

---

# User Manual

for S32K14X DIO Driver

Document Number: UM2DIOASR4.2 Rev0002R1.0.1  
Rev. 1.0





# Contents

Section number	Title	Page
<b>Chapter 1</b>		
<b>Revision History</b>		
<b>Chapter 2</b>		
<b>Introduction</b>		
2.1	Supported Derivatives.....	9
2.2	Overview.....	9
2.3	About this Manual.....	10
2.4	Acronyms and Definitions.....	10
2.5	Reference List.....	11
<b>Chapter 3</b>		
<b>Driver</b>		
3.1	Requirements.....	13
3.2	Driver Design Summary.....	13
3.3	Hardware Resources.....	13
3.4	Deviation from Requirements.....	14
3.5	Driver limitations.....	15
3.6	Driver usage and configuration tips.....	15
3.7	Runtime Errors.....	18
3.8	Software specification.....	18
3.8.1	Define Reference.....	18
3.8.1.1	Define DIO_E_PARAM_CONFIG.....	18
3.8.1.2	Define DIO_E_PARAM_INVALID_CHANNEL_ID.....	18
3.8.1.3	Define DIO_E_PARAM_INVALID_GROUP_ID.....	19
3.8.1.4	Define DIO_E_PARAM_INVALID_PORT_ID.....	19
3.8.1.5	Define DIO_E_PARAM_POINTER.....	19
3.8.1.6	Define DIO_E_PARAM_LEVEL.....	20
3.8.1.7	Define DIO_GETVERSIONINFO_ID.....	20
3.8.1.8	Define DIO_READCHANNEL_ID.....	21

Section number	Title	Page
3.8.1.9	Define DIO_READCHANNELGROUP_ID.....	21
3.8.1.10	Define DIO_READPORT_ID.....	21
3.8.1.11	Define DIO_WRITECHANNEL_ID.....	22
3.8.1.12	Define DIO_WRITECHANNELGROUP_ID.....	22
3.8.1.13	Define DIO_WRITEPORT_ID.....	22
3.8.1.14	Define DIO_MASKEDWRITEPORT_ID.....	23
3.8.1.15	Define DIO_INSTANCE_ID.....	23
3.8.1.16	Define DIO_FLIPCHANNEL_ID.....	23
3.8.1.17	Define DIO_DEV_ERROR_DETECT.....	24
3.8.1.18	Define DIO_FLIP_CHANNEL_API.....	24
3.8.1.19	Define DIO_MASKEDWRITEPORT_API.....	24
3.8.1.20	Define DIO_READZERO_UNDEFINEDPORTS.....	24
3.8.1.21	Define DIO_REVERSEPORTBITS.....	25
3.8.1.22	Define DIO_VERSION_INFO_API.....	25
3.8.1.23	Define DIO_NUM_PORTS_U16.....	25
3.8.1.24	Define DIO_NUM_CHANNELS_PER_PORT_U16.....	26
3.8.1.25	Define DIO_NUM_CHANNELS_U16.....	26
3.8.1.26	Define DIO_NO_AVAILABLE_CHANNELS.....	26
3.8.1.27	Define DIO_MAX_VALID_OFFSET.....	27
3.8.1.28	Define DIO_INOUT_CONFIG_SUPPORTED.....	27
3.8.1.29	Define DIO_USER_MODE_SOFT_LOCKING.....	28
3.8.1.30	Define DioConf_DioChannel_DioChannel_0.....	28
3.8.1.31	Define DioConf_DioChannelGroup_DioChannelGroup_0.....	29
3.8.1.32	Define DioConf_DioPort_DioPort_0.....	29
3.8.1.33	Define DIO_PRECOMPILE_SUPPORT.....	29
3.8.2	Enum Reference.....	29
3.8.3	Function Reference.....	30
3.8.3.1	Function Dio_ReadChannel.....	30
3.8.3.2	Function Dio_WriteChannel.....	30

Section number	Title	Page
3.8.3.3	Function Dio_FlipChannel.....	31
3.8.3.4	Function Dio_ReadPort.....	32
3.8.3.5	Function Dio_WritePort.....	32
3.8.3.6	Function Dio_ReadChannelGroup.....	32
3.8.3.7	Function Dio_WriteChannelGroup.....	33
3.8.3.8	Function Dio_GetVersionInfo.....	34
3.8.3.9	Function Dio_MaskedWritePort.....	34
3.8.4	Structs Reference.....	35
3.8.4.1	Structure Dio_ChannelGroupType.....	35
3.8.4.2	Structure Dio_ConfigType.....	36
3.8.5	Types Reference.....	37
3.8.5.1	Typedef Dio_PortType.....	37
3.8.5.2	Typedef Dio_ChannelType.....	37
3.8.5.3	Typedef Dio_PortLevelType.....	37
3.8.5.4	Typedef Dio_LevelType.....	37
3.9	Symbolic Names Disclaimer.....	37

## Chapter 4

### Tresos Configuration Plug-in

4.1	Configuration elements of Dio.....	39
4.2	Form IMPLEMENTATION_CONFIG_VARIANT.....	39
4.3	Form DioGeneral.....	40
4.3.1	DioDevErrorDetect (DioGeneral).....	40
4.3.2	DioVersionInfoApi (DioGeneral).....	40
4.3.3	DioReversePortBits (DioGeneral).....	41
4.3.4	DioFlipChannelApi (DioGeneral).....	41
4.3.5	DioReadZeroForUndefinedPortPins (DioGeneral).....	42
4.3.6	DioMaskedWritePortApi (DioGeneral).....	42
4.3.7	DioEnableUserModeSupport (DioGeneral).....	43
4.4	Form CommonPublishedInformation.....	43

Section number	Title	Page
4.4.1	ArReleaseMajorVersion (CommonPublishedInformation).....	44
4.4.2	ArReleaseMinorVersion (CommonPublishedInformation).....	44
4.4.3	ArReleaseRevisionVersion (CommonPublishedInformation).....	45
4.4.4	ModuleId (CommonPublishedInformation).....	45
4.4.5	SwMajorVersion (CommonPublishedInformation).....	46
4.4.6	SwMinorVersion (CommonPublishedInformation).....	46
4.4.7	SwPatchVersion (CommonPublishedInformation).....	47
4.4.8	VendorApiInfix (CommonPublishedInformation).....	47
4.4.9	VendorId (CommonPublishedInformation).....	47
4.5	Form DioConfig.....	48
4.5.1	Form DioPort.....	48
4.5.1.1	DioPortId (DioPort).....	49
4.5.1.2	Form DioChannel.....	49
4.5.1.2.1	DioChannelId (DioChannel).....	50
4.5.1.3	Form DioChannelGroup.....	50
4.5.1.3.1	DioChannelGroupIdentification (DioChannelGroup).....	51
4.5.1.3.2	DioPortMask (DioChannelGroup).....	51
4.5.1.3.3	DioPortOffset (DioChannelGroup).....	52
4.5.1.3.4	DioPortBitNumber (DioChannelGroup).....	52

# Chapter 1

## Revision History

**Table 1-1. Revision History**

Revision	Date	Author	Description
1.0	13/07/2018	NXP MCAL Team	Updated version for ASR 4.2.2S32K14X1.0.1 Release





# Chapter 2

## Introduction

This User Manual describes NXP Semiconductors AUTOSAR Digital Input Output ( Dio ) for S32K14X .

