Z20K14xM PORT 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 PORT driver for Z20K14xM. This document provide deviations from the specification and limitations of PORT driver. AUTOSAR PORT driver requirements and APIs are described in the AUTOSAR PORT driver software specification document.

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

2.2. Intended audience

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

2.3. Glossary

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

2.4. Reference documents

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

1. Requirements on PORT Driver, AUTOSAR_SRS_PortDriver.pdf, AUTOSAR release R20-11
2. Specification of PORT Driver, AUTOSAR_SWS_PortDriver.pdf, AUTOSAR release R20-11
3. Specification of MCU Driver, AUTOSAR_SWS_MCUDriver.pdf, AUTOSAR release R20-11
4. Specification of ECU State Manager, AUTOSAR_SWS_ECUStateManager.pdf, AUTOSAR release

R20-11

5. Specification of Diagnostic Event Manager, AUTOSAR_SWS_DiagnosticEventManager.pdf, AUTOSAR release R20-11
6. Specification of Default Error Tracer, AUTOSAR_SWS_DefaultErrorTracer.pdf, AUTOSAR release R20-11
7. Specification of RTE Software, AUTOSAR_SWS_RTE.pdf, AUTOSAR release R20-11
8. Z20K14xM Series Reference Manual

3. Introduction

The PORT driver module provides the service for initializing the whole PORT structure of the micro controller. Many ports and port pins can be assigned to various functionalities, for example: SPI/I2C/CAN/UART.

The PORT driver module also provides overall configuration and initialization of this port structure. The configuration and mode of these port pins is micro controller and ECU dependent.

The PORT driver module completes the overall configuration and initialization of the port structure which is used in the DIO driver module. Therefore, the DIO driver works on pins and ports which are configured by the PORT driver.

The PORT driver shall be initialized prior to use of the DIO functions. Otherwise DIO functions will exhibit undefined behavior.

4. Deviations and limitations

4.1. Deviations

The following are the the deviations from software specification.

Requirement	Description	Status	Comments
SWS_Port_00138	If the MCU port control hardware provides an output latch for setting the output level on a port pin, switching the port pin direction shall not alter the level set in this output latch	Out of scope	Hardware not supports output latch

4.2. Limitations

The following are the limitations for the PORT driver.

- Due to the hardware limitation, some of the pin mode is not available for some pins, please refer to pinmux document for more information
- The specific available pins are limited to different variant of Z20K14xM series and different package.

5. Hardware software mapping

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

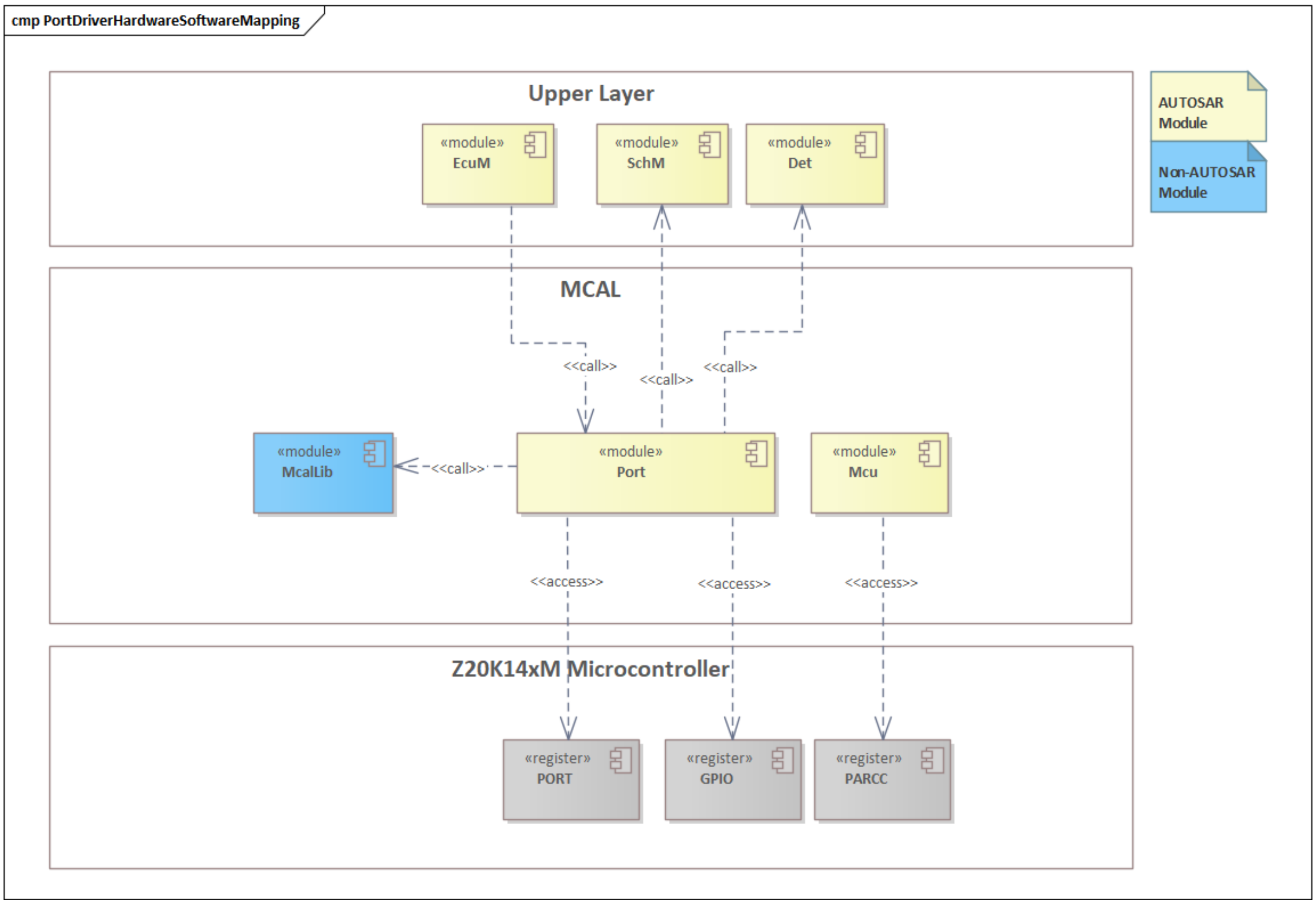


Figure 1. Hardware software mapping

5.1. PORT: primary hardware peripheral

The Port module provides support for port pinmux mode, pull configuration and filter functions. Most functions can be configured independently for each pin in the 32-bit port and affect the pin regardless of its pinmux state. There is one instance of the PORT module for each port. However, not all pins within each port are implemented on a specific device.

