Z20K14xM DIO Driver User Manual

Table of contents

1. Revision history

Revision	Date	Author	Description
1.0	06.04.2023	ZhiXin MCAL Team	New creation

2. About this manual

2.1. Scope and purpose

This document describes DIO driver for Z20K14xM. This document provide deviations from the specification and limitations of DIO driver. AUTOSAR DIO driver requirements and APIs are

described in the AUTOSAR DIO driver software specification document.

The purpose of this document is to enable users to integrate the DIO driver with basic software (BSW) stack.

2.2. Intended audience

This document is intended for anyone using the DIO driver of the Z20K14xM MCAL software.

2.3. Glossary

Term	Description
DIO	Digital Input Output
PORT	Represents a whole configurable port on an MCU device.
API	Application Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basic software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
MCU	Microcontroller Unit
PARCC	Peripheral Access & Reset & Clock Controller
SWS	Software Specification

2.4. Reference documents

This manual should be read in conjunction with the following documents:

1. Specification of DIO Driver, AUTOSAR_SWS_DIODriver.pdf, AUTOSAR release R20-11
2. Requirements on DIO Driver, AUTOSAR_SRS_DIODriver.pdf, AUTOSAR release R20-11
3. Z20K14xM Series Reference Manual
4. Specification of MCU Driver, AUTOSAR_SWS_MCUDriver.pdf, AUTOSAR release R20-11
5. Specification of PORT Driver, AUTOSAR_SWS_PortDriver.pdf, AUTOSAR release R20-11
6. Specification of ECU State Manager, AUTOSAR_SWS_ECUStateManager.pdf, AUTOSAR release R20-11
7. Specification of Diagnostic Event Manager, AUTOSAR_SWS_DiagnosticEventManager.pdf, AUTOSAR release R20-11
8. Specification of Default Error Tracer, AUTOSAR_SWS_DefaultErrorTracer.pdf, AUTOSAR release R20-11
9. Specification of RTE Software, AUTOSAR_SWS_RTE.pdf, AUTOSAR release R20-11

10. Z20K14xM pinmux.xlsx

3. Introduction

The DIO Driver provides services for reading and writing to/from:

- DIO Channels (Pins)
- DIO Ports
- DIO Channel Groups.

The DIO driver works on pins and ports which are configured by the PORT driver for this purpose. For this reason, there is no configuration and initialization of this port structure in the DIO Driver.

4. Deviations and limitations

4.1. Deviations

The following are the the deviations from software specification.

Requirement	Description	Status	Comments
SWS_Dio_00084	If the microcontroller does not support the direct read-back of a pin value, the Dio module's read functions shall provide the value of the output register, when they are used on a channel which is configured as an output channel.	Not implemented	SWS_Dio_00083 conflicts with the SWS_Dio_00084 because the microcontroller support the direct read-back of a pin value

