User Manual

for S32K1 DIO Driver

Document Number: UM2DIOASR4.4 Rev0000R1.0.1 Rev. 1.0

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	24.02.2022	NXP RTD Team	Prepared for release RTD S32K1 Version 1.0.1

Chapter 2

Introduction

This User Manual describes NXP Semiconductor AUTOSAR DIO for S32K1. AUTOSAR DIO driver configuration parameters and deviations from the specification are described in 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:

- s32k116_qfn32
- s32k116_lqfp48
- s32k118_lqfp48
- s32k118_lqfp64
- s32k142_lqfp48
- s32k142_lqfp64
- s32k142_lqfp100
- s32k142w_lqfp48
- s32k142w_lqfp64
- s32k144_lqfp48

- s32k144_lqfp64
- s32k144_lqfp100
- s32k144_mapbga100
- s32k144w_lqfp48
- s32k144w_lqfp64
- s32k146_lqfp64
- s32k146_lqfp100
- s32k146_mapbga100
- s32k146_lqfp144
- s32k148_lqfp100
- s32k148_mapbga100
- s32k148_lqfp144
- s32k148_lqfp176

All of the above microcontroller devices are collectively named as S32K1.

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** style: Used for important terms, notes and warnings.
- *Italic* style: 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.

Warning

This is a warning

2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CS	Chip Select
CTU	Cross Trigger Unit
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
FIFO	First In First Out
LSB	Least Signifigant Bit
MCU	Micro Controller Unit
MIDE	Multi Integrated Development Environment
MSB	Most Significant Bit
N/A	Not Applicable
RAM	Random Access Memory
SIU	Systems Integration Unit
SWS	Software Specification
VLE	Variable Length Encoding
XML	Extensible Markup Language

2.5 Reference List

#	Title	Version
1	Specification of Dio Driver	AUTOSAR Release 4.4.0
2	S32K1 Reference Manual	S32K1xx Series Reference Manual, Rev. 14, 09/2021
3	Errata	S32K116_0N96V Rev. 22/OCT/2021
		S32K118_0N97V Rev. 22/OCT/2021
		S32K142_0N33V Rev. 22/OCT/2021
		S32K144_0N57U Rev. 22/OCT/2021
		S32K144W_0P64A Rev. 22/OCT/2021
		S32K146_0N73V Rev. 22/OCT/2021
		S32K148_0N20V Rev. 22/OCT/2021
4	Datasheet	S32K1xx Data Sheet, Rev.14, 08/2021

Chapter 3

Driver

3.1 Requirements

Requirements for this driver are detailed in the Autosar 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 the S32K1 IO Muxing documentation
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
  - PortB
  - PortC
  - PortD
  - PortE
- 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 Deviations from Requirements

The driver deviates from the AUTOSAR DIO Driver software specification in some places. The table identifies the AUTOSAR requirements that are not fully implemented, not implemented or out of scope for the DIO Driver.

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

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

Requirement	Status	Description	Notes
SWS_Dio_00083	N/S	If the microcontroller supports the direct readback 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	Not a requirement.

Requirement	Status	Description	Notes
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 S↩ WS_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_Port↩ Type.
SWS_Dio_00195	N/S	These requirements are not applicable to this specification.	This is not a requirement.
SWS_Dio_00204	N/S	When writing a port which is smaller than the Dio_PortLevelType using the Dio_MaskedWritePort function (see [SWS_Dio_00103]), the function shall ignore the MSB.	Dio_MaskedWritePort function always reads a port of exactly the size defined by Dio_PortLevelType It cannot read a port smaller than the size of the Dio↩ _PortLevelType.

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) 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_Dio↩ ChannelGroup_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.

3.7 Runtime errors

This driver doesn't generate any runtime error.

3.8 Symbolic Names Disclaimer

All containers having symbolicNameValue set to TRUE in the AUTOSAR schema will generate defines like:

#define <Mip>Conf_<Container_ShortName>_<Container_ID>

For this reason it is forbidden to duplicate the names of such containers across the RTD configurations 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 driver. All the parameters are described below.

