User's Manual

AUTOSAR MCAL R4.0.3
User's Manual

DIO Driver Component Ver.1.0.6

Embedded User's Manual

Target Device:
RH850/P1x

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics
www.renesas.com

Rev 0.02 May 2015

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of 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. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. 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 of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by

   you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may 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.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: "Standard", "High Quality", and "Specific". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as "Specific" without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as "Specific" or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is "Standard" unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   "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.

   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti- crime systems; safety equipment; and medical equipment not specifically designed for life support.

   "Specific": Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its 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 be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a

   Renesas Electronics product, 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. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be 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, or if you have any other inquiries.

(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
ADC	Analog Digital Converter
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
LIN	Local Interconnnect Network
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
SPI	Serial Peripheral Interface
WDT	WatchDog Timer

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 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:

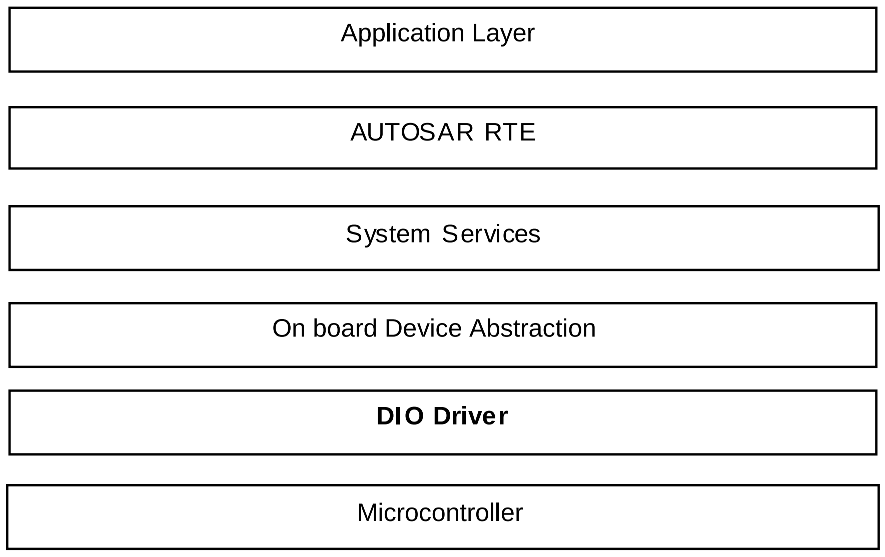

Figure 1-1 System Overview Of AUTOSAR Architecture

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:

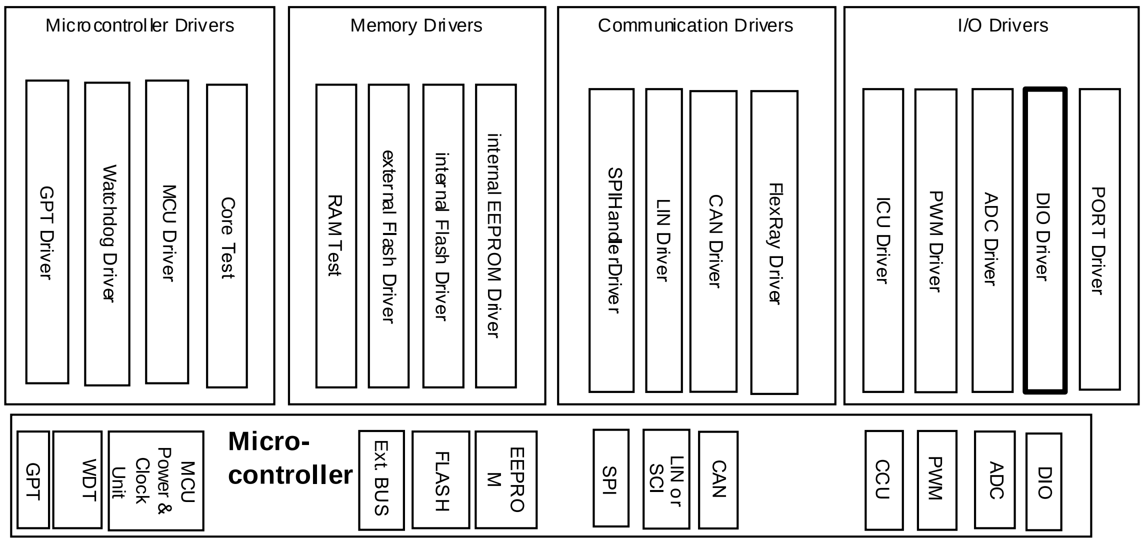

Figure 1-2 System Overview Of The DIO Driver In AUTOSAR MCAL Layer

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) Dio_PBcfg.c and Dio_Cfg.h respectively. The ECU configuration description is an XML file that contains information about the configuration for Port Pins.

