User Manual

for MPC574XG DIO Driver

Contents

User Manual, Rev. 1.0

Chapter 4
Tresos Configuration Plug-in

User Manual, Rev. 1.0

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	17-Feb-2017	Duc Ta (B53913)	Version for Calypso Release 1.0.0

Chapter 2
Introduction

This User Manual describes NXP Semiconductor AUTOSAR Digital Input Output (Dio) for MPC574XG .

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

2.1 Supported Derivatives

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

Table 2-1.   MPC574XG Derivatives

NXP Semiconductor	MPC5748G_LQFP176,
	MPC5748G_MAPBGA256,
	MPC5748G_MAPBGA324,
	MPC5747G_LQFP176,
	MPC5747G_MAPBGA256,
	MPC5747G_MAPBGA324,
	MPC5746G_LQFP176,
	MPC5746G_MAPBGA256,
	MPC5746G_MAPBGA324,
	MPC5748C_LQFP176,
	MPC5748C_MAPBGA256,
	MPC5748C_MAPBGA324,
	MPC5747C_LQFP176,
	MPC5747C_MAPBGA256,
	MPC5747C_MAPBGA324,
	MPC5746C_LQFP176,
	MPC5746C_MAPBGA256,
	MPC5746C_MAPBGA324,
	MPC5746C_MAPBGA100,
	MPC5745C_LQFP176,
	MPC5745C_MAPBGA256,
	MPC5745C_MAPBGA100,
	MPC5744C_LQFP176,
	MPC5744C_MAPBGA256,
	MPC5744C_MAPBGA100,

Table 2-1. MPC574XG Derivatives

	MPC5746B_LQFP176, MPC5746B_MAPBGA256, MPC5746B_MAPBGA100, MPC5744B_LQFP176, MPC5744B_MAPBGA256, MPC5744B_MAPBGA100, MPC5745B_LQFP176, MPC5745B_MAPBGA256, MPC5745B_MAPBGA100

All of the above microcontroller devices are collectively named as MPC574XG .

2.2 Overview

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

AUTOSAR

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

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

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

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

Notes and warnings are shown as below:

Note

This is a note.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

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

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	AUTOSAR 4.2 Rev0002Dio Driver Software Specification Document.	2.5.0
2	MPC5748G Reference Manual	Rev. 5, 12/2016
3	MPC5746C Reference Manual	Rev. 4, 12/2016
4	MPC5748G_1N81M_Rev.2 (official document) (1N81M)	Jun-16
5	MPC5748G_1N81M_0N78S_Comparison_Summary_v2_0 (internal document) (1N81M, 0N78S)	31.10.2016
6	MPC5746C_1N06M_Rev.4 (official document) (1N06M)	Jul-16
7	MPC5746C_cut1.1_cut2.0_cut2.1_comparison_v0 (internal document) (1N06M, 0N84S, 1N84S)	14-Sep-16
8	C3M_cut2.1_new_errata_20170113 (internal document) (1N84S)	13-Jan-17

User Manual, Rev. 1.0

Chapter 3
Driver

3.1 Requirements

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

3.2 Driver Design 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.

3.3 Hardware Resources

The hardware configured by the Dio driver is SIUL2.

3.4 Deviation from Requirements

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

Table 3-1. Deviations Status Column Description

Term	Definition
N/A	Not available
N/T	Not testable
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, implemented differently, or out of scope for the driver.

Table 3-2. Driver Deviations Table

Requirement	Status	Description	Notes
SWS_Dio_00083	N/I	If the microcontroller supports the direct read-back of a pin value, the Dio module's read functions shall provide the real pin level, when they are used on a channel which is configured as an output channel.	Rejected and replaced by DIO_SW001: The Dio module's read functions shall always provide the value from the input register, regardless of the configured pin direction. Note: The channels must be configured as input or input-output using the PORT driver.
SWS_Dio_00084	N/I	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.	Rejected and replaced by DIO_SW001: The Dio module's read functions shall always provide the value from the input register, regardless of the configured pin direction. Note: The channels must be configured as input or input-output using the PORT driver.
SWS_Dio_00104	N/A	When reading a port which is smaller than the Dio_PortType using the Dio_ReadPort function (see [DIO103]), the function shall set the bits corresponding to undefined port pins to 0.	All ports have equal size (the size of Dio_PortType)
SWS_Dio_00105	N/A	When writing a port which is smaller than the Dio_PortType using the Dio_WritePort function, the function shall ignore the MSB.	All ports have equal size (the size of Dio_PortType)
SWS_Dio_00164	N/I	Dio_ConfigType is the type for all post-build configurable parameters of the DIO driver.	Requirement not applicable. Dio support only pre-compile time configuration.
SWS_Dio_00176	N/A	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 just an empty function. It does not do anything, as there is nothing to configure at run time in the Dio driver, so it performs no check on the received configuration pointer parameter.

