MCAL Configuration Verification Manual for Gpt

32-bit TriCore™ AURIX™ TC3xx microcontroller family

About this document

Scope and purpose

This Configuration Data Reference document is applicable to all TC3xx devices in the TriCore™ AURIX™ family of 32-bit microcontrollers.

The purpose of this document is to facilitate the integrator to verify the generated code based on the input configuration parameters. This document describes details of structures, defines, macros and variables generated from the configuration parameters.

Intended audience

This document is intended for integrators who need to understand the logic of the generated configuration code of AURIX™ AUTOSAR MCAL.

Reference documents

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

- AURIX™ TC3xx MCAL User Manual Gpt

Table of contents

1 Gpt driver

This chapter describes the details of the configuration data generated from the Gpt driver.

1.1 File: Gpt_Cfg.h

The generated header file contains all pre-compile configuration parameters. Pre-compile time configuration allows decoupling of the static configuration from implementation. The file is generated in 'inc' folder.

1.1.1 Macro: GPT_AR_RELEASE_MAJOR_VERSION

Table 1 GPT_AR_RELEASE_MAJOR_VERSION

Name	GPT_AR_RELEASE_MAJOR_VERSION	
Description	Major version number of AUTOSAR release on which the Gpt implementation is based on.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/ArMajorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Gpt_Cfg.h file with ArMajorVersion 4	`#define GPT_AR_RELEASE_MAJOR_VERSION (4U)`

1.1.2 Macro: GPT_AR_RELEASE_MINOR_VERSION

Table 2 GPT_AR_RELEASE _MINOR_VERSION

Name	GPT_AR_RELEASE _MINOR_VERSION	
Description	Minor version number of AUTOSAR release on which the GPT implementation is based on.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/ArMinorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Gpt_Cfg.h file with ArMinorVersion 2	`#define GPT_AR_RELEASE_MINOR_VERSION (2U)`

1.1.3 Macro: GPT_AR_RELEASE_REVISION_VERSION

Table 3 GPT_AR_RELEASE_REVISION_VERSION

Name	GPT_AR_RELEASE_REVISION_VERSION

Description	Revision version number of AUTOSAR release on which the Gpt implementation is based on.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/ArPatchVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Gpt_Cfg.h file with ArPatchVersion 2	`#define GPT_AR_RELEASE_REVISION_VERSION (2U)`

1.1.4 Macro: GPT_SW_MAJOR_VERSION

Table 4 **GPT_SW_MAJOR_VERSION**

Name	GPT_SW_MAJOR_VERSION	
Description	Major version number of the GPT module.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/SwMajorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Gpt_Cfg.h file with SwMajorVersion 10	`#define GPT_SW_MAJOR_VERSION (10U)`

1.1.5 Macro: GPT_SW_MINOR_VERSION

Table 5 **GPT_SW_MINOR_VERSION**

Name	GPT_SW_MINOR_VERSION	
Description	Minor version number of the Gpt module.	
Verification method	The macro is generated with the value present in 'CommonPublishedInformation/SwMinorVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Gpt_Cfg.h file with SwMinorVersion 10	`#define GPT_SW_MINOR_VERSION (10U)`

1.1.6 Macro: GPT_SW_PATCH_VERSION

Table 6 **GPT_SW_PATCH_VERSION**

Name	GPT_SW_PATCH_VERSION
Description	Patch level version number of the Gpt module.

Verification method	The macro is generated with the value present in 'CommonPublishedInformation/SwPatchVersion'. *Note: The macro is not user configurable.*	
Example(s)	**Action**	**Generated output**
	Generate Gpt_Cfg.h file with SwPatchVersion 0	`#define GPT_SW_PATCH_VERSION (0U)`

1.1.7 Macro: GPT_SAFETY_ENABLE

Table 7 GPT_SAFETY_ENABLE

Name	GPT_SAFETY_ENABLE	
Description	Enables/disables safety features	
Verification method	The macro is generated as STD_ON if GptSafetyEnable configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptSafetyEnable = True	`#define GPT_SAFETY_ENABLE (STD_ON)`
	GptSafetyEnable = False	`#define GPT_SAFETY_ENABLE (STD_OFF)`

1.1.8 Macro: GPT_INITCHECK_API

Table 8 GPT_INITCHECK_API

Name	GPT_INITCHECK_API	
Description	Enables/disables Gpt_InitCheck API	
Verification method	The macro is generated as STD_ON if GptInitCheckApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptInitCheckApi = True	`#define GPT_INITCHECK_API  (STD_ON)`
	GptInitCheckApi = False	`#define GPT_INITCHECK_API  (STD_OFF)`

1.1.9 Macro: GPT_VERSION_INFO_API

Table 9 GPT_VERSION_INFO_API

Name	GPT_VERSION_INFO_API	
Description	Enables/disables Gpt_GetVersionInfo API	
Verification method	The macro is generated as STD_ON if GptVersionInfoApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptVersionInfoApi = True	`#define GPT_VERSION_INFO_API`

		`(STD_ON)`
	GptVersionInfoApi = False	`#define GPT_VERSION_INFO_API (STD_OFF)`

1.1.10 Macro: GPT_TIME_ELAPSED_API

Table 10 GPT_TIME_ELAPSED_API

Name	GPT_TIME_ELAPSED_API	
Description	Enables/disables Gpt_GetTimeElapsed API	
Verification method	The macro is generated as STD_ON if GptTimeElapsedApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptTimeElapsedApi = True	`#define GPT_TIME_ELAPSED_API (STD_ON)`
	GptTimeElapsedApi = False	`#define GPT_TIME_ELAPSED_API (STD_OFF)`

1.1.11 Macro: GPT_TIME_REMAINING_API

Table 11 GPT_TIME_REMAINING_API

Name	GPT_TIME_REMAINING_API	
Description	Enables/disables Gpt_GetTimeRemaining API	
Verification method	The macro is generated as STD_ON if GptTimeRemainingApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptTimeRemainingApi = True	`#define GPT_TIME_REMAINING_API (STD_ON)`
	GptTimeRemainingApi = False	`#define GPT_TIME_REMAINING_API (STD_OFF)`

1.1.12 Macro: GPT_ENABLE_DISABLE_NOTIFICATION_API

Table 12 GPT_ENABLE_DISABLE_NOTIFICATION_API

Name	GPT_ENABLE_DISABLE_NOTIFICATION_API	
Description	Enables/disables Gpt_EnableNotification and Gpt_DisableNotification APIs	
Verification method	The macro is generated as STD_ON if GptEnableDisableNotificationApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**

	GptEnableDisableNotificationApi = True	`#define GPT_ENABLE_DISABLE_NOTIFICATION_API (STD_ON)`
	GptEnableDisableNotificationApi = False	`#define GPT_ENABLE_DISABLE_NOTIFICATION_API (STD_OFF)`

1.1.13 Macro: GPT_WAKEUP_FUNCTIONALITY_API

Table 13 GPT_WAKEUP_FUNCTIONALITY_API

Name	GPT_WAKEUP_FUNCTIONALITY_API	
Description	Enables/disables the following wakeup related APIs Gpt_EnableWakeup Gpt_DisableWakeup Gpt_SetMode Gpt_CheckWakeup	
Verification method	The macro is generated as STD_ON if GptWakeupFunctionalityApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptWakeupFunctionalityApi = True	`#define GPT_WAKEUP_FUNCTIONALITY_API (STD_ON)`
	GptWakeupFunctionalityApi = False	`#define GPT_WAKEUP_FUNCTIONALITY_API (STD_OFF)`

1.1.14 Macro: GPT_DEINIT_API

Table 14 GPT_DEINIT_API

Name	GPT_DEINIT_API	
Description	Enables/disables Gpt_InitCheck API	
Verification method	The macro is generated as STD_ON if GptDeinitApi configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptDeinitApi = True	`#define GPT_DEINIT_API  (STD_ON)`
	GptDeinitApi = False	`#define GPT_DEINIT_API  (STD_OFF)`

1.1.15 Macro: GPT_DEV_ERROR_DETECT

Table 15 GPT_DEV_ERROR_DETECT

Name	GPT_DEV_ERROR_DETECT	
Description	Enables/disables the Development Error Detection.	
Verification method	The macro is generated as STD_ON if GptDevErrorDetect configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptDevErrorDetect = True	`#define GPT_DEV_ERROR_DETECT (STD_ON)`
	GptDevErrorDetect = False	`#define GPT_DEV_ERROR_DETECT (STD_OFF)`

1.1.16 Macro: GPT_MULTICORE_ERROR_DETECT

Table 16 GPT_MULTICORE_ERROR_DETECT

Name	GPT_MULTICORE_ERROR_DETECT	
Description	Enables/disables MultiCore DET Check	
Verification method	The macro is generated as STD_ON if GptMultiCoreErrorDetect configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptMultiCoreErrorDetect = True	`#define GPT_MULTICORE_ERROR_DETECT (STD_ON)`
	GptMultiCoreErrorDetect = False	`#define GPT_MULTICORE_ERROR_DETECT (STD_OFF)`

1.1.17 Macro: GPT_REPORT_WAKEUP_SOURCE

Table 17 GPT_REPORT_WAKEUP_SOURCE

Name	GPT_REPORT_WAKEUP_SOURCE	
Description	Enables/disables the wakeup source reporting.	
Verification method	The macro is generated as STD_ON if GptReportWakeupSource configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptReportWakeupSource = True	`#define GPT_REPORT_WAKEUP_SOURCE (STD_ON)`
	GptReportWakeupSource = False	`#define GPT_REPORT_WAKEUP_SOURCE (STD_OFF)`

1.1.18 Macro: GPT_PREDEF_TIMER_100US_32BIT_EN

Table 18 GPT_PREDEF_TIMER_100US_32BIT_EN

Name	GPT_PREDEF_TIMER_100US_32BIT_EN	
Description	Enables/disables 100us predefined timer.	
Verification method	The macro is generated as STD_ON if GptPredefTimer100us32bitEnable configuration parameter is set to 'True' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptPredefTimer100us32bitEnable = True	`#define GPT_PREDEF_TIMER_100US_32BIT_EN (STD_ON)`
	GptPredefTimer100us32bitEnable = False	`#define GPT_PREDEF_TIMER_100US_32BIT_EN (STD_OFF)`

1.1.19 Macro: GPT_PREDEF_EXTRA_CH_100US

Table 19 GPT_PREDEF_EXTRA_CH_100US

Name	GPT_PREDEF_EXTRA_CH_100US	
Description	Indicates the usage of additional TOM channel for frequency tuning	
User configurable	No	
Verification method	The macro is generated as STD_ON if the predef channel frequency cannot be derived directly from the GTM clock and an additional TOM channel is used to derive the required 10KHz frequency The macro is generated as STD_OFF the predef channel frequency can be derived directly from the GTM clock.	
Example(s)	**Action**	**Generated output**
	Input clock to the TOM is 1MHz	`#define GPT_PREDEF_EXTRA_CH_100US (STD_ON)`
	Input clock to the TOM is 10KHz	`#define GPT_PREDEF_EXTRA_CH_100US (STD_OFF)`

1.1.20 Macro: GPT_PREDEF_TIMER_1US_32BIT_EN

Table 20 GPT_PREDEF_TIMER_1US_32BIT_EN

Name	GPT_PREDEF_TIMER_1US_32BIT_EN	
Description	Enables/disables 1us 32bit predefined timer.	
Verification method	The macro is generated as STD_ON if GptPredefTimer1usEnablingGrade configuration parameter is set to 'GPT_PREDEF_TIMER_1US_16_24_32BIT_ENABLED' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**