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 (Register 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 (P1M Specific Information)	This section provides the P1M 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_SWS_DIODriver.pdf	2.5.0
2.	r01uh0436ej0070_rh850p1x.pdf	0.70
3.	AUTOSAR_SWS_CompilerAbstraction.pdf	3.2.0
4.	AUTOSAR_SWS_MemoryMapping.pdf	1.4.0
5.	AUTOSAR_SWS_PlatformTypes.pdf	2.5.0
6.	AUTOSAR_BSW_MakefileInterface.pdf	0.3
7.	AUTOSAR BUGZILLA (http://www.autosar.org/bugzilla) Note: AUTOSAR BUGZILLA is a database, which contains concerns raised against information present in AUTOSAR Specifications.	-

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 in this section.

Remark The details about the C Source and Header files that are generated by the DIO Driver Generation Tool are mentioned in the DIO Driver Component Generation Tool User's Manual.

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_LTTypes.h
                                       \Dio_Ram.h
                                       \Dio_Version.h

X1X\P1x\modules\dio\sample_application\<SubVariant>\make\ghs
                                           \App_DIO_P1M_Sample.mak

X1X\P1x\modules\dio\sample_application\ <SubVariant>

                                      \obj\<Compiler>
```

X1X\P1x\modules\dio\generator

\R403_DIO_P1x_BSWMDT.arxml

X1X\common_platform\modules\dio\generator\Dio_X1x.exe

X1X\P1x\common_family\generator\P1x_translation.h

\Sample_Applications_P1x.trxml

X1X\P1x\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 P1M

3. <Device_name> can be 701304, 701305, 701310, 701311, 701312, 701313, 701314, 701315, 701318, 701319, 701320, 701321, 701322 and 701323.
4. <Compiler> can be ghs

Chapter 4 Forethoughts

4.1. General

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

- 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 implemented by the DIO Driver.
- DIO Driver does not implement any callback notification functions.
- DIO Driver does not implement any scheduled functions.
- DIO Driver supports the Readback feature.
- The DIO Driver Component is Postbuild time configurable for R4.0.3.
- The file Interrupt_VectorTable.c provided is just a Demo and not all interrupts will be mapped in this file. So the user has to update the Interrupt_VectorTable.c as per his configuration.

4.2. Preconditions

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

