# User Manual

for MPC563xM GPT Driver

Document Number: UM14GPTASR3.0R2.0.0
Rev. 2.0

freescale™

# Contents

**Section Number** **Title** **Page**

## Chapter 1
## Revision History

## Chapter 2
## Introduction



## Chapter 3
## Driver

# Chapter 1
# Revision History

**Table 1-1. Revision History**

| Revision | Date | Author | Description |
|---|---|---|---|
| 1.0 | 20.01.2011 | Chandrakanth.P | Initial Version |
| 2.0 | 28.11.2011 | Hari SS (b10728) | Updated for RTM 2.0.0 |

# Chapter 2
# Introduction

This User Manual describes Freescale Semiconductor AUTOSAR General Purpose Timer (GPT) driver for MPC5634M. AUTOSAR GPT driver configuration parameters and deviations from the specification are described in GPT Driver chapter. AUTOSAR GPT driver requirements and APIs are described in the AUTOSAR GPT driver software specification document. (Table 1 2)

## 2.1 Supported Derivatives

The software described in this document is intended to be used with the following microcontroller devices of Freescale Semiconductor :

**Table 2-1. MPC5634M Derivatives**

| | |
|---|---|
| Freescale Semiconductor | mpc5634m_bga208,<br>mpc5634m_qfp144,<br>mpc5634m_qfp176 |

All of the above microcontroller devices are collectively named as MPC5634M.

## 2.2 Overview

AUTOSAR (Automotive Open System Architecture) is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

## 2.3 About this Manual

This Technical Reference employs the following typographical conventions:

**Boldface** type: Bold is used for important terms, notes and warnings.

*Italic* font: Italic typeface is used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

## 2.4 Acronyms and Definitions

**Table 2-2. Acronyms and Definitions**

| Term | Definition |
|---|---|
| API | Application Programming Interface |
| AUTOSAR | Automotive Open System Architecture |
| ASM | Assembler |
| BSMI | Basic Software Make file Interface |
| GPT | General Purpose Timer |
| DEM | Diagnostic Event Manager |
| DET | Development Error Tracer |
| C/CPP | C and C++ Source Code |
| VLE | Variable Length Encoding |
| N/A | Not Applicable |
| MCU | Micro Controller Unit |
| STM | System Timer Module |
| PIT | Periodic Interrupt Timer |
| RTC | Real Time Clock |
| API | Autonomous Periodic Interrupt |

## 2.5 Reference List

**Table 2-3. Reference List**

| # | Title | Version |
|---|---|---|
| 1 | AUTOSAR 3.0GPT Driver Software Specification Document | V2.2.0 R3.0 Rev 0001 |
| 2 | MPC5634M Reference Manual | Rev. 6, 4 October 2011 |

# Chapter 3
# Driver

## 3.1 Requirements

Requirements for this driver are detailed in the AUTOSAR 3.0GPT Driver Software Specification document (See Table **Reference List**).

## 3.2 Driver Design Sumary

The GPT Driver implements 17 channels on 3 MPC563xM peripherals.

8 channels are implemented on the Enhanced Modular I/O Subsystem (eMIOS) module.

4 channels are implemented on System Timer Module (STM).

5 channels are implemented on Periodic Interrupt Timer with Real Time Interrupt(PITRTI).

**eMIOS channels**

- 2 selectable clock sources
- 8 x 24-bit wide counter buses

**PITRTI timer channel**

- Independent timeout periods for each timer

**STM timer channel**

- One 32-bit up counter with 8-bit prescaler (1 to 256).

- Four 32-bit compare channels
- Independent interrupt source for each channel

**Table 3-1. GPTHardware channels availability for MPC563xM family**

| Device | Total eMIOS channels | Total PITRTI channels | Total STM channels |
|---|---|---|---|
| MPC5634M | 8 ch, 24-bit | 5 ch | 4 ch |

## GPT Dual Clock Mode

The Dual Clock Mode permits to keep the same functionality (preset time constants) of the counters even the clock of the hw module (STM or eMIOS) is changed by the application (for example for power saving reasons).

This is done by using the parameter GptSTMChannelPrescale_Alternate for the STM module and by the parameter eMIOSBusClkPrescale_Alternate for the eMIOS module. These 2 parameters are used to change the local division factor of the module clock.

This feature is an AutoSAR extension of the GPT Driver.

### Interrupt request usage

The interrupts are validated only for the channels configured in plugin. There are generated some constants like followings: #define GPT_PITRTI_CH_x_ISR_USED #define GPT_eMIOS_y_CH_x_ISR_USED #define GPT_STM_CH_x_ISR_USED The processing of intererupts is guarded using #ifdefs on these defines. So the interrupts corresponding to the configured channels are used and the others are removed. An example of the #ifdef guard is:

#if (defined GPT_eMIOS_y_CH_x_ISR_USED)

eMIOS_Gpt_LLD_ProcessInterrupt(eMIOS_y_CH_x);

#endif

### Description of the symbolic names

When the plugin is generated, symbolic names of the channels are created by define macros. The symbolic names can be used as parameter in the calls of the functions.

The templates of the defines are:

#define <GptChannelSymbolicName> <GptChannelID>

where GptChannelSymbolicName is the name of the container, GptChannelConfiguration_x, and GptChannelID is an index also set by the user.

### The Catastrophic Error Recovery (CER) procedure

If the driver detects an error in the API calls (usual wrong parameter values or bad context for calling a function) it raises a CER exception calling a macro, provided by the application level, with the error code as parameter. Because this errors are catastrophic it is mandatory for the user to process the raised exception.

## 3.3 Additional Requirement and Deviations

The driver deviates from the AUTOSAR GPT Driver software specification in some places.

Deviation from the AUTOSAR GPT Driver software specification are listed in table **Gpt Deviation table**

**Table 3-2. Deviations Status Column Description**

| Term | Definition |
|---|---|
| N/A | Not Available |
| N/V | Not Verifiable |
| N/S | Out of scope |
| N/I | Not implemented |
| N/F | Not fully implemented |
| N/R | Unclear requirement |

**Table 3-3. Gpt Deviations Table**

| Requirement | Status | Description | Notes |
|---|---|---|---|
| GPT004 | N/I | Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the GPT device specific implementation specification. The classification and enumeration shall be compatible to the errors listed | No additional errors are supported by hardware, hence no additional errors are implemented. |

## 3.4 Function Definitions

APIs of all functions supported by the driver are as per AUTOSAR GPT Driver software specification Version 3.0

### 3.4.1 Function Gpt_Init

GPT driver initialization function.

