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

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

2.2. Intended audience

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

2.3. Glossary

Term	Description
WDG	Watchdog
WDOG	Watchdog Timer
API	Application Interface
AoU	Assumption of Use
AUTOSAR	Automotive Open System Architecture
BSW	Basic software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
GPT	General Purpose Timer
ECU	Electronic Control Unit
ISR	Interrupt service routine
MCU	Microcontroller Unit
NVIC	Nested Vectored Interrupt Controller
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. Requirements on WDG Driver, AUTOSAR_SRS_WatchdogDriver.pdf, AUTOSAR release R20-11
2. Specification of WDG Driver, AUTOSAR_SWS_WatchdogDriver.pdf, AUTOSAR release R20-11

3. Specification of MCU Driver, AUTOSAR_SWS_MCUDriver.pdf, AUTOSAR release R20-11
4. Specification of GPT Driver, AUTOSAR_SWS_GPTDriver.pdf, AUTOSAR release R20-11
5. Specification of ECU State Manager, AUTOSAR_SWS_ECUStateManager.pdf, AUTOSAR release R20-11
6. Specification of Diagnostic Event Manager, AUTOSAR_SWS_DiagnosticEventManager.pdf, AUTOSAR release R20-11
7. Specification of Default Error Tracer, AUTOSAR_SWS_DefaultErrorTracer.pdf, AUTOSAR release R20-11
8. Specification of RTE Software, AUTOSAR_SWS_RTE.pdf, AUTOSAR release R20-11
9. Z20K14xM Series Reference Manual

3. Introduction

The WDG driver initializes and controls the Watchdog Timer(WDOG) of the microcontroller. It provides services for initialization, changing the operation mode and setting the trigger condition(timeout). Furthermore, it provides a notification mechanism before and after refresh the WDOG driver.

4. Deviations and limitations

4.1. Deviations

The following are the the deviations from software specification.

Requirement	Description	Status	Comments
SRS_Wdg_12165	For an external watchdog driver the same requirements shall apply like for an internal watchdog driver.	Not implemented	External watchdog is not supported in this driver
SRS_Wdg_12166	A driver for an external SPI watchdog shall allow the static configuration of the required SPI parameters.	Not implemented	External SPI watchdog is not supported in this driver
SRS_Wdg_12167	The external watchdog driver shall have a semantically identical API as an internal watchdog driver.	Not implemented	External watchdog is not supported in this driver
SRS_Wdg_12168	The source code of the external watchdog driver shall be independent from the underlying microcontroller.	Not implemented	

Requirement	Description	Status	Comments
SWS_Wdg_00055	The Wdg module for an external watchdog driver shall have source code that is independent of the microcontroller platform.	Not implemented	External watchdog is not supported in this driver
SWS_Wdg_00076	To access the external watchdog hardware, the corresponding Wdg module instance shall use the functionality and API of the corresponding handler or driver, e.g. the SPI handler or DIO driver.	Not implemented	External watchdog is not supported in this driver
SWS_Wdg_00077	A Wdg module for an external watchdog shall satisfy the same functional requirements and offer the same functional scope as a Wdg module for an internal watchdog. Hence their respective APIs are semantically identical.	Not implemented	External watchdog is not supported in this driver
SWS_Wdg_00078	The Wdg module shall add all parameters required for accessing the external watchdog hardware, e.g. the used SPI channel or DIO port, to the module's published parameters and to the module's configuration parameters.	Not implemented	External watchdog is not supported in this driver
SWS_Wdg_00093	If the watchdog hardware requires an activation code which can be configured or changed, the Wdg Driver shall handle the activation code internally. In this case, the Wdg Driver shall pass the correct activation code to the watchdog hardware and the watchdog hardware in turn shall update the Wdg module's internal variable where the next expected access code is stored	Not implemented	Not supported by hardware
SWS_Wdg_00094	If the watchdog hardware requires an activation code which can be configured or changed, the trigger cycle of the Wdg Driver shall be defined with a value so that updating the activation code by the watchdog hardware can be guaranteed.	Not implemented	Not supported by hardware

Requirement	Description	Status	Comments
SWS_Wdg_00095	If the watchdog hardware requires an activation code which can be configured or changed and the initial activation code can be configured, the activation code shall be provided in the Wdg Driver’s configuration set. If the activation code is fixed for a particular hardware the above requirement can be ignored.	Not implemented	Not supported by hardware

4.2. Limitations

The following are the limitations for the WDG driver.