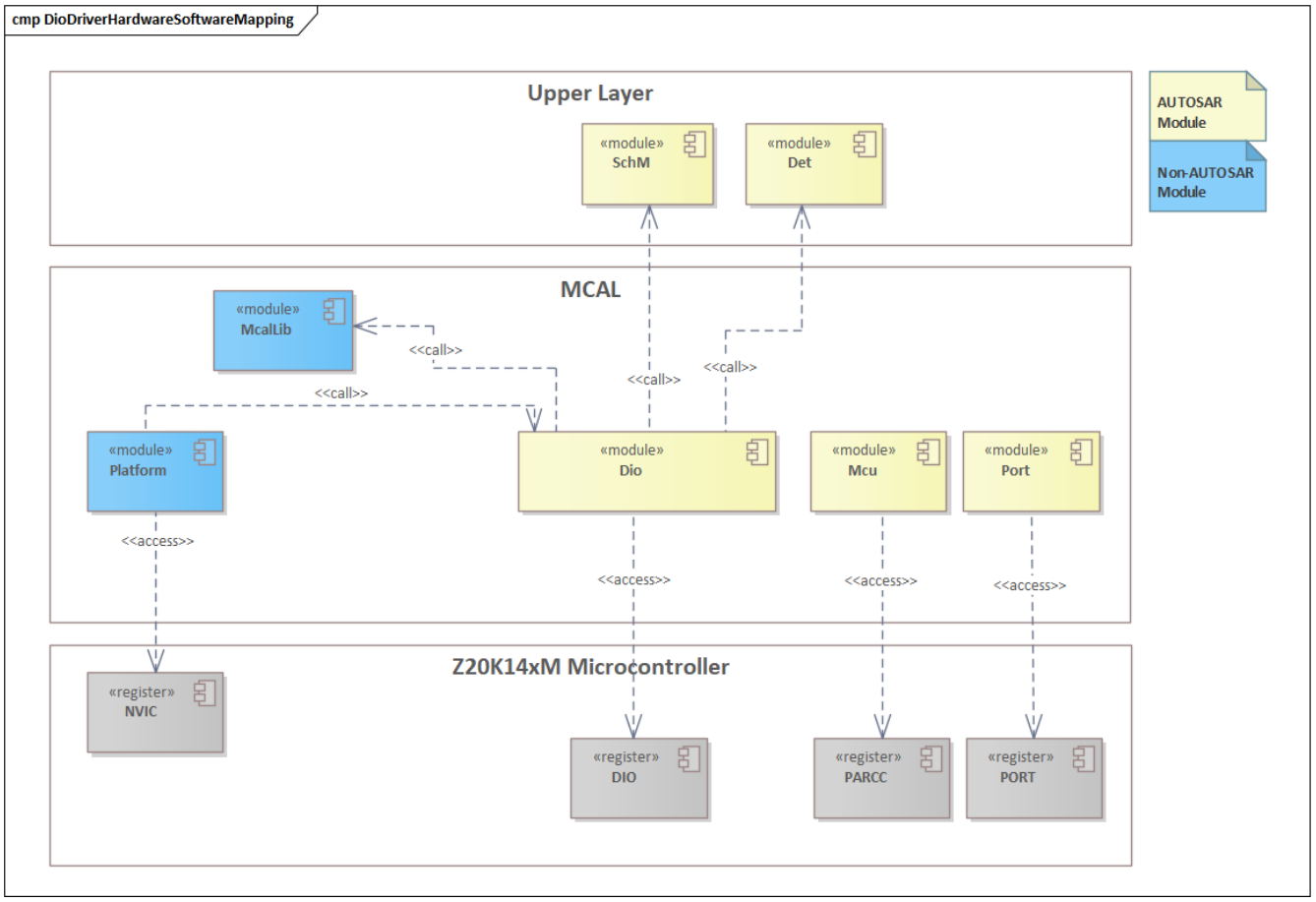
4.2. Limitations

The following are the limitations for the Dio driver.

- The available pins in different packages are different. Therefore, user needs to query the pinmux table and select the pin in a certain package.

5. Hardware software mapping

This section describes the system view of the driver and hardware peripherals.



5.1. DIO: primary hardware peripheral

The DIO Driver abstracts the access to the microcontroller's hardware pins. Furthermore, it allows the grouping of those pins.

The key hardware functional features used by the driver are:

- The logic level of the pin is visible in all digital pin-multiplexing modes.
- Allow one or more outputs within one port to be set, cleared, or toggled.

