

| | |
|-----------------------------------|-----------------------------|
| Document Title | Specification of GPT Driver |
| Document Owner | AUTOSAR |
| Document Responsibility | AUTOSAR |
| Document Identification No | 030 |

| | |
|---------------------------------|------------------|
| Document Status | Final |
| Part of AUTOSAR Standard | Classic Platform |
| Part of Standard Release | 4.3.1 |

| Document Change History | | | |
|-------------------------|---------|----------------------------|---|
| Date | Release | Changed by | Change Description |
| 2017-12-08 | 4.3.1 | AUTOSAR Release Management | <ul style="list-style-type: none"> 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 | <ul style="list-style-type: none"> Variant chapter reworked. Remove redundant requirement SWS_Gpt_00342. Remove any reference to Dem. |
| 2015-07-31 | 4.2.2 | AUTOSAR Release Management | <ul style="list-style-type: none"> Det renaming and extension incorporation Debugging support marked as obsolete Remove duplicated requirements in traceability |
| 2014-10-31 | 4.2.1 | AUTOSAR Release Management | <ul style="list-style-type: none"> 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 | <ul style="list-style-type: none"> Editorial changes |
| 2013-03-15 | 4.1.1 | AUTOSAR Administration | <ul style="list-style-type: none"> GPT Predef Timer functionality added Gpt_GetTimeElapsed and Gpt_GetTimeRemaining are fully reentrant now MemMap.h renamed to Gpt_MemMap.h |

| Document Change History | | | |
|-------------------------|---------|------------------------|--|
| Date | Release | Changed by | Change Description |
| 2011-12-22 | 4.0.3 | AUTOSAR Administration | <ul style="list-style-type: none"> 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 Administration | <ul style="list-style-type: none"> 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 Administration | <ul style="list-style-type: none"> 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 Administration | <ul style="list-style-type: none"> Legal disclaimer revised |

| Document Change History | | | |
|-------------------------|---------|------------------------|---|
| Date | Release | Changed by | Change Description |
| 2007-12-21 | 3.0.1 | AUTOSAR Administration | <ul style="list-style-type: none"> • 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 Administration | <ul style="list-style-type: none"> • 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 changed • Interface Dem_ReportErrorEvent() removed • Legal disclaimer revised • Release Notes added • “Advice for users” revised • “Revision Information” added |
| 2006-05-16 | 2.0 | AUTOSAR Administration | <ul style="list-style-type: none"> • 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 Administration | <ul style="list-style-type: none"> • 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 | 8 |
| 2 | Acronyms, abbreviations and terms | 9 |
| 3 | Related documentation | 10 |
| 3.1 | Input documents | 10 |
| 3.2 | Related standards and norms | 11 |
| 3.3 | 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 traceability | 15 |
| 7 | Functional specification | 23 |
| 7.1 | General behavior | 23 |
| 7.2 | GPT Predef Timers | 26 |
| 7.3 | Version checking | 27 |
| 7.4 | Error classification | 28 |
| 7.4.1 | Development Errors | 28 |
| 7.4.2 | Runtime Errors | 28 |
| 7.4.3 | Transient Faults | 28 |
| 7.4.4 | Production Errors | 28 |
| 7.4.5 | Extended Production Errors | 28 |
| 7.5 | Error detection | 29 |
| 8 | API specification | 30 |
| 8.1 | Imported types | 30 |
| 8.2 | Type Definitions | 30 |
| 8.3 | Error notification | 30 |
| 8.3.1 | Gpt_ConfigType | 30 |
| 8.3.2 | Gpt_ChannelType | 30 |
| 8.3.3 | Gpt_ValueType | 31 |
| 8.3.4 | Gpt_ModeType | 31 |
| 8.3.5 | Gpt_PredefTimerType | 31 |
| 8.4 | Function definitions | 31 |
| 8.4.1 | Gpt_GetVersionInfo | 31 |
| 8.4.2 | Gpt_Init | 32 |
| 8.4.3 | Gpt_DeInit | 33 |
| 8.4.4 | Gpt_GetTimeElapsed | 34 |
| 8.4.5 | Gpt_GetTimeRemaining | 36 |
| 8.4.6 | Gpt_StartTimer | 37 |
| 8.4.7 | Gpt_StopTimer | 38 |
| 8.4.8 | Gpt_EnableNotification | 39 |
| 8.4.9 | Gpt_DisableNotification | 40 |

| | | |
|--------|--|----|
| 8.4.10 | Gpt_SetMode..... | 40 |
| 8.4.11 | Gpt_DisableWakeup..... | 42 |
| 8.4.12 | Gpt_EnableWakeup..... | 43 |
| 8.4.13 | Gpt_CheckWakeup..... | 44 |
| 8.4.14 | Gpt_GetPredefTimerValue | 45 |
| 8.5 | Call-back Notifications | 46 |
| 8.6 | Scheduled functions | 46 |
| 8.7 | Expected Interfaces..... | 46 |
| 8.7.1 | Mandatory Interfaces | 46 |
| 8.7.2 | Optional Interfaces..... | 47 |
| 8.7.3 | Configurable Interfaces..... | 47 |
| 9 | Sequence diagrams | 49 |
| 9.1 | Gpt_Init..... | 49 |
| 9.2 | GPT continuous mode..... | 50 |
| 9.3 | GPT one-shot mode | 51 |
| 9.4 | Disable/Enable Notifications | 52 |
| 9.5 | Wakeup | 53 |
| 10 | Configuration specification | 54 |
| 10.1 | How to read this chapter..... | 54 |
| 10.2 | Containers and configuration parameters..... | 55 |
| 10.2.1 | Variants | 55 |
| 10.2.2 | Gpt..... | 56 |
| 10.2.3 | GptDriverConfiguration | 56 |
| 10.2.4 | GptClockReferencePoint | 58 |
| 10.2.5 | GptChannelConfigSet..... | 59 |
| 10.2.6 | GptChannelConfiguration | 59 |
| 10.2.7 | GptWakeupConfiguration | 61 |
| 10.2.8 | GptConfigurationOfOptApiServices | 62 |
| 10.3 | Published Information..... | 64 |
| 11 | Not applicable requirements..... | 65 |

