User's Manual

AUTOSAR MCAL R4.0.3

User's Manual

DIO Driver Component Ver.1.0.4
Embedded User's Manual

Target Device:
RH850\P1x-C

Renesas Electronics
www.renesas.com

Rev.1.02 Jun 2017

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other disputes involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawing, chart, program, algorithm, application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics products.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.

   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (space and undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. When using the Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat radiation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions or failure or accident arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please ensure to implement safety measures to guard them against the possibility of bodily injury, injury or damage caused by fire, and social damage in the event of failure or malfunction of Renesas Electronics products, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures by your own responsibility as warranty for your products/system. Because the evaluation of microcomputer software alone is very difficult and not practical, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please investigate applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive carefully and sufficiently and use Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall not use Renesas Electronics products or technologies for (1) any purpose relating to the development, design, manufacture, use, stockpiling, etc., of weapons of mass destruction, such as nuclear weapons, chemical weapons, or biological weapons, or missiles (including unmanned aerial vehicles (UAVs)) for delivering such weapons, (2) any purpose relating to the development, design, manufacture, or use of conventional weapons, or (3) any other purpose of disturbing international peace and security, and you shall not sell, export, lease, transfer, or release Renesas Electronics products or technologies to any third party whether directly or indirectly with knowledge or reason to know that the third party or any other party will engage in the activities described above. When exporting, selling, transferring, etc., Renesas Electronics products or technologies, you shall comply with any applicable export control laws and regulations promulgated and administered by the governments of the countries asserting jurisdiction over the parties or transactions.
10. Please acknowledge and agree that you shall bear all the losses and damages which are incurred from the misuse or violation of the terms and conditions described in this document, including this notice, and hold Renesas Electronics harmless, if such misuse or violation results from your resale or making Renesas Electronics products available any third party.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

Abbreviations and Acronyms

Abbreviation / Acronym	Description
ANSI	American National Standards Institute
API	Application Programming Interface
AUTOSAR	AUTomotive Open System ARchitecture
CAN	Controller Area Network
DEM	Diagnostic Event Manager
DET/Det	Development Error Tracer
DIO	Digital Input Output
ECU	Electronic Control Unit
EEPROM	Electrical Erasable Programmable Read Only Memory
Id	Identifier
I/O	Input/Output
MCAL	MicroController Abstraction Layer
MCU	MicroController Unit
PWM	Pulse Width Modulation
RAM	Random Access Memory
ROM	Read Only Memory
RTE	Run Time Environment
SCI	Serial Communication Interface

Definitions

Term	Represented by
Port	Represents a whole configurable port on a microcontroller device.
Port Pin	Represents a single configurable input or output pin on a microcontroller device.
Sl. No.	Serial Number

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

The purpose of this document is to describe the information related to DIO Driver Component for Renesas P1x-C microcontrollers.

This document shall be used as reference by the users of DIO Driver Component. The system overview of complete AUTOSAR architecture is shown in the below Figure:

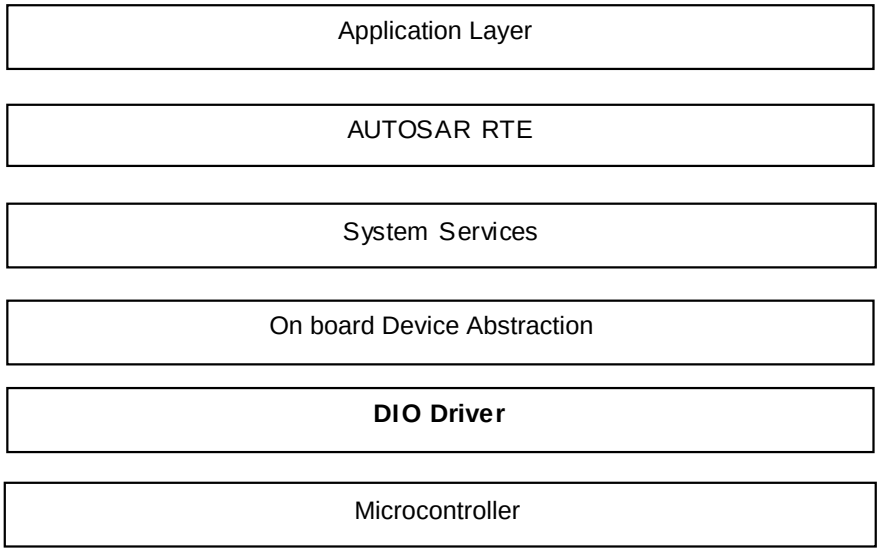

Figure 1-1 System Overview of AUTOSAR Architecture

The DIO Driver is part of the MCAL, the lowest layer of Basic Software in the AUTOSAR environment.

The Figure in the following page depicts the DIO Driver as part of layered AUTOSAR MCAL Layer:

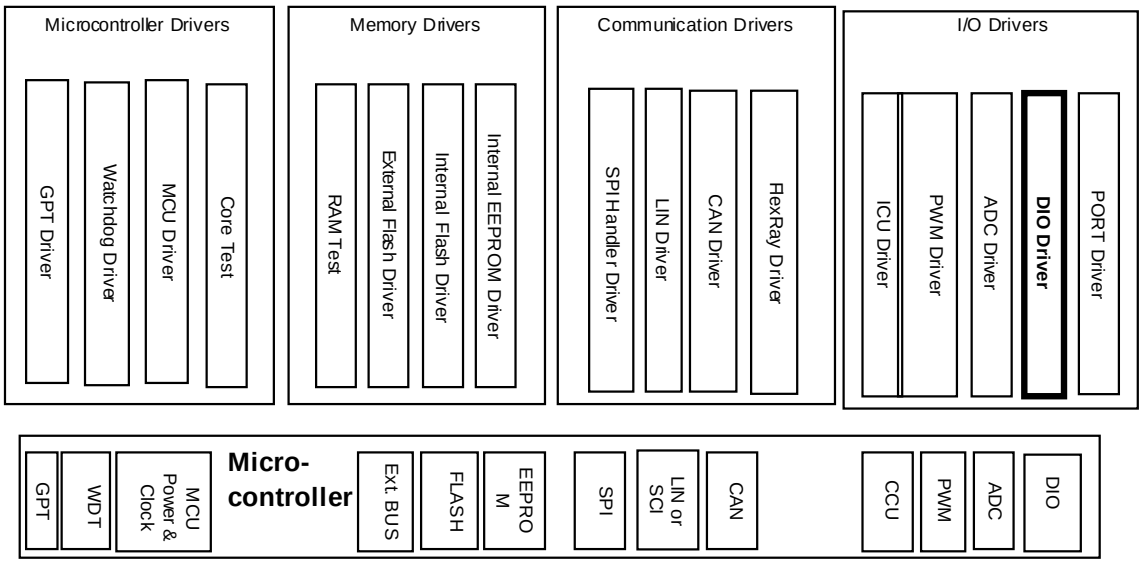

Figure 1-2 System Overview Of The DIO Driver In AUTOSAR MCAL Layer

The DIO Driver Component provides the service for reading and writing to Channels, ChannelGroups and Ports.

The DIO Driver Component comprises of two sections, that is, Embedded Software and the Configuration Tool to achieve scalability and configurability. The DIO Driver Component Code Generation Tool is a command line tool that accepts ECU configuration description file(s) as input and generates source and header file(s). The ECU configuration description is an XML file that contains information about the configuration for Port Pins. The tool generates Dio_PBcfg.c, Dio_Lcfg.c, Dio_Hardware.c, Dio_Hardware.h and Dio_Cfg.h, Dio_Cbk.h files.