- The Dio_Cfg.h and 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.

4.3. Data Consistency

To support the re-entrance services, the AUTOSAR DIO module will ensure the data consistency while accessing its own RAM storage or hardware registers. The DIO module 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 resource:

- REGISTER_PROTECTION

The functions SchM_Enter_Dio_<Exclusive Area> and SchM_Exit_Dio_<Exclusive Area> can be disabled by disabling the configuration parameter “DioCriticalSectionProtection”.

4.4. Deviation List

Table 4-1 Driver Deviation List Of DIO module

Sl. No.	Description AUTOSAR	Bugzilla
1	The API DIO_GetVersionInfo is implemented as macro without DET error DIO_E_PARAM_POINTER.	-
2	The API Dio_MaskedWritePort which is available in R3.2.2 is also added to R4.0.3 release.	-
3.	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.	-
4.	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 4-2 User mode and Supervisor mode details

Sl.No.	List of API'S	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	-

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:

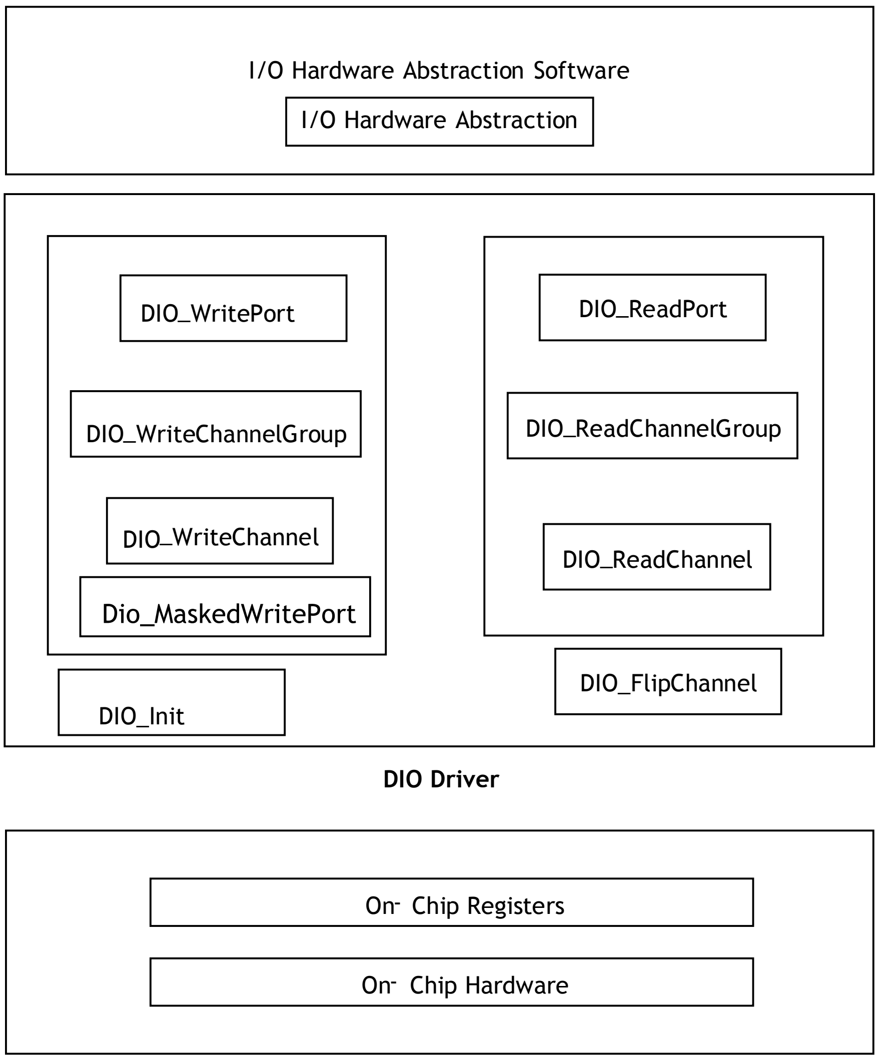

Figure 5-1 DIO Driver Architecture

DIO Driver Component:

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

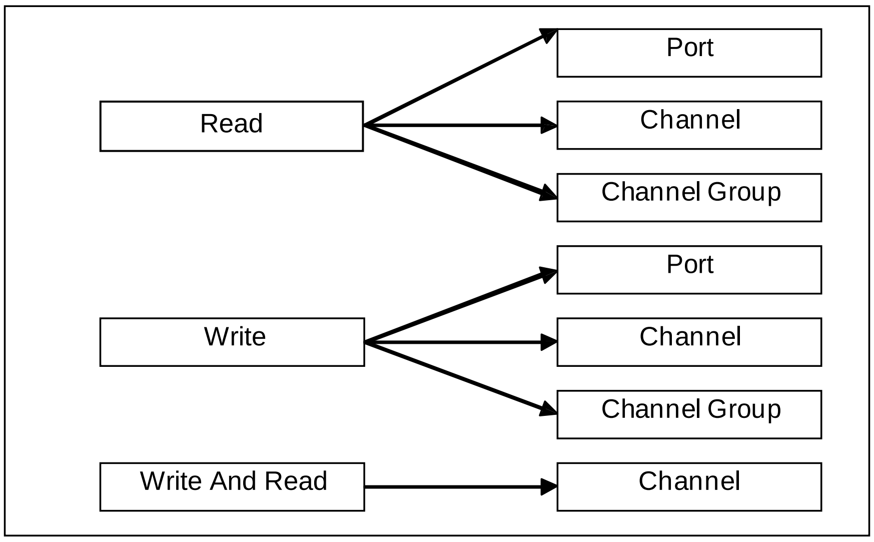

Figure 5-2 DIO Driver Services

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

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	Config Parameter	Macro/Variable
Dio_Init	-	-	-
Dio_ReadPort	PPRn	-	LddPortLevel
	JPPRn	-	LddPortLevel
	APPRn	-	LddPortLevel
	IPPRn	-	LddPortLevel
Dio_WritePort	PSRn	-	Level
	JPSRn	-	Level
	APSRn	-	Level
	PMSRn	-	LddPortModeLevel
Dio_ReadChannel	PPRn	-	LddPortLevel
	JPPRn	-	LddPortLevel
	APPRn	-	LddPortLevel
	IPPRn	-	LddPortLevel
Dio_WriteChannel	PSRn	-	LunPSRContent.ulLongWord
	JPSRn	-	LunPSRContent.ulLongWord
	APSRn	-	LunPSRContent.ulLongWord
	PMSRn	-	LddPortModeLevel
Dio_FlipChannel	PNOTn	DioChannelBitPosition	-
	JPNOTn	DioChannelBitPosition	-
	APNOTn	DioChannelBitPosition	-
	PPRn	-	LddPortLevel
	JPPRn	-	LddPortLevel
	APPRn	-	LddPortLevel
	PMSRn	-	LddPortModeLevel
Dio_ReadChannelGroup	PPRn	-	LddPortLevel
	JPPRn	-	LddPortLevel
	APPRn	-	LddPortLevel
	IPPRn	-	LddPortLevel
Dio_WriteChannelGroup	PSRn	-	LunPSRContent.ulLongWord
	JPSRn	-	LunPSRContent.ulLongWord
	APSRn	-	LunPSRContent.ulLongWord
	PMSRn	-	LddPortModeLevel
