Document Title	Specification of GPT Driver
Document Category	Software specification (SWS)
Document UID	30
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Removed GetVersionInfo API • Aligned the API chapter structure with the updated SWS template • Changed the extended production errors into production errors
2025-11-27	R25-11	AUTOSAR Release Management	• No content changes
2024-11-27	R24-11	AUTOSAR Release Management	• Gpt notification defined in [SWS_Gpt_00292] shall be available into header Gpt_Externals.h. • Remove multicore constraint [SWS_Gpt_CONSTR_00005]. • Remove reference of [SRS_BSW_00334] from list of Non applicable requirements (Appendix A). • Naming of BSW module component is defined into CP SWS BSWGeneral instead of CP TR BSWModuleList.
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-25	R22-11	AUTOSAR Release Management	• Rename [SWS_Gpt_00381] into [SWS_Gpt_NA_00381].
2021-11-25	R21-11	AUTOSAR Release Management	• Update optional interfaces relative to EcuM.

△

2020-11-30	R20-11	AUTOSAR Release Management	• Delete requirement [SWS_Gpt_00270]. • Replace requirements defined for each error by global requirement for each error table defined in §7.4. • Move chapter Error detection in chapter 8.
2019-11-28	R19-11	AUTOSAR Release Management	• editorial changes
2018-10-31	4.4.0	AUTOSAR Release Management	• Incorporation of concept MCAL Multicore Distribution (Draft). • Header File Cleanup.
2017-12-08	4.3.1	AUTOSAR Release Management	• Ensure consistency between default error tracer and development errors. • Add support of runtime errors and change type of errors GPT_E_MODE and GPT_E_BUSY.
2016-11-30	4.3.0	AUTOSAR Release Management	• Variant chapter reworked. • Remove redundant requirement [SWS_Gpt_00342]. • Remove any reference to Dem.
2015-07-31	4.2.2	AUTOSAR Release Management	• Det renaming and extension incorporation. • Debugging support marked as obsolete. • Remove duplicated requirements in traceability.
2014-10-31	4.2.1	AUTOSAR Release Management	• Init pointer check harmonized with BSW_General, redundant [SWS_GPT_00294], [SWS_GPT_00340] items removed. • Added new error code GPT_E_INIT_FAILED.
2013-10-31	4.1.2	AUTOSAR Release Management	• editorial changes
2013-03-15	4.1.1	AUTOSAR Release Management	• GPT Predef Timer functionality added • Gpt_GetTimeElapsed and Gpt_GetTimeRemaining are fully reentrant now • MemMap.h renamed to Gpt_MemMap.h

▽

△

2011-12-22	4.0.3	AUTOSAR Release Management	• Range added to [ECUC_Gpt_00331]. • module short name replaced by module abbreviation. • Chapter 6 revised and chapter 13 added due to new traceability mechanism.
2011-04-15	4.0.2	AUTOSAR Release Management	• GPT208, GPT376 and GPT378 removed. • Multiplicity changed in [ECUC_Gpt_00312] (chapter 10.2.6 updated). • [SWS_Gpt_00256] rephrased. • [SWS_Gpt_00256] changed according to changed [SRS_BSW_00004].
2009-12-18	4.0.1	AUTOSAR Release Management	• Revised completely, a lot of SWS items deleted, replaced, changed and added. • Gpt_Cbk_CheckWakeup renamed to Gpt_CheckWakeup. • Parameter names of API services renamed. • Configuration parameters renamed, deleted and added. • Debugging Concept incorporated. • ClockReferencePoint mechanism incorporated. • Traceability tables updated. • Legal disclaimer revised. • Chapter 10.3 revised.
2008-08-13	3.1.1	AUTOSAR Release Management	• legal disclaimer revised.

▽

△

2007-12-21	3.0.1	AUTOSAR Release Management	• Introduction of consistent description of wakeup concept (as evaluated in Startup/ Wakeup Taskforce). This includes modifications and extensions of textual descriptions as well as the modification of sequence charts related to wakeup. • SWS Improvement: improvement of wording, alignment of API description. • Introduction of additional development error in case of already initialized module. • Document meta information extended. • Small layout adaptations made.
2007-01-24	2.1.15	AUTOSAR Release Management	• Header file structure changed significantly. • Return values and development errors for Gpt_GetTimeRemaining() and Gpt_GetTimeElapsed() changed. • Development error checking of ConfigPtr in Gpt_Init() changed. • Configuration container structure and configuration parameters. • Interface Dem_ReportErrorEvent() removed. • Legal disclaimer revised. • Release Notes added. • Advice for users revised. • Revision Information added.
2006-05-16	2.0	AUTOSAR Release Management	• Document structure adapted to common Release 2.0 SWS Template. • Added wake-up functionality. • For more details see chapter 11.
2005-05-31	1.0	AUTOSAR Release Management	• Initial release.

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 9
2 Acronyms and Abbreviations 10
3 Related documentation 11
3.1 Input documents & related standards and norms . . . 11
3.2 Related specification . . . 11
4 Constraints and assumptions 12
4.1 Assumptions . . . 12
4.2 Limitations . . . 12
4.3 Applicability to car domains . . . 12
5 Dependencies to other modules 13
6 Requirements Tracing 14
7 Functional specification 17
7.1 GPT Predef Timers . . . 20
7.2 Error Classification . . . 22
7.2.1 Development Errors . . . 22
7.2.2 Runtime Errors . . . 22
7.2.3 Production Errors . . . 22
8 API specification 23
8.1 Imported types . . . 23
8.2 Type definitions . . . 23
8.2.1 Gpt_ConfigType . . . 23
8.2.2 Gpt_ChannelType . . . 24
8.2.3 Gpt_ValueType . . . 24
8.2.4 Gpt_ModeType . . . 25
8.2.5 Gpt_PredefTimerType . . . 25
8.3 Function definitions . . . 25
8.3.1 Gpt_Init . . . 26
8.3.2 Gpt_DeInit . . . 28
8.3.3 Gpt_GetTimeElapsed . . . 29
8.3.4 Gpt_GetTimeRemaining . . . 31
8.3.5 Gpt_StartTimer . . . 32
8.3.6 Gpt_StopTimer . . . 34
8.3.7 Gpt_EnableNotification . . . 35
8.3.8 Gpt_DisableNotification . . . 36
8.3.9 Gpt_SetMode . . . 37
8.3.10 Gpt_DisableWakeup . . . 39
8.3.11 Gpt_EnableWakeup . . . 41
8.3.12 Gpt_CheckWakeup . . . 42

8.3.13 Gpt_GetPredefTimerValue . . . 43
8.3.14 Callback notifications . . . 44
8.4 Scheduled functions . . . 44
8.5 Expected interfaces . . . 45
8.5.1 Mandatory interfaces . . . 45
8.5.2 Optional interfaces . . . 45
8.5.3 Callout interfaces . . . 46
8.5.4 Configurable interfaces . . . 46
8.5.4.1 GPT Notification . . . 46
8.6 Error detection . . . 47
9 Sequence diagrams 48
9.1 Gpt_Init . . . 48
9.2 GPT continuous mode . . . 48
9.3 GPT one-shot mode . . . 49
9.4 Disable/Enable Notifications . . . 50
9.5 Wakeup . . . 51
10 Configuration specification 52
10.1 How to read this chapter . . . 52
10.2 Containers and configuration parameters . . . 52
10.2.1 Gpt . . . 52
10.2.2 GptDriverConfiguration . . . 53
10.2.3 GptClockReferencePoint . . . 57
10.2.4 GptChannelConfigSet . . . 58
10.2.5 GptChannelConfiguration . . . 59
10.2.6 GptWakeupConfiguration . . . 64
10.2.7 GptConfigurationOfOptApiServices . . . 64
10.3 Published Information . . . 67
A Not applicable requirements 68
B Change history of AUTOSAR traceable items 69
B.1 Change History of this document according to AUTOSAR Release R26-11 69
B.1.1 Added Specification Items in R26-11 . . . 69
B.1.2 Changed Specification Items in R26-11 . . . 69
B.1.3 Deleted Specification Items in R26-11 . . . 69
B.1.4 Added Constraints in R26-11 . . . 69
B.1.5 Changed Constraints in R26-11 . . . 69
B.1.6 Deleted Constraints in R26-11 . . . 69
B.2 Change History of this document according to AUTOSAR Release R25-11 70
B.2.1 Added Specification Items in R25-11 . . . 70
B.2.2 Changed Specification Items in R25-11 . . . 70
B.2.3 Deleted Specification Items in R25-11 . . . 70
B.2.4 Added Constraints in R25-11 . . . 70

B.2.5 Changed Constraints in R25-11 70
B.2.6 Deleted Constraints in R25-11 70
B.3 Change History of this document according to AUTOSAR Release R24-11 70
B.3.1 Added Specification Items in R24-11 70
B.3.2 Changed Specification Items in R24-11 70
B.3.3 Deleted Specification Items in R24-11 71
B.3.4 Added Constraints in R24-11 71
B.3.5 Changed Constraints in R24-11 71
B.3.6 Deleted Constraints in R24-11 71
B.4 Change History of this document according to AUTOSAR Release R23-11 71
B.4.1 Added Constraints in R23-11 71
B.4.2 Changed Constraints in R23-11 71
B.4.3 Deleted Constraints in R23-11 71

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module GTP driver.

The GPT driver is part of the microcontroller abstraction layer (MCAL). It initializes and controls the internal General Purpose Timer(s) (GPT) of the microcontroller.

The GPT driver provides services and configuration parameters for

- Starting and stopping hardware timers
- Getting timer values
- Controlling time triggered interrupt notifications, if supported by hardware
- Controlling time triggered wakeup interrupts, if supported by hardware

The tick duration of a timer channel depends on channel specific settings (part of GPT driver) as well as on system clock and settings of the clock tree controlled by the MCU module. The tick duration is not limited by this specification.

Not all hardware timers must be controlled by the GPT module. Some timers may be controlled by AUTOSAR Operating System or Complex Drivers directly. The number of timer channels controlled by the GPT driver depends on hardware, implementation and system configuration.

Beside the possibility to configure individual timer channels with individual properties, some free running up counters - so-called GPT Predef Timers - are defined. These timers have predefined tick durations and predefined number of bits (physical time units and ranges). The GPT Predef Timers are used by the Time Service module.

The GPT driver only generates time bases. Further time based functionality on driver level is covered by other MCAL modules like:

- PWM Driver (driver for pulse width modulation)
- ICU Driver (driver for input capture unit)
- OCU Driver (driver for output compare unit)

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the GPT driver module that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
BSW	Basic Software
DET	Default Error Tracer
ECU	Electronic Control Unit
GPT	General Purpose Timer
ICU	Input Capture Unit
MCU	Micro Controller Unit
NOP, nop	Null Operation
OS	Operating System

Table 2.1: Acronyms and abbreviations used in the scope of this Document

Term:	Description:
Timer channel	Represents a logical timer entity assigned to a timer hardware
Target time	Time, something shall occur, when the value is reached. The behavior depends on the configuration and the enabled functionality.
Tick	Defines the timer resolution, the duration of a timer increment
GPT Predef Timer	A GPT Predef Timer is a free running up counter provided by the GPT driver. Which GPT Predef Timer(s) are available depends on hardware (clock, hardware timers, prescaler, width of timer register, ...) and configuration. A GPT Predef Timer has predefined physical time unit and range.

Table 2.2: Terms used in the scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[3] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer

[4] Specification of MCU Driver
AUTOSAR_CP_SWS_MCUDriver

[5] Specification of ECU State Manager
AUTOSAR_CP_SWS_ECUStateManager

[6] Requirements on GPT Driver
AUTOSAR_CP_RS_GPTDriver

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2], which is also valid for GPT driver.

Thus, the specification SWS BSW General shall be considered as additional and required specification for GPT driver.

4 Constraints and assumptions

4.1 Assumptions

No assumptions.

4.2 Limitations

No limitations.

4.3 Applicability to car domains

No restrictions.

5 Dependencies to other modules

Module DET

In development mode the Error hook-function of module DET [3] will be called.

Module MCU

The GPT depends on the system clock, prescaler(s) and PLL. Thus, changes of the system clock (e.g. PLL on PLL off) also affect the clock settings of the GPT hardware. Module GPT will not take care of settings which configure the clock, prescaler(s) and PLL in its init function. This has to be done by the MCU module [4].

Hence the conversions between time and ticks shall be part of an upper layer.

Module EcuM

The GPT driver reports the wakeup interrupts to the ECU State Manager [5] for further processing.

