`
	`(`
	`  Dio_ChannelType ChannelId,`
	`  Dio_LevelType  Level`
	`)`

Service ID:	1
Sync/Async:	Synchronous
Reentrancy:	Re-entrant
Parameters (in):	ChannelId — ID of DIO channel
	Level — Level
Parameters (out):	None
Return value:	None
Description:	**DIO028:** If the specified channel is configured as an output channel, the Dio_WriteChannel service shall set the specified Level for the specified channel. **DIO029:** If the specified channel is configured as an input channel, the Dio_WriteChannel service shall have no influence on the physical output. **DIO079:** If the specified channel is configured as an input channel, the Dio_WriteChannel service shall have no influence on the result of the next Read-Service. DIO005, DIO119
Caveats:	--
Configuration:	DIO026

8.3.3 Dio_ReadPort

Service name:	Dio_ReadPort

Syntax:	`Dio_PortLevelType Dio_ReadPort`
	`(`
	`  Dio_PortType PortId`
	`)`

Service ID:	2
Sync/Async:	Synchronous
Reentrancy:	Re-entrant

Parameters (in):	PortId	ID of DIO Port
Parameters (out):	None	
Return value:	Dio_PortLevelType	The return value depends on [DIO083] and [DIO084].
Description:	**DIO031:** Returns the level of all channels of that port. **DIO104:** When reading a port which is smaller than the Dio_PortType (see [DIO103]) the bits corresponding to undefined port pins shall be set to 0. DIO005, DIO118	
Caveats:	-	
Configuration:	DIO026	

8.3.4 Dio_WritePort

Service name:	Dio_WritePort	

Syntax:	`void Dio_WritePort`
	`(`
	`  Dio_PortType  PortId,`
	`  Dio_PortLevelType Level`
	`)`	

Service ID:	3	
Sync/Async:	Synchronous	
Reentrancy:	Re-entrant	
Parameters (in):	PortId	ID of DIO Port
	Level	Value to be written
Parameters (out):	None	
Return value:	None	
Description:	**DIO034:** The Dio_WritePort service shall set the specified value for the specified port. **DIO035:** DIO Channels that are configured as input shall remain unchanged. **DIO105:** When writing a port which is smaller than the Dio_PortType (see [DIO103]) the MSB shall be ignored. **DIO108:** The DIO port write service shall have no effect on channel in this port which are configured as input channels. DIO005, DIO119	
Caveats:	-	
Configuration:	DIO026	

8.3.5 Dio_ReadChannelGroup

Service name:	Dio_ReadChannelGroup
Syntax:	`Dio_PortLevelType Dio_ReadChannelGroup ( const Dio_ChannelGroupType *ChannelGroupIdPtr )`
Service ID:	4
Sync/Async:	Synchronous
Reentrancy:	Re-entrant
Parameters (in):	`ChannelGroupIdPtr` Pointer to ChannelGroup
Parameters (out):	None
Return value:	Dio_PortLevelType The return value depends on [DIO083] and [DIO084].
Description:	**DIO037:** The Dio_ReadChannelGroup shall read a subset of the adjoining bits of a port (channel group). **DIO092:** The Dio_ReadChannelGroup service shall do the masking of the channel group. **DIO093:** The values read by the Dio_ReadChannelGroup service are aligned to the LSB. DIO005, DIO056 DIO083, DIO084, DIO118
Caveats:	-
Configuration:	DIO026

8.3.6 Dio_WriteChannelGroup

Service name:	Dio_WriteChannelGroup
Syntax:	`void Dio_WriteChannelGroup ( const Dio_ChannelGroupType *ChannelGroupIdPtr, Dio_PortLevelType Level )`
Service ID:	5
Sync/Async:	Synchronous
Reentrancy:	Re-entrant
Parameters (in):	`ChannelGroupIdPtr` Pointer to ChannelGroup `Level` Value to be written
Parameters (out):	None
Return value:	None
Description:	**DIO039:** The Dio_WriteChannelGroup service shall set a subset of the adjoining bits of a port (channel group) to a specified level. .**DIO040:** The remaining channels of the port and channels which are configured as input remain unchanged. **DIO090:** The Dio_WriteChannelGroup service shall do the masking of the channel group. **DIO091:** The values written by the Dio_WriteChannelGroup service are aligned to the LSB, the service shall do the shifting.