**Prototype:** `void Gpt_Init(const Gpt_ConfigType *configPtr);`

**Table 3-4. Gpt_Init Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| const `Gpt_ConfigType *` | configPtr | **input** | Pointer to a selected configuration structure |

Service for driver initialization. The Initialization function shall initialize all relevant registers of the configured hardware with the values of the structure referenced by the parameter ConfigPtr.

All time units used within the API services of the GPT driver shall be of the unit ticks.

This function shall only initialize the configured resources. Resources that are not configured in the configuration file shall not be touched.

The following rules regarding initialization of controller registers shall apply to the GPT Driver implementation:

[1] If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register

[2] If the register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver

[3] If the register can affect several hardware modules and if it is not an I/O register it shall be initialized by the MCU driver

[4] One-time writable registers that require initialization directly after reset shall be initialized by the startup code

[5] All other registers shall be initialized by the startup code

**Pre Condition:** The driver shall not be already initialized. Otherwise the function raises the development error GPT_E_ALREADY_INITIALIZED.

### 3.4.2 Function Gpt_DeInit

GPT driver de-initialization function.

**Prototype:** `void Gpt_DeInit(void);`

Service for deinitializing all hardware timer channels to their power on reset state.

The state of the peripheral after DeInit shall be the same as after power on reset.

The service influences only the peripherals, which are allocated by static configuration and the runtime configuration set passed by the previous call of `Gpt_Init()`

**Pre Conditions:**

The driver needs to be initialized before calling it. Otherwise, the function Gpt_DeInit shall raise the development error GPT_E_UNINIT and leave the desired deinitialization functionality without any action.

No channel in RUNNING status. Otherwise, it shall raise the development error GPT_E_BUSY and leave the desired deinitialization functionality without any action.

The precompiling parameter GPT_DEINIT_API = STD_ON

### 3.4.3 Function Gpt_EnableNotification

GPT driver function for enabling the notification for a timer channel.

**Prototype:** `void Gpt_EnableNotification(Gpt_ChannelType channel);`

**Table 3-5. Gpt_EnableNotification Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Gpt_ChannelType` | channel | **input** | channel id |

Service for enabling the notification for a channel during runtime.

This function can be called, while the timer is already running.

Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel.

**Pre Conditions:**

The driver shall be initialized before to call it. Otherwise the function shall raise the development error GPT_E_UNINIT.

The parameter 'channel' must have a valid value. Otherwise the function shall raise the development error GPT_E_PARAM_CHANNEL.

The precompiling parameter GPT_ENABLE_DISABLE_NOTIFICATION_API = STD_ON

## 3.4.4 Function Gpt_DisableNotification

GPT driver function for disabling the notification for a timer channel.

**Prototype:** `void Gpt_DisableNotification(Gpt_ChannelType channel);`

**Table 3-6. Gpt_DisableNotification Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Gpt_ChannelType` | channel | **input** | channel id |

Service for disabling the notification for a channel during runtime.

This function can be called, while the timer is already running

When disabled, no notification will be sent. When re-enabled again, the user will not be notified of events, occurred while notifications have been disabled.

Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel.

**Pre Conditions:**

The driver shall be initialized before to call it. Otherwise the function shall raise the development error GPT_E_UNINIT.

The parameter 'channel' must have a valid value. Otherwise the function shall raise the development error GPT_E_PARAM_CHANNEL.

The precompiling parameter GPT_ENABLE_DISABLE_NOTIFICATION_API = STD_ON

## 3.4.5 Function Gpt_DisableWakeup

GPT driver function for disabling the wakeup interrupt invocation for a timer channel.

**Prototype:** `void Gpt_DisableWakeup(Gpt_ChannelType channel);`

**Table 3-7. Gpt_DisableWakeup Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Gpt_ChannelType` | channel | **input** | channel id |

This service shall disable the wakeup interrupt invocation of a single GPT channel.

Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel.

**Pre Conditions:**

The driver shall be initialized before to call it. Otherwise the function shall raise the development error GPT_E_UNINIT.

The parameter 'channel' must have a valid value. Otherwise the function shall raise the development error GPT_E_PARAM_CHANNEL.

The selected channel must be configured as wakeup capable. Otherwise the function shall raise the development error GPT_E_PARAM_CHANNEL.

The precompiling parameter GPT_WAKEUP_FUNCTIONALITY_API = STD_ON

## 3.4.6 Function Gpt_EnableWakeup

GPT driver function for enabling the wakeup interrupt invocation for a timer channel.

**Prototype:** `void Gpt_EnableWakeup(Gpt_ChannelType channel);`

**Table 3-8. Gpt_EnableWakeup Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Gpt_ChannelType` | channel | **input** | channel id |

This service shall re-enable the wakeup interrupt invocation of a single GPT channel.

If supported by hardware and enabled, an internal hardware timer can serve as a wakeup source

Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel.

**Pre Conditions:**

The driver shall be initialized before to call it. Otherwise the function shall raise the development error GPT_E_UNINIT.

The parameter 'channel' must have a valid value. Otherwise the function shall raise the development error GPT_E_PARAM_CHANNEL.

The selected channel must be configured as wakeup capable. Otherwise the function shall raise the development error GPT_E_PARAM_CHANNEL.

The precompiling parameter GPT_WAKEUP_FUNCTIONALITY_API = STD_ON

## 3.4.7 Function Gpt_GetTimeElapsed

GPT driver function for fetching the elapsed timer value.

**Prototype:** `Gpt_ValueType Gpt_GetTimeElapsed(Gpt_ChannelType channel);`

**Table 3-9. Gpt_GetTimeElapsed Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Gpt_ChannelType` | channel | **input** | channel id |

Service for querying the time already elapsed.

In one shot mode, this is the value relative to the point in time, the channel has been started with Gpt_StartTimer (calculated by the normal operation function by subtracting the current minus the initial timer value and returning the absolute value).

In continuous mode, the function returns the timer value relative to the last timeout / the start of the channel.

All time units used within the API services of the GPT driver shall be of the unit ticks.

Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel

To get times out of register values it is necessary to know the oscillator frequency, prescalers and so on. Since these settings are made in MCU and/or in other modules it is not possible to calculate such times. Hence the conversions between time and ticks shall be part of an upper layer.

**Pre Conditions:**

The driver needs to be initialized before calling it. Otherwise the function shall raise the development error GPT_E_UNINIT and return 0.

The parameter 'channel' must have a valid value. Otherwise, the function shall raise the development error GPT_E_PARAM_CHANNEL and return 0.

The selected channel must be in RUNNING status. Otherwise, the function shall raise the development error GPT_E_NOT_STARTED and return 0.