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module GPT 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, abbreviations and terms

Only a few acronyms and abbreviations are listed here which are helpful to understand this document or which have a local scope. Further information can be found in the official AUTOSAR glossary [13].

| Acronym / Abbreviation | 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 1: Acronyms and abbreviations

The terms defined in the table below have a local scope within 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: Terms

3 Related documentation

3.1 Input documents

- [1] List of Basic Software Modules,
AUTOSAR_TR_BSWModuleList.pdf
- [2] Layered Software Architecture,
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf
- [3] General Requirements on Basic Software Modules,
AUTOSAR_SRS_BSWGeneral.pdf
- [4] Specification of Standard Types,
AUTOSAR_SWS_StandardTypes.pdf
- [5] Specification of Default Error Tracer,
AUTOSAR_SWS_DefaultErrorTracer.pdf
- [6] Specification of ECU Configuration,
AUTOSAR_TPS_ECUConfiguration.pdf
- [7] Specification of Diagnostic Event Manager,
AUTOSAR_SWS_DiagnosticEventManager.pdf
- [8] Specification of ECU State Manager,
AUTOSAR_SWS_ECUSTateManager.pdf
- [9] General Requirements on SPAL,
AUTOSAR_SRS_SPALGeneral.pdf
- [10] Requirements on GPT Driver,
AUTOSAR_SRS_GPTDriver.pdf
- [11] Specification of ICU Driver,
AUTOSAR_SWS_ICUDriver.pdf
- [12] Specification of MCU Driver,
AUTOSAR_SWS_MCUDriver.pdf
- [13] Glossary,
AUTOSAR_TR_Glossary.pdf
- [14] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

- [15] General Specification of Basic Software Modules,
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

- [16] IEC 7498-1 The Basic Model, IEC Norm, 1994

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [15] (SWS BSW General), 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 [5]

In development mode the Error hook-function of module DET [5] will be called.

Module MCU [12]

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

Module EcuM [8]

The GPT driver reports the wakeup interrupts to the ECU State Manager 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_00293] [`Gpt.c` shall include `Gpt.h`.

Comment: `Gpt.c` has implicit access to the `Gpt_Cfg.h` through the `Gpt.h` file.
] ()

[SWS_Gpt_00261] [`Gpt_Irq.c` shall include `Gpt.h` for the prototype declaration of the notification functions.] (SRS_BSW_00164)

[SWS_Gpt_00375] [`Gpt.c` shall include `Det.h` in any case to be able to raise runtime error.] ()