File structure

The file structure is not defined within this specification completely. It depends on the implementation. The GPT driver shall provide at least the following files, if the conditions described are fulfilled:

[SWS_Gpt_00261]

Upstream requirements: SRS_BSW_00164

⌈Gpt_Irq.c shall include Gpt.h for the prototype declaration of the notification functions.⌋

[SWS_Gpt_00375] ⌈Gpt.c shall include Det.h in any case to be able to raise runtime error.⌋

6 Requirements Tracing

The following tables reference the requirements specified in [6, SRS documents] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_Gpt_00006] [SWS_Gpt_00280]
[SRS_BSW_00164]	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	[SWS_Gpt_00261]
[SRS_BSW_00171]	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	[SWS_Gpt_00194] [SWS_Gpt_00195] [SWS_Gpt_00196] [SWS_Gpt_00199] [SWS_Gpt_00200] [SWS_Gpt_00201] [SWS_Gpt_00202] [SWS_Gpt_00203]
[SRS_BSW_00305]	Data types naming convention	[SWS_Gpt_00357] [SWS_Gpt_00358] [SWS_Gpt_00359] [SWS_Gpt_00360]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_Gpt_00218] [SWS_Gpt_00399] [SWS_Gpt_00403]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_Gpt_00008] [SWS_Gpt_00281]
[SRS_BSW_00348]	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	[SWS_Gpt_00278]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_Gpt_00280]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_Gpt_00403]
[SRS_BSW_00375]	Basic Software Modules shall report wake-up reasons	[SWS_Gpt_00209] [SWS_Gpt_00292]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_Gpt_00280] [SWS_Gpt_00357]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[SWS_Gpt_00280] [SWS_Gpt_00357]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_Gpt_00220] [SWS_Gpt_00222] [SWS_Gpt_00223] [SWS_Gpt_00224] [SWS_Gpt_00225] [SWS_Gpt_00226] [SWS_Gpt_00227] [SWS_Gpt_00228] [SWS_Gpt_00229] [SWS_Gpt_00230] [SWS_Gpt_00325] [SWS_Gpt_00398] [SWS_Gpt_00402]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_Gpt_00280] [SWS_Gpt_00357]
[SRS_BSW_00438]	Configuration data shall be defined in a structure	[SWS_Gpt_00280] [SWS_Gpt_00357]
[SRS_BSW_00441]	Naming convention for type, macro and function	[SWS_Gpt_00360]

▽

△

Requirement	Description	Satisfied by
[SRS_Gpt_12116]	The GPT Driver shall provide the functionality to deinitialize timer channels to their power on reset state	[SWS_Gpt_00008] [SWS_Gpt_00162] [SWS_Gpt_00281] [SWS_Gpt_00308]
[SRS_Gpt_12117]	The GPT Driver shall provide a synchronous service for reading the current timer value of each timer channel	[SWS_Gpt_00010] [SWS_Gpt_00083] [SWS_Gpt_00282] [SWS_Gpt_00283]
[SRS_Gpt_12119]	The GPT driver shall provide the service for stopping each channel of the timer	[SWS_Gpt_00013] [SWS_Gpt_00285]
[SRS_Gpt_12120]	The GPT Driver shall provide a notification per channel that is called when the time period has elapsed	[SWS_Gpt_00233]
[SRS_Gpt_12121]	The GPT Driver shall provide the functionality to enable the call of a notification function per channel during the runtime	[SWS_Gpt_00014] [SWS_Gpt_00286]
[SRS_Gpt_12122]	The GPT Driver shall provide the functionality to disable the call of a notification function per channel during the runtime	[SWS_Gpt_00015] [SWS_Gpt_00287]
[SRS_Gpt_12128]	The GPT driver shall provide a service for starting a timer with specific parameters	[SWS_Gpt_00274] [SWS_Gpt_00275] [SWS_Gpt_00284]
[SRS_Gpt_12328]	The GPT driver shall use the time unit ticks for all API services which are related to GPT timer channels	[SWS_Gpt_00359]
[SRS_Gpt_13601]	The GPT Driver shall be capable of performing wakeup events, whenever a predefined wakeup period has expired	[SWS_Gpt_00127]
[SRS_Gpt_13602]	The GPT driver shall provide a service for enabling / disabling the wake-up capability of single timer channels	[SWS_Gpt_00159] [SWS_Gpt_00160] [SWS_Gpt_00289] [SWS_Gpt_00290]
[SRS_Gpt_13603]	The GPT driver shall provide a service for selecting the Wake-up mode	[SWS_Gpt_00151] [SWS_Gpt_00152] [SWS_Gpt_00153] [SWS_Gpt_00288]
[SRS_Gpt_13604]	The GPT driver shall support special free running up counters, so-called GPT Predef Timers	[SWS_Gpt_00382]
[SRS_Gpt_13605]	Different types of GPT Predef Timers shall be supported by the GPT driver	[SWS_Gpt_00383] [SWS_Gpt_00389]
[SRS_Gpt_13606]	The GPT driver shall make it possible to configure statically which GPT Predef Timers are enabled	[SWS_Gpt_00385]
[SRS_Gpt_13607]	The GPT Predef Timers shall be started/stopped automatically by the GPT driver	[SWS_Gpt_00390] [SWS_Gpt_00391] [SWS_Gpt_00392] [SWS_Gpt_00393]
[SRS_Gpt_13608]	The GPT driver shall provide a synchronous service for reading the current timer value of each GPT Predef Timer	[SWS_Gpt_00394] [SWS_Gpt_00395] [SWS_Gpt_00397]
[SRS_SPAL_00157]	All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers	[SWS_Gpt_00014] [SWS_Gpt_00015] [SWS_Gpt_00406]

▽

△

Requirement	Description	Satisfied by
[SRS_SPAL_12057]	All driver modules shall implement an interface for initialization	[SWS_Gpt_00006] [SWS_Gpt_00280]
[SRS_SPAL_12063]	All driver modules shall only support raw value mode	[SWS_Gpt_00359]
[SRS_SPAL_12067]	All driver modules shall set their wake-up conditions depending on the selected operation mode	[SWS_Gpt_00014] [SWS_Gpt_00015] [SWS_Gpt_00233]
[SRS_SPAL_12069]	All drivers of the SPAL that wake up from a wake-up interrupt shall report the wake-up reason	[SWS_Gpt_00209] [SWS_Gpt_00292]
[SRS_SPAL_12125]	All driver modules shall only initialize the configured resources	[SWS_Gpt_00068]
[SRS_SPAL_12129]	The ISRs shall be responsible for resetting the interrupt flags and calling the according notification function	[SWS_Gpt_00206] [SWS_Gpt_00327]
[SRS_SPAL_12163]	All driver modules shall implement an interface for de-initialization	[SWS_Gpt_00008] [SWS_Gpt_00281]
[SRS_SPAL_12169]	All driver modules that provide different operation modes shall provide a service for mode selection	[SWS_Gpt_00151] [SWS_Gpt_00288]
[SRS_SPAL_12263]	The implementation of all driver modules shall allow the configuration of specific module parameter types at link time	[SWS_Gpt_00357]
[SRS_SPAL_12448]	All driver modules shall have a specific behavior after a development error detection	[SWS_Gpt_00332]
[SRS_SPAL_12461]	Specific rules regarding initialization of controller registers shall apply to all driver implementations	[SWS_Gpt_00352] [SWS_Gpt_00353] [SWS_Gpt_00354] [SWS_Gpt_00355] [SWS_Gpt_00356]

Table 6.1: Requirements Tracing

7 Functional specification

The GPT driver provides services for starting and stopping timer channels (logical timer instances assigned to a timer hardware), individual for each channel by calling of:

- `Gpt_StartTimer`
- `Gpt_StopTimer`

The "target time" is passed as a parameter to `Gpt_StartTimer`. So, for each start of a timer channel, the target time can be set individually. The states and the state transitions of a timer channel are shown in 7.1

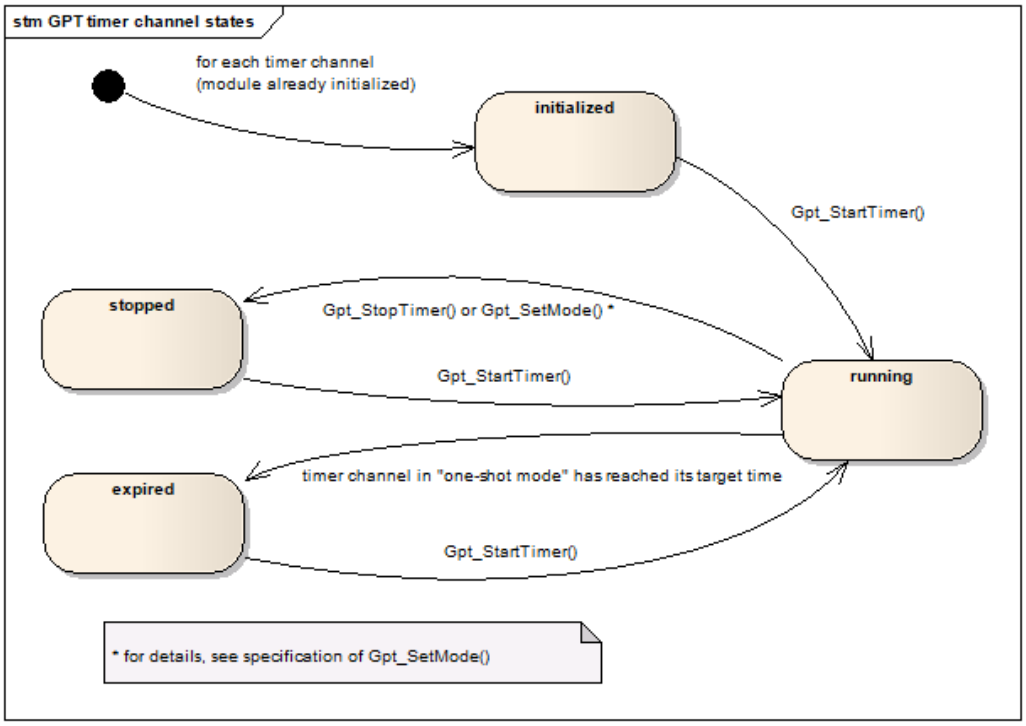


Figure 7.1: Channel states and state transitions

A timer channel can be configured in "one-shot mode" or in "continuous mode".

[SWS_Gpt_00329] ⌈A timer channel starts counting at value zero.⌋

[SWS_Gpt_00185] ⌈If a timer channel is configured in "one-shot mode":

If the timer has reached the target time (timer value = target time), the timer shall stop automatically and maintain its timer value unchanged. The channel state shall change from "running" to "expired".⌋

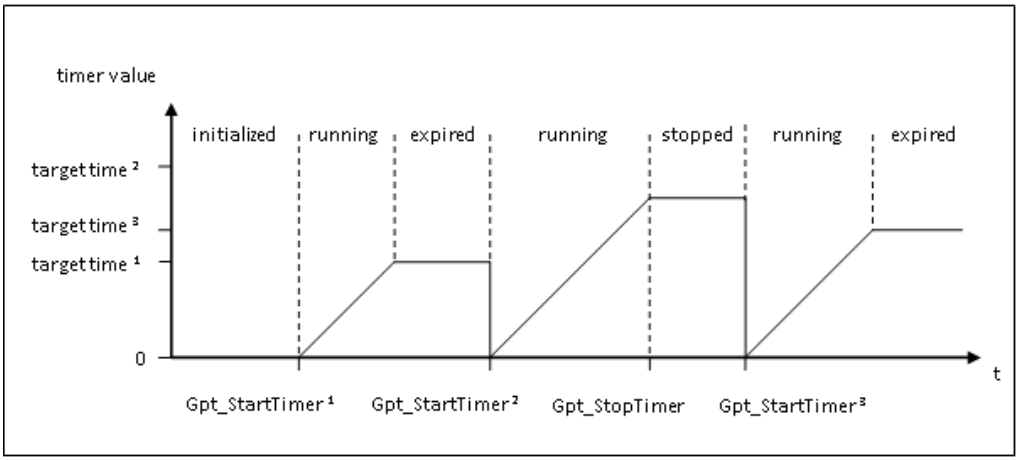


Figure 7.2: Timer channel in "one-shot mode"

[SWS_Gpt_00186] ⌈If a timer channel is configured in "continuous mode":

If the timer has reached the target time (timer value = target time), the timer shall continue running with the value "0" at next timer tick. So, the time interval of the recurrence is: target time + 1. This interval shall be independently of implementation, e.g. interrupt delays.⌋

[SWS_Gpt_00330] ⌈If a timer channel is configured in "continuous mode":

If supported by hardware, it shall be possible to realize a free running timer. This means: A timer which rolls over automatically by hardware, if the target time is set to the maximum value the timer is able to count (max value = 2n -1, n=number of bits).⌋

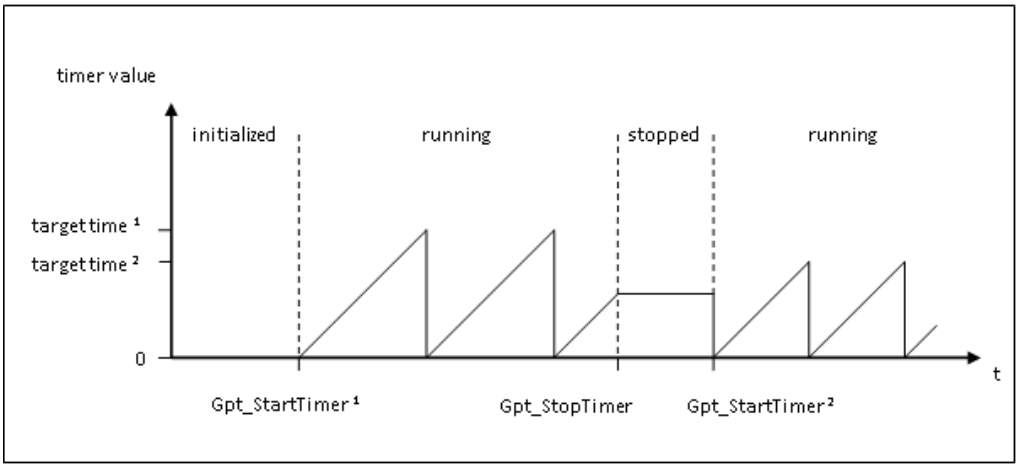


Figure 7.3: Timer channel in "continuous mode"

Both, the relative time elapsed and the time remaining can be queried by calling:

- Gpt_GetTimeElapsed
- Gpt_GetTimeRemaining

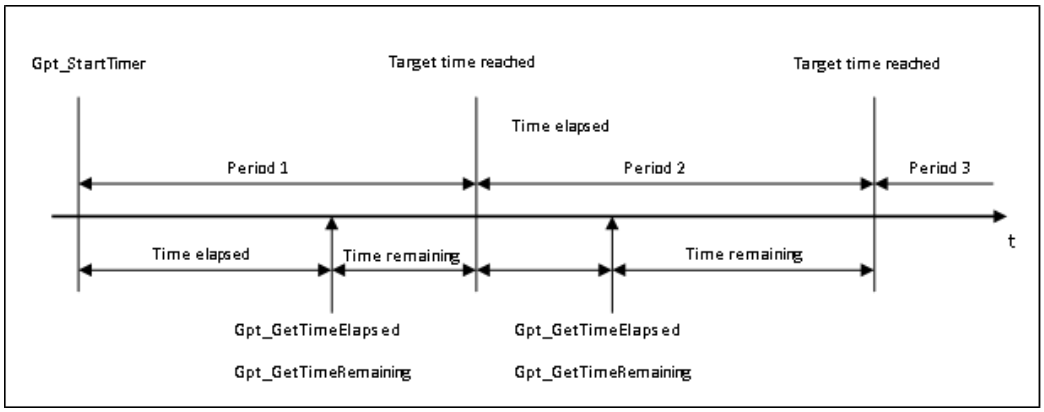


Figure 7.4: of time elapsed / time remaining for a timer channel in "continuous mode"

[SWS_Gpt_00331] ⌈If supported by hardware, a timer channel shall be able to be configured to call a notification function. If enabled, the function is called when the target time is reached (timer value = target time).⌋

Interrupt notifications can be enabled and disabled at runtime individually for each channel by calling of:

- `Gpt_EnableNotification`
- `Gpt_DisableNotification`

[SWS_Gpt_00127]

Upstream requirements: SRS_Gpt_13601

⌈If supported by hardware, a timer channel shall be able to be configured as wakeup source of the ECU. If enabled, the wakeup occurs when the target time is reached (timer value = target time).⌋

Wakeup interrupts can be enabled and disabled at runtime individually for each channel by calling of:

- `Gpt_EnableWakeup`
- `Gpt_DisableWakeup`

After initialization the GPT driver is in "normal mode". A wakeup interrupt can only occur when the driver is switched to "sleep mode". The operation mode can be set by calling of:

- `Gpt_SetMode`

For a detailled description on wakeup handling please refer to the ECU State Manager specification [5]. The operation modes and the possible mode transitions of the GPT driver are shown in 7.5.