5.2. PARCC: dependent hardware peripheral

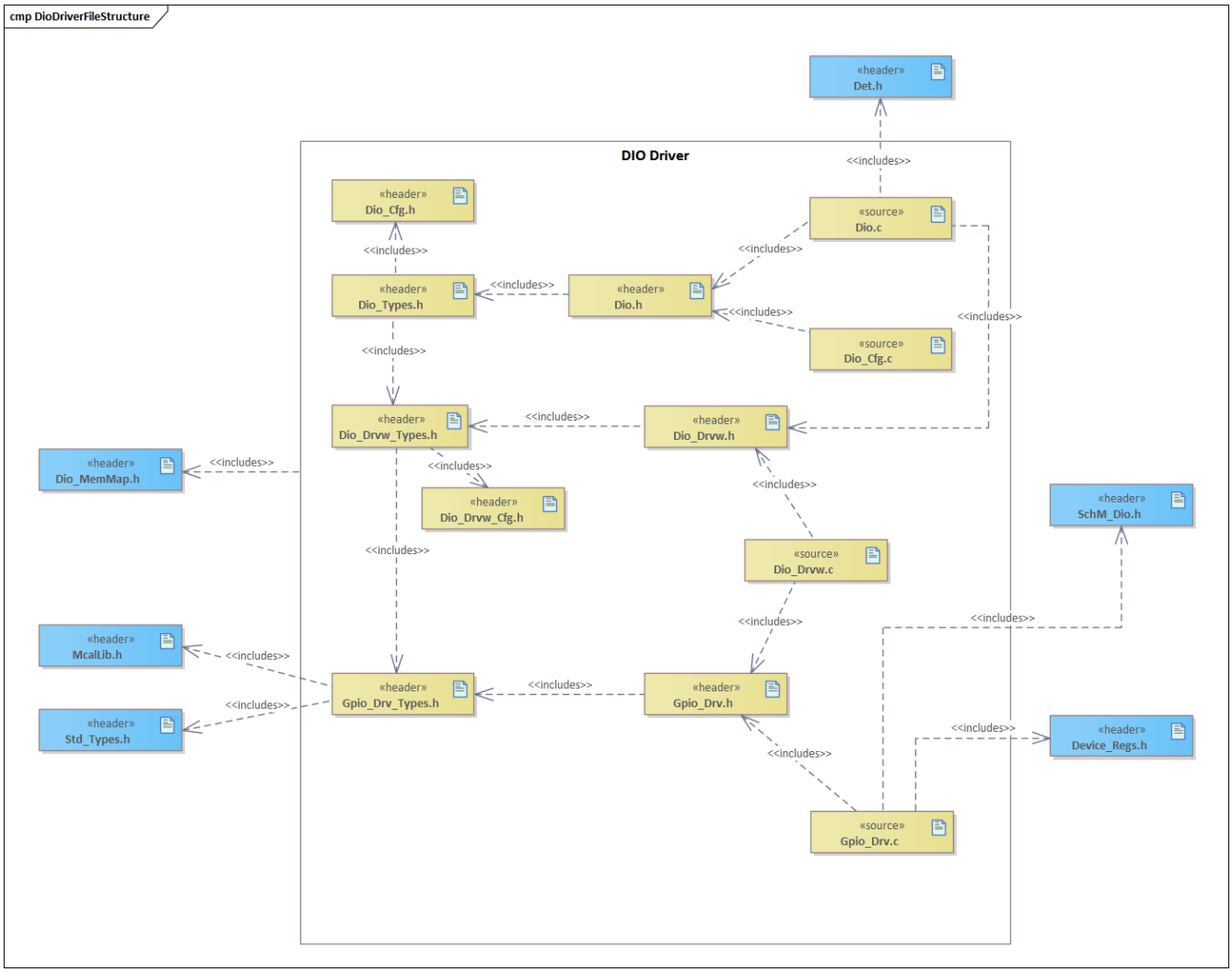
The DIO driver depends on the PARCC IP for functional clock source and clock mode configuration. These are configured through the MCU Driver.

5.3. PORT: dependent hardware peripheral

The DIO driver depends on the PORT IP for pin-multiplexing modes and pin direction. These are configured through the PORT driver.

6. File structure

This section provides details of the C files of the DIO driver.



File name	Description
Dio.c	AUTOSAR level source File containing implementation of APIs
Dio.h	AUTOSAR level header file containing declarations of APIs
Dio_Cfg.c	AUTOSAR level DIO configuration source file containing definition of the pre-processor configuration data structures
Dio_Cfg.h	AUTOSAR level configuration Header file (Generated) containing pre-processor macros
Dio_Types.h	AUTOSAR level header file containing definition of data types
Dio_Drvw.h	DIO Driver wrapper layer header file containing declarations of APIs
Dio_Drvw.c	DIO Driver wrapper layer source file containing implementations of APIs
Dio_Drvw_Cfg.h	DIO Driver wrapper layer configuration Header file (Generated) containing pre-processor macros
Dio_Drvw_Types.h	DIO Driver wrapper layer header file containing definition of data types
Gpio_Drv_Types.h	GPIO Low-level driver header file containing definition of data types
Gpio_Drv.h	GPIO Low-level driver header file containing declarations of APIs
Gpio_Drv.c	GPIO Low-level driver source file containing implementations of APIs