1.1. Document Overview

The document has been segmented for easy reference. The table below provides the user with an overview of the contents of each section:

Section	Contents
Section 1 (Introduction)	This section provides an introduction and overview of DIO Driver Component.
Section 2 (Reference Documents)	This section lists the documents referred for developing this document.
Section 3 (Integration And Build Process)	This section explains the folder structure, Makefile structure for DIO Driver Component. This section also explains about the Makefile descriptions, Integration of DIO Driver Component with other components, building the DIO Driver Component along with a sample application.
Section 4 (Forethoughts)	This section provides brief information about the DIO Driver Component, the preconditions that should be known to the user before it is used, data consistency details and deviation list.
Section 5 (Architecture Details)	This section describes the layered architectural details of the DIO Driver Component.
Section 6 (Registers Details)	This section describes the register details of DIO Driver Component.
Section 7 (Interaction Between User And DIO Driver Component)	This section describes interaction of the DIO Driver Component with the upper layers.
Section 8 (DIO Driver Component Header And Source File Description)	This section provides information about the DIO Driver Component source files. This section also contains the brief note on the tool generated output file.
Section 9 (Generation Tool Guide)	This section provides information on the DIO Driver Component Generation Tool.
Section 10 (Application Programming Interface)	This section explains all the APIs provided by the DIO Driver Component.
Section 11 (Development And Production Errors)	This section lists the DET and DEM errors.
Section 12 (Memory Organization)	This section provides the typical memory organization, which must be met for proper functioning of component.
Section 13 (P1x-C Specific Information)	This section provides the P1x-C Specific Information.
Section 14 (Release Details)	This section provides release details with version name and base version.

Chapter 2 Reference Documents

Sl. No.	Title	Version
1.	AUTOSAR R4.0 Specification of DIO Driver (AUTOSAR_SWS_DIODriver.pdf)	2.5.0
2.	AUTOSAR BUGZILLA (http://www.autosar.org/bugzilla) Note: AUTOSAR BUGZILLA is a database, which contains concerns raised against information present in AUTOSAR Specifications.	-
3.	RH850/P1x-C Group User's Manual: Hardware (r01uh0517ej0120_rh850p1x-c_Open.pdf)	1.20
4.	Specification of Compiler Abstraction (AUTOSAR_SWS_CompilerAbstraction.pdf)	3.2.0
5.	Specification of Memory Mapping (AUTOSAR_SWS_MemoryMapping.pdf)	1.4.0
6.	Specification of Platform Types (AUTOSAR_SWS_PlatformTypes.pdf)	2.5.0

Chapter 3 Integration and Build Process

In this section, the folder structure of the DIO Driver Component is explained. Description of the makefiles along with samples is provided.

Remark The details about the C Source and Header files that are generated by the DIO Driver Generation Tool are mentioned in the “R20UT3640EJ0102-AUTOSAR.pdf”.

3.1 DIO Driver Component Makefile

The Makefile provided with the DIO Driver Component consists of the GNU Make compatible script to build the DIO Driver Component in case of any change in the configuration. This can be used in the upper level Makefile (of the application) to link and build the final application executable.

3.1.1 Folder Structure

The files are organized in the following folders:

Remark Trailing slash ‘\’ at the end indicates a folder

X1X\common_platform\modules\dio\src\Dio.c
\Dio_Version.c
\Dio_Ram.c

X1X\common_platform\modules\dio\include\Dio.h
\Dio_Debug.h
\Dio_PBTypes.h
\Dio_Ram.h
\Dio_Version.h
\Dio_RegWrite.h

X1X\ P1x-C \modules\dio\sample_application\make\ghs
\App_Dio_P1x-C_Sample.mak
\App_Dio_P1x-C_Sample.ld

X1X\ P1x-C\modules\dio\sample_application\<SubVariant>
\obj\<Compiler>

X1X\P1x-C \modules\dio\generator
\R403_DIO_ P1x-C_BSWMDT.arxml

X1X\ P1x-C \modules\dio\user_manual
(User manuals will be available in this folder).

Note: 1. <AUTOSAR_version> should be 4.0.3

2. <SubVariant> can be P1H-C, P1H-CE or P1M-C.

Chapter 4 Forethoughts

4.1. General

Following information will aid the user to use the DIO Driver Component software efficiently.

- At initialization state, DIO Driver does not initialize or write on any registers
- The DIO Driver does not enable or disable the ECU or Microcontroller power supply. The upper layer should handle this operation.
- Start-up code is not part of DIO Driver.
- DIO Driver does not provide any callback notification to upper layer.
- DIO Driver does not have any scheduled functions.
- DIO Driver supports the Read-back feature.
- The DIO Driver Component is Post-build time configurable for R4.0.3.
- To access the Hardware Registers, The MCAL user should use the Low Level Driver Layer. The MCAL user does not write or read any data directly from any register, but uses the AUTOSAR standard APIs provided by the MCAL Layer.
- When the channel is configured as Output, then the PPR register will reflect the Hardware pin level only when the Port Bi-Direction Control Register (PBDC) is enabled. Or else only the Port register value will be updated in the PPR register for the Output configured channels.

4.2. Preconditions

Following preconditions have to be adhered by the user, for proper functioning of the DIO Driver Component:

- The Dio_Cfg.h, Dio_PBcfg.c files generated by the DIO Driver Component Code Generation Tool have to be linked along with DIO Driver Component source files.
- File Dio_PBcfg.c generated by DIO Driver Component Generation Tool can be compiled and linked independently.
- The application has to be rebuilt, if there is any change in the Dio_Cfg.h and Dio_PBcfg.c file generated by the DIO Driver Component Generation Tool.
- Dio_Ram.c and Dio_Ram.h are used only for Autosar release 4.0.3.
- The DIO Driver Component needs to be initialized before accepting any API requests. Dio_Init should be called to initialize DIO Driver Component.
- The authorization of the user for calling the software triggering of a hardware reset is not checked in the DIO Driver. This will be the responsibility of the upper layer.
- The user should ensure that DIO Driver Component API requests are invoked with correct input arguments.

- Input parameters are validated only when the static configuration parameter DioDevErrorDetect is enabled. Application should ensure that the right parameters are passed while invoking the APIs when DioDevErrorDetect is disabled.
- A mismatch in the version numbers of header and the source files will result in a compilation error. User should ensure that the correct versions of the header and the source files are used.
- The DIO functions shall be called only after PORT Driver initialization, otherwise DIO functions will exhibit undefined behavior.
- User/Integrator should ensure that same Ports/Pins are configured for DIO Driver component.
- The user shall configure the exact Module Short Name Dio in configurations, as specified in config.xml file and the same shall be given in command line.
- Safety features related to Register_write verification are only available if 'DioWriteVerify' is enabled.

4.3. Data Consistency

To support the re-entrance and interrupt services, the AUTOSAR DIO component will ensure the data consistency while accessing its own RAM storage or hardware registers. The DIO component will use SchM_Enter_Dio_<Exclusive Area> and SchM_Exit_Dio_<Exclusive Area> functions. The SchM_Enter_Dio_<Exclusive Area> function is called before the data needs to be protected and SchM_Exit_Dio_<Exclusive Area>function is called after the data is accessed.

The following exclusive area along with scheduler services is used to provide data integrity for shared resources:

- DIO_REGISTER_PROTECTION

This function can be disabled by disabling the configuration parameter 'Dio_CriticalSectionProtection'.