The key hardware functional features used by the driver are:

- Individual pull control fields, which supports pullup, pulldown, and pull-disable
- Enable and disable the individual open-drain mode
- Individual mux control field supporting analog or pin disabled, GPIO, and up to six chip-specific digital functions
- Pad configuration fields are functional in all digital pin-multiplexing modes

5.2. GPIO: primary hardware peripheral

The general-purpose input and output (GPIO) module communicate with the CPU via a zero-wait state interface for maximum pin performance.

The GPIO data direction and output data registers control the direction and output data of each pin when the pin is configured for the GPIO function, only if the corresponding PORT module for that pin is enabled.

The key hardware functional features used by the driver are:

- Port data Output set function
- Port data Output clear function
- Port data Direction register

5.3. PARCC: dependent hardware peripheral

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

6. File structure

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

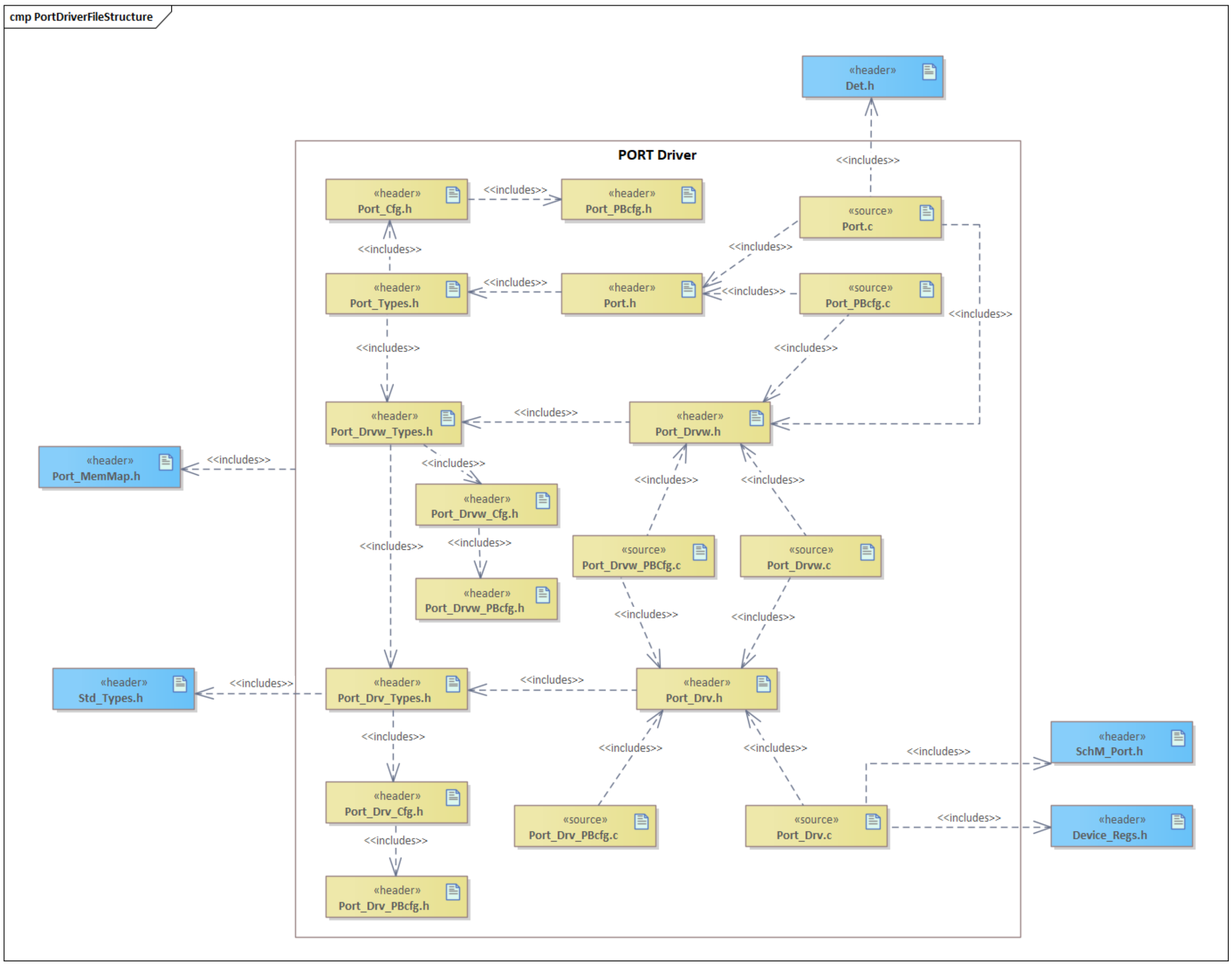


Figure 2. C File structure

File name	Description
Port.c	AUTOSAR level source file containing implementations of APIs
Port.h	AUTOSAR level header file containing declarations of APIs