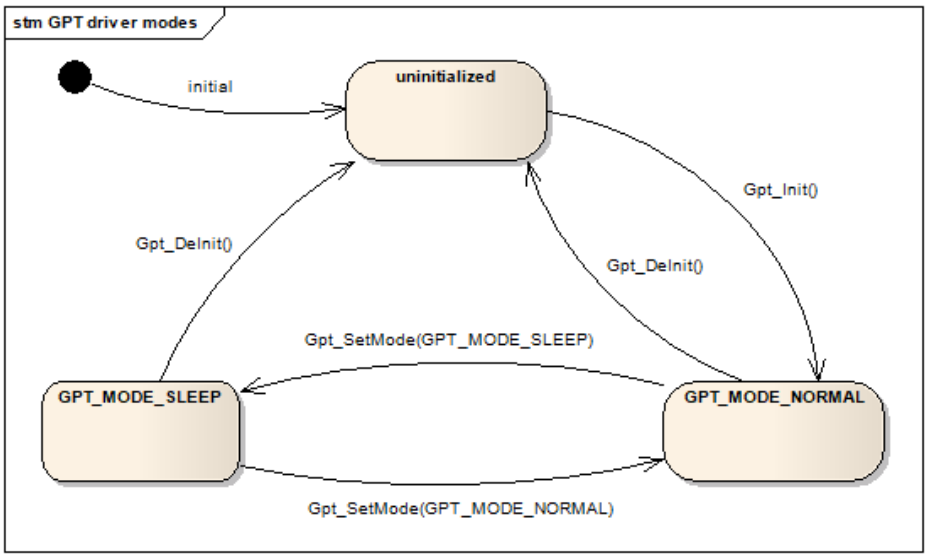


Figure 7.5: GPT driver modes

7.1 GPT Predef Timers

Beside the possibility to configure individual timer channels with individual properties, some GPT Predef Timers are defined. The API specified for "GPT timer channels" can not be used for GPT Predef Timers.

[SWS_Gpt_00382] Types of GPT Predef Timers

Upstream requirements: SRS_Gpt_13604

⌈A GPT Predef Timer is a free running up counter (user point of view). If the timer has reached the maximum value (max value = 2n -1, n=number of bits), the timer shall continue running with the value "0" at next timer tick.⌋

[SWS_Gpt_00383]

Upstream requirements: SRS_Gpt_13605

⌈

Name of GPT Predef Timer	*Tick duration*	*Maximum tick value*	*Number of bits*	*Maximum time span (circa values)*
GPT_PREDEF_TIMER_1US_16BIT	1 µs	65535	16 bit	65 ms
GPT_PREDEF_TIMER_1US_24BIT		16777215	24 bit	16 s
GPT_PREDEF_TIMER_1US_32BIT		4294967295	32 bit	71 minutes
GPT_PREDEF_TIMER_100US_32BIT	100 µs	4294967295	32 bit	4.9 days

⌋

[SWS_Gpt_00384] ⌈A GPT Predef Timer shall have a maximum tick tolerance of +/- 1 tick to ensure accuracy of time based functionality.⌋

Which GPT Predef Timer(s) can be enabled depends on clock and available timer hardware (prescaler, width of timer register). It is recommended to enable all GPT Predef Timers to ensure compatibility of time based functionality for all platforms.

It is recommended to use one hardware timer per tick duration and to supply the hardware timer directly with the clock source "fclock = 1 / (tick duration)" by good choice of clock and prescaler(s). By this, the values of the timer counter register can be used directly without any need of adaptation (computation) for performance reasons. A lower bit timer can be derived from a higher bit timer by a simple software mask operation.

For implementation of GPT Predef Timers, special hardware features may be used:

- Timers may be cascaded asynchronously to use a timer as a prescaler
- Timers may be cascaded synchronously to extend the timer range (number of bits)
- Timers with bit number greater than 32 bit may be used
- Assembler code may be used to perform 64 bit arithmetic, if necessary GPT internal, e.g. if a 48 bit timer with tick duration 250 ns or 1 µs is used for all GPT Predef Timers

[SWS_Gpt_00385]

Upstream requirements: SRS_Gpt_13606

⌈It shall be possible to configure which GPT Predef Timers are enabled.⌋

[SWS_Gpt_00386] ⌈If a GPT Predef Timer is enabled, the timer(s) with the same tick duration and lower bit number(s) shall be enabled also.⌋

Implementation specific configuration parameters are allowed if needed, e.g. to select the used hardware unit. All enabled GPT Predef Timers run after calling of:

- `Gpt_Init` ([SWS_Gpt_00390])
- `Gpt_SetMode`(GPT_MODE_NORMAL) ([SWS_Gpt_00392])

All enabled GPT Predef Timers are stopped by calling of:

- `Gpt_DeInit` ([SWS_Gpt_00391])
- `Gpt_SetMode`(GPT_MODE_SLEEP) ([SWS_Gpt_00393])

The current time value of the GPT Predef Timers can be got by calling of:

- `Gpt_GetPredefTimerValue` ([SWS_Gpt_00394])

7.2 Error Classification

Chapter [2, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.2.1 Development Errors

[SWS_Gpt_91000] Definition of development errors in module Gpt ⌈

Type of error	*Related error code*	*Error value*
API service called without module initialization	GPT_E_UNINIT	0x0A
API service for initialization called when already initialized	GPT_E_ALREADY_INITIALIZED	0x0D
API error return code: Init function failed	GPT_E_INIT_FAILED	0x0E
API parameter checking: invalid channel	GPT_E_PARAM_CHANNEL	0x14
API parameter checking: invalid value	GPT_E_PARAM_VALUE	0x15
API parameter checking: invalid pointer	GPT_E_PARAM_POINTER	0x16
API parameter checking: invalid Predef Timer	GPT_E_PARAM_PREDEF_TIMER	0x17
API parameter checking: invalid mode	GPT_E_PARAM_MODE	0x1F

⌋

7.2.2 Runtime Errors

[SWS_Gpt_91001] Definition of runtime errors in module Gpt ⌈

Type of error	*Related error code*	*Error value*
API service called when timer channel is still busy (running)	GPT_E_BUSY	0x0B
API service called when driver is in wrong mode	GPT_E_MODE	0x0C

⌋

7.2.3 Production Errors

There are no production errors.

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed.

[SWS_Gpt_00278] Definition of imported datatypes of module Gpt

Upstream requirements: SRS_BSW_00348

⌈

Module	*Header File*	*Imported Type*
EcuM	EcuM.h	EcuM_WakeupSourceType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

8.2.1 Gpt_ConfigType

[SWS_Gpt_00357] Definition of datatype Gpt_ConfigType

Upstream requirements: SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00438, SRS_BSW_00305, SRS_BSW_00414, SRS_SPAL_12263

⌈

Name	Gpt_ConfigType	
Kind	Structure	
Elements	--	
	Type	–
	Comment	Implementation specific configuration data structure, see chapter 10 for configurable parameters.
Description	This is the type of the data structure including the configuration set required for initializing the GPT timer unit.	
Available via	Gpt.h	

⌋

8.2.2 Gpt_ChannelType

[SWS_Gpt_00358] Definition of datatype Gpt_ChannelType

Upstream requirements: SRS_BSW_00305

⌈

Name	Gpt_ChannelType		
Kind	Type		
Derived from	uint		
Range	--	–	Implementation specific. But not all values may be valid within this type. This type shall be chosen in order to have the most efficient implementation on a specific micro controller platform.
Description	Numeric ID of a GPT channel.		
Available via	Gpt.h		

⌋

8.2.3 Gpt_ValueType

[SWS_Gpt_00359] Definition of datatype Gpt_ValueType

Upstream requirements: SRS_BSW_00305, SRS_SPAL_12063, SRS_Gpt_12328

⌈

Name	Gpt_ValueType		
Kind	Type		
Derived from	uint		
Range	--	–	The range of this type is µC dependent (width of the timer register) and has to be described by the supplier.
Description	Type for reading and setting the timer values (in number of ticks).		
Available via	Gpt.h		

⌋

8.2.4 Gpt_ModeType

[SWS_Gpt_00360] Definition of datatype Gpt_ModeType

Upstream requirements: SRS_BSW_00441, SRS_BSW_00305

⌈

Name	Gpt_ModeType		
Kind	Enumeration		
Range	GPT_MODE_NORMAL	0x00	Normal operation mode of the GPT
	GPT_MODE_SLEEP	0x01	Operation for reduced power operation mode. In sleep mode only wakeup capable channels are available.
Description	Modes of the GPT driver.		
Available via	Gpt.h		

⌋

8.2.5 Gpt_PredefTimerType

[SWS_Gpt_00389] Definition of datatype Gpt_PredefTimerType

Upstream requirements: SRS_Gpt_13605

⌈

Name	Gpt_PredefTimerType		
Kind	Enumeration		
Range	GPT_PREDEF_TIMER_1US_16BIT	0x00	GPT Predef Timer with tick duration 1µs and range 16bit
	GPT_PREDEF_TIMER_1US_24BIT	0x01	GPT Predef Timer with tick duration 1µs and range 24bit
	GPT_PREDEF_TIMER_1US_32BIT	0x02	GPT Predef Timer with tick duration 1µs and range 32bit
	GPT_PREDEF_TIMER_100US_32BIT	0x03	GPT Predef Timer with tick duration 100µs and range 32bit
Description	Type for GPT Predef Timers		
Available via	Gpt.h		

⌋

8.3 Function definitions

This is a list of functions provided for upper layer modules.

8.3.1 Gpt_Init

[SWS_Gpt_00280] Definition of API function Gpt_Init

Upstream requirements: SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00438, SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00414, SRS_SPAL_12057

⌈

Service Name	Gpt_Init	
Syntax	`void Gpt_Init ( const Gpt_ConfigType* ConfigPtr )`	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to a selected configuration structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initializes the GPT driver.	
Available via	Gpt.h	

⌋

[SWS_Gpt_00006]

Upstream requirements: SRS_BSW_00101, SRS_SPAL_12057

⌈The function `Gpt_Init` shall initialize the hardware timer module according to a configuration set referenced by `ConfigPtr`.⌋

[SWS_Gpt_00107] ⌈The function `Gpt_Init` shall disable all interrupt notifications, controlled by the GPT driver.⌋

[SWS_Gpt_00068]

Upstream requirements: SRS_SPAL_12125

⌈The function `Gpt_Init` 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 this driver implementation:

- **[SWS_Gpt_00352]**

  Upstream requirements: SRS_SPAL_12461

  ⌈If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register.⌋

- **[SWS_Gpt_00353]**

  Upstream requirements: SRS_SPAL_12461

  ⌈If the register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver.⌋

- **[SWS_Gpt_00354]**

  Upstream requirements: SRS_SPAL_12461

  ⌈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.⌋

- **[SWS_Gpt_00355]**

  Upstream requirements: SRS_SPAL_12461

  ⌈One-time writable registers that require initialization directly after reset shall be initialized by the startup code.⌋

- **[SWS_Gpt_00356]**

  Upstream requirements: SRS_SPAL_12461

  ⌈All other registers shall be initialized by the startup code.⌋

[SWS_Gpt_00307] ⌈If development error detection is enabled for the GPT module:

If the GPT driver is not in operation mode "uninitialized", the function `Gpt_Init` shall raise the error `GPT_E_ALREADY_INITIALIZED`.⌋

[SWS_Gpt_00258] ⌈The function `Gpt_Init` shall disable all wakeup interrupts, controlled by the GPT driver.⌋

[SWS_Gpt_00339] ⌈The function `Gpt_Init` shall set the operation mode of the GPT driver to "normal mode". This leads to a behavior like `Gpt_SetMode` is called with parameter GPT_MODE_NORMAL.⌋

[SWS_Gpt_00309] ⌈A re-initialization of the GPT driver by executing the `Gpt_Init` function requires a de-initialization before by executing a `Gpt_DeInit`.⌋

[SWS_Gpt_00390]

Upstream requirements: SRS_Gpt_13607

⌈The function `Gpt_Init` shall start all enabled GPT Predef Timers at value "0".⌋

8.3.2 Gpt_DeInit

[SWS_Gpt_00281] Definition of API function Gpt_DeInit

Upstream requirements: SRS_BSW_00336, SRS_SPAL_12163, SRS_Gpt_12116

⌈

Service Name	Gpt_DeInit
Syntax	`void Gpt_DeInit ( void )`
Service ID [hex]	0x02
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Deinitializes the GPT driver.
Available via	Gpt.h

⌋

[SWS_Gpt_00008]

Upstream requirements: SRS_BSW_00336, SRS_SPAL_12163, SRS_Gpt_12116

⌈The function `Gpt_DeInit` shall deinitialize the hardware used by the GPT driver (depending on configuration to the power on reset state. Values of registers which are not writeable are excluded. It's the responsibility of the hardware design that the state does not lead to undefined activities in the µC.⌋

[SWS_Gpt_00105] ⌈The function `Gpt_DeInit` shall disable all interrupt notifications and wakeup interrupts, controlled by the GPT driver.⌋

[SWS_Gpt_00162]

Upstream requirements: SRS_Gpt_12116

⌈The function `Gpt_DeInit` shall influence only the peripherals, which are allocated by the static configuration.⌋

[SWS_Gpt_00308]

Upstream requirements: SRS_Gpt_12116

⌈If a postbuild multiple selectable configuration variant was used, the function `Gpt_DeInit` shall further influence only the peripherals, which are allocated by the runtime configuration set passed by the previous call of the function `Gpt_Init`.⌋

[SWS_Gpt_00194]

Upstream requirements: SRS_BSW_00171

⌈The function `Gpt_DeInit` shall be pre compile time configurable On/Off by the configuration parameter: `GptDeinitApi`.⌋