The precompiling parameter GPT_TIME_ELAPSED_API = STD_ON

**Return:** Gpt_ValueType -Elapsed Time in number of ticks

## 3.4.8 Function Gpt_GetTimeRemaining

GPT driver function for fetching the remaining timer value.

**Prototype:** `Gpt_ValueType Gpt_GetTimeRemaining(Gpt_ChannelType channel);`

**Table 3-10. Gpt_GetTimeRemaining Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Gpt_ChannelType` | channel | **input** | channel id |

This function returns the timer value remaining until the next timeout period will expire (calculated by the normal operation function by subtracting the timeout minus the current timer value and returning the absolute value)

All time units used within the API services of the GPT driver shall be of the unit ticks.

Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel.

To get times out of register values it is necessary to know the oscillator frequency, prescalers and so on. Since these settings are made in MCU and/or in other modules it is not possible to calculate such times. Hence the conversions between time and ticks shall be part of an upper layer.

**Pre Conditions:**

The driver needs to be initialized before calling it. Otherwise the function shall raise the development error GPT_E_UNINIT and return 0.

The parameter 'channel' must have a valid value. Otherwise, the function shall raise the development error GPT_E_PARAM_CHANNEL and return 0.

The selected channel must be in RUNNING status. Otherwise, the function shall raise the development error GPT_E_NOT_STARTED and return 0.

The precompiling parameter GPT_TIME_REMAINING_API = STD_ON

**Return:** Gpt_ValueType -Remaining Time in number of ticks

## 3.4.9 Function Gpt_SetMode

GPT driver function for setting the operation mode.

**Prototype:** `void Gpt_SetMode(Gpt_ModeType mode);`

**Table 3-11. Gpt_SetMode Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Gpt_ModeType` | mode | **input** | operation mode |

Service for GPT mode selection. This service shall set the operation mode to the given mode parameter .

When sleep mode is requested, the ECU State Manager calls Gpt_SetMode with mode parameter "GPT_MODE_SLEEP" and prepares the GPT for sleep mode. The MCU Driver is then putting the controller into SLEEP mode

**Pre Conditions:**

The driver shall be initialized before to call it. Otherwise the function shall raise the development error GPT_E_UNINIT.

The parameter 'mode' must have a valid value. Otherwise the function shall raise the development error GPT_E_PARAM_CHANNEL.

The precompiling parameter GPT_WAKEUP_FUNCTIONALITY_API = STD_ON

## 3.4.10 Function Gpt_StartTimer

GPT driver function for starting a timer channel.

**Prototype:** `void Gpt_StartTimer(Gpt_ChannelType channel, Gpt_ValueType value);`

**Table 3-12. Gpt_StartTimer Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Gpt_ChannelType` | channel | **input** | channel id |
| `Gpt_ValueType` | value | **input** | timeout period (in number of ticks) after a notification shall occur |

The function Gpt_StartTimer shall start the selected timer channel with a defined timeout period.

The function Gpt_StartTimer shall invoke the configured notification for that channel (see also GPT292) after the timeout period referenced via the parameter value (if enabled).

All time units used within the API services of the GPT driver shall be of the unit ticks.

In production mode no error is generated. The rational is that it adds no additional functionality to the driver. In this case the timer will be restarted with the timeout value, given as a parameter to the service.

Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel.

To get times out of register values it is necessary to know the oscillator frequency, prescalers and so on. Since these settings are made in MCU and/or in other modules it is not possible to calculate such times. Hence the conversions between time and ticks shall be part of an upper layer.

**Pre Condition:**

The driver needs to be initialized before calling it. Otherwise the function shall raise the development error GPT_E_UNINIT.

The value of the timeout must to be valid for the selected channel. Otherwise, the function shall raise the development error GPT_E_PARAM_CHANNEL.

The parameter 'channel' must have a valid value. Otherwise, the function shall raise the development error GPT_E_PARAM_CHANNEL.

The selected channel must be in the STOPPED status. Otherwise, the function shall raise the development error GPT_E_BUSY.

**Post Condition:**

The selected channel status is set to RUNNING.

## 3.4.11 Function Gpt_StopTimer

GPT driver function for stopping a timer channel.

**Prototype:** `void Gpt_StopTimer(Gpt_ChannelType channel);`

**Table 3-13. Gpt_StopTimer Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Gpt_ChannelType` | channel | **input** | channel id |

Service for stopping the selected timer channel

Stopping a timer channel, not been started before will not return a development error Timer channels configured in one shot mode are stopped automatically, when the timeout period has expired.

Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel.

**Pre Condition:**

The driver needs to be initialized before calling it. Otherwise the function shall raise the development error GPT_E_UNINIT.

The parameter 'channel' must have a valid value. Otherwise, the function shall raise the development error GPT_E_PARAM_CHANNEL.

**Post Condition:**

The selected channel status is set to STOPPED.

## 3.4.12 Function Gpt_GetVersionInfo

This function returns the version information of this module.

**Prototype:** `void Gpt_GetVersionInfo(Std_VersionInfoType *versioninfo);`

**Table 3-14. Gpt_GetVersionInfo Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Std_VersionInfoType *` | versioninfo | **output** | - pointer to location to store version info |

This service returns the version information of this module. The version information includes: Module Id Vendor Id Vendor specific version numbers (BSW00407)

**Pre:**The precompiling parameter GPT_VERSION_INFO_API = STD_ON

## 3.4.13 Function Gpt_Cbk_CheckWakeup

GPT driver function for checking if a wakeup capable GPT channel is the source for a wakeup event.

**Prototype:** `void Gpt_Cbk_CheckWakeup(EcuM_WakeupSourceType wakeupSource);`

**Table 3-15. Gpt_Cbk_CheckWakeup Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `EcuM_WakeupSourceType` | wakeupSource | **input** | wakeup source |

Checks if a wakeup capable GPT channel is the source for a wakeup event and calls the ECU state manager service EcuM_SetWakeupEvent in case of a valid GPT channel wakeup event.

**Pre Conditions:**

The driver shall be initialized before to call it. Otherwise the function shall raise the development error GPT_E_UNINIT.

The precompiling parameter GPT_WAKEUP_FUNCTIONALITY_API = STD_ON

### 3.4.14 Function Gpt_SetClockMode

This function changes the channel prescaler.

**Prototype:** `void Gpt_SetClockMode(Gpt_ClockModeType clkMode);`

**Table 3-16. Gpt_SetClockMode Arguments**

| Type | Name | Direction | Description |
|---|---|---|---|
| `Gpt_ClockModeType` | clkMode | **input** | prescaler setting ( NORMAL or ALTERNATE ) |

This function sets all channels prescalers based on the input mode.