If the 'DioCriticalSectionProtection' parameter is enabled then the critical section protection is applicable to the APIs in DIO Module. The details of applicable APIs are given below:

Table 4-1 DIO Driver Protected Resources List

API Name	Exclusive Area Type	Protected Resources
Dio_WritePort	DIO_REGISTER_PROTECTION	HW Registers: PMSR
Dio_WriteChannel	DIO_REGISTER_PROTECTION	HW Registers: PMSR
Dio_FlipChannel	DIO_REGISTER_PROTECTION	HW Registers: PMSR PNOT
Dio_WriteChannelGroup	DIO_REGISTER_PROTECTION	HW Registers: PMSR

API Name	Exclusive Area Type	Protected Resources
Dio_MaskedWritePort	DIO_REGISTER_PROTECTION	HW Registers: PMSR

Note: The highest measured duration of a critical section was 0.0724 micro seconds measured for Dio_WriteChannelGroup API

4.4. Deviation List

Table 4-2 DIO Driver Deviation List

Sl. No.	Description	AUTOSAR Bugzilla
1.	The API Dio_MaskedWritePort which is available in R3.2.2 is also added to R4.0.3 release.	-
2.	In case of Multiple configuration sets used, configuring different short name for the containers 'DioPort', 'DioChannel' and 'DioChannelGroup' across the configuration set shall result in generation error.	-
3.	In case of Multiple configuration sets used, configuring different number of ‘DioPort’ or 'DioChannel' or 'DioChannelGroup' containers across the configuration set shall result in generation error.	-

4.5. User mode and supervisor mode

The below table specifies the APIs which can run in user mode, supervisor mode or both modes.

Table 4-3 User mode and Supervisor mode details

Sl. No.	API Name	User Mode	Supervisor Mode	Known limitation in User mode
1.	Dio_Init	x	x	-
2.	Dio_ReadPort	x	x	-
3.	Dio_WritePort	x	x	-
4.	Dio_ReadChannel	x	x	-
5.	Dio_WriteChannel	x	x	-
6.	Dio_FlipChannel	x	x	-
7.	Dio_ReadChannelGroup	x	x	-
8.	Dio_WriteChannelGroup	x	x	-
9.	Dio_MaskedWritePort	x	x	-
10.	Dio_GetVersionInfo	x	x	-

Note: Implementation of Critical Section is not dependent on MCAL. Hence Critical Section is not considered to the entries for User mode in the above table.
The user can switch between user mode and supervisor mode during Enter/Exit critical section functions, so that these functions will work properly even though critical section protection is ON.

Chapter 5 Architecture Details

The overview of the AUTOSAR DIO software architecture is given in the following figure. The DIO Driver Component access the hardware features directly. The upper layers call the functionalities provided by DIO Driver Component:

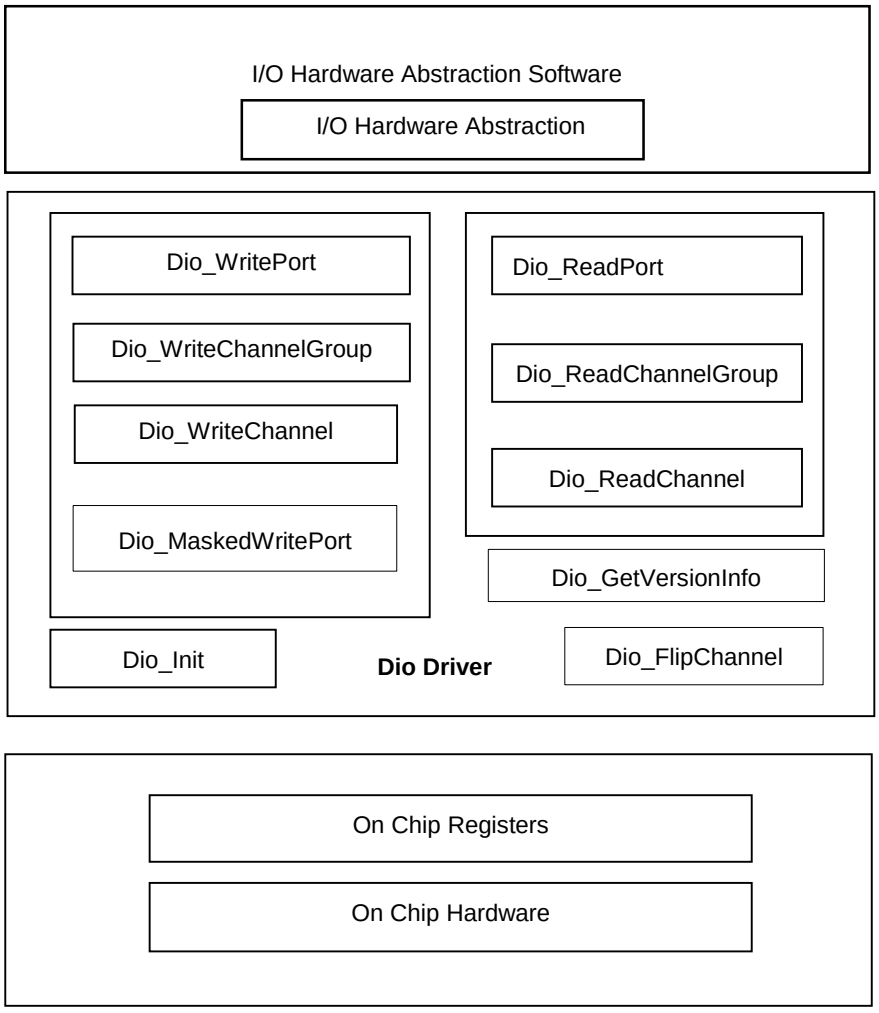

Figure 5-1 DIO Driver Architecture

DIO Driver Component:

The following figure shows the various functionalities as offered by the DIO Driver.

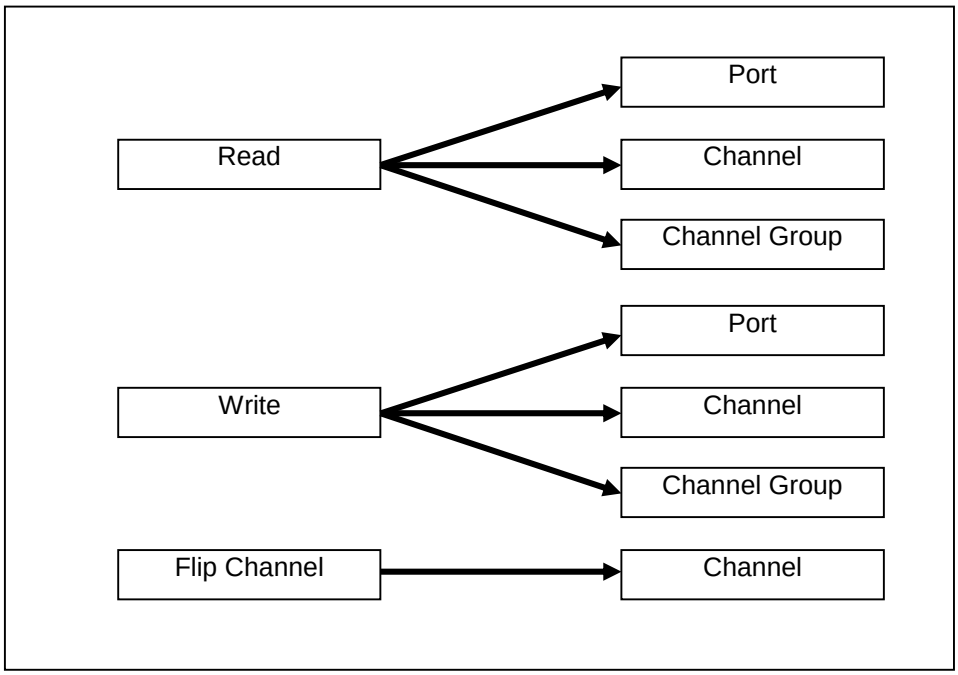

Figure 5-2 DIO Driver Services

The DIO Driver Component is divided into the following sub modules based on the functionality required:

- Port Read and Port Write
- Channel Read and Channel Write
- ChannelGroup Read and ChannelGroup Write
- Flip channel

DIO Read Port

The levels of all channels of a DIO Port will be read. A bit value '0' indicates the individual channels of the Port to physical STD_LOW, a bit value '1' indicates the individual channels of the port to physical STD_HIGH. The level of all channels of Port are read simultaneously.