- Module Dio
  - Container DioConfig
    - Container DioPort
      - Parameter DioPortId
      - Reference DioPortEcucPartitionRef
      - Container DioChannel
      - Parameter DioChannelId
      - Reference DioChannelEcucPartitionRef
      - Container DioChannelGroup
      - Parameter DioChannelGroupIdentification
      - Parameter DioPortBitNumber
      - Parameter DioPortOffset
      - Parameter DioPortMask
      - Reference DioChannelGroupEcucPartitionRef
  - Container DioGeneral
    - Parameter DioDevErrorDetect
    - Parameter GPIODioIPDevErrorDetect
    - Parameter DioVersionInfoApi
    - Parameter DioReversePortBits
    - Parameter DioFlipChannelApi
    - Parameter DioReadZeroForUndefinedPortPins
    - Parameter DioMaskedWritePortApi
    - Parameter DioEnableUserModeSupport
    - Parameter DioMulticoreSupport
    - Reference DioEcucPartitionRef

- Container CommonPublishedInformation
  - Parameter ArReleaseMajorVersion
  - Parameter ArReleaseMinorVersion
  - Parameter ArReleaseRevisionVersion
  - Parameter ModuleId
  - Parameter SwMajorVersion
  - Parameter SwMinorVersion
  - Parameter SwPatchVersion
  - Parameter VendorApiInfix
  - Parameter VendorId

4.1 Module Dio

Configuration of the Dio (Digital IO) module.

Included containers:

- DioConfig
- DioGeneral
- CommonPublishedInformation

Property	Value
type	ECUC-MODULE-DEF
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantSupport	false
supportedConfigVariants	VARIANT-LINK-TIME, VARIANT-PRE-COMPILE

4.2 Container DioConfig

This container contains the configuration parameters and sub containers of the AUTOSAR DIO module. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.

Included subcontainers:

- DioPort

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.3 Container 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.

Included subcontainers:

- DioChannel
- DioChannelGroup

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.4 Parameter DioPortId

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4
min	0

4.5 Reference DioPortEcucPartitionRef

Maps the Dio Port to zero a multiple ECUC partitions. The ECUC partitions referenced are a subset of the ECUC partitions where the Dio driver is mapped to.

EN: Tags: atp.Status=draft

NoteThe S32K1 platform will not support the Multicore feature.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.6 Container 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.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.7 Parameter DioChannelId

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	31
min	0

4.8 Reference DioChannelEcucPartitionRef

EN: Maps the Dio Channel to zero a multiple ECUC partitions. The ECUC partitions referenced are a subset of the ECUC partitions where the related Dio port is mapped to.

EN: Tags: atp.Status=draft

NoteThe S32K1 platform will not support the Multicore feature.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.9 Container 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 and mask.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	255
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.10 Parameter DioChannelGroupIdentification

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

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	DioChannelGroup

4.11 Parameter DioPortBitNumber

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	32
min	1

4.12 Parameter DioPortOffset

The position of the Channel Group on the port, counted from the least significant bit. This value can be derived from DioPortMask.

calculationFormula = Position of the first bit of DioPortMask which is set to 1 counted from least significant bit

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	31
min	0

4.13 Parameter DioPortMask

This shall be the mask which defines the positions of the channel group.

This field holds a decimal value that, when converted into binary, represents a bitmask for the channels in the group(e.g a value of 14 = 0x0E selects channels 1, 2 and 3).

The binary value of the mask should have a single continuous group of 1 bits.

The data type depends on the port width.

To display the correct result you must click on the "calculator" icon near the Dio Port Mask field

Calculation formula: DioPortMask = DioPortBitNumber « DioPortOffset

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: LINK
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.14 Reference DioChannelGroupEcucPartitionRef

EN: Maps the Dio Channel Group to zero a multiple ECUC partitions. The ECUC partitions referenced are a subset of the ECUC partitions where the related Dio port is mapped to.

