User Manual

for S32K1XX DIO Driver

Rev. 1 — 11 January 2022   **User manual**
UM2DIOASR4.2 Rev0002R1.0.5

Revision History

Revision	Date	Author	Description
1.0	11/01/2022	NXP MCAL Team	Updated version for ASR 4.2.2 S32K1XX R1.0.5

1   Introduction

This User Manual describes NXP Semiconductors AUTOSAR Digital Input Output (Dio) for S32K1XX .

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.

1.1   Supported Derivatives

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

Table 1.   S32K1XX Derivatives

NXP Semiconductors	s32k142_lqfp100, s32k142_lqfp64, s32k142_lqfp48, s32k144_lqfp100, s32k144_mapbga100, s32k144_lqfp64, s32k144_lqfp48, s32k146_lqfp144, s32k146_lqfp100, s32k146_mapbga100, s32k146_lqfp64, s32k148_lqfp176, s32k148_lqfp144, s32k148_lqfp100, s32k148_mapbga100, s32k118_lqfp48, s32k118_lqfp64, s32k116_lqfp48, s32k116_qfn32, s32k144w_lqfp48, s32k144w_lqfp64, s32k142w_lqfp64, s32k142w_lqfp48

All of the above microcontroller devices are collectively named as S32K1XX .

1.2   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.

1.3   About this Manual

This Technical Reference employs the following typographical conventions:

AUTOSAR_MCAL_DIO_UM

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

User manual

Rev. 1 — 11 January 2022

UM2DIOASR4.2 Rev0002R1.0.5

3 / 35

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.*

1.4 Acronyms and Definitions

Table 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
DIO	Digital Input Output

1.5 Reference List

Table 3. Reference List

#	Title	Version
1	Specification of Dio Driver	AUTOSAR Release 4.2.2
2	S32K1xx Series Reference Manual	Rev. 13, 04/2020
3	S32K1xx Data Sheet	Rev. 13, 04/2020
4	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	Rev. 20/APR/2020
5	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	Rev. 20/APR/2020
6	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	Rev. 20/APR/2020
7	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	Rev. 20/APR/2020
8	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	Rev. 20/APR/2020
9	S32K116 Mask Set Errata for Mask 0N96V (0N96V)	Rev. 20/APR/2020
10	S32K144W Mask Set Errata for Mask 0P64A (0P64A)	Rev. 14 FEB 2020

2 Driver

AUTOSAR_MCAL_DIO_UM

User manual

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Rev. 1 — 11 January 2022

UM2DIOASR4.2 Rev0002R1.0.5

4 / 35

2.1 Requirements

Requirements for this driver are detailed in the AUTOSAR 4.2 Rev0002 Dio Driver Software Specification document (See Table <u>Section 1.5</u>).

2.2 Driver Design Sumary

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.

2.3 Hardware Resources

The hardware configured by the Dio driver is GPIO.

The channel to microcontroller pin mapping can be done by using "S32K1XX_IO_Signal_Description_Input_Multiplexing.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).

2.4 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 4 provides Status column description.

Table 4. Deviations Status Column Description

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

AUTOSAR_MCAL_DIO_UM

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

User manual

Rev. 1 — 11 January 2022

UM2DIOASR4.2 Rev0002R1.0.5

5 / 35

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

Table 5. Driver Deviations Table

Requirement	Status	Description	Notes
SWS_Dio_00083	N/S	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.	This works only if the output channel is configured as input, too. This requirement is rejected and replaced by DIO_SW001.dio
SWS_Dio_00084	N/S	If the microcontroller does not support the direct read-back of a pin value, the Dio module's read functions shall provide the value of the output register, when they are used on a channel which is configured as an output channel.	The read functions will only read the input registers, regardless of the channel configuration. This requirement is rejected and replaced by DIO_SW001.dio
SWS_Dio_00104	N/S	When reading a port which is smaller than the Dio_PortType using the Dio_ReadPort function (see [SWS_Dio_00103]), the function shall set the bits corresponding to undefined port pins to 0.Furthermore, the requirements SWS_Dio_00005, SWS_Dio_00118 and SWS_Dio_00026 are applicable to the Dio_ReadPort function.	Requirement not applicable. Dio_ReadPort function always reads a port of exactly the size defined by Dio_PortType. It cannot read a port smaller than the size of the Dio_PortType.
SWS_Dio_00105	N/S	When writing a port which is smaller than the Dio_PortType using the Dio_WritePort function (see [SWS_Dio_00103]), the function shall ignore the MSB.	Requirement not applicable. Dio_WritePort function always writes a port of exactly the size defined by Dio_PortType. It cannot write a port smaller than the size of the Dio_PortType.
SWS_Dio_00164	N/S	Dio_ConfigType is the type for all configurable parameters of the DIO driver.	Requirement not applicable. Dio support only pre-compile time configuration.
SWS_Dio_00176	N/S	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 4.2.2

2.5 Driver limitations

None

2.6 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())

Autosar extension 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.