	GptPredefTimer1usEnablingGrade = GPT_PREDEF_TIMER_1US_16_24_32BIT_ENABLED	`#define GPT_PREDEF_TIMER_1US_32BIT_EN (STD_ON)`
	GptPredefTimer1usEnablingGrade = GPT_PREDEF_TIMER_1US_16_24BIT_ENABLED	`#define GPT_PREDEF_TIMER_1US_32BIT_EN (STD_OFF)`
	GptPredefTimer1usEnablingGrade = GPT_PREDEF_TIMER_1US_16BIT_ENABLED	`#define GPT_PREDEF_TIMER_1US_32BIT_EN (STD_OFF)`
	GptPredefTimer1usEnablingGrade = GPT_PREDEF_TIMER_1US_DISABLED	`#define GPT_PREDEF_TIMER_1US_32BIT_EN (STD_OFF)`

1.1.21 Macro: GPT_PREDEF_TIMER_1US_24BIT_EN

Table 21 GPT_PREDEF_TIMER_1US_24BIT_EN

Name	GPT_PREDEF_TIMER_1US_24BIT_EN	
Description	Enables/disables 1us 24bit predefined timer.	
Verification method	The macro is generated as STD_ON if GptPredefTimer1usEnablingGrade configuration parameter is set to 'GPT_PREDEF_TIMER_1US_16_24_32BIT_ENABLED' or 'GPT_PREDEF_TIMER_1US_16_24BIT_ENABLED' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptPredefTimer1usEnablingGrade = GPT_PREDEF_TIMER_1US_16_24_32BIT_ENABLED	`#define GPT_PREDEF_TIMER_1US_24BIT_EN (STD_ON)`
	GptPredefTimer1usEnablingGrade = GPT_PREDEF_TIMER_1US_16_24BIT_ENABLED	`#define GPT_PREDEF_TIMER_1US_24BIT_EN (STD_ON)`
	GptPredefTimer1usEnablingGrade = GPT_PREDEF_TIMER_1US_16BIT_ENABLED	`#define GPT_PREDEF_TIMER_1US_24BIT_EN (STD_OFF)`
	GptPredefTimer1usEnablingGrade = GPT_PREDEF_TIMER_1US_DISABLED	`#define GPT_PREDEF_TIMER_1US_24BIT_EN (STD_OFF)`

1.1.22 Macro: GPT_PREDEF_TIMER_1US_16BIT_EN

Table 22 GPT_PREDEF_TIMER_1US_16BIT_EN

Name	GPT_PREDEF_TIMER_1US_16BIT_EN	
Description	Enables/disables 1us, 16bit predefined timer	
Verification method	The macro is generated as STD_ON if GptPredefTimer1usEnablingGrade configuration parameter is set to any one of the following values 'GPT_PREDEF_TIMER_1US_16BIT_ENABLED' or 'GPT_PREDEF_TIMER_1US_16_24BIT_ENABLED' or 'GPT_PREDEF_TIMER_1US_16_24_32BIT_ENABLED' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptPredefTimer1usEnablingGrade = GPT_PREDEF_TIMER_1US_16_24_32BIT_ENABLED	`#define GPT_PREDEF_TIMER_1US_16BIT_EN (STD_ON)`
	GptPredefTimer1usEnablingGrade = GPT_PREDEF_TIMER_1US_16_24BIT_ENABLED	`#define GPT_PREDEF_TIMER_1US_16BIT_EN (STD_ON)`
	GptPredefTimer1usEnablingGrade = GPT_PREDEF_TIMER_1US_16BIT_ENABLED	`#define GPT_PREDEF_TIMER_1US_16BIT_EN (STD_ON)`
	GptPredefTimer1usEnablingGrade = GPT_PREDEF_TIMER_1US_DISABLED	`#define GPT_PREDEF_TIMER_1US_16BIT_EN (STD_OFF)`

1.1.23 Macro: GPT_PREDEF_EXTRA_CH_1US

Table 23 GPT_PREDEF_EXTRA_CH_1US

Name	GPT_PREDEF_EXTRA_CH_1US	
Description	Represents the usage of additional TOM channel for frequency tuning	
User configurable	No	
Verification method	The macro is generated as STD_ON if the predef channel frequency cannot be derived directly from the GTM clock and an additional TOM channel is used to derive the required 1MHz frequency The macro is generated as STD_OFF if the predef channel frequency cannot be derived directly from the GTM clock.	
Example(s)	**Action**	**Generated output**
	Input clock to the TOM is 2MHz	`#define GPT_PREDEF_EXTRA_CH_100US (STD_ON)`

	Input clock to the TOM is 1MHz	`#define GPT_PREDEF_EXTRA_CH_100US (STD_OFF)`

1.1.24 Macro: GPT_ONESHOT_USED

Table 24 GPT_ONESHOT_USED

Name	GPT_ONESHOT_USED	
Description	Enables/disables one shot mode	
Verification method	The macro is generated as STD_ON if GptChannelMode configuration parameter is set to 'GPT_CH_MODE_ONESHOT' else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	GptChannelMode = GPT_CH_MODE_ONESHOT	`#define GPT_ONESHOT_USED (STD_ON)`
	GptChannelMode = GPT_CH_MODE_CONTINUOUS	`#define GPT_ONESHOT_USED (STD_OFF)`

1.1.25 Macro: GPT_MAX_CHANNELS

Table 25 GPT_MAX_CHANNELS

Name	GPT_MAX_CHANNELS	
Description	Indicates the total number of Gpt normal channels configured.	
User configurable	No	
Verification method	The macro is generated as a numeric value which corresponds to the total number of Gpt normal channels configured. *Note: Predef timers are not considered for calculating GPT_MAX_CHANNELS.*	
Example(s)	**Action**	**Generated output**
	Configure three channels in any core (GptChannelConfiguration_0 to GptChannelConfiguration_2). Configure both predefined timers (GptChannelConfiguration_3 and GptChannelConfiguration_4). Configuring three channels: Allocating TOM in Mcu module: Select	`#define GPT_MAX_CHANNELS (3U)`

'GTM_TOM_CHANNEL_USED_BY_GPT' in the following path /Mcu/Mcu/McuHardwareResourceAllocationConf_0/McuGtmAllocationConf_0/McuGtmTomAllocationConf_0/McuGtmTomChannelAllocationConf_0 Allocating ATOM in Mcu module: Select 'GTM_ATOM_CHANNEL_USED_BY_GPT' in the following path /Mcu/Mcu/McuHardwareResourceAllocationConf_0/McuGtmAllocationConf_0/McuGtmAtomAllocationConf_0/McuGtmAtomChannelAllocationConf_0 Allocating GPT12 in Mcu module: Select 'GPT_TIMER_USED_BY_GPT_DRIVER' in the following path /Mcu/Mcu/McuHardwareResourceAllocationConf_0/McuGpt12ModuleAllocationConf_0 In Gpt module Select the TOM/ATOM allocated to Gpt by the Mcu module, for Gpt\GptChannelConfigSet\GptChannelConfiguration\GtmTimerOutputModuleConfiguration\GtmTimerUsed Select the GPT12 allocated to Gpt by the Mcu module, for Gpt\GptChannelConfigSet\GptChannelConfiguration\Gpt12TimerOutputModuleConfiguration\Gpt12TimerUsed Allocating the channel to a given core in resource manager Select GPT for	

	ResourceM/ResourceMMcalConfig/ResourceMMcalConfig_0/ResourceMMcalCore/ResourceMMcalCore_0/ResourceMAllocation/ResourceMAllocation_0/ResourceMModuleName, ResourceM/ResourceMMcalConfig/ResourceMMcalConfig_0/ResourceMMcalCore/ResourceMMcalCore_0/ResourceMAllocation/ResourceMAllocation_1/ResourceMModuleName and ResourceM/ResourceMMcalConfig/ResourceMMcalConfig_0/ResourceMMcalCore/ResourceMMcalCore_0/ResourceMAllocation/ResourceMAllocation_2/ResourceMModuleName Select the required Gpt channel under ResourceM/ResourceMMcalConfig/ResourceMMcalConfig_0/ResourceMMcalCore/ResourceMMcalCore_0/ResourceMAllocation/ResourceMAllocation_0/ResourceMResourceRef, ResourceM/ResourceMMcalConfig/ResourceMMcalConfig_0/ResourceMMcalCore/ResourceMMcalCore_0/ResourceMAllocation/ResourceMAllocation_1/ResourceMResourceRef and ResourceM/ResourceMMcalConfig/ResourceMMcalConfig_0/ResourceMMcalCore/ResourceMMcalCore_0/ResourceMAllocation/ResourceMAllocation_2/ResourceMResourceRef	
	Configure six channels in core0	`#define GPT_MAX_CHANNELS (6U)`

1.1.26 Macro: GPT_MAX_CHANNELS_CORE<x>

Table 26 GPT_MAX_CHANNELS_CORE<x>

Name	GPT_MAX_CHANNELS_CORE<x>	
Description	Indicates the number of Gpt normal channels mapped to core<x>	
Verification method	The macro is generated as total number of Gpt normal channels allocated to CORE<x>. *Note:* *Channels not assigned to any core are assigned to master core (ResourceMMasterCore).*	
Example(s)	**Action**	**Generated output**

	• Configure 6 Gpt channels(GptChannelConfiguration_0 to GptChannelConfiguration_5
	• Set ResourceMMasterCore as CORE1 in ResourceM/ResourceMMcalConfig/*[1]/ResourceMMasterCore.
	• Do not assign Gpt channels in any ResourceMAllocation
		`#define GPT_MAX_CHANNELS_CORE0 (0U)`
		`#define GPT_MAX_CHANNELS_CORE1 (6U)`
		`#define GPT_MAX_CHANNELS_CORE2 (0U)`
		`#define GPT_MAX_CHANNELS_CORE3 (0U)`
		`#define GPT_MAX_CHANNELS_CORE4 (0U)`
		`#define GPT_MAX_CHANNELS_CORE5 (0U)`

	• Configure 16 Gpt channels (GptChannelConfiguration_0 to GptChannelConfiguration_15).
	• Assign GptChannelConfiguration_0 under ResourceMAllocation with ResourceMCoreID as CORE0.
	• Assign GptChannelConfiguration_1, GptChannelConfiguration_2 under ResourceMAllocation with ResourceMCoreID as CORE2.
	• Assign GptChannelConfiguration_3, GptChannelConfiguration_4, GptChannelConfiguration_5 under ResourceMAllocation with ResourceMCoreID as CORE3.
	• Assign GptChannelConfiguration_6,
		`#define GPT_MAX_CHANNELS_CORE0 (1U)`
		`#define GPT_MAX_CHANNELS_CORE1 (1U)`
		`#define GPT_MAX_CHANNELS_CORE2 (2U)`
		`#define GPT_MAX_CHANNELS_CORE3 (3U)`
		`#define GPT_MAX_CHANNELS_CORE4 (4U)`
		`#define GPT_MAX_CHANNELS_CORE5 (5U)`
		
		Note: *Observe core1 which is master core is having one channel allocated though not configured*

	GptChannelConfiguration_7, GptChannelConfiguration_8, GptChannelConfiguration_9 under ResourceMAllocation with ResourceMCoreID as CORE4. • Assign GptChannelConfiguration_10, GptChannelConfiguration_11, GptChannelConfiguration_12, GptChannelConfiguration_13, GptChannelConfiguration_14 under ResourceMAllocation with ResourceMCoreID as CORE5	