Dio_GetVersionInfo	-	-	-
Dio_MaskedWritePort	PSRn	-	-
	JPSRn	-	-
	APSRn	-	-

	PMSRn	-	LddPortModeLevel

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 write specified value with a specified mask into given DIO Port.
- To flip the level of a 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

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

- Dio_PBcfg.c

The DIO Driver Component C header files:

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

The DIO Driver Component C source files:

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

Note : Dio_LTTypes.h is not applicable for AUTOSAR release 4.0.3.

The port specific C header files:

- Compiler.h
- Compiler_Cfg.h
- MemMap.h
- Platform_Types.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_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.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_LTTypes.h	This file contains AUTOSAR DIO link time parameters.
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.

Chapter 9 Generation Tool Guide

For more information on the Dio Driver Component Generation Tool, please refer “Generation Tool User Manual” 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 Defintions

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

Refer "AUTOSAR_SWS_DIODriver.pdf" for more type definitions

10.2.1. Dio_ChannelType

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

10.2.2. Dio_PortType

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

10.2.3. 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

Table 10-1 DIO Driver Component's API list.

Sl no	API's 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

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 (1/2)

Sl. No.	**1**
Error Code	DIO_E_PARAM_INVALID_CHANNEL_ID
Related API(s)	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(s)	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 API(s)	Dio_ReadPort, Dio_WritePort , Dio_WriteChannel, Dio_FlipChannel, Dio_WriteChannelGroup and Dio_MaskedWritePort
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 API(s)	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_INVALID_DATABASE
Related API(s)	Dio_Init
Source of Error	When the above mentioned API is called without a database or invalid database

Sl. No.	**6**
Error Code	DIO_E_UNINIT
Related API(s)	Dio_ReadChannel, Dio_WriteChannel, Dio_ReadPort, Dio_WritePort, Dio_ReadChannelGroup , Dio_WriteChannelGroup and Dio_FlipChannel