File name	Description
Port_Types.h	AUTOSAR level header file containing definition of data types
Port_Cfg.h	AUTOSAR level configuration Header file (Generated) containing pre-processor macros
Port_PBcfg.c	AUTOSAR level post-build configuration source file (Generated) containing definition of the post-build configuration data structures
Port_PBcfg.h	AUTOSAR level post-build configuration Header file (Generated) containing declaration of the post-build configuration data structures
Port_Drvw.c	PORT driver wrapper source file containing implementations of APIs
Port_Drvw.h	PORT driver wrapper header file containing declarations of APIs
Port_Drvw_Types.h	PORT driver wrapper header file containing definition of data types
Port_Drvw_Cfg.h	PORT driver wrapper configuration Header file (Generated) containing pre-processor macros
Port_Drvw_PBcfg.c	PORT driver wrapper post-build configuration source file (Generated) containing definition of the post-build configuration data structures
Port_Drvw_PBcfg.h	PORT driver wrapper post-build configuration Header file (Generated) containing declaration of the post-build configuration data structures
Port_Drv.c	PORT Low-level driver source file containing implementations of APIs
Port_Drv.h	PORT Low-level driver header file containing declarations of APIs
Port_Drv_Types.h	PORT Low-level driver header file containing definition of data types
Port_Drv_Cfg.h	PORT Low-level driver configuration Header file (Generated) containing pre-processor macros
Port_Drv_PBcfg.c	PORT Low-level driver post-build configuration source file (Generated) containing definition of the post-build configuration data structures
Port_Drv_PBcfg.h	PORT Low-level driver post-build configuration Header file (Generated) containing declaration of the post-build configuration data structures
Device_Regs.h	Registers definition header file containing type definition of register data structure and IRQ number
SchM_Port.h	Port SchM header file containing enter and exit exclusive areas function declarations of Port driver
Port_MemMap.h	Port driver memory mapping header file containing definitions of memory section
Det.h	Header file of Det module containing declarations of APIs

7. Configuration tips

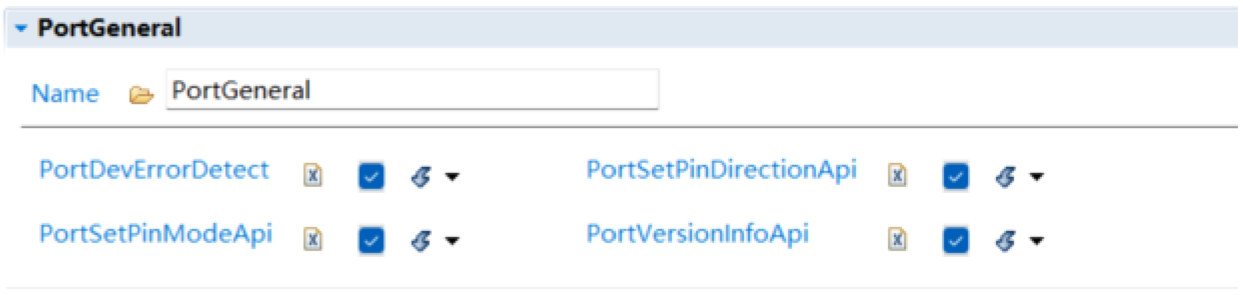
This section lists the configuration tips that an integrator or user of the PORT driver may need.

7.1. Port API generation control

Port driver provides several configurable APIs if needed.

These APIs are not automatically generated.

- Enable/disable these APIs via parameters in PortGeneral container.



PortGeneral

Name: PortGeneral

PortDevErrorDetect	✓	PortSetPinDirectionApi	✓
PortSetPinModeApi	✓	PortVersionInfoApi	✓

Figure 3. PORT Enable Configuration

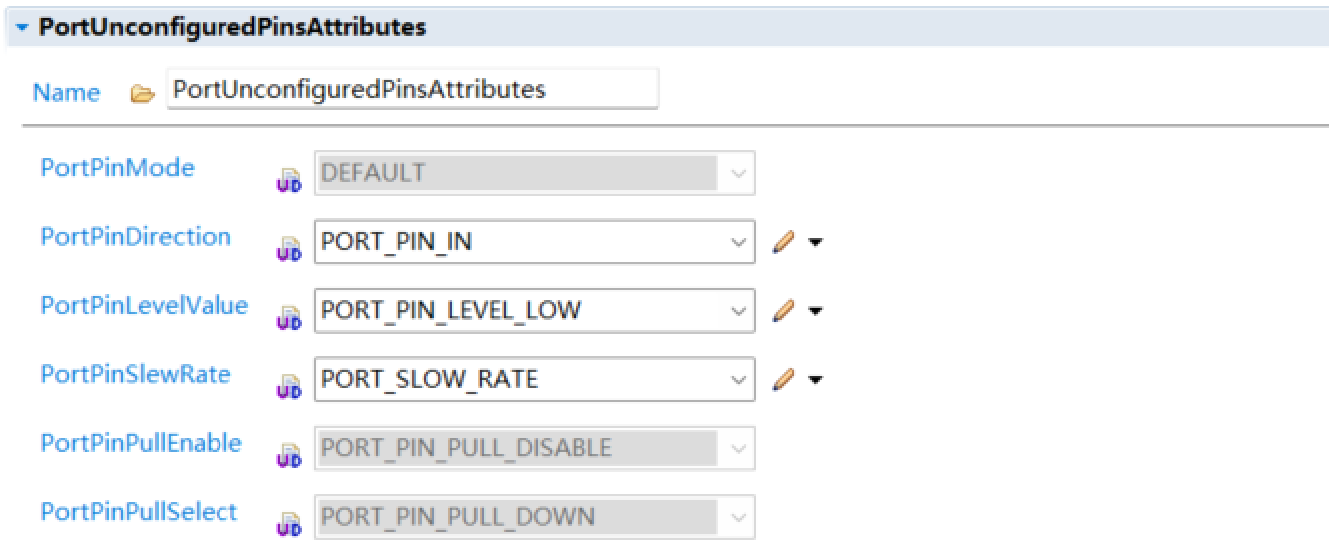
7.2. Configure un-configured pins

According to the AUTOSAR spec, the PORT Driver module shall configure all ports and port pins that are not used (neither as GPIO nor special purpose IO) to be set to a defined state by the PORT Driver module configuration.

Please carefully configured the pins attributes which are not used in application, these configurations will be set in PORT init process. Some attributes are not selectable for these pins to avoid un-expected behavior.

For example, SWD/JTAG might not be connectable when corresponding pins are configured as other pin mode value or pull attribute is enabled.