3.5   Driver limitations

None

3.6   Driver usage and configuration tips

None

3.7   Runtime Errors

This driver doesn't generate any runtime error.

3.8   Software specification

The following sections contains driver software specifications.

3.8.1   Define Reference

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

3.8.1.1   Define DIO_E_PARAM_CONFIG

API service called with "NULL pointer" parameter.

Details:

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

Table 3-3.   Define DIO_E_PARAM_CONFIG Description

Name	DIO_E_PARAM_CONFIG

Table continues on the next page...

User Manual, Rev. 1.0

Table 3-3. Define DIO_E_PARAM_CONFIG Description (continued)

Initializer	((uint8)0x10)

3.8.1.2 Define DIO_E_PARAM_INVALID_CHANNEL_ID

API service called with invalid channel identifier.

Details:

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

Table 3-4. Define DIO_E_PARAM_INVALID_CHANNEL_ID Description

Name	DIO_E_PARAM_INVALID_CHANNEL_ID
Initializer	((uint8)0x0A)

3.8.1.3 Define DIO_E_PARAM_INVALID_GROUP_ID

API service called with invalid channel group identifier.

Details:

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

Table 3-5. Define DIO_E_PARAM_INVALID_GROUP_ID Description

Name	DIO_E_PARAM_INVALID_GROUP_ID
Initializer	((uint8)0x1F)

3.8.1.4 Define DIO_E_PARAM_INVALID_PORT_ID

API service called with invalid port identifier.

Details:

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

Table 3-6.   Define DIO_E_PARAM_INVALID_PORT_ID Description

Name	DIO_E_PARAM_INVALID_PORT_ID
Initializer	((uint8)0x14)

3.8.1.5   Define DIO_E_PARAM_POINTER

API service called with a NULL pointer.

Details:

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

Table 3-7.   Define DIO_E_PARAM_POINTER Description

Name	DIO_E_PARAM_POINTER
Initializer	((uint8)0x20)

3.8.1.6   Define DIO_E_PARAM_LEVEL

API service called with invalid channel level value.

Details:

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

In detail: If development error detection is enabled, the service Dio_WriteChannel shall check if the specified channel level is valid (either STD_HIGH or STD_LOW). If the "channel level" parameter is invalid, the functions shall report the error code DIO_E_PARAM_LEVEL to the DET.

Table 3-8.   Define DIO_E_PARAM_LEVEL Description

Name	DIO_E_PARAM_LEVEL
Initializer	((uint8)0x21)

3.8.1.7   Define DIO_READCHANNEL_ID

API service ID for `Dio_ReadChannel()` function.

Details:

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

Table 3-9.   Define DIO_READCHANNEL_ID Description

Name	DIO_READCHANNEL_ID
Initializer	((uint8)0x00)

3.8.1.8   Define DIO_WRITECHANNEL_ID

API service ID for `Dio_WriteChannel()` function.

Details:

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

Table 3-10.   Define DIO_WRITECHANNEL_ID Description

Name	DIO_WRITECHANNEL_ID
Initializer	((uint8)0x01)

3.8.1.9   Define DIO_FLIPCHANNEL_ID

API service ID for `Dio_FlipChannel()` function.

Details:

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

Table 3-11.   Define DIO_FLIPCHANNEL_ID Description

Name	DIO_FLIPCHANNEL_ID
Initializer	((uint8)0x11)

3.8.1.10   Define DIO_READPORT_ID

API service ID for `Dio_ReadPort()` function.