2.7 Runtime Errors

This driver doesn't generate any runtime error.

2.8 Software specification

The following sections contains driver software specifications.

AUTOSAR_MCAL_DIO_UM

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

User manual

Rev. 1 — 11 January 2022

UM2DIOASR4.2 Rev0002R1.0.5

8 / 35

2.8.1 Define Reference

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

2.8.1.1 Define DIO_E_PARAM_CONFIG

API service called with "NULL pointer" parameter.

<u>**Details**</u> :

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

Table 6. Define DIO_E_PARAM_CONFIG Description

Name	DIO_E_PARAM_CONFIG
Initializer	((uint8)0x10)

2.8.1.2 Define DIO_E_PARAM_INVALID_CHANNEL_ID

API service called with invalid channel identifier.

<u>**Details**</u> :

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

Table 7. Define DIO_E_PARAM_INVALID_CHANNEL_ID Description

Name	DIO_E_PARAM_INVALID_CHANNEL_ID
Initializer	((uint8)0x0A)

2.8.1.3 Define DIO_E_PARAM_INVALID_GROUP_ID

API service called with invalid channel group identifier.

<u>**Details**</u> :

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

Table 8. Define DIO_E_PARAM_INVALID_GROUP_ID Description

Name	DIO_E_PARAM_INVALID_GROUP_ID
Initializer	((uint8)0x1F)

2.8.1.4 Define DIO_E_PARAM_INVALID_PORT_ID

API service called with invalid port identifier.

<u>**Details**</u> :

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

Table 9. Define DIO_E_PARAM_INVALID_PORT_ID Description

Name	DIO_E_PARAM_INVALID_PORT_ID
Initializer	((uint8)0x14)

2.8.1.5 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 10. Define DIO_E_PARAM_POINTER Description

Name	DIO_E_PARAM_POINTER
Initializer	((uint8)0x20)

2.8.1.6 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 11. Define DIO_E_PARAM_LEVEL Description

Name	DIO_E_PARAM_LEVEL
Initializer	((uint8)0x21)

2.8.1.7 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 12. Define DIO_GETVERSIONINFO_ID Description

Name	DIO_GETVERSIONINFO_ID
Initializer	((uint8)0x12)

2.8.1.8 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 13. Define DIO_READCHANNEL_ID Description

Name	DIO_READCHANNEL_ID
Initializer	((uint8)0x00)

2.8.1.9 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 14. Define DIO_READCHANNELGROUP_ID Description

Name	DIO_READCHANNELGROUP_ID
Initializer	((uint8)0x04)

2.8.1.10 Define DIO_READPORT_ID

API service ID for `Dio_ReadPort()` function.

<u>**Details**</u> :

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

Table 15. Define DIO_READPORT_ID Description

Name	DIO_READPORT_ID
Initializer	((uint8)0x02)

2.8.1.11 Define DIO_WRITECHANNEL_ID

API service ID for `Dio_WriteChannel()` function.

<u>**Details**</u> :

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

Table 16. Define DIO_WRITECHANNEL_ID Description

Name	DIO_WRITECHANNEL_ID
Initializer	((uint8)0x01)

2.8.1.12 Define DIO_WRITECHANNELGROUP_ID

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

<u>**Details**</u> :

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

Table 17. Define DIO_WRITECHANNELGROUP_ID Description

Name	DIO_WRITECHANNELGROUP_ID
Initializer	((uint8)0x05)

2.8.1.13 Define DIO_WRITEPORT_ID

API service ID for `Dio_WritePort()` function.

<u>**Details**</u> :

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

Table 18. Define DIO_WRITEPORT_ID Description

Name	DIO_WRITEPORT_ID
Initializer	((uint8)0x03)

2.8.1.14 Define DIO_MASKEDWRITEPORT_ID

API service ID for `Dio_MaskedWritePort()` function.

<u>**Details**</u> :

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

Table 19. Define DIO_MASKEDWRITEPORT_ID Description

Name	DIO_MASKEDWRITEPORT_ID
Initializer	((uint8)0x20)

2.8.1.15 Define DIO_INSTANCE_ID

API service ID for `Det_ReportError()` function.