DIO Read Channel

The level of a single DIO Channel will be read as either STD_LOW or STD_HIGH.

DIO Read ChannelGroup

The levels of all channels of a DIO Channel Group will be read. A bit value '0' indicates the corresponding pin to physical STD_LOW, a bit value '1' indicates the corresponding channel to physical STD_HIGH.

DIO Write Port

The levels of all channels of a Port will be set simultaneously without affecting the functionality of the input channels. A bit value '0' sets the corresponding channel to physical STD_LOW, a bit value '1' sets the corresponding channel to physical STD_HIGH.

DIO Write Channel

The level of a single DIO Channel is set STD_HIGH or STD_LOW.

DIO Write ChannelGroup

All adjoining subset of DIO channels of a port are set simultaneously. A bit value '0' sets the corresponding channel to physical STD_LOW, a bit value '1' sets the corresponding channel to physical STD_HIGH.

DIO Initialization

All the global variables of the DIO modules will be initialized.

DIO Flip Channel

The level of a single DIO channel will be set with compliment of current level that is, if current value is STD_HIGH then STD_LOW will be set and vice versa and then the level of a single DIO Channel will be read.

DIO GetVersionInfo

The Dio_GetVersionInfo function shall return the version information of this module. The version information includes module ID, Vendor ID and Vendor specific version numbers.

DIO MaskedWrite Port

The Dio_MaskedWritePort function shall set the specified value for the channels in the specified port if the corresponding bit in Mask is ‘1’.

Chapter 6 Registers Details

This section describes the register details of DIO Driver Component.

Table 6-1 Register Details

API Name	Registers	Configuration Parameter	Register Access R/W/RW	Macro/Variable
Dio_Init	-	-		-
Dio_ReadPort	PPRn	-	R	LddPortLevel
	JPPRn	-	R	LddPortLevel
Dio_WritePort	PSRn	-	R/W	Level
	JPSRn	-	R/W	Level
	PMSRn	-	R	LulPortModeLevel
	JPMSRn	-	R	LulPortModeLevel
Dio_ReadChannel	PPRn	DioChannelBitPosition	R	LddPortLevel
	JPPRn	DioChannelBitPosition	R	LddPortLevel
Dio_WriteChannel	PSRn	DioChannelBitPosition	R/W	LunPSRContent.ulLongWord
	JPSRn	DioChannelBitPosition	R/W	LunPSRContent.ulLongWord
	PMSRn	-	R	LulPortModeLevel
	JPMSRn	-	R	LulPortModeLevel
Dio_FlipChannel	PNOTn	DioChannelBitPosition	W	-
	JPNOTn	DioChannelBitPosition	W	-
	PPRn	DioChannelBitPosition	R	LddPortLevel
	JPPRn	DioChannelBitPosition	R	LddPortLevel
	PMSRn	-	R	U16pindirection
	JPMSRn	-	R	U16pindirection
Dio_ReadChannelGroup	PPRn	DioPortMask and DioPortOffset	R	LddPortLevel
	JPPRn	DioPortMask and DioPortOffset	R	LddPortLevel
Dio_WriteChannelGroup	PSRn	DioPortMask and DioPortOffset	R/W	LulPortModeLevel
	JPSRn	DioPortMask and DioPortOffset	R/W	LulPortModeLevel
	PMSRn	-	R	LulPortModeLevel
	JPMSRn	-	R	LulPortModeLevel
Dio_GetVersionInfo	-	-		-
Dio_MaskedWritePort	PSRn	-	R/W	-
	JPSRn	-	R/W	-
	PMSRn	-	R	LulPortModeLevel
	JPMSRn	-	R	LulPortModeLevel

Chapter 7 Interaction Between The User and DIO Driver Component

The details of the services supported by the DIO Driver Component to the upper layers users and the mapping of the channels to the hardware units is provided in the following sections:

7.1. Services Provided By DIO Driver Component to the User

The DIO Driver Component provides the following functionalities to the upper layers or users:

- To initialize the DIO module.
- To read the physical level value from DIO Port.
- To write specified value into given DIO Port.
- To read physical level value from DIO Channel.
- To write a specified value into the given DIO Channel.
- To read physical level value from DIO ChannelGroup.
- To write specified value into DIO ChannelGroup.
- To read the DIO Driver Component version information.
- To flip the level of the channel and return the level of the channel after flip.

Chapter 8 DIO Driver Component Header And Source File Description

This section explains the DIO Driver Component's source and header files. These files have to be included in the project application while integrating with other modules.

The C header file generated by DIO Driver Component Code Generation Tool:

- Dio_Cfg.h
- Dio_Hardware.h
- Dio_Cbk.h

The C source file generated by DIO Driver Component Code Generation Tool:

- Dio_PBcfg.c
- Dio_Lcfg.c
- Dio_Hardware.c

The DIO Driver Component C header files:

- Dio.h
- Dio_Debug.h
- Dio_PBTypes.h
- Dio_Ram.h
- Dio_Version.h
- Dio_RegWrite.h

The DIO Driver Component C source files:

- Dio.c
- Dio_Ram.c
- Dio_Version.c

The Stub C header files:

- Compiler.h
- Compiler_Cfg.h
- MemMap.h
- Platform_Types.h
- rh850_Types.h
- Det.h
- Rte.h
- SchM.h
- SchM_Dio.h
- Dem.h
- Dem_cfg.h

The description of the DIO Driver Component files is provided in the table below:

Table 8-1 Description Of The DIO Driver Component Files

File	Details
Dio_Cfg.h	This file is generated by the DIO Driver Component Code Generation Tool for DIO Driver Component pre-compile time parameters. This file contains macro definitions for the configuration elements. The macros and the parameters generated will vary with respect to the configuration in the input ARXML file.
Dio_Cbk.h	This file is generated by the DIO Driver Component Code Generation Tool for Prototype Declarations of Dio callback notification functions.
Dio_Hardware.h	This file is generated by the DIO Generation Tool include declaration of hardware registers specific to P1x-C DIO
Dio_PBcfg.c	This file contains post-build configuration data. The structures related to DIO Initialization are provided in this file. Data structures will vary with respect to parameters configured.
Dio_Lcfg.c	This file contains link-time configuration data. Data structures will vary with respect to parameters configured.
Dio_Hardware.c	This file is generated by the DIO Generation Tool include definition of hardware registers specific to P1x-C DIO
Dio.h	This file provides extern declarations for all the DIO Driver Component APIs. This file provides service Ids of APIs, DET Error codes and Global Data Types for DIO Driver component. This header file shall be included in other modules to use the features of DIO Driver Component.
Dio_Debug.h	This file provides Provision of global variables for debugging purpose.
Dio_RegWrite.h	This file is to have macro definitions for the registers write and verification.
Dio_PBTypes.h	This file contains the macros used internally by the DIO Driver Component code and the structure declarations related to hardware unit registers, HARDWARE unit, Channel configuration, Group configuration, Group RAM data and HARDWARE unit RAM data.
Dio_Ram.h	This file contains the extern declarations for the global variables that are defined in Dio_Ram.c file and the version information of the file.
Dio_Version.h	This file contains the macros of AUTOSAR version numbers of all modules that are interfaced to DIO.
Dio.c	This file contains the implementation of all DIO Driver Component APIs.
Dio_Version.c	This file contains the code for checking version of all modules that are interfaced to DIO.
Dio_Ram.c	This file contains the global variables used by DIO Driver Component.
Compiler.h	Provides compiler specific (non-ANSI) keywords. All mappings of keywords, which are not standardized, and/or compiler specific are placed and organized in this compiler specific header.
Compiler_Cfg.h	This file contains the memory and pointer classes.
MemMap.h	This file allows to map variables, constants and code of modules to individual memory sections. Memory mapping can be modified as per ECU specific needs.
Platform_Types.h	This file provides provision for defining platform and compiler dependent types.