AUTOSAR Dio driver configuration parameters and deviations from the specification are described in Dio Driver chapter of this document. AUTOSAR Dio driver requirements and APIs are described in the AUTOSAR Dio driver software specification document.

### 2.1 Supported Derivatives

The software described in this document is intended to be used with the following microcontroller devices of NXP Semiconductors .

**Table 2-1. S32K14X Derivatives**

NXP Semiconductors	s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64
--------------------	---

All of the above microcontroller devices are collectively named as S32K14X .

### 2.2 Overview

**AUTOSAR (AUTomotive Open System ARchitecture)** is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

AUTOSAR

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.
- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".
- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.

## 2.3 About this Manual

This Technical Reference employs the following typographical conventions:

**Boldface type:** Bold is used for important terms, notes and warnings.

*Italic font:* Italic typeface is used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

### Note

This is a note.

## 2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
DEM	Diagnostic Event Manager
DET	Development Error Tracer
C/CPP	C and C++ Source Code
VLE	Variable Length Encoding
N/A	Not Applicable
MCU	Micro Controller Unit

Table continues on the next page...

**Table 2-2. Acronyms and Definitions (continued)**

Term	Definition
DIO	Digital Input Output

## 2.5 Reference List

**Table 2-3. Reference List**

#	Title	Version
1	Specification of Dio Driver	AUTOSAR Release 4.2.2
2	S32K14X Reference Manual	Reference Manual, Rev. 7, 4/2018
3	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	30/11/2017
4	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	30/11/2017
5	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	30/11/2017
6	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	30/11/2017
7	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	26/02/2018



## Chapter 3 Driver

### 3.1 Requirements

Requirements for this driver are detailed in the AUTOSAR 4.2 Rev0002Dio Driver Software Specification document (See Table [Reference List](#) ).

### 3.2 Driver Design Summary

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.

### 3.3 Hardware Resources

The hardware configured by the Dio driver is GPIO.

The channel to microcontroller pin mapping can be done by using "S32K14X\_IO\_Signal\_Description\_Input\_Multiplexing\_Table.xlsx" from the Reference manual.

Value of actual channel is identified by formula:

**Channel = DioChannelId + DioPortId\*32**

Where:

- DioPortId is the numeric identifier of the DIO port. Symbolic names will be generated for each port pin id for the pins which being used for configuration.

- PortA=0
- PortB=1
- PortC=2
- PortD=3
- PortE=4

- DioChannelId is selected channel in the port what is selected by choosing the value of DioPortId. The maximum channel in 1 port is 32, so the range of DioChannelId is: 0-31

Example: Channel GPIO[35] can be found in the xls file, it is connected to pin PTB3. In order to use GPIO[35] in the Dio driver, the corresponding channel is DioChannelId = 3 and DioPortId = 1 (Port B channel 3).

### 3.4 Deviation from Requirements

The driver deviates from the AUTOSAR Dio Driver software specification in some places. Table identifies the AUTOSAR requirements that are not fully implemented, implemented differently, or out of scope for the Dio driver. Table [Table 3-1](#) provides Status column description.

**Table 3-1. Deviations Status Column Description**

Term	Definition
N/S	Out of scope
N/I	Not implemented

Below table identifies the AUTOSAR requirements that are not fully implemented, implemented differently, or out of scope for the driver.

**Table 3-2. Driver Deviations Table**

Requirement	Status	Description	Notes
SWS_Dio_0008 3	N/I	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.	Rejected and replaced by DIO_SW001: The Dio module's read functions shall always provide the value from the input register, regardless of the configured pin direction. Note: The channels must be configured as input or input-output using the PORT driver.
SWS_Dio_0008 4	N/I	If the microcontroller does not support the direct read-back of a pin value, the Dio	Rejected and replaced by DIO_SW001: The Dio module's read functions shall always

*Table continues on the next page...*

**Table 3-2. Driver Deviations Table (continued)**

Requirement	Status	Description	Notes
		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.	provide the value from the input register, regardless of the configured pin direction. Note: The channels must be configured as input or input-output using the PORT driver.
SWS_Dio_0010 4	N/I	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.	All ports have equal size (the size of Dio_PortType)
SWS_Dio_0010 5	N/I	When writing a port which is smaller than the Dio_PortType using the Dio_WritePort function, the function shall ignore the MSB.	All ports have equal size (the size of Dio_PortType)
SWS_Dio_0016 4	N/I	Dio_ConfigType is the type for all post-build configurable parameters of the DIO driver.	Requirement not applicable. Dio support only pre-compile time configuration.
SWS_Dio_0017 6	N/I	The Dio module shall detect the following type of error: API service called with "NULL pointer" parameter; Error relevance: Development; Related error code: DIO_E_PARAM_CONFIG; Error value (hex): 0x10	Requirement not applicable. DIO_E_PARAM_CONFIG error should have been reported by Dio_Init() function. But Dio_Init function is no longer required in ASR 4.2.1.

## 3.5 Driver limitations

None

## 3.6 Driver usage and configuration tips

The Dio driver APIs work with channels, ports and channel groups.

### Dio channels

A channel is represented by a microcontroller hardware pin. In order to be able to use the Dio channel APIs (Dio\_ReadChannel(), Dio\_WriteChannel() and Dio\_FlipChannel()) for a specific pin, there are a couple steps to be done:

- Open the platform reference manual or the IoMuxing Excel attached to it
- Identify the microcontroller pin you want to use (eg. PE[5])
- Go to DioPort container inside the Dio plugin and add a new port
- Click on the Dio Port Id attribute and observe the content of the Description field
- Take the numeric identifier of the port containing the pin you want to use (eg. 4 corresponding to port E for PE[5]) and set the Dio Port Id to this value
- Go to the DioChannel container inside the DioPort container and add a new channel

- Take the numeric identifier of the pin inside the port for the hardware pin you want to use (eg. 5 for PE[5]) and set the Dio Channel Id attribute to this value
- Generate the code
- Go to Dio\_Cfg.h file and look inside the ‘DEFINES AND MACROS’ section of the file for a define that represents the symbolic name of the Dio Channel (eg. DioConf\_DioChannel\_DioChannel\_0)
- Always use this define as ChannelId parameter when calling Dio APIs related to channels (Dio\_ReadChannel(), Dio\_WriteChannel() and Dio\_FlipChannel())

## **Dio ports**

A port represents several DIO channels that are grouped by hardware (typically controlled by one hardware register). In order to be able to use the Dio port APIs (Dio\_ReadPort(), Dio\_WritePort() and Dio\_MaskedWritePort()) for a specific port, there are a couple steps to be done:

- Open the platform reference manual or the IoMuxing Excel attached to it
- Identify the microcontroller port you want to use (eg. PE)
- Go to DioPort container inside the Dio plugin and add a new port
- Click on the Dio Port Id attribute and observe the content of the Description field
- Take the numeric identifier of the port you want to use (eg. 4 corresponding to port E for PE) and set the Dio Port Id to this value
- Generate the code
- Go to Dio\_Cfg.h file and look inside the ‘DEFINES AND MACROS’ section of the file for a define that represents the symbolic name of the Dio port (eg. DioConf\_DioPort\_DioPort\_0)
- Always use this define as PortId parameter when calling Dio APIs related to ports (Dio\_ReadPort(), Dio\_WritePort() and Dio\_MaskedWritePort())