File name	Description
Device_Regs.h	Registers definition header file containing type definition of register data structure and IRQ number
McalLib.h	McalLib header file containing declarations of APIs and definitions of data types
SchM_Dio.h	DIO SchM header file containing enter and exit exclusive areas function declarations of DIO driver
Dio_MemMap.h	DIO driver memory mapping header file containing definitions of memory section
Det.h	Header file of Det module containing declarations of APIs

7. Configuration tips

This section lists the configuration tips that an integrator or user of the DIO driver may need.

7.1. Channel Configuration

DIO driver provides Dio_WriteChannel() and Dio_FlipChannel() to change the Physical Level of the channel.Therefore, the channel (pin) must be pinmux into GPIO mode and output in the port module.

In addition, make sure that the package selected by the Resource module has a channel that user wants to apply.

- Select the chip package for the actual application:

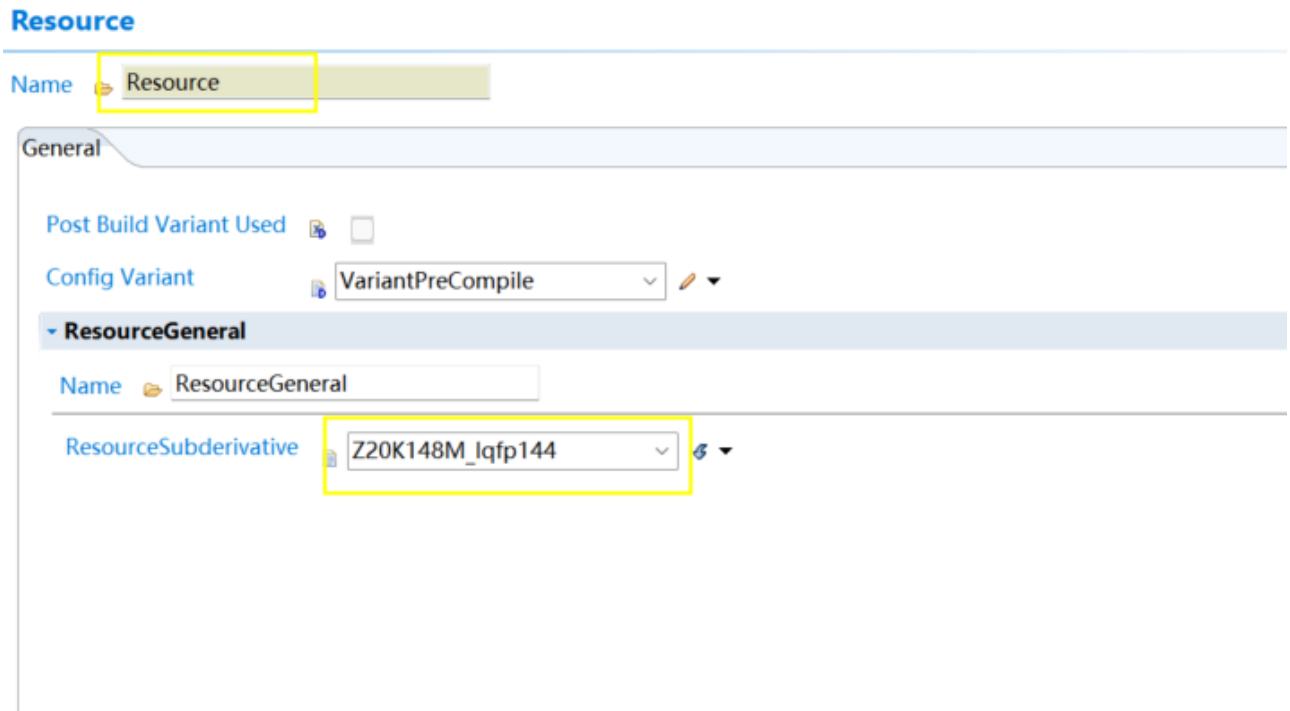


Figure 1. Resource package

- Select one channel(pin) to configure as GPIO mode output:

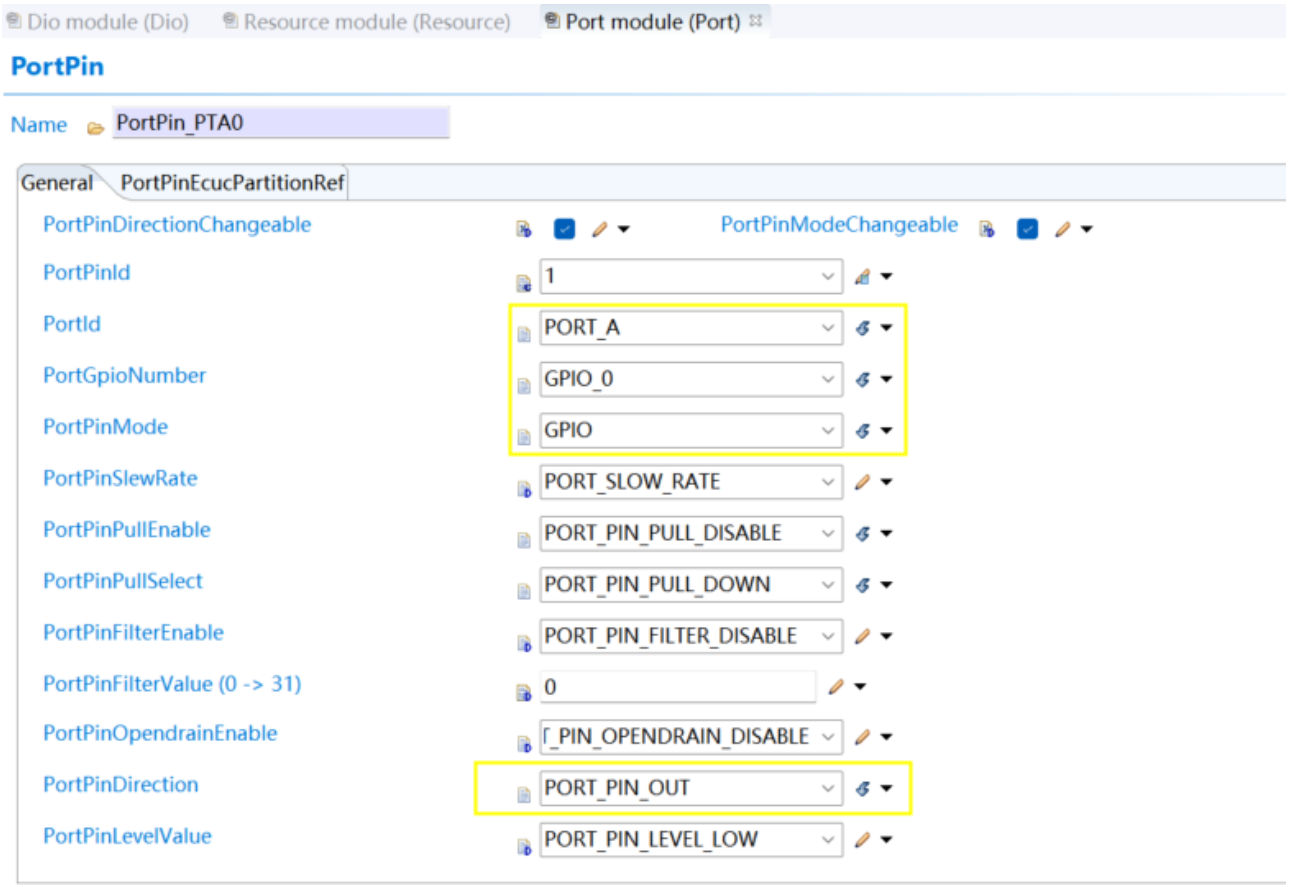


Figure 2. GPIO mode output

- Select the available channel:

Figure 3. Available channel

7.2. Channel group Configuration

DIO driver provides functions to modify the levels of a channel group and read the level of a channel group. A channel group is a formal logical combination of several adjoining DIO channels within a DIO port.