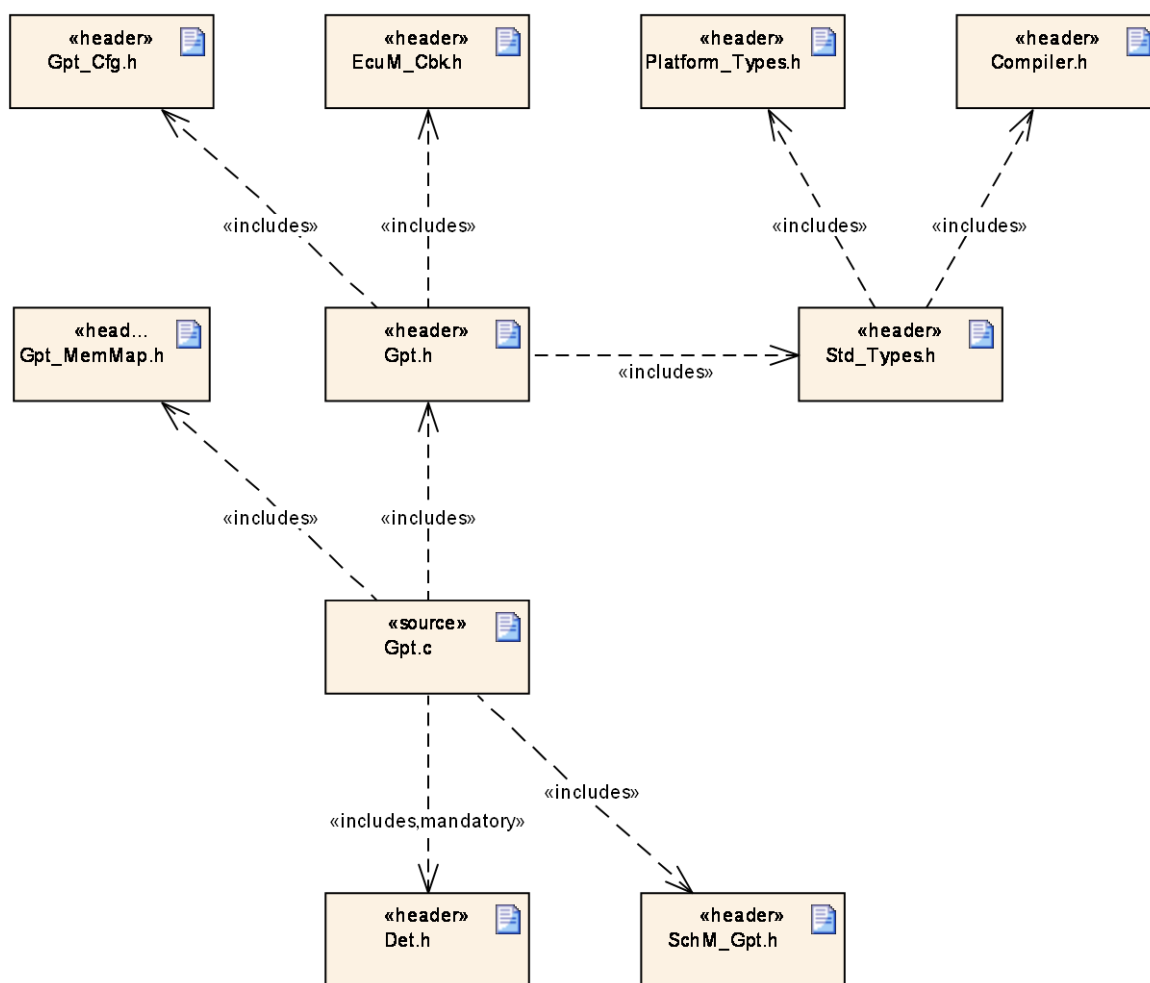


Figure 1: Header file include structure

6 Requirements traceability

This chapter refers to input requirements specified in the SRS documents (Software Requirements Specifications) that are applicable for this software module.

The table below lists links to specification items of the GPT driver SWS document, which satisfy the input requirements. Only functional requirements are referenced.

| Requirement | Description | Satisfied by |
|---------------|--|------------------------------|
| SRS_BSW_00005 | Modules of the μ C Abstraction Layer (MCAL) may not have hard coded horizontal interfaces | SWS_Gpt_00381 |
| SRS_BSW_00006 | The source code of software modules above the μ C Abstraction Layer (MCAL) shall not be processor and compiler dependent. | SWS_Gpt_00381 |
| SRS_BSW_00007 | All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard. | SWS_Gpt_00381 |
| SRS_BSW_00009 | All Basic SW Modules shall be documented according to a common standard. | SWS_Gpt_00381 |
| SRS_BSW_00010 | The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms. | SWS_Gpt_00381 |
| 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_00159 | All modules of the AUTOSAR Basic Software shall support a tool based configuration | SWS_Gpt_00381 |
| SRS_BSW_00160 | Configuration files of AUTOSAR Basic SW module shall be readable for human beings | SWS_Gpt_00381 |
| SRS_BSW_00161 | The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers | SWS_Gpt_00381 |
| SRS_BSW_00162 | The AUTOSAR Basic Software shall provide a hardware abstraction layer | SWS_Gpt_00381 |
| SRS_BSW_00164 | The Implementation of interrupt service routines shall be done by the Operating System, complex | SWS_Gpt_00261 |

| | | |
|---------------|---|--|
| | drivers or modules | |
| SRS_BSW_00167 | All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks | SWS_Gpt_00381 |
| SRS_BSW_00168 | SW components shall be tested by a function defined in a common API in the Basis-SW | SWS_Gpt_00381 |
| SRS_BSW_00170 | The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands | SWS_Gpt_00381 |
| 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_00172 | The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system | SWS_Gpt_00381 |
| SRS_BSW_00305 | Data types naming convention | SWS_Gpt_00357, SWS_Gpt_00358, SWS_Gpt_00359, SWS_Gpt_00360 |
| SRS_BSW_00306 | AUTOSAR Basic Software Modules shall be compiler and platform independent | SWS_Gpt_00381 |
| SRS_BSW_00307 | Global variables naming convention | SWS_Gpt_00381 |
| SRS_BSW_00308 | AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file | SWS_Gpt_00381 |
| SRS_BSW_00309 | All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword | SWS_Gpt_00381 |
| SRS_BSW_00321 | The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules | SWS_Gpt_00381 |
| SRS_BSW_00323 | All AUTOSAR Basic Software Modules shall check passed API parameters for validity | SWS_Gpt_00218, SWS_Gpt_00338, SWS_Gpt_00399, SWS_Gpt_00403 |
| SRS_BSW_00325 | The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short | SWS_Gpt_00381 |
| SRS_BSW_00328 | All AUTOSAR Basic Software Modules shall avoid the duplication of code | SWS_Gpt_00381 |
| SRS_BSW_00330 | It shall be allowed to use macros | SWS_Gpt_00381 |

| | | |
|---------------|--|------------------------------|
| | instead of functions where source code is used and runtime is critical | |
| SRS_BSW_00331 | All Basic Software Modules shall strictly separate error and status information | SWS_Gpt_00381 |
| SRS_BSW_00333 | For each callback function it shall be specified if it is called from interrupt context or not | SWS_Gpt_00381 |
| SRS_BSW_00334 | All Basic Software Modules shall provide an XML file that contains the meta data | SWS_Gpt_00381 |
| SRS_BSW_00335 | Status values naming convention | SWS_Gpt_00381 |
| SRS_BSW_00336 | Basic SW module shall be able to shutdown | SWS_Gpt_00008, SWS_Gpt_00281 |
| SRS_BSW_00341 | Module documentation shall contains all needed informations | SWS_Gpt_00381 |
| SRS_BSW_00342 | It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed | SWS_Gpt_00381 |
| SRS_BSW_00344 | BSW Modules shall support link-time configuration | SWS_Gpt_00381 |
| SRS_BSW_00347 | A Naming separation of different instances of BSW drivers shall be in place | SWS_Gpt_00381 |
| SRS_BSW_00348 | All AUTOSAR standard types and constants shall be placed and organized in a standard type header file | SWS_Gpt_00278, SWS_Gpt_00381 |
| SRS_BSW_00353 | All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header | SWS_Gpt_00381 |
| SRS_BSW_00357 | For success/failure of an API call a standard return type shall be defined | SWS_Gpt_00381 |
| SRS_BSW_00358 | The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void | SWS_Gpt_00280 |
| SRS_BSW_00359 | All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible | SWS_Gpt_00381 |
| SRS_BSW_00360 | AUTOSAR Basic Software Modules callback functions are allowed to have parameters | SWS_Gpt_00381 |
| SRS_BSW_00361 | All mappings of not standardized | SWS_Gpt_00381 |