- Due to the hardware limitation, the activation code is not implemented by driver.

5. Hardware software mapping

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

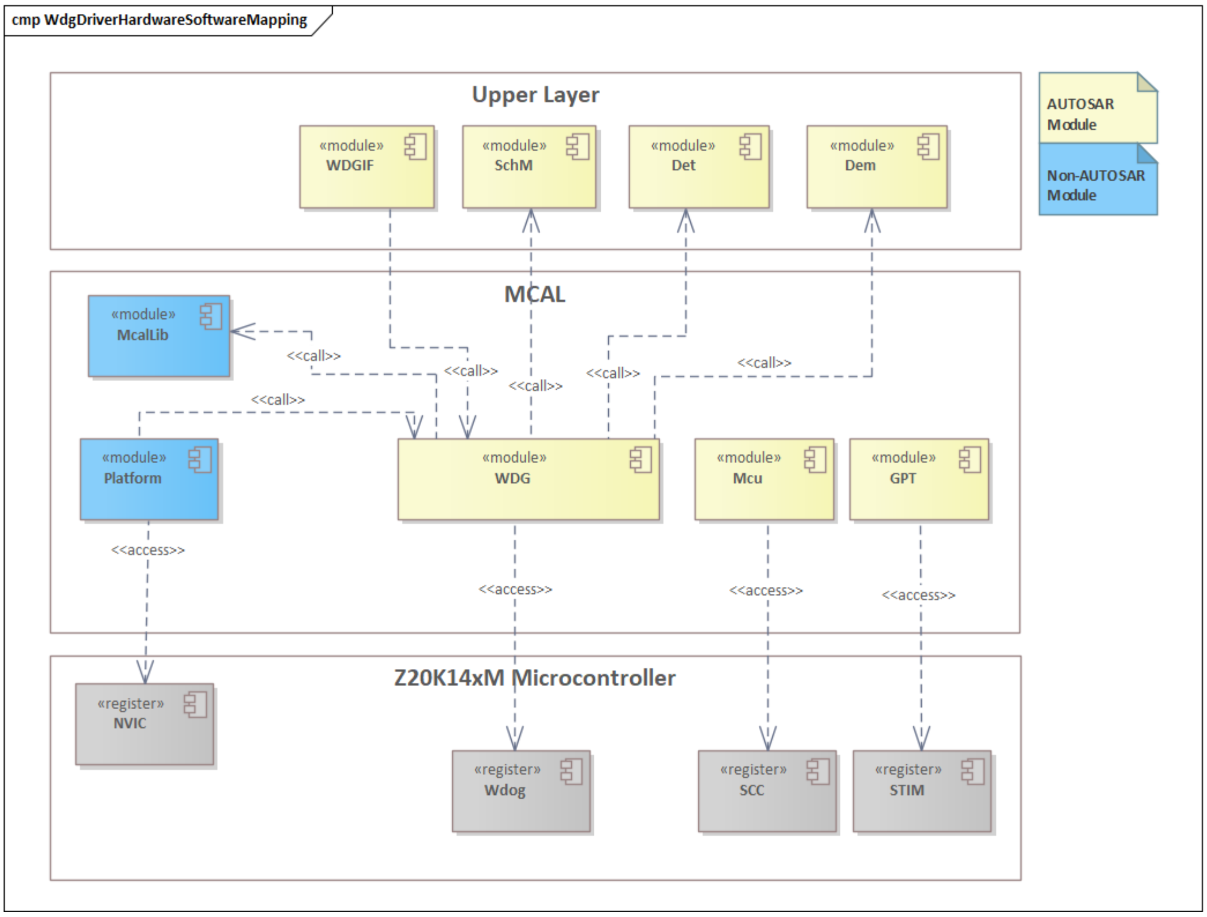


Figure 1. Hardware software mapping

5.1. WDG: primary hardware peripheral

The WDG driver uses the WDOG IP to monitor program flow. It provides a safety measure that ensures the software is running as planned and the CPU is not stuck in an infinite loop or running in an unintended code. If the WDOG module is not refreshed within a certain period, it resets the CPU.