<u>**Details**</u> :

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

Table 20. Define DIO_INSTANCE_ID Description

Name	DIO_INSTANCE_ID
Initializer	((uint8)0x00)

2.8.1.16 Define DIO_FLIPCHANNEL_ID

API service ID for `Dio_FlipChannel()` function.

<u>**Details**</u> :

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

Table 21. Define DIO_FLIPCHANNEL_ID Description

Name	DIO_FLIPCHANNEL_ID
Initializer	((uint8)0x11)

2.8.1.17 Define DIO_DEV_ERROR_DETECT

Enable/Disable Development Error Detection.

Table 22. Define DIO_DEV_ERROR_DETECT Description

Name	DIO_DEV_ERROR_DETECT
Initializer	(STD_ON)

2.8.1.18 Define DIO_FLIP_CHANNEL_API

Function `Dio_FlipChannel()` enable switch.

Table 23. Define DIO_FLIP_CHANNEL_API Description

Name	DIO_FLIP_CHANNEL_API
Initializer	(STD_ON)

2.8.1.19 Define DIO_MASKEDWRITEPORT_API

Function `Dio_MaskedWritePort()` enable switch.

Table 24. Define DIO_MASKEDWRITEPORT_API Description

Name	DIO_MASKEDWRITEPORT_API
Initializer	(STD_ON)

2.8.1.20 Define DIO_READZERO_UNDEFINEDPORTS

Undefined pins masking enable switch.

AUTOSAR_MCAL_DIO_UM

© NXP B.V. 2022. All rights reserved.

User manual

Rev. 1 — 11 January 2022

UM2DIOASR4.2 Rev0002R1.0.5

12 / 35

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 25. Define DIO_READZERO_UNDEFINEDPORTS Description

Name	DIO_READZERO_UNDEFINEDPORTS
Initializer	(STD_ON)

2.8.1.21 Define DIO_REVERSEPORTBITS

Reversed port functionality enable switch.

Table 26. Define DIO_REVERSEPORTBITS Description

Name	DIO_REVERSEPORTBITS
Initializer	(STD_OFF)

2.8.1.22 Define DIO_VERSION_INFO_API

Function `Dio_GetVersionInfo()` enable switch.

Table 27. Define DIO_VERSION_INFO_API Description

Name	DIO_VERSION_INFO_API
Initializer	(STD_ON)

2.8.1.23 Define DIO_NUM_PORTS_U16

Number of implemented ports.

Note: Used for channel, port and channel group validation.

Table 28. Define DIO_NUM_PORTS_U16 Description

Name	DIO_NUM_PORTS_U16
Initializer	(`uint16`)[!"num:inttohex(count(ecu:list('Dio.AvailablePortPinsFor Write')))"!]

2.8.1.24 Define DIO_NUM_CHANNELS_PER_PORT_U16

Number channels in a port.

Note: Used for channel, port and channel group validation.

Table 29. Define DIO_NUM_CHANNELS_PER_PORT_U16 Description

Name	DIO_NUM_CHANNELS_PER_PORT_U16
Initializer	(`uint16`)(sizeof(`Dio_PortLevelType`) * 0x8U)

2.8.1.25 Define DIO_NUM_CHANNELS_U16

Number of channels available on the implemented ports.

Note: Used for channel validation.

Table 30. Define DIO_NUM_CHANNELS_U16 Description

Name	DIO_NUM_CHANNELS_U16
Initializer	`(uint16)`(DIO_NUM_PORTS_U16 * DIO_NUM_CHANNELS_PER_PORT_U16)

2.8.1.26 Define DIO_NO_AVAILABLE_CHANNELS

Mask representing no available channels on a port.

Note: *Used for channel validation.*

Table 31. Define DIO_NO_AVAILABLE_CHANNELS Description

Name	DIO_NO_AVAILABLE_CHANNELS
Initializer	`((Dio_PortLevelType)`0x0)

2.8.1.27 Define DIO_MAX_VALID_OFFSET

Mask representing the maximum valid offset for a channel group.

Note: *Used for channel group validation.*

Table 32. Define DIO_MAX_VALID_OFFSET Description

Name	DIO_MAX_VALID_OFFSET
Initializer	`(uint8)`(0x1F)

2.8.1.28 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 33. Define DIO_INOUT_CONFIG_SUPPORTED Description

Name	DIO_USER_MODE_SOFT_LOCKING
Initializer	(STD_OFF)

2.8.1.29 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 34. Define DIO_USER_MODE_SOFT_LOCKING Description

Name	DIO_USER_MODE_SOFT_LOCKING
Initializer	(STD_OFF)

2.8.1.30 Define DioConf_DioChannel_DioChannel_0

Symbolic name for the channel DioChannel_0.