## **Dio channel groups**

A Dio channel group consists of several adjoining Dio channels that belong to one Dio port. In order to be able to use the Dio channel group APIs (Dio\_ReadChannelGroup(), Dio\_WriteChannelGroup()), there are a couple steps to be done:

- Open the platform reference manual or the IoMuxing Excel attached to it
- Identify the microcontroller pins you want to use (eg. PE[5], PE[6], PE[7])
- Go to DioPort container inside the Dio plugin and add a new port
- Click on the Dio Port Id attribute and observe the content of the Description field
- Take the numeric identifier of the port containing the pin you want to use (eg. 4 corresponding to port E for PE[5], PE[6], PE[7]) and set the Dio Port Id to this value
- Go to the DioChannelGroup container inside the DioPort container and add a new channel group



- Configure the channel group. The information that is needed by the driver is the one in the 'Dio Port Mask' attribute. There is no need to write that information directly, the attributes 'Dio Port Bit Number' and 'Dio Port Offset' are here to help. Just fill them with the number of continuous channels that create the channel group and with the position of the channel group in the port, counted from the least significant bit and hit the 'Calculate value' button on the right side of the 'Dio Port Mask' attribute
- Generate the code
- Go to Dio\_Cfg.h file and look inside the 'DEFINES AND MACROS' section of the file for a define that represents the symbolic name of the Dio Channel Group (eg. DioConf\_DioChannelGroupIdentification\_DioChannelGroup\_0)
- Always use this define as ChannelGroupIdPtr parameter when calling Dio APIs related to channel groups (Dio\_ReadChannelGroup(), Dio\_WriteChannelGroup())

### Non Autosar functionality

- 1.Reverse bits in ports. This option is configurable on/off per entire driver, using the checkbox 'Dio Reverse Port Bits' in DioGeneral container. It affects the functionality of the following APIs working with Dio ports: Dio\_ReadPort(), Dio\_WritePort(), Dio\_ReadChannelGroup() and Dio\_WriteChannelGroup(). If the 'Dio Reverse Port Bits' box is checked, the bits written to ports by the 4 functions above will be reversed. For example, writing 3 to a port with checkbox disabled will set pins 0 and 1 while writing 3 to a port with checkbox enabled will set pins 14 and 15 if the port has 16 bits width or pins 30 and 31 if the port has 32 bits width.
- 2. Read zero for undefined port pins. This option is configurable on/off per entire driver, using the checkbox 'Dio Read Zero For Undefined Port Pins' in DioGeneral container. It affects the functionality of the Dio\_ReadPort() API. It is possible for a given microcontroller port to not have all pins physically implemented. Checking this option will ensure that all not implemented pins in a port read will be read as 0 logic when API Dio\_ReadPort() is called for that port.
- 3. Support to run driver's code from User Mode. This option is configurable on/off per entire driver, using the checkbox 'Enable Dio User Mode Support' in DioGeneral container. When this parameter is enabled, the Dio module will adapt to run from user mode so that the registers under protection can be accessed from user mode. For more information, please see the IM chapter 'User Mode Support'.
- 4. API to write a port using mask. In DioGeneral container there is an attribute called 'Dio Masked Write Port Api'. If the attribute is checked, the Dio driver code will include one extra API for writing the value of a port, called Dio\_MaskedWritePort(). Compared with the Dio\_WritePort() API, this function has one extra parameter called 'Mask', which has the size of the port width. When using this API, only the port channels having the corresponding bits in the 'Mask' set to 1 will be set to the value of the corresponding bits in the 'Level' parameter.

## 3.7 Runtime Errors

This driver doesn't generate any runtime error.

## 3.8 Software specification

The following sections contains driver software specifications.

### 3.8.1 Define Reference

Constants supported by the driver are as per AUTOSAR Dio Driver software specification Version 4.2 Rev0002 .

#### 3.8.1.1 Define DIO\_E\_PARAM\_CONFIG

API service called with "NULL pointer" parameter.

##### Details:

In case of this error, the API service will return immediately without any further action, beside reporting this development error.

**Table 3-3. Define DIO\_E\_PARAM\_CONFIG Description**

<b>Name</b>	DIO_E_PARAM_CONFIG
<b>Initializer</b>	((uint8)0x10)

#### 3.8.1.2 Define DIO\_E\_PARAM\_INVALID\_CHANNEL\_ID

API service called with invalid channel identifier.

##### Details:

In case of this error, the API service will return immediately without any further action, beside reporting this development error.

**Table 3-4. Define DIO\_E\_PARAM\_INVALID\_CHANNEL\_ID Description**

<b>Name</b>	DIO_E_PARAM_INVALID_CHANNEL_ID
<b>Initializer</b>	((uint8)0x0A)

### 3.8.1.3 Define DIO\_E\_PARAM\_INVALID\_GROUP\_ID

API service called with invalid channel group identifier.

#### Details:

In case of this error, the API service will return immediately without any further action, beside reporting this development error.

**Table 3-5. Define DIO\_E\_PARAM\_INVALID\_GROUP\_ID Description**

<b>Name</b>	DIO_E_PARAM_INVALID_GROUP_ID
<b>Initializer</b>	((uint8)0x1F)

### 3.8.1.4 Define DIO\_E\_PARAM\_INVALID\_PORT\_ID

API service called with invalid port identifier.

#### Details:

In case of this error, the API service will return immediately without any further action, beside reporting this development error.

**Table 3-6. Define DIO\_E\_PARAM\_INVALID\_PORT\_ID Description**

<b>Name</b>	DIO_E_PARAM_INVALID_PORT_ID
<b>Initializer</b>	((uint8)0x14)

### 3.8.1.5 Define DIO\_E\_PARAM\_POINTER

API service called with a NULL pointer.

**Details:**

In case of this error, the API service will return immediately without any further action, beside reporting this development error.

**Table 3-7. Define DIO\_E\_PARAM\_POINTER Description**

<b>Name</b>	DIO_E_PARAM_POINTER
<b>Initializer</b>	((uint8)0x20)

### 3.8.1.6 Define DIO\_E\_PARAM\_LEVEL

API service called with invalid channel level value.

**Details:**

In case of this error, the API service will return immediately without any further action, beside reporting this development error.

In detail: If development error detection is enabled, the service Dio\_WriteChannel shall check if the specified channel level is valid (either STD\_HIGH or STD\_LOW). If the “channel level” parameter is invalid, the functions shall report the error code DIO\_E\_PARAM\_LEVEL to the DET.

**Table 3-8. Define DIO\_E\_PARAM\_LEVEL Description**

<b>Name</b>	DIO_E_PARAM_LEVEL
<b>Initializer</b>	((uint8)0x21)

### 3.8.1.7 Define DIO\_GETVERSIONINFO\_ID

API service ID for Dio\_GetVersionInfo() Group function.