Source of Error	When the above mentioned APIs are invoked without module initialization.

11.2. Dio Driver Component Production Errors

None.

Chapter 12 Memory Organization

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

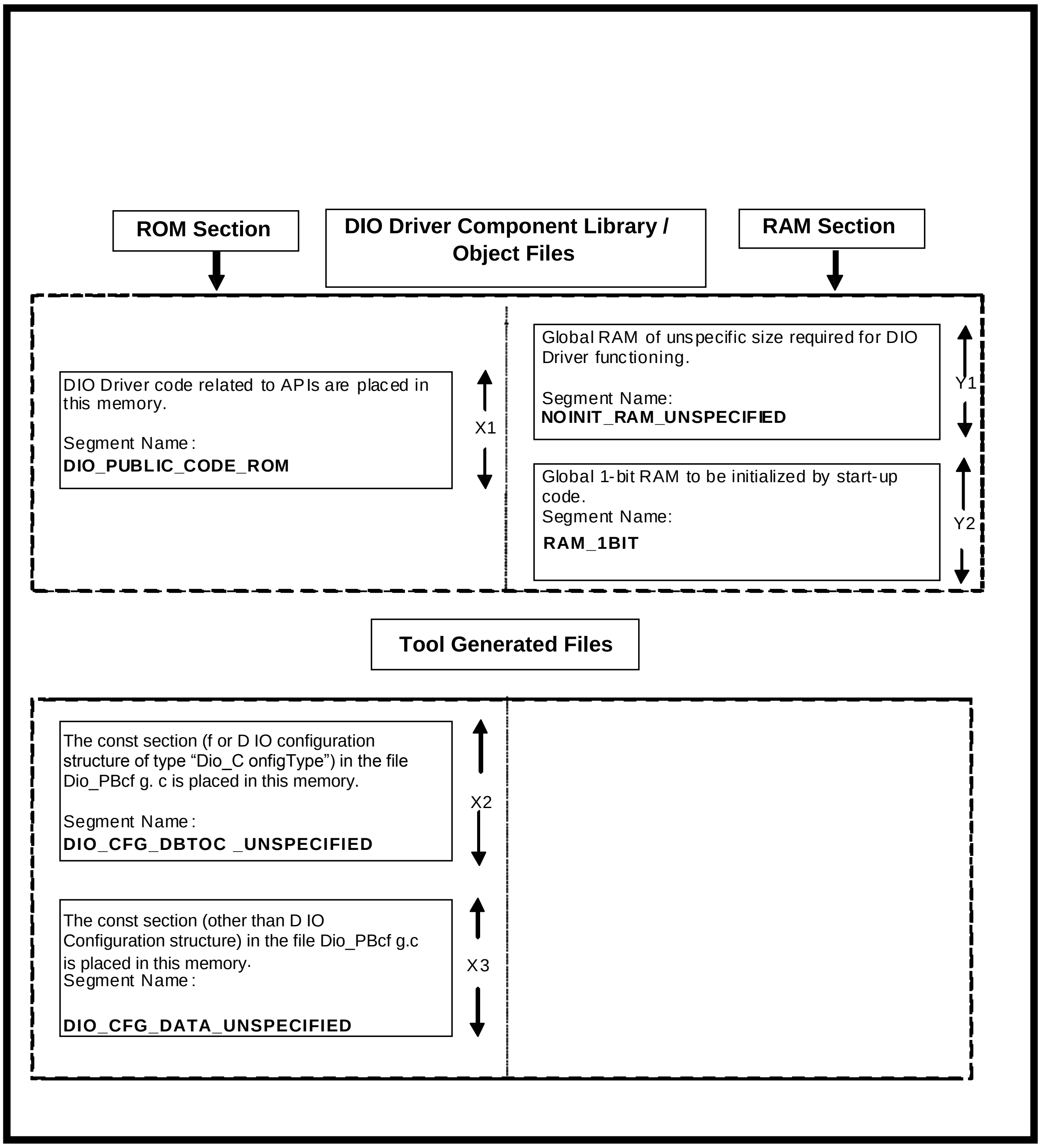
ROM Section
DIO Driver Component Library / Object Files
RAM Section
DIO Driver code related to APIs are placed in this memory.
Segment Name:
DIO_PUBLIC_CODE_ROM
X1
Global RAM of unspecific size required for DIO Driver functioning.
Segment Name:
NOINIT_RAM_UNSPECIFIED
Y1
Global 1-bit RAM to be initialized by start-up code.
Segment Name:
RAM_1BIT
Y2
Tool Generated Files
The const section (f or D IO configuration structure of type "Dio_C onfigType") in the file Dio_PBcf g. c is placed in this memory.
Segment Name:
DIO_CFG_DBTOC _UNSPECIFIED
X2
The const section (other than D IO Configuration structure) in the file Dio_PBcf g.c is placed in this memory.
Segment Name:
DIO_CFG_DATA_UNSPECIFIED
X3

Figure 12-1 DIO Driver Component Memory Organization

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

DIO_CFG_DBTOC_UNSPECIFIED (X2): This section consists of DIO Driver Component database table of contents generated by the DIO Driver Component Generation Tool. This can be located in code memory.

DIO_CFG_DATA_UNSPECIFIED (X3): This section consists of DIO Driver Component constant configuration structures. This can be located in code memory.

RAM Section (Y1, and Y2):

NOINIT_RAM_UNSPECIFIED (Y1): This section consists of the global RAM pointer variables that are used internally by DIO Driver Component. This can be located in data memory.

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.

Notes:

- X1, Y1, and Y2 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 P1M Specific Information

P1M supports following devices:

- R7F701304
- R7F701305
- R7F701310
- R7F701311
- R7F701312
- R7F701313
- R7F701314
- R7F701315
- R7F701318
- R7F701319
- R7F701320
- R7F701321
- R7F701322
- R7F701323

13.1. Interaction Between The User And DIO Driver Component

13.1.1. Translation Header File

The P1x_translation.h file supports following devices:

- R7F701304
- R7F701305
- R7F701310
- R7F701311
- R7F701312
- R7F701313
- R7F701314
- R7F701315
- R7F701318
- R7F701319
- R7F701320
- R7F701321
- R7F701322
- R7F701323

13.1.2. Parameter Definition File