The key hardware functional features used by the driver are:

- Configurable clock source to drive counter
- Programmable timeout period
- Robust refresh mechanism for counter clear
- Optional window mode for the acceptance of refresh sequence
- Optional interrupt before triggering CPU reset to allow post-processing for handling critical schemes
- After unlocking, the configuration registers are write-once registers to make sure the watchdog configuration cannot be overwritten by mistake
- Robust unlock mechanism for reconfiguration without resetting

5.2. STIM: primary hardware peripheral

The WDG driver uses the STIM IP for refresh WDOG hardware.

The key hardware functional features used by the driver are:

- This STIM counter clock source will clock the prescaler and glitch filter
- The divided clock by prescaler will clock the counter
- Match trigger will be generated when the counter matches a configurable value
- Each channel counter can work in the free counting mode or pulse counting mode

5.3. PARCC: dependent hardware peripheral

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

5.4. NVIC: dependent hardware peripheral

The WDG driver depends on the NVIC for interrupts controlling. These are configured through the Platform driver.

6. File structure

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

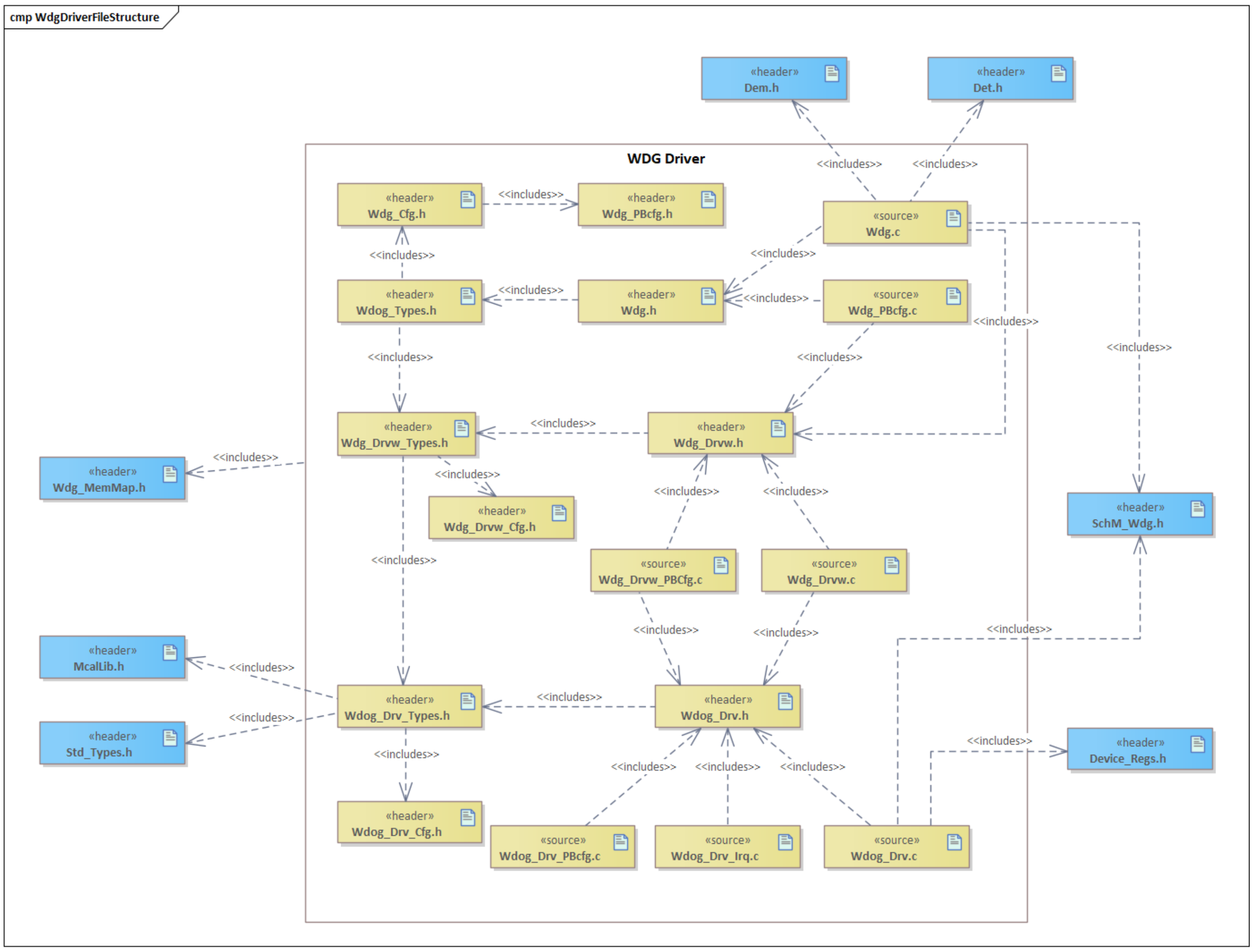


Figure 2. C File structure

File name	Description
Wdg.c	AUTOSAR level source file containing implementations of APIs
Wdg.h	AUTOSAR level header file containing declarations of APIs
Wdg_Types.h	AUTOSAR level header file containing definition of data types
Wdg_Cfg.h	AUTOSAR level configuration Header file (Generated) containing pre-processor macros
Wdg_PBcfg.c	AUTOSAR level post-build configuration source file (Generated) containing definition of the post-build configuration data structures
Wdg_PBcfg.h	AUTOSAR level post-build configuration Header file (Generated) containing declaration of the post-build configuration data structures
Wdog_Drv.c	WDG Low-level driver source file containing implementations of APIs