| | | |
|---------------|--|---|
| | keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header | |
| SRS_BSW_00369 | All AUTOSAR Basic Software Modules shall not return specific development error codes via the API | SWS_Gpt_00403 |
| SRS_BSW_00373 | The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention | SWS_Gpt_00381 |
| SRS_BSW_00375 | Basic Software Modules shall report wake-up reasons | SWS_Gpt_00209, SWS_Gpt_00292 |
| SRS_BSW_00377 | A Basic Software Module can return a module specific types | SWS_Gpt_00381 |
| SRS_BSW_00378 | AUTOSAR shall provide a boolean type | SWS_Gpt_00381 |
| SRS_BSW_00398 | The link-time configuration is achieved on object code basis in the stage after compiling and before linking | SWS_Gpt_00381 |
| 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 | A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called | 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_00407 | Each BSW module shall provide a function to read out the version information of a dedicated module implementation | SWS_Gpt_00279 |
| SRS_BSW_00413 | An index-based accessing of the instances of BSW modules shall be done | SWS_Gpt_00381 |
| SRS_BSW_00414 | Init functions shall have a pointer to a configuration structure as single parameter | SWS_Gpt_00280, SWS_Gpt_00357 |
| SRS_BSW_00415 | Interfaces which are provided exclusively for one module shall be separated into a dedicated header file | SWS_Gpt_00381 |
| SRS_BSW_00416 | The sequence of modules to be initialized shall be configurable | SWS_Gpt_00381 |
| SRS_BSW_00417 | Software which is not part of the SW-C shall report error events only after the DEM is fully | SWS_Gpt_00381 |

| | | |
|---------------|--|--|
| | operational. | |
| SRS_BSW_00422 | Pre-de-bouncing of error status information is done within the DEM | SWS_Gpt_00381 |
| SRS_BSW_00423 | BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template | SWS_Gpt_00381 |
| SRS_BSW_00424 | BSW module main processing functions shall not be allowed to enter a wait state | SWS_Gpt_00381 |
| SRS_BSW_00425 | The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects | SWS_Gpt_00381 |
| SRS_BSW_00426 | BSW Modules shall ensure data consistency of data which is shared between BSW modules | SWS_Gpt_00381 |
| SRS_BSW_00427 | ISR functions shall be defined and documented in the BSW module description template | SWS_Gpt_00381 |
| SRS_BSW_00428 | A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence | SWS_Gpt_00381 |
| SRS_BSW_00429 | Access to OS is restricted | SWS_Gpt_00381 |
| SRS_BSW_00432 | Modules should have separate main processing functions for read/receive and write/transmit data path | SWS_Gpt_00381 |
| SRS_BSW_00433 | Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler | SWS_Gpt_00381 |
| SRS_BSW_00437 | Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup | SWS_Gpt_00381 |
| SRS_BSW_00438 | Configuration data shall be defined in a structure | SWS_Gpt_00280, SWS_Gpt_00357 |
| SRS_BSW_00439 | Enable BSW modules to handle interrupts | SWS_Gpt_00381 |
| SRS_BSW_00440 | The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via Rte_Call API | SWS_Gpt_00381 |
| SRS_BSW_00441 | Naming convention for type, macro and function | SWS_Gpt_00360 |
| SRS_Gpt_12116 | The GPT Driver shall provide the functionality to deinitialize timer channels to their power on reset | SWS_Gpt_00008, SWS_Gpt_00162, SWS_Gpt_00281, SWS_Gpt_00308 |

| | | |
|---------------|--|---|
| | state | |
| 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 | SWS_Gpt_00394, SWS_Gpt_00395, |

| | | |
|----------------|--|--|
| | synchronous service for reading the current timer value of each GPT Predef Timer | 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_00405, SWS_Gpt_00406 |
| 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_12064 | All driver modules shall raise an error if the change of the operation mode leads to degradation of running operations | SWS_Gpt_00381, SWS_Gpt_00405 |
| 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_12068 | The modules of the MCAL shall be initialized in a defined sequence | SWS_Gpt_00381 |
| 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_12075 | All drivers with random streaming capabilities shall use application buffers | SWS_Gpt_00381 |
| SRS_SPAL_12077 | All drivers shall provide a non blocking implementation | SWS_Gpt_00381 |
| SRS_SPAL_12078 | The drivers shall be coded in a way that is most efficient in terms of memory and runtime resources | SWS_Gpt_00381 |
| SRS_SPAL_12092 | The driver's API shall be accessed by its handler or manager | SWS_Gpt_00381 |
| 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 | SWS_Gpt_00151, SWS_Gpt_00288 |

| | | |
|----------------|--|---|
| | selection | |
| 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_12265 | Configuration data shall be kept constant | SWS_Gpt_00381 |
| 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 |
| SRS_SPAL_12462 | The register initialization settings shall be published | SWS_Gpt_00381 |
| SRS_SPAL_12463 | The register initialization settings shall be combined and forwarded | SWS_Gpt_00381 |