[SWS_Gpt_00363] ⌈The function Gpt_DeInit shall set the operation mode of the GPT driver to "uninitialized".⌋

[SWS_Gpt_00234] ⌈If any timer channel is in state "running", the function Gpt_DeInit shall raise the runtime error GPT_E_BUSY.⌋

[SWS_Gpt_00220]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled for the GPT module:

If the driver is not initialized, the function Gpt_DeInit shall raise the error GPT_E_UNINIT.⌋

[SWS_Gpt_00391]

Upstream requirements: SRS_Gpt_13607

⌈The function Gpt_DeInit shall stop all enabled GPT Predef Timers.⌋

8.3.3 Gpt_GetTimeElapsed

[SWS_Gpt_00282] Definition of API function Gpt_GetTimeElapsed

Upstream requirements: SRS_Gpt_12117

⌈

Service Name	Gpt_GetTimeElapsed	
Syntax	`Gpt_ValueType Gpt_GetTimeElapsed ( Gpt_ChannelType Channel )`	
Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Gpt_ValueType	Elapsed timer value (in number of ticks)
Description	Returns the time already elapsed.	
Available via	Gpt.h	

⌋

[SWS_Gpt_00010]

Upstream requirements: SRS_Gpt_12117

⌈The function Gpt_GetTimeElapsed shall return the time already elapsed. When the Channel is in mode "one-shot mode", this is the value relative to the point in time, the Channel has been started.⌋

[SWS_Gpt_00361] ⌈When the Channel is in mode "continuous mode", the return value of Gpt_GetTimeElapsed is the value relative to the last recurrence (target time reached) or to the start of the Channel before the first recurrence occurs.⌋

[SWS_Gpt_00295] ⌈If the function Gpt_GetTimeElapsed is called on a timer Channel in state "initialized" (Channel started never before), the function shall return the value "0".⌋

[SWS_Gpt_00297] ⌈If the function Gpt_GetTimeElapsed is called on a timer Channel in state "stopped", the function shall return the time value at the moment of stopping.⌋

[SWS_Gpt_00299] ⌈If the function Gpt_GetTimeElapsed is called on a Channel configured for "one-shot mode" in state "expired" (timer has reached the target time), the function shall return the target time.⌋

[SWS_Gpt_00113] ⌈The function Gpt_GetTimeElapsed shall be fully reentrant, this means even for the same timer channel.⌋

[SWS_Gpt_00195]

Upstream requirements: SRS_BSW_00171

⌈The function Gpt_GetTimeElapsed shall be pre compile time configurable On/Off by the configuration parameter: GptTimeElapsedApi.⌋

[SWS_Gpt_00222]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled for GPT module:

If the driver is not initialized, the function Gpt_GetTimeElapsed shall raise the error GPT_E_UNINIT.⌋

[SWS_Gpt_00210] ⌈If development error detection is enabled for GPT module:

If the parameter Channel is invalid (not within the range specified by configuration), the function Gpt_GetTimeElapsed shall raise the error GPT_E_PARAM_CHANNEL.⌋

State / Circumstance	Timer channel state	Return value	Development error (if enabled)
Driver uninitialized	-	0	GPT_E_UNINIT
Driver initialized	initialized	0	-
	running	elapsed time	-
	stopped	elapsed time at moment of stopping	-
	expired (only one-shot mode)	target time	-
Invalid parameter "Channel"	all	0	GPT_E_PARAM_CHANNEL

Table 8.1: Return values and DET errors of Gpt_GetTimeElapsed

8.3.4 Gpt_GetTimeRemaining

[SWS_Gpt_00283] Definition of API function Gpt_GetTimeRemaining

Upstream requirements: SRS_Gpt_12117

⌈

Service Name	Gpt_GetTimeRemaining	
Syntax	`Gpt_ValueType Gpt_GetTimeRemaining ( Gpt_ChannelType Channel )`	
Service ID [hex]	0x04	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Gpt_ValueType	Remaining timer value (in number of ticks)
Description	Returns the time remaining until the target time is reached.	
Available via	Gpt.h	

⌋

[SWS_Gpt_00083]

Upstream requirements: SRS_Gpt_12117

⌈The function `Gpt_GetTimeRemaining` shall return the timer value remaining until the target time will be reached next time. The remaining time is the "target time" minus the time already elapsed.⌋

[SWS_Gpt_00301] ⌈If the function `Gpt_GetTimeRemaining` is called on a timer `Channel` in state "initialized" (`Channel` started never before), the function shall return the value "0".⌋

[SWS_Gpt_00303] ⌈If the function `Gpt_GetTimeRemaining` is called on a timer `Channel` in state "stopped", the function shall return the remaining time value at the moment of stopping.⌋

[SWS_Gpt_00305] ⌈If the function `Gpt_GetTimeRemaining` is called on a `Channel` configured for "one-shot mode" in state "expired" (timer has reached the target time), the function shall return the value "0".⌋

[SWS_Gpt_00114] ⌈The function `Gpt_GetTimeRemaining` shall be fully reentrant, this means even for the same timer `Channel`.⌋

[SWS_Gpt_00196]

Upstream requirements: SRS_BSW_00171

⌈The function `Gpt_GetTimeRemaining` shall be pre compile time configurable On/Off by the configuration parameter: `GptTimeRemainingApi`.⌋

[SWS_Gpt_00223]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled for GPT module:

If the driver is not initialized, the function `Gpt_GetTimeRemaining` shall raise the error `GPT_E_UNINIT`.⌋

[SWS_Gpt_00211] ⌈If development error detection is enabled for GPT module:

If the parameter `Channel` is invalid (not within the range specified by configuration), the function `Gpt_GetTimeRemaining` shall raise the error `GPT_E_PARAM_CHANNEL`.⌋

State / Circumstance	Timer channel state	Return value	Development error (if enabled)
Driver uninitialized	-	0	`GPT_E_UNINIT`
Driver initialized	initialized	0	-
	running	remaining time	-
	stopped	remaining time at moment of stopping	-
	expired (only one-shot mode)	0	-
Invalid parameter "Channel"	all	0	`GPT_E_PARAM_CHANNEL`

Table 8.2: Return values and DET errors of Gpt_GetTimeRemaining

8.3.5 Gpt_StartTimer

[SWS_Gpt_00284] Definition of API function Gpt_StartTimer

Upstream requirements: SRS_Gpt_12128

⌈

Service Name	Gpt_StartTimer	
Syntax	`void Gpt_StartTimer ( Gpt_ChannelType Channel, Gpt_ValueType Value )`	
Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Reentrant (but not for the same timer channel)	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
	Value	Target time in number of ticks.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Starts a timer channel.	
Available via	Gpt.h	

⌋

[SWS_Gpt_00274]

Upstream requirements: SRS_Gpt_12128

⌈The function Gpt_StartTimer shall start the selected timer Channel with a defined target time.⌋

[SWS_Gpt_00275]

Upstream requirements: SRS_Gpt_12128

⌈If configured and enabled, an interrupt notification or a wakeup interrupt occurs, when the target time is reached.⌋

[SWS_Gpt_00115] ⌈The function Gpt_StartTimer shall be reentrant, if the timer Channels used in concurrent calls are different.⌋

[SWS_Gpt_00364] ⌈The state of the selected timer Channel shall be changed to "running" if Gpt_StartTimer is called.⌋

[SWS_Gpt_00212] ⌈If development error detection is enabled for GPT module:

If the parameter Channel is invalid (not within the range specified by configuration), the function Gpt_StartTimer shall raise the error GPT_E_PARAM_CHANNEL.⌋

[SWS_Gpt_00218]

Upstream requirements: SRS_BSW_00323

⌈If development error detection is enabled for GPT module:

The function Gpt_StartTimer shall raise the error GPT_E_PARAM_VALUE if the parameter Value is "0" or not within the allowed range (exceeding the maximum timer resolution).⌋

[SWS_Gpt_00224]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled for GPT module:

If the driver is not initialized, the function Gpt_StartTimer shall raise the error GPT_E_UNINIT.⌋

[SWS_Gpt_00084] ⌈If the function Gpt_StartTimer is called on a Channel in state "running", the function shall raise the runtime error GPT_E_BUSY.⌋

8.3.6 Gpt_StopTimer

[SWS_Gpt_00285] Definition of API function Gpt_StopTimer

Upstream requirements: SRS_Gpt_12119

⌈

Service Name	Gpt_StopTimer	
Syntax	`void Gpt_StopTimer ( Gpt_ChannelType Channel )`	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Reentrant (but not for the same timer channel)	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Stops a timer channel.	
Available via	Gpt.h	

⌋

[SWS_Gpt_00013]

Upstream requirements: SRS_Gpt_12119

⌈The function `Gpt_StopTimer` shall stop the selected timer `Channel`.⌋

[SWS_Gpt_00343] ⌈The state of the selected timer `Channel` shall be changed to "stopped" if `Gpt_StopTimer` is called.⌋

[SWS_Gpt_00099] ⌈If development error detection is enabled for GPT module:

If the function `Gpt_StopTimer` is called on a `Channel` in state "initialized", "stopped" or "expired", the function shall not raise a development error.⌋

[SWS_Gpt_00344] ⌈If the function `Gpt_StopTimer` is called on a `Channel` in state "initialized", "stopped" or "expired", the function shall leave without any action (no change of the `Channel` state).⌋

[SWS_Gpt_00116] ⌈The function `Gpt_StopTimer` shall be reentrant, if the timer `Channel`s used in concurrent calls are different.⌋

[SWS_Gpt_00213] ⌈If development error detection is enabled for GPT module:

If the parameter `Channel` is invalid (not within the range specified by configuration), the function `Gpt_StopTimer` shall raise the error `GPT_E_PARAM_CHANNEL`.⌋

[SWS_Gpt_00225]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled for GPT module:

If the driver is not initialized, the function Gpt_StopTimer shall raise the error GPT_E_UNINIT.⌋

8.3.7 Gpt_EnableNotification

[SWS_Gpt_00286] Definition of API function Gpt_EnableNotification

Upstream requirements: SRS_Gpt_12121

⌈

Service Name	Gpt_EnableNotification	
Syntax	`void Gpt_EnableNotification ( Gpt_ChannelType Channel )`	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Reentrant (but not for the same timer channel)	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Enables the interrupt notification for a channel (relevant in normal mode).	
Available via	Gpt.h	

⌋

[SWS_Gpt_00014]

Upstream requirements: SRS_SPAL_00157, SRS_SPAL_12067, SRS_Gpt_12121

⌈The function Gpt_EnableNotification shall enable the interrupt notification of the referenced Channel configured for notification (see also [SWS_Gpt_00233]). The function shall save an attribute like "notification enabled" of the Channel.⌋

Comment: This attribute affects the interrupt notification always when the driver is in "normal mode". In "sleep mode" the attribute has no influence.

[SWS_Gpt_00117] ⌈The function Gpt_EnableNotification shall be reentrant, if the timer Channels used in concurrent calls are different.⌋

[SWS_Gpt_00199]

Upstream requirements: SRS_BSW_00171

⌈The function Gpt_EnableNotification shall be pre compile time configurable On/Off by the configuration parameter: GptEnableDisableNotificationApi.⌋

[SWS_Gpt_00226]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled for GPT module:

If the driver is not initialized, the function Gpt_EnableNotification shall raise the error GPT_E_UNINIT.⌋

[SWS_Gpt_00214] ⌈If development error detection is enabled for GPT module:

If the parameter Channel is invalid (not within the range specified by configuration), the function Gpt_EnableNotification shall raise the error GPT_E_PARAM_CHANNEL.⌋

[SWS_Gpt_00377] ⌈If development error detection is enabled for GPT module:

If no valid notification function is configured (GptNotification), the function Gpt_EnableNotification shall raise the error GPT_E_PARAM_CHANNEL.⌋

8.3.8 Gpt_DisableNotification

[SWS_Gpt_00287] Definition of API function Gpt_DisableNotification

Upstream requirements: SRS_Gpt_12122

⌈

Service Name	Gpt_DisableNotification	
Syntax	`void Gpt_DisableNotification ( Gpt_ChannelType Channel )`	
Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	Reentrant (but not for the same timer channel)	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Disables the interrupt notification for a channel (relevant in normal mode).	
Available via	Gpt.h	

⌋

[SWS_Gpt_00015]

Upstream requirements: SRS_SPAL_00157, SRS_Gpt_12122, SRS_SPAL_12067

⌈The function Gpt_DisableNotification shall disable the interrupt notification of the referenced Channel configured for notification (see also [SWS_Gpt_00233]). The function shall save an attribute like "notification disabled" of the Channel.⌋

Comment: This attribute affects the interrupt notification always when the driver is in "normal mode". In "sleep mode" the attribute has no influence.

[SWS_Gpt_00118] ⌈The function Gpt_DisableNotification shall be reentrant, if the timer Channels used in concurrent calls are different.⌋

[SWS_Gpt_00200]

Upstream requirements: SRS_BSW_00171

⌈The function `Gpt_DisableNotification` shall be pre compile time configurable On/Off by the configuration parameter: GptEnableDisableNotificationApi.⌋

[SWS_Gpt_00227]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled for GPT module:

If the driver is not initialized, the function `Gpt_DisableNotification` shall raise the error `GPT_E_UNINIT`.⌋

[SWS_Gpt_00217] ⌈If development error detection is enabled for GPT module:

If the parameter `Channel` is invalid (not within the range specified by configuration), the function `Gpt_DisableNotification` shall raise the error `GPT_E_PARAM_CHANNEL`.⌋

[SWS_Gpt_00379] ⌈If development error detection is enabled for GPT module:

If no valid notification function is configured (GptNotification), the function `Gpt_DisableNotification` shall raise the error `GPT_E_PARAM_CHANNEL`.⌋

8.3.9 Gpt_SetMode

[SWS_Gpt_00288] Definition of API function Gpt_SetMode

Upstream requirements: SRS_SPAL_12169, SRS_Gpt_13603

⌈

Service Name	Gpt_SetMode	
Syntax	`void Gpt_SetMode ( Gpt_ModeType Mode )`	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	Mode	GPT_MODE_NORMAL: Normal operation mode of the GPT driver. GPT_MODE_SLEEP: Sleep mode of the GPT driver (wakeup capable). See also Gpt_ModeType.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Sets the operation mode of the GPT.	