1.1.27 Macro:GPT_CONFIGURED_CORE<x>

Table 27 GPT_CONFIGURED_CORE<x>

Name	GPT_CONFIGURED_CORE<x>	
Description	Indicates whether the core has any Gpt normal channels allocated to it or not.	
User configurable	No	
Verification method	The macro is generated as STD_ON if at least one Gpt normal channel is associated with the core<x> else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**

	• Configure only one Gpt channel (GptChannelConfiguration_0)
	• Set ResourceMMasterCore as CORE1.
	• Do not assign Gpt channels in any ResourceMAllocation
		`#define GPT_CONFIGURED_CORE0 (STD_OFF)`
		`#define GPT_CONFIGURED_CORE1 (STD_ON)`
		`#define GPT_CONFIGURED_CORE2 (STD_OFF)`
		`#define GPT_CONFIGURED_CORE3 (STD_OFF)`
		`#define GPT_CONFIGURED_CORE4 (STD_OFF)`
		`#define GPT_CONFIGURED_CORE5 (STD_OFF)`

	Assign GptChannelConfiguration_0 under ResourceMAllocation with ResourceMCoreID as	`#define GPT_CONFIGURED_CORE0 (STD_ON)`
		`#define GPT_CONFIGURED_CORE1`

	CORE0	(STD_OFF) #define GPT_CONFIGURED_CORE2 (STD_OFF) #define GPT_CONFIGURED_CORE3 (STD_OFF) #define GPT_CONFIGURED_CORE4 (STD_OFF) #define GPT_CONFIGURED_CORE5 (STD_OFF)
	Assign GptChannelConfiguration_0 under ResourceMAllocation with ResourceMCoreID as CORE3	#define GPT_CONFIGURED_CORE0 (STD_OFF) #define GPT_CONFIGURED_CORE1 (STD_OFF) #define GPT_CONFIGURED_CORE2 (STD_OFF) #define GPT_CONFIGURED_CORE3 (STD_ON) #define GPT_CONFIGURED_CORE4 (STD_OFF) #define GPT_CONFIGURED_CORE5 (STD_OFF)
	Assign GptChannelConfiguration_0 under ResourceMAllocation with ResourceMCoreID as CORE5	#define GPT_CONFIGURED_CORE0 (STD_OFF) #define GPT_CONFIGURED_CORE1 (STD_OFF) #define GPT_CONFIGURED_CORE2 (STD_OFF) #define GPT_CONFIGURED_CORE3 (STD_OFF) #define GPT_CONFIGURED_CORE4 (STD_OFF) #define GPT_CONFIGURED_CORE5 (STD_ON)

1.1.28 Macro: GPT_MAX_CORE_USED

Table 28 GPT_MAX_CORE_USED

Name	GPT_MAX_CORE_USED	
Description	Indicates the total number of cores configured.	
Verification method	The macro is generated as a numeric value which corresponds to the total number of cores for which GPT channels have been associated.	
Example(s)	**Action**	**Generated output**
	• Configure 6 Gpt	#define GPT_MAX_CORE_USED

	channels(GptChannelConfiguration_0 to GptChannelConfiguration_5) • Set ResourceMMasterCore as CORE1. • Do not assign Gpt channels in any ResourceMAllocation	`(1U)`
	GptTimeElapsedApi = False	`#define GPT_TIME_ELAPSED_API (STD_OFF)`
	Assign all of the available channels under ResourceMAllocation with ResourceMCoreID as CORE0	`#define GPT_MAX_CORE_USED (1U)`
	Assign GptChannelConfiguration_5 under ResourceMAllocation with ResourceMCoreID as CORE1	`#define GPT_MAX_CORE_USED (2U)`
	Assign GptChannelConfiguration_2, GptChannelConfiguration_3 under ResourceMAllocation with ResourceMCoreID as CORE5	`#define GPT_MAX_CORE_USED (3U)`
	Assign one channel in each core	`#define GPT_MAX_CORE_USED (6U)`

1.1.29 Macro: GPT_TOM_USED

Table 29 GPT_TOM_USED

Name	GPT_TOM_USED	
Description	Indicates the usage of TOM channel in the driver	
Verification method	The macro is generated as STD_ON if atleat one TOM channel is allocated using GtmTimerUsed, else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	Configure at least one Gpt channel to TOM channel	`#define GPT_TOM_USED (STD_ON)`
	Configure all of the GPT channels to ATOM/GPT12 and none to TOM	`#define GPT_TOM_USED (STD_OFF)`

1.1.30 Macro: GPT_ATOM_USED

Table 30 GPT_ATOM_USED

Name	GPT_ATOM_USED	
Description	Indicates the usage of TOM channel in the driver	
Verification method	The macro is generated as STD_ON if atleast one ATOM channel is allocated using GtmTimerUsed, else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	Configure at least one Gpt channel to ATOM channel	`#define GPT_ATOM_USED (STD_ON)`
	Configure all of the GPT channels to TOM/GPT12 and none to ATOM	`#define GPT_ATOM_USED (STD_OFF)`

1.1.31 Macro: GPT_GPT12_USED

Table 31 GPT_GPT12_USED

Name	GPT_GPT12_USED	
Description	Indicates the usage of GPT12 timer in the driver	
Verification method	The macro is generated as STD_ON if atleast one GPT12 timer is allocated using Gpt12TimerUsed, else the macro is generated as STD_OFF.	
Example(s)	**Action**	**Generated output**
	Configure at least one Gpt channel to GPT12 channel	`#define GPT_GPT12_USED (STD_ON)`
	Configure all of the GPT channels to TOM/ATOM and none to GPT12	`#define GPT_GPT12_USED (STD_OFF)`

1.1.32 Macro: GptConf_GptChannelConfiguration_<channel name>

Table 32 GptConf_GptChannelConfiguration_<channel name>

Name	GptConf_GptChannelConfiguration_<channel name>	
Description	The macro is the symbolic name generated for the configuration parameter 'GptChannelConfigSet/GptChannelConfiguration/GptChannelId'	
Verification method	The macro is generated as a numeric value which is configured in 'GptChannelConfigSet/GptChannelConfiguration/GptChannelId'. <channel name> is the name of the Gpt channel's container name.	
Example(s)	**Action**	**Generated output**
	• Configure 3 Gpt channels (GptChannelConfiguration_0 , GptChannelConfiguration_1	`#define GptConf_GptChannelConfiguration_GptChannelConfiguration_0`

	and GptChannelConfiguration_2) and GptChannelId as 0, 2 and 1 respectively).	((Gpt_ChannelType)0U) #define GptConf_GptChannelConfiguration_GptChannelConfiguration_1 ((Gpt_ChannelType)2U) #define GptConf_GptChannelConfiguration_GptChannelConfiguration_2 ((Gpt_ChannelType)1U)

1.1.33 Macro: GPT_READ_ACROSS_CORES

Table 33 GPT_READ_ACROSS_CORES

Name	GPT_READ_ACROSS_CORES	
Description	The macro indicates whether APIs Gpt_GetTimeElapsed and Gpt_GetTimeRemaining can be used to read a channel which is configured in another core.	
Verification method	The macro is generated as STD_ON if the configuration parameter GptReadAcrossCores is set to true. The macro is generated as STD_OFF if the configuration parameter GptReadAcrossCores is set to false.	
Example(s)	**Action**	**Generated output**
	GptReadAcrossCores = True	`#define GPT_READ_ACROSS_CORES (STD_ON)`
	GptReadAcrossCores = False	`#define GPT_READ_ACROSS_CORES (STD_OFF)`

1.1.34 Macro: GPT_RUNTIME_ERROR_DETECT

Table 34 GPT_RUNTIME_ERROR_DETECT

Name	GPT_RUNTIME_ERROR_DETECT	
Description	Enables/disables the Runtime Error Detection.	
Verification method	The macro is generated as STD_ON if GptRunTimeErrorDetect configuration parameter is set to 'True' else the macro is generated as STD_OFF. GPT_RUNTIME_ERROR_DETECT will always be generated as STD_OFF for Autosar version 4.2.2.	
Example(s)	**Action**	**Generated output**
	GptRunTimeErrorDetect= True	`#define GPT_RUNTIME_ERROR_DETECT (STD_ON)`
	GptRunTimeErrorDetect= False	`#define GPT_RUNTIME_ERROR_DETECT (STD_OFF)`

1.2 File: Gpt[_<variant>]_PBcfg.c

The generated source file contains all post-build configuration parameters. Post-build time configuration mechanism allows configurable functionality of Gpt driver that is deployed as object code. The file is generated in 'src' folder.

1.2.1 Structure: Gpt_Config[_<variant>]

Table 35 Gpt_Config[_<variant>]

Name	Gpt_Config[_<variant>]	
Type	Gpt_ConfigType	
Description	Root configuration structure of Gpt driver which will be used during initialization.	
Verification method	The generated structure is present in Gpt[_<variant>]_PBcfg.c file. <Variant> indicates the name of the post-build variant. For a variant-aware configuration the structure name is appended with the variant name. For variant-unaware configuration <variant> is ignored.	
Example(s)	**Action**	**Generated output**

	Only core1 configured (variant unaware)	`const Gpt_ConfigType Gpt_Config =`
		`{`
		`  /* Pointer to Gpt Core Specific Config Set */`
		`  {`
		`    NULL_PTR,            /* CORE 0 */`
		`    &Gpt_kConfig_Core1, /* CORE 1 */`
		`    NULL_PTR,            /* CORE 2 */`
		`    NULL_PTR,            /* CORE 3 */`
		`    NULL_PTR,            /* CORE 4 */`
		`    NULL_PTR,            /* CORE 5 */`
		`  }`
		`};`

	Only core1 configured (variant aware. Variant name is 'Petrol')	`const Gpt_ConfigType Gpt_Config_Petrol =`
		`{`
		`  /* Pointer to Gpt Core Specific Config Set */`
		`  {`
		`    NULL_PTR,            /* CORE 0 */`
		`    &Gpt_kConfig_Core1, /* CORE 1 */`
		`    NULL_PTR,            /* CORE 2 */`
		`    NULL_PTR,            /* CORE 3 */`
		`    NULL_PTR,            /* CORE 4 */`
		`    NULL_PTR,            /* CORE 5 */`

	`  }`
	`};`

Only core1 configured (variant aware. Variant name is 'Diesel')	`const Gpt_ConfigType Gpt_Config_Diesel =`
	`{`
	`  /* Pointer to Gpt Core Specific Config Set */`
	`  {`
	`    &Gpt_kConfig_Core0, /* CORE 0 */`
	`    &Gpt_kConfig_Core1, /* CORE 1 */`
	`    NULL_PTR,           /* CORE 2 */`
	`    NULL_PTR,           /* CORE 3 */`
	`    NULL_PTR,           /* CORE 4 */`
	`    NULL_PTR,           /* CORE 5 */`
	`  }`
	`};`

Only core1 configured (variant aware. Variant name is 'Gasoline')	`const Gpt_ConfigType Gpt_Config_Gasoline =`
	`{`
	`  /* Pointer to Gpt Core Specific Config Set */`
	`  {`
	`    &Gpt_kConfig_Core0, /* CORE 0 */`
	`    &Gpt_kConfig_Core1, /* CORE 1 */`
	`    NULL_PTR,           /* CORE 2 */`
	`    NULL_PTR,           /* CORE 3 */`
	`    NULL_PTR,           /* CORE 4 */`
	`    NULL_PTR,           /* CORE 5 */`
	`  }`
	`};`

1.2.1.1 Member: Gpt_Config_CorePtr[MCAL_NO_OF_CORES]

Table 36 Gpt_Config_CorePtr[MCAL_NO_OF_CORES]

Name	Gpt_Config_CorePtr[MCAL_NO_OF_CORES]
Type	Gpt_CoreConfigType*
Description	Array of core-specific configuration. The array size is based on the number of available cores in hardware.