Table 35. Define DioConf_DioChannel_DioChannel_0 Description

Name	DioConf_DioChannel_DioChannel_0
Initializer	((uint8)0x81U)

2.8.1.31 Define DioConf_DioChannelGroup_DioChannelGroup_0

Symbolic name for the channel group DioChannelGroup_0.

Table 36. Define DioConf_DioChannelGroup_DioChannelGroup_0 Description

Name	DioConf_DioChannelGroup_DioChannelGroup_0
Initializer	(&`DioConfig_0_aChannelGroupList`[0])

2.8.1.32 Define DioConf_DioPort_DioPort_0

Symbolic name for the port DioPort_0.

Table 37. Define DioConf_DioPort_DioPort_0 Description

Name	DioConf_DioPort_DioPort_0
Initializer	((uint8)0x08U)

2.8.1.33 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 38. Define DIO_PRECOMPILE_SUPPORT Description

Name	DIO_PRECOMPILE_SUPPORT
Initializer	

2.8.2 Enum Reference

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

2.8.3 Function Reference

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

2.8.3.1 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 39. Dio_ReadChannel Arguments

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

Table 40. 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.

2.8.3.2 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 41. 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.

2.8.3.3 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 42. Dio_FlipChannel Arguments

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

Table 43. 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.

2.8.3.4 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 44. Dio_ReadPort Arguments

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

2.8.3.5 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 45. 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.

2.8.3.6 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 46. Dio_ReadChannelGroup Arguments

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

2.8.3.7 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 47. Dio_WriteChannelGroup Arguments

Type	Name	Direction	Description
const `Dio_ChannelGroupType *`	ChannelGroupIdPtr	**input**	Pointer to the channel group.
const `Dio_PortLevelType`	Level	**input**	Desired levels for the channel group.

2.8.3.8 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);`

AUTOSAR_MCAL_DIO_UM

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

User manual

Rev. 1 — 11 January 2022

UM2DIOASR4.2 Rev0002R1.0.5

18 / 35

Table 48. Dio_GetVersionInfo Arguments

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

2.8.3.9 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 49. 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.

2.8.4 Structs Reference

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

2.8.4.1 Structure Dio_ChannelGroupType

Type of a DIO channel group representation.
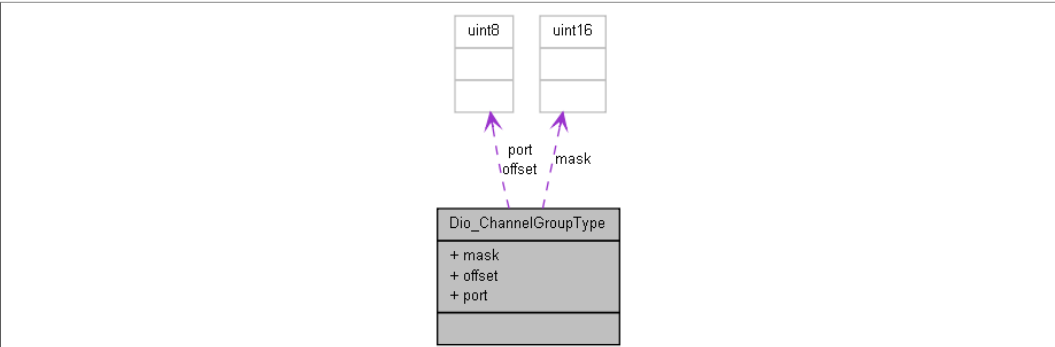
Figure 1. Struct Dio_ChannelGroupType

Declaration:

```
typedef struct
{
    VAR(Dio_PortType, AUTOMATIC)       port;
    VAR(uint8, AUTOMATIC)              offset;
```

AUTOSAR_MCAL_DIO_UM

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

User manual

Rev. 1 — 11 January 2022

UM2DIOASR4.2 Rev0002R1.0.5

19 / 35

```
        VAR(Dio_PortLevelType, AUTOMATIC) mask;
} Dio_ChannelGroupType;
```

Table 50. Structure Dio_ChannelGroupType member description

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

2.8.4.2 Structure Dio_ConfigType

Type of a DIO configuration structure.