- The pin mode is set to default whose ID is defined in pinmux document and is not selectable.
- The pull attributes are also not selectable



PortUnconfiguredPinsAttributes

Name: PortUnconfiguredPinsAttributes

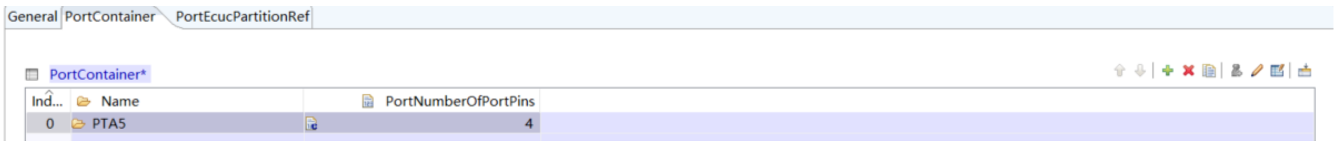
PortPinMode	DEFAULT
PortPinDirection	PORT_PIN_IN
PortPinLevelValue	PORT_PIN_LEVEL_LOW
PortPinSlewRate	PORT_SLOW_RATE
PortPinPullEnable	PORT_PIN_PULL_DISABLE
PortPinPullSelect	PORT_PIN_PULL_DOWN

Figure 4. Unconfigured Pin Configuration

7.3. Port Container

The user need to configure the pin attributes that are used in application.

In the PortContainer, user need to add a Port container that has the configured pins in it.

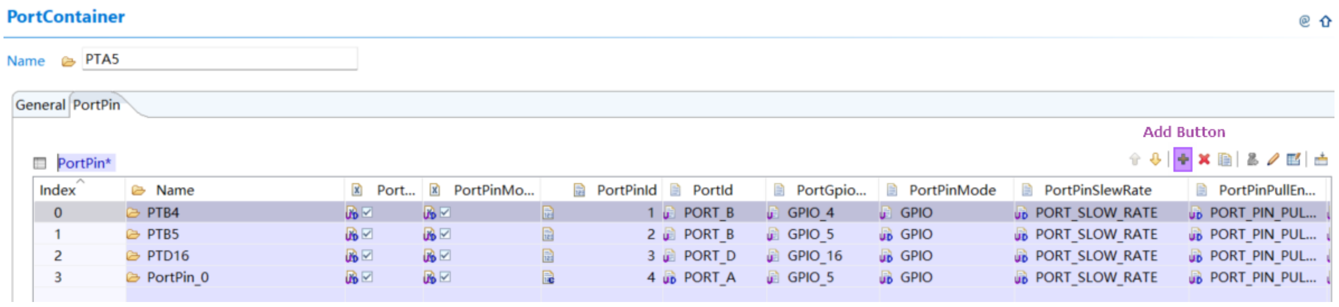


Ind...	Name	PortNumberOfPortPins
0	PTA5	4

Figure 5. Port Container

7.4. Pin configuration in Port Container

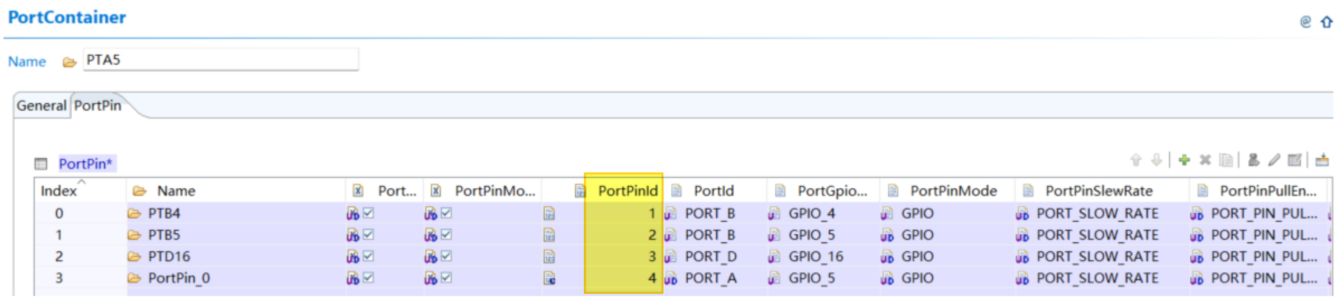
User can add the pins in the Port container by "+" button



Index	Name	Port...	PortPinMo...	PortPinId	PortId	PortGpio...	PortPinMode	PortPinSlewRate	PortPinPullEn...
0	PTB4	✓	✓	1	PORT_B	GPIO_4	GPIO	PORT_SLOW_RATE	PORT_PIN_PUL...
1	PTB5	✓	✓	2	PORT_B	GPIO_5	GPIO	PORT_SLOW_RATE	PORT_PIN_PUL...
2	PTD16	✓	✓	3	PORT_D	GPIO_16	GPIO	PORT_SLOW_RATE	PORT_PIN_PUL...
3	PortPin_0	✓	✓	4	PORT_A	GPIO_5	GPIO	PORT_SLOW_RATE	PORT_PIN_PUL...

Figure 6. Add pin in Port Container

The EB will automatically calculate and check the next available PortPinId value in the container.