**pre::** Gpt_Init must be called before.

## 3.5 Configuration Parameters

As per the AUTOSAR specification the driver has two types of configurations parameters:**Pre-Compile** parameters and **Post-Build** parameters. Pre-Compile parameters are stored in the file Gpt_Cfg.h and Gpt_Cfg.c. Post-Build parameters are stored in the file Gpt_PBcfg.c.

The files to be used for different configuration types are listed below:

1. **Variant PC**:Gpt_Cfg.h, Gpt_Cfg.c
2. V**ariant PB**: Gpt_Cfg.h, Gpt_PBcfg.c
3. **Variant LT**: Not Applicable

A section for **Gpt_PBcfg.c** file is needed in linker file to place the post build configuration in desired location. Please refer to section-8 of AUTOSAR_SWS_C_ImplementationRules file for complete details on configuration types.

### 3.5.1 Pre-Compile parameters

The following picture gives an example of configuration screen for all GPT precompile parameters in Tresos® Studio configuration tool from Tresos 2010a.sr4 20100415-release2010a-sr4

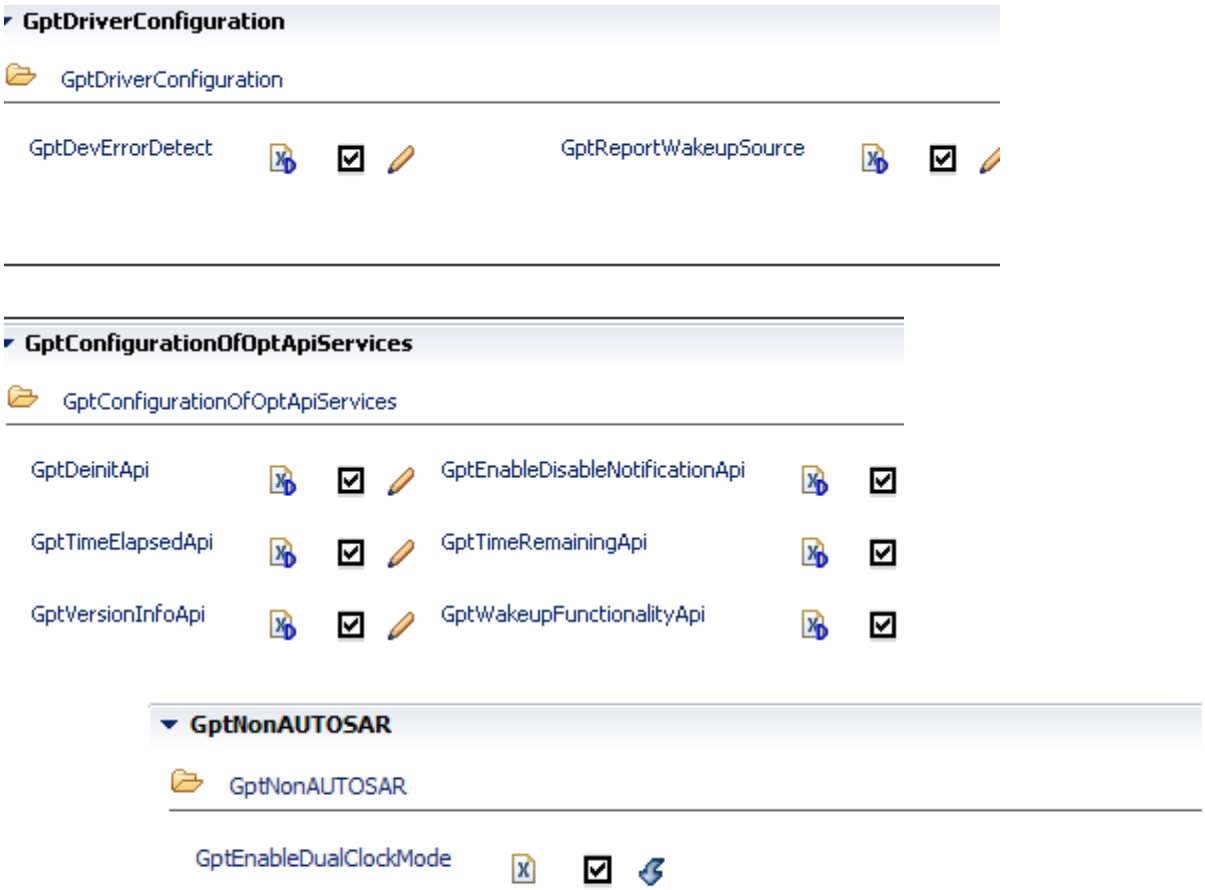



**Figure 3-1. GptNonAutosar**

## NOTE

If pre-compile variant is preferred,

- IMPLEMENTATION_CONFIG_VARIANT in Tresos GUI should be selected as “VariantPreCompile” as shown below.

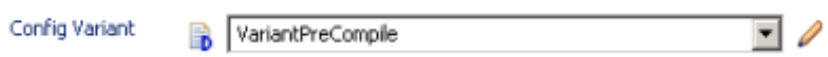



**Figure 3-2. Post-Config-variant**

- Gpt_Cfg.c should be compiled
- Gpt_PBcfg.c should not be compiled

### 3.5.1.1 GptDevErrorDetect

**Table 3-17. GptDevErrorDetect**

| | |
|---|---|
| **Description** | Enables/Disables development error detection. Defines whether support of development error detection should be included at compile time (STD_ON) or excluded (STD_OFF). When enabled API parameter checking is done at runtime. |
| **Class** | Autosar Parameter |
| **Range** | True, False |
| **Default** | True |
| **Source File** | Gpt_Cfg.h |
| **Source Representation** | *#define GPT_DEV_ERROR_DETECT STD_OFF / STD_ON* |
| ***NOTE*** | *Setting this control to false will generate code that is reduced, but some Autosar requirements are not tested (no Det errors are reported in this way).* |

### 3.5.1.2 GptReportWakeupSource

**Table 3-18. GptReportWakeupSource**

| | |
|---|---|
| **Description** | Enables/Disables wakeup source reporting |
| **Class** | Autosar Parameter |
| **Range** | True, False |
| **Default** | True |
| **Source File** | Gpt_Cfg.h |
| **Source Representation** | *#define GPT_REPORT_WAKEUP_SOURCE STD_OFF / STD_ON* |

### 3.5.1.3 GptDeinitApi

**Table 3-19. GptDeinitApi**

| | |
|---|---|
| **Description** | Adds / removes the service Gpt_DeInit() from the code. |
| **Class** | Autosar Parameter |
| **Range** | True, False |
| **Default** | True |
| **Source File** | Gpt_Cfg.h |
| **Source Representation** | *#define GPT_DEINIT_API STD_OFF / STD_ON* |
| ***NOTE*** | *Setting this control to false will generate code that is reduced, but some AutoSAR requirements are not fulfilled.* |