- Configure a channel group of a port:

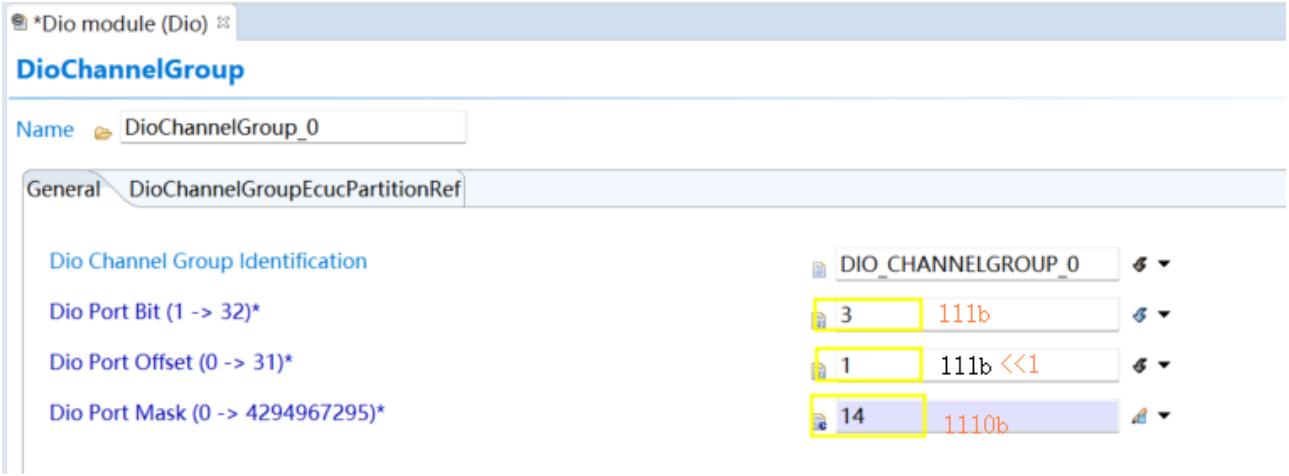


Figure 4. A channel group configuration

8. Integration hints

This section lists the key points that an integrator or user of the DIO driver must consider.

8.1. Integration with AUTOSAR

This section lists the modules, which are not part of the MCAL, but are required to integrate the DIO driver.

Det

The DET module is is a BSW module that handles all detected development and runtime errors reported by the BSW modules. The DIO driver reports all the development errors to the DET module through the Det_ReportError() API, and report runtime errors to the DET module through Det_ReportRuntimeError() API.

The Det module is implemented as stub code in the MCAL package, and it must be replaced with a complete Det module during the integration phase.

SchM

The SchM is a part of the RTE that apply data consistency mechanisms for BSW modules. ExclusiveAreas mechanism is used to guarantee data consistency. The ExclusiveArea concept is to block potential concurrent accesses to get data consistency. ExclusiveAreas implement critical section.

The DIO driver uses the exclusive areas defined in SchM_Dio.c file to protect the registers and variables access.

The SchM_Dio.c and SchM_Dio.h files are provided in the MCAL package as an example code and needs to be updated by the integrator.

Memory mapping

AUTOSAR specifies mechanisms for the mapping of code and data to specific memory sections via memory mapping files. For many ECUs and microcontroller platforms it is of utmost necessity to be able to map code, variables and constants to specific memory sections.

Memory mapping macros are provided to select specific memory sections. These macros are defined in the Dio_MemMap.h file.

The Dio_MemMap.h file is provided as an example code in the MCAL package, and it must be replaced with appropriate compiler pragmas during the integration phase.

8.2. MCU support

The MCU driver is primarily responsible for initializing and controlling the chip's internal clock sources and clock prescalers. The DIO driver depends on MCU driver to configure clock sources and clock prescalers.