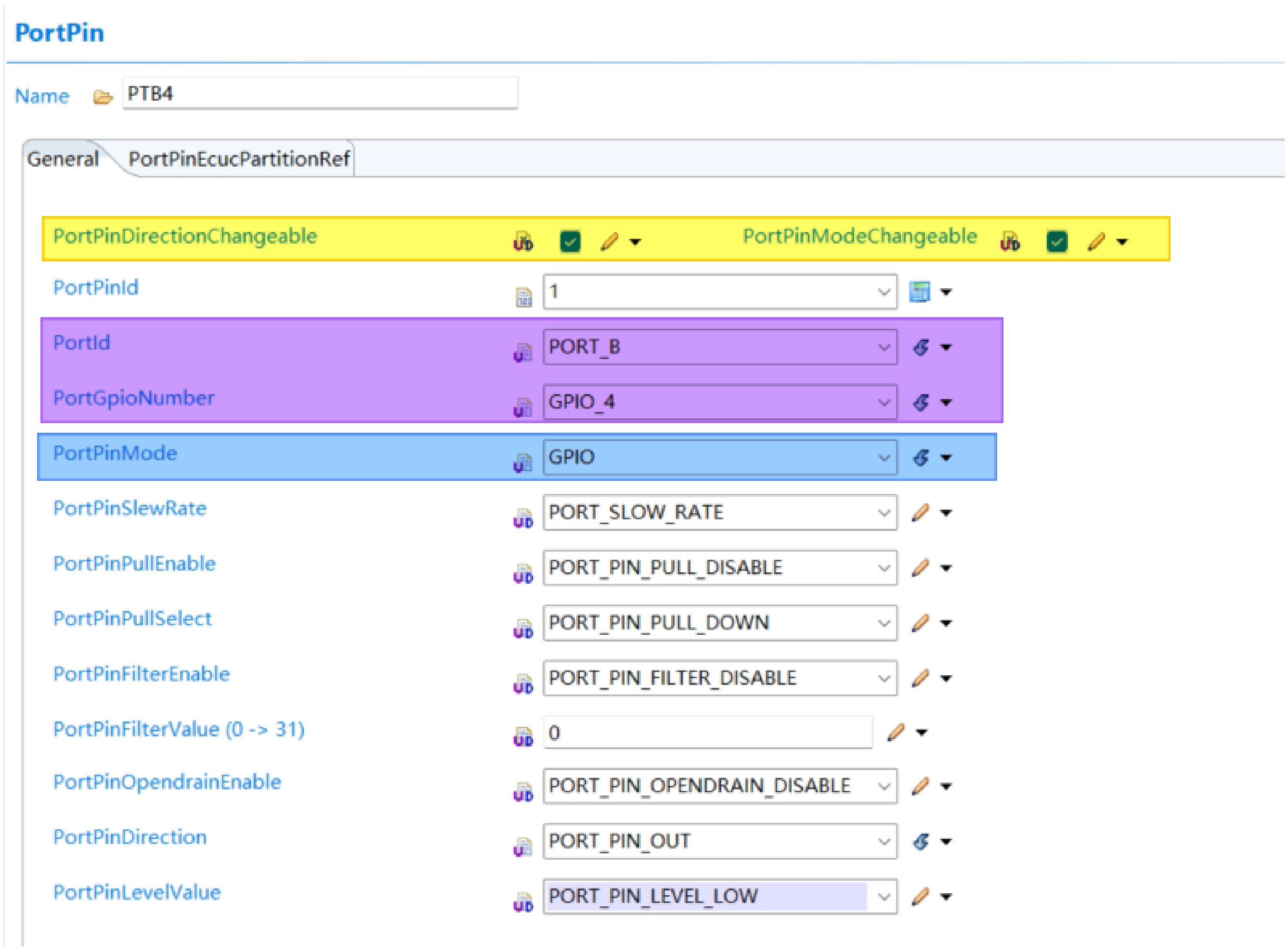
Index	Name	Port...	PortPinMo...	PortPinId	PortId	PortGpio...	PortPinMode	PortPinSlewRate	PortPinPullEn...
0	PTB4	✓	✓	1	PORT_B	GPIO_4	GPIO	PORT_SLOW_RATE	PORT_PIN_PUL...
1	PTB5	✓	✓	2	PORT_B	GPIO_5	GPIO	PORT_SLOW_RATE	PORT_PIN_PUL...
2	PTD16	✓	✓	3	PORT_D	GPIO_16	GPIO	PORT_SLOW_RATE	PORT_PIN_PUL...
3	PortPin_0	✓	✓	4	PORT_A	GPIO_5	GPIO	PORT_SLOW_RATE	PORT_PIN_PUL...

Figure 7. Port Pin In Port Container

7.5. Pin configuration

By selecting the specific pin in the Port Container, user need to configure the pin attributes for this pin. Please take care of the pin attributes that to be configured.

- User can set pin direction and mode changeability by clicking the configuration in EB
- Select the pin that need to be configured
- Select the pin attributes for this pin



PortPin

Name PTB4

General | PortPinEcucPartitionRef

PortPinDirectionChangeable	✓
PortPinModeChangeable	✓
PortPinId	1
PortId	PORT_B
PortGpioNumber	GPIO_4
PortPinMode	GPIO
PortPinSlewRate	PORT_SLOW_RATE
PortPinPullEnable	PORT_PIN_PULL_DISABLE
PortPinPullSelect	PORT_PIN_PULL_DOWN
PortPinFilterEnable	PORT_PIN_FILTER_DISABLE
PortPinFilterValue (0 -> 31)	0
PortPinOpendrainEnable	PORT_PIN_OPENDRAIN_DISABLE
PortPinDirection	PORT_PIN_OUT
PortPinLevelValue	PORT_PIN_LEVEL_LOW

Figure 8. Port Pin Direction and Mode changeability

IMPORTANT

Note: *When configuring the PortPinMode as shown in the above diagram, if it involves the ports listed in the table below, please take into account whether their default functions are required. If their default functions are needed and you intend to configure different functions for the port, consider using alternative ports for implementation.*

For example: PTA11's default function is EXTAL32, providing a working voltage to OSC32K. If you enable the external clock OSC32K and need to use a non-default multiplexed function for this pin, then OSC32K will not function correctly, leading to unexpected errors. You can use another pin to replace PTA11's non-default function.

pin name	default function
PTA4	JTAG_TMS/SWD_DIO
PTA5	RESET_b
PTA10	XTAL32
PTA11	EXTAL32
PTB6	XTAL
PTB7	EXTAL
PTC4	JTAG_TCLCK/SWD__CLK

pin name	default function
PTC5	JTAG_TDI
PTE0	JTAG_TDO

7.6. Number of port pins

The tool can automatically calculate the number of pins that are selected and configured.

User can select the "Calculate value" function to finish the configurations, after this operation and if no error occurs, user can generated the code.