**Details:**

Parameter used for DET when raising an error from Dio\_GetVersionInfo() function.

**Table 3-9. Define DIO\_GETVERSIONINFO\_ID Description**

<b>Name</b>	DIO_GETVERSIONINFO_ID
<b>Initializer</b>	((uint8)0x12)

### 3.8.1.8 Define DIO\_READCHANNEL\_ID

API service ID for `Dio_ReadChannel()` function.

#### Details:

Parameter used for DET when raising an error from `Dio_ReadChannel()` function.

**Table 3-10. Define DIO\_READCHANNEL\_ID Description**

<b>Name</b>	DIO_READCHANNEL_ID
<b>Initializer</b>	((uint8)0x00)

### 3.8.1.9 Define DIO\_READCHANNELGROUP\_ID

API service ID for `Dio_ReadChannelGroup()` Group function.

#### Details:

Parameter used for DET when raising an error from `Dio_ReadChannelGroup()` function.

**Table 3-11. Define DIO\_READCHANNELGROUP\_ID Description**

<b>Name</b>	DIO_READCHANNELGROUP_ID
<b>Initializer</b>	((uint8)0x04)

### 3.8.1.10 Define DIO\_READPORT\_ID

API service ID for `Dio_ReadPort()` function.

#### Details:

Parameter used for DET when raising an error from Dio\_ReadPort() function.

**Table 3-12. Define DIO\_READPORT\_ID Description**

<b>Name</b>	DIO_READPORT_ID
<b>Initializer</b>	((uint8)0x02)

### 3.8.1.11 Define DIO\_WRITECHANNEL\_ID

API service ID for Dio\_WriteChannel() function.

#### Details:

Parameter used for DET when raising an error from Dio\_WriteChannel() function.

**Table 3-13. Define DIO\_WRITECHANNEL\_ID Description**

<b>Name</b>	DIO_WRITECHANNEL_ID
<b>Initializer</b>	((uint8)0x01)

### 3.8.1.12 Define DIO\_WRITECHANNELGROUP\_ID

API service ID for Dio\_WriteChannelGroup() Group function.

#### Details:

Parameter used for DET when raising an error from Dio\_WriteChannelGroup() function.

**Table 3-14. Define DIO\_WRITECHANNELGROUP\_ID Description**

<b>Name</b>	DIO_WRITECHANNELGROUP_ID
<b>Initializer</b>	((uint8)0x05)

### 3.8.1.13 Define DIO\_WRITEPORT\_ID

API service ID for Dio\_WritePort() function.

#### Details:

Parameter used for DET when raising an error from Dio\_WritePort() function.

**Table 3-15. Define DIO\_WRITEPORT\_ID Description**

<b>Name</b>	DIO_WRITEPORT_ID
<b>Initializer</b>	((uint8)0x03)

### 3.8.1.14 Define DIO\_MASKEDWRITEPORT\_ID

API service ID for Dio\_MaskedWritePort() function.

#### Details:

Parameter used for DET when raising an error from Dio\_MaskedWritePort() function.

**Table 3-16. Define DIO\_MASKEDWRITEPORT\_ID Description**

<b>Name</b>	DIO_MASKEDWRITEPORT_ID
<b>Initializer</b>	((uint8)0x20)

### 3.8.1.15 Define DIO\_INSTANCE\_ID

API service ID for Det\_ReportError() function.

#### Details:

Parameter used for DET when raising an error from Det\_ReportError() function.

**Table 3-17. Define DIO\_INSTANCE\_ID Description**

<b>Name</b>	DIO_INSTANCE_ID
<b>Initializer</b>	((uint8)0x00)

### 3.8.1.16 Define DIO\_FLIPCHANNEL\_ID

API service ID for Dio\_FlipChannel() function.

**Details:**

Parameter used for DET when raising an error from Dio\_FlipChannel() function.

**Table 3-18. Define DIO\_FLIPCHANNEL\_ID Description**

<b>Name</b>	DIO_FLIPCHANNEL_ID
<b>Initializer</b>	((uint8)0x11)

### 3.8.1.17 Define DIO\_DEV\_ERROR\_DETECT

Enable/Disable Development Error Detection.

**Table 3-19. Define DIO\_DEV\_ERROR\_DETECT Description**

<b>Name</b>	DIO_DEV_ERROR_DETECT
<b>Initializer</b>	(STD_ON)

### 3.8.1.18 Define DIO\_FLIP\_CHANNEL\_API

Function Dio\_FlipChannel() enable switch.

**Table 3-20. Define DIO\_FLIP\_CHANNEL\_API Description**

<b>Name</b>	DIO_FLIP_CHANNEL_API
<b>Initializer</b>	(STD_ON)

### 3.8.1.19 Define DIO\_MASKEDWRITEPORT\_API

Function Dio\_MaskedWritePort() enable switch.

**Table 3-21. Define DIO\_MASKEDWRITEPORT\_API Description**

<b>Name</b>	DIO_MASKEDWRITEPORT_API
<b>Initializer</b>	(STD_ON)



### 3.8.1.20 Define DIO\_READZERO\_UNDEFINEDPORTS

Undefined pins masking enable switch.

Undefined pins masking enable switch. Defines whether the Dio\_ReadPort() function includes the capability to read the undefined port pins as 0.

- True - Enables the Dio\_ReadPort() functionality to read the undefined port pins as 0.
- False - Disables the Dio\_ReadPort() functionality to read the undefined port pins as 0 (Supports the normal functionality with Dio\_ReadPort())

. This functionality is an AutoSAR extension.

**Table 3-22. Define DIO\_READZERO\_UNDEFINEDPORTS Description**

<b>Name</b>	DIO_READZERO_UNDEFINEDPORTS
<b>Initializer</b>	(STD_ON)

### 3.8.1.21 Define DIO\_REVERSEPORTBITS

Reversed port functionality enable switch.

**Table 3-23. Define DIO\_REVERSEPORTBITS Description**

<b>Name</b>	DIO_REVERSEPORTBITS
<b>Initializer</b>	(STD_OFF)

### 3.8.1.22 Define DIO\_VERSION\_INFO\_API

Function Dio\_GetVersionInfo() enable switch.

**Table 3-24. Define DIO\_VERSION\_INFO\_API Description**

<b>Name</b>	DIO_VERSION_INFO_API
<b>Initializer</b>	(STD_ON)

### 3.8.1.23 Define DIO\_NUM\_PORTS\_U16

Number of implemented ports.

**Note**

Used for channel, port and channel group validation.

**Table 3-25. Define DIO\_NUM\_PORTS\_U16 Description**

<b>Name</b>	DIO_NUM_PORTS_U16
<b>Initializer</b>	(uint16)[! "num:inttohex(count(ecu:list('Dio.AvailablePortPinsForWrite')))"!]

**3.8.1.24 Define DIO\_NUM\_CHANNELS\_PER\_PORT\_U16**

Number channels in a port.

**Note**

Used for channel, port and channel group validation.

**Table 3-26. Define DIO\_NUM\_CHANNELS\_PER\_PORT\_U16 Description**

<b>Name</b>	DIO_NUM_CHANNELS_PER_PORT_U16
<b>Initializer</b>	(uint16)(sizeof(Dio_PortLevelType) * 0x8U)

**3.8.1.25 Define DIO\_NUM\_CHANNELS\_U16**

Number of channels available on the implemented ports.