EN: Tags: atp.Status=draft

NoteThe S32K1 platform will not support the Multicore feature.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.15 Container DioGeneral

General DIO module configuration parameters.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.16 Parameter DioDevErrorDetect

Switches the Development Error Detection and Notification ON or OFF.

True: Enabled.

False: Disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.17 Parameter GPIODioIPDevErrorDetect

Enables and Disables DevAssert checks in IP code.

True: Enabled.

False: Disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.18 Parameter DioVersionInfoApi

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).

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.19 Parameter DioReversePortBits

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 30 and 31 of the port.

This functionality is an AutoSAR extension.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.20 Parameter DioFlipChannelApi

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).

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.21 Parameter DioReadZeroForUndefinedPortPins

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.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.22 Parameter DioMaskedWritePortApi

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.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.23 Parameter DioEnableUserModeSupport

This parameter is added in Dio configuration in order to keep a consistent design over the entire set of RTD drivers.

It cannot be configured by the user and is always set to 'false'.

There are no registers used by the driver which require special measures in order to be accessed from user mode, so Dio driver can be run from either user or supervisor mode.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.24 Parameter DioMulticoreSupport

This parameter globally enables the possibility to support multicore. If this parmeter is enabled, at least one EcucPartition needs to be defined (in all variants).

NoteThe S32K1 platform will not support the Multicore feature.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.25 Reference DioEcucPartitionRef

Maps the Dio driver to zero a multiple ECUC partitions to make the modules API available in this partition.

Tags: atp.Status=draft

NoteThe S32K1 platform will not support the Multicore feature.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.26 Container CommonPublishedInformation

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

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.27 Parameter ArReleaseMajorVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.28 Parameter ArReleaseMinorVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.29 Parameter ArReleaseRevisionVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.30 Parameter ModuleId

Module ID of this module from Module List.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	120
max	120
min	120

4.31 Parameter SwMajorVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	1
max	1
min	1

4.32 Parameter SwMinorVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.33 Parameter SwPatchVersion

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	1
max	1
min	1

4.34 Parameter VendorApiInfix

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>.

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.

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-LINK-TIME: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	

4.35 Parameter VendorId

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

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-LINK-TIME: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	43
max	43
min	43

Chapter 5

Module Index

5.1 Software Specification

Here is a list of all modules:

Chapter 6

Module Documentation

6.1 Dio HLD

6.1.1 Detailed Description

Macros

- #define DIO_E_PARAM_CONFIG
  The DIO module is not properly configured.
- #define DIO_E_PARAM_INVALID_CHANNEL_ID
  Invalid channel name requested.
- #define DIO_E_PARAM_INVALID_PORT_ID
  Invalid port name requested.
- #define DIO_E_PARAM_INVALID_GROUP_ID
  Invalid ChannelGroup id passed.
- #define DIO_E_PARAM_POINTER
  API service called with a NULL pointer.
- #define DIO_E_PARAM_LEVEL
  API service called with invalid channel level value.
- #define DIO_READCHANNEL_ID
  API service ID for `Dio_ReadChannel()` function.
- #define DIO_WRITECHANNEL_ID
  API service ID for `Dio_WriteChannel()` function.
- #define DIO_FLIPCHANNEL_ID
  API service ID for `Dio_FlipChannel()` function.
- #define DIO_READPORT_ID
  API service ID for `Dio_ReadPort()` function.
- #define DIO_WRITEPORT_ID
  API service ID for `Dio_WritePort()` function.
- #define DIO_READCHANNELGROUP_ID
  API service ID for `Dio_ReadChannel()` Group function.
- #define DIO_WRITECHANNELGROUP_ID

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

- #define DIO_GETVERSIONINFO_ID

  API service ID for DIO Get Version() Info function.
- #define DIO_MASKEDWRITEPORT_ID

  API service ID for `Dio_MaskedWritePort()` function.
- #define DIO_INSTANCE_ID

  Instance ID of the Dio driver.