File	Details
rh850_Types.h	This file provides macros to perform supervisor mode (SV) write enabled Register ICxxx and IMR register writing using OR/AND/Direct operation.
Det.h	This file is a stub for DET Component
Rte.h	This file is a stub for Rte Component
SchM.h	This file is a stub for Schm Component
SchM_Dio.h	Header file information for application.
Dem.h	This file is a stub for DEM component
Dem_cfg.h	This file contains the stub values for Dem_Cfg.h

Chapter 9 Generation Tool Guide

For more information on the MCAL Code Generator Tool, please refer “R20UT3640EJ0102-AUTOSAR.pdf” document.

Chapter 10 Application Programming Interface

This section explains the Data types and APIs provided by the DIO Driver Component to the Upper layers.

10.1. Imported Types

This section explains the Data types imported by the DIO Driver Component and lists its dependency on other modules.

10.1.1. Standard Types

In this section all types included from the Std_Types.h are listed:

Std_VersionInfoType

10.1.2. Other Module Types

DIO Driver Component module does not depend on other modules. Hence, no types from other modules are imported.

10.2. Type Definitions

This section explains the type definitions of DIO Driver Component according to AUTOSAR Specification.

10.2.1. Dio_PortType

Name:	Dio_PortType
Type:	uint8
Range:	0 to 255
Description:	Type definition for Dio_PortType used by Dio_ReadPort and Dio_WritePort.

10.2.2. Dio_ChannelType

Name:	Dio_ChannelType
Type:	uint8
Range:	0 to 255
Description:	Type definition for Dio_ChannelType used by Dio_ReadChannel and Dio_WriteChannel

10.2.3. Dio_PortLevelType

Name:	Dio_PortLevelType
Type:	uint16
Range:	0 to 65535
Description:	Type definition for Dio_PortLevelType used by Dio_ReadPort, Dio_WritePort, Dio_ReadChannelGroup and Dio_WriteChannelGroup.

10.2.4. Dio_ConfigType

Name:	Dio_ConfigType
Type:	Structure
Range:	Implementation specific.
Description:	This structure contains all post-build configurable parameters of the DIO driver. A pointer to this structure is passed to the DIO driver initialization function for configuration.

10.3. Function Definitions

This section explains the APIs provided by the DIO Driver Component.

Table 10-1 APIs Used in DIO Driver Component

	APIs of R4.0.3
1.	Dio_Init
2.	Dio_ReadPort
3.	Dio_WritePort
4.	Dio_ReadChannel
5.	Dio_WriteChannel
6.	Dio_FlipChannel
7.	Dio_ReadChannelGroup
8.	Dio_WriteChannelGroup
9.	Dio_MaskedWritePort
10.	Dio_GetVersionInfo

10.3.1. Dio_ReadChannel

Name:	Dio_ReadChannel		
Prototype:	FUNC(Dio_LevelType, DIO_PUBLIC_CODE) Dio_ReadChannel(Dio_ChannelType ChannelId)		
Service ID:	0x00		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Dio_ChannelType	ChannelId	0-255
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	Dio_LevelType	STD_HIGH The physical level of the corresponding Pin is STD_HIGH STD_LOW The physical level of the corresponding Pin is STD_LOW	
Description:	This service returns the value of the specified DIO Channel.		
Configuration Dependency:	None		
Preconditions:	At least one channel shall be configured for each configured port group. Different values shall be considered for the configuration parameter DioChannelBitPosition from DioChannel container		
	DIO Driver must be initialized.		

10.3.2. Dio_WriteChannel

Name:	Dio_WriteChannel		
Prototype:	FUNC(void, DIO_PUBLIC_CODE) Dio_WriteChannel (Dio_ChannelType ChannelId, Dio_LevelType Level)		
Service ID:	0x01		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Dio_ChannelType	ChannelId	0-255
	Dio_LevelType	Level	0-255
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This service writes the given value into the specified DIO Channel.		
Configuration Dependency:	None		

Preconditions:	At least one channel shall be configured for each configured port group. Different values shall be considered for the configuration parameter DioChannelBitPosition from DioChannel container
	DIO Driver must be initialized.

10.3.3. Dio_FlipChannel

Name:	Dio_FlipChannel		
Prototype:	FUNC(Dio_LevelType, DIO_PUBLIC_CODE) Dio_FlipChannel(Dio_ChannelType ChannelId)		
Service ID:	0x11		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Dio_ChannelType	ChannelId	0-255
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	Dio_LevelType	STD_HIGH: The physical level of the corresponding Pin is STD_HIGH. STD_LOW: The physical level of the corresponding Pin is STD_LOW.	
Description:	This service flip the level of a channel and return the level of the channel after flip.		
Configuration Dependency:	None		
Preconditions:	DIO Driver must be initialized.		
	At least one channel shall be configured for each configured port group. Different values shall be considered for DioChannelBitPosition configuration parameter from DioChannel container.		
	The precompile switch DioFlipChannelApi shall be enabled.		

10.3.4. Dio_ReadPort

Name:	Dio_ReadPort		
Prototype:	FUNC(Dio_PortLevelType, DIO_PUBLIC_CODE) Dio_ReadPort(Dio_PortType PortId)		
Service ID:	0x02		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Dio_PortType	PortId	0-255
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	Dio_PortLevelType	Level of all channels of given port	

Description:	This service returns the level of all channels of given Port.
Configuration Dependency:	None
Preconditions:	The configuration file shall be contained at least one 16-bit port and one less than 16-bit port depending upon the availability of ports from the particular controller
	DIO Driver must be initialized.

10.3.5. Dio_WritePort

Name:	Dio_WritePort		
Prototype:	FUNC(void, DIO_PUBLIC_CODE) Dio_WritePort (Dio_PortType PortId, Dio_PortLevelType Level)		
Service ID:	0x03		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Dio_PortType	PortId	0-255
	Dio_PortLevelType	Level	0-65535
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This service writes the specified level to all the channels in given Port.		
Configuration Dependency:	None		
Preconditions:	The configuration file shall be contained at least one 16-bit port and one less than 16-bit port depending upon the availability of ports from the particular controller		
	DIO Driver must be initialized.		

10.3.6. Dio_MaskedWritePort

Name:	Dio_MaskedWritePort		
Prototype:	FUNC(void, DIO_PUBLIC_CODE) Dio_MaskedWritePort (Dio_PortType PortId, Dio_PortLevelType Level, Dio_PortLevelType Mask)		
Service ID:	0x13		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Dio_PortType	PortId	0-255
	Dio_PortLevelType	Level	0-65535
	Dio_PortLevelType	Mask	0-65535

Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This service writes the specified level to all the channels that are Masked in given Port.		
Configuration Dependency:	None		
Preconditions:	DIO Driver must be initialized.		
	The configuration file shall be contained at least one 16-bit port and one less than 16-bit port depending upon the availability of ports from the particular controller		
	The precompile switch DioMaskedWritePortApi shall be enabled.		

10.3.7. Dio_ReadChannelGroup

Name:	Dio_ReadChannelGroup		
Prototype:	FUNC(Dio_PortLevelType, DIO_PUBLIC_CODE) Dio_ReadChannelGroup (P2CONST(Dio_ChannelGroupType, DIO_VAR, DIO_CONFIG_CONST) ChannelGroupIdPtr)		
Service ID:	0x04		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Dio_ChannelGroupType	ChannelGroupIdPtr	NA
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	Dio_PortLevelType	Level of a subset of the adjoining bits of a port	
Description:	This Service reads a subset of the adjoining bits of a port.		
Configuration Dependency:	None		
Preconditions:	DIO Driver must be initialized.		
	At least one channel group shall be configured for each configured port group. Different values shall be considered for DioPortMask and DioPortOffset configuration parameters from DioChannelGroup container.		