7 Functional specification

7.1 General behavior

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 Figure 2

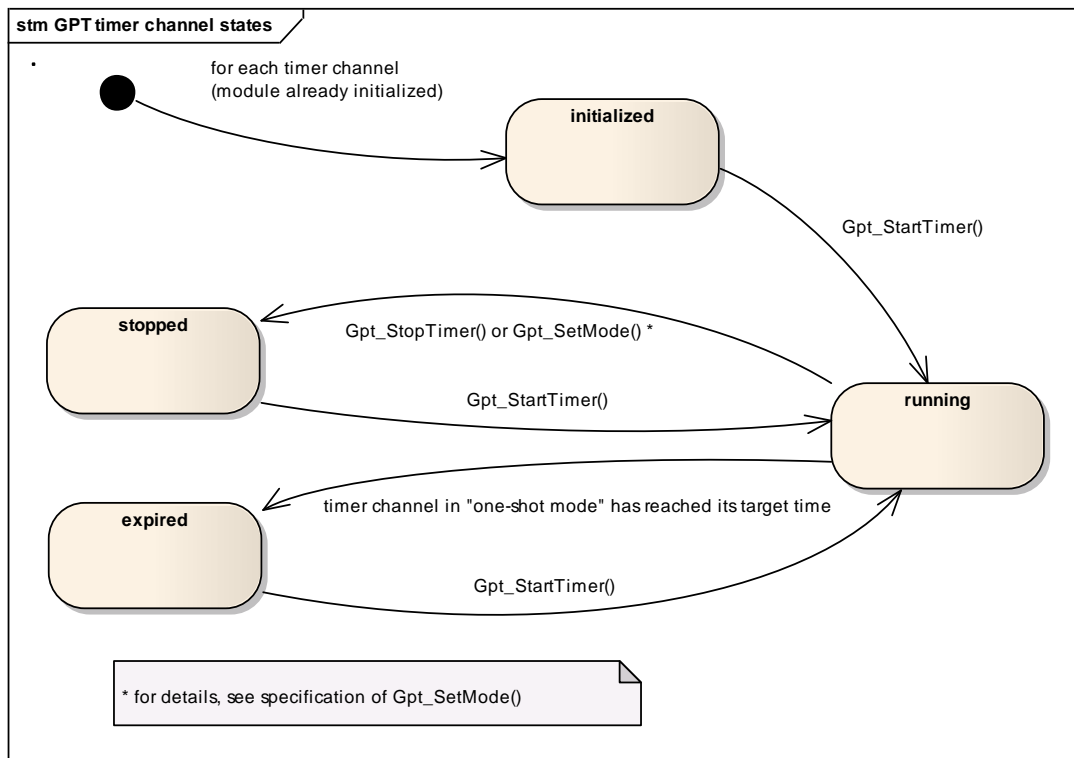


Figure 2: Channel states and state transitions

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".] ()

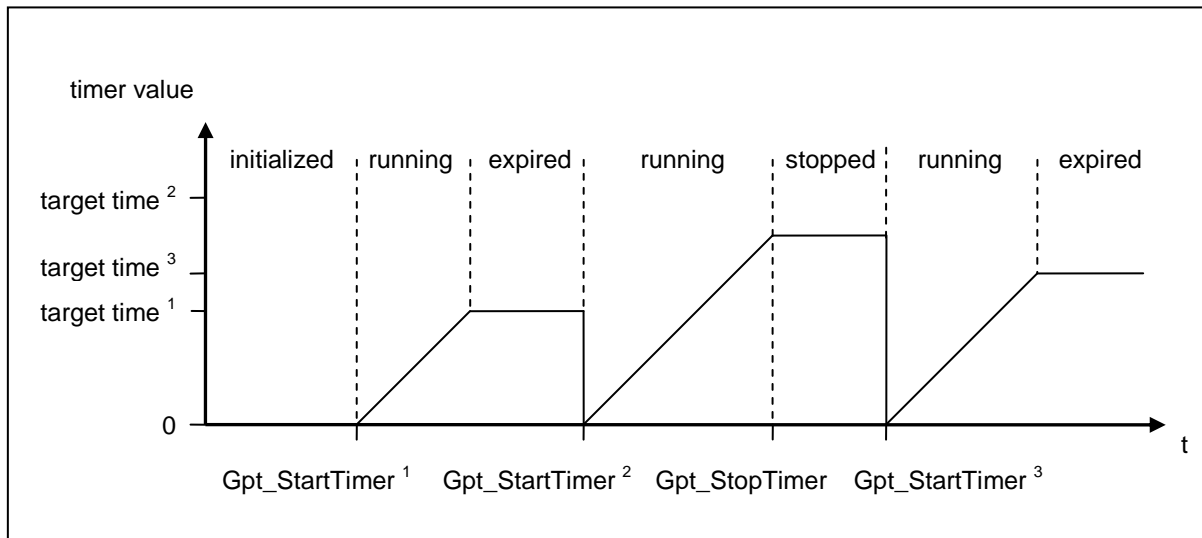


Figure 3: Timer channel in "one-shot mode"

[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 = $2^n - 1$, n =number of bits).] ()