The use of the DIO driver must be started after completion of the MCU initialization and PORT initialization.

The following parcc configuration must be considered while configuring the MCU driver in the EB tresos:

- Configure PARCC for GPIO used by DIO driver.

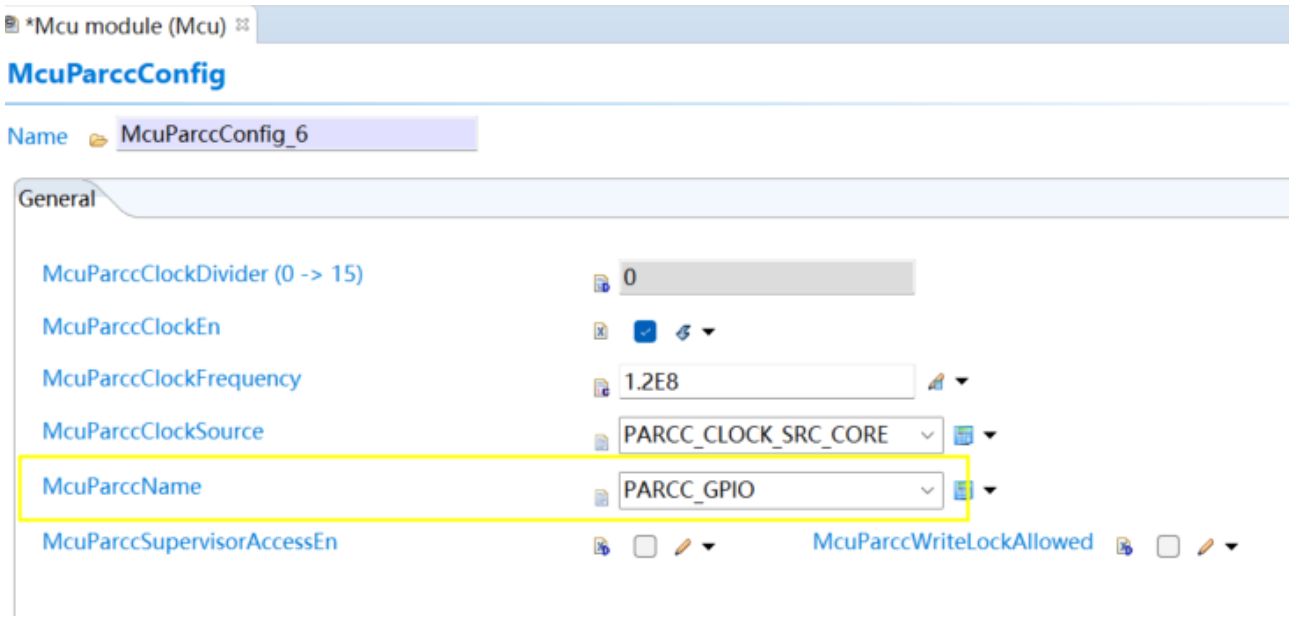


Figure 5. PARCC Configuration in MCU driver

- Configure clock reference point for GPIO used by DIO driver.