Details:

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

Table 3-12.   Define DIO_READPORT_ID Description

Name	DIO_READPORT_ID
Initializer	((uint8)0x02)

3.8.1.11   Define DIO_WRITEPORT_ID

API service ID for `Dio_WritePort()` function.

Details:

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

Table 3-13.   Define DIO_WRITEPORT_ID Description

Name	DIO_WRITEPORT_ID
Initializer	((uint8)0x03)

3.8.1.12   Define DIO_READCHANNELGROUP_ID

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

Details:

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

Table 3-14. Define DIO_READCHANNELGROUP_ID Description

Name	DIO_READCHANNELGROUP_ID
Initializer	((uint8)0x04)

3.8.1.13 Define DIO_WRITECHANNELGROUP_ID

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

Details:

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

Table 3-15. Define DIO_WRITECHANNELGROUP_ID Description

Name	DIO_WRITECHANNELGROUP_ID
Initializer	((uint8)0x05)

3.8.1.14 Define DIO_GETVERSIONINFO_ID

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

Details:

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

Table 3-16. Define DIO_GETVERSIONINFO_ID Description

Name	DIO_GETVERSIONINFO_ID
Initializer	((uint8)0x12)

3.8.1.15 Define DIO_MASKEDWRITEPORT_ID

API service ID for `Dio_MaskedWritePort()` function.

Details:

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

Table 3-17.   Define DIO_MASKEDWRITEPORT_ID Description

Name	DIO_MASKEDWRITEPORT_ID
Initializer	((uint8)0x20)

3.8.1.16   Define DIO_DEV_ERROR_DETECT

Enable/Disable Development Error Detection.

Table 3-18.   Define DIO_DEV_ERROR_DETECT Description

Name	DIO_DEV_ERROR_DETECT
Initializer	(STD_ON)

3.8.1.17   Define DIO_FLIP_CHANNEL_API

Function `Dio_FlipChannel()` enable switch.

Table 3-19.   Define DIO_FLIP_CHANNEL_API Description

Name	DIO_FLIP_CHANNEL_API
Initializer	(STD_ON)

3.8.1.18   Define DIO_MASKEDWRITEPORT_API

Function `Dio_MaskedWritePort()` enable switch.

Table 3-20.   Define DIO_MASKEDWRITEPORT_API Description

Name	DIO_MASKEDWRITEPORT_API
Initializer	(STD_ON)

3.8.1.19   Define DIO_VERSION_INFO_API

Function `Dio_GetVersionInfo()` enable switch.

Table 3-21.   Define DIO_VERSION_INFO_API Description

Name	DIO_VERSION_INFO_API
Initializer	(STD_ON)

3.8.1.20   Define DIO_READZERO_UNDEFINEDPORTS

Undefined pins masking enable switch.

Undefined pins masking enable switch. Defines whether the Dio_ReadPort() function includes the capability to read the undefined port pins as 0.
- True - Enables the Dio_ReadPort() functionality to read the undefined port pins as 0.
- False - Disables the Dio_ReadPort() functionality to read the undefined port pins as 0 (Supports the normal functionality with Dio_ReadPort())

. This functionality is an AutoSAR extension.

Table 3-22.   Define DIO_READZERO_UNDEFINEDPORTS Description

Name	DIO_READZERO_UNDEFINEDPORTS
Initializer	(STD_ON)

3.8.1.21   Define DIO_REVERSEPORTBITS

Reversed port functionality enable switch.

Table 3-23.   Define DIO_REVERSEPORTBITS Description

Name	DIO_REVERSEPORTBITS
Initializer	(STD_OFF)

3.8.1.22   Define DIO_NUM_PORTS_U16

Number of implemented ports.

Note

Used for channel, port and channel group validation.

Table 3-24.   Define DIO_NUM_PORTS_U16 Description

Name	DIO_NUM_PORTS_U16
Initializer	(uint16)0x0A

3.8.1.23   Define DIO_NUM_CHANNELS_PER_PORT_U16

Number channels in a port.

Note

Used for channel, port and channel group validation.

Table 3-25.   Define DIO_NUM_CHANNELS_PER_PORT_U16 Description

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

3.8.1.24   Define DIO_NUM_CHANNELS_U16

Number of channels available on the implemented ports.

Note

Used for channel validation.