	DIO005, DIO056, DIO119
Caveats:	-
Configuration:	DIO026

8.3.7 Dio_GetVersionInfo

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 (out):	`versioninfo` Pointer to where to store the version information of this module.
Return value:	none
Description:	**DIO123:** This service returns the version information of this module. The version information includes: - Module Id (See Literature [2]) - Vendor Id - Vendor specific version numbers (BSW00407). Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.
Caveats:	None
Configuration:	**DIO124:** This service shall be pre compile time configurable `On/Off` by the configuration parameter: `DIO_VERSION_INFO_API.` See also chapter 0, “ Configuration of optional API services”.

8.4 Call-back notifications

DIO101: There are no callback notifications from the DIO Driver. These are implemented in another module (ICU Driver and/or complex drivers).

8.5 Scheduled functions

There are no scheduled functions within the DIO Driver.

8.6 Expected Interfaces

In this chapter all interfaces required from other modules are listed.

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.

API function	Det_ReportError
Module	DET
Description	Development error notification
Configuration parameter	DIO_DEV_ERROR_DETECT

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

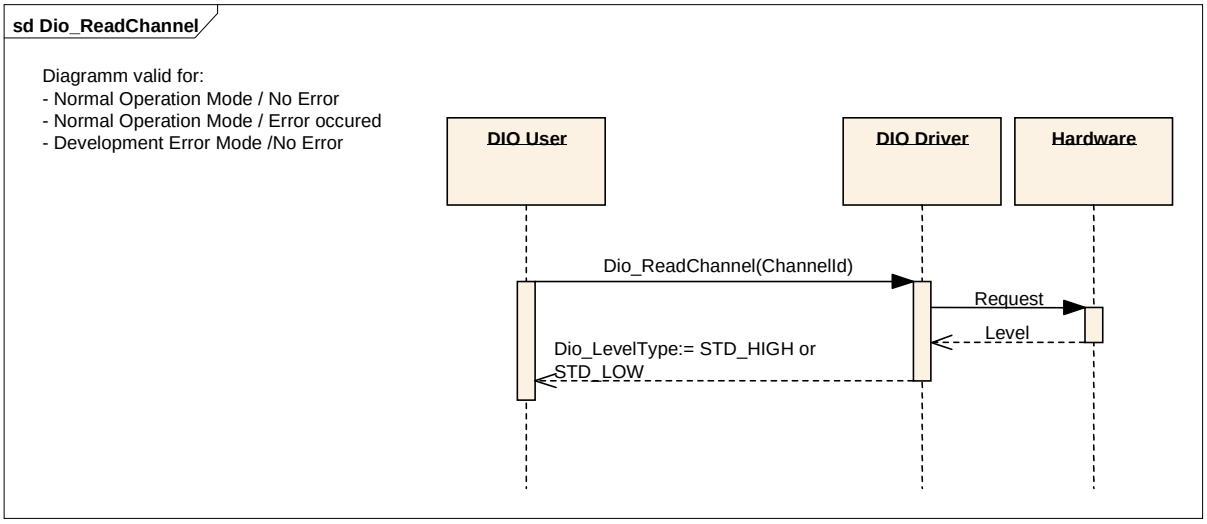

Figure 5: Read Service Sequence Chart (normal operation mode)