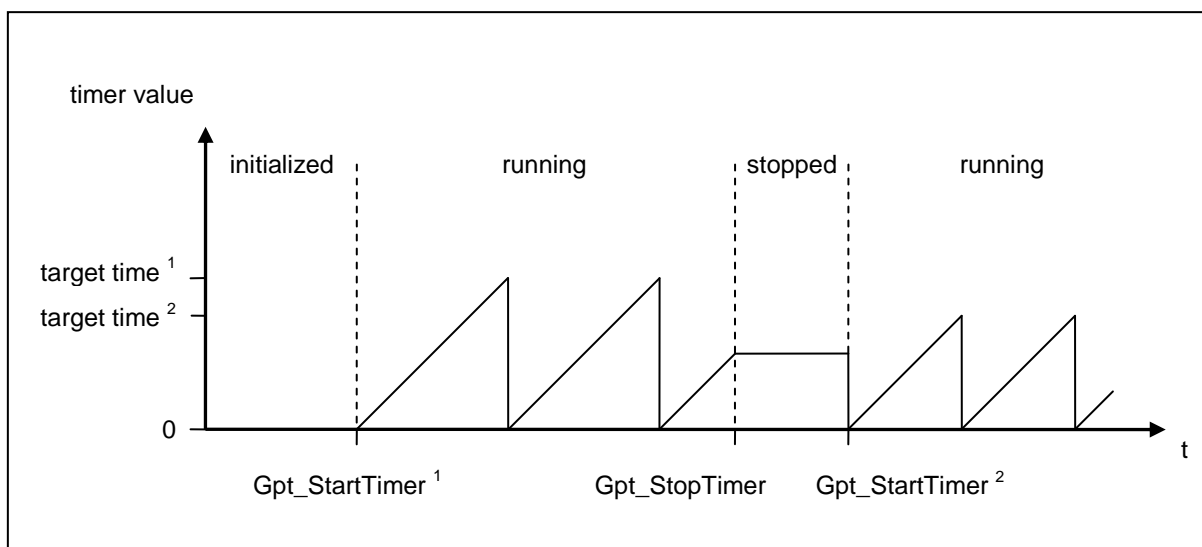


Figure 4: Timer channel in "continuous mode"

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

```
Gpt_GetTimeElapsed
Gpt_GetTimeRemaining
```

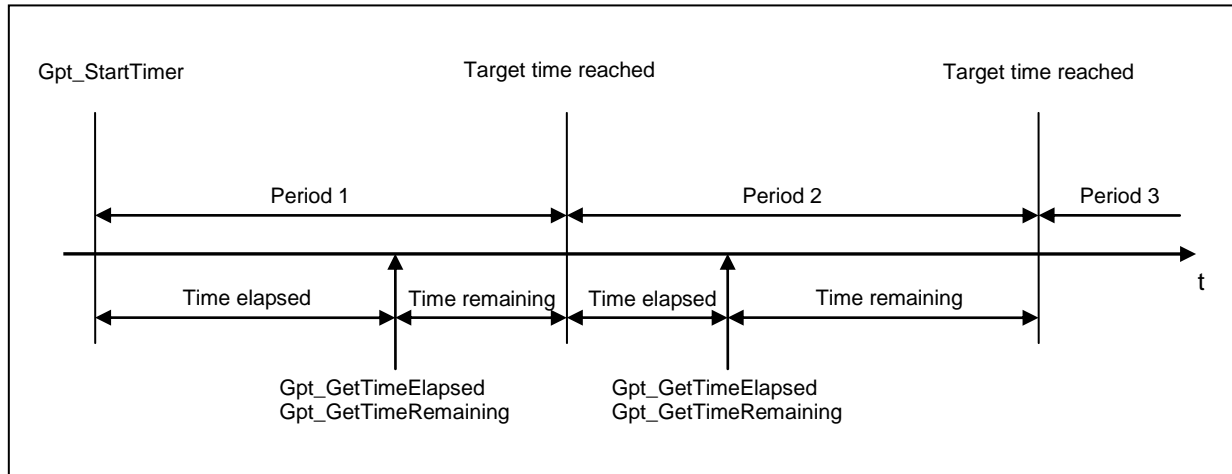


Figure 5: Querying of time elapsed / time remaining for a timer channel in "continuous mode"

[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] [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).] (SRS_Gpt_13601)

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 detailed description on wakeup handling please refer to the ECU State Manager specification [8].

The operation modes and the possible mode transitions of the GPT driver are shown in Figure 6.