Parameter definition files support information for P1M

Table 13-1 PDF information for P1M

PDF Files	Devices Supported
R403_DIO_P1M_04_05_12_13_20_21.arxml	701304, 701305, 701312, 701313, 701320, 701321
R403_DIO_P1M_10_11_14_15_18_19_22_23.arxml	701310, 701311, 701314, 701315, 701318, 701319, 701322, 701323

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.

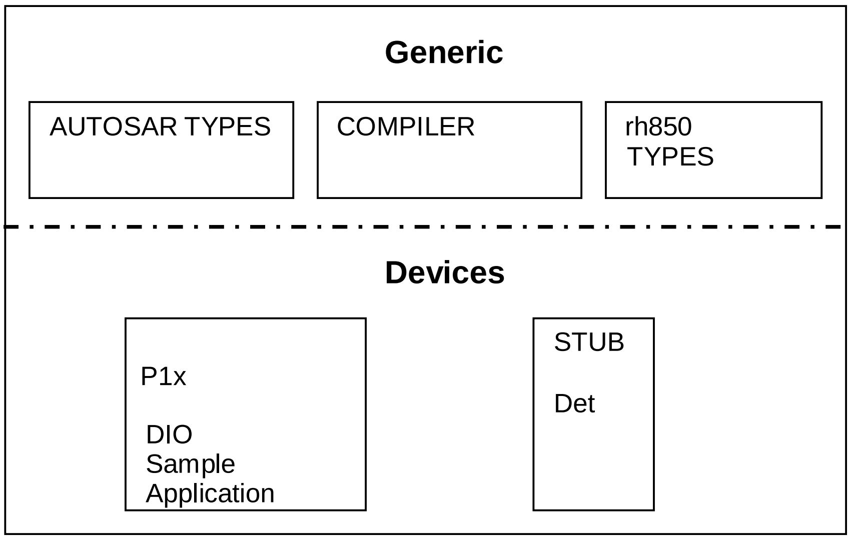

Figure 13-1 Overview Of DIO Driver Sample Application

The Sample Application of the P1M is available in the path

X1X/P1x/modules/dio/sample_application

The Sample Application consists of the following folder structure

X1X/P1x/modules/dio/definition/<Subvariant>/R403_DIO_P1M_04_05_12_13_20_21.arxml

X1X/P1x/modules/dio/definition/<Subvariant>/R403_DIO_P1M_10_11_14_15_18_19_22_23.arxml

X1X/P1x/modules/dio/sample_application/<Subvariant> /<AUTOSAR_version>

/src/Dio_PBcfg.c
/inc/Dio_Cfg.h
/config/App_DIO_P1M_701304_Sample.one
/config/App_DIO_P1M_701304_Sample.arxml
/config/App_DIO_P1M_701304_Sample.html
/config/App_DIO_P1M_701305_Sample.one
/config/App_DIO_P1M_701305_Sample.arxml
/config/App_DIO_P1M_701305_Sample.html
/config/App_DIO_P1M_701310_Sample.one
/config/App_DIO_P1M_701310_Sample.arxml
/config/App_DIO_P1M_701310_Sample.html
/config/App_DIO_P1M_701311_Sample.one
/config/App_DIO_P1M_701311_Sample.arxml
/config/App_DIO_P1M_701311_Sample.html
/config/App_DIO_P1M_701312_Sample.one
/config/App_DIO_P1M_701312_Sample.arxml
/config/App_DIO_P1M_701312_Sample.html
/config/App_DIO_P1M_701313_Sample.one
/config/App_DIO_P1M_701313_Sample.arxml
/config/App_DIO_P1M_701313_Sample.html
/config/App_DIO_P1M_701314_Sample.one
/config/App_DIO_P1M_701314_Sample.arxml
/config/App_DIO_P1M_701314_Sample.html
/config/App_DIO_P1M_701315_Sample.one
/config/App_DIO_P1M_701315_Sample.arxml
/config/App_DIO_P1M_701315_Sample.html
/config/App_DIO_P1M_701318_Sample.one
/config/App_DIO_P1M_701318_Sample.arxml
/config/App_DIO_P1M_701318_Sample.html
/config/App_DIO_P1M_701319_Sample.one
/config/App_DIO_P1M_701319_Sample.arxml
/config/App_DIO_P1M_701319_Sample.html
/config/App_DIO_P1M_701320_Sample.one
/config/App_DIO_P1M_701320_Sample.arxml
/config/App_DIO_P1M_701320_Sample.html
/config/App_DIO_P1M_701321_Sample.one
/config/App_DIO_P1M_701321_Sample.arxml
/config/App_DIO_P1M_701321_Sample.html
/config/App_DIO_P1M_701322_Sample.one
/config/App_DIO_P1M_701322_Sample.arxml