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

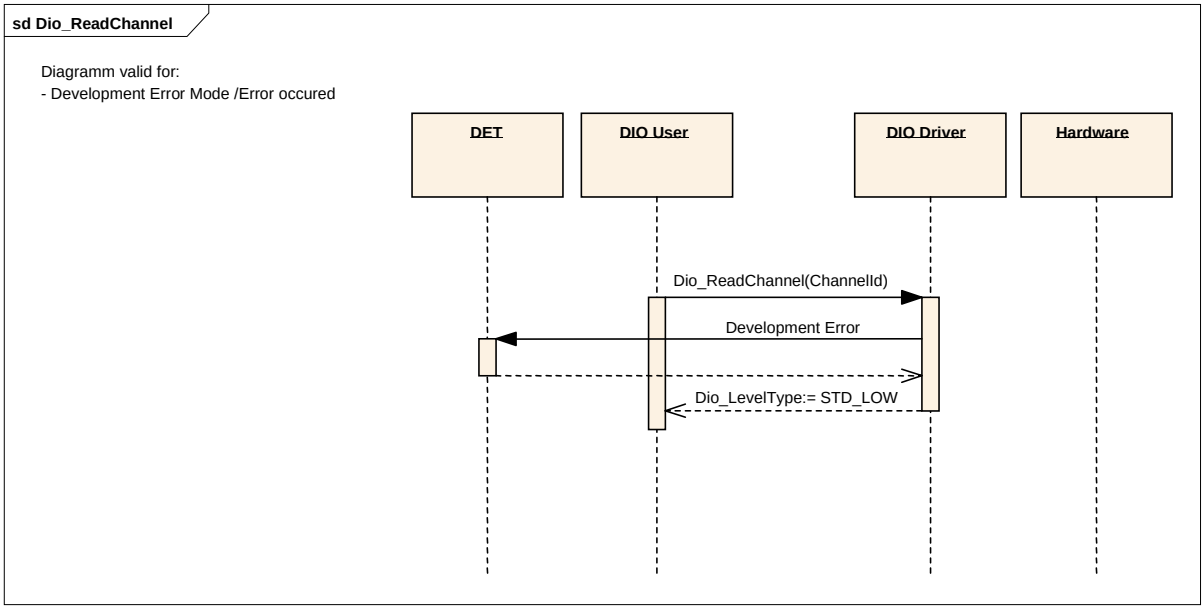

Figure 6: Read Service Sequence Chart (development error mode)

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

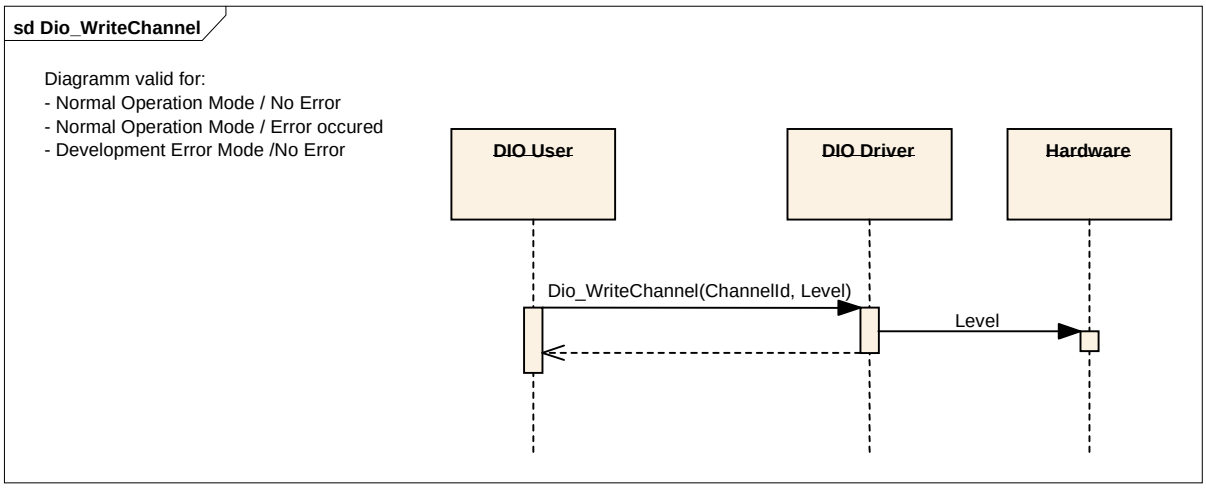

Figure 7: Write Service Sequence Chart (normal operation mode)