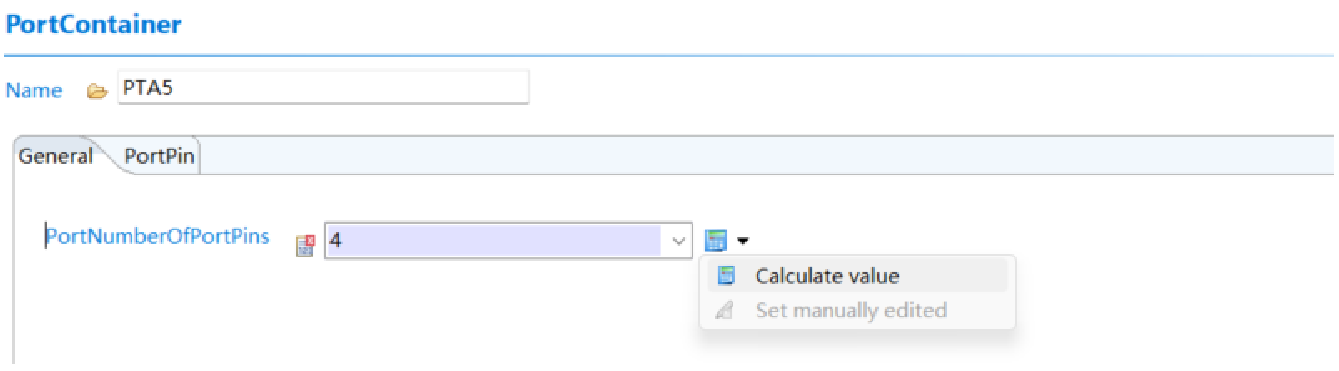


Figure 9. Calculate number of configured pins

8. Integration hints

This section lists the key points that an integrator or user of the PORT 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 PORT driver.

Det

The Det module is a BSW module that handles all detected development and runtime errors reported by the BSW modules. The PORT 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. Exclusive areas mechanism is used to guarantee data consistency. The Exclusive area concept is to block potential concurrent accesses to get data consistency. Exclusive areas implement critical section.

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

The SchM_Port.c and SchM_Port.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 micro controller 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 Port_MemMap.h file.

The Port_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.

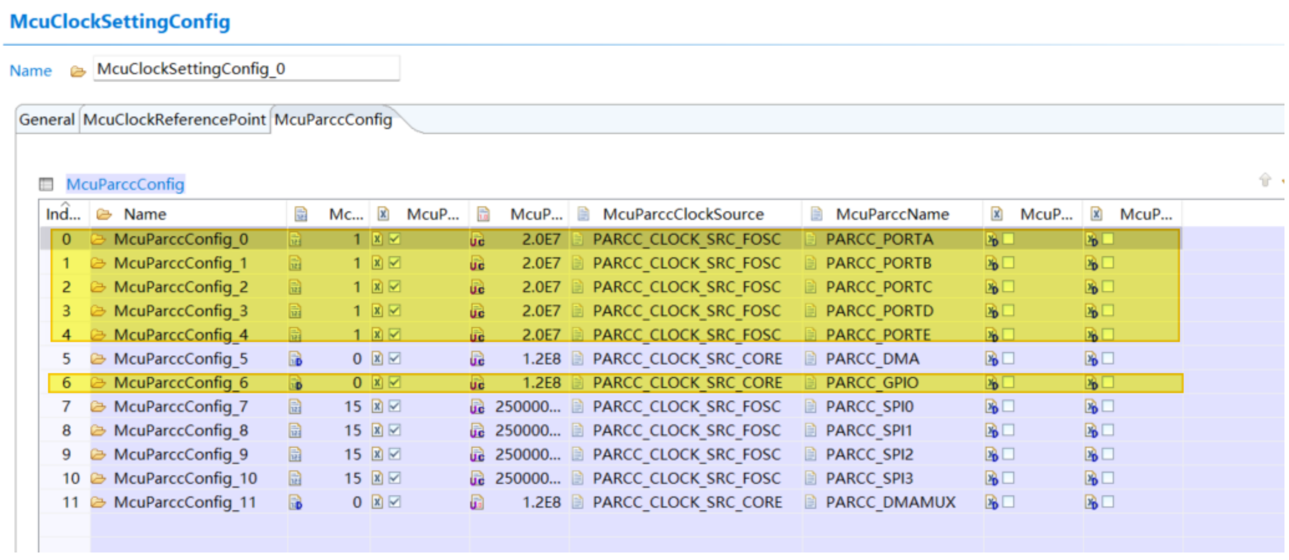
8.2. MCU support

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

The initialization of the PORT driver must be started only after completion of the MCU initialization.

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

- Configure PARCC for PORTA to PORTE and GPIO peripherals used by PORT driver.