### 3.5.1.4 GptEnableDisableNotificationApi

**Table 3-20. GptEnableDisableNotificationApi**

| | |
|---|---|
| **Description** | Adds / removes the services Gpt_EnableNotification() and Gpt_DisableNotification from the code. |
| **Class** | Autosar Parameter |
| **Range** | True, False |
| **Default** | True |
| **Source File** | Gpt_Cfg.h |
| **Source Representation** | *#define GPT_ENABLE_DISABLE_NOTIFICATION_API STD_OFF / STD_ON* |
| ***NOTE*** | *Setting this control to false will generate code that is reduced, but some AutoSAR requirements are not fulfilled.* |

### 3.5.1.5 GptTimeElapsedApi

**Table 3-21. GptTimeElapsedApi**

| | |
|---|---|
| **Description** | Adds / removes the service Gpt_GetTimeElapsed() from the code |
| **Class** | Autosar Parameter |
| **Range** | True, False |
| **Default** | True |
| **Source File** | Gpt_Cfg.h |
| **Source Representation** | *#define GPT_TIME_ELAPSED_API STD_OFF / STD_ON* |
| ***NOTE*** | *Setting this control to false will generate code that is reduced, but some AutoSAR requirements are not fulfilled.* |

### 3.5.1.6 GptTimeRemainingApi

**Table 3-22. GptTimeRemainingApi**

| | |
|---|---|
| **Description** | Adds / removes the service Gpt_GetTimeRemaining() from the code. |
| **Class** | Autosar Parameter |
| **Range** | True, False |
| **Default** | True |
| **Source File** | Gpt_Cfg.h |
| **Source Representation** | *#define GPT_TIME_REMAINING_API STD_OFF / STD_ON* |
| ***NOTE*** | *Setting this control to false will generate code that is reduced, but some AutoSAR requirements are not fulfilled.* |

### 3.5.1.7 GptVersionInfoApi

**Table 3-23. GptVersionInfoApi**

| | |
|---|---|
| **Description** | Adds / removes the service Gpt_GetVersionInfo() from the code. |
| **Class** | Autosar Parameter |
| **Range** | True, False |
| **Default** | True |
| **Source File** | Gpt_Cfg.h |
| **Source Representation** | *#define GPT_VERSION_INFO_API STD_OFF / STD_ON* |
| ***NOTE*** | *Setting this control to false will generate code that is reduced, but some AutoSAR requirements are not fulfilled.* |

### 3.5.1.8 GptWakeupFunctionalityApi

**Table 3-24. GptWakeupFunctionalityApi**

| | |
|---|---|
| **Description** | Adds / removes the services Gpt_SetMode(), Gpt_EnableWakeup() Gpt_DisableWakeup() and Gpt_Cbk_CheckWakeup() from the code. |
| **Class** | Autosar Parameter |
| **Range** | True, False |
| **Default** | True |
| **Source File** | Gpt_Cfg.h |
| **Source Representation** | *#define GPT_WAKEUP_FUNCTIONALITY_API STD_OFF / STD_ON* |
| ***NOTE*** | *Setting this control to false will generate code that is reduced, but some AutoSAR requirements(the wakeup functionality) are not fulfilled.* |