Figure 2. Struct Dio_ConfigType
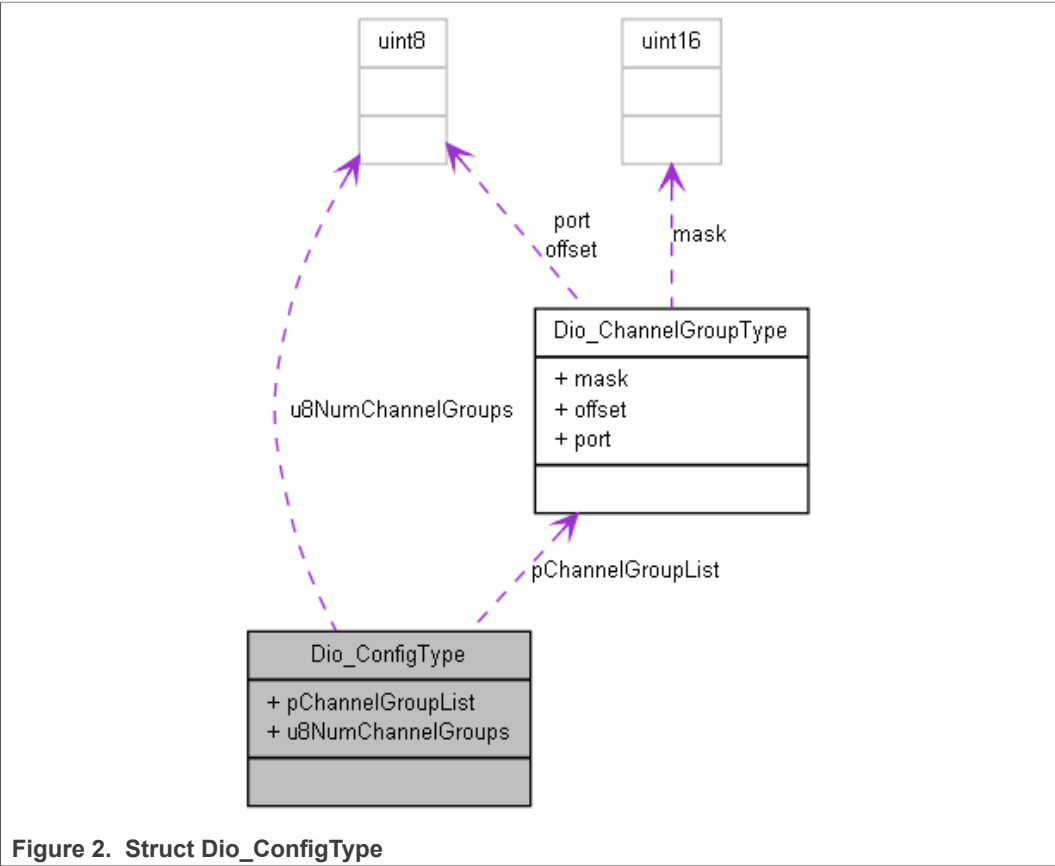
Note: In this implementation there is no need for a configuration structure there is only a dummy field, it is recommended to initialize this field to zero.

Declaration:

```
typedef struct
{

    VAR(uint8, AUTOMATIC)
 u8NumChannelGroups;
    P2CONST(Dio_ChannelGroupType, AUTOMATIC, DIO_APPL_DATA)
 pChannelGroupList;
} Dio_ConfigType;
```

Table 51. Structure Dio_ConfigType member description

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

2.8.5 Types Reference

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

2.8.5.1 Typedef Dio_PortType

Type of a DIO port representation.

Type: `uint8`

2.8.5.2 Typedef Dio_ChannelType

Type of a DIO channel representation.

Type: `uint16`

2.8.5.3 Typedef Dio_PortLevelType

Type of a DIO port levels representation.

Type: `uint16`

2.8.5.4 Typedef Dio_LevelType

Type of a DIO channel levels representation.

Type: `uint8`

2.9 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).

3 Tresos Configuration Plug-in

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

3.1 Configuration elements of Dio

Included forms :

- IMPLEMENTATION_CONFIG_VARIANT
- DioGeneral
- DioConfig

AUTOSAR_MCAL_DIO_UM

User manual

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

Rev. 1 — 11 January 2022

UM2DIOASR4.2 Rev0002R1.0.5

21 / 35

• CommonPublishedInformation

3.2 Form IMPLEMENTATION_CONFIG_VARIANT

VariantPreCompile: Only precompile time configuration parameters.