McuClockSettingConfig

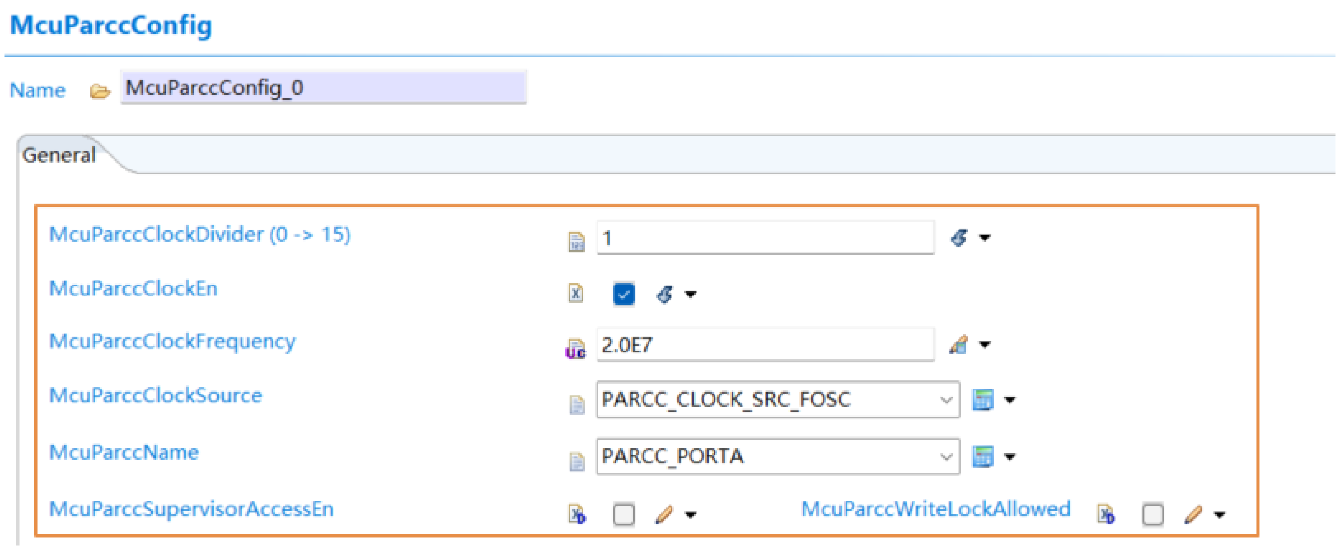
Name: McuClockSettingConfig_0

General | McuClockReferencePoint | McuParccConfig

McuParccConfig

Ind...	Name	Mc...	McuP...	McuP...	McuParccClockSource	McuParccName	McuP...	McuP...
0	McuParccConfig_0	1	✓	2.0E7	PARCC_CLOCK_SRC_FOSC	PARCC_PORTA	☐	☐
1	McuParccConfig_1	1	✓	2.0E7	PARCC_CLOCK_SRC_FOSC	PARCC_PORTB	☐	☐
2	McuParccConfig_2	1	✓	2.0E7	PARCC_CLOCK_SRC_FOSC	PARCC_PORTC	☐	☐
3	McuParccConfig_3	1	✓	2.0E7	PARCC_CLOCK_SRC_FOSC	PARCC_PORTD	☐	☐
4	McuParccConfig_4	1	✓	2.0E7	PARCC_CLOCK_SRC_FOSC	PARCC_PORTE	☐	☐
5	McuParccConfig_5	0	✓	1.2E8	PARCC_CLOCK_SRC_CORE	PARCC_DMA	☐	☐
6	McuParccConfig_6	0	✓	1.2E8	PARCC_CLOCK_SRC_CORE	PARCC_GPIO	☐	☐
7	McuParccConfig_7	15	✓	250000...	PARCC_CLOCK_SRC_FOSC	PARCC_SPI0	☐	☐
8	McuParccConfig_8	15	✓	250000...	PARCC_CLOCK_SRC_FOSC	PARCC_SPI1	☐	☐
9	McuParccConfig_9	15	✓	250000...	PARCC_CLOCK_SRC_FOSC	PARCC_SPI2	☐	☐
10	McuParccConfig_10	15	✓	250000...	PARCC_CLOCK_SRC_FOSC	PARCC_SPI3	☐	☐
11	McuParccConfig_11	0	✓	1.2E8	PARCC_CLOCK_SRC_CORE	PARCC_DMAMUX	☐	☐

Figure 10. Configure PORTx and GPIO


McuParccConfig

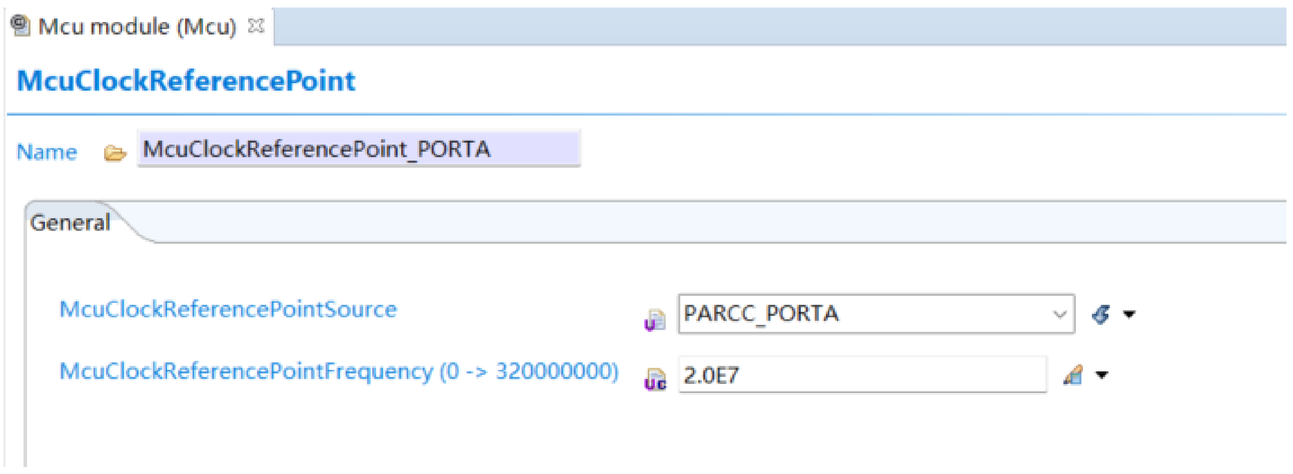
Name: McuParccConfig_0

General

McuParccClockDivider (0 -> 15)	1
McuParccClockEn	✓
McuParccClockFrequency	2.0E7
McuParccClockSource	PARCC_CLOCK_SRC_FOSC
McuParccName	PARCC_PORTA
McuParccSupervisorAccessEn	☐
McuParccWriteLockAllowed	☐

Figure 11. Clock configuration in MCU driver

- Configure clock reference point for PORTA to PORTE and GPIO peripherals used by PORT driver.