User configurable	No	
Verification method	The generated structure member is present in Gpt_Config [_<variant>] structure. If a Core<x> is allocated at least one Gpt channel, then the element <x> shall be generated as pointer to Gpt_CoreConfigType ('&Gpt_kConfig_Core<x>') else 'NULL_PTR' is generated.(x in range 0 to number of available cores in hardware).	
Example(s)	**Action**	**Generated output**

	All the Gpt channels are allocated to Core 0	`/* Pointer to Gpt Core Specific Config Set */`
		`{`
		`&Gpt_kConfig_Core0, /* CORE 0 */`
		`NULL_PTR, /* CORE 1 */`
		`NULL_PTR, /* CORE 2 */`
		`NULL_PTR, /* CORE 3 */`
		`NULL_PTR, /* CORE 4 */`
		`NULL_PTR, /* CORE 5 */`
		`}`

	All the Gpt channels are distributed across all cores except Core 0	`/* Pointer to Gpt Core Specific Config Set */`
		`{`
		`NULL_PTR, /* CORE 0 */`
		`&Gpt_kConfig_Core1, /* CORE 1 */`
		`&Gpt_kConfig_Core2, /* CORE 2 */`
		`&Gpt_kConfig_Core3, /* CORE 3 */`
		`&Gpt_kConfig_Core4, /* CORE 4 */`
		`&Gpt_kConfig_Core5, /* CORE 5 */`
		`}`

1.2.2 Structure: Gpt_kConfig_Core<x>

Table 37 Gpt_kConfig_Core<x>

Name	Gpt_kConfig_Core<x>	
Type	Gpt_CoreConfigType	
Description	Configuration structure of Gpt driver for Core <x> which will be referenced in root configuration structure. (x ranges from 0 to MCAL_NO_OF_CORES)	
Verification method	The generated file has this structure if at least one channel is assigned to Core <x>.	
Example(s)	**Action**	**Generated output**

	Configure a Gpt channel to core1	`static const Gpt_CoreConfigType Gpt_kConfig_Core1 =`
		`{`
		`/* Pointer to Channels allocated to Core 1 */`

		` Gpt_ChannelIndex_Core1,`
		
		` /* Pointer to channel configuration of Core1 */`
		` Gpt_kChannelConfig_Core1,`
		
		` /* Channel configerd for 1us Predef Timer in Core 1 */`
		` #if ((GPT_PREDEF_TIMER_1US_16BIT_EN == STD_ON)     || \`
		`  (GPT_PREDEF_TIMER_1US_24BIT_EN == STD_ON)     || \`
		`  (GPT_PREDEF_TIMER_1US_32BIT_EN == STD_ON))`
		`&Gpt_k1UsPredefTimerChannelConfig_Core1,`
		` #endif`
		
		` /* Channel configerd for 100us Predef Timer in Core 1 */`
		` #if (GPT_PREDEF_TIMER_100US_32BIT_EN == STD_ON)`
		`&Gpt_k100UsPredefTimerChannelConfig_Core1,`
		` #endif`
		
		` /* Maximum Normal Channels allocated to core 1 */`
		` GPT_MAX_CHANNELS_CORE1`
		`};`

1.2.2.1 Member: Gpt_ChannelIndexPtr<x>

Table 38 Gpt_ChannelIndexPtr<x>

Name	Gpt_ChannelIndexPtr<x>
Type	uint8 *
Description	Pointer to the array which holds the channel number among all the channels configured to the core, in the order it is configured.
Verification method	The pointer is generated when channels are configured to a core under ResourceMAllocation and will point to the channel index array of the corresponding

	Core<x>. *Note: This configuration parameter will not generated if all the channels are mapped to master core*	
Example(s)	**Action**	**Generated output**
	Configure at least 1 Gpt channel to Core 0	`Gpt_ChannelIndex_Core0`
	Configure at least 1 Gpt channel to Core 3	`Gpt_ChannelIndex_Core3`

1.2.2.2 Member: ChannelConfigPtr

Table 39 ChannelConfigPtr

Name	ChannelConfigPtr	
Type	Gpt_ChannelConfigType *	
Description	Pointer to the base of array which stores the data of each channel configured to Core<x>.	
Verification method	The structure member is generated with base address of array which stores the channel data of Core <x>.	
Example(s)	**Action**	**Generated output**
	Configure at least 1 Gpt channel to Core 0	`Gpt_kChannelConfig_Core0`

1.2.2.3 Member: Predef1UsChannelConfigPtr

Table 40 Predef1UsChannelConfigPtr

Name	Predef1UsChannelConfigPtr	
Type	Gpt_1UsPredefTimerChannelConfigType *	
Description	Pointer to the structure holding information about the 1us predefined timer.	
Verification method	The structure member is generated with address of structure holding information about the 1us predefined timer. *1. This will not be generated when 'GptPredefTimer1usEnablingGrade' = GPT_PREDEF_TIMER_1US_DISABLED* *2. If configured, the structure member is generated for all the cores*	
Example(s)	**Action**	**Generated output**
	Configure 100us predefined timer in master core.	`Gpt_k1UsPredefTimerChannelConfig_Core0`
	Configure 100us predefined timer in master core.	`Gpt_k1UsPredefTimerChannelConfig_Core3`

1.2.2.4 Member: Predef100UsChannelConfigPtr

Table 41 Predef100UsChannelConfigPtr

Name	Predef100UsChannelConfigPtr	
Type	Gpt_100UsPredefTimerChannelConfigType *	
Description	Pointer to the structure holding information about the 100us predefined timer.	
Verification method	The structure member is generated with address of structure holding information about the 100us predefined timer. *1. This will not be generated when 'GptPredefTimer100us32bitEnable' is disabled* *2. If configured, the structure member is generated for all the cores*	
Example(s)	**Action**	**Generated output**
	Configure 100us predefined timer in master core.	`Gpt_k100UsPredefTimerChannelConfig_Core0`
	Configure 100us predefined timer in master core.	`Gpt_k100UsPredefTimerChannelConfig_Core0`

1.2.2.5 Member: Gpt_MaxNormalChannels

Table 42 Gpt_MaxNormalChannels

Name	Gpt_MaxNormalChannels	
Type	uint8	
Description	Indicates the total number of Gpt normal channels assigned to a core.	
Verification method	The structure member is generated as total number of channels allocated to a core.	
Example(s)	**Action**	**Generated output**
	• Configure 4 Gpt channels. 3 are allocated to Core 0. • 1 channel is allocated to Core 1. Output is shown for Core 0	`3`
	• Configure 14 Gpt channels. 3 are allocated to Core 1. • ResourceMMasterCore is CORE0. • Rest of the channels are not allocated to any core. Output is shown for Core 0	`11`

1.2.3 Structure: Gpt_kChannelConfig_Core<x>

Table 43 Gpt_kChannelConfig_Core<x>

Name	Gpt_kChannelConfig_Core<x>	
Type	Gpt_ChannelConfigType	
Description	Configuration structure of Gpt normal channel which will be referenced in Gpt_kConfig_Core<x> (x ranges from 0 to available number of cores in the hardware5)	
Verification method	The structure member is generated as list of all Gpt normal channels allocated to CORE<x>. *Note: Channels not assigned to any core are assigned to master core (ResourceMMasterCore).*	
Example(s)	**Action**	**Generated output**

	• Configure only one Gpt normal channel in core3 with following channel configuration settings
	3. One-shot Mode
	4. Notification OFF
	5. Wakeup OFF
	
	Output is shown for Core 3
		`static const Gpt_ChannelConfigType Gpt_kChannelConfig_Core3[ ] =`
		`{`
		`  /*`
		`    Channel Symbolic Name(ChannelId) : GptChannelConfiguration_0`
		`    GTM TOM/ATOM/GPT12 Channel : GTM_TOM1_CHANNEL9 in GPT_MODE_ONESHOT`
		`  */`
		`  {`
		`    #if (GPT_ENABLE_DISABLE_NOTIFICATION_API == STD_ON)`
		`    NULL_PTR, /* Notification Function */`
		`    #endif`
		
		`    #if ( (GPT_WAKEUP_FUNCTIONALITY_API == STD_ON) \`
		`    && (GPT_REPORT_WAKEUP_SOURCE == STD_ON) )`
		`    0U, /* Wakeup Info */`
		`    #endif`
		
		`    #if (GPT_WAKEUP_FUNCTIONALITY_API == STD_ON)`
		`    (boolean)FALSE, /* Wakeup`

	```
	Capability */
	    #endif

	    GPT_MODE_ONESHOT, /* Channel
	Mode */

	    #if ((GPT_ATOM_USED == STD_ON)
	|| (GPT_TOM_USED == STD_ON))
	    &GptGtmTimerInfo_Core3_Ch0,
	    #endif
	    #if (GPT_GPT12_USED == STD_ON)
	    NULL_PTR
	    #endif
	  }
	};
	```

• Configure two Gpt normal channels, one ATOM and one GPT12 timer in core1 with following channel configuration settings
1. Continuous mode
2. Notification ON
3. Wakeup ON
 	
Output is shown for Core 1
	```
	static const Gpt_ChannelConfigType
	Gpt_kChannelConfig_Core1[ ] =
	{
	  /*
	    Channel Symbolic Name(ChannelId)
	: GptChannelConfiguration_0
	    GTM TOM/ATOM/GPT12 Channel :
	MCU_GPT12_TIMER4 in
	GPT_MODE_CONTINUOUS
	  */
	  {
	    #if
	(GPT_ENABLE_DISABLE_NOTIFICATION_API
	== STD_ON)
	    &IoHwAb_GptNotification0, /*
	Notification Function */
	    #endif

	    #if (
	(GPT_WAKEUP_FUNCTIONALITY_API ==
	STD_ON) \
	    && (GPT_REPORT_WAKEUP_SOURCE ==
	STD_ON) )
	    1U, /* Wakeup Info */
	    #endif
	```

```
    #if
(GPT_WAKEUP_FUNCTIONALITY_API ==
STD_ON)
    (boolean)TRUE, /* Wakeup
Capability */
    #endif


    GPT_MODE_CONTINUOUS, /* Channel
Mode */


    #if ((GPT_ATOM_USED == STD_ON)
|| (GPT_TOM_USED == STD_ON))
    NULL_PTR,
    #endif
    #if (GPT_GPT12_USED == STD_ON)
    GptGpt12TimerInfo_Core1_Ch0
    #endif
  },


  /*
  /*
    Channel Symbolic Name(ChannelId)
: GptChannelConfiguration_1
    GTM TOM/ATOM/GPT12 Channel :
GTM_ATOM0_CHANNEL0 in
GPT_MODE_CONTINUOUS
  */
  {
    #if
(GPT_ENABLE_DISABLE_NOTIFICATION_API
== STD_ON)
    &IoHwAb_GptNotification1, /*
Notification Function */
    #endif


    #if (
(GPT_WAKEUP_FUNCTIONALITY_API ==
STD_ON) \
    && (GPT_REPORT_WAKEUP_SOURCE ==
STD_ON) )
    2U, /* Wakeup Info */
    #endif
```

		`#if (GPT_WAKEUP_FUNCTIONALITY_API == STD_ON)`
		`(boolean)TRUE, /* Wakeup Capability */`
		`#endif`
		
		`GPT_MODE_CONTINUOUS, /* Channel Mode */`
		
		`#if ((GPT_ATOM_USED == STD_ON) || (GPT_TOM_USED == STD_ON))`
		`&GptGtmTimerInfo_Core1_Ch1,`
		`#endif`
		`#if (GPT_GPT12_USED == STD_ON)`
		`NULL_PTR`
		`#endif`
		`}`
		`};`