The files Dio_Cfg.h and Dio_Cfg.c are used.
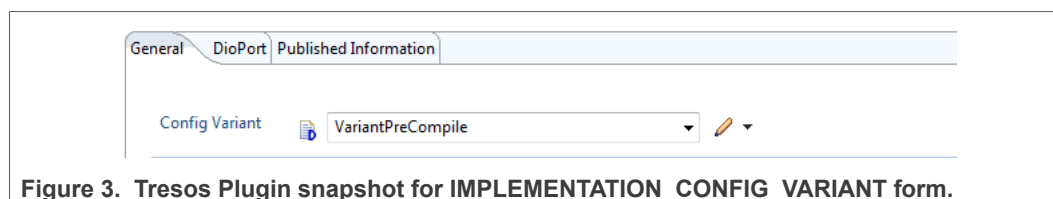
Figure 3. Tresos Plugin snapshot for IMPLEMENTATION_CONFIG_VARIANT form.

Table 52. Attribute IMPLEMENTATION_CONFIG_VARIANT detailed description

Property	Value
Label	Config Variant
Type	ENUMERATION
Default	VariantPreCompile
Range	VariantPreCompile

3.3 Form DioGeneral

General DIO module configuration parameters.
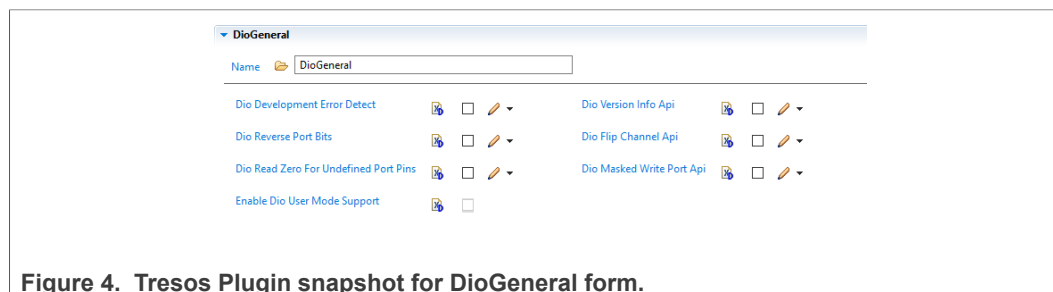
Figure 4. Tresos Plugin snapshot for DioGeneral form.

3.3.1 DioDevErrorDetect (DioGeneral)

Switches the Development Error Detection and Notification ON or OFF.

True: Enabled.

False: Disabled.

Table 53. Attribute DioDevErrorDetect (DioGeneral) detailed description

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

3.3.2 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 54. Attribute DioVersionInfoApi (DioGeneral) detailed description

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

3.3.3 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 55. Attribute DioReversePortBits (DioGeneral) detailed description

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

3.3.4 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 56. Attribute DioFlipChannelApi (DioGeneral) detailed description

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

3.3.5 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 57. Attribute DioReadZeroForUndefinedPortPins (DioGeneral) detailed description

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

3.3.6 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 58. Attribute DioMaskedWritePortApi (DioGeneral) detailed description

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

3.3.7 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 59. Attribute DioEnableUserModeSupport (DioGeneral) detailed description

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

3.4 Form DioConfig

Included forms :

- Section 3.4.1 "Form DioPort"

Figure 5. Tresos Plugin snapshot for DioConfig form.

3.4.1 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 : Section 3.4

Included forms :

- Section 3.4.1.2 "Form DioChannel"
- Section 3.4.1.3 "Form DioChannelGroup"
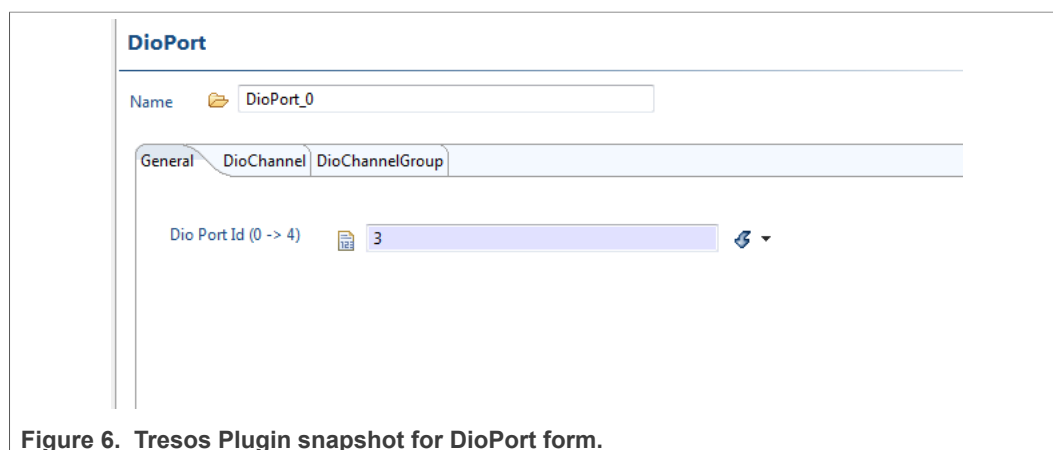
Figure 6. Tresos Plugin snapshot for DioPort form.