▽

△

Available via	Gpt.h

⌋

[SWS_Gpt_00151]

Upstream requirements: SRS_SPAL_12169, SRS_Gpt_13603

⌈The function Gpt_SetMode shall set the operation mode of the GPT driver to the given Mode parameter.⌋

[SWS_Gpt_00255] ⌈The function Gpt_SetMode is only available if the configuration parameter GptReportWakeupSource is enabled.⌋

[SWS_Gpt_00152]

Upstream requirements: SRS_Gpt_13603

⌈If the parameter Mode has the value GPT_MODE_NORMAL:

The function Gpt_SetMode shall enable the interrupt notification for all channels which are configured for notification and the notification is enabled (stored attribute) via the function Gpt_EnableNotification prior. All other interrupt notifications shall be disabled.⌋

[SWS_Gpt_00153]

Upstream requirements: SRS_Gpt_13603

⌈If the parameter Mode has the value GPT_MODE_SLEEP:

The function Gpt_SetMode shall enable the wakeup interrupts for all channels which are configured for wakeup and the wakeup is enabled (stored attribute) via the function Gpt_EnableWakeup prior. All other wakeup interrupts shall be disabled.⌋

[SWS_Gpt_00164] ⌈If the function Gpt_SetMode is called with parameter Mode has the value GPT_MODE_SLEEP: All timer channels in state "running" which are not configured for wakeup or not enabled for wakeup interruption (stored attribute) via Gpt_EnableWakeup shall be stopped and their state shall be changed to "stopped".⌋

[SWS_Gpt_00165] ⌈If the parameter Mode has the value GPT_MODE_NORMAL, the function Gpt_SetMode shall not restart automatically the timer channels which have been stopped by entering the sleep Mode.⌋

[SWS_Gpt_00341] ⌈If the parameter has the value GPT_MODE_SLEEP the function Gpt_SetMode shall not start a wakeup timer automatically. First, the user shall call Gpt_StartTimer to start a wakeup timer, after this the user shall call Gpt_SetMode with parameter GPT_MODE_SLEEP.⌋

[SWS_Gpt_00228]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled for GPT module:

If the driver is not initialized, the function Gpt_SetMode shall raise the error GPT_E_UNINIT.⌋

[SWS_Gpt_00231] ⌈If development error detection is enabled for GPT module:

The function Gpt_SetMode shall raise the error GPT_E_PARAM_MODE if the parameter Mode is invalid.⌋

[SWS_Gpt_00201]

Upstream requirements: SRS_BSW_00171

⌈The function Gpt_SetMode shall be pre compile time configurable On/Off by the configuration parameter: GptWakeupFunctionalityApi.⌋

[SWS_Gpt_00392]

Upstream requirements: SRS_Gpt_13607

⌈If the parameter Mode has the value GPT_MODE_NORMAL:

If the driver is in "sleep mode", the function Gpt_SetMode shall restart all enabled GPT Predef Timers at value "0".⌋

[SWS_Gpt_00393]

Upstream requirements: SRS_Gpt_13607

⌈If the parameter Mode has the value GPT_MODE_SLEEP:

The function Gpt_SetMode shall stop all enabled GPT Predef Timers.⌋

8.3.10 Gpt_DisableWakeup

[SWS_Gpt_00289] Definition of API function Gpt_DisableWakeup

Upstream requirements: SRS_Gpt_13602

⌈

Service Name	Gpt_DisableWakeup	
Syntax	`void Gpt_DisableWakeup ( Gpt_ChannelType Channel )`	
Service ID [hex]	0x0a	
Sync/Async	Synchronous	
Reentrancy	Reentrant (but not for the same timer channel)	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Disables the wakeup interrupt of a channel (relevant in sleep mode).	

▽

△

Available via	Gpt.h

⌋

[SWS_Gpt_00159]

Upstream requirements: SRS_Gpt_13602

⌈The function `Gpt_DisableWakeup` shall disable the wakeup interrupt of the referenced `Channel` configured for wakeup. The function shall save an attribute like "wakeup disabled" of the `Channel`.⌋

Comment: This attribute affects the wakeup interrupt always when the driver is in "sleep mode". In "normal mode" the attribute has no influence.

[SWS_Gpt_00157] ⌈The function `Gpt_DisableWakeup` is only feasible, if GptReportWakeupSource is statically configured available.⌋

[SWS_Gpt_00155] ⌈The function `Gpt_DisableWakeup` shall be reentrant, if the timer channels used in concurrent calls are different.⌋

[SWS_Gpt_00202]

Upstream requirements: SRS_BSW_00171

⌈The function `Gpt_DisableWakeup` shall be pre compile time configurable On/Off by the configuration parameter: `GptWakeupFunctionalityApi`.⌋

[SWS_Gpt_00215] ⌈If development error detection is enabled for GPT module:

If the parameter `Channel` is invalid (not within the range specified by configuration) or channel wakeup is not enabled by configuration (GptEnableWakeup), the function `Gpt_DisableWakeup` shall raise the error `GPT_E_PARAM_CHANNEL`.⌋

[SWS_Gpt_00229]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled for GPT module:

If the driver is not initialized, the function `Gpt_DisableWakeup` shall raise the error `GPT_E_UNINIT`.⌋

8.3.11 Gpt_EnableWakeup

[SWS_Gpt_00290] Definition of API function Gpt_EnableWakeup

Upstream requirements: SRS_Gpt_13602

⌈

Service Name	Gpt_EnableWakeup	
Syntax	`void Gpt_EnableWakeup ( Gpt_ChannelType Channel )`	
Service ID [hex]	0x0b	
Sync/Async	Synchronous	
Reentrancy	Reentrant (but not for the same timer channel)	
Parameters (in)	Channel	Numeric identifier of the GPT channel.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Enables the wakeup interrupt of a channel (relevant in sleep mode).	
Available via	Gpt.h	

⌋

[SWS_Gpt_00160]

Upstream requirements: SRS_Gpt_13602

⌈The function Gpt_EnableWakeup shall enable the wakeup interrupt of the referenced Channel configured for wakeup. The function shall save an attribute like "wakeup enabled" of the channel.⌋

Comment: This attribute affects the wakeup interrupt always when the driver is in "sleep mode". In "normal mode" the attribute has no influence.

[SWS_Gpt_00158] ⌈The function Gpt_EnableWakeup is only feasible, if GptReportWakeupSource is statically configured available.⌋

[SWS_Gpt_00156] ⌈The function Gpt_EnableWakeup shall be reentrant, if the timer Channels used in concurrent calls are different.⌋

[SWS_Gpt_00203]

Upstream requirements: SRS_BSW_00171

⌈The function Gpt_EnableWakeup shall be pre compile time configurable On/Off by the configuration parameter: GptWakeupFunctionalityApi.⌋

[SWS_Gpt_00230]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled for GPT module:

If the driver is not initialized, the function Gpt_EnableWakeup shall raise the error GPT_E_UNINIT.⌋

[SWS_Gpt_00216] ⌈If development error detection is enabled for GPT module:

If the parameter `Channel` is invalid (not within the range specified by configuration) or channel wakeup is not enabled by configuration (GptEnableWakeup), the function `Gpt_EnableWakeup` shall raise the error `GPT_E_PARAM_CHANNEL`.⌋

8.3.12 Gpt_CheckWakeup

[SWS_Gpt_00328] Definition of API function Gpt_CheckWakeup ⌈

Service Name	Gpt_CheckWakeup	
Syntax	`void Gpt_CheckWakeup ( EcuM_WakeupSourceType WakeupSource )`	
Service ID [hex]	0x0c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	WakeupSource	Information on wakeup source to be checked. The associated GPT channel can be determined from configuration data.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	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.	
Available via	Gpt.h	

⌋

[SWS_Gpt_00321] ⌈The function `Gpt_CheckWakeup` shall check if a wakeup capable GPT channel is the source for a wakeup event and call `EcuM_SetWakeupEvent` to indicate a valid timer wakeup event to the ECU State Manager [5].⌋

[SWS_Gpt_00322] ⌈The function `Gpt_CheckWakeup` is only feasible, if GptReportWakeupSource is statically configured available.⌋

[SWS_Gpt_00323] ⌈The function `Gpt_CheckWakeup` shall be reentrant, by reason of possible usage in concurrent interrupt service routines.⌋

[SWS_Gpt_00324] ⌈The function `Gpt_CheckWakeup` shall be pre compile time configurable On/Off by the configuration parameter: `GptWakeupFunctionalityApi`.⌋

[SWS_Gpt_00325]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled for GPT module:

If the driver is not initialized, the function `Gpt_CheckWakeup` shall raise the error `GPT_E_UNINIT`.⌋

8.3.13 Gpt_GetPredefTimerValue

[SWS_Gpt_00394] Definition of API function Gpt_GetPredefTimerValue

Upstream requirements: SRS_Gpt_13608

⌈

Service Name	Gpt_GetPredefTimerValue	
Syntax	`Std_ReturnType Gpt_GetPredefTimerValue ( Gpt_PredefTimerType PredefTimer, uint32* TimeValuePtr )`	
Service ID [hex]	0x0d	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	PredefTimer	GPT Predef Timer
Parameters (inout)	None	
Parameters (out)	TimeValuePtr	Pointer to time value destination data in RAM
Return value	Std_ReturnType	`E_OK`: no error has been detected `E_NOT_OK`: aborted due to errors
Description	Delivers the current value of the desired GPT Predef Timer.	
Available via	Gpt.h	

⌋

Note: It is strongly recommended to check the return value of the function `Gpt_GetPredefTimerValue` on user software level. When `E_NOT_OK` is returned the time value - pointed by `TimeValuePtr` - may be invalid and must not be used.

[SWS_Gpt_00395]

Upstream requirements: SRS_Gpt_13608

⌈The function `Gpt_GetPredefTimerValue` shall return the current value of the GPT Predef Timer passed by `PredefTimer`.⌋

[SWS_Gpt_00396] ⌈If the timer value of the function `Gpt_GetPredefTimerValue` is less than 32 bit (16bit or 24bit timer), the upper bits shall be filled with zero.⌋

[SWS_Gpt_00397]

Upstream requirements: SRS_Gpt_13608

⌈The function `Gpt_GetPredefTimerValue` shall be fully reentrant, this means even for the same GPT Predef Timer.⌋

[SWS_Gpt_00402]

Upstream requirements: SRS_BSW_00406

⌈If the GPT driver is not initialized, in "sleep mode" or the GPT Predef Timer is not enabled, the function `Gpt_GetPredefTimerValue` shall return `E_NOT_OK`.⌋

Note: This is to inform user software if the hardware timer is not running, independent of development error detection is enabled for GPT module enabled/disabled for the GPT module. The function `Gpt_GetPredefTimerValue` is used by the Time Service

module which is part of the Services Layer. The user of the Time Service module shall have a chance to cope with missed timer support.

[SWS_Gpt_00398]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled for GPT module:

If the driver is not initialized, the function `Gpt_GetPredefTimerValue` shall raise the error `GPT_E_UNINIT`.⌋

[SWS_Gpt_00399]

Upstream requirements: SRS_BSW_00323

⌈If development error detection is enabled for GPT module:

If the parameter `PredefTimer` is invalid, the function `Gpt_GetPredefTimerValue` shall raise the development error `GPT_E_PARAM_PREDEF_TIMER`.⌋

[SWS_Gpt_00400] ⌈If development error detection is enabled for GPT module:

If the GPT Predef Timer passed by the parameter `PredefTimer` is not enabled, the function `Gpt_GetPredefTimerValue` shall raise the development error `GPT_E_PARAM_PREDEF_TIMER`.⌋

[SWS_Gpt_00401] ⌈If the driver is in "sleep mode", the function `Gpt_GetPredefTimerValue` shall raise the runtime error `GPT_E_MODE`.⌋

[SWS_Gpt_00403]

Upstream requirements: SRS_BSW_00369, SRS_BSW_00323

⌈If development error detection is enabled for GPT module:

If the parameter `TimeValuePtr` is a null pointer, the function `Gpt_GetPredefTimerValue` shall raise the error `GPT_E_PARAM_POINTER`.⌋

8.3.14 Callback notifications

Since the GPT is a driver module it doesn't provide any callback functions for lower layer modules.

8.4 Scheduled functions

None.

8.5 Expected interfaces

In this chapter all interfaces required from other modules are listed.

8.5.1 Mandatory interfaces

Note: This section defines all interfaces, which are required to fulfill the core functionality of the module.

[SWS_Gpt_91002] Definition of mandatory interfaces required by module Gpt ⌈

API Function	*Header File*	*Description*
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.

⌋

8.5.2 Optional interfaces

This section defines all interfaces, which are required to fulfill an optional functionality of the module.

[SWS_Gpt_00406] Definition of optional interfaces requested by module Gpt

Upstream requirements: SRS_SPAL_00157

⌈

API Function	*Header File*	*Description*
Det_ReportError	Det.h	Service to report development errors.
EcuM_CheckWakeup	EcuM.h	This function can be called to check the given wakeup sources. It will pass the argument to the integrator function EcuM_CheckWakeupHook. It can also be called by the ISR of a wakeup source to set up the PLL and check other wakeup sources that may be connected to the same interrupt.
EcuM_SetWakeupEvent	EcuM.h	Sets the wakeup event.

⌋

[SWS_Gpt_00326] ⌈EcuM_CheckWakeup shall be called within the Interrupt Service Routine, servicing the GPT channel wakeup event on wakeup-capable channels.⌋

[SWS_Gpt_00327]

Upstream requirements: SRS_SPAL_12129

⌈The ISR's, providing the wakeup events, shall be responsible for resetting the interrupt flags (if needed by hardware).⌋

8.5.3 Callout interfaces

No callout interfaces.

8.5.4 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

8.5.4.1 GPT Notification

[SWS_Gpt_00292] Definition of configurable interface Gpt_Notification_<channel>

Upstream requirements: SRS_BSW_00375, SRS_SPAL_12069

⌈

Service Name	Gpt_Notification_<channel>
Syntax	`void Gpt_Notification_<channel> ( void )`
Sync/Async	Synchronous
Reentrancy	GPT user implementation dependant.
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Callback routine provided by the user to notify the caller when defined target time of the channel is reached.
Available via	Gpt_Externals.h

⌋

The GPT module's environment shall declare a separate notification for each channel to avoid parameters in notification services and to improve run time efficiency.