1.2.3.1 Member: GptNotificationPtr

Table 44 GptNotificationPtr

Name	GptNotificationPtr	
Type	Gpt_NotificationPtrType	
Description	Pointer referring to user defined notification function	
Verification method	This configuration parameter will be generated only when 'GptEnableDisableNotificationApi' is enabled in general container of Gpt driver. This pointer refers to a notification function if configured in 'GptNotification' else it will point to NULL	
Example(s)	**Action**	**Generated output**

	Configure GptGeneral/ GptEnableDisableNotificationApi = true
	Configure Gpt/GptChannelConfigSet/GptChannelConfiguration/GptChannelConfiguration_<x>/GptNotification = IoHwAb_GptNotification
	
	Where <x> is the channel number
		`#if (GPT_ENABLE_DISABLE_NOTIFICATION_API == STD_ON)`
		`&IoHwAb_GptNotification, /* Notification Function */`
		`#endif`

	Configure GptGeneral/ GptEnableDisableNotificationApi = false
	Configure Gpt/GptChannelConfigSet/GptChannelConfiguration/GptChannelConfiguration_<x>/GptNotification =""
		`#if (GPT_ENABLE_DISABLE_NOTIFICATION_API == STD_ON)`
		`NULL_PTR, /* Notification Function */`
		`#endif`

1.2.3.2 Member: GptChannelWakeupInfo

Table 45 GptChannelWakeupInfo

Name	GptChannelWakeupInfo	
Type	EcuM_WakeupSourceType	
Description	Wakeup information to EcuM_SetWakeupEvent	
Verification method	This configuration parameter holds the Wakeup source of the Ecu to which the channel needs to wakeup	
Example(s)	**Action**	**Generated output**

	Configure GptGeneral/ GptReportWakeupSource = true
	Configure Gpt/GptChannelConfigSet/GptChannelConfiguration/GptChannelConfiguration_<x>/GptWakeu
		`#if ( (GPT_WAKEUP_FUNCTIONALITY_API == STD_ON) \`
		`&& (GPT_REPORT_WAKEUP_SOURCE == STD_ON) )`
		`31U, /* Wakeup Info */`

	pConfiguration_<y>/GptWakeupSourceRef = EcuMWakeupSource_31 Where <x> is the channel number and y is the wakeup configuration instance	`#endif`
	Configure Gpt/GptChannelConfigSet/GptChannelConfiguration/GptChannelConfiguration_<x>/GptWakeupConfiguration_<y>/GptWakeupSourceRef = EcuMWakeupSource_7 and generate	`#if (` `(GPT_WAKEUP_FUNCTIONALITY_API == STD_ON) \` `&& (GPT_REPORT_WAKEUP_SOURCE == STD_ON) )` `7U, /* Wakeup Info */` `#endif`
	Configure Gpt/GptChannelConfigSet/GptChannelConfiguration/GptChannelConfiguration_<x>/GptWakeupConfiguration_<y>/GptWakeupSourceRef = EcuMWakeupSource_0 and generate	`#if ( (GPT_WAKEUP_FUNCTIONALITY_API == STD_ON) \` `&& (GPT_REPORT_WAKEUP_SOURCE == STD_ON) )` `0U, /* Wakeup Info */` `#endif`

1.2.3.3 Member: GptEnableWakeupState

Table 46 **GptEnableWakeupState**

Name	GptEnableWakeupState	
Type	Gpt_EnableWakeupType	
Description	Enable/Disable channel wakeup capability	
Verification method	This configuration parameter is TRUE if GptEnableWakeup is enabled in GptChannelConfiguration else it is FALSE	
Example(s)	**Action**	**Generated output**
	Configure GptEnableWakeup = true in Gpt/GptChannelConfigSet/GptChannelConfiguration/GptChannelConfiguration_<x>/General Where <x> is the channel id	`#if (GPT_WAKEUP_FUNCTIONALITY_API == STD_ON)` `(boolean)TRUE, /* Wakeup Capability */` `#endif`
	Configure GptEnableWakeup = false in Gpt/GptChannelConfigSet/GptChannelConfiguration/GptChannelConfiguration_<x>/Genera Where <x> is the channel id l	`#if` `(GPT_WAKEUP_FUNCTIONALITY_API == STD_ON)` `(boolean)FALSE, /* Wakeup Capability */` `#endif`

1.2.3.4 Member: GptChannelMode

Table 47 GptChannelMode

Name	GptChannelMode	
Type	Gpt_ChannelModeType	
Description	Sets the channel to continuous/one-shot mode	
Verification method	This configuration parameter is chosen from the drop down list Gpt/GptChannelConfigSet/GptChannelConfiguration/GptChannelConfiguration_<x>/ General/ GptChannelMode	
Example(s)	**Action**	**Generated output**
	Select 'GPT_CH_MODE_CONTINUOUS ' for the drop down list Gpt/GptChannelConfigSet/GptChannelConfiguration/GptChannelConfiguration_<x>/General/ GptChannelMode	`GPT_MODE_CONTINUOUS, /* Channel Mode */`
	Select 'GPT_CH_MODE_ONESHOT ' for the drop down list Gpt/GptChannelConfigSet/GptChannelConfiguration/GptChannelConfiguration_<x>/General/ GptChannelMode	`GPT_MODE_ONESHOT, /* Channel Mode */`

1.2.3.5 Member: GptGtmTimerInfo

Table 48 GptGtmTimerInfo

Name	GptGtmTimerInfo	
Type	Mcu_17_Gtm_TomAtomChConfigType	
Description	Pointer to the structure containing information about Gtm timer instance used for the channel	
Verification method	This structure pointer is generated based on whether GPT12 or GTM(TOM/ATOM) is used for the channel.	
Example(s)	**Action**	**Generated output**

	Select TOM0Ch0 in 'GtmTimerOutputModuleConfiguration' and GPT_CH_MODE_ONESHOT for 'GptChannelMode'	`#if ((GPT_ATOM_USED == STD_ON) || (GPT_TOM_USED == STD_ON))`
		`&GptGtmTimerInfo_Core1_Ch0,`
		`#endif`
		`#if (GPT_GPT12_USED == STD_ON)`
		`NULL_PTR`
		`#endif`

	Do not configure 'GtmTimerOutputModuleConfig	`#if ((GPT_ATOM_USED == STD_ON) || (GPT_TOM_USED == STD_ON))`

	uration'. Select Timer2 in 'Gpt12TimerOutputModuleConfiguration' and GPT_CH_MODE_ONESHOT for 'GptChannelMode'	`NULL_PTR,`
		`#endif`
		`#if (GPT_GPT12_USED == STD_ON)`
		`GptGpt12TimerInfo_Core1_Ch1`
		`#endif`

1.2.3.6 Member: GptGpt12TimerInfo

Table 49 GptGpt12TimerInfo

Name	GptGpt12TimerInfo	
Type	Mcu_17_Gpt12_TimerConfigType	
Description	Pointer to the structure containing information about Gpt12 timer instance used for the channel	
Verification method	This structure pointer is generated based on whether GPT12 or GTM(TOM/ATOM) is used for the channel.	
Example(s)	**Action**	**Generated output**

	Select Timer2 in 'Gpt12TimerOutputModuleConfiguration' and GPT_CH_MODE_ONESHOT for 'GptChannelMode'	`#if ((GPT_ATOM_USED == STD_ON) || (GPT_TOM_USED == STD_ON))`
		`NULL_PTR,`
		`#endif`
		`#if (GPT_GPT12_USED == STD_ON)`
		`GptGpt12TimerInfo_Core1_Ch0`
		`#endif`

	Do not configure 'Gpt12TimerOutputModuleConfiguration'. Select TOM0Ch0 in 'GtmTimerOutputModuleConfiguration' and GPT_CH_MODE_ONESHOT for 'GptChannelMode'	`#if ((GPT_ATOM_USED == STD_ON) || (GPT_TOM_USED == STD_ON))`
		`&GptGtmTimerInfo_Core1_Ch1,`
		`#endif`
		`#if (GPT_GPT12_USED == STD_ON)`
		`NULL_PTR`
		`#endif`

1.2.4 Structure: GptGtmTimerInfo_Core<x>_Ch<y>

Table 50 GptGtmTimerInfo_Core<x>_Ch<y>

Name	GptGtmTimerInfo_Core<x>_Ch<y>	
Type	Mcu_17_Gtm_TomAtomChConfigType	
Description	Structure containing information about Gtm timer instance used for the channel	
Verification method	The structure is generated as per the configuration in the channel and the contents of each member of the structure is explained in the further sections	
Example(s)	**Action**	**Generated output**
	Select TOM0Ch0 in 'GtmTimerOutputModuleConfig	`{`

uration' and GPT_CH_MODE_ONESHOT for 'GptChannelMode'	`MCU_GTM_TIMER_TOM, /* Timer Type (TOM/ATOM)*/`
	`0x0, /* Timer Number Module No | Timer Channel No */`
	`0x4000800U, /* Channel Control Register */`
	`0x0U, /* CN0 in ticks */`
	`0x0U, /* CM0 in ticks */`
	`0x0U, /* CM1 in ticks */`
	`0x0U, /* SR0 in ticks */`
	`0x0U, /* SR1 in ticks */`
	`0x0U, /* Port Out */`
	`0x80U /* Interrupt status and mode*/`
	`}`

Select TOM0Ch0 in 'GtmTimerOutputModuleConfiguration' and GPT_CH_MODE_CONTINUOUS for 'GptChannelMode'	`{`
	`MCU_GTM_TIMER_TOM, /* Timer Type (TOM/ATOM)*/`
	`0x0, /* Timer Number Module No | Timer Channel No */`
	`0x800U, /* Channel Control Register */`
	`0x0U, /* CN0 in ticks */`
	`0x0U, /* CM0 in ticks */`
	`0x0U, /* CM1 in ticks */`
	`0x0U, /* SR0 in ticks */`
	`0x0U, /* SR1 in ticks */`
	`0x0U, /* Port Out */`
	`0x80U /* Interrupt status and mode*/`
	`}`

Select TOM1Ch5 in 'GtmTimerOutputModuleConfiguration' and GPT_CH_MODE_CONTINUOUS for 'GptChannelMode'	`{`
	`MCU_GTM_TIMER_TOM, /* Timer Type (TOM/ATOM)*/`
	`0x105, /* Timer Number Module No | Timer Channel No */`
	`0x800U, /* Channel Control Register */`
	`0x0U, /* CN0 in ticks */`
	`0x0U, /* CM0 in ticks */`
	`0x0U, /* CM1 in ticks */`
	`0x0U, /* SR0 in ticks */`

		`0x0U, /* SR1 in ticks */`
		`0x0U, /* Port Out */`
		`0x80U /* Interrupt status and mode*/`
		`}`

1.2.4.1 Member: TimerType

Table 51 TimerType

Name	TimerType	
Type	Mcu_17_Gtm_TimerOutType	
Description	Indicates whether TOM/ATOM is used for the channel	
Verification method	This configuration parameter will be TOM if 'McuGtmTomAllocationConf_<x>' is selected in GtmTimerOutputModuleConfiguration, where x is the TOM module number And will be ATOM if 'McuGtmAtomAllocationConf_<x>' is selected in GtmTimerOutputModuleConfiguration, where x is the ATOM module number	
Example(s)	**Action**	**Generated output**
	Select a TOM channel for GtmTimerUsed in GtmTimerOutputModuleConfiguration	`MCU_GTM_TIMER_TOM, /* Timer Type (TOM/ATOM)*/`
	Select an ATOM channel for GtmTimerUsed in GtmTimerOutputModuleConfiguration	`MCU_GTM_TIMER_ATOM, /* Timer Type (TOM/ATOM)*/`