**Note**

Used for channel validation.

**Table 3-27. Define DIO\_NUM\_CHANNELS\_U16 Description**

<b>Name</b>	DIO_NUM_CHANNELS_U16
<b>Initializer</b>	(uint16)(DIO_NUM_PORTS_U16 * DIO_NUM_CHANNELS_PER_PORT_U16)

**3.8.1.26 Define DIO\_NO\_AVAILABLE\_CHANNELS**

Mask representing no available channels on a port.

**Note**

Used for channel validation.

**Table 3-28. Define DIO\_NO\_AVAILABLE\_CHANNELS**  
**Description**

<b>Name</b>	DIO_NO_AVAILABLE_CHANNELS
<b>Initializer</b>	((Dio_PortLevelType)0x0)

### 3.8.1.27 Define DIO\_MAX\_VALID\_OFFSET

Mask representing the maximum valid offset for a channel group.

**Note**

Used for channel group validation.

**Table 3-29. Define DIO\_MAX\_VALID\_OFFSET Description**

<b>Name</b>	DIO_MAX_VALID_OFFSET
<b>Initializer</b>	(uint8)(0x1F)

### 3.8.1.28 Define DIO\_INOUT\_CONFIG\_SUPPORTED

States if the current platform supports configuring of port pins as both input-output:

**STD\_ON:** Current platform supports configuring a port pin as input-output.

**STD\_OFF:** Current platform does not support configuring a port pin as input-output. The port pins can be configured as either input or output.

**Note**

Used by the Dio\_FlipChannel() function. When this define is set to STD\_ON, Dio\_FlipChannel() will toggle the value in the output buffer and will return the one in the input buffer. Port pins for which Dio\_FlipChannel() is called in this case should be configured as both input-output. When this define is set to STD\_OFF, Dio\_FlipChannel() will toggle the value in the

output buffer and will return the one in the output buffer. Port pins for which Dio\_FlipChannel() is called in this case should be configured as output.

**Table 3-30. Define DIO\_INOUT\_CONFIG\_SUPPORTED Description**

<b>Name</b>	DIO_USER_MODE_SOFT_LOCKING
<b>Initializer</b>	(STD_OFF)

### 3.8.1.29 Define DIO\_USER\_MODE\_SOFT\_LOCKING

Enables or disables the access to a hardware register from user mode:

**USER\_MODE\_SOFT\_LOCKING:** All reads to hw registers will be done via REG\_PROT, user mode access

**SUPERVISOR\_MODE\_SOFT\_LOCKING:** Locks the access to the registers only for supervisor mode

**Note:**

Currently, no register protection mechanism is used for Dio driver.

**Note**

Used for channel group validation.

**Table 3-31. Define DIO\_USER\_MODE\_SOFT\_LOCKING Description**

<b>Name</b>	DIO_USER_MODE_SOFT_LOCKING
<b>Initializer</b>	(STD_OFF)

### 3.8.1.30 Define DioConf\_DioChannel\_DioChannel\_0

Symbolic name for the channel DioChannel\_0.

**Table 3-32. Define DioConf\_DioChannel\_DioChannel\_0 Description**

<b>Name</b>	DioConf_DioChannel_DioChannel_0
<b>Initializer</b>	((uint8)0x81U)

### 3.8.1.31 Define DioConf\_DioChannelGroup\_DioChannelGroup\_0

Symbolic name for the channel group DioChannelGroup\_0.

**Table 3-33. Define DioConf\_DioChannelGroup\_DioChannelGroup\_0 Description**

<b>Name</b>	DioConf_DioChannelGroup_DioChannelGroup_0
<b>Initializer</b>	(&DioConfig_0_aChannelGroupList[0])

### 3.8.1.32 Define DioConf\_DioPort\_DioPort\_0

Symbolic name for the port DioPort\_0.

**Table 3-34. Define DioConf\_DioPort\_DioPort\_0 Description**

<b>Name</b>	DioConf_DioPort_DioPort_0
<b>Initializer</b>	((uint8)0x08U)

### 3.8.1.33 Define DIO\_PRECOMPILE\_SUPPORT

Dio driver Pre-Compile configuration switch.

When the switch is enabled, the define DIO\_PRECOMPILE\_SUPPORT is generated in the code and VariantPreCompile is selected. This means that only precompile time configuration parameters are available. The files Dio\_Cfg.h and Dio\_Cfg.c are used.

Dio driver does not support postbuild time configuration so this switch is always enabled.

**Table 3-35. Define DIO\_PRECOMPILE\_SUPPORT Description**

<b>Name</b>	DIO_PRECOMPILE_SUPPORT
<b>Initializer</b>	

## 3.8.2 Enum Reference

Enumeration of all constants supported by the driver are as per AUTOSAR Dio Driver software specification Version 4.2 Rev0002 .

### 3.8.3 Function Reference

Functions of all functions supported by the driver are as per AUTOSAR Dio Driver software specification Version 4.2 Rev0002 .

#### 3.8.3.1 Function Dio\_ReadChannel

Returns the value of the specified DIO channel.

##### Details:

This function returns the value of the specified DIO channel.

**Return:** Returns the level of the corresponding pin as STD\_HIGH or STD\_LOW.

**Pre:** None

**Prototype:** `Dio_LevelType Dio_ReadChannel(const Dio_ChannelType ChannelId);`

**Table 3-36. Dio\_ReadChannel Arguments**

Type	Name	Direction	Description
const Dio_ChannelType	ChannelId	input	Specifies the required channel id.

**Table 3-37. Dio\_ReadChannel Returns**

Value	Description
STD_HIGH	The logical level of the corresponding 'pin' is 1.
STD_LOW	The logical level of the corresponding 'pin' is 0.

#### 3.8.3.2 Function Dio\_WriteChannel

Sets the level of a channel.

##### Details:

If the specified channel is configured as an output channel, this function will set the specified level on the specified channel. If the specified channel is configured as an input channel, this function will have no influence on the physical output and on the result of

**Pre:** None

**Prototype:** `void Dio_WriteChannel(const Dio_ChannelType ChannelId, const Dio_LevelType Level);`

**Table 3-38. Dio\_WriteChannel Arguments**

Type	Name	Direction	Description
const Dio_ChannelType	ChannelId	input	Specifies the required channel id.
const Dio_LevelType	Level	input	Specifies the channel desired level.

### 3.8.3.3 Function Dio\_FlipChannel

Inverts the level of a channel.

**Details:**

If the specified channel is configured as an output channel, this function will invert the level of the specified channel. If the specified channel is configured as an input channel, this function will have no influence on the physical output and on the result of the next read service.

**Return:** Returns the level of the corresponding pin as STD\_HIGH or STD\_LOW.

**Pre:** This function can be used only if DIO\_FLIP\_CHANNEL\_API has been enabled.

**Prototype:** `Dio_LevelType Dio_FlipChannel(const Dio_ChannelType ChannelId);`

**Table 3-39. Dio\_FlipChannel Arguments**

Type	Name	Direction	Description
const Dio_ChannelType	ChannelId	input	Specifies the required channel id.