Function Reference

- void Dio_GetVersionInfo (Std_VersionInfoType *VersionInfo)

  Service to get the version information of this module.
- Dio_LevelType Dio_ReadChannel (Dio_ChannelType ChannelId)

  Returns the value of the specified DIO channel.
- void Dio_WriteChannel (Dio_ChannelType ChannelId, Dio_LevelType Level)

  Sets the level of a channel.
- Dio_LevelType Dio_FlipChannel (Dio_ChannelType ChannelId)

  Inverts the level of a channel.
- Dio_PortLevelType Dio_ReadPort (Dio_PortType PortId)

  Returns the level of all channels of specified port.
- void Dio_WritePort (Dio_PortType PortId, Dio_PortLevelType Level)

  Sets the value of a port.
- Dio_PortLevelType Dio_ReadChannelGroup (const Dio_ChannelGroupType *ChannelGroupIdPtr)

  This service reads a subset of the adjoining bits of a port.
- void Dio_WriteChannelGroup (const Dio_ChannelGroupType *ChannelGroupIdPtr, Dio_PortLevelType Level)

  Sets a subset of the adjoining bits of a port to the specified levels.
- void Dio_MaskedWritePort (Dio_PortType PortId, Dio_PortLevelType Level, Dio_PortLevelType Mask)

  DIO Mask write port using mask.

6.1.2 Macro Definition Documentation

6.1.2.1 DIO_E_PARAM_CONFIG

```
#define DIO_E_PARAM_CONFIG
```

The DIO module is not properly configured.

Definition at line 108 of file Dio.h.

6.1.2.2 DIO_E_PARAM_INVALID_CHANNEL_ID

```
#define DIO_E_PARAM_INVALID_CHANNEL_ID
```

Invalid channel name requested.

Definition at line 115 of file Dio.h.

6.1.2.3 DIO_E_PARAM_INVALID_PORT_ID

```
#define DIO_E_PARAM_INVALID_PORT_ID
```

Invalid port name requested.

Definition at line 122 of file Dio.h.

6.1.2.4 DIO_E_PARAM_INVALID_GROUP_ID

```
#define DIO_E_PARAM_INVALID_GROUP_ID
```

Invalid ChannelGroup id passed.

Definition at line 129 of file Dio.h.

6.1.2.5 DIO_E_PARAM_POINTER

```
#define DIO_E_PARAM_POINTER
```

API service called with a NULL pointer.

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

Definition at line 139 of file Dio.h.

6.1.2.6 DIO_E_PARAM_LEVEL

```
#define DIO_E_PARAM_LEVEL
```

API service called with invalid channel level value.

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

Definition at line 149 of file Dio.h.

6.1.2.7 DIO_READCHANNEL_ID

```
#define DIO_READCHANNEL_ID
```

API service ID for `Dio_ReadChannel()` function.

Parameters used when raising an error/exception.

Definition at line 156 of file Dio.h.

6.1.2.8 DIO_WRITECHANNEL_ID

```
#define DIO_WRITECHANNEL_ID
```

API service ID for `Dio_WriteChannel()` function.

Parameters used when raising an error/exception.

Definition at line 162 of file Dio.h.

6.1.2.9 DIO_FLIPCHANNEL_ID

```
#define DIO_FLIPCHANNEL_ID
```

API service ID for `Dio_FlipChannel()` function.

Parameters used when raising an error/exception.

Definition at line 168 of file Dio.h.

6.1.2.10 DIO_READPORT_ID

```
#define DIO_READPORT_ID
```

API service ID for `Dio_ReadPort()` function.

Parameters used when raising an error/exception.

Definition at line 174 of file Dio.h.

6.1.2.11 DIO_WRITEPORT_ID

```
#define DIO_WRITEPORT_ID
```

API service ID for `Dio_WritePort()` function.

Parameters used when raising an error/exception.