1.2.4.2 Member: TimerId

Table 52 TimerId

Name	TimerId	
Type	Mcu_17_Gtm_TimerChIdentifierType	
Description	Indicates the module and channel number of the TOM/ATOM used	
Verification method	This configuration parameter is a numerical value whose first 8bits represent channel number and next 8bits represent module number	
Example(s)	**Action**	**Generated output**
	Select TOM0CH0 for GtmTimerUsed in GtmTimerOutputModuleConfiguration	`0x0, /* Timer Number Module No | Timer Channel No */`
	Select TOM1CH4 for GtmTimerUsed in GtmTimerOutputModuleConfiguration	`0x104, /* Timer Number Module No | Timer Channel No */`

1.2.4.3 Member: TimerChCtrlReg

Table 53 TimerChCtrlReg

Name	TimerChCtrlReg	
Type	uint32	
Description	Holds the contents of control register based on the channel configuration	
Verification method	This configuration parameter will be a 32bit numerical value which will be generated as per configuration strings as under. Bits 0,1 will be 2 for ATOM(SOMP) else 0 for TOM Bit11 – 1 (SL (signal level is high TOM_OUT is low)) Bits 12-14 – (CLK_SRC_SR) As per selection of GtmTimerClockSelect in GtmTimerOutputModuleConfiguration_<x> (where x corresponds to Gpt channel number) Bit25 – OSM as per selection of GptChannelMode All other bits are set to zero.	
Example(s)	**Action**	**Generated output**
	• Configure channel 0 to use with TOM0Ch0 with the help of configuration parameter GtmTimerUsed • Select GTM_FIXED_CLOCK_0 for GtmTimerClockSelect • Select GPT_CH_MODE_CONTINUOUS for GptChannelMode and generate	`0x800U, /* Channel Control Register */`
	• Configure channel 0 to use with TOM0Ch0 with the help of configuration parameter GtmTimerUsed. Select GTM_FIXED_CLOCK_0 for GtmTimerClockSelect • Select GPT_CH_MODE_ONESHOT for GptChannelMode and generate	`0x4000800U, /* Channel Control Register */`
	• Configure channel 0 to use with TOM0Ch0 with the help of configuration parameter GtmTimerUsed • Select GTM_FIXED_CLOCK_2 for GtmTimerClockSelect • Select GPT_CH_MODE_ONESHOT for GptChannelMode and	`0x4002800U, /* Channel Control Register */`

	generate	
	• Configure channel 0 to use with TOM0Ch0 with the help of configuration parameter GtmTimerUsed • Select GTM_FIXED_CLOCK_4 for GtmTimerClockSelect • Select GPT_CH_MODE_CONTINUOUS for GptChannelMode and generate	`0x4800U, /* Channel Control Register */`
	• Configure channel 0 to use with ATOM0Ch0 with the help of configuration parameter GtmTimerUsed • Select GTM_CONFIGURABLE_CLOCK_0 for GtmTimerClockSelect • Select GPT_CH_MODE_CONTINUOUS for GptChannelMode and generate	`0x802U, /* Channel Control Register */`
	• Configure channel 0 to use with ATOM0Ch0 with the help of configuration parameter GtmTimerUsed • Select GTM_CONFIGURABLE_CLOCK_7 for GtmTimerClockSelect • Select GPT_CH_MODE_CONTINUOUS for GptChannelMode and generate	`0x7802U, /* Channel Control Register */`
	• Configure channel 0 to use with ATOM0Ch0 with the help of configuration parameter GtmTimerUsed • Select GTM_CONFIGURABLE_CLOCK_3 for GtmTimerClockSelect • Select GPT_CH_MODE_ONESHOT for GptChannelMode and generate	`0x4003802U, /* Channel Control Register */`

1.2.4.4 Member: TimerChCN0Reg

Table 54 TimerChCN0Reg

Name	TimerChCN0Reg	
Type	uint32	
Description	Holds the initialization value of the CN0 register	
Verification method	This will be always be generated as zero	
Example(s)	**Action**	**Generated output**
	Generate Gpt_PBcfg.c	`0x0U, /* CN0 in ticks */`

1.2.4.5 Member: TimerChCM0Reg

Table 55 TimerChCM0Reg

Name	TimerChCM0Reg	
Type	uint32	
Description	Holds the initialization value of the CM0 register	
Verification method	In case of predefined timers where extra channel is used for frequency tuning, this will be set to the prescaling factor to achieve the desired frequency, otherwise this will be set to max (0xFFFF) In case of normal channels it will be always generated as zero	
Example(s)	**Action**	**Generated output**

	Configure a predefined timer to use 3TOM channels and generate to check the first timer is initialized with desired prescaling factor and the other to with 0xffff	`For tuning timer`
		`0x64U, /* CM0 in ticks */`
		`For Predefined timer L and H words`
		`0xffffU, /* CM0 in ticks */`
		`0xffffU, /* CM0 in ticks */`

	Configure a normal channel and generate to check the output is zero	`0x0U, /* CM0 in ticks */`

1.2.4.6 Member: TimerChCM1Reg

Table 56 TimerChCM1Reg

Name	TimerChCM1Reg	
Type	uint32	
Description	Holds the initial value of CM1 register	
Verification method	This configuration register is always generated as zero	
Example(s)	**Action**	**Generated output**
	Generate Gpt_PBcfg.c	`0x0U, /* CM1 in ticks */`

1.2.4.7 Member: TimerChSR0Reg

Table 57 TimerChSR0Reg

Name	TimerChSR0Reg	
Type	uint32	
Description	This is a shadow register to CM0	
Verification method	This will always holds the same value as CM0	
Example(s)	**Action**	**Generated output**
	Generate Gpt_PBcfg.c	`NA`

1.2.4.8 Member: TimerChSR1Reg

Table 58 TimerChSR1Reg

Name	TimerChSR1Reg	
Type	uint32	
Description	This is a shadow register to CM1	
Verification method	This will always holds the same value as CM1	
Example(s)	**Action**	**Generated output**
	Generate Gpt_PBcfg.c	NA

1.2.4.9 Member: TimerChPortOutConfig

Table 59 TimerChPortOutConfig

Name	TimerChPortOutConfig	
Type	uint32	
Description	Indicates the ports used for outputting the timer events	
Verification method	This configuration register is always generated as zero	
Example(s)	**Action**	**Generated output**
	Generate Gpt_PBcfg.c	`0x0U, /* Port Out      */`

1.2.4.10 Member: TimerChIntEnMode

Table 60 TimerChIntEnMode

Name	TimerChIntEnMode	
Type	uint8	
Description	Indicates the interrupt mode used	
Verification method	This value should be 0x80 to represent the pulse notify mode for normal channels and 0x00 representing level mode for predefined timers.	
Example(s)	**Action**	**Generated output**
	Configure a normal channel and a predefined timer channel The output shown is for normal channel	`0x80U  /* Interrupt status and mode*/`

	Configure a normal channel and a predefined timer channel The output shown is for predefined timer channel	`0x0U /* Interrupt status and mode*/`

1.2.5 Structure: GptGpt12TimerInfo_Core<x>_Ch<y>

Table 61 GptGpt12TimerInfo_Core<x>_Ch<y>

Name	GptGpt12TimerInfo_Core<x>_Ch<y>	
Type	Mcu_17_Gpt12_TimerConfigType	
Description	Structure containing information about Gpt12 timer instance used for the channel	
Verification method	The structure is generated as per the configuration in the channel and the contents of each member of the structure is explained in the further sections. For continuous mode using GPT1 timer block an array of two structures are generated. For all other cases, only one structure element is generated.	
Example(s)	**Action**	**Generated output**

	Select T2 in 'Gpt12TimerOutputModuleConfiguration' and GPT_CH_MODE_ONESHOT for 'GptChannelMode'	`/*`
		`  Channel Symbolic Name(ChannelId) : GptChannelConfiguration_0`
		`  GTM TOM/ATOM/GPT12 Channel : MCU_GPT12_TIMER2 in GPT_MODE_ONESHOT`
		`*/`
		`static const Mcu_17_Gpt12_TimerConfigType GptGpt12TimerInfo_Core2_Ch0[ ]=`
		`{`
		`  {`
		`    MCU_GPT12_TIMER2, /* Timer Type (GPT1/GPT2)*/`
		`    0x80U, /* Channel Control Register */`
		`    0x0U,`
		`    0x0U`
		`  }`
		`};`

	Select T6 in 'Gpt12TimerOutputModuleConfiguration' and GPT_CH_MODE_CONTINUOUS for 'GptChannelMode'
	
	Note: If T6 is used to realize continuous mode, auxillary
		`/*`
		`  Channel Symbolic Name(ChannelId) : GptChannelConfiguration_3`
		`  GTM TOM/ATOM/GPT12 Channel : MCU_GPT12_TIMER6 in GPT_MODE_CONTINUOUS`
		`*/`
		`static const Mcu_17_Gpt12_TimerConfigType GptGpt12TimerInfo_Core1_Ch1[ ]=`

	timer (T5) is not used.	`{`
		`  {`
		`    MCU_GPT12_TIMER6, /* Timer Type (GPT1/GPT2)*/`
		`    0x8280U, /* Channel Control Register */`
		`    0x0U,`
		`    0x0U`
		`  }`
		`};`

	Select T3 and T4 in ‘Gpt12TimerOutputModuleConfiguration’ and GPT_CH_MODE_CONTINUOUS for ‘GptChannelMode’
	
	Note: *Two structures are generated in this case. One for core timer T3 and one for auxillary timer T4. For continuous mode using timer block GPT1, two timers are needed.*
		`/*`
		`  Channel Symbolic Name(ChannelId) : GptChannelConfiguration_0`
		`  GTM TOM/ATOM/GPT12 Channel : MCU_GPT12_TIMER4 in GPT_MODE_CONTINUOUS`
		`*/`
		`static const Mcu_17_Gpt12_TimerConfigType GptGpt12TimerInfo_Core1_Ch0[ ]=`
		`{`
		
		`  {`
		`    MCU_GPT12_TIMER3,      /* Timer Type (GPT1/GPT2)*/`
		`    0x280U, /* Channel Control Register */`
		`    0x0U,`
		`    0x0U`
		`  },`
		`  {`
		`    MCU_GPT12_TIMER4,      /* Timer Type (GPT1/GPT2)*/`
		`    0x227U, /* Channel Control Register */`
		`    0x0U,`
		`    0x0U`
		`  }`
		`};`

1.2.5.1 Member: TimerId

Table 62 TimerId

Name	TimerId	
Type	Mcu_17_Gpt12_TimerChIdentifierType	
Description	Indicates which GPT12 timer is used for the channel	
Verification method	Based on the configuration in the Gpt12TimerOutputModuleConfiguration, the generated output will be MCU_GPT12_TIMER<x> where x is 2 to 6.	
Example(s)	**Action**	**Generated output**
	Select T2 for Gpt12TimerUsed in Gpt12TimerOutputModuleConfiguration	`MCU_GPT12_TIMER2, /* Timer Type (GPT1/GPT2)*/`
	Select T5 for Gpt12TimerUsed in Gpt12TimerOutputModuleConfiguration	`MCU_GPT12_TIMER5, /* Timer Type (GPT1/GPT2)*/`