Wdog_Drv.h	WDG Low-level driver header file containing declarations of APIs
Wdog_Drv_Types.h	WDG Low-level driver header file containing definition of data types
Wdog_Drv_Cfg.h	WDG Low-level driver configuration Header file (Generated) containing pre-processor macros

File name	Description
Wdog_Drv_PBcfg.c	WDG Low-level driver post-build configuration source file (Generated) containing definition of the post-build configuration data structures
Wdog_Drv_PBcfg.h	WDG 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
McalLib.h	McalLib header file containing declarations of APIs and definitions of data types
SchM_Wdg.h	WDG SchM header file containing enter and exit exclusive areas function declarations of WDG driver
Wdg_MemMap.h	WDG driver memory mapping header file containing definitions of memory section
Det.h	Header file of Det module containing declarations of APIs
Dem.h	Header file of Dem module containing declarations of APIs

7. Configuration tips

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

7.1. WDG service methods

WDG driver provides two kind of methods to service the WDOG. The user can only select one of them by WDG Enable/Disable Direct Service Configuration.

One method is direct service, this method shall provide Wdg_Service() API to refresh the WDOG hardware directly.

Another method is serviced by GPT module, if this method is enabled, a GPT channel shall be configured for WDG, and the API Wdg_CallbackForGptNotification() shall be called automatically by GPT to refresh WDOG hardware.

- Enable/disable direct service shall be configured in General container:

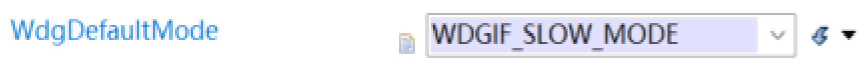
WdgEnableDirectServiceApi

Figure 3. WDG Enable/Disable Direct Service Configuration

7.2. WDG mode configuration

WDG driver supports to configure different modes. Different modes mean different WDG actions. If WdgDefaultMode is WDGIF_OFF_MODE, it means WDOG is disabled by default. If WdgDefaultMode is WDGIF_SLOW_MODE or WDGIF_FAST_MODE, user needs to configure the related parameters.