10.3.8. Dio_WriteChannelGroup

Name:	Dio_WriteChannelGroup
Prototype:	FUNC(void, DIO_PUBLIC_CODE) Dio_WriteChannelGroup (P2CONST(Dio_ChannelGroupType, DIO_VAR, DIO_CONFIG_CONST) ChannelGroupIdPtr, Dio_PortLevelType Level)
Service ID:	0x05

Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	Dio_ChannelGroupType	ChannelGroupIdPtr	NA
	Dio_PortLevelType	Level	0-65535
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This service writes specified level to the ChannelGroup		
Configuration Dependency:	None		
Preconditions:	DIO Driver must be initialized.		
	At least one channel group shall be configured for each configured port group. Different values shall be considered for DioPortMask and DioPortOffset configuration parameters from DioChannelGroup container.		

10.3.9. Dio_GetVersionInfo

Name:	Dio_GetVersionInfo		
Prototype:	FUNC(void, DIO_PUBLIC_CODE)Dio_GetVersionInfo (P2VAR(Std_VersionInfoType, AUTOMATIC, DIO_APPL_DATA)versioninfo)		
Service ID:	0x12		
Sync/Async:	Synchronous		
Reentrancy:	Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	None	NA	NA
Parameters InOut:	None	NA	NA
Parameters out:	Std_VersionInfoType	versioninfo	Pointer to where to store the version information of this module.
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This service flip the level of a channel and return the level of the channel after flip.		
Configuration Dependency:	None		
Preconditions:	The precompile switch DioVersionInfoApi shall be enabled.		

10.3.10. Dio_Init

Name:	Dio_Init		
Prototype:	FUNC(void, DIO_PUBLIC_CODE) Dio_Init (P2CONST(Dio_ConfigType, DIO_VAR, DIO_APPL_CONST) ConfigPtr)		
Service ID:	0x10		
Sync/Async:	Synchronous		
Reentrancy:	Non-Reentrant		
Parameters In:	**Type**	**Parameter**	**Value/Range**
	ConfigPtr	Dio_ConfigType	Pointer to post-build configuration data
Parameters InOut:	None	NA	NA
Parameters out:	None	NA	NA
Return Value:	**Type**	**Possible Return Values**	
	None	NA	
Description:	This service initialize the DIO driver.		
Configuration Dependency:	None		
Preconditions:	None		

Chapter 11 Development And Production Errors

In this section the development errors that are reported by the DIO Driver Component are tabulated. The development errors will be reported only when the pre compiler option DIO_DEV_ERROR_DETECT is enabled in the configuration. The DIO Driver Component does not support any production errors.

11.1. Dio Driver Component Development Errors

The following contains the DET errors that are reported by DIO Driver Component. These errors are reported to Development Error Tracer Module when the DIO Driver Component APIs are invoked with wrong input parameters or without initialization of the driver.

Table 11-1 DET Errors of DIO Driver Component

Sl. No.	**1**
Error Code	DIO_E_PARAM_INVALID_CHANNEL_ID
Related APIs	Dio_ReadChannel, Dio_WriteChannel and Dio_FlipChannel
Source of Error	When the above mentioned APIs are invoked with invalid Channel ID
Sl. No.	**2**
Error Code	DIO_E_PARAM_CONFIG
Related API	Dio_Init
Source of Error	When the above mentioned API is invoked with NULL parameter.
Sl. No.	**3**
Error Code	DIO_E_PARAM_INVALID_PORT_ID
Related APIs	Dio_ReadPort, Dio_WritePort, Dio_MaskedWritePort, Dio_WriteChannelGroup, Dio_WriteChannel and Dio_FlipChannel
Source of Error	When the above mentioned APIs are invoked with invalid Port ID
Sl. No.	**4**
Error Code	DIO_E_PARAM_INVALID_GROUP
Related APIs	Dio_ReadChannelGroup and Dio_WriteChannelGroup
Source of Error	When the above mentioned APIs are invoked with invalid ChannelGroup ID
Sl. No.	**5**
Error Code	DIO_E_PARAM_POINTER
Related API	Dio_GetVersionInfo, Dio_ReadChannelGroup and Dio_WriteChannelGroup
Source of Error	When the above mentioned API is invoked with the NULL parameter.
Sl. No.	**6**
Error Code	DIO_E_INVALID_DATABASE
Related API	Dio_Init
Source of Error	When the above mentioned API is called without a database or invalid database
Sl. No.	**7**
Error Code	DIO_E_UNINIT

Related APIs	Dio_ReadChannel, Dio_WriteChannel, Dio_ReadPort, Dio_WritePort, Dio_ReadChannelGroup, Dio_WriteChannelGroup, Dio_FlipChannel and Dio_MaskedWritePort
Source of Error	When the above mentioned APIs are invoked without module initialization.

11.2. Dio Driver Component Production Errors

The following table contains the DEM errors that are reported by DIO software component.

Sl. No.	**1**
Error Code	DIO_E_REG_WRITE_VERIFY
Related API(s)	Dio_WritePort, Dio_WriteChannel, Dio_FlipChannel, Dio_WriteChannelGroup, Dio_MaskedWritePort
Source of Error	When register write-verify fails.

Chapter 12 Memory Organization

Following picture depicts a typical memory organization, which shall be met for proper functioning of DIO Driver Component software.