McuClockReferencePoint

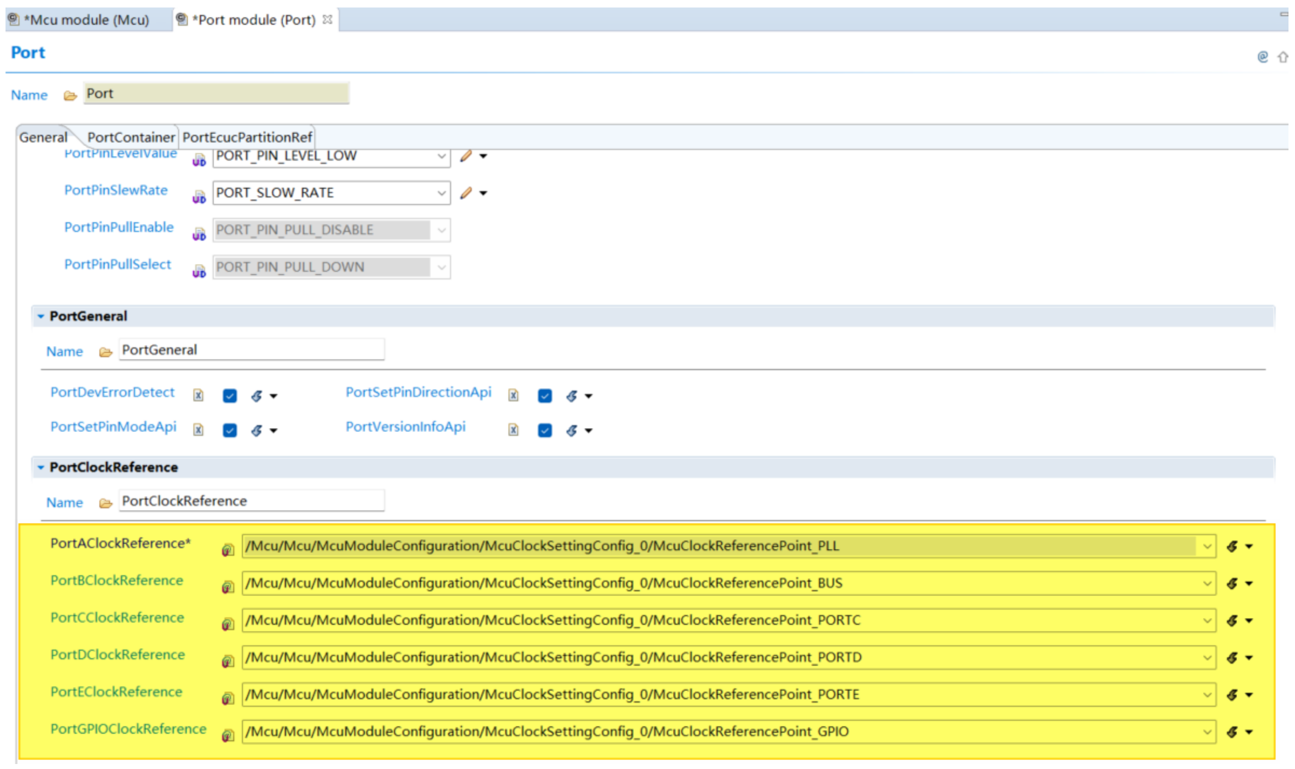
Name: McuClockReferencePoint_PORTA

General

McuClockReferencePointSource	PARCC_PORTA
McuClockReferencePointFrequency (0 -> 320000000)	2.0E7

Figure 12. Clock Reference Point configuration in MCU driver

- Select clock reference point for PORTA to PORTE and GPIO hardware unit.



Port

Name: Port

PortPinLevelValue: PORT_PIN_LEVEL_LOW
PortPinSlewRate: PORT_SLOW_RATE
PortPinPullEnable: PORT_PIN_PULL_DISABLE
PortPinPullSelect: PORT_PIN_PULL_DOWN

PortGeneral

Name: PortGeneral

PortDevErrorDetect ✓ | PortSetPinDirectionApi ✓
PortSetPinModeApi ✓ | PortVersionInfoApi ✓

PortClockReference

Name: PortClockReference

PortAClockReference*	/Mcu/Mcu/McuModuleConfiguration/McuClockSettingConfig_0/McuClockReferencePoint_PLL
PortBClockReference	/Mcu/Mcu/McuModuleConfiguration/McuClockSettingConfig_0/McuClockReferencePoint_BUS
PortCClockReference	/Mcu/Mcu/McuModuleConfiguration/McuClockSettingConfig_0/McuClockReferencePoint_PORTC
PortDClockReference	/Mcu/Mcu/McuModuleConfiguration/McuClockSettingConfig_0/McuClockReferencePoint_PORTD
PortEClockReference	/Mcu/Mcu/McuModuleConfiguration/McuClockSettingConfig_0/McuClockReferencePoint_PORTE
PortGPIOClockReference	/Mcu/Mcu/McuModuleConfiguration/McuClockSettingConfig_0/McuClockReferencePoint_GPIO

Figure 13. Select clock reference for PORTA to PORTE and GPIO hardware unit

9. Assumptions of Use

This section lists the AoUs that an integrator or user of the PORT driver must consider.

Correct Port Pin passed to APIs

The user shall verify whether the symbolic-name macros generated for PORT in the Port_Cfg.h file holds the correct values, and use these symbolic-name macros when invoking the APIs of the PORT driver.

Correct pointer of configuration structure

The user shall ensure whether correct pointer to the configuration structure is passed for initializing the PORT driver.

Correct Port pin configurations for both used and unused pins

For the unused pins, the configured attributes is for all the pins, make sure the pin attributes are expected otherwise it might bring unexpected behavior.

For the used pins, every pin has its own attributes, make sure the pin attributes are expected. For example, when expecting to output a HIGH level on specific pin, you should configure the port pin in GPIO mode and the PortPinDirection must be set as PORT_PIN_OUT.

For some specific usage and state, user can also set the pin with pull attributes enabled, this can make sure the pin level is not in floating state after PORT initialization.

Port Pin resource

The Port pins that are available depends on the Z20K14xM MCU variants and its package, different variant and package information has different pin resource.

Port Pin mode

Not all the pinmux mode are available in all the pins, user need to select the available modes on the specific pin, user can refer to pinmux doc for more information.

How to Reach Us:

Home Page:

zhixin-semi.com/

Document Number: Z20K14xM-MCAL-PORT-UM

Revision 1.0.0, April, 2023