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

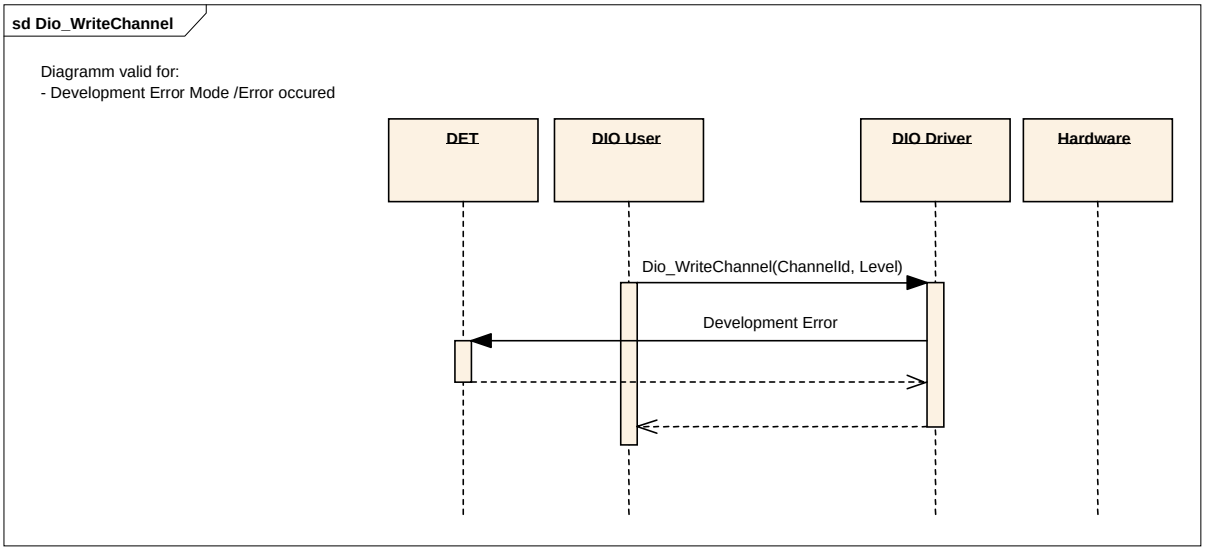

Figure 8: Write Service Sequence Chart (development error mode)

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

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

10.1.2 DioDriver

The top level DioDriver container holds parameters that apply to the complete DIO configuration and all DioPort subcontainers.

SWS Item	
Container Name	DioDriver
Description	Configuration of individual DIO ports, channels and channel groups as well as overall configurations for the DIO driver..
Configuration Parameters	

Name	DIO_DEV_ERROR_DETECT		
Description	Switches the Development Error Detection and Notification ON or OFF.		
Type	`#define`		
Unit	`--`		
Range	`ON`	enabled	
	`OFF`	disabled	
Configuration Class	***Pre-compile***	x	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	none		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
Configuration of optional API services (see chapter 0)	1	The module contains configuration switches for configuring optional API services of the DIO driver.
DioPort	1..*	Configuration of individual DIO ports, consisting of channels and possible channel groups.

10.1.3 DioPort

The DioPort container holds configuration parameters that apply to complete ports, 1 plus the channels and channel groups defined within that port (as subcontainers). Which individual channels are part of a port is typically defined by hardware.

SWS Item	
Container Name	DioPort
Description	Configuration of individual DIO ports, consisting of channels and possible channel groups.
Configuration Parameters	

Name	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.		
Type	`Dio_PortType` (symbolic name generated for this parameter)		
Unit	`--`		
Range	0..<number of ports>		
Configuration Class	***Pre-compile***	x	PC
	Link time	x	LT
	Post Build	--	--
Scope	Module		
Dependency	none		

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. Dependency: If there is no DioChannelGroup instantiated there has to be at least one instance of DioChannel
DioChannelGroup	0..*	Definition and configuration of DIO channel groups. A channel group represents several adjoining DIO channels represented by a logical group. Dependency: If there is no DioChannel instantiated there has to be at least one instance of DioChannelGroup

10.1.4 DioChannel

The DioChannel container holds configuration parameters that apply to each channel.