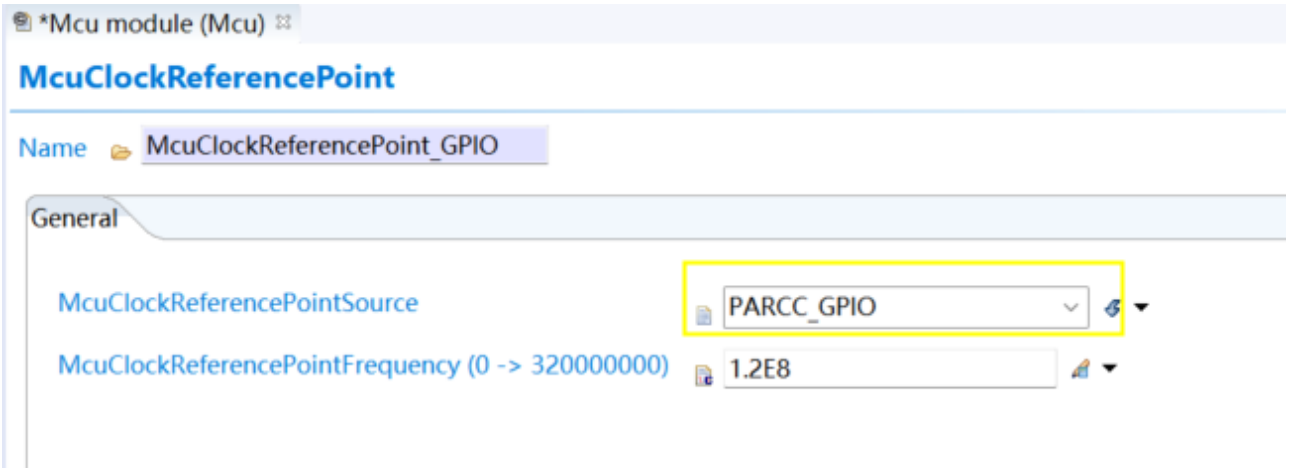


Figure 6. Clock Reference Point configuration in MCU driver

- Select clock reference point for GPIO hardware unit in PORT module.

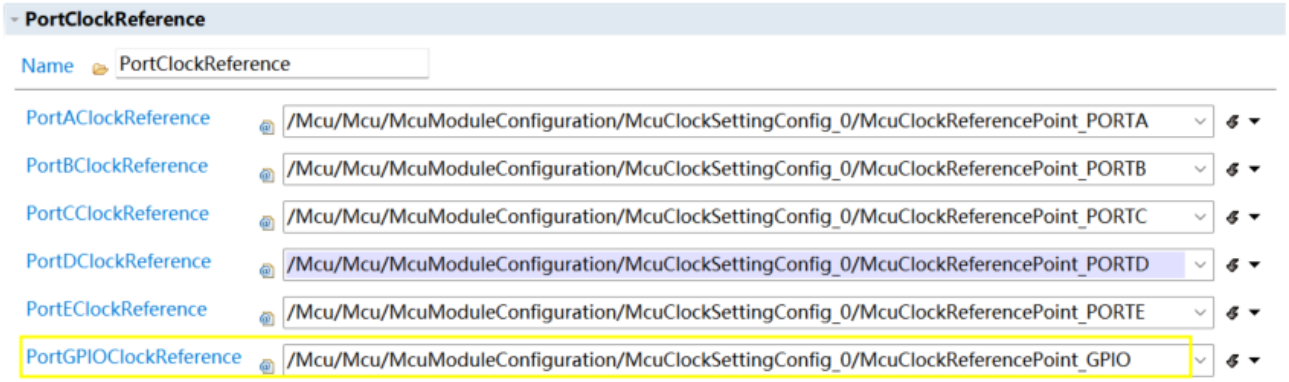


Figure 7. Select clock reference for GPIO hardware unit

8.3. PORT support

The PORT driver shall configure the port pins used by the DIO driver.The available pins in different packages are different,so user should consider it.

The Dio module shall not provide an interface for initialization of the hardware. The Port Driver performs this.

9. Assumptions of Use (AoUs)

This section lists the AoUs that an integrator or user of the DIO driver must consider:

Correct channel/port/channel group ID passed to APIs

The integrator shall verify that the symbolic-name macros generated for DIO channels/ports/channel groups in the Dio_Cfg.h file holds the correct values, and use these symbolic-name macros when invoking the APIs of the DIO driver.

Correct pointer of configuration structure

The integrator shall ensure that correct pointer to the configuration structure is passed for the DIO driver.

How to Reach Us:

Home Page:

zhixin-semi.com/

Information in this document is provided solely to enable system and software implementers to use Zhixin products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. Zhixin reserves the right to make changes without further notice to any products herein. Zhixin makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Zhixin assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Zhixin data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. Zhixin does not convey any license under its patent rights nor the rights of others. Zhixin sells products pursuant to standard terms and conditions of sale, which can be found at the following address: http://www.zhixin-semi.com.

Document Number: Z20K14xM-MCAL-DIO-UM

Revision 1.0.0, April, 2023