**Table 3-40. Dio\_FlipChannel Returns**

Value	Description
STD_HIGH	The logical level of the corresponding 'pin' is 1.
STD_LOW	The logical level of the corresponding 'pin' is 0.

### 3.8.3.4 Function Dio\_ReadPort

Returns the level of all channels of specified port.

**Details:**

This function will return the level of all channels belonging to the specified port.

**Return:** Levels of all channels of specified port.

**Pre:** None

**Prototype:** `Dio_PortLevelType Dio_ReadPort (const Dio_PortType PortId);`

**Table 3-41. Dio\_ReadPort Arguments**

Type	Name	Direction	Description
const Dio_PortType	PortId	input	Specifies the required port id.

### 3.8.3.5 Function Dio\_WritePort

Sets the value of a port.

**Details:**

This function will set the specified value on the specified port.

**Pre:** None

**Prototype:** `void Dio_WritePort (const Dio_PortType PortId, const Dio_PortLevelType Level);`

**Table 3-42. Dio\_WritePort Arguments**

Type	Name	Direction	Description
const Dio_PortType	PortId	input	Specifies the required port id.
const Dio_PortLevelType	Level	input	Specifies the required levels for the port pins.



### 3.8.3.6 Function Dio\_ReadChannelGroup

This service reads a subset of the adjoining bits of a port.

#### Details:

This function will read a subset of adjoining bits of a port (channel group).

**Return:** The channel group levels.

**Pre:** None

**Prototype:** `Dio_PortLevelType Dio_ReadChannelGroup(const Dio_ChannelGroupType *ChannelGroupIdPtr);`

**Table 3-43. Dio\_ReadChannelGroup Arguments**

Type	Name	Direction	Description
const Dio_ChannelGroupType *	ChannelGroupIdPtr	input	Pointer to the channel group.

### 3.8.3.7 Function Dio\_WriteChannelGroup

Sets a subset of the adjoining bits of a port to the specified levels.

#### Details:

This function will set a subset of adjoining bits of a port (channel group) to the specified levels without changing the remaining channels of the port and channels that are configured as input. This function will do the masking of the channels and will do the shifting so that the values written by the function are aligned to the LSB.

**Pre:** None

**Prototype:** `void Dio_WriteChannelGroup(const Dio_ChannelGroupType *ChannelGroupIdPtr, const Dio_PortLevelType Level);`

**Table 3-44. Dio\_WriteChannelGroup Arguments**

Type	Name	Direction	Description
const Dio_ChannelGroupType *	ChannelGroupIdPtr	input	Pointer to the channel group.

*Table continues on the next page...*

**Table 3-44. Dio\_WriteChannelGroup Arguments (continued)**

Type	Name	Direction	Description
const Dio_PortLevelType	Level	input	Desired levels for the channel group.

### 3.8.3.8 Function Dio\_GetVersionInfo

Service to get the version information of this module.

#### Details:

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

- Module Id.
- Vendor Id.
- Vendor specific version numbers.

**Pre:** This function can be used only if `DIO_VERSION_INFO_API` has been enabled.

**Prototype:** `void Dio_GetVersionInfo(Std_VersionInfoType *versioninfo);`

**Table 3-45. Dio\_GetVersionInfo Arguments**

Type	Name	Direction	Description
Std_VersionInfoType pe *	versioninfo	input, output	Pointer to where to store the version information of this module.

### 3.8.3.9 Function Dio\_MaskedWritePort

DIO Mask write port using mask.

#### Details:

DIO write port using mask.

**Pre:** This function can be used only if `DIO_MASKEDWRITEPORT_API` has been enabled.

**Prototype:** void Dio\_MaskedWritePort(const Dio\_PortType PortId, const Dio\_PortLevelType Level, const Dio\_PortLevelType Mask);

**Table 3-46. Dio\_MaskedWritePort Arguments**

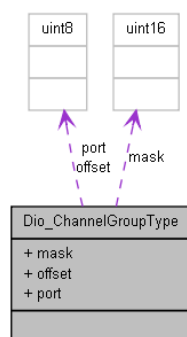
Type	Name	Direction	Description
const Dio_PortType	PortId	input	Specifies the required port id.
const Dio_PortLevelType	Level	input	Specifies the required levels for the port pins.
const Dio_PortLevelType	Mask	input	Specifies the Mask value of the port.

## 3.8.4 Structs Reference

Data structures supported by the driver are as per AUTOSAR Dio Driver software specification Version 4.2 Rev0002 .

### 3.8.4.1 Structure Dio\_ChannelGroupType

Type of a DIO channel group representation.



**Figure 3-1. Struct Dio\_ChannelGroupType**

#### Declaration:

```
typedef struct
{
    VAR(Dio_PortType, AUTOMATIC)    port;
    VAR(uint8, AUTOMATIC)           offset;
    VAR(Dio_PortLevelType, AUTOMATIC) mask;
} Dio_ChannelGroupType;
```

**Table 3-47. Structure Dio\_ChannelGroupType member description**

Member	Description
port	Port identifier.

*Table continues on the next page...*

Table 3-47. Structure Dio\_ChannelGroupType member description (continued)

Member	Description
offset	Bit offset within the port.
mask	Group mask.

3.8.4.2 Structure Dio\_ConfigType

Type of a DIO configuration structure.

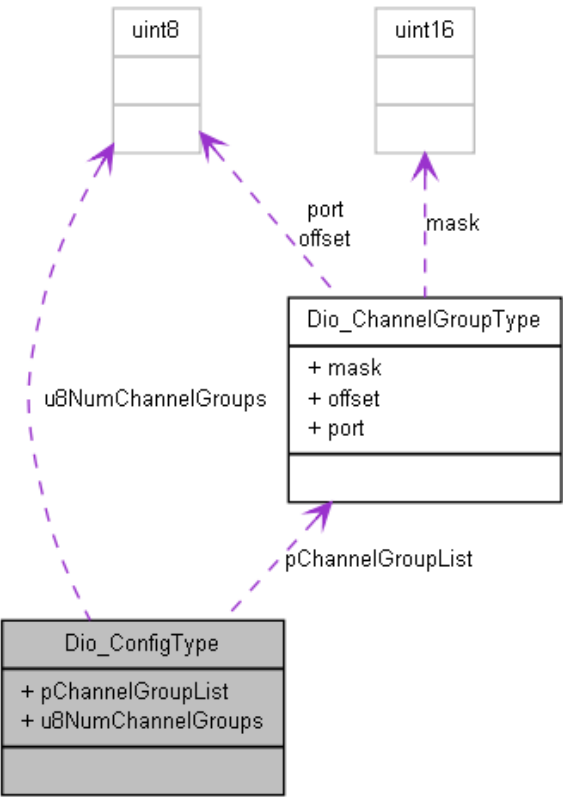


Figure 3-2. Struct Dio\_ConfigType

Note

In this implementation there is no need for a configuration structure there is only a dummy field, it is recommended to initialize this field to zero.