Definition at line 180 of file Dio.h.

6.1.2.12 DIO_READCHANNELGROUP_ID

```
#define DIO_READCHANNELGROUP_ID
```

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

Parameters used when raising an error/exception.

Definition at line 186 of file Dio.h.

6.1.2.13 DIO_WRITECHANNELGROUP_ID

```
#define DIO_WRITECHANNELGROUP_ID
```

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

Parameters used when raising an error/exception.

Definition at line 192 of file Dio.h.

6.1.2.14 DIO_GETVERSIONINFO_ID

```
#define DIO_GETVERSIONINFO_ID
```

API service ID for DIO Get Version() Info function.

Parameters used when raising an error/exception.

Definition at line 198 of file Dio.h.

6.1.2.15 DIO_MASKEDWRITEPORT_ID

```
#define DIO_MASKEDWRITEPORT_ID
```

API service ID for `Dio_MaskedWritePort()` function.

Parameters used when raising an error/exception.

Definition at line 204 of file Dio.h.

6.1.2.16 DIO_INSTANCE_ID

```
#define DIO_INSTANCE_ID
```

Instance ID of the Dio driver.

Definition at line 211 of file Dio.h.

6.1.3 Function Reference

6.1.3.1 Dio_GetVersionInfo()

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

Service to get the version information of this module.

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

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

Parameters

in	*VersionInfo*	Pointer to where to store the version information of this module.

6.1.3.2 Dio_ReadChannel()

```
Dio_LevelType Dio_ReadChannel (
            Dio_ChannelType ChannelId )
```

Returns the value of the specified DIO channel.

This function returns the value of the specified DIO channel.

Parameters

in	*ChannelId*	Specifies the required channel id.

Returns

Returns the level of the corresponding pin STD_HIGH or STD_LOW.

6.1.3.3 Dio_WriteChannel()

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

Sets the level of a channel.

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

Parameters

in	*ChannelId*	Specifies the required channel id.
in	*Level*	Specifies the channel desired level.

6.1.3.4 Dio_FlipChannel()

```
Dio_LevelType Dio_FlipChannel (
            Dio_ChannelType ChannelId )
```

Inverts the level of a channel.

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

Parameters

in	*Channel↩ Id*	Specifies the required channel id.

Returns

Returns the level of the corresponding pin as STD_HIGH or STD_LOW.

6.1.3.5 Dio_ReadPort()

```
Dio_PortLevelType Dio_ReadPort (
            Dio_PortType PortId )
```

Returns the level of all channels of specified port.

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

Parameters

in	*Port↩ Id*	Specifies the required port id.

Returns

Levels of all channels of specified port.

6.1.3.6 Dio_WritePort()

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

Sets the value of a port.

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

Parameters

in	*Port↩ Id*	Specifies the required port id.
in	*Level*	Specifies the required levels for the port pins.

6.1.3.7 Dio_ReadChannelGroup()

```
Dio_PortLevelType Dio_ReadChannelGroup (
            const Dio_ChannelGroupType * ChannelGroupIdPtr )
```

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

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

Parameters

in	*ChannelGroupIdPtr*	Pointer to the channel group.

Returns

The channel group levels.

6.1.3.8 Dio_WriteChannelGroup()

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

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

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.

Parameters

in	*ChannelGroupIdPtr*	Pointer to the channel group.
in	*Level*	Desired levels for the channel group.

6.1.3.9 Dio_MaskedWritePort()

```
void Dio_MaskedWritePort (
            Dio_PortType PortId,
            Dio_PortLevelType Level,
            Dio_PortLevelType Mask )
```

DIO Mask write port using mask.

Writes a DIO port with masked value.

Parameters

in	*PortId*	Specifies the required port id.
in	*Level*	Specifies the required levels for the port pins.
in	*Mask*	Specifies the Mask value of the port.

Precondition

This function can be used only if DIO_MASKEDWRITEPORT_API has been enabled.