Table 3-26.   Define DIO_NUM_CHANNELS_U16 Description

Name	DIO_NUM_CHANNELS_U16
Initializer	(uint16)(DIO_NUM_PORTS_U16 * DIO_NUM_CHANNELS_PER_PORT_U16)

3.8.1.25   Define DIO_NO_AVAILABLE_CHANNELS

Mask representing no available channels on a port.

User Manual, Rev. 1.0

Note

Used for channel validation.

Table 3-27.   Define DIO_NO_AVAILABLE_CHANNELS Description

Name	DIO_NO_AVAILABLE_CHANNELS
Initializer	((Dio_PortLevelType)0x0)

3.8.1.26   Define DIO_MAX_VALID_OFFSET_U8

Mask representing the maximum valid offset for a channel group.

Note

Used for channel group validation.

Table 3-28.   Define DIO_MAX_VALID_OFFSET Description

Name	DIO_MAX_VALID_OFFSET_U8
Initializer	((uint8)0x0F)

3.8.1.27   Define DIO_PRECOMPILE_SUPPORT

Dio driver Pre-Compile configuration switch.

When the switch is enabled, the define DIO_PRECOMPILE_SUPPORT is generated in the code and VariantPreCompile is selected. This means that only precompile time configuration parameters are available. The files Dio_Cfg.h and Dio_Cfg.c are used.

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

Table 3-29.   Define DIO_PRECOMPILE_SUPPORT Description

Name	DIO_PRECOMPILE_SUPPORT
Initializer	

3.8.1.28   Define DIO_ENABLE_USER_MODE_SUPPORT

Note

If this parameter has been configured to 'STD_ON', the Dio driver code can be executed from both supervisor and user mode.

Table 3-30.   Define DIO_ENABLE_USER_MODE_SUPPORT Description

Name	DIO_ENABLE_USER_MODE_SUPPORT
Initializer	(STD_OFF)

3.8.1.29   Define DioConf_DioChannel_DioChannel_0

Symbolic name for the channel DioChannel_0.

Table 3-31.   Define DioConf_DioChannel_DioChannel_0 Description

Name	DioConf_DioChannel_DioChannel_0
Initializer	((uint8)0x81U)

3.8.1.30   Define DioConf_DioChannelGroupIdentification_DioChannelGroup_0

Symbolic name for the channel group DioChannelGroup_0.

Table 3-32.   Define DioConf_DioChannelGroupIdentification_DioChannelGroup_0 Description

Name	DioConf_DioChannelGroupIdentification_DioChannelGroup_0
Initializer	(&DioConfig_0_aChannelGroupList[0])

3.8.1.31   Define DioConf_DioPort_DioPort_0

Symbolic name for the port DioPort_0.

Table 3-33.   Define DioConf_DioPort_DioPort_0 Description

Name	DioConf_DioPort_DioPort_0
Initializer	((uint8)0x08U)

3.8.2   Enum Reference

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

3.8.3   Function Reference

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

3.8.3.1   Function Dio_ReadChannel

Returns the value of the specified DIO channel.

Details:

This function returns the value of the specified DIO channel.

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

Pre: None

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

Table 3-34.   Dio_ReadChannel Arguments

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

Table 3-35.   Dio_ReadChannel Returns

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

3.8.3.2   Function Dio_WriteChannel

Sets the level of a channel.

Details:

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

Pre: None

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

Table 3-36.   Dio_WriteChannel Arguments

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

3.8.3.3   Function Dio_FlipChannel

Inverts the level of a channel.

Details:

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

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

Pre: This function can be used only if DIO_FLIP_CHANNEL_API has been enabled.

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

User Manual, Rev. 1.0

Table 3-37.   Dio_FlipChannel Arguments

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

Table 3-38.   Dio_FlipChannel Returns

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

3.8.3.4   Function Dio_ReadPort

Returns the level of all channels of specified port.

Details:

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

Return: Levels of all channels of specified port.

Pre: None

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

Table 3-39.   Dio_ReadPort Arguments

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

3.8.3.5   Function Dio_WritePort

Sets the value of a port.

Details:

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

Pre: None

User Manual, Rev. 1.0

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

Table 3-40. Dio_WritePort Arguments

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

3.8.3.6 Function Dio_ReadChannelGroup

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

Details:

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

Return: The channel group levels.