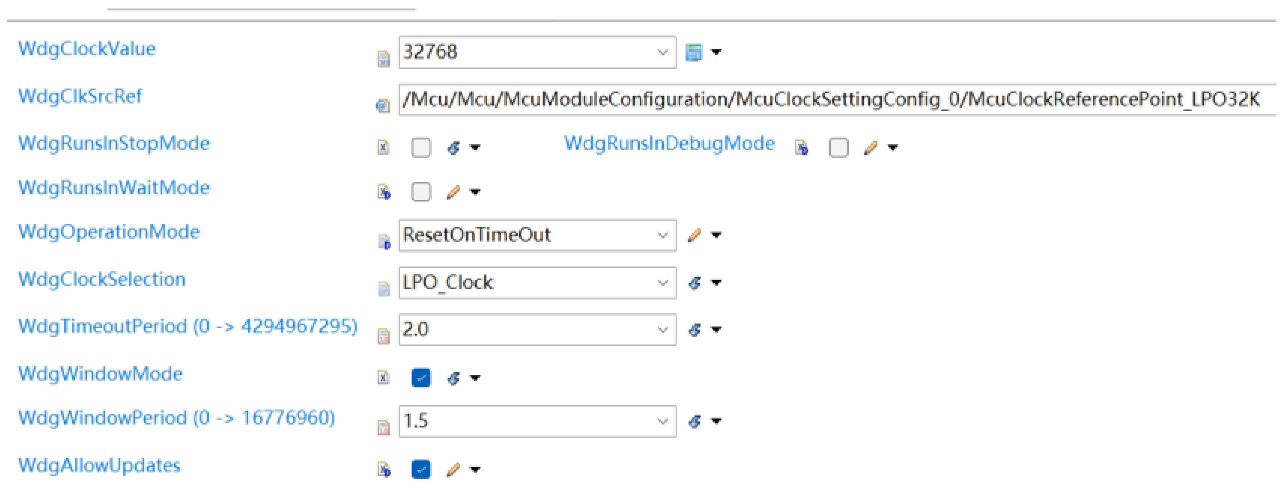
- Configure the WdgDefaultMode



WdgDefaultMode	WDGIF_SLOW_MODE

Figure 4. Configure the WdgDefaultMode

- Configure WDGIF_FAST_MODE/WDGIF_SLOW_MODE:



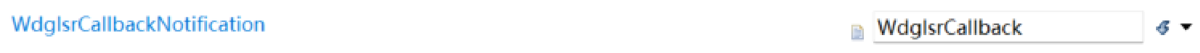
WdgClockValue	32768
WdgClkSrcRef	/Mcu/Mcu/McuModuleConfiguration/McuClockSettingConfig_0/McuClockReferencePoint_LPO32K
WdgRunsInStopMode	☐
WdgRunsInDebugMode	☐
WdgRunsInWaitMode	☐
WdgOperationMode	ResetOnTimeOut
WdgClockSelection	LPO_Clock
WdgTimeoutPeriod (0 -> 4294967295)	2.0
WdgWindowMode	✓
WdgWindowPeriod (0 -> 16776960)	1.5
WdgAllowUpdates	✓

Figure 5. Configure WDGIF_FAST_MODE/WDGIF_SLOW_MODE

7.3. WDOG interrupt notification configuration

WDG driver supports notification for WDOG interrupt. If WDOG interrupt is enabled, an interrupt will be generated before WDOG reset when WDG timeout occurs. If it is required to add some handling code before WDOG reset, user can enable WDOG interrupt in platform and configure the notification as below.

- Configure the WDOG interrupt callback notification



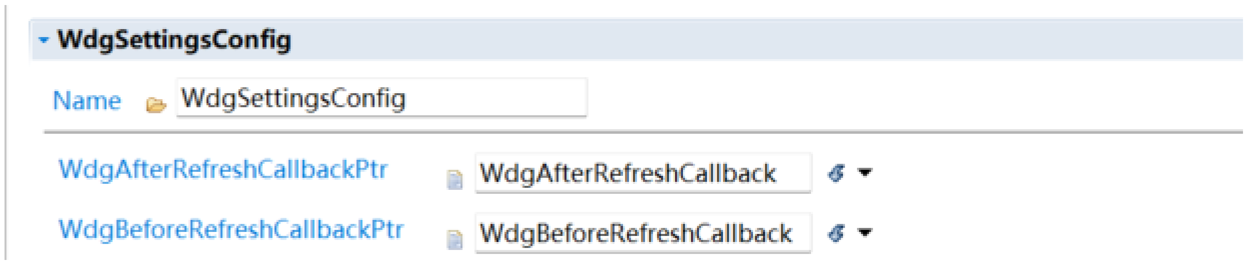
WdgIsrCallbackNotification	WdgIsrCallback

Figure 6. Configure the WDOG interrupt callback notification

7.4. WDG before/after WDOG refresh notification configuration

WDG driver supports before/after WDOG refresh notification. It is used to avoid DMA influence WDOG refresh by suspending DMA before WDOG refresh and resuming DMA after WDOG refresh.

- Configure the before/after WDOG refresh notification


WdgSettingsConfig

Name WdgSettingsConfig

WdgAfterRefreshCallbackPtr WdgAfterRefreshCallback

WdgBeforeRefreshCallbackPtr WdgBeforeRefreshCallback

Figure 7. Configure the before/after WDOG refresh notification

8. Integration hints

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

EcuM

The ECU Manager module is a BSW module that manages common aspects of ECU. Specifically, the ECU Manager module is used for initialization and de-initialization of the software drivers.

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

Det

The DET module is a BSW module that handles all detected development and runtime errors reported by the BSW modules. The WDG driver reports all the development errors to the DET module through the Det_ReportError() API, and reports 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 WDG driver uses the exclusive areas defined in SchM_Wdg.c file to protect the registers and variables access.