6.2 Dio IPL

6.2.1 Detailed Description

Types Reference

- typedef uint32 Gpio_Dio_Ip_PinsChannelType

  Type of a GPIO channel representation Implements : Gpio_Dio_Ip_PinsChannelType_Class.
- typedef uint8 Gpio_Dio_Ip_PinsLevelType

  Type of a port levels representation. Implements : Gpio_Dio_Ip_PinsLevelType_Class.

Function Reference

- void Gpio_Dio_Ip_WritePin (GPIO_Type *const base, Gpio_Dio_Ip_PinsChannelType pin, Gpio_Dio_Ip_PinsLevelTy value)

  Write a pin of a port with a given value.
- void Gpio_Dio_Ip_WritePins (GPIO_Type *const base, Gpio_Dio_Ip_PinsChannelType pins)

  Write all pins of a port.
- Gpio_Dio_Ip_PinsChannelType Gpio_Dio_Ip_GetPinsOutput (const GPIO_Type *const base)

  Get the current output from a port.
- void Gpio_Dio_Ip_SetPins (GPIO_Type *const base, Gpio_Dio_Ip_PinsChannelType pins)

  Write pins with 'Set' value.
- void Gpio_Dio_Ip_ClearPins (GPIO_Type *const base, Gpio_Dio_Ip_PinsChannelType pins)

  Write pins to 'Clear' value.
- void Gpio_Dio_Ip_TogglePins (GPIO_Type *const base, Gpio_Dio_Ip_PinsChannelType pins)

  Toggle pins value.
- Gpio_Dio_Ip_PinsChannelType Gpio_Dio_Ip_ReadPins (const GPIO_Type *const base)

  Read input pins.
- Gpio_Dio_Ip_PinsLevelType Gpio_Dio_Ip_ReadPin (const GPIO_Type *const base, Gpio_Dio_Ip_PinsChannelType pin)

  Read input pin.

6.2.2 Types Reference

6.2.2.1 Gpio_Dio_Ip_PinsChannelType

```
typedef uint32 Gpio_Dio_Ip_PinsChannelType
```

Type of a GPIO channel representation Implements : Gpio_Dio_Ip_PinsChannelType_Class.

Definition at line 120 of file Gpio_Dio_Ip.h.

6.2.2.2 Gpio_Dio_Ip_PinsLevelType

```
typedef uint8 Gpio_Dio_Ip_PinsLevelType
```

Type of a port levels representation. Implements : Gpio_Dio_Ip_PinsLevelType_Class.

Definition at line 126 of file Gpio_Dio_Ip.h.

6.2.3 Function Reference

6.2.3.1 Gpio_Dio_Ip_WritePin()

```
void Gpio_Dio_Ip_WritePin (
            GPIO_Type *const base,
            Gpio_Dio_Ip_PinsChannelType pin,
            Gpio_Dio_Ip_PinsLevelType value )
```

Write a pin of a port with a given value.

This function writes the given pin from a port, with the given value ('0' represents LOW, '1' represents HIGH).

Parameters

base	GPIO base pointer (PTA, PTB, PTC, etc.)
pin	pin number to be written
value	pin value to be written • 0: corresponding pin is set to LOW • 1: corresponding pin is set to HIGH

6.2.3.2 Gpio_Dio_Ip_WritePins()

```
void Gpio_Dio_Ip_WritePins (
            GPIO_Type *const base,
            Gpio_Dio_Ip_PinsChannelType pins )
```

Write all pins of a port.

This function writes all pins configured as output with the values given in the parameter pins. '0' represents LOW, '1' represents HIGH.

Parameters

in	*base*	GPIO base pointer (PTA, PTB, PTC, etc.)

Parameters

in	*pins*	Pin mask to be written • 0: corresponding pin is set to LOW • 1: corresponding pin is set to HIGH

6.2.3.3 Gpio_Dio_Ip_GetPinsOutput()

```
Gpio_Dio_Ip_PinsChannelType Gpio_Dio_Ip_GetPinsOutput (
            const GPIO_Type *const base )
```

Get the current output from a port.