1.2.5.2 Member: TimerCtrlReg

Table 63 TimerCtrlReg

Name	TimerCtrlReg	
Type	uint32	
Description	Holds the contents of control register based on the channel configuration	
Verification method	This configuration parameter will be generated as per configuration as under. Bits 2:0 represents the prescalar factor. Values can be 0 to 7 based on the value in Gpt12ChannelClockDivider in Gpt12TimerOutputModuleConfiguration. For auxillary timers T2 and T4 used in GPT_CH_MODE_CONTINUOUS value will always be 7. Bits 5:3 will always be generated 000_B for all timers in GPT_CH_MODE_ONESHOT and core timers in GPT_CH_MODE_CONTINUOUS. For auxillary timers T2 and T4 in GPT_CH_MODE_CONTINUOUS, it will be 100_B. Bit 7 – will always be generated 1_B for all timers in GPT_CH_MODE_ONESHOT and core timers in GPT_CH_MODE_CONTINUOUS. For auxillary timers, T2 and T4 in GPT_CH_MODE_CONTINUOUS, it will be 0_B. Bit 9 – will always be generated 0_B for GPT_CH_MODE_ONESHOT and 1_B for both core and auxillary timers in GPT_CH_MODE_CONTINUOUS. Bit 15 – will be generated 1_B for T6 in GPT_CH_MODE_CONTINUOUS. Otherwise this bit will always be generated as 0_B. All other bits are set to zero.	
Example(s)	**Action**	**Generated output**
	• Configure channel 0 to with T2 in GPT_CH_MODE_ONESHOT mode • Configure 5 for Gpt12ChannelClockDivider	`0x85U, /* Channel Control Register */`

• Configure channel 0 to with T6 in GPT_CH_MODE_CONTINUOUS mode • Configure 5 for Gpt12ChannelClockDivider	`0x8285U, /* Channel Control Register */`

• Configure channel 0 to with T3 and T4 in GPT_CH_MODE_CONTINUOUS mode
• Configure 2 for Gpt12ChannelClockDivider
 	
Note: *2 outputs are available in this case*
	`0x282U, /* Channel Control Register */`
	
	`0x227U, /* Channel Control Register */`

1.2.5.3 Member: TimerCntReg

Table 64 TimerCntReg

Name	TimerCntReg	
Type	uint32	
Description	Holds the initialization value of the timer count register	
Verification method	This will be always be generated as zero	
Example(s)	**Action**	**Generated output**
	Generate Gpt_PBcfg.c	`0x0U,`

1.2.5.4 Member: PortInSelReg

Table 65 PortInSelReg

Name	PortInSelReg	
Type	uint32	
Description	Indicates the ports used for outputting the timer events	
Verification method	This configuration register is always generated as zero	
Example(s)	**Action**	**Generated output**
	Generate Gpt_PBcfg.c	`0x0U`

1.2.6 Structure: Gpt_k1UsPredefTimerChannelConfig_Core<x>

Table 66 Gpt_k1UsPredefTimerChannelConfig_Core<x>

Name	Gpt_k1UsPredefTimerChannelConfig_Core<x>
Type	Gpt_1UsPredefTimerChannelConfigType
Description	Configuration structure of 1 micro second predefined timer, which will be referenced

	in Gpt_kConfig_Core<x>	
Verification method	Configuration structure of Gpt 1 microsecond predefine timer for Core <x> which will be referenced in core configuration structure. (x ranges from 0 to 5) *Note:* *Predefined timers can only be controlled from master core in ResourceMMasterCore. But it can be read from all the available cores.*	
Example(s)	**Action**	**Generated output**

	Configure 1us predefined timer with extra channel required as true	`static const Gpt_1UsPredefTimerChannelConfigType \`
		`Gpt_k1UsPredefTimerChannelConfig_Core0 =`
		`{`
		`  #if (GPT_PREDEF_EXTRA_CH_1US == STD_ON)`
		`  {`
		`    MCU_GTM_TIMER_TOM, /* Timer Type (TOM/ATOM) */`
		`    0x1, /* Timer Number Module No | Timer Channel No */`
		`    0x1000800U, /* Channel Control Register */`
		`    0x0U, /* CN0 in ticks */`
		`    0x64U, /* CM0 in ticks */`
		`    0x0U, /* CM1 in ticks */`
		`    0x64U, /* SR0 in ticks */`
		`    0x0U, /* SR1 in ticks */`
		`    0x0U, /* Port Out     */`
		`    0x0U /* Interrupt status and mode*/`
		`  },`
		`  #endif`
		`  {`
		`    MCU_GTM_TIMER_TOM, /* Timer Type (TOM/ATOM) */`
		`    0x2, /* Timer Number Module No | Timer Channel No */`
		`    0x100d800U, /* Channel Control Register */`
		`    0x0U, /* CN0 in ticks */`
		`    0xffffU, /* CM0 in ticks */`
		`    0x0U, /* CM1 in ticks */`
		`    0xffffU, /* SR0 in ticks */`

		`0x0U, /* SR1 in ticks */` `0x0U, /* Port Out */` `0x0U /* Interrupt status and mode*/` `},` `#if ((GPT_PREDEF_TIMER_1US_24BIT_EN == STD_ON) || \` `(GPT_PREDEF_TIMER_1US_32BIT_EN == STD_ON))` `{` `MCU_GTM_TIMER_TOM, /* Timer Type (TOM/ATOM) */` `0x3, /* Timer Number Module No | Timer Channel No */` `0xd800U, /* Channel Control Register */` `0x0U, /* CN0 in ticks */` `0xffffU, /* CM0 in ticks */` `0x0U, /* CM1 in ticks */` `0xffffU, /* SR0 in ticks */` `0x0U, /* SR1 in ticks */` `0x0U, /* Port Out */` `0x0U /* Interrupt status and mode*/` `},` `#endif` `1, /* Is extra Channel for frequency tuning required */` `/* Types of Predef timers enabled : */` `GPT_PREDEF_TIMER_1US_32BIT,` `GPT_PREDEF_TIMER_1US_24BIT,` `GPT_PREDEF_TIMER_1US_16BIT` `};`

1.2.6.1 Member: GptGtm1UsTimerInfo0

Table 67 GptGtm1UsTimerInfo0

Name	GptGtm1UsTimerInfo0
Type	Mcu_17_Gtm_TomAtomChConfigType
Description	Timer used for frequency tuning (This will be the extra channel)
Verification method	The verification method is similar to that of GptGtmTimerInfo structure and its related members. Refer Section 1.2.3.5.1 to 1.2.3.5.10 for more details on the generation

Example(s)	Action	Generated output
	1.2.3.5	

1.2.6.2 Member: GptGtm1UsTimerInfo1

Table 68 GptGtm1UsTimerInfo1

Name	GptGtm1UsTimerInfo1	
Type	Mcu_17_Gtm_TomAtomChConfigType	
Description	Lower word (First 16bits) of the 32bit timer	
Verification method	The verification method is similar to that of GptGtmTimerInfo structure and its related members. Refer Section 1.2.3.5.1 to 1.2.3.5.10 for more details on the generation	
Example(s)	**Action**	**Generated output**
	1.2.3.5	

1.2.6.3 Member: GptGtm1UsTimerInfo2

Table 69 GptGtm1UsTimerInfo2

Name	GptGtm1UsTimerInfo2	
Type	Mcu_17_Gtm_TomAtomChConfigType	
Description	Upper word (Next 16bits) of the 32bit timer	
Verification method	The verification method is similar to that of GptGtmTimerInfo structure and its related members. Refer Section 1.2.3.5.1 to 1.2.3.5.10 for more details on the generation	
Example(s)	**Action**	**Generated output**
	1.2.3.5	

1.2.6.4 Member: ExtraChRequirement1Us

Table 70 ExtraChRequirement1Us

Name	ExtraChRequirement1Us	
Type	Boolean	
Description	Indicates the usage of additional TOM channel for frequency tuning	
Verification method	The generated value of this configuration parameter is TRUE if the predef channel frequency cannot be derived directly from the GTM clock The generated value of this configuration parameter is FALSE if the predef channel frequency cannot be derived directly from the GTM clock.	
Example(s)	**Action**	**Generated output**
	Input clock to the TOM is 2MHz	1, /* Is extra Channel for frequency tuning required */
	Input clock to the TOM is 4MHz	1, /* Is extra Channel for frequency tuning required */
	Input clock to the TOM is 1MHz	0, /* Is extra Channel for frequency tuning required

		*/

1.2.6.5 Member: Gpt1UsPredefTimerUsed0

Table 71 Gpt100UsPredefTimerUsed0

Name	Gpt1UsPredefTimerUsed0	
Type	Gpt_PredefTimerType	
Description	Types of Predef timers enabled	
Verification method	If GPT_PREDEF_TIMER_1US_16_24_32BIT_ENABLED is chosen for Gpt/General/GptPredefTimer1usEnablingGrade then this configuration parameter will be generated as GPT_PREDEF_TIMER_1US_32BIT If GPT_PREDEF_TIMER_1US_16_24BIT_ENABLED is chosen for Gpt/General/GptPredefTimer1usEnablingGrade then this configuration parameter will be generated as GPT_PREDEF_TIMER_1US_24BIT If GPT_PREDEF_TIMER_1US_16BIT_ENABLED is chosen for Gpt/General/GptPredefTimer1usEnablingGrade then this configuration parameter will be generated as GPT_PREDEF_TIMER_1US_16BIT	
Example(s)	**Action**	**Generated output**
	Configure a 1us 32bit predefined timer and generate. Configure GPT_PREDEF_TIMER_1US_16_24BIT_ENABLED in Gpt/General/GptPredefTimer1usEnablingGrade Configure GptTimerChannelUsage in Gpt/GptChannelConfigSet/GptChannelConfiguration/GptChannelConfiguration_0/GptTimerChannelUsage as GPT_PREDEF_TIMERCH_1US_16_24_32BIT_ENABLED and configure the TOM channels in GtmTimerOutputModuleConfiguration *Note:* *Predefined timers will work only with TOM channels*	`GPT_PREDEF_TIMER_1US_32BIT,`

	Configure a 1us 24bit predefined timer similar to above example and generate	`GPT_PREDEF_TIMER_1US_24BIT,`
	Configure a 1us 16bit predefined timer and generate	`GPT_PREDEF_TIMER_1US_16BIT,`

1.2.6.6 Member: Gpt1UsPredefTimerUsed1

Table 72 Gpt1UsPredefTimerUsed1

Name	Gpt1UsPredefTimerUsed1	
Type	Gpt_PredefTimerType	
Description	Types of Predef timers enabled	
Verification method	If GPT_PREDEF_TIMER_1US_16_24_32BIT_ENABLED is chosen for GptPredefTimer1usEnablingGrade then this configuration parameter will be generated as GPT_PREDEF_TIMER_1US_24BIT If GPT_PREDEF_TIMER_1US_16_24BIT_ENABLED is chosen for GptPredefTimer1usEnablingGrade then this configuration parameter will be generated as GPT_PREDEF_TIMER_1US_16BIT If GPT_PREDEF_TIMER_1US_16BIT_ENABLED is chosen for GptPredefTimer1usEnablingGrade then this configuration parameter will be generated as GPT_PREDEF_TIMER_1US_16BIT	
Example(s)	**Action**	**Generated output**
	Configure a 1us 32bit predefined timer similar to the first example in section 1.2.6.5 and generate	`GPT_PREDEF_TIMER_1US_24BIT,`
	Configure a 1us 24bit predefined timer similar to the first example in section 1.2.6.5 and generate	`GPT_PREDEF_TIMER_1US_16BIT,`
	Configure a 1us 16bit predefined timer similar to the first example in section 1.2.6.5 and generate	`GPT_PREDEF_TIMER_1US_16BIT,`