[SWS_Gpt_00086] ⌈The callback notifications `Gpt_Notification_<channel>` shall be configurable as pointers to user defined functions within the configuration structure.⌋

[SWS_Gpt_00209]

Upstream requirements: SRS_BSW_00375, SRS_SPAL_12069

⌈Each channel shall provide its own notification if configured.⌋

[SWS_Gpt_00093] ⌈When disabled, the GPT Driver will send no notification.⌋

[SWS_Gpt_00233]

Upstream requirements: SRS_SPAL_12067, SRS_Gpt_12120

⌈The GPT Driver shall invoke a notification whenever the defined target time of the channel is reached.⌋

[SWS_Gpt_00206]

Upstream requirements: SRS_SPAL_12129

⌈The ISR's, providing the timer events, shall be responsible for resetting the interrupt flags (if needed by hardware) and calling the according notification function.⌋

[SWS_Gpt_00362] ⌈For all available channels, callback functions have to be declared by the configuration tool.⌋

8.6 Error detection

[SWS_Gpt_00332]

Upstream requirements: SRS_SPAL_12448

⌈If the GptDevErrorDetect switch is enabled:

When a development error occurs the corresponding GPT function shall skip the desired functionality (leave service without any action).⌋

9 Sequence diagrams

All functions except Gpt_Init, Gpt_DeInit and Gpt_SetMode are synchronous and re-entrant.

9.1 Gpt_Init

The ECU State Manager (EcuM) is responsible for calling the init function.

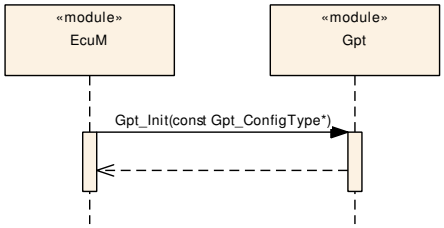


Figure 9.1: Sequence Diagram - Gpt_Init

9.2 GPT continuous mode

Channel 2 is configured as "Continuous Mode"

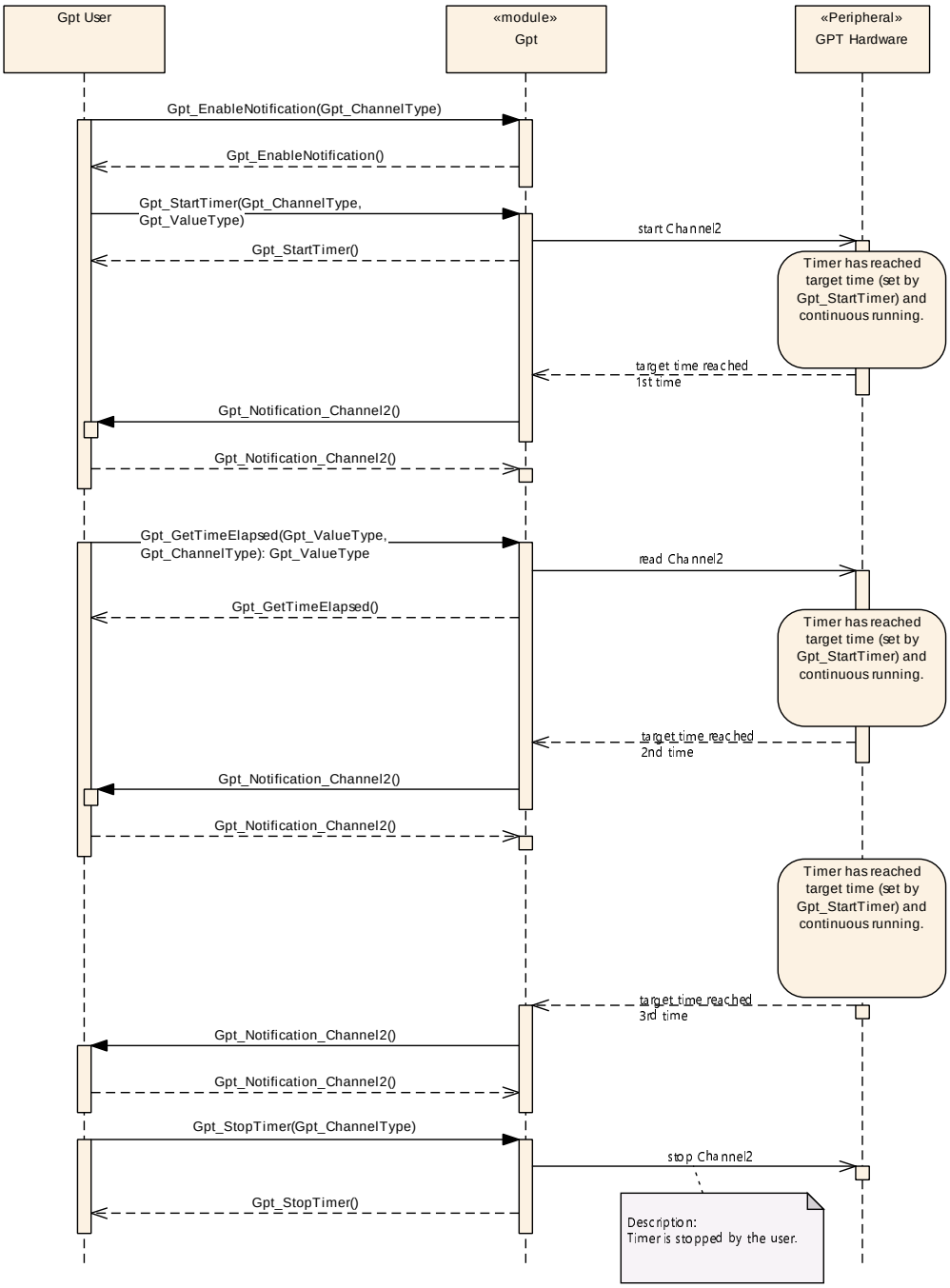


Figure 9.2: Sequence Diagram - GPT continuous mode

9.3 GPT one-shot mode

Channel 1 is configured for "One-shot Mode"

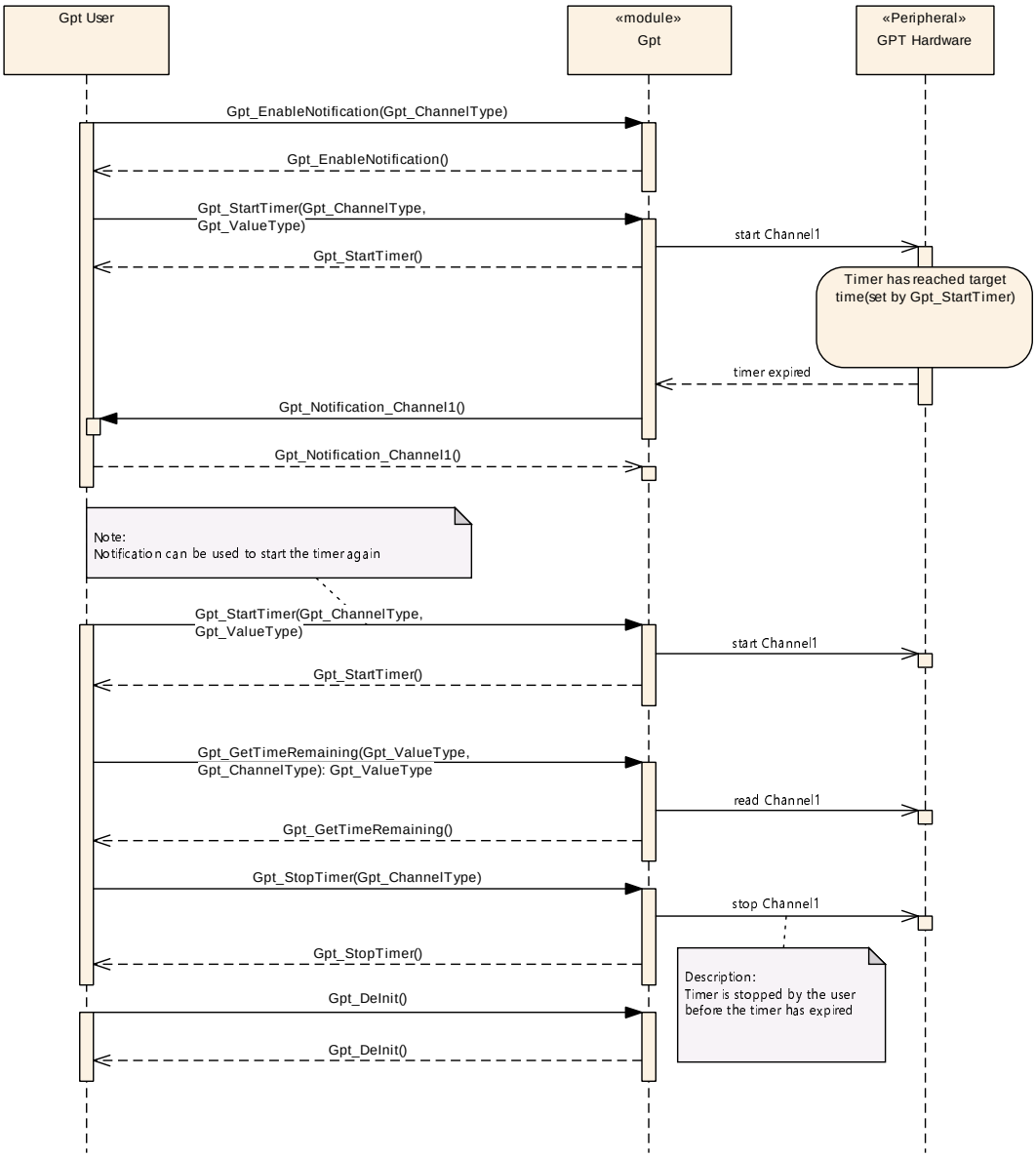


Figure 9.3: Sequence Diagram - GPT one-shot mode

9.4 Disable/Enable Notifications

The sequence diagram shown in this chapter explains the behavior of the driver, when the notification is disabled, while the timer is still running in continuous mode. If the notification is disabled, the user will not be informed, when the timer reaches the target time the 2nd time (period 2).

This notification is discarded and not made up again, when the notification is re-enabled.

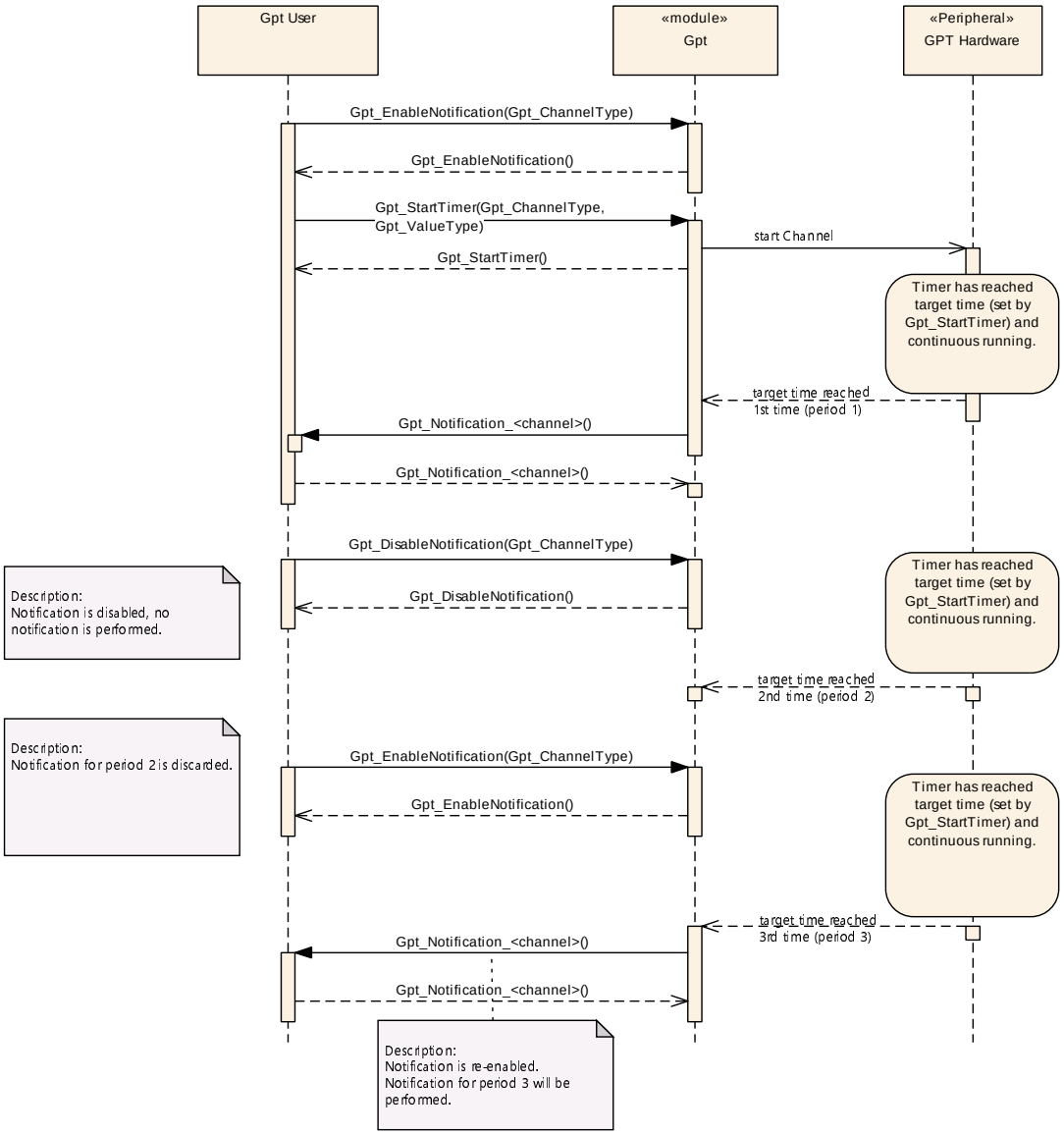


Figure 9.4: Sequence Diagram - Disable/Enable Notifications

9.5 Wakeup

Note: Sequence charts on timer wakeup can be found in the ECU state manager specification [5].

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module GPT.

Chapter 10.3 specifies published information of the module GPT.