This function returns the current output that is written to a port. Only pins that are configured as output will have meaningful values.

Parameters

in	*base*	GPIO base pointer (PTA, PTB, PTC, etc.)

Returns

GPIO outputs. Each bit represents one pin (LSB is pin 0, MSB is pin 31). For each bit:

- 0: corresponding pin is set to LOW
- 1: corresponding pin is set to HIGH

6.2.3.4 Gpio_Dio_Ip_SetPins()

```
void Gpio_Dio_Ip_SetPins (
            GPIO_Type *const base,
            Gpio_Dio_Ip_PinsChannelType pins )
```

Write pins with 'Set' value.

This function configures output pins listed in parameter pins (bits that are '1') to have a value of 'set' (HIGH). Pins corresponding to '0' will be unaffected.

Parameters

in	*base*	GPIO base pointer (PTA, PTB, PTC, etc.)
in	*pins*	Pin mask of bits to be set. Each bit represents one pin (LSB is pin 0, MSB is pin 31). For each bit: • 0: corresponding pin is unaffected • 1: corresponding pin is set to HIGH

6.2.3.5 Gpio_Dio_Ip_ClearPins()

```
void Gpio_Dio_Ip_ClearPins (
            GPIO_Type *const base,
            Gpio_Dio_Ip_PinsChannelType pins )
```

Write pins to 'Clear' value.

This function configures output pins listed in parameter pins (bits that are '1') to have a 'cleared' value (LOW). Pins corresponding to '0' will be unaffected.

Parameters

in	*base*	GPIO base pointer (PTA, PTB, PTC, etc.)
in	*pins*	Pin mask of bits to be cleared. Each bit represents one pin (LSB is pin 0, MSB is pin 31). For each bit: • 0: corresponding pin is unaffected • 1: corresponding pin is cleared(set to LOW)

6.2.3.6 Gpio_Dio_Ip_TogglePins()

```
void Gpio_Dio_Ip_TogglePins (
            GPIO_Type *const base,
            Gpio_Dio_Ip_PinsChannelType pins )
```

Toggle pins value.

This function toggles output pins listed in parameter pins (bits that are '1'). Pins corresponding to '0' will be unaffected.

Parameters

in	*base*	GPIO base pointer (PTA, PTB, PTC, etc.)
in	*pins*	Pin mask of bits to be toggled. Each bit represents one pin (LSB is pin 0, MSB is pin 31). For each bit: • 0: corresponding pin is unaffected • 1: corresponding pin is toggled

6.2.3.7 Gpio_Dio_Ip_ReadPins()

```
Gpio_Dio_Ip_PinsChannelType Gpio_Dio_Ip_ReadPins (
            const GPIO_Type *const base )
```

Read input pins.

This function returns the current input values from a port. Only pins configured as input will have meaningful values.

Parameters

in	*base*	GPIO base pointer (PTA, PTB, PTC, etc.)

Returns

GPIO inputs. Each bit represents one pin (LSB is pin 0, MSB is pin 31). For each bit:

- 0: corresponding pin is read as LOW
- 1: corresponding pin is read as HIGH

6.2.3.8 Gpio_Dio_Ip_ReadPin()

```
Gpio_Dio_Ip_PinsLevelType Gpio_Dio_Ip_ReadPin (
            const GPIO_Type *const base,
            Gpio_Dio_Ip_PinsChannelType pin )
```

Read input pin.

This function returns the current input value of the given pin from port. Only pin configured as input will have meaningful value.

Parameters

in	*base*	GPIO base pointer (PTA, PTB, PTC, etc.)
in	*pin*	Pin index (0,1,2,3,..,15)

Returns

GPIO input value for coressponding pin

- 0: corresponding pin is read as LOW
- 1: corresponding pin is read as HIGH

How to Reach Us:

Home Page:
nxp.com

Web Support:
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.