3.4.1.1 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 60. Attribute DioPortId (DioPort) detailed description

Property	Value
Label	Dio Port Id
Type	INTEGER
Origin	AUTOSAR_ECUC

AUTOSAR_MCAL_DIO_UM

All information provided in this document is subject to legal disclaimers.

© NXP B.V. 2022. All rights reserved.

User manual

Rev. 1 — 11 January 2022

UM2DIOASR4.2 Rev0002R1.0.5

25 / 35

Table 60. Attribute DioPortId (DioPort) detailed description_...continued_

Property	Value
Symbolic Name	true
Invalid	Range
	<=4
	>=0

3.4.1.2 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 : <u>Section 3.4.1</u>
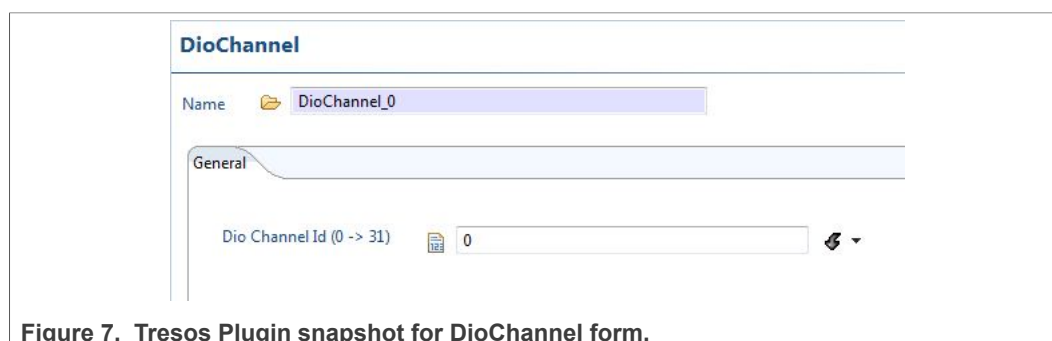
Figure 7. Tresos Plugin snapshot for DioChannel form.

3.4.1.2.1 DioChannelId (DioChannel)

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

Table 61. Attribute DioChannelId (DioChannel) detailed description

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

3.4.1.3 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 : <u>Section 3.4.1</u>
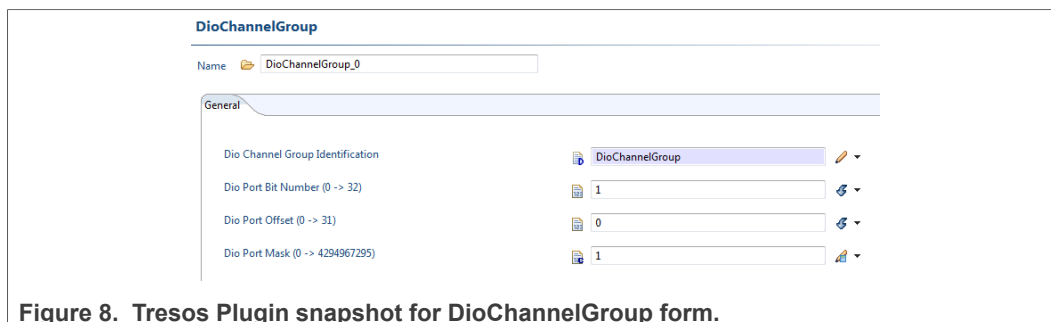
Figure 8. Tresos Plugin snapshot for DioChannelGroup form.