The SchM_Wdg.c and SchM_Wdg.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 Wdg_MemMap.h file.

The Wdg_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.

Callback and Notification

The WDG driver implements Wdg_CallbackForGptNotification() callback for GPT channel that used by WDG. The callback is invoked by GPT driver when GPT timeout.

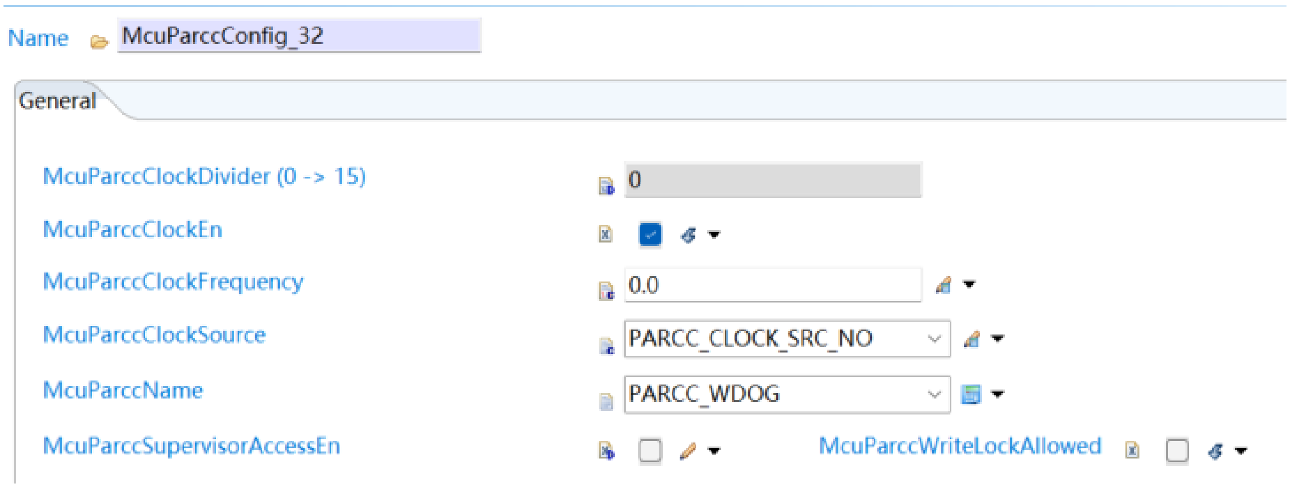
8.2. MCU support

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

The initialization of the WDG 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 WDOG peripherals used by WDG driver.