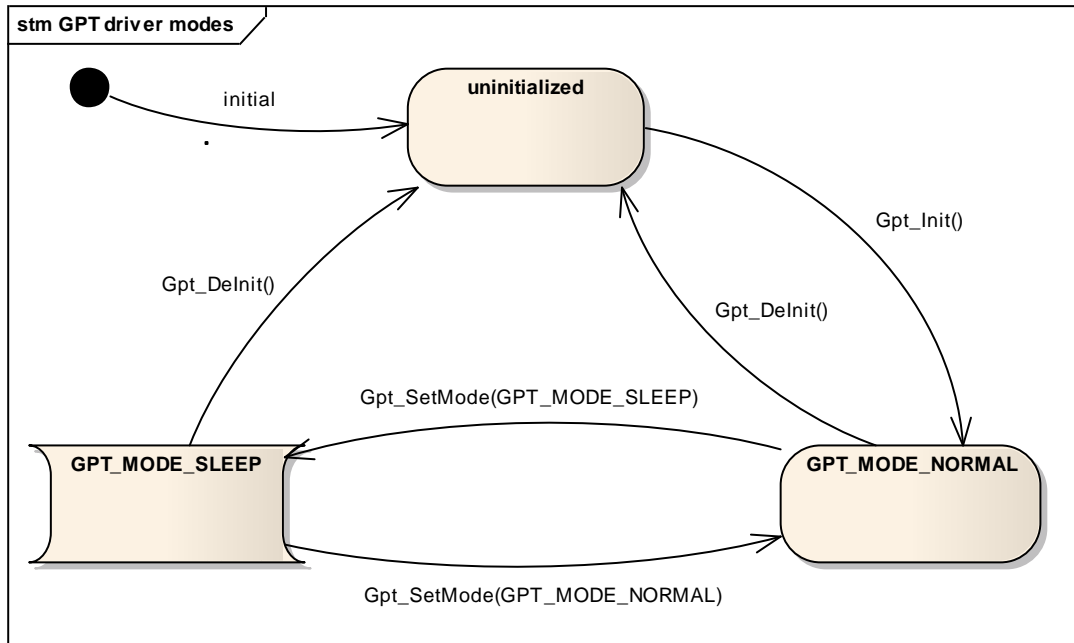


Figure 6: GPT driver modes

7.2 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] [A GPT Predef Timer is a free running up counter (user point of view). If the timer has reached the maximum value (max value = $2^n - 1$, n =number of bits), the timer shall continue running with the value "0" at next timer tick.]
(SRS_Gpt_13604)

[SWS_Gpt_00383] [Types of GPT Predef Timers

| Name of GPT Predef Timer | Tick duration | Maximum tick value | Number of bits | Maximum time span (circa values) |
|------------------------------|---------------|--------------------|----------------|-------------------------------------|
| GPT_PREDEF_TIMER_1US_16BIT | 1 μ s | 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 μ s | 4294967295 | 32 bit | 4.9 days |

] (SRS_Gpt_13605)

[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 " $f_{\text{clock}} = 1 / (\text{tick duration})$ " 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 μ s is used for all GPT Predef Timers

[SWS_Gpt_00385] [It shall be possible to configure which GPT Predef Timers are enabled.] (SRS_Gpt_13606)

[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](#))

7.3 Version checking

For details refer to the chapter 5.1.8 "Version Check" in *SWS_BSWGeneral*.

7.4 Error classification

7.4.1 Development Errors

| <i>ID</i> | <i>Type of error</i> | <i>Relevance</i> | <i>Related error code</i> | <i>Value [hex]</i> |
|----------------------|--|------------------|---------------------------|--------------------|
| SWS_Gpt_00345 | API service called without module initialization | Development | GPT_E_UNINIT | 0x0A |
| SWS_Gpt_00347 | API service for initialization called when already initialized | Development | GPT_E_ALREADY_INITIALIZED | 0x0D |
| SWS_Gpt_00404 | API error return code: Init function failed | Development | GPT_E_INIT_FAILED | 0x0E |
| SWS_Gpt_00348 | API parameter checking: invalid channel | Development | GPT_E_PARAM_CHANNEL | 0x14 |
| SWS_Gpt_00349 | API parameter checking: invalid value | Development | GPT_E_PARAM_VALUE | 0x15 |
| SWS_Gpt_00350 | API parameter checking: invalid pointer | Development | GPT_E_PARAM_POINTER | 0x16 |
| SWS_Gpt_00388 | API parameter checking: invalid Predef Timer | Development | GPT_E_PARAM_PREDEF_TIMER | 0x17 |
| SWS_Gpt_00351 | API parameter checking: invalid mode | Development | GPT_E_PARAM_MODE | 0x1F |

Table 3: Development Error Tables

7.4.2 Runtime Errors

| <i>ID</i> | <i>Type of error</i> | <i>Relevance</i> | <i>Related error code</i> | <i>Value [hex]</i> |
|----------------------|---|------------------|---------------------------|--------------------|
| SWS_Gpt_00346 | API service called when timer channel is still busy (running) | Development | GPT_E_BUSY | 0x0B |
| SWS_Gpt_00387 | API service called when driver is in wrong mode | Development | GPT_E_MODE | 0x0C |

Table 4: Runtime Error Tables

7.4.3 Transient Faults

There are no transient faults.

7.4.4 Production Errors

There are no production errors.

7.4.5 Extended Production Errors

There are no extended production errors.

7.5 Error detection

[SWS_Gpt_00332] [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).] (SRS_SPAL_12448)

8 API specification

8.1 Imported types

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

[SWS_Gpt_00278] [

| Module | Imported Type |
|-----------|-----------------------|
| EcuM | EcuM_WakeupSourceType |
| Std_Types | Std_ReturnType |
| | Std_VersionInfoType |

] (SRS_BSW_00348)

8.2 Type Definitions

8.3 Error notification

For details refer to the chapter 7.4 “Error notification” in *SWS_BSWGeneral*.

8.3.1 Gpt_ConfigType

[SWS_Gpt_00357] [

| | | |
|---------------------|--|---|
| Name: | Gpt_ConfigType | |
| Type: | Structure | |
| Range: | -- | 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. | |

] (SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00438, SRS_BSW_00305, SRS_BSW_00414, SRS_SPAL_12263)

8.3.2 Gpt_ChannelType

[SWS_Gpt_00358] [

| | | |
|---------------------|------------------------------|---|
| Name: | Gpt_ChannelType | |
| Type: | 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. | |

] (SRS_BSW_00305)

8.3.3 Gpt_ValueType

[SWS_Gpt_00359] [

| | | | |
|---------------------|---|----|--|
| Name: | Gpt_ValueType | | |
| Type: | uint | | |
| Range: | -- | -- | The range of this type is μ C 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). | | |