/config/App_DIO_P1M_701322_Sample.html
/config/App_DIO_P1M_701323_Sample.one

/config/App_DIO_P1M_701323_Sample.arxml
/config/App_DIO_P1M_701323_Sample.html

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 level of the required channel. The level read for the channel is actual physical level of that channel if the channel is configured in input mode and input buffer for that channel is enabled.
- 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 level of all the channels of a required port. The level read is actual physical level of the channels that are configured in input mode and whose input buffer is also 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 level of group of channels of a required port. The level read is actual physical level of the channels that are configured in input mode and whose input buffer is also enabled.
- The API Dio_MaskedWritePort provides service to set value of given port with required mask.The Dio_MaskedWritePort funcion shall set the specified value for the channel in specified port if the corresponding bit in mask is ‘1’.
- 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_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.

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.

Configuration Details:

Name of the configuration file: App_DIO_P1M_<Device_name>.html

13.2.2.2 Debugging the Sample Application

Remark 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\common_family\make\<Compiler>”

Now execute batch file SampleApp.bat with following parameters:

SampleApp.bat dio 4.0.3 <Device_name>

<Device_name> can be 701304, 701305, 701310, 701311, 701312, 701313, 701314, 701315, 701318, 701319, 701320, 701321, 701322 and 701323.

After this, the tool output files will be generated with the configuration as mentioned in App_DIO_P1M_<Device_name>_Sample.html file available in the path:

- “X1X\P1x\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\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.

Remark Executable files with ‘*.out’ extension can be downloaded into the target hardware with the help of Green Hills debugger.

- If any configuration changes (only post-build) are made to the ECU Configuration Description files
- “X1X\P1x\modules\dio\sample_application\< SubVariant > \<AUTOSAR_version>config \ App_DIO_P1M_701310_Sample.arxml”

The database alone can be generated by using the following commands.