3.4.1.3.1 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 inorder to point to the structure parameters. Example: OutputGroup

Table 62. Attribute DioChannelGroupIdentification (DioChannelGroup) detailed description

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

3.4.1.3.2 DioPortMask (DioChannelGroup)

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

Table 63. Attribute DioPortMask (DioChannelGroup) detailed description

Property	Value
Label	Dio Port Mask
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	1
Invalid	Range >=0 <=4294967295

3.4.1.3.3 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 64. Attribute DioPortOffset (DioChannelGroup) detailed description

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

3.4.1.3.4   DioPortBitNumber (DioChannelGroup)
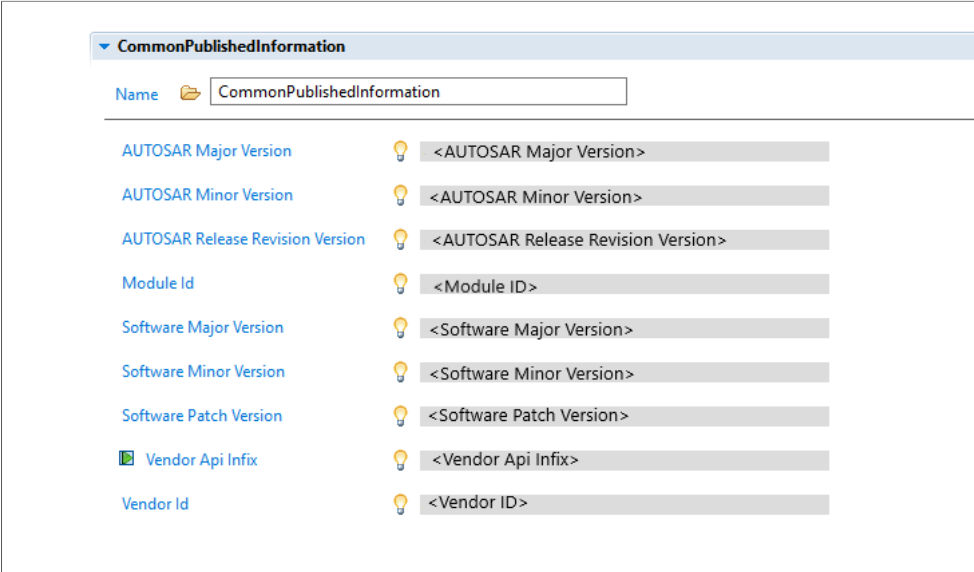
This is the number of continuous channels that create a channel group

Table 65. Attribute DioPortBitNumber (DioChannelGroup) detailed description

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

3.5 Form CommonPublishedInformation

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

Figure 9. Tresos Plugin snapshot for CommonPublishedInformation form.

3.5.1 ArReleaseMajorVersion (CommonPublishedInformation)

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

Table 66. Attribute ArReleaseMajorVersion (CommonPublishedInformation) detailed description

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

3.5.2 ArReleaseMinorVersion (CommonPublishedInformation)

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

Table 67. Attribute ArReleaseMinorVersion (CommonPublishedInformation) detailed description

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

3.5.3 ArReleaseRevisionVersion (CommonPublishedInformation)

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

Table 68. Attribute ArReleaseRevisionVersion (CommonPublishedInformation) detailed description

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

AUTOSAR_MCAL_DIO_UM
© NXP B.V. 2022. All rights reserved.

User manual
Rev. 1 — 11 January 2022
UM2DIOASR4.2 Rev0002R1.0.5
29 / 35

3.5.4 ModuleId (CommonPublishedInformation)

Module ID of this module from Module List.

Table 69. Attribute ModuleId (CommonPublishedInformation) detailed description

Property	Value
Label	Module Id
Type	INTEGER_LABEL
Origin	NXP
Symbolic Name	false
Default	120
Invalid	Range >=120 <=120

3.5.5 SwMajorVersion (CommonPublishedInformation)

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

Table 70. Attribute SwMajorVersion (CommonPublishedInformation) detailed description

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

3.5.6 SwMinorVersion (CommonPublishedInformation)

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

Table 71. Attribute SwMinorVersion (CommonPublishedInformation) detailed description

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

3.5.7 SwPatchVersion (CommonPublishedInformation)

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

Table 72. Attribute SwPatchVersion (CommonPublishedInformation) detailed description

Property	Value
Label	Software Patch Version
Type	INTEGER_LABEL
Origin	NXP
Symbolic Name	false
Default	5
Invalid	Range >=5 <=5

3.5.8 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 73. Attribute VendorApiInfix (CommonPublishedInformation) detailed description

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

3.5.9 VendorId (CommonPublishedInformation)

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

Table 74. Attribute VendorId (CommonPublishedInformation) detailed description

Property	Value
Label	Vendor Id
Type	INTEGER_LABEL
Origin	NXP
Symbolic Name	false
Default	43

Table 74. Attribute VendorId (CommonPublishedInformation) detailed description*...continued*

Property	Value
Invalid	Range >=43 <=43

4   Legal information

4.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

4.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory. Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification. Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products. NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer. In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages. Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately. Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

4.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents