MICROSAR DIO

Technical Reference

MCAL Emulation in VTT

Version 1.1.0

Authors	Peter Lang, Christian Leder
Status	Released

Document Information

History

Author	Date	Version	Remarks
Peter Lang	2013-09-17	1.00.00	Initial creation of document
Christian Leder	2015-02-03	1.01.00	> Global renaming of Vip to Vtt > Usage of template 5.11.0 for the Technical reference

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_DIODriver.pdf	V2.5.0
[2]	AUTOSAR	AUTOSAR_SWS_DevelopmentErrorTracer.pdf	V3.2.0
[3]	AUTOSAR	AUTOSAR_SWS_DiagnosticEventManager.pdf	V4.2.0
[4]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	V1.6.0

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
1.0.x	Initial version of the Vip DIO driver
2.0.x	Global renaming of Vip to Vtt

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module DIO as specified in [1].

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	DIO_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	DIO_MODULE_ID	120 decimal (according to ref. [4])

* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

The DIO module provides services to control the state of emulated digital I/O pins. These services include

- reading and writing values of individual channels (meaning pins of the microcontroller)
- reading and writing values of whole ports
- reading and writing values of channel groups (adjoining pins in one port)

In this MCAL emulation the microcontroller is emulated. Instead of reading and setting I/O registers on a hardware device, this driver reads and sets CANoe system variables to simulate digital input / output.

2.1 Architecture Overview

The following figure shows where the DIO is located in the AUTOSAR architecture.

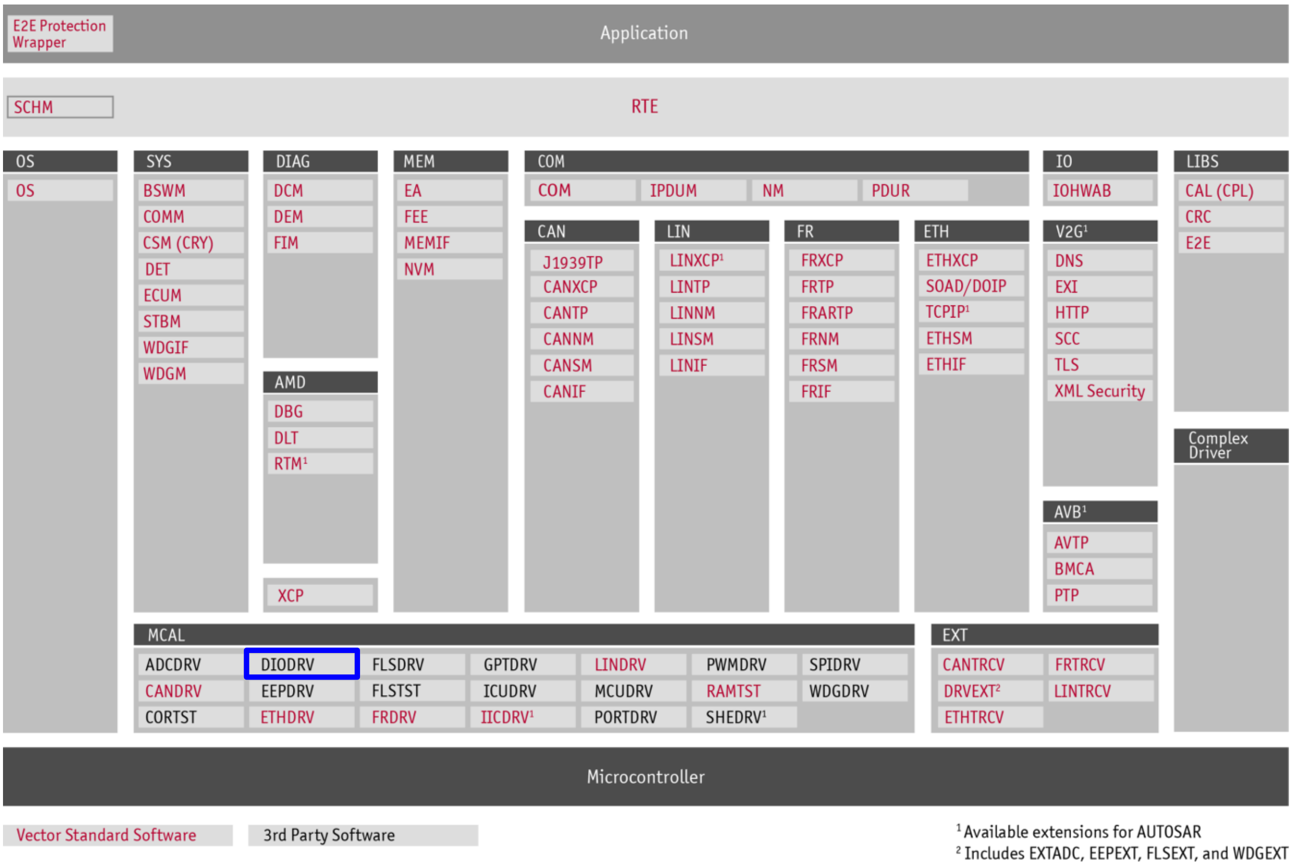

Figure 2-1 AUTOSAR 4.x Architecture Overview

Note
The Microcontroller in Figure 2-1 is emulated by the VTT framework.

The next figure shows the interfaces to adjacent modules of the DIO. These interfaces are described in chapter 5.

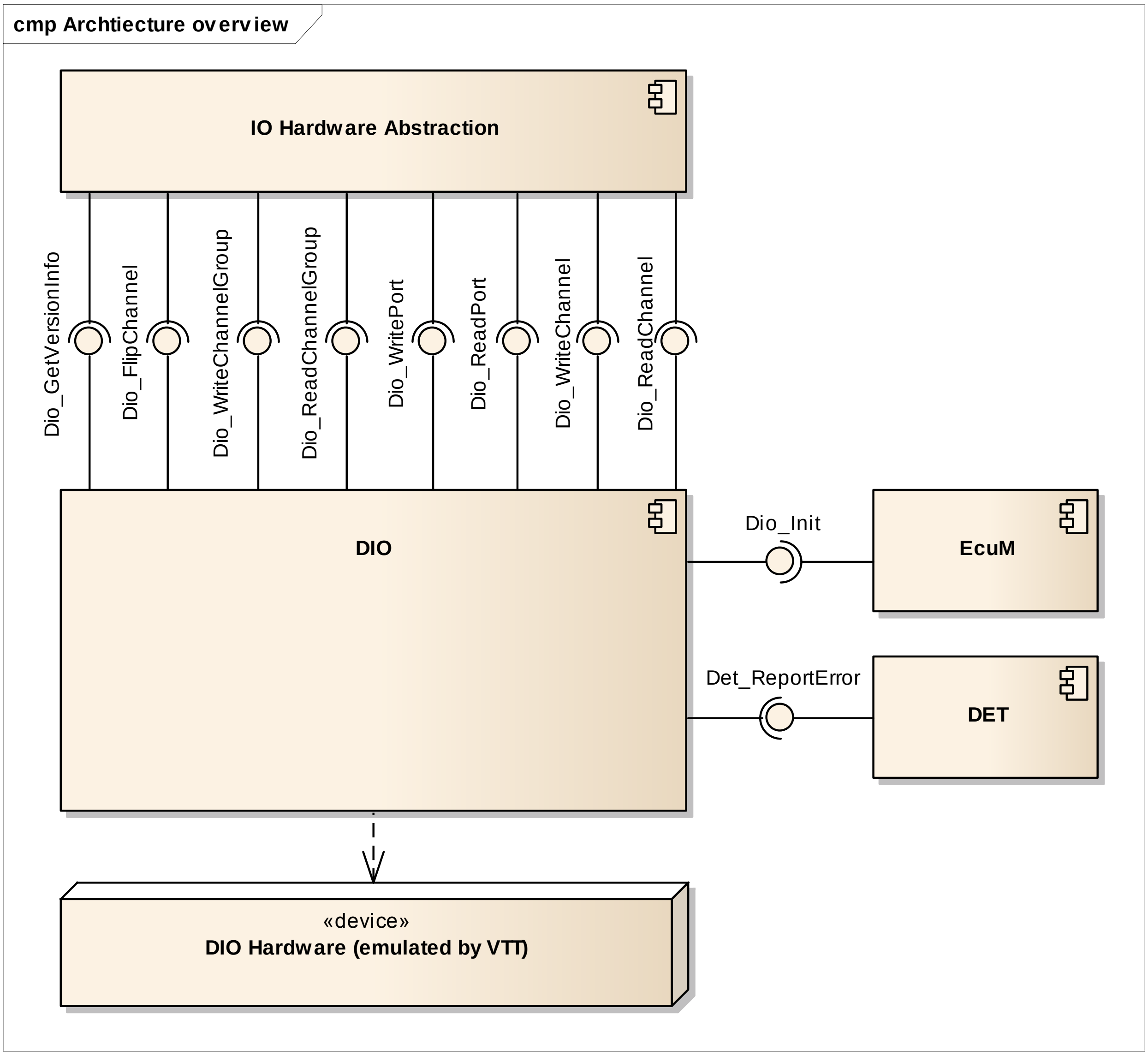

Figure 2-2 Interfaces to adjacent modules of the DIO

Info
The Digital I/O Hardware in the picture above is emulated by the VTT framework.

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for the DIO.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported AUTOSAR standard conform features

> Table 3-2 Not supported AUTOSAR standard conform features

Vector Informatik provides further DIO functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Reading and writing channels
Reading and writing ports
Reading and writing channel groups

Table 3-1 Supported AUTOSAR standard conform features

3.1.1 Deviations

The following features specified in [1] are not supported:

Not Supported AUTOSAR Standard Conform Features
None

Table 3-2 Not supported AUTOSAR standard conform features

3.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
None

Table 3-3 Features provided beyond the AUTOSAR standard

3.1.3 Limitations

No limitations exist due to Autosar version 4.x.

3.1.4 No dependencies between channels, channel groups and ports

The levels or values of a channel, a channel group or a port are stored in separate CANoe system variables. These variables are **not synchronized**. Changing a port does not change its channels or channel groups and vice versa.

Example: If a DIO port is set to zero by calling `Dio_WritePort(Port_A, 0)`, then only the environment variable of this port is set to zero but channels or channel groups that belong to that port are not changed. If a channel, that is part of the port, is read afterwards by calling `Dio_ReadChannel(Port_A_Channel_0)`, then it does not automatically return zero, because the channel's environment variable has not been changed when setting the whole port.

3.2 Emulation

This driver emulates the digital input / output of a microcontroller in the VTT framework.

The whole functionality is emulated with the help of system variables. Instead of reading and writing channels, ports and channel groups, the respective values are stored in system variables. These variables act as an interface for communication between module DIO and the VTT framework.

Caution

Be careful using while loops in order to poll any status.

The user has to ensure, that the application does not block the emulation. So, within every while loop the following function call has to be called:

```
while(ANY_STATUS == temp_status)
{
  Schedule();
}
```

Use the function call Schedule() which is available once the header file of the module DIO is included.

3.3 Initialization

The DIO module is being initialized by calling `Dio_Init(&DioConfig)`. All global variables are initialized by calling `Dio_InitMemory()`. So, `Dio_InitMemory()` has to be **called prior** to `Dio_Init()`.

3.4 States

Module DIO does not implement a state machine.

3.5 Main Functions

Module DIO does not provide any cyclic main functions.

3.6 Error Handling

3.6.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `DIO_DEV_ERROR_DETECT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported DIO ID is 120.

The reported service IDs identify the services which are described in 5.2. The following table presents the service IDs and the related services:

Service ID	Service
0x00	`Dio_ReadChannel`
0x01	`Dio_WriteChannel`
0x02	`Dio_ReadPort`
0x03	`Dio_WritePort`
0x04	`Dio_ReadChannelGroup`
0x05	`Dio_WriteChannelGroup`
0x10	`Dio_Init`
0x12	`Dio_GetVersionInfo`

Table 3-4 Service IDs

The errors reported to DET are described in the following table:

Error Code		Description
0x0A	`DIO_E_PARAM_INVALID_CHANNEL_ID`	The parameter `ChannelId` passed to `Dio_ReadChannel` or `Dio_WriteChannel` is invalid, that is, no channel with that ID is configured.
0x10	`DIO_E_PARAM_CONFIG`	In case the DIO configuration pointer is a `NULL_PTR` this error will be reported to the DET module
0x14	`DIO_E_PARAM_INVALID_PORT_ID`	The parameter `PortId` passed to `Dio_ReadPort` or `Dio_WritePort` is invalid, that is, no port with that ID is configured.
0x1F	`DIO_E_PARAM_INVALID_GROUP`	The parameter `ChannelGroupIdPtr` passed to `Dio_ReadChannelGroup` or `Dio_WriteChannelGroup` either references `NULL_PTR` or the contained port is invalid, that is, no port with that ID is configured.
0x20	`DIO_E_PARAM_VINFO`	The parameter `VersionInfo` references `NULL_PTR`.

Table 3-5 Errors reported to DET

3.6.1.1 Parameter Checking

AUTOSAR requires that API functions check the validity of their parameters. The checks in Table 3-6 are internal parameter checks of the API functions. These checks are for development error reporting and can be en-/disabled.

The following table shows which parameter checks are performed on which services:

Check Service	DIO_E_PARAM_INVALID_CHANNEL_ID	DIO_E_PARAM_INVALID_PORT_ID	DIO_E_PARAM_INVALID_GROUP	DIO_E_PARAM_VINFO
Dio_ReadChannel	■			
Dio_WriteChannel	■			
Dio_ReadPort		■		
Dio_WritePort		■		
Dio_ReadChannelGroup			■	
Dio_WriteChannelGroup			■	
Dio_GetVersionInfo				■

Table 3-6 Development Error Detection: Assignment of checks to services

3.6.2 Production Code Error Reporting

Info

Production errors are not supported in this emulation.

4 Integration

This chapter gives necessary information for the integration of the MICROSAR DIO into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the DIO contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Description
Dio.h	The module header defines the interface of the DIO. This file must be included by upper layer software components
Dio.c	This C-source contains the implementation of the module's functionalities
DrvDio_VttCanoe01Asr.jar	This jar-file contains the generator and the validator for the DaVinci Configurator
VTTDio_bswmd.arxml	Basic Software Module Description according to AUTOSAR for VTT Emulation
Dio_bswmd.arxml	Optional Basic Software Module Description. Placeholder for real target (semiconductor manufacturer) in VTT only use case

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator.

File Name	Description
Dio_Cfg.h	The configuration-header contains the static configuration part of this module
Dio_PBcfg.c	The configuration-source contains the object independent part of the runtime configuration

Table 4-2 Generated files

4.2 Include Structure

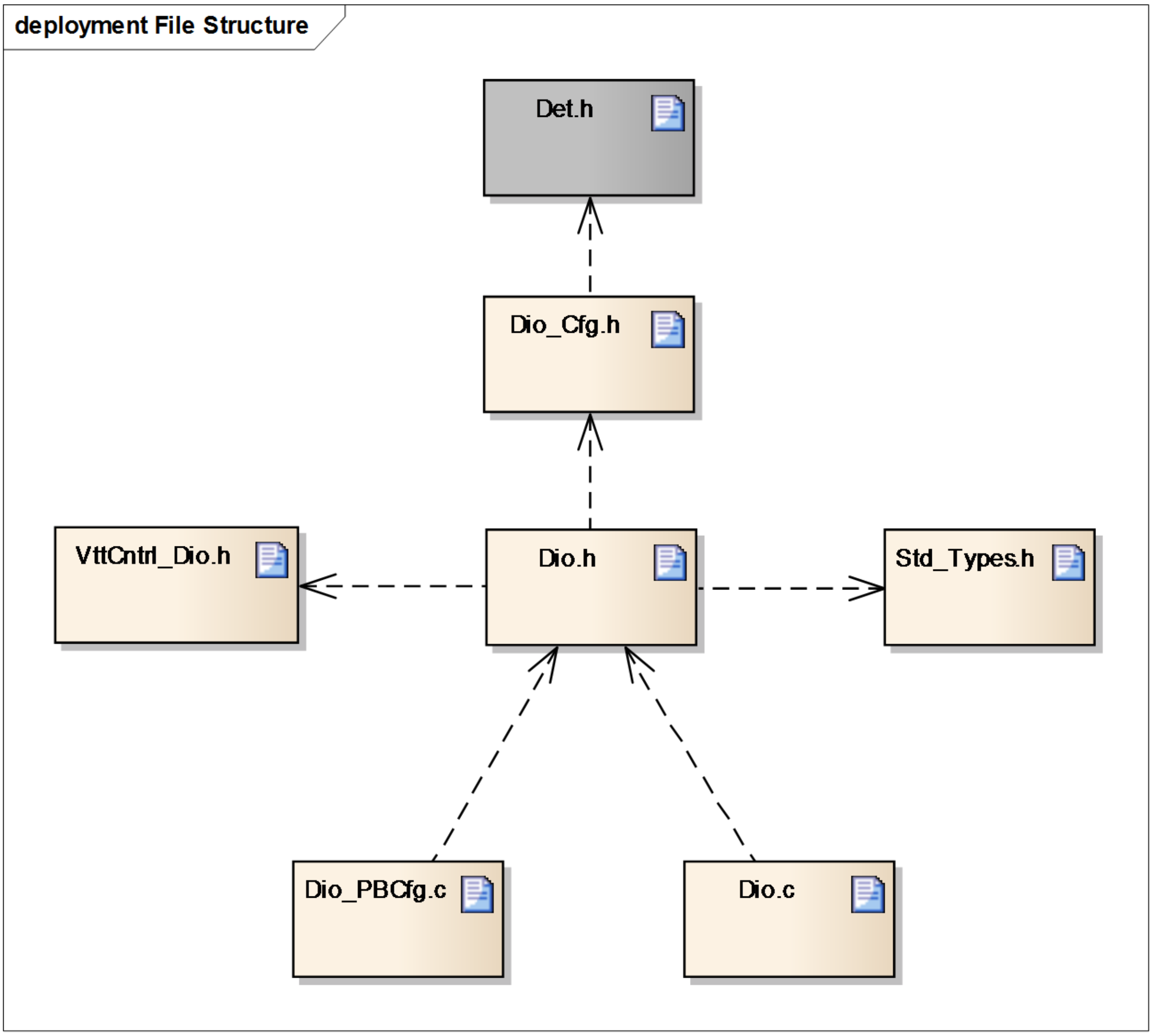

Figure 4-1 Include structure

4.3 Dependencies on SW Modules

4.3.1 DET (Optional)

The DIO module depends on the DET (by default) in order to report development errors. Detection and reporting of development errors can be enabled or disabled by the switch "Enable Development Error Detection".

4.3.2 EcuM

The EcuM cares for the initialization of the module DIO.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

The types defined by the DIO are described in this chapter.

Type Name	C-Type	Description	Value Range
Dio_ChannelType	uint32	This type identifies a single pin of a port.	Although this type allows a 32bit range, only distinct values are accepted by the API services. These values are generated by DaVinci Configurator into symbolic names that should be used instead.
Dio_PortType	uint16	This type identifies a whole port.	Although this type allows a 16bit range, only distinct values are accepted by the API services. These values are generated by DaVinci Configurator into symbolic names that should be used instead.
Dio_LevelType	uint8	This type is used for the state of a single pin of a port.	STD_HIGH The physical state of the port's pin is "HIGH".
			STD_LOW The physical state of the port's pin is "LOW".
Dio_PortLevelType	uint16	This type is used for the state of a whole port.	A "1" corresponds to a HIGH-level applied physically to the pin at the corresponding bit position.
			A "0" corresponds to a LOW-level applied physically to the pin at the corresponding bit position.
Dio_ConfigType	uint8	This type is used for initialization of the module	Symbolic Name Value generated by DaVinci Configurator

Table 5-1 Type definitions

DIO_ChannelGroupType

Struct Element Name	C-Type	Description	Value Range
port	Dio_ PortType	Port ID of the channel group.	Refer to Dio_PortType for value range.
mask	uint8/16/32	This shall be the mask which defines the	Refer to Dio_PortLevelType for value range.

Struct Element Name	C-Type	Description	Value Range
		positions of the channel group. The channels shall consist of adjoining bits in the same port. The data type depends on the port width.	
offset	uint8	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	0 to 31 Where “0” is used if the channel group begins with the first pin (least significant bit) and “31” means, only the last pin of the port (most significant bit) is used.

Table 5-2 Dio_ChannelGroupType

Info
The channel groups can easily be configured in DaVinci Configurator and presumably there won’t be any use case where it is necessary to build up a Dio_ChannelGroupType-instance on your own. So consider the above explanation of the structure only as informational and simply use the generated symbolic names.

5.2 Services provided by DIO

The DIO API consists of services, which are realized by function calls.

5.2.1 Dio_Init

Prototype	
`void Dio_Init (P2CONST(Dio_ConfigType, DIO_CONST, DIO_CODE)ConfigPtr)`	
Parameter	
`ConfigPtr`	Pointer to the configuration struct of the DIO
Return code	
-	-
Functional Description	
The service initializes the module.	

Particularities and Limitations
> This function is synchronous. > This function is non-reentrant. > Module must not be initialized.
Expected Caller Context
> ECU State Manager or comparable software module, responsible for driver initialization after startup.

Table 5-3 Dio_Init

5.2.2 Dio_ReadChannel

Prototype	
`Dio_LevelType **Dio_ReadChannel** (Dio_ChannelType ChannelId)`	
Parameter	
ChannelId	Symbolic channel name of a port pin as generated into Dio_Cfg.h.
Return code	
STD_HIGH	Physical level “HIGH” is applied to the pin
STD_LOW	Physical level “LOW” is applied to the pin
Functional Description	
Reads out the physical level of the selected port pin and returns this value.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different channels.	
Expected Caller Context	
> Task context	

Table 5-4 Dio_ReadChannel

5.2.3 Dio_WriteChannel

Prototype	
`void **Dio_WriteChannel** (Dio_ChannelType ChannelId, Dio_LevelType Level)`	
Parameter	
ChannelId	Symbolic channel name of a port pin as generated into Dio_Cfg.h.
Level	Level to write to the port pin (`STD_HIGH` or `STD_LOW`)
Return code	
-	-
Functional Description	
Writes the given level to the selected port pin. A value of `STD_HIGH` switches a pin to physical state “HIGH” and `STD_LOW` switches a pin to physical state “LOW”	

Particularities and Limitations
> This function is synchronous. > This function is reentrant for different channels.
Expected Caller Context
> Task context

Table 5-5 Dio_WriteChannel

5.2.4 Dio_ReadPort

Prototype	
`Dio_PortLevelType **Dio_ReadPort** (Dio_PortType PortId)`	
Parameter	
`PortId`	Symbolic name of a port as generated into Dio_Cfg.h.
Return code	
`Dio_PortLevelType`	This mask denotes the level of the port's pins. A "1" indicates that the physical level "HIGH" is applied to the pin at the corresponding bit position. A "0" indicates that the physical level "LOW" is applied to the pin at the corresponding bit position.
Functional Description	
Reads out the physical level of the selected port and returns this value.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different ports.	
Expected Caller Context	
> Task context	

Table 5-6 Dio_ReadPort

5.2.5 Dio_WritePort

Prototype	
`void **Dio_WritePort** (Dio_PortType PortId, Dio_PortLevelType Level)`	
Parameter	
`PortId`	Symbolic name of a port as generated into Dio_Cfg.h.
`Level`	Mask to write to the port. A "1" indicates that the pin at the corresponding bit position should be set to the physical level "HIGH". A "0" indicates that the pin at the corresponding bit position should be set to the physical level "LOW".
Return code	
-	-
Functional Description	
Writes the given level mask to the selected port.	

Particularities and Limitations
> This function is synchronous. > This function is reentrant for different ports.
Expected Caller Context
> Task context

Table 5-7 Dio_WritePort

5.2.6 Dio_ReadChannelGroup

Prototype	

`Dio_PortLevelType` **`Dio_ReadChannelGroup`**
`(`
`  P2CONST(Dio_ChannelGroupType, AUTOMATIC, DIO_APPL_CONST)`
`  ChannelGroupId`
`)`	

Parameter	
`ChannelGroupId`	Reference to a channel group definition.
Return code	
`Dio_PortLevelType`	This mask denotes the level of the channel group’s pins. A “1” indicates that the physical level “HIGH” is applied to the pin at the corresponding bit position. A “0” indicates that the physical level “LOW” is applied to the pin at the corresponding bit position. This mask is “right aligned” so that the channel group’s rightmost (“least significant”) pin is shifted to bit position “0” in the mask.
Functional Description	
Reads out the physical level of the referenced channel group’s pins and returns this mask.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different channel groups.	
Expected Caller Context	
> Task context	

Table 5-8 Dio_ReadChannelGroup

5.2.7 Dio_WriteChannelGroup

Prototype	

void **Dio_WriteChannelGroup**
(
  P2CONST(Dio_ChannelGroupType, AUTOMATIC, DIO_APPL_CONST)
  ChannelGroupId,
  Dio_PortLevelType Level
)	

Parameter	
ChannelGroupIdPtr	Reference to a channel group definition.
Level	Mask to write to the channel group's pins. A "1" indicates that the pin at the corresponding bit position should be set to the physical level "HIGH". A "0" indicates that the pin at the corresponding bit position should be set to the physical level "LOW". This mask is "right aligned", so bit position "0" in the mask is shifted to the left to match the rightmost ("least significant") pin in the channel group. Any bit representing a pin not included in the channel group or which does not physically exist will be ignored.
Return code	
-	-
Functional Description	
Writes the given level mask to the selected channel group.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different channel groups.	
Expected Caller Context	
> Task context	

Table 5-9 Dio_WriteChannelGroup

5.2.8 Dio_GetVersionInfo

Prototype	
void **Dio_GetVersionInfo** (P2VAR(Std_VersionInfoType, AUTOMATIC, DIO_APPL_DATA) versioninfo)	
Parameter	
versionInfo	Pointer to where to store the version information of this module.
Return code	
-	-

Functional Description
This function returns the version information of the module. The version information includes: > Module Id > Vendor Id > Software version numbers
Particularities and Limitations
> This function is synchronous. > This function is reentrant. > This function is configurable.
Expected Caller Context
> Task context

Table 5-10 Dio_GetVersionInfo

5.2.9 Dio_FlipChannel

Prototype	
void **Dio_FlipChannel** (Dio_ChannelType ChannelId)	
Parameter	
ChannelId	Symbolic channel name of a port pin as generated into Dio_Cfg.h.
Return code	
-	-
Functional Description	
Inverts the current PIN level	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different channels. > This function is configurable.	
Expected Caller Context	
> Task context	

Table 5-11 Dio_FlipChannel

5.3 Services used by DIO

In the following table services provided by other components, which are used by the DIO are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET (optional, configurable)	Det_ReportError

Table 5-12 Services used by the DIO

6 Configuration

6.1 Configuration Variants

The DIO supports the configuration variants

- `VARIANT-PRE-COMPILE`

The configuration classes of the DIO parameters depend on the supported configuration variants. For their definitions please see the VTTDio_bswmd.arxml file.

6.2 Configuration with DaVinci Configurator 5

The DIO module is configured with the help of the configuration tool DaVinci Configurator 5 (CFG5). The definition of each parameter is given in the corresponding BSWMD file.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
CANoe	Tool for simulation and testing of networks and electronic control units.
DaVinci Configurator	Configuration and generation tool for MICROSAR components

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DIO	Digital Input Output
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
VTT	vVIRTUALtarget

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com