Pre: None

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

Table 3-41. Dio_ReadChannelGroup Arguments

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

3.8.3.7 Function Dio_WriteChannelGroup

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

Details:

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

User Manual, Rev. 1.0

Pre: None

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

Table 3-42. 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.

3.8.3.8 Function Dio_GetVersionInfo

Service to get the version information of this module.

Details:

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

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

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

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

Table 3-43. Dio_GetVersionInfo Arguments

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

3.8.3.9 Function Dio_MaskedWritePort

DIO Mask write port using mask.

Details:

DIO write port using mask.

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

Prototype: `void Dio_MaskedWritePort(const Dio_PortType PortId, const Dio_PortLevelType Level, const Dio_PortLevelType Mask);`

Table 3-44. Dio_MaskedWritePort Arguments

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

3.8.4 Structs Reference

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

3.8.4.1 Structure Dio_ChannelGroupType
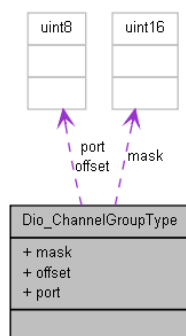
Type of a DIO channel group representation.

Figure 3-1. Struct Dio_ChannelGroupType

Declaration:

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

User Manual, Rev. 1.0

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

Table 3-45.   Structure Dio_ChannelGroupType member description

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

3.8.4.2   Structure Dio_ConfigType
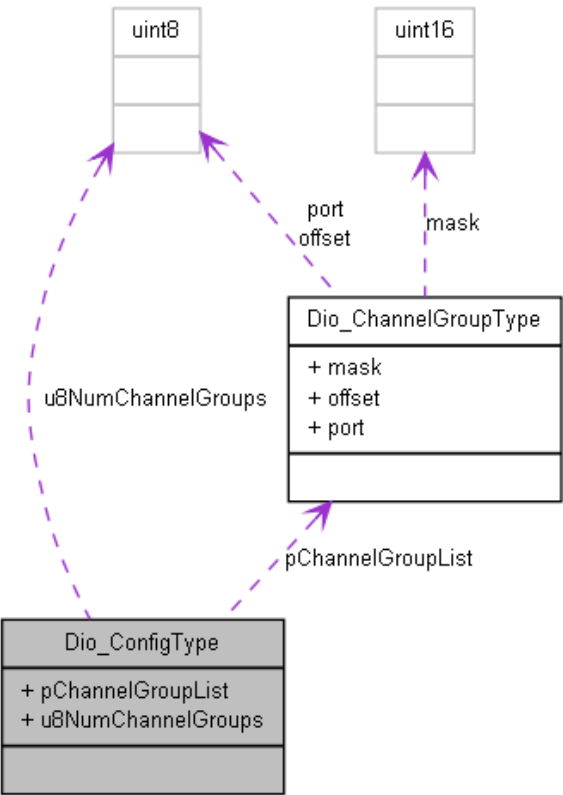
Type of a DIO configuration structure.

Figure 3-2. Struct Dio_ConfigType

Note

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

Declaration:

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

Table 3-46. Structure Dio_ConfigType member description

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

3.8.5   Types Reference

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

3.8.5.1   Typedef Dio_PortType

Type of a DIO port representation.

Type: uint8

3.8.5.2   Typedef Dio_ChannelType

Type of a DIO channel representation.

Type: uint16

3.8.5.3   Typedef Dio_PortLevelType

Type of a DIO port levels representation.

Type: uint16

3.8.5.4   Typedef Dio_LevelType

Type of a DIO channel levels representation.

Type: uint8

3.9 Symbolic Names Disclaimer

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

#define <Container_Short_Name> <Container_ID>

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

Chapter 4
Tresos Configuration Plug-in

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

4.1 Configuration elements of Dio

Included forms :
- IMPLEMENTATION_CONFIG_VARIANT
- DioGeneral
- CommonPublishedInformation
- DioConfig

Table 4-1. Revision table

Revision	Date
4.1.0	2010-12-03

4.2 Form IMPLEMENTATION_CONFIG_VARIANT

VariantPreCompile: Only precompile time configuration parameters.