Declaration:

```
typedef struct
{
    VAR(uint8, AUTOMATIC)                                u8NumChannelGroups;
```

```

    P2CONST(Dio_ChannelGroupType, AUTOMATIC, DIO_APPL_DATA) pChannelGroupList;
} Dio_ConfigType;

```

**Table 3-48. Structure Dio\_ConfigType member description**

Member	Description
u8NumChannelGroups	Number of channel groups in configuration.
pChannelGroupList	Pointer to list of channel groups in configuration.

## 3.8.5 Types Reference

Types supported by the driver are as per AUTOSAR Dio Driver software specification Version 4.2 Rev0002 .

### 3.8.5.1 Typedef Dio\_PortType

Type of a DIO port representation.

**Type:** `uint8`

### 3.8.5.2 Typedef Dio\_ChannelType

Type of a DIO channel representation.

**Type:** `uint16`

### 3.8.5.3 Typedef Dio\_PortLevelType

Type of a DIO port levels representation.

**Type:** `uint16`

### 3.8.5.4 Typedef Dio\_LevelType

Type of a DIO channel levels representation.

**Type:** `uint8`

## 3.9 Symbolic Names Disclaimer

All containers having the symbolic name tag set as true in the Autosar schema will generate defines like:

```
#define <Container_Short_Name> <Container_ID>
```

For this reason it is forbidden to duplicate the name of such containers across the MCAL configuration, or to use names that may trigger other compile issues (e.g. match existing #ifdefs arguments).

## Chapter 4

# Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the Dio Driver. The most of the parameters are described below.

### 4.1 Configuration elements of Dio

Included forms :

- IMPLEMENTATION\_CONFIG\_VARIANT
- DioGeneral
- CommonPublishedInformation
- DioConfig

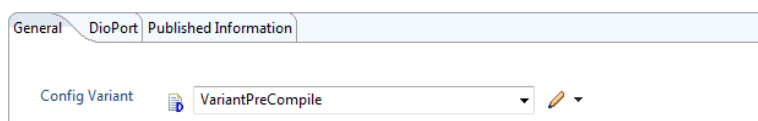
**Table 4-1. Revision table**

Revision	Date
4.1.0	2010-12-03

### 4.2 Form IMPLEMENTATION\_CONFIG\_VARIANT

VariantPreCompile: Only precompile time configuration parameters.

The files Dio\_Cfg.h and Dio\_Cfg.c are used.



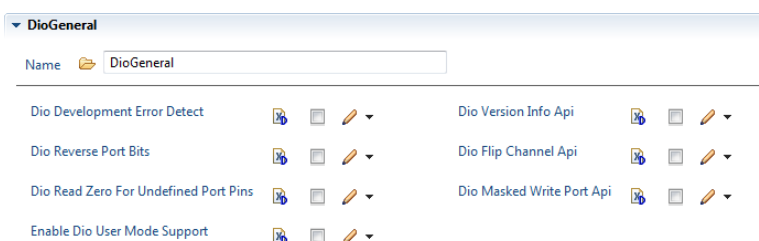
**Figure 4-1. Tresos Plugin snapshot for IMPLEMENTATION\_CONFIG\_VARIANT form.**

**Table 4-2. Attribute IMPLEMENTATION\_CONFIG\_VARIANT detailed description**

Property	Value
Label	Config Variant
Type	ENUMERATION
Default	VariantPreCompile
Range	VariantPreCompile

## 4.3 Form DioGeneral

General DIO module configuration parameters.

**Figure 4-2. Tresos Plugin snapshot for DioGeneral form.**

### 4.3.1 DioDevErrorDetect (DioGeneral)

Switches the Development Error Detection and Notification ON or OFF.

True: Enabled.

False: Disabled.

**Table 4-3. Attribute DioDevErrorDetect (DioGeneral) detailed description**

Property	Value
Label	Dio Development Error Detect
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false



### 4.3.2 DioVersionInfoApi (DioGeneral)

Adds / removes the service Dio\_GetVersionInfo() from the code.

True - Dio\_GetVersionInfo() API is enabled.

False - Dio\_GetVersionInfo() API is disabled (it cannot be used).

**Table 4-4. Attribute DioVersionInfoApi (DioGeneral) detailed description**

Property	Value
Label	Dio Version Info Api
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

### 4.3.3 DioReversePortBits (DioGeneral)

If this box is checked, the bits written to defined ports will be reversed, meaning that writing 3 to a port with checkbox disabled will set pins 0 and 1 of the port while writing 3 to a port with checkbox enabled will set pins 14 and 15 of the port.

This functionality is an AutoSAR extension.

**Table 4-5. Attribute DioReversePortBits (DioGeneral) detailed description**

Property	Value
Label	Dio Reverse Port Bits
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

### 4.3.4 DioFlipChannelApi (DioGeneral)

Adds / removes the service Dio\_FlipChannel() from the code.

True - Dio\_FlipChannel() API is enabled.

False - Dio\_FlipChannel() API is disabled (it cannot be used).

**Table 4-6. Attribute DioFlipChannelApi (DioGeneral) detailed description**

Property	Value
Label	Dio Flip Channel Api
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

### 4.3.5 DioReadZeroForUndefinedPortPins (DioGeneral)

Defines whether the Dio\_ReadPort() function includes the capability to read the undefined port pins as 0.

True - Enables the Dio\_ReadPort() functionality to read the undefined port pins as 0.

False - Disables the Dio\_ReadPort() functionality to read the undefined port pins as 0 (Supports the normal functionality with Dio\_ReadPort()).

This functionality is an AutoSAR extension.

**Table 4-7. Attribute DioReadZeroForUndefinedPortPins (DioGeneral) detailed description**

Property	Value
Label	Dio Read Zero For Undefined Port Pins
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	true

### 4.3.6 DioMaskedWritePortApi (DioGeneral)

Defines whether the driver function Dio\_MaskedWritePort() will be included at compile time or excluded.

This API is an AutoSAR extension.

True - Dio\_MaskedWritePort() API enabled.

False - Dio\_MaskedWritePort() API disabled.

**Table 4-8. Attribute DioMaskedWritePortApi (DioGeneral) detailed description**

Property	Value
Label	Dio Masked Write Port Api
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

### 4.3.7 DioEnableUserModeSupport (DioGeneral)

When this parameter is enabled, the Dio module will adapt to run from User Mode, configuring REG\_PROT for SIUL2 IP so that the registers under protection can be accessed from user mode by setting UAA bit in REG\_PROT\_GCR to 1.

For more information, please see chapter 'User Mode Support' in IM

Note: Implementation Specific Parameter

**Table 4-9. Attribute DioEnableUserModeSupport (DioGeneral) detailed description**

Property	Value
Label	Enable Dio User Mode Support
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

## 4.4 Form CommonPublishedInformation

Common container, aggregated by all modules. It contains published information about vendor and versions.

Dio

Name

Dio

General

DioPort

Published Information

CommonPublishedInformation

Name

CommonPublishedInformation

AUTOSAR Major Version

4

AUTOSAR Minor Version

2

AUTOSAR Release Revision Version

2

Module Id

120

Software Major Version

1

Software Minor Version

0

Software Patch Version

1

Vendor Api Infix

Vendor Id

43

Figure 4-3. Tresos Plugin snapshot for CommonPublishedInformation form.