10.1 How to read this chapter

For details refer to [2] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

[SWS_Gpt_00407] ⌈The GPT module shall reject configurations with partition mappings which are not supported by the implementation.⌋

10.2.1 Gpt

[ECUC_Gpt_00336] Definition of EcucModuleDef Gpt ⌈

Module Name	Gpt
Description	Configuration of the Gpt (General Purpose Timer) module.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
GptChannelConfigSet	1	This container is the base of a Configuration Set which contains the configured GPT channels. This way, different configuration sets can be defined for post-build process.
GptConfigurationOfOptApiServices	1	This container contains all configuration switches for configuring optional API services of the GPT driver.
GptDriverConfiguration	1	This container contains the module-wide configuration (parameters) of the GPT Driver

⌋

10.2.2 GptDriverConfiguration

[ECUC_Gpt_00183] Definition of EcucParamConfContainerDef GptDriverConfiguration ⌈

Container Name	GptDriverConfiguration
Parent Container	Gpt
Upstream mapping defined	–
Description	This container contains the module-wide configuration (parameters) of the GPT Driver
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
GptDevErrorDetect	1	[ECUC_Gpt_00321]
GptPredefTimer100us32bitEnable	1	[ECUC_Gpt_00335]
GptPredefTimer1usEnablingGrade	1	[ECUC_Gpt_00334]
GptReportWakeupSource	1	[ECUC_Gpt_00322]
GptEcucPartitionRef	0..*	[ECUC_Gpt_00337]
GptKernelEcucPartitionRef	0..1	[ECUC_Gpt_00338]

Included Containers		
Container Name	**Multiplicity**	**Description**
GptClockReferencePoint	1..*	This container contains a parameter, which represents a reference to a container of the type McuClockReferencePoint (defined in module MCU).

⌋

[ECUC_Gpt_00321] Definition of EcucBooleanParamDef GptDevErrorDetect ⌈

Parameter Name	GptDevErrorDetect
Parent Container	GptDriverConfiguration
Upstream mapping defined	–
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_Gpt_00335] Definition of EcucBooleanParamDef GptPredefTimer100us32bitEnable ⌈

Parameter Name	GptPredefTimer100us32bitEnable
Parent Container	GptDriverConfiguration
Upstream mapping defined	–
Description	Enables/disables the GPT Predef Timer 100µs32bit.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Gpt_00334] Definition of EcucEnumerationParamDef GptPredefTimer1usEnablingGrade ⌈

Parameter Name	GptPredefTimer1usEnablingGrade	
Parent Container	GptDriverConfiguration	
Upstream mapping defined	–	
Description	Specifies the grade of enabling the GPT Predef Timers with 1µs tick duration.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	GPT_PREDEF_TIMER_1US_16BIT_ENABLED	16bit timer enabled
	GPT_PREDEF_TIMER_1US_16_24BIT_ENABLED	16 and 24bit timers enabled
	GPT_PREDEF_TIMER_1US_16_24_32BIT_ENABLED	16, 24 and 32bit timers enabled
	GPT_PREDEF_TIMER_1US_DISABLED	disabled
Pre-Compile only	true	

⌋

[ECUC_Gpt_00322] Definition of EcucBooleanParamDef GptReportWakeupSource ⌈

Parameter Name	GptReportWakeupSource
Parent Container	GptDriverConfiguration
Upstream mapping defined	–
Description	Enables/Disables wakeup source reporting.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Gpt_00337] Definition of EcucReferenceDef GptEcucPartitionRef ⌈

Parameter Name	GptEcucPartitionRef	
Parent Container	GptDriverConfiguration	
Upstream mapping defined	–	
Description	Maps the GPT driver to zero or multiple ECUC partitions to make the driver API available in the according partition. Depending on the addressed timer resource the interfaces operate as follows: a) In case of partition local timer resources (n:1 mapping) the API operates as an independent instance in the according ECUC partition. b) In case of global timer resources (1:m mapping) the API operates on the global timer resource either by protected access to the resource or by implementing an according kernel.	
Multiplicity	0..*	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_Gpt_00338] Definition of EcucReferenceDef GptKernelEcucPartitionRef ⌈

Parameter Name	GptKernelEcucPartitionRef	
Parent Container	GptDriverConfiguration	
Upstream mapping defined	–	
Description	Maps the GPT kernel to zero or one ECUC partitions to assign the driver kernel to a certain core. The ECUC partition referenced is a subset of the ECUC partitions where the GPT driver is mapped to. Note: The kernel reference shall not be set in case the GPT driver is implemented without a kernel (refer to definition of GptEcucPartitionRef).	
Multiplicity	0..1	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

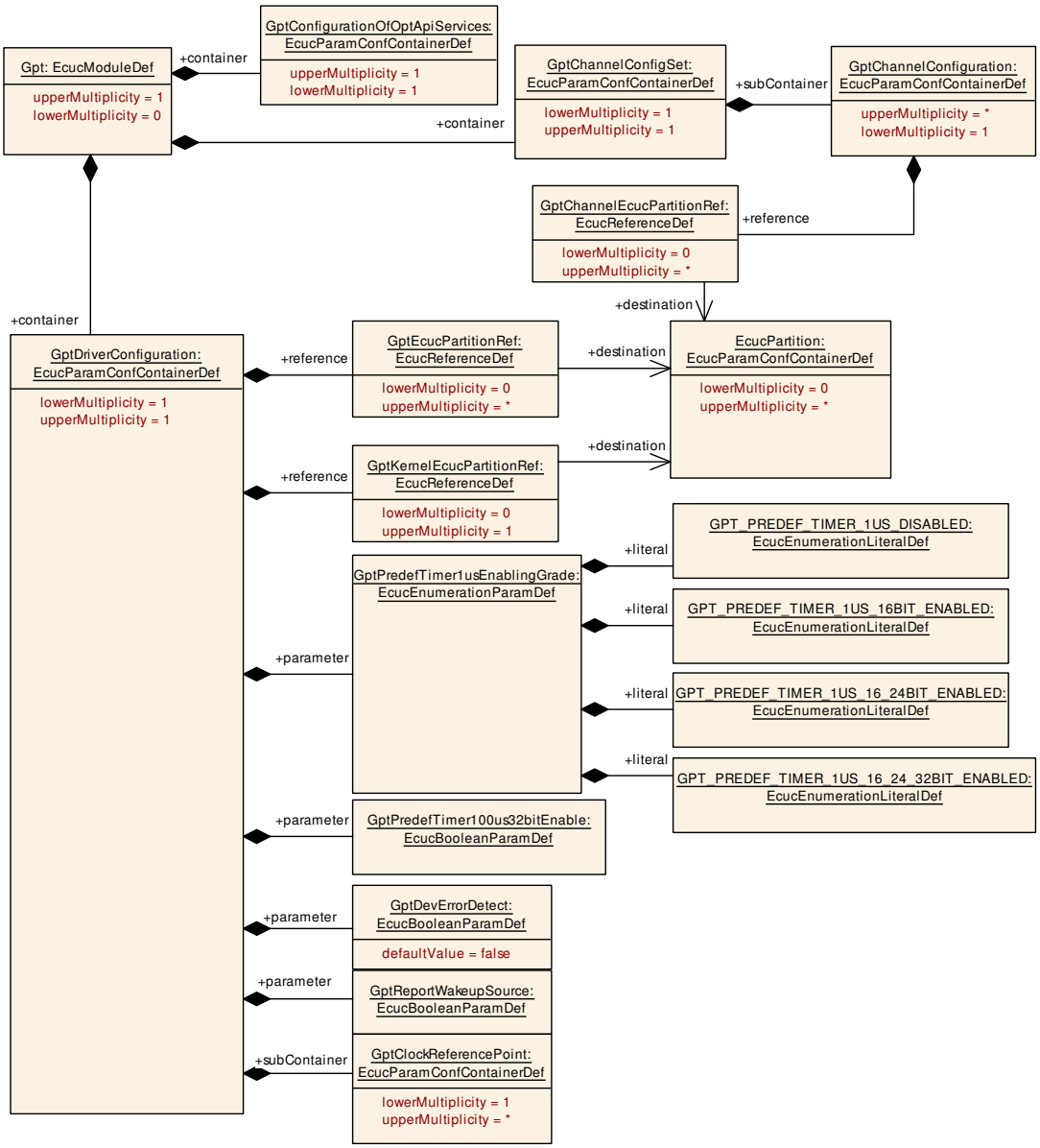


Figure 10.1: Scope of the GPT Driver configuration

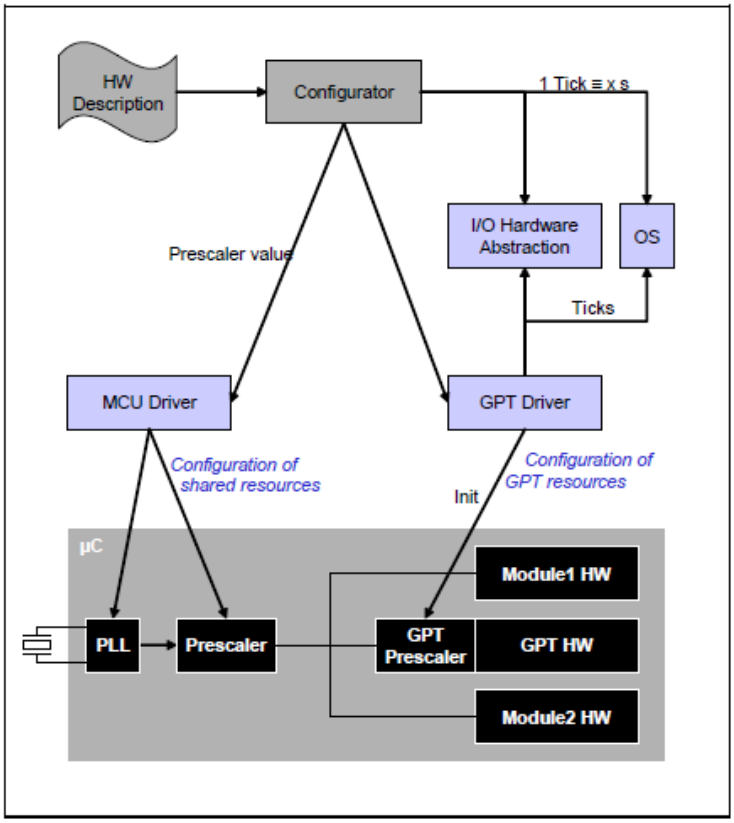


Figure 10.2: Scope of the GPT Clock Configuration

[SWS_Gpt_CONSTR_00001] ⌈The ECUC partitions referenced by GptKernelEcucPartitionRef shall be a subset of the ECUC partitions referenced by GptEcucPartitionRef.⌋

[SWS_Gpt_CONSTR_00003] ⌈If GptEcucPartitionRef references one or more ECUC partitions, GptKernelEcucPartitionRef shall have a multiplicity of one and reference one of these ECUC partitions as well.⌋

10.2.3 GptClockReferencePoint

[ECUC_Gpt_00329] Definition of EcucParamConfContainerDef GptClockReferencePoint ⌈

Container Name	GptClockReferencePoint
Parent Container	GptDriverConfiguration
Upstream mapping defined	–
Description	This container contains a parameter, which represents a reference to a container of the type McuClockReferencePoint (defined in module MCU). A container is needed to support multiple clock references (hardware dependent).

▽

△

Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
GptClockReference	1	[ECUC_Gpt_00330]

No Included Containers

⌋

[ECUC_Gpt_00330] Definition of EcucReferenceDef GptClockReference ⌈

Parameter Name	GptClockReference
Parent Container	GptClockReferencePoint
Upstream mapping defined	–
Description	Reference to a container of the type McuClockReferencePoint, to select an input clock. The configuration editor for the GPT module can support the integrator by only allowing a selection of those clock reference points that can be connected physically to the GPT hardware peripheral. The desired frequency (desired by GPT) has to be the same as the selected and provided frequency of the MCU configuration. This has to be checked automatically.
Multiplicity	1
Type	Reference to McuClockReferencePoint
Pre-Compile only	true

⌋

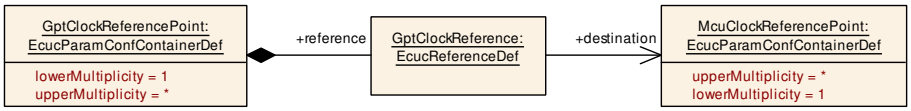


Figure 10.3: GptClockreferencePoint

10.2.4 GptChannelConfigSet

[ECUC_Gpt_00269] Definition of EcucParamConfContainerDef GptChannelConfigSet ⌈

Container Name	GptChannelConfigSet
Parent Container	Gpt
Upstream mapping defined	–
Description	This container is the base of a Configuration Set which contains the configured GPT channels. This way, different configuration sets can be defined for post-build process.
Multiplicity	1

▽

△

Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
GptChannelConfiguration	1..*	This container contains the channel specific configuration of the GPT Driver.

⌋

10.2.5 GptChannelConfiguration

[ECUC_Gpt_00184] Definition of EcucParamConfContainerDef GptChannelConfiguration ⌈

Container Name	GptChannelConfiguration
Parent Container	GptChannelConfigSet
Upstream mapping defined	–
Description	Configuration of an individual GPT channel.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
GptChannelId	1	[ECUC_Gpt_00308]
GptChannelMode	1	[ECUC_Gpt_00309]
GptChannelTickFrequency	1	[ECUC_Gpt_00331]
GptChannelTickValueMax	1	[ECUC_Gpt_00332]
GptEnableWakeup	1	[ECUC_Gpt_00311]
GptNotification	0..1	[ECUC_Gpt_00312]
GptChannelClkSrcRef	1	[ECUC_Gpt_00333]
GptChannelEcucPartitionRef	0..*	[ECUC_Gpt_00339]

Included Containers		
Container Name	**Multiplicity**	**Description**
GptWakeupConfiguration	0..1	Function pointer to callback function (for non-wakeup notification).