### 3.5.1.9 GptEnableDualClockMode

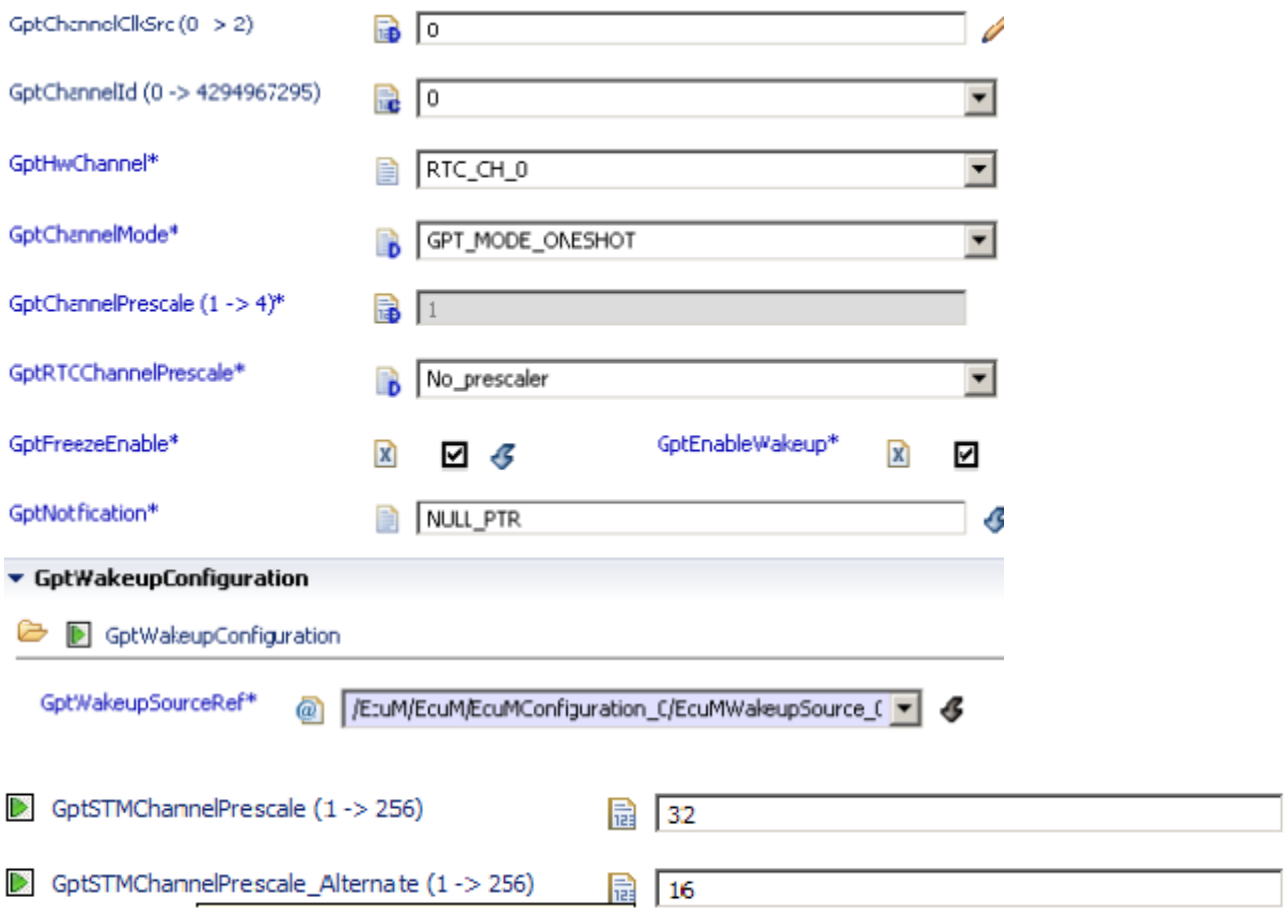
**Table 3-25. GptEnableDualClockMode**

| | |
|---|---|
| **Description** | Enables prescaler settings at mode transition. |
| **Class** | Implementation specific Non-Autosar Parameter |
| **Range** | True, False |
| **Default** | True |
| **Source File** | Gpt_Cfg.h |
| **Source Representation** | *#define GPT_DUAL_CLOCK_MODE STD_OFF / STD_ON* |
| ***NOTE*** | *Setting this control to true will generate code that implements the nonAutoSAR Dual Clock mode support.* |

## 3.5.2 Post-Build Parameters

Post Build parameters, their possible values and their meaning are described in the following text. The Post Build parameters are implemented as constant structures and arrays stored in flash memory of the MCU.

The following picture gives an example of configuration screen for all postbuild parameters in Tresos® Studio configuration tool from Tresos 2010a.sr4 20100415-release2010a-sr4.




### 3.5.2.1 Structure Gpt_ConfigType

Gpt configuration type - this is the type of the data structure including the configuration set required for initializing the GPT driver.

#### Declaration

```
typedef struct
{
  uint8 Gpt_Channel_Count,
  const Gpt_LLD_ChannelConfigType * Gpt_ChannelConfigPtr
} Gpt_ConfigType;
```

**Table 3-26. Structure Gpt_ConfigType member description**

| Member | Description |
|---|---|
| Gpt_Channel_Count | the number of GPT channels (configured in tresos plugin builder) |
| Gpt_ChannelConfigPtr | pointer to the GPT channel configuration |

### 3.5.2.1.1 Structure Gpt_LLD_ChannelConfigType

Gpt channel configuration type.

## Declaration

```
typedef struct
{
  Gpt_ChannelType Gpt_HW_Channel,
  Gpt_ChannelType Gpt_HW_Module,
  Gpt_ChannelModeType Gpt_ChannelMode,
  uint8 Gpt_ChannelClkSrc,
  Gpt_PrescaleType Gpt_ChannelPrescale,
  Gpt_PrescaleType Gpt_ChannelPrescale_Alternate,
  boolean Gpt_FreezeEnable,
  boolean Gpt_EnableWakeup,
  Gpt_NotifyType Gpt_Notification,
  Gpt_WakeUpType Gpt_ChannelWakeUpInfo
} Gpt_LLD_ChannelConfigType;
```

**Table 3-27. Structure Gpt_LLD_ChannelConfigType member description**

| Member | Description |
|---|---|
| Gpt_HW_Channel | GPT hw channel ID. |
| Gpt_HW_Module | GPT hardware module used(eMios, PIT, etc). |
| Gpt_ChannelMode | GPT channel mode. |
| Gpt_ChannelClkSrc | clock source |
| Gpt_ChannelPrescale | GPT channel prescaler value. |
| Gpt_ChannelPrescale_Alternate | ch 2nd prescaler val |
| Gpt_FreezeEnable | GPT channel freeze enable indication. |
| Gpt_EnableWakeup | GPT channel WakeUp enable indication. |
| Gpt_Notification | pointer to the GPT channel notification function |
| Gpt_ChannelWakeUpInfo | EcuM wake up reference, corresponding to the GPT channel - descripion unclear. |

## 3.5.2.2 GptChannelId

**Table 3-28. GptChannelId**

| | |
|---|---|
| **Description** | Channel Id of the GPT channel. This value will be assigned to the symbolic name derived of the GptChannelConfiguration container short name. |

*Table continues on the next page...*

**Table 3-28. GptChannelId (continued)**

| | |
|---|---|
| **Class** | Autosar Parameter |
| **Range** | 0 -> 4294967295 |
| **Default** | 0 |
| **Source File** | Gpt_Cfg.h |
| **Source Representation** | *#define GptChannelConfiguration_0 0* |

### 3.5.2.3 GptSTMChannelPrescale

**Table 3-29. GptSTMChannelPrescale**

| | |
|---|---|
| **Description** | Selects the clock divide value for the STM prescaler (1-256) |
| **Class** | Implementation specific Non-Autosar Parameter |
| **Range** | 1 - 256 |
| **Default** | 1 |
| **Source File** | Gpt_Cfg.c,Gpt_PBcfg.c |
| **Source Representation** | `STATIC CONST(Gpt_LLD_ChannelConfigType, GPT_CONST)`<br>`Gpt_InitChannelPC_1[GPT_CHANNEL_NUM_1]=`<br>`{`<br>`  {`<br>`  (Gpt_ChannelType)(STM_CH_2), /* GPT physical channel no. */`<br>`  (Gpt_ChannelType)(GPT_STM_MODULE), /* hardware module ID */`<br>`  GPT_MODE_ONESHOT, /* Timer mode:continous/one-shot */`<br>`  (uint8)0, /* not used*/`<br>`  **(Gpt_PrescaleType)(0U)**, /* STM Clock divider */`<br>`  (boolean)TRUE, /* Freeze Enable */`<br>`  (boolean)FALSE, /* Wakeup capability */`<br>`  NULL_PTR, /* Channel notification */`<br>`  (Gpt_WakeUpType)(0) /* Wakeup information */`<br>`  }`<br>`};` |

### 3.5.2.4 GptSTMChannelPrescaleAlternate

**Table 3-30. GptSTMChannelPrescaleAlternate**

| | |
|---|---|
| **Description** | This is a non AutoSAR feature implementation.Selects the clock divide value for the STM prescaler (1-256) in dual mode |
| **Class** | Implementation specific Non-Autosar Parameter |
| **Range** | 1 - 256 |
| **Default** | 1 |
| **Source File** | Gpt_Cfg.c,Gpt_PBcfg.c |