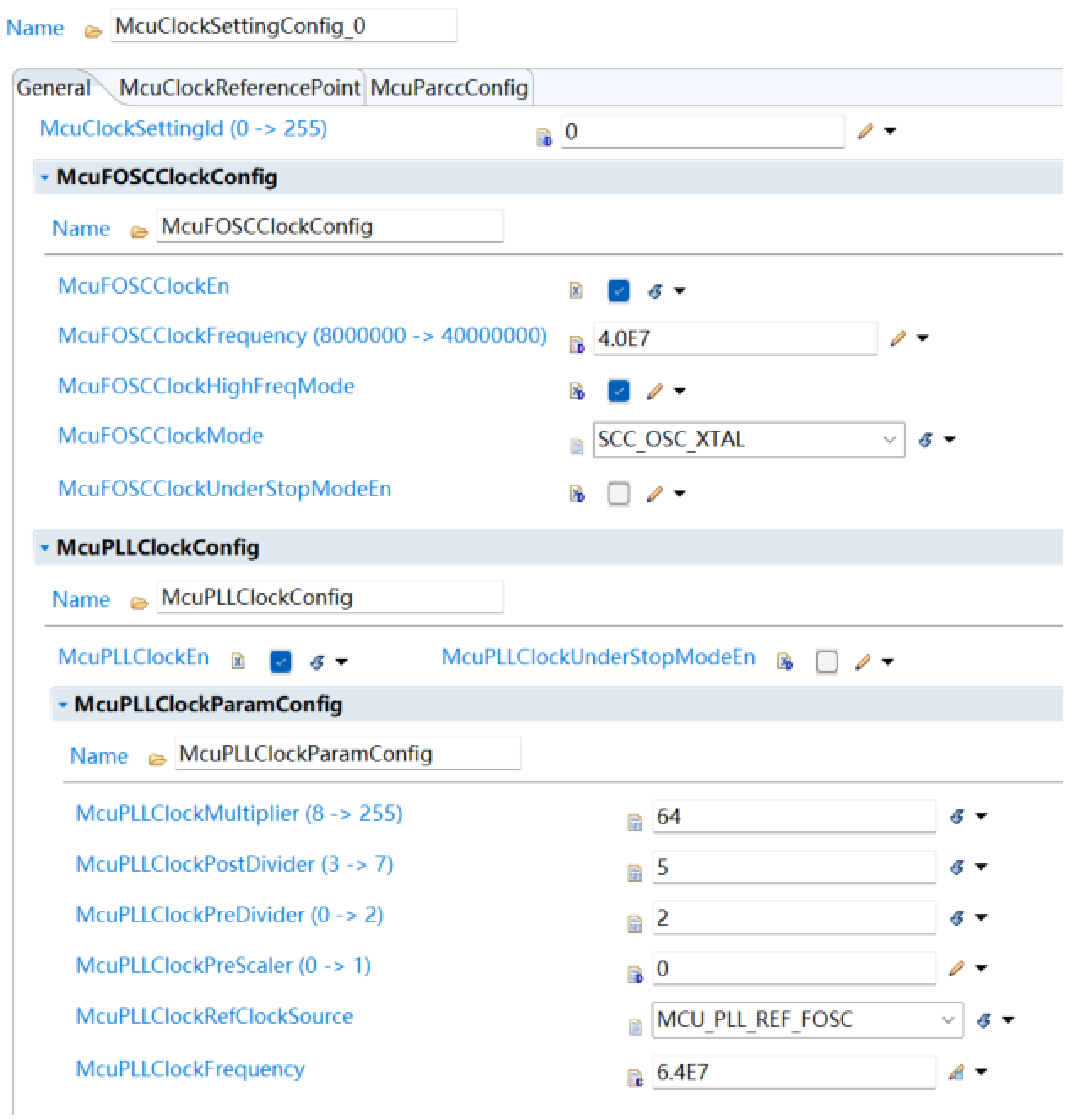
Name: McuParccConfig_32

General

McuParccClockDivider (0 -> 15)	0
McuParccClockEn	✓
McuParccClockFrequency	0.0
McuParccClockSource	PARCC_CLOCK_SRC_NO
McuParccName	PARCC_WDOG
McuParccSupervisorAccessEn	☐
McuParccWriteLockAllowed	☐

Figure 8. PARCC Configuration in MCU driver

- Configure clock reference point for WDOG peripherals used by WDG driver.


Name McuClockSettingConfig_0

General | McuClockReferencePoint | McuParccConfig

McuClockSettingId (0 -> 255) 0

McuFOSCClockConfig

Name McuFOSCClockConfig

McuFOSCClockEn ✓

McuFOSCClockFrequency (8000000 -> 40000000) 4.0E7

McuFOSCClockHighFreqMode ✓

McuFOSCClockMode SCC_OSC_XTAL

McuFOSCClockUnderStopModeEn ☐

McuPLLClockConfig

Name McuPLLClockConfig

McuPLLClockEn ✓ McuPLLClockUnderStopModeEn ☐

McuPLLClockParamConfig

Name McuPLLClockParamConfig

McuPLLClockMultiplier (8 -> 255) 64

McuPLLClockPostDivider (3 -> 7) 5

McuPLLClockPreDivider (0 -> 2) 2

McuPLLClockPreScaler (0 -> 1) 0

McuPLLClockRefClockSource MCU_PLL_REF_FOSC

McuPLLClockFrequency 6.4E7

Figure 9. Clock Reference Point configuration in MCU driver

- Select clock reference point for WDOG hardware unit.

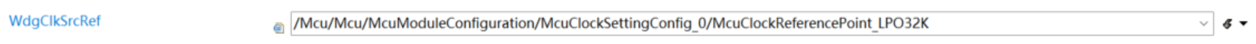

WdgClkSrcRef /Mcu/Mcu/McuModuleConfiguration/McuClockSettingConfig_0/McuClockReferencePoint_LPO32K