⌋

[ECUC_Gpt_00308] Definition of EcucIntegerParamDef GptChannelId ⌈

Parameter Name	GptChannelId	
Parent Container	GptChannelConfiguration	
Upstream mapping defined	–	
Description	Channel Id of the GPT channel. This value will be assigned to the symbolic name derived of the GptChannelConfiguration container short name.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Gpt_00309] Definition of EcucEnumerationParamDef GptChannelMode ⌈

Parameter Name	GptChannelMode	
Parent Container	GptChannelConfiguration	
Upstream mapping defined	–	
Description	Specifies the behavior of the timer channel after the target time is reached.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	GPT_CH_MODE_CONTINUOUS	After reaching the target time, the timer continues running with the value "zero" again.
	GPT_CH_MODE_ONESHOT	After reaching the target time, the timer stops automatically (timer expired).
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Gpt_00331] Definition of EcucFloatParamDef GptChannelTickFrequency ⌈

Parameter Name	GptChannelTickFrequency	
Parent Container	GptChannelConfiguration	
Upstream mapping defined	–	
Description	Specifies the tick frequency of the timer channel in Hz.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Gpt_00332] Definition of EcucIntegerParamDef GptChannelTickValueMax ⌈

Parameter Name	GptChannelTickValueMax	
Parent Container	GptChannelConfiguration	
Upstream mapping defined	–	
Description	Maximum value in ticks, the timer channel is able to count. With the next tick, the timer rolls over to zero.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 18446744073709551615	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Gpt_00311] Definition of EcucBooleanParamDef GptEnableWakeup ⌈

Parameter Name	GptEnableWakeup	
Parent Container	GptChannelConfiguration	
Upstream mapping defined	–	
Description	Enables wakeup capability of MCU for a channel.	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Gpt_00312] Definition of EcucFunctionNameDef GptNotification ⌈

Parameter Name	GptNotification	
Parent Container	GptChannelConfiguration	
Upstream mapping defined	–	
Description	Function pointer to callback function (for non-wakeup notification)	
Multiplicity	0..1	
Type	EcucFunctionNameDef	
Default value	–	
Regular Expression	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Gpt_00333] Definition of EcucReferenceDef GptChannelClkSrcRef ⌈

Parameter Name	GptChannelClkSrcRef
Parent Container	GptChannelConfiguration
Upstream mapping defined	–
Description	Reference to the GptClockReferencePoint from which the channel clock is derived.

▽

△

Multiplicity	1	
Type	Reference to GptClockReferencePoint	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_Gpt_00339] Definition of EcucReferenceDef GptChannelEcucPartitionRef ⌈

Parameter Name	GptChannelEcucPartitionRef	
Parent Container	GptChannelConfiguration	
Upstream mapping defined	–	
Description	Maps a GPT channel to zero or multiple ECUC partitions to limit the access to this channel group. The ECUC partitions referenced are a subset of the ECUC partitions where the GPT driver is mapped to.	
Multiplicity	0..*	
Type	Reference to EcucPartition	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

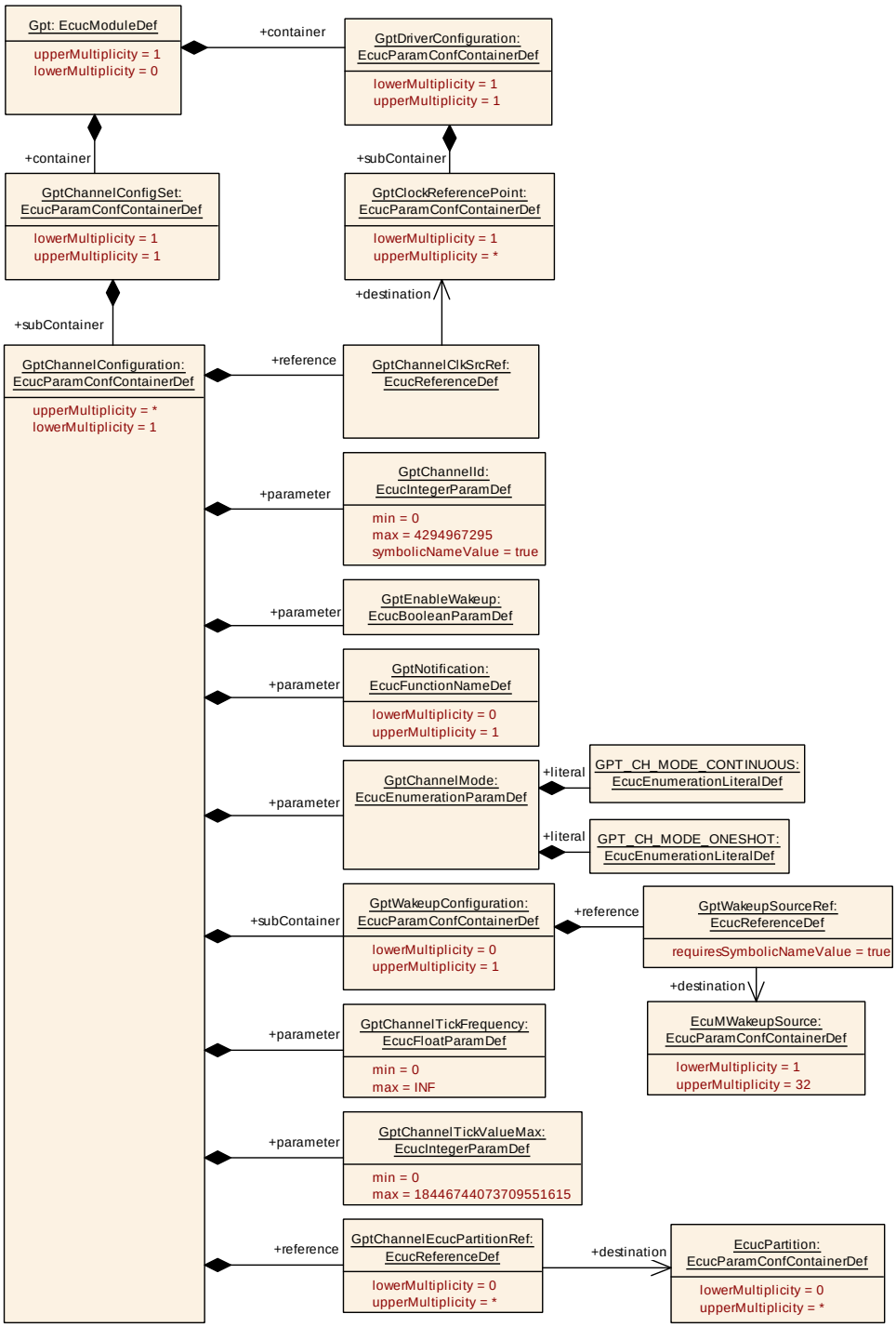


Figure 10.4: GptChannelConfiguration

[SWS_Gpt_CONSTR_00002] ⌈The ECUC partitions referenced by GptGroupEcucPartitionRef shall be a subset of the ECUC partitions referenced by GptEcucPartitionRef.⌋

[SWS_Gpt_CONSTR_00004] ⌈If GptEcucPartitionRef references one or more ECUC partitions, GptKernelEcucPartitionRef shall have a multiplicity of greater than zero and reference one or several of these ECUC partitions as well.⌋

10.2.6 GptWakeupConfiguration

[ECUC_Gpt_00235] Definition of EcucParamConfContainerDef GptWakeupConfiguration ⌈

Container Name	GptWakeupConfiguration
Parent Container	GptChannelConfiguration
Upstream mapping defined	–
Description	Function pointer to callback function (for wakeup notification).
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
GptWakeupSourceRef	1	[ECUC_Gpt_00313]

No Included Containers

⌋

[ECUC_Gpt_00313] Definition of EcucReferenceDef GptWakeupSourceRef ⌈

Parameter Name	GptWakeupSourceRef	
Parent Container	GptWakeupConfiguration	
Upstream mapping defined	–	
Description	In case the wakeup-capability is true this value is transmitted to the Ecu State Manager. Implementation Type: reference to EcuM_WakeupSourceType	
Multiplicity	1	
Type	Symbolic name reference to EcuMWakeupSource	
Post-Build Value	Loadable	Selectable

⌋

10.2.7 GptConfigurationOfOptApiServices

[ECUC_Gpt_00193] Definition of EcucParamConfContainerDef GptConfigurationOfOptApiServices ⌈

Container Name	GptConfigurationOfOptApiServices
Parent Container	Gpt
Upstream mapping defined	–
Description	This container contains all configuration switches for configuring optional API services of the GPT driver.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
GptDeinitApi	1	[ECUC_Gpt_00314]
GptEnableDisableNotificationApi	1	[ECUC_Gpt_00315]
GptTimeElapsedApi	1	[ECUC_Gpt_00317]
GptTimeRemainingApi	1	[ECUC_Gpt_00318]
GptWakeupFunctionalityApi	1	[ECUC_Gpt_00320]

No Included Containers

⌋

[ECUC_Gpt_00314] Definition of EcucBooleanParamDef GptDeinitApi ⌈

Parameter Name	GptDeinitApi
Parent Container	GptConfigurationOfOptApiServices
Upstream mapping defined	–
Description	Adds / removes the service Gpt_DeInit() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Gpt_00315] Definition of EcucBooleanParamDef GptEnableDisableNotificationApi ⌈

Parameter Name	GptEnableDisableNotificationApi
Parent Container	GptConfigurationOfOptApiServices
Upstream mapping defined	–
Description	Adds / removes the services Gpt_EnableNotification() and Gpt_DisableNotification from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Gpt_00317] Definition of EcucBooleanParamDef GptTimeElapsedApi ⌈

Parameter Name	GptTimeElapsedApi
Parent Container	GptConfigurationOfOptApiServices
Upstream mapping defined	–
Description	Adds / removes the service Gpt_GetTimeElapsed() from the code
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–

▽

△

Pre-Compile only	true

⌋

[ECUC_Gpt_00318] Definition of EcucBooleanParamDef GptTimeRemainingApi ⌈

Parameter Name	GptTimeRemainingApi
Parent Container	GptConfigurationOfOptApiServices
Upstream mapping defined	–
Description	Adds / removes the service Gpt_GetTimeRemaining() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_Gpt_00320] Definition of EcucBooleanParamDef GptWakeupFunctionalityApi ⌈

Parameter Name	GptWakeupFunctionalityApi
Parent Container	GptConfigurationOfOptApiServices
Upstream mapping defined	–
Description	Adds / removes the services Gpt_SetMode(), Gpt_EnableWakeup() Gpt_DisableWakeup() and Gpt_CheckWakeup() from the code.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

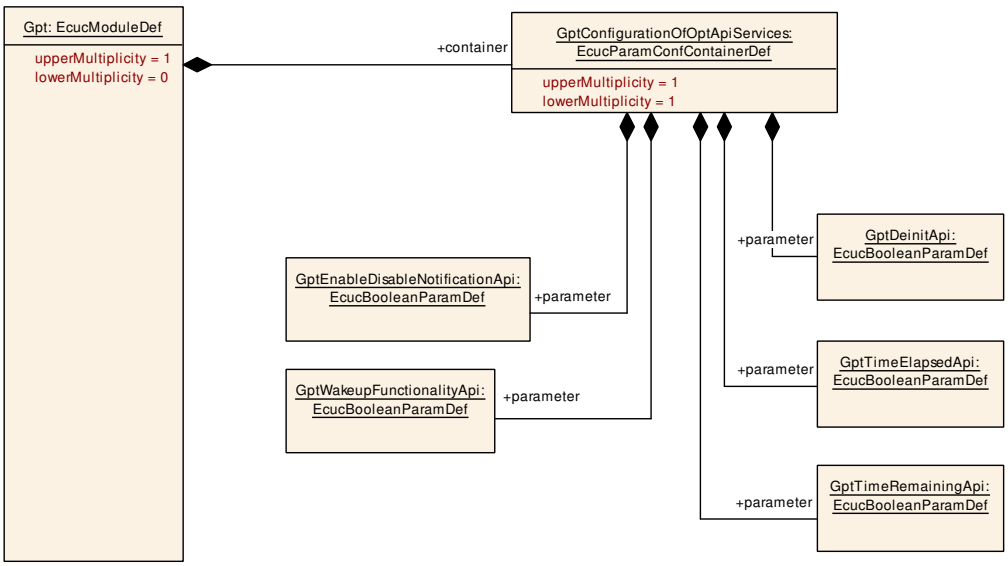


Figure 10.5: GptConfigurationOfOptApiServices

10.3 Published Information

For details refer to [2] Chapter 10.3 *"Published Information"*.

[SWS_Gpt_00380] ⌈The standardized common published parameters as required by [SRS_BSW_00402] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation is defined into General Specification of Basic Software Modules [2].⌋

Additional module-specific published parameters are listed below if applicable.

A Not applicable requirements

[SWS_Gpt_NA_00381]

Upstream requirements: SRS_BSW_00344, SRS_BSW_00159, SRS_BSW_00167, SRS_BSW_00170, SRS_BSW_00398, SRS_BSW_00416, SRS_BSW_00437, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00422, SRS_BSW_00417, SRS_SPAL_12462, SRS_SPAL_12463, SRS_SPAL_12068, SRS_SPAL_12075, SRS_SPAL_12064, SRS_SPAL_12077, SRS_SPAL_12078, SRS_SPAL_12092, SRS_SPAL_12265

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Change History of this document according to AUTOSAR Release R26-11

B.1.1 Added Specification Items in R26-11

none

B.1.2 Changed Specification Items in R26-11

[ECUC_Gpt_00193] [SWS_Gpt_00278]

B.1.3 Deleted Specification Items in R26-11

[ECUC_Gpt_00319] [SWS_Gpt_00279] [SWS_Gpt_00338]

B.1.4 Added Constraints in R26-11

none

B.1.5 Changed Constraints in R26-11

none

B.1.6 Deleted Constraints in R26-11

none

B.2 Change History of this document according to AUTOSAR Release R25-11

B.2.1 Added Specification Items in R25-11

none

B.2.2 Changed Specification Items in R25-11

none

B.2.3 Deleted Specification Items in R25-11

none

B.2.4 Added Constraints in R25-11

none

B.2.5 Changed Constraints in R25-11

none

B.2.6 Deleted Constraints in R25-11

none

B.3 Change History of this document according to AUTOSAR Release R24-11

B.3.1 Added Specification Items in R24-11

[SWS_Gpt_91002]

B.3.2 Changed Specification Items in R24-11

[SWS_Gpt_00194] [SWS_Gpt_00292] [SWS_Gpt_00380]

B.3.3 Deleted Specification Items in R24-11

[SWS_Gpt_00405]

B.3.4 Added Constraints in R24-11

none

B.3.5 Changed Constraints in R24-11

none

B.3.6 Deleted Constraints in R24-11

[SWS_Gpt_CONSTR_00005]

B.4 Change History of this document according to AUTOSAR Release R23-11

B.4.1 Added Constraints in R23-11

[SWS_Gpt_CONSTR_00001] [SWS_Gpt_CONSTR_00002] [SWS_Gpt_CONSTR_00003] [SWS_Gpt_CONSTR_00004] [SWS_Gpt_CONSTR_00005]

B.4.2 Changed Constraints in R23-11

none

B.4.3 Deleted Constraints in R23-11

none