*Table continues on the next page...*

**Table 3-30. GptSTMChannelPrescaleAlternate (continued)**

| | |
|---|---|
| **Source Representation** | <pre>STATIC CONST(Gpt_LLD_ChannelConfigType, GPT_CONST)<br>Gpt_InitChannelPB_1[GPT_CHANNEL_NUM_1]=<br>{<br><br>    {<br>        (Gpt_ChannelType)(STM_CH_2), /* GPT physical channel no. */<br>        (Gpt_ChannelType)(GPT_STM_MODULE), /* hardware module ID */<br>        GPT_MODE_CONTINOUS, /* Timer mode:continous/one-shot */<br>        (uint8)0, /* not used*/<br>        (Gpt_PrescaleType)(49U), /* STM Clock divider */<br>#if GPT_DUAL_CLOCK_MODE==STD_ON<br>       (Gpt_PrescaleType)(39U), /* STM Clock Dual Mode divider */<br>    #endif<br>          (boolean)TRUE, /* Freeze Enable */<br>        (boolean)FALSE, /* Wakeup capability */<br>        GptTestCallback02, /* Channel notification */<br>        (Gpt_WakeUpType)(0) /* Wakeup information */<br>    }<br>}</pre> |

## 3.5.2.5 GptHwChannel

**Table 3-31. GpHwChannel**

| | |
|---|---|
| **Description** | Selects the physical GPT Channel. |
| **Class** | Implementation specific Non-Autosar parameter |
| **Range** | EMIOS_0_CH_0<br>EMIOS_0_CH_8<br>EMIOS_0_CH_9<br>EMIOS_0_CH_10<br>EMIOS_0_CH_12<br>EMIOS_0_CH_14<br>EMIOS_0_CH_15<br>EMIOS_0_CH_23<br>PITRTI_RTC_CH_0<br>PITRTI_CH_0<br>PITRTI_CH_1<br>PITRTI_CH_2<br>PITRTI_CH_3<br>STM_CH_0<br>STM_CH_1<br>STM_CH_2<br>STM_CH_3 |
| **Default** | EMIOS_0_CH_0 |
| **Source File** | Gpt_Cfg.h |

*Table continues on the next page...*

**Table 3-31. GpHwChannel (continued)**

| | |
|---|---|
| **Source Representation** | `STATIC CONST(Gpt_LLD_ChannelConfigType, GPT_CONST)`<br>`Gpt_InitChannelPC_1[GPT_CHANNEL_NUM_1]=`<br>`{`<br>`  {`<br>`  **(Gpt_ChannelType)(EMIOS_0_CH_0), /* GPT physical channel no. */**`<br>`  (Gpt_ChannelType)(GPT_EMIOS_MODULE), /* hardware module ID */`<br>`  GPT_MODE_ONESHOT, /* Timer mode:continous/one-shot */`<br>`  (uint8)0, /* not used*/`<br>`  (Gpt_PrescaleType)EMIOS_GPT_LLD_CH_PRES_1,/* EMIOS Clock divider */`<br>`  (boolean)TRUE, /* Freeze Enable */`<br>`  (boolean)FALSE, /* Wakeup capability */`<br>`  NULL_PTR, /* Channel notification */`<br>`  (Gpt_WakeUpType)(0) /* Wakeup information */`<br>`  }`<br>`};` |

## 3.5.2.6 GptChannelMode

**Table 3-32. GptChannelMode**

| | |
|---|---|
| **Description** | Specifies the behaviour of the timer channel after the timeout has expired |
| **Class** | Autosar Parameter |
| **Range** | GPT_MODE_ONESHOT, GPT_MODE_CONTINOUS |
| **Default** | GPT_MODE_ONESHOT |
| **Source File** | Gpt_Cfg.c,Gpt_PBcfg.c |
| **Source Representation** | `STATIC CONST(Gpt_LLD_ChannelConfigType, GPT_CONST)`<br>`Gpt_InitChannelPC_1[GPT_CHANNEL_NUM_1]=`<br>`{`<br>`  {`<br>`  (Gpt_ChannelType)(PITRTI_RTC_CH_0), /* GPT physical channel no. */`<br>`  (Gpt_ChannelType)(GPT_PITRTI_MODULE), /* hardware module ID */`<br>`  **GPT_MODE_ONESHOT, /* Timer mode:continous/one-shot */**`<br>`  (uint8)0, /* not used*/`<br>`  (Gpt_PrescaleType)0, /* Not used */`<br>`  (boolean)TRUE, /* Freeze Enable */`<br>`  (boolean)FALSE, /* Wakeup capability */`<br>`  NULL_PTR, /* Channel notification */`<br>`  (Gpt_WakeUpType)(0) /* Wakeup information */`<br>`  }`<br>`};` |

## 3.5.2.7 GptChannelPrescale

**Table 3-33. GptChannelPrescale**

| | |
|---|---|
| **Description** | GPT module specific prescaler factor per channel. Selects the clock divide value for the prescaler. This parameter is used ONLY for EMIOS channels. Select the clock divider between 1- 4 only. |

*Table continues on the next page...*

**Table 3-33. GptChannelPrescale (continued)**

| | |
|---|---|
| **Class** | Autosar Parameter |
| **Range** | 1-4 |
| **Default** | 1 |
| **Source File** | Gpt_Cfg.c,Gpt_PBcfg.c |
| **Source Representation** | `STATIC CONST(Gpt_LLD_ChannelConfigType, GPT_CONST) Gpt_InitChannelPC_1[GPT_CHANNEL_NUM_1]= { { (Gpt_ChannelType)(EMIOS_0_CH_0), /* GPT physical channel no. */ (Gpt_ChannelType)(GPT_EMIOS_MODULE), /* hardware module ID */ GPT_MODE_ONESHOT, /* Timer mode:continous/one-shot */ (uint8)0, /* not used*/ (Gpt_PrescaleType)EMIOS_GPT_LLD_CH_PRES_1,/* EMIOS Clock divider */ (boolean)TRUE, /* Freeze Enable */ (boolean)FALSE, /* Wakeup capability */ NULL_PTR, /* Channel notification */ (Gpt_WakeUpType)(0) /* Wakeup information */ } };` |

## 3.5.2.8 GptChannelPrescaleAlternate

**Table 3-34. GptChannelPrescaleAlternate**

| | |
|---|---|
| **Description** | This is a non AuroSAR feature implementation.GPT module specific prescaler factor per channel in dual mode.Selects the clock divide value for the prescaler. |
| **Class** | Non Autosar Parameter |
| **Range** | 1-4 |
| **Default** | 1 |
| **Source File** | Gpt_Cfg.c,Gpt_PBcfg.c |