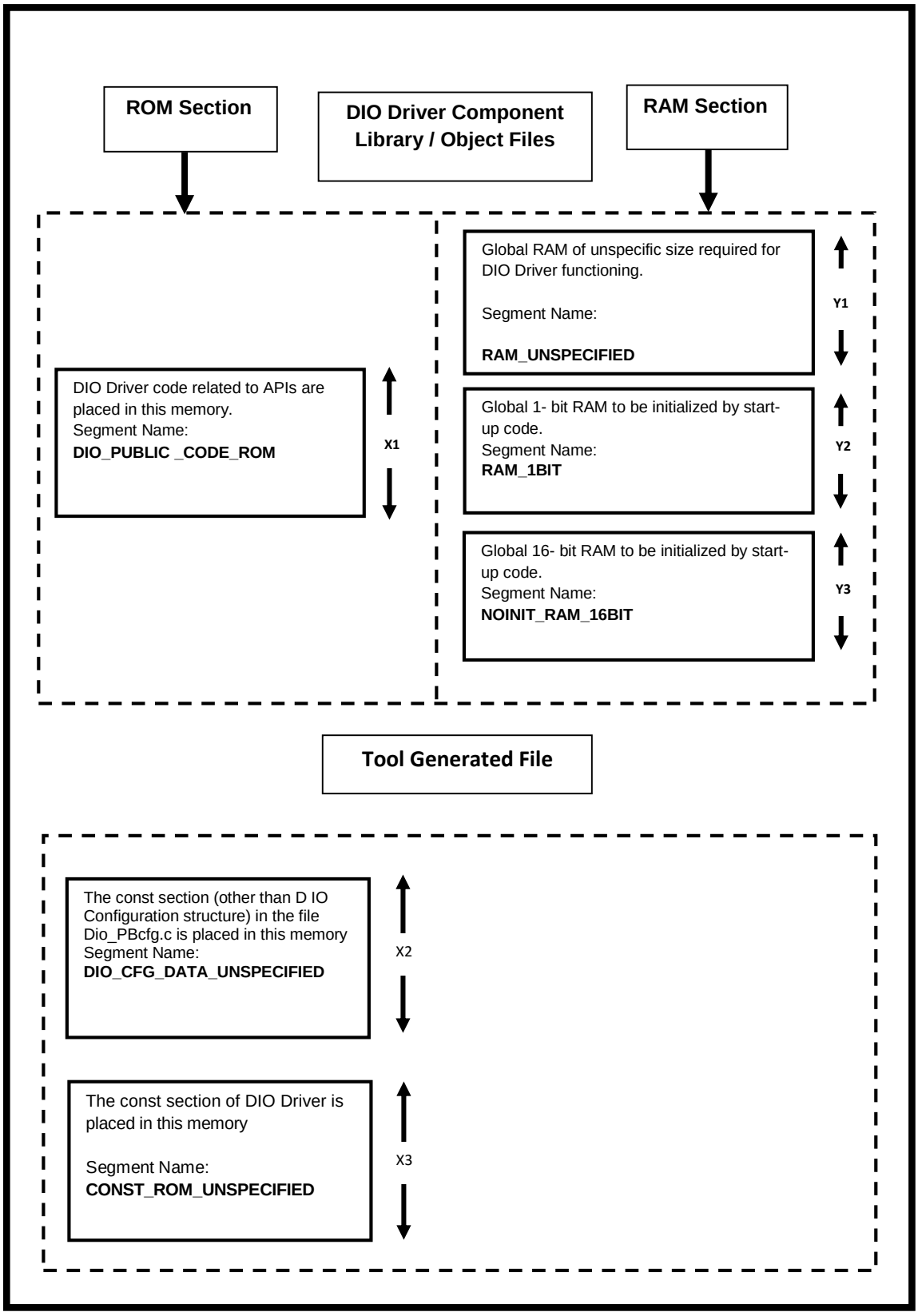

Figure 12-1 DIO Driver Component Memory Organization

ROM Section (X1, X2 and X3):

DIO_PUBLIC_CODE_ROM (X1): API(s) of DIO Driver Component, which can be located in code memory.

DIO_CFG_DATA_UNSPECIFIED (X2): This section consists of DIO Driver Component constant configuration structures. This can be located in code memory.

CONST_ROM_UNSPECIFIED (X3): The constant section of DIO Driver Component code that can be located in code memory.

RAM Section (Y1, Y2 and Y3):

RAM_UNSPECIFIED (Y1): This section consists of the global RAM variables that are initialized by start-up code and used internally by DIO software component and other software components. The specific sections of respective software components will be merged into this RAM section accordingly.

RAM_1BIT (Y2): This section consists of the global RAM variables of 1-bit size that are initialized by start-up code and used internally by DIO Driver Component and other software components. The specific sections of respective software components will be merged into this RAM section accordingly. This can be located in data memory.

NOINIT_RAM_16BIT (Y3): This section consists of the global RAM variables of 16-bit size that are used internally by DIO Driver Component. This can be located in data memory.

Notes:

- X1, Y1, Y2 and Y3 pertain to only DIO Driver Component and do not include memory occupied by Dio_PBcfg.c files generated by DIO Driver Component Generation Tool.
- User must ensure that none of the memory areas overlap with each other. Even ‘debug’ information should not overlap.

Chapter 13 P1x-C Specific Information

P1x-C supports following devices:

R7F701370A(CPU1(PE1)), R7F701371(CPU1(PE1)),
R7F701372(CPU1(PE1)), R7F701373, R7F701374

13.1. Interaction between the User and DIO Driver Component

The details of the services supported by the DIO Driver Component to the upper layers users and the mapping of the channels to the hardware units is provided in the following sections:

13.1.1. Parameter Definition File

Parameter definition files support information for P1x-C

Table 13-1 PDF information for P1x-C

PDF Files	Devices Supported
R403_DIO_P1X-C_70A_71_72.arxml	701370A(CPU1(PE1)), 701371(CPU1(PE1)), 701372(CPU1(PE1))
R403_DIO_P1X-C_73.arxml	701373
R403_DIO_P1X-C_74.arxml	701374

13.2. Sample Application

13.2.1 Sample Application Structure

The Sample Application is provided as reference to the user to understand the method in which the DIO APIs can be invoked from the application.

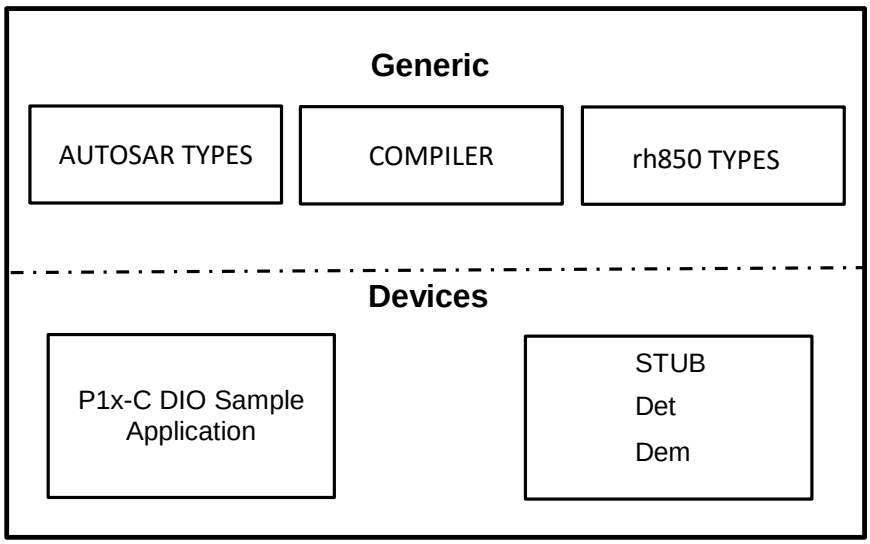

Figure 13-1 Overview of DIO Driver Sample Application

The Sample Application of the P1x-C is available in the path

X1X/ P1x-C /modules/dio/sample_application

The Sample Application consists of the following folder structure

X1X/ P1x-C/modules/dio/definition/ AUTOSAR_version /<Subvariant>/
R403_DIO_P1X-C_70A_71_72.arxml

X1X/ P1x-C/modules/dio/definition/ AUTOSAR_version /<Subvariant>/
R403_DIO_P1X-C_73.arxml

X1X/ P1x-C/modules/dio/definition/ AUTOSAR_version /<Subvariant /
R403_DIO_P1X-C_74.arxml

X1X/ P1x-C /modules/dio/sample_application/<Subvariant>
/<AUTOSAR_version>
/src/Dio_PBcfg.c
/src/Dio_Lcfg.c
/inc/Dio_Cfg.h
/inc/Dio_Cbk.h
/config/App_DIO_P1x-C_701370A_Sample.arxml
/config/App_DIO_P1x-C_701372_Sample.arxml
/config/App_DIO_P1x-C_701371_Sample.arxml
/config/App_DIO_P1x-C_701373_Sample.arxml
/config/App_DIO_P1x-C_701374_Sample.arxml

In the Sample Application all the DIO APIs are invoked in the following sequence:

- The API Dio_GetVersionInfo is invoked to get the version of the DIO Driver module with a variable of Std_VersionInfoType, after the call of this API the passed parameter will get updated with the DIO Driver version details.
- The API Dio_Init is invoked with a valid database address for the proper initialization of the DIO Driver, all the DIO Driver control registers and RAM variables will get initialized after this API is called.
- The API Dio_WriteChannel is invoked to set the required level of a particular channel. It sets the output level of the channel if that particular channel is configured in output mode. If channel is configured for input mode, Dio_WriteChannel has no effect.
- The API Dio_ReadChannel reads the actual physical level of the required channel. The level read for the input channel is actual physical level of that channel if input buffer for that channel is enabled. The level read for the output channel is actual physical level of that channel if bidirectional control for that channel is enabled.
- The API Dio_FlipChannel reads the level of the channel and invert it, then write the inverted level to the channel if the channel is configured as output channel and the function shall have no influence on the physical output if the channel is configured as input channel.
- The API Dio_WritePort is invoked to simultaneously set the required

levels to all channels of a port. It sets the output level of the channels which are configured in output mode, for the channels which are configured in input mode, Dio_WritePort has no effect.