The files Dio_Cfg.h and Dio_Cfg.c are used.
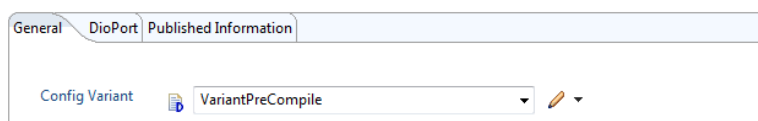
Figure 4-1. Tresos Plugin snapshot for IMPLEMENTATION_CONFIG_VARIANT form.

Table 4-2.   Attribute IMPLEMENTATION_CONFIG_VARIANT detailed description

Property	Value
Label	Config Variant
Type	ENUMERATION
Default	VariantPreCompile
Range	VariantPreCompile

4.3   Form DioGeneral

General DIO module configuration parameters.
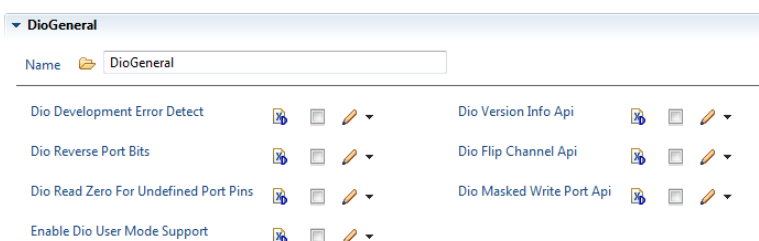
Figure 4-2. Tresos Plugin snapshot for DioGeneral form.

4.3.1   DioDevErrorDetect (DioGeneral)

Switches the Development Error Detection and Notification ON or OFF.

True: Enabled.

False: Disabled.

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

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

4.3.2 DioVersionInfoApi (DioGeneral)

Adds / removes the service Dio_GetVersionInfo() from the code.

True - Dio_GetVersionInfo() API is enabled.

False - Dio_GetVersionInfo() API is disabled (it cannot be used).

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

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

4.3.3 DioReversePortBits (DioGeneral)

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

This functionality is an AutoSAR extension.

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

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

4.3.4 DioFlipChannelApi (DioGeneral)

Adds / removes the service Dio_FlipChannel() from the code.

True - Dio_FlipChannel() API is enabled.

False - Dio_FlipChannel() API is disabled (it cannot be used).

User Manual, Rev. 1.0

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

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

4.3.5   DioReadZeroForUndefinedPortPins (DioGeneral)

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

True - Enables the Dio_ReadPort() functionality to read the undefined port pins as 0.

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

This functionality is an AutoSAR extension.

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

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

4.3.6   DioMaskedWritePortApi (DioGeneral)

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

This API is an AutoSAR extension.

True - Dio_MaskedWritePort() API enabled.

False - Dio_MaskedWritePort() API disabled.

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

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

4.3.7   DioEnableUserModeSupport (DioGeneral)

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

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

Note: Implementation Specific Parameter

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

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

4.4   Form CommonPublishedInformation

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

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

4.4.1   ArReleaseMajorVersion (CommonPublishedInformation)

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

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

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

4.4.2   ArReleaseMinorVersion (CommonPublishedInformation)

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

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

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

4.4.3 ArReleaseRevisionVersion (CommonPublishedInformation)

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

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

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

4.4.4 ModuleId (CommonPublishedInformation)

Module ID of this module from Module List.

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

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

Table continues on the next page...

User Manual, Rev. 1.0

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

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

4.4.5 SwMajorVersion (CommonPublishedInformation)

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

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

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

4.4.6 SwMinorVersion (CommonPublishedInformation)

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

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

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

4.4.7 SwPatchVersion (CommonPublishedInformation)

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

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

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

4.4.8 VendorApiInfix (CommonPublishedInformation)

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name. This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows:
<ModuleName>_>VendorId>_<VendorApiInfix><Api name from SWS>. E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name Can_Write defined in the SWS will translate to Can_123_v11r456Write. This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

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

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

User Manual, Rev. 1.0

4.4.9 VendorId (CommonPublishedInformation)

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

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

Property	Value
Label	Vendor Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	43
Invalid	Range >=43 <=43

4.5 Form DioConfig

Included forms :
  • Form DioPort
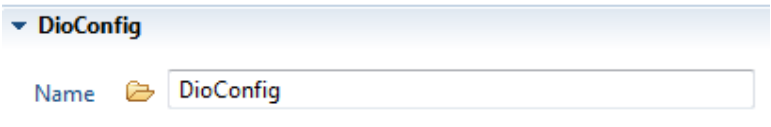
Figure 4-4. Tresos Plugin snapshot for DioConfig form.