4.4.1 ArReleaseMajorVersion (CommonPublishedInformation)

Major version number of AUTOSAR specification on which the appropriate implementation is based on.

Table 4-10. Attribute ArReleaseMajorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	AUTOSAR Major Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	4
Invalid	Range >=4 <=4

4.4.2 ArReleaseMinorVersion (CommonPublishedInformation)

Minor version number of AUTOSAR specification on which the appropriate implementation is based on.

**Table 4-11. Attribute ArReleaseMinorVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	AUTOSAR Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	2
Invalid	Range >=2 <=2

### 4.4.3 ArReleaseRevisionVersion (CommonPublishedInformation)

Revision version number of AUTOSAR specification on which the appropriate implementation is based on.

**Table 4-12. Attribute ArReleaseRevisionVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	AUTOSAR Release Revision Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	2
Invalid	Range >=2 <=2

### 4.4.4 ModuleId (CommonPublishedInformation)

Module ID of this module from Module List.

**Table 4-13. Attribute ModuleId (CommonPublishedInformation) detailed description**

Property	Value
Label	Module Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false

*Table continues on the next page...*

**Table 4-13. Attribute ModuleId (CommonPublishedInformation) detailed description (continued)**

Property	Value
Default	120
Invalid	Range >=120 <=120

#### 4.4.5 SwMajorVersion (CommonPublishedInformation)

Major version number of the vendor specific implementation of the module. The numbering is vendor specific.

**Table 4-14. Attribute SwMajorVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	Software Major Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	1
Invalid	Range >=1 <=1

#### 4.4.6 SwMinorVersion (CommonPublishedInformation)

Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

**Table 4-15. Attribute SwMinorVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	Software Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range >=0 <=0

#### 4.4.7 SwPatchVersion (CommonPublishedInformation)

Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

**Table 4-16. Attribute SwPatchVersion (CommonPublishedInformation) detailed description**

Property	Value
Label	Software Patch Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	1
Invalid	Range >=1 <=1

#### 4.4.8 VendorApiInfix (CommonPublishedInformation)

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name. This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows:

<ModuleName>\_>VendorId>\_<VendorApiInfix><Api name from SWS>. E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name Can\_Write defined in the SWS will translate to Can\_123\_v11r456Write. This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

**Table 4-17. Attribute VendorApiInfix (CommonPublishedInformation) detailed description**

Property	Value
Label	Vendor Api Infix
Type	STRING_LABEL
Origin	Custom
Symbolic Name	false
Default	
Enable	false

## 4.4.9 VendorId (CommonPublishedInformation)

Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.

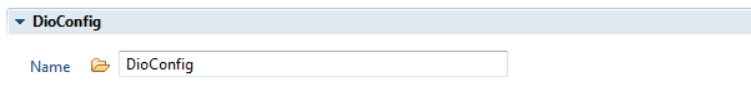
**Table 4-18. Attribute VendorId (CommonPublishedInformation) detailed description**

Property	Value
Label	Vendor Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	43
Invalid	Range <div> <div>&gt;=43</div> <div>&lt;=43</div> </div>

## 4.5 Form DioConfig

Included forms :

- [Form DioPort](#)



**Figure 4-4. Tresos Plugin snapshot for DioConfig form.**

### 4.5.1 Form DioPort

Configuration of individual DIO ports, consisting of channels and possible channel groups. The single DIO channel levels inside a DIO port represent a bit in the DIO port value. A channel group is a formal logical combination of several adjoining DIO channels within a DIO port. The configuration process for Dio module shall provide symbolic names for each configured DIO channel, port and group.

Is included by form : [Form DioConfig](#)

Included forms :

- [Form DioChannel](#)
- [Form DioChannelGroup](#)



**Figure 4-5. Tresos Plugin snapshot for DioPort form.**

#### 4.5.1.1 DioPortId (DioPort)

Numeric identifier of the DIO port. Symbolic names will be generated for each port pin id for the pins which being used for configuration. NOTE: Use the following values to configure different ports.

- PortA=0
- PortB=1
- PortC=2
- PortD=3
- PortE=4

**Table 4-19. Attribute DioPortId (DioPort) detailed description**

Property	Value
Label	Dio Port Id
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	true
Invalid	Range <div style="margin-left: 20px;"> <math>\leq 4</math>  <math>\geq 0</math> </div>

#### 4.5.1.2 Form DioChannel

Configuration of an individual DIO channel. Symbolic names will be generated for each channel. A general purpose digital IO pin represents a DIO channel which will be having value either STD\_HIGH or STD\_LOW.

**Is included by form :** [Form DioPort](#)




Figure 4-6. Tresos Plugin snapshot for DioChannel form.

4.5.1.2.1 DioChannelId (DioChannel)

Channel Id of the DIO channel. This value will be assigned to the symbolic names.

Table 4-20. Attribute DioChannelId (DioChannel) detailed description

Property	Value
Label	Dio Channel Id
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	true
Invalid	Range <=31 >=0

4.5.1.3 Form DioChannelGroup

A channel group represents several adjoining DIO channels represented by a logical group. This container definition does not explicitly define a symbolic name parameter, but symbolic names will be generated for each channel group. Each group provides a structure with parameters port, offset, Bit\_NO and mask.

Is included by form : [Form DioPort](#)

**Figure 4-7. Tresos Plugin snapshot for DioChannelGroup form.**

#### 4.5.1.3.1 DioChannelGroupIdentification (DioChannelGroup)

A DIO channel group is identified in DIO APIs by a pointer to a data structure of type `Dio_ChannelGroupType`. This 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, a string value should be given for this parameter. Symbolic names will be generated for each `DioChannelGroup`, which will be assigned with address of this string in order to point to the structure parameters. Example: `OutputGroup`

**Table 4-21. Attribute DioChannelGroupIdentification (DioChannelGroup) detailed description**

Property	Value
Label	Dio Channel Group Identification
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	true

#### 4.5.1.3.2 DioPortMask (DioChannelGroup)

This shall be the mask which defines the positions of the channel group. The data type depends on the port width.

**Table 4-22. Attribute DioPortMask (DioChannelGroup) detailed description**

Property	Value
Label	Dio Port Mask
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false

*Table continues on the next page...*

**Table 4-22. Attribute DioPortMask (DioChannelGroup) detailed description (continued)**

Property	Value
Default	1
Invalid	Range >=0 <=4294967295

#### 4.5.1.3.3 DioPortOffset (DioChannelGroup)

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

**Table 4-23. Attribute DioPortOffset (DioChannelGroup) detailed description**

Property	Value
Label	Dio Port Offset
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Invalid	Range >=0 <=31

#### 4.5.1.3.4 DioPortBitNumber (DioChannelGroup)

This is the number of continuous channels that create a channel group

**Table 4-24. Attribute DioPortBitNumber (DioChannelGroup) detailed description**

Property	Value
Label	DioPortBitNumber
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	1
Invalid	Range >=0 <=32

**How to Reach Us:****Home Page:**[nxp.com](http://nxp.com)**Web Support:**[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. Arm, AMBA, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and  $\mu$ Vision are registered trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. Arm7, Arm9, Arm11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, Mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2018 NXP B.V.

Document Number UM2DIOASR4.2 Rev0002R1.0.1  
Revision 1.0