1.2.6.7 Member: Gpt1UsPredefTimerUsed2

Table 73 Gpt1UsPredefTimerUsed2

Name	Gpt1UsPredefTimerUsed2
Type	Gpt_PredefTimerType
Description	Types of Predef timers enabled
Verification method	If GPT_PREDEF_TIMER_1US_16_24_32BIT_ENABLED is chosen for GptPredefTimer1usEnablingGrade then this configuration parameter will be generated as GPT_PREDEF_TIMER_1US_16BIT If GPT_PREDEF_TIMER_1US_16_24BIT_ENABLED is chosen for

	GptPredefTimer1usEnablingGrade then this configuration parameter will be generated as GPT_PREDEF_TIMER_1US_24BIT If GPT_PREDEF_TIMER_1US_16BIT_ENABLED is chosen for GptPredefTimer1usEnablingGrade then this configuration parameter will be generated as GPT_PREDEF_TIMER_1US_16BIT	
Example(s)	**Action**	**Generated output**
	Configure a 1us 32bit predefined timer similar to the first example in section 1.2.6.5 and generate	`GPT_PREDEF_TIMER_1US_16BIT`
	Configure a 1us 24bit predefined timer similar to the first example in section 1.2.6.5 and generate	`GPT_PREDEF_TIMER_1US_24BIT`
	Configure a 1us 16bit predefined timer similar to the first example in section 1.2.6.5 and generate	`GPT_PREDEF_TIMER_1US_16BIT`

1.2.7 Structure: Gpt_k100UsPredefTimerChannelConfig_Core<x>

Table 74 Gpt_k100UsPredefTimerChannelConfig_Core<x>

Name	Gpt_k100UsPredefTimerChannelConfig_Core<x>	
Type	Gpt_100UsPredefTimerChannelConfigType	
Description	Configuration structure of 100 micro second predefined timer, which will be referenced in Gpt_kConfig_Core<x>	
Verification method	Configuration structure of Gpt 100 microsecond predefine timer for Core <x> which will be referenced in core configuration structure. (x ranges from 0 to 5) *Note: Predefined timers can only be controlled from master core and (ResourceMMasterCore). But it can be read from all the available cores.*	
Example(s)	**Action**	**Generated output**

	Configure 100us 32bit predefined timer and generate.
	
	Configure Gpt/General/GptPredefTimer100us32bitEnable as TRUE
	
	Configure GptTimerChannelUsage in Gpt/GptChannelConfigSet/GptChannelConfiguration/GptChannelConfiguration_0/GptTimerChannelUsage as GPT_PREDEF_TIMERCH_100US_32BIT_ENABLED and configure the TOM channels in
		`static const Gpt_100UsPredefTimerChannelConfigType \ Gpt_k100UsPredefTimerChannelConfig_Core1 =`
		`{`
		`  #if (GPT_PREDEF_EXTRA_CH_100US == STD_ON)`
		`  {`
		`    MCU_GTM_TIMER_TOM, /* Timer Type (TOM/ATOM) */`
		`    0x4, /* Timer Number Module No | Timer Channel No */`
		`    0x1000800U, /* Channel Control`

	GtmTimerOutputModuleConfiguration	Register */
		    0x0U, /* CN0 in ticks */
		    0x2710U, /* CM0 in ticks */
		    0x0U, /* CM1 in ticks */
		    0x2710U, /* SR0 in ticks */
		    0x0U, /* SR1 in ticks */
		    0x0U, /* Port Out */
		    0x0U /* Interrupt status and mode*/
		  },
		  #endif
		  {
		    MCU_GTM_TIMER_TOM, /* Timer Type (TOM/ATOM) */
		    0x5, /* Timer Number Module No | Timer Channel No */
		    0x100d800U, /* Channel Control Register */
		    0x0U, /* CN0 in ticks */
		    0xffffU, /* CM0 in ticks */
		    0x0U, /* CM1 in ticks */
		    0xffffU, /* SR0 in ticks */
		    0x0U, /* SR1 in ticks */
		    0x0U, /* Port Out */
		    0x0U /* Interrupt status and mode*/
		  },
		  {
		    MCU_GTM_TIMER_TOM, /* Timer Type (TOM/ATOM) */
		    0x6, /* Timer Number Module No | Timer Channel No */
		    0xd800U, /* Channel Control Register */
		    0x0U, /* CN0 in ticks */
		    0xffffU, /* CM0 in ticks */
		    0x0U, /* CM1 in ticks */
		    0xffffU, /* SR0 in ticks */
		    0x0U, /* SR1 in ticks */
		    0x0U, /* Port Out */
		    0x0U /* Interrupt status and mode*/

		`},`
		`1, /* Is extra Channel for frequency tuning required */`
		`/* Type of the Predef timer enabled : */`
		`GPT_PREDEF_TIMER_100US_32BIT`
		`};`

1.2.7.1 Member: GptGtm100UsTimerInfo0

Table 75 GptGtm100UsTimerInfo0

Name	GptGtm100UsTimerInfo0	
Type	Mcu_17_Gtm_TomAtomChConfigType	
Description	Timer used for frequency tuning (This will be the extra channel)	
Verification method	The verification method is similar to that of GptGtmTimerInfo structure and its related members. Refer Section 1.2.3.5.1 to 1.2.3.5.10 for more details on the generation	
Example(s)	**Action**	**Generated output**
	1.2.3.5	

1.2.7.2 Member: GptGtm100UsTimerInfo1

Table 76 GptGtm100UsTimerInfo1

Name	GptGtm100UsTimerInfo1	
Type	Mcu_17_Gtm_TomAtomChConfigType	
Description	Lower word (First 16bits) of the 32bit timer	
Verification method	The verification method is similar to that of GptGtmTimerInfo structure and its related members. Refer Section 1.2.3.5.1 to 1.2.3.5.10 for more details on the generation	
Example(s)	**Action**	**Generated output**
	1.2.3.5	

1.2.7.3 Member: GptGtm100UsTimerInfo2

Table 77 GptGtm100UsTimerInfo2

Name	GptGtm100UsTimerInfo2	
Type	Mcu_17_Gtm_TomAtomChConfigType	
Description	Upper word (Next 16bits) of the 32bit timer	
Verification method	The verification method is similar to that of GptGtmTimerInfo structure and its related members. Refer Section 1.2.3.5.1 to 1.2.3.5.10 for more details on the generation	
Example(s)	**Action**	**Generated output**
	1.2.3.5	

1.2.7.4 Member: ExtraChRequirement100Us

Table 78 ExtraChRequirement100Us

Name	ExtraChRequirement100Us	
Type	boolean	
Description	Indicates the usage of additional TOM channel for frequency tuning	
Verification method	The generated value of this configuration parameter is TRUE if the input clock to the TOM channel is too high and an additional TOM channel is used to further divide the clock and feed to the predefined timer to arrive at 10KHz frequency The generated value of this configuration parameter is FALSE if the input clock to the TOM channel is 10KHz.	
Example(s)	**Action**	**Generated output**
	Input clock to the TOM is 1MHz	1, /* Is extra Channel for frequency tuning required */
	Input clock to the TOM is 2MHz	1, /* Is extra Channel for frequency tuning required */
	Input clock to the TOM is 10KHz	0, /* Is extra Channel for frequency tuning required */

1.2.7.5 Member: Gpt100UsPredefTimerUsed0

Table 79 Gpt100UsPredefTimerUsed0

Name	Gpt100UsPredefTimerUsed0	
Type	Gpt_PredefTimerType	
Description	Types of Predef timers enabled	
Verification method	This configuration parameter will be generated as GPT_PREDEF_TIMER_100US_32BIT	
Example(s)	**Action**	**Generated output**

	Configure a 100us predefined timer and then generate	`/* Type of the Predef timer enabled : */`
		`  GPT_PREDEF_TIMER_100US_32BIT`

1.2.8 Array: Gpt_ChannelCoreIndex

Table 80 Gpt_ChannelCoreIndex

Name	Gpt_ChannelCoreIndex	
Type	uint32	
Description	Configuration array which holds the core id of each logical channel.	
Verification method	The array is generated to create a mapping of the configured channel to the allocated core. This is generated only when "GptReadAcrossCores cores" is ON. The size of array is equal to the number of GPT channels configured. Channels which are not allocated to any core will be mapped to the Master core.	
Example(s)	**Action**	**Generated output**
	Configure	`static const uint32`

Channel 0,1,3,4 as Core1.
Channel 2 as Core2.
Channel 5 as Core3.
	`Gpt_ChannelCoreIndex[GPT_MAX_CHANNELS] =`
	`{`
	`  0x01U,  /*  Core1*/`
	`  0x01U,  /*  Core1*/`
	`  0x02U,  /*  Core2*/`
	`  0x01U,  /*  Core1*/`
	`  0x01U,  /*  Core1*/`
	`  0x03U,  /*  Core3*/`
	`};`

Configure
Channel 0 as Core0.
Channel 5 as Core1.
Channel 2 as Core2.
Channel 4 as Core3.
 	
Channel 1 and 3 are not allocated to any core.
 	
Configure “Core1” as master core.
	`static const uint32 Gpt_ChannelCoreIndex[GPT_MAX_CHANNELS] =`
	`{`
	`  0x00U,  /*  Core0*/`
	`  0x01U,  /*  Core1*/`
	`  0x02U,  /*  Core2*/`
	`  0x01U,  /*  Core1*/`
	`  0x03U,  /*  Core3*/`
	`  0x01U,  /*  Core1*/`
	`};`

1.3 File: Gpt[_<variant>]_PBcfg.h

The generated header file contains the declaration of the root configuration structure. Post-build time configuration mechanism allows configurable functionality of GPT driver that is deployed as object code. The file is generated in ‘inc’ folder.

1.3.1 Structure: Gpt_Config[_<variant>]

Table 81 Gpt_Config[_<varaint>]

Name	Gpt_Config[_<variant>]	
Type	Gpt_ConfigType	
Description	Declaration of root configuration structure of GPT driver which will be used during initialization.	
Verification method	The generated structure is present in Gpt[_<variant>]_PBcfg.h file. The <variant> indicates the name of the post-build variant. For a variant-aware configuration the structure name is appended with the variant name. For variant-unaware configuration <variant> is ignored.	
Example(s)	**Action**	**Generated output**
	Configure atleast one GPT channel and generate (variant-unaware)	`extern const Gpt_ConfigType Gpt_Config;`

	Configure atleast one GPT channel and generate (variant-aware. Variant name is 'Petrol')	`extern const Gpt_ConfigType Gpt_Config_Petrol;`

Revision history

Major changes since the last revision

Date	Version	Description
2020-10-22	4.0	Released version.
2020-10-21	3.1	Following changes are updated. • Gpt driver chapter moved from MC-ISAR_TC3xx_Config_Verification_Manual_Basic.pdf to this document. • Added macros GPT_READ_ACROSS_CORES and GPT_RUNTIME_ERROR_DETECT. (sections 1.1.33 and 1.1.34) • Added an array Gpt_ChannelCoreIndex (section 1.2.8)
2019-07-19	3.0	Released version.
2019-07-19	2.1	Added information on GPT12 configuration and editorial changes. Newly added sections – 1.1.31, 1.2.3.5 and 1.2.3.6. Updated sections 1.1.25, 1.1.29, 1.1.30, 1.1.32, 1.2.3, 1.2.6 and 1.2.7.
2019-02-27	1.10.0_2.0	Added Pbcfg.h file, Common published information.
2019-02-26	1.10.0_1.0	Initial Release.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2020-10-22
Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Email: erratum@infineon.com

Document reference
Doc_Number