SWS Item	
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.
Configuration Parameters	

Name	DIO_CHANNEL_ID		
Description	Channel Id of the DIO channel.		
Type	`Dio_ChannelType` (symbolic name generated for this parameter)		
Unit	`--`		
Range	0..<number of channels>		
Configuration Class	***Pre-compile***	x	PC
	Link time	x	LT
	Post Build	--	--
Scope	Module		
Dependency	none		

10.1.5 DioChannelGroup

The DioChannelGroup container holds configuration parameters that apply to each channel group.

SWS Item	
Container Name	DioChannelGroup
Description	Definition and configuration of DIO channel groups. A channel group represents several adjoining DIO channels represented by a logical group.
Configuration Parameters	

Name	DIO_PORT_OFFSET		
Description	This shall be the position of the Channel Group on the port, counted from the LSB		
Type	`uint8`		
Unit	--		
Range	0..<width of ports>		
Configuration Class	***Pre-compile***	x	PC
	Link time	x	LT
	Post Build	--	--
Scope	Module		
Dependency	none		

Name	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		
Type	`uint8..uint32`		
Unit	--		
Range	0..<range of data type>		
Configuration Class	***Pre-compile***	x	PC
	Link time	x	LT
	Post Build	--	--
Scope	Module		
Dependency	none		

Name	DIO_CHANNEL_GROUP_IDENTIFICATION		
Description	used as a symbolic name to reference this container. 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.		
Type	`String`		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	x	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	none		

10.1.6 Configuration of optional API services

SWS Item	**DIO125:**
Container Name	Configuration of optional API services
Description	This container contains all configuration switches for configuring optional API services of the DIO driver.
Configuration Parameters	

Name	DIO_VERSION_INFO_API		
Description	Adds / removes the service Dio_ GetVersionInfo() from the code.		
Type	#define		
Unit	--		
Range	ON	Dio_GetVersionInfo() can be used	
	OFF	Dio_GetVersionInfo() can not be used	
Configuration Class	***Pre-compile***	x	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

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***
DIO_DEV_ERROR_DETECT	#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.

SWS Item	DIO082: The following table specifies parameters that shall be published in the module's header file Dio.h and also in the module's description file.	
Information elements		
Information element name	***Type / Range***	***Information element description***

DIO_MODULE_ID	#define / uint8	Module ID of this module from Module List
DIO_AR_MAJOR_VERSION	#define / uint8	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
DIO_AR_MINOR_VERSION	#define / uint8	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
DIO_AR_PATCH_VERSION	#define / uint8	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
DIO_SW_MAJOR_VERSION	#define / uint8	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
DIO_SW_MINOR_VERSION	#define / uint8	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
DIO_SW_PATCH_VERSION	#define / uint8	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

SWS Item	**DIO115:** The following table specifies parameters that shall be published only in the module's description file.	
Information elements		
Information element name	***Type / Range***	***Information element description***
DIO_VENDOR_ID	#define / uint16	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list

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,
   }
};
```

11 Changes to Release 1

11.1 Deleted SWS Items

SWS Item	*Rationale*
DIO080	Deleted. Unable to check this
DIO081	Deleted. Unable to check this

11.2 Replaced SWS Items

No replaced SWS Items

11.3 Changed SWS Items

SWS Item	*Rationale*
DIO065	Deleted DIO_E_PARAM_INVALID_PORT_VALUE and DIO_E_PARAM_INVALID_GROUP_VALUE
DIO104	Changed phrase "MSB" with "bits corresponding to undefined port pins" to resolve ambiguity if not only adjacent pins of a port are undefined

11.4 Added SWS Items

SWS Item	*Rationale*
DIO120	Configuration of detection of development errors
DIO121	API parameter checking if the DIO_DEV_ERROR_DETECT switch is enabled
DIO123	New service Dio_GetVersionInfo() established in Release 2
DIO124	Make this service pre compile time configurable On/Off.