*Table continues on the next page...*

**Table 3-34. GptChannelPrescaleAlternate (continued)**

| | |
|---|---|
| **Source Representation** | <pre>STATIC CONST(Gpt_LLD_ChannelConfigType, GPT_CONST)<br>Gpt_InitChannelPB_1[GPT_CHANNEL_NUM_1]=<br>{<br>    {<br>        (Gpt_ChannelType)(EMIOS_0_CH_8), /* GPT physical channel<br>no. */<br>        (Gpt_ChannelType)(GPT_EMIOS_MODULE), /* hardware module ID<br>*/<br>       GPT_MODE_ONESHOT, /* Timer mode:continous/one-shot */<br>       (uint8)0, /* not used*/<br>       (Gpt_PrescaleType)EMIOS_GPT_LLD_CH_PRES_1,/* EMIOS Clock<br>divider */<br>#if GPT_DUAL_CLOCK_MODE==STD_ON<br>           (Gpt_PrescaleType)EMIOS_GPT_LLD_CH_PRES_3,/* EMIOS<br>Clock Dual Mode divider */<br>    #endif<br>         (boolean)TRUE, /* Freeze Enable */<br>       (boolean)FALSE, /* Wakeup capability */<br>       GptTestCallback01, /* Channel notification */<br>       (Gpt_WakeUpType)(0) /* Wakeup information */<br>    }<br>}</pre> |

## 3.5.2.9 GptFreezeEnable

**Table 3-35. GptFreezeEnable**

| | |
|---|---|
| **Description** | Select to set Freeze enable for EMIOS, STM, PITRTI channels |
| **Class** | Implementation specific Non-Autosar Parameter |
| **Range** | True, False |
| **Default** | True |
| **Source File** | Gpt_Cfg.h |
| **Source Representation** | <pre>STATIC CONST(Gpt_LLD_ChannelConfigType, GPT_CONST)<br>Gpt_InitChannelPC_1[GPT_CHANNEL_NUM_1]=<br>{<br>  {<br>    (Gpt_ChannelType)(EMIOS_0_CH_0), /* GPT physical channel no. */<br>    (Gpt_ChannelType)(GPT_EMIOS_MODULE), /* hardware module ID */<br>    GPT_MODE_ONESHOT, /* Timer mode:continous/one-shot */<br>    (uint8)0, /* not used*/<br>    (Gpt_PrescaleType)EMIOS_GPT_LLD_CH_PRES_1,/* EMIOS Clock<br>divider */<br>    (boolean)TRUE, /* Freeze Enable */<br>    (boolean)FALSE, /* Wakeup capability */<br>    NULL_PTR, /* Channel notification */<br>    (Gpt_WakeUpType)(0) /* Wakeup information */<br>  }<br>};</pre> |

### 3.5.2.10 GptEnableWakeup

**Table 3-36. GptEnableWakeup**

| | |
|---|---|
| **Description** | Enables wakeup capability of CPU for a channel when timeout period expires. This might be different to enabling the notification depending on hardware capabilities |
| **Class** | Autosar Parameter |
| **Range** | True, False |
| **Default** | False |
| **Source File** | Gpt_Cfg.c,Gpt_PBcfg.c |
| **Source Representation** | `STATIC CONST(Gpt_LLD_ChannelConfigType, GPT_CONST) Gpt_InitChannelPC_1[GPT_CHANNEL_NUM_1]= { { (Gpt_ChannelType)(EMIOS_0_CH_0), /* GPT physical channel no. */ (Gpt_ChannelType)(GPT_EMIOS_MODULE), /* hardware module ID */ GPT_MODE_ONESHOT, /* Timer mode:continous/one-shot */ (uint8)0, /* not used*/ (Gpt_PrescaleType)EMIOS_GPT_LLD_CH_PRES_1,/* EMIOS Clock divider */ (boolean)TRUE, /* Freeze Enable */ **(boolean)FALSE, /* Wakeup capability */** NULL_PTR, /* Channel notification */ (Gpt_WakeUpType)(0) /* Wakeup information */ } };` |

### 3.5.2.11 GptNotification

**Table 3-37. GptNotification**

| | |
|---|---|
| **Description** | Function pointer to callback function.Note:The field is editable only,If the switch GptEnableDisableNotificationApi is true. |
| **Class** | Autosar Parameter |
| **Range** | NA |
| **Default** | NULL_PTR |
| **Source File** | Gpt_Cfg.C,Gpt_PBcfg.c |
| **Source Representation** | `STATIC CONST(Gpt_LLD_ChannelConfigType, GPT_CONST) Gpt_InitChannelPC_1[GPT_CHANNEL_NUM_1]= { { (Gpt_ChannelType)(EMIOS_0_CH_0), /* GPT physical channel no. */ (Gpt_ChannelType)(GPT_EMIOS_MODULE), /* hardware module ID */ GPT_MODE_ONESHOT, /* Timer mode:continous/one-shot */ (uint8)0, /* not used*/ (Gpt_PrescaleType)EMIOS_GPT_LLD_CH_PRES_1,/* EMIOS Clock divider */ (boolean)TRUE, /* Freeze Enable */ (boolean)FALSE, /* Wakeup capability */ **NULL_PTR, /* Channel notification */** (Gpt_WakeUpType)(0) /* Wakeup information */ } };` |

### 3.5.2.12 GptWakeupSourceRef

**Table 3-38. GptWakeupSourceRef**

| | |
|---|---|
| **Description** | In case the wakeup-capability is true this value is transmitted to the Ecu State Manager. Implementation Type: reference to EcuM_WakeupSourceType |
| **Class** | Autosar Parameter |
| **Range** | NA |
| **Default** | NA |
| **Source File** | Gpt_Cfg.c,Gpt_PBcfg.c |
| **Source Representation** | The highlighted portion of the following structure:<br>*STATIC CONST(Gpt_LLD_ChannelConfigType, GPT_CONST) Gpt_InitChannelPB_1[GPT_CHANNEL_NUM_1]=*<br>{<br>{<br>(Gpt_ChannelType)(PITRTI_CH_0), /* GPT physical channel no. */<br>(Gpt_ChannelType)(GPT_PITRTI_MODULE), /* hardware module ID */<br>GPT_MODE_CONTINOUS, /* Timer mode:continous/one-shot */<br>(uint8)0, /* not used*/<br>(Gpt_PrescaleType) 0,/* not used */<br>(boolean)TRUE, /* Freeze Enable */<br>*(boolean)FALSE, /* Wakeup capability */*<br>GptTestCallback03, /* Channel notification */<br>***(Gpt_WakeUpType)(0) /* Wakeup information */***<br>}<br>} |

## *How to Reach Us:*


**Home Page:**
www.freescale.com

**Web Support:**
http://www.freescale.com/support

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

***For Literature Requests Only:***
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com



Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductors products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claims alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-complaint and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2010 Freescale Semiconductor, Inc.



Document Number: UM14GPTASR3.0R2.0.0
Rev. 2.0


freescale™