4.5.1 Form DioPort

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

Is included by form : Form DioConfig

Included forms :
  • Form DioChannel
  • Form DioChannelGroup

User Manual, Rev. 1.0

NXP Semiconductors

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

4.5.1.1 DioPortId (DioPort)

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

- PortA=0
- PortB=1
- PortC=2
- PortD=3
- PortE=4
- PortF=5
- PortG=6
- PortH=7
- PortI=8
- PortJ=9
- PortK=10
- PortL=11
- PortM=12
- PortN=13
- PortO=14
- PortP=15
- PortQ=16

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

Property	Value
Label	Dio Port Id
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	true
Invalid	Range <=16 >=0

4.5.1.2   Form DioChannel

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

Is included by form : Form DioPort
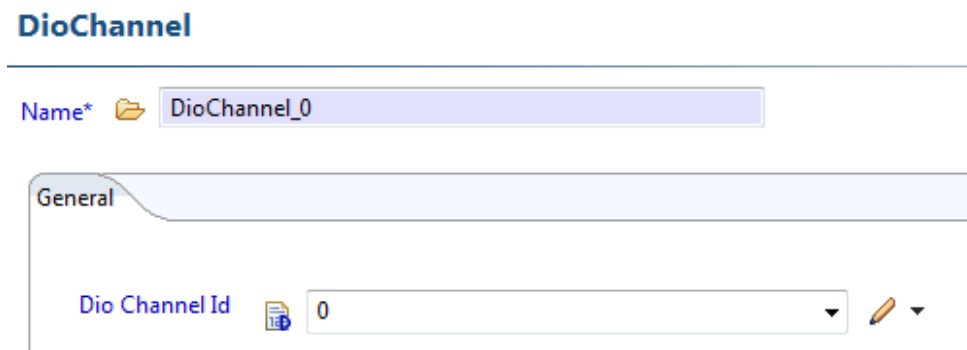
Figure 4-6. Tresos Plugin snapshot for DioChannel form.

4.5.1.2.1   DioChannelId (DioChannel)

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

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

Property	Value
Label	Dio Channel Id
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	true
Invalid	Range <=15 >=0

4.5.1.3   Form DioChannelGroup

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

User Manual, Rev. 1.0

Is included by form : Form DioPort

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

4.5.1.3.1   DioChannelGroupIdentification (DioChannelGroup)

A DIO channel group is identified in DIO APIs by a pointer to a data structure of type Dio_ChannelGroupType. This data structure contains the channel group information. This parameter contains the code fragment that has to be inserted in the API call of the calling module to get the address of the variable in memory which holds the channel group information, a string value should be given for this parameter. Symbolic names will be generated for each DioChannelGroup, which will be assigned with address of this string inorder to point to the structure parameters. Example: OutputGroup

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

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

4.5.1.3.2   DioPortBitNumber (DioChannelGroup)

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

Table 4-22.   Attribute DioPortBitNumber (DioChannelGroup) detailed description

Property	Value
Label	Dio Port Bit Number
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	1
Invalid	Range >=1 <=16

4.5.1.3.3   DioPortMask (DioChannelGroup)

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

Table 4-23.   Attribute DioPortMask (DioChannelGroup) detailed description

Property	Value
Label	Dio Port Mask
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	1
Invalid	Range >=1 <=65535

4.5.1.3.4   DioPortOffset (DioChannelGroup)

The position of the Channel Group on the port, counted from the LSB. This value can be derived from DioPortMask. calculationFormula = Position of the first bit of DioPortMask which is set to '1' counted from LSB

Table 4-24.   Attribute DioPortOffset (DioChannelGroup) detailed description

Property	Value
Label	Dio Port Offset
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Invalid	Range

Table 4-24. Attribute DioPortOffset (DioChannelGroup) detailed description

Property	Value
	>=0 <=15