```
make –f App_DIO_P1M_Sample.mak generate_dio_config
make –f App_DIO_P1M_Sample.mak App_Dio_P1M_Sample.s37
```

After this, a flashable Motorola S-Record file App_Dio_P1M_Sample.s37 is available in the output folder.

13.3. Memory And Throughput

13.3.1 ROM/RAM Usage

The details of memory usage for the typical configuration provided in Section 13.3.2.1 *Configuration Example* are provided in this section.

Table 13-2 ROM/RAM Details with DET

Sl. No.	ROM/RAM	Segment Name	Size in bytes for 144 pin device
1.	ROM	DIO_PUBLIC_CODE_ROM	1296
		DIO_PRIVATE_CODE_ROM	0
		CONST_ROM_UNSPECIFIED	0
		DIO_CFG_DATA_UNSPECIFIED	40
		DIO_CFG_DBTOC_UNSPECIFIED	16
2.	RAM	RAM_1BIT	1
		NOINIT_RAM_UNSPECIFIED	8
		DIO_CFG_RAM_UNSPECIFIED	0

The details of memory usage for the typical configuration, with DET enabled and all other configurations as provided in 13.3.2.1 *Configuration Example* are provided in this section.

Table 13-3 ROM/RAM Details without DET

Sl. No.	ROM/RAM	Segment Name	Size in bytes for 144 pin device
1.	ROM	DIO_PUBLIC_CODE_ROM	690
		DIO_PRIVATE_CODE_ROM	0
		CONST_ROM_UNSPECIFIED	0
		DIO_CFG_DATA_UNSPECIFIED	40
		DIO_CFG_DBTOC_UNSPECIFIED	16

2.	RAM	RAM_1BIT NOINIT_RAM_UNSPECIFIED DIO_CFG_RAM_UNSPECIFIED	0 8 0

13.3.2. Stack Depth

The worst-case stack depth for DIO Driver Component for the typical

Configuration provided in Section 13.3.2.1 is 24 bytes.

13.3.3. Throughput Details

The throughput details of the APIs for the configuration mentioned in the Section13.3.2.1 Configuration Example.
The clock frequency used to measure the throughput is 80 MHz for all APIs

Table 13-4 Throughput Details of the APIs

Sl. No.	API Name	Throughput in microseconds	Remarks
1.	Dio_Init	0.212	-
2.	Dio_WriteChannel	0.912	-
3.	Dio_ReadChannel	0.337	-
4.	Dio_WritePort	0.787	-
5.	Dio_ReadPort	0.212	-
6.	Dio_WriteChannelGroup	0.937	-
7.	Dio_ReadChannelGroup	0.350	-
8.	Dio_MaskedWritePort	0.787	-
9.	Dio_FlipChannel	0.875	-
10.	Dio_GetVersionInfo	0.12	-

Chapter 14 Release Details

Embedded DIO Driver Software

Version: 1.0.9

Revision History

Sl.No.	Description	Version	Date
1.	Initial Version	1.0.0	30-Sep-2013
2	Updated for 100 pins and 144 pins device for P1x E4.02	1.0.1	27-Jan-2014
3	Following change is made: • In page no 40, Header is changed. • In section 13.3.2.1, configuration file mentioned changed. • In page no 54, Header is added.	1.0.2	13-Mar-2014
4.	Following change is made: • In section 13.4 RAM/ROM details and throughput details are updated. • Section 13.1.2 is added. • Section 13.2 is updated for compile and Linker and Assembler options.	1.0.3	25-Aug-2014
5.	Following changes are made: • Table 11-1 is updated. • Repeated DIO_CFG_DBTOC_UNSPECIFIED is removed from RAM section from table 13-5 and from table 13-6. • DIO_E_PARAM_POINTER is removed from DET Error codes.	1.0.4	04-Sep-2014
6.	Following changes are made: • User manual is updated as per Template. • Dio_MaskedWritePort information is provided in section 5 and section 7. • Throughput Details are updated in section 13.4.3	1.0.5	18-Nov-2014
7.	Following changes are made: • Chapter 2 reference documents section is updated. • Chapter 3 section 3.1.1 is updated for new devices. • Chapter 4 section 4.1 General is updated and a note is added regarding interrupt vector (.c) file • Chapter 4 section 4.4 Deviation list is updated. • Chapter 13 Section for compiler, linker, assembler details is removed. • Chapter 13 13.1.1 Translation header files section is updated as per new devices. 13.1.2 PDF section is updated as per new devices. 13.2.1 Sample application structure is updated for new devices 13.2.2 Building sample application section is updated 13.3 Memory and throughput details are updated for 144 pin devices.	1.0.6	27-May-2015

AUTOSAR MCAL R4.0.3 User's Manual
DIO Driver Component Ver.1.0.6

Publication Date: Rev.0.02, May 27, 2015

Published by: Renesas Electronics Corporation

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.
2880 Scott Boulevard Santa Clara, CA 95050-2554, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
1101 Nicholson Road, Newmarket, Ontario L3Y 9C3, Canada
Tel: +1-905-898-5441, Fax: +1-905-898-3220

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: +49-211-65030, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
7th Floor, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100083, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 204, 205, AZIA Center, No.1233 Lujiazui Ring Rd., Pudong District, Shanghai 200120, China
Tel: +86-21-5877-1818, Fax: +86-21-6887-7858 / -7898

Renesas Electronics Hong Kong Limited
Unit 1601-1613, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2886-9318, Fax: +852 2886-9022/9044

Renesas Electronics Taiwan Co., Ltd.
7F, No. 363 Fu Shing North Road Taipei, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
1 harbourFront Avenue, #06-10, keppel Bay Tower, Singapore 098632
Tel: +65-6213-0200, Fax: +65-6278-8001

Renesas Electronics Malaysia Sdn.Bhd.
Unit 906, 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 Korea Co., Ltd.
11F., Samik Lavied' or Bldg., 720-2 Yeoksam-Dong, Kangnam-Ku, Seoul 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

© 2015 Renesas Electronics Corporation. All rights reserved.

AUTOSAR MCAL R4.0.3

User’s Manual

Renesas Electronics Corporation