Figure 10. Select clock reference for WDOG hardware unit

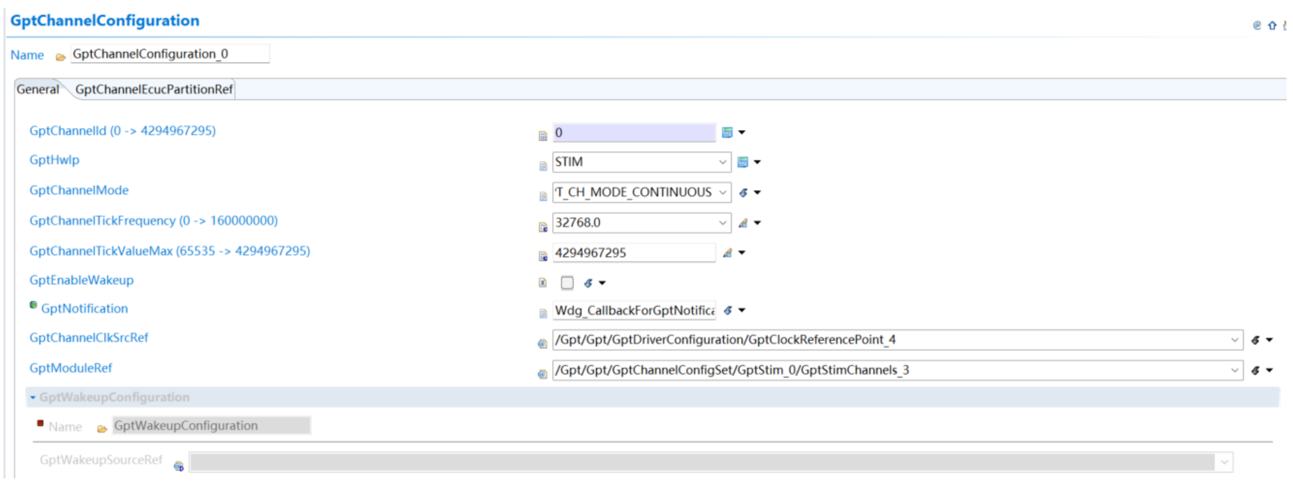
8.3. GPT support

The GPT driver is primarily responsible for providing a GPT channel for WDG. The WDG driver depends on GPT driver to refresh the WDOG hardware if direct service is disabled in WDG configuration.

The initialization of the WDG driver must be started only after completion of the GPT initialization.

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

- Configure GPT for WDOG peripherals used by WDG driver.



GptChannelConfiguration

Name: GptChannelConfiguration_0

General | GptChannelEcucPartitionRef

GptChannelId (0 -> 4294967295)	0
GptHwIp	STIM
GptChannelMode	T_CH_MODE_CONTINUOUS
GptChannelTickFrequency (0 -> 160000000)	32768.0
GptChannelTickValueMax (65535 -> 4294967295)	4294967295
GptEnableWakeup	☐
GptNotification	Wdg_CallbackForGptNotifica
GptChannelClkSrcRef	/Gpt/Gpt/GptDriverConfiguration/GptClockReferencePoint_4
GptModuleRef	/Gpt/Gpt/GptChannelConfigSet/GptStim_0/GptStimChannels_3

GptWakeupConfiguration

Name: GptWakeupConfiguration

GptWakeupSourceRef

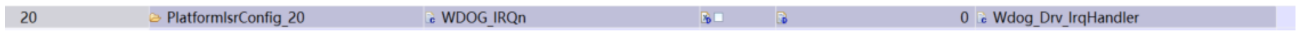
Figure 11. GPT Configuration for WDG driver

8.4. Interrupt connections

The interrupt connections of the WDG driver are described in this section.

Interrupt enabled

Corresponding interrupts of WDG shall be enabled in platform module or OS module(use AUTOSAR OS).



20	PlatformIsrConfig_20	WDOG_IRQn		0	Wdog_Drv_IrqHandler

Figure 12. Configure WDG interrupts in Platform module

An example code of WDG ISR is provided in Wdog_Drv_Irq.c as below:

```
#if (WDOG_DRV_ISR_ENABLED == STD_ON)
ISR(Wdog_Drv_IrqHandler)
{
    Wdog_Drv_IntHandler();
    EXIT_INTERRUPT();
}
#endif
```

9. Assumptions of Use

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

Correct configuration for GPT

The user shall ensure that the GPT have been configured correctly and one channel have been configured for WDG.

Correct pointer of configuration structure

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

How to Reach Us:

Home Page:

zhixin-semi.com/

Document Number: Z20K14xM-MCAL-WDG-UM

Revision 1.0.0, April, 2023