- The API Dio_ReadPort reads the actual physical level of all the port pins of a required port. The level read for the input port pin is actual physical level of that port pin if input buffer for that port pin is enabled. The level read for the output port pin is actual physical level of that port pin if bidirectional control for that port pin is enabled.
- The API Dio_WriteChannelGroup is invoked to simultaneously set the required levels to group of channels of a port. It sets the output level of the channels which are configured in output mode, for the channels which are configured in input mode, Dio_WriteChannelGroup has no effect.
- The API Dio_ReadChannelGroup reads the actual physical level of channel group of a required port. The level read for the input channel group is actual physical level of that channel group if input buffer for that channel group is enabled. The level read for the output channel group is actual physical level of that channel group if bidirectional control for that channel group is enabled.

- The API Dio_MaskedWritePort provides service to set value of given port with required mask. The Dio_MaskedWritePort function shall set the

- The specified value for the channel in specified port if the corresponding bit in mask is ‘1’.

13.2.2 Building Sample Application

13.2.2.1 Configuration Example

This section contains the typical configuration which is used for measuring RAM/ROM consumption, stack depth and throughput details.

13.2.2.2 Debugging the Sample Application

GNU Make utility version 3.81 or above must be installed and available in the path as defined by the environment user variable “GNUMAKE” to complete the build process using the delivered sample files.

Open a Command window and change the current working directory to “make” directory present as mentioned in below path:

“X1X\P1x-C\common_family\ Sample_Application\<Compiler>”

Now execute batch file SampleApp.bat with following parameters:

SampleApp.bat dio <Device_name>

After this, the tool output files will be generated with the configuration as mentioned in the path:

- “X1X\P1x -C\modules\dio\sample_application\<SubVariant> \<AUTOSAR_version> \config”
- After this, all the object files, map file and the executable file Sample.out will be available in the output folder (“X1X\P1x-

C\modules\dio\sample_application\<SubVariant>\obj\<Compiler>" in this case).

- The executable can be loaded into the debugger and the sample application can be executed.

13.3. Memory and Throughput

Typical DIO configuration

- DET OFF
- All other Pre-Compile Settings ON
- 2 DioPorts
- 2 DioChannels
- 2 DioChannelGroups

13.3.1 ROM/RAM Usage

The details of memory usage for the typical configuration provided in Section Section Typical DIO configuration

Table 13-1 ROM/RAM Details without DET

Sl. No.	ROM/RAM	Segment Name	Size in bytes for 701372 (CPU1(PE1))
1.	ROM	DIO_PUBLIC_CODE_ROM	820
		CONST_ROM_UNSPECIFIED	44
		DIO_CFG_DATA_UNSPECIFIED	60
2.	RAM	RAM_1BIT	0
		NOINIT_RAM_16BIT	2
		RAM_UNSPECIFIED	4

Table 13-2 ROM/RAM Details with DET

Sl. No.	ROM/RAM	Segment Name	Size in bytes for 701372 (CPU1(PE1))
1.	ROM	DIO_PUBLIC_CODE_ROM	1438
		CONST_ROM_UNSPECIFIED	44
		DIO_CFG_DATA_UNSPECIFIED	60
2.	RAM	RAM_1BIT	1
		NOINIT_RAM_16BIT	2
		RAM_UNSPECIFIED	4

13.3.2 Stack Depth

The worst-case stack depth for DIO Driver Component is 28 bytes for the typical configuration provided in the Section Typical DIO configuration.

13.3.3 Throughput Details

The throughput details of the APIs with DET disabled for the configuration mentioned in the Section Typical DIO configuration is as follows. The clock frequency used to measure the throughput is 160MHz CPU Clock and 80MHZ PCLK for all APIs.

Table 13-3 Throughput Details of the APIs

Sl. No.	API Name	Throughput in microseconds for 701372 (CPU1(PE1))	Remarks
1.	Dio_Init	0.250	-
2.	Dio_WriteChannel	1.312	-
3.	Dio_ReadChannel	0.562	-
4.	Dio_WritePort	1.187	-
5.	Dio_ReadPort	0.425	-
6.	Dio_WriteChannelGroup	1.312	-
7.	Dio_ReadChannelGroup	0.525	-
8.	Dio_MaskedWritePort	1.187	-
9.	Dio_FlipChannel	1.475	-
10.	Dio_GetVersionInfo	0.137	-

13.4. Critical Section Details

The critical section throughput details are listed below. The clock frequency used to measure the throughput is 160MHz for all APIs.

Table 13-4 Critical Section Throughput Details of the APIs

Sl. No.	API Name	Critical section throughput in microseconds in GHS for 701372 (CPU1(PE1))	Remarks
1.	Dio_WritePort	0. 0625	-
2.	Dio_WriteChannel	0. 0655	-
3.	Dio_FlipChannel	0. 064	-
4.	Dio_WriteChannelGroup	0. 0725	-
5.	Dio_MaskedWritePort	0. 0705	-

Chapter 14 Release Details

Embedded DIO Driver Software

Version: 1.0.4

Revision History

Sl.No.	Description	Version	Date
1.	Initial Version	1.0.0	04-Aug-2015
2.	The following changes are made: 1. Sections13.2.1 is updated for Sample Application configuration details. 2. R number is added in the last page 3. Added DET error for Dio_GetVersionInfo in Chapter 11 4. Section 13.3 Memory and Throughput updated. 5. Section 4.2 is updated with note about the user Configuration of Module Short Name.	1.0.1	30-Mar-2016
3.	The following changes are made: 1. Chapter 2, Updated versions of References documents. 2. Chapter 3, section 3.1.1 is updated for new devices. 3. Updated section 4.2 Preconditions. 4. Updated section 4.4 Deviation list. 5. Section 4.5 updated with a note regarding critical section. 6. Table 6-1 in Chapter 6 updated 7. Chapter 8 and Table 8-1 updated with the stub files. 8. Chapter 9, Updated R number 9. Section 13.1 Compiler, Linker and Assembler removed 10. Critical section added in Chapter 13.1 11. Chapter 13.3 'Memory and Throughput' is updated. 12. Updated section 13.2, removed reference to .one and .html files 13. Updated Chapter 14 'Release Details' 14. Chapter 13, Added Processor name along with Device variants 15. Updated copyright information. 16. Updated section 11.1. 17. Added section 11.2	1.0.2	23-Feb-2017
4.	The following changes are made: 1. Updated Chapter 12 to remove unused memory section, DIO_CFG_DBTOC_UNSPECIFIED and added memory section, CONST_ROM_UNSPECIFIED. 2. Updated Section 13.2.1 and 13.2.2 with latest memory consumption and stack depth details. 3. Copyright information and notice are changed. 4. Added note in section 4.5 5. Added function definitions of Autosar APIs in section 10.3 6. Added Section 13.1, Interaction between the User and DIO Driver Component. 7. Path for Parameter definition updated in section 13.2.1. 8. Updated section 13.3.	1.0.3	29-May-2017
5.	The following changes are made: 1. Path of the PDF and Sample Application is updated in section 13.2. 2. Updated section 13.3 with latest memory consumption, stack depth and throughput details. 3. Updated the R number of the Tool User Manual in Chapter 3 and Chapter 9. 4. Software version in the chapter 14 is updated.	1.0.4	16-Jun-2017

AUTOSAR MCAL R4.0.3 User's Manual
DIO Driver Component Ver.1.0.4

Publication Date: Rev.1.02, June 16, 2017

Published by: Renesas Electronics Corporation

RENESAS

SALES OFFICES Renesas Electronics Corporation http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

AUTOSAR MCAL R4.0.3

User's Manual

Renesas Electronics Corporation

R20UT3639EJ0102