] (SRS_BSW_00305, SRS_SPAL_12063, SRS_Gpt_12328)

8.3.4 Gpt_ModeType

[SWS_Gpt_00360] [

| | | | |
|---------------------|--------------------------|------|---|
| Name: | Gpt_ModeType | | |
| Type: | 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. | | |

] (SRS_BSW_00441, SRS_BSW_00305)

8.3.5 Gpt_PrefDefTimerType

[SWS_Gpt_00389] [

| | | | |
|---------------------|------------------------------|------|--|
| Name: | Gpt_PrefDefTimerType | | |
| Type: | Enumeration | | |
| Range: | GPT_PREDEF_TIMER_1US_16BIT | 0x00 | GPT PrefDef Timer with tick duration 1 μ s and range 16bit |
| | GPT_PREDEF_TIMER_1US_24BIT | 0x01 | GPT PrefDef Timer with tick duration 1 μ s and range 24bit |
| | GPT_PREDEF_TIMER_1US_32BIT | 0x02 | GPT PrefDef Timer with tick duration 1 μ s and range 32bit |
| | GPT_PREDEF_TIMER_100US_32BIT | 0x03 | GPT PrefDef Timer with tick duration 100 μ s and range 32bit |
| Description: | Type for GPT PrefDef Timers | | |

] (SRS_Gpt_13605)

8.4 Function definitions

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

8.4.1 Gpt_GetVersionInfo

[SWS_Gpt_00279] [

| | |
|-------------------------|---|
| Service name: | Gpt_GetVersionInfo |
| Syntax: | <pre>void Gpt_GetVersionInfo(Std_VersionInfoType* VersionInfoPtr)</pre> |
| Service ID[hex]: | 0x00 |

| | |
|----------------------------|--|
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant |
| Parameters (in): | None |
| Parameters (inout): | None |
| Parameters (out): | VersionInfoPtr Pointer to where to store the version information of this module. |
| Return value: | None |
| Description: | Returns the version information of this module. |

] (SRS_BSW_00407)

[SWS_Gpt_00338] [If development error detection is enabled for the GPT module:
If the parameter `VersionInfoPtr` is a null pointer, the function
`Gpt_GetVersionInfo` shall raise the error `GPT_E_PARAM_POINTER`.]
(SRS_BSW_00323)

8.4.2 Gpt_Init

[SWS_Gpt_00280] [

| | |
|----------------------------|---|
| Service name: | Gpt_Init |
| Syntax: | <pre>void Gpt_Init(const Gpt_ConfigType* ConfigPtr)</pre> |
| 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. |

] (SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00438, SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00414, SRS_SPAL_12057)

[SWS_Gpt_00006] [The function `Gpt_Init` shall initialize the hardware timer module according to a configuration set referenced by `ConfigPtr`.]
(SRS_BSW_00101, SRS_SPAL_12057)

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

[SWS_Gpt_00068] [The function `Gpt_Init` shall only initialize the configured resources. Resources that are not configured in the configuration file shall not be touched.] (SRS_SPAL_12125)

The following rules regarding initialization of controller registers shall apply to this driver implementation:

- **[SWS_Gpt_00352]** [If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register.] (SRS_SPAL_12461)

- **[SWS_Gpt_00353]** [If the register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver.]
(SRS_SPAL_12461)
- **[SWS_Gpt_00354]** [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.]
(SRS_SPAL_12461)
- **[SWS_Gpt_00355]** [One-time writable registers that require initialization directly after reset shall be initialized by the startup code.]
(SRS_SPAL_12461)
- **[SWS_Gpt_00356]** [All other registers shall be initialized by the startup code.]
(SRS_SPAL_12461)

[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] [The function `Gpt_Init` shall start all enabled GPT Predef Timers at value "0".] (SRS_Gpt_13607)

8.4.3 Gpt_DeInit

[SWS_Gpt_00281] [

| | |
|----------------------------|-------------------------------|
| 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. |

] (SRS_BSW_00336, SRS_SPAL_12163, SRS_Gpt_12116)

[SWS_Gpt_00008] [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 μ C.]
(SRS_BSW_00336, SRS_SPAL_12163, SRS_Gpt_12116)

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

[SWS_Gpt_00162] [The function `Gpt_DeInit` shall influence only the peripherals, which are allocated by the static configuration.] (SRS_Gpt_12116)

[SWS_Gpt_00308] [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`.] (SRS_Gpt_12116)

[SWS_Gpt_00194] [The function `Gpt_DeInit` shall be pre compile time configurable On/Off by the configuration parameter: `GptDeInitApi`.]
(SRS_BSW_00171)

[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] [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`.] (SRS_BSW_00406)

[SWS_Gpt_00391] [The function `Gpt_DeInit` shall stop all enabled GPT Predef Timers.] (SRS_Gpt_13607)

8.4.4 Gpt_GetTimeElapsed

[SWS_Gpt_00282] [

| | | |
|----------------------------|---|--|
| 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. | |

└ (SRS_Gpt_12117)

[SWS_Gpt_00010] └ 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. └ (SRS_Gpt_12117)

[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] └ The function `Gpt_GetTimeElapsed` shall be pre compile time configurable On/Off by the configuration parameter: `GptTimeElapsedApi`. └ (SRS_BSW_00171)

[SWS_Gpt_00222] └ 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`. └ (SRS_BSW_00406)

[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`. └ ()

| <i>State / Circumstance</i> | <i>Timer channel state</i> | <i>Return value</i> | <i>Development error (if enabled)</i> |
|-----------------------------|------------------------------|------------------------------------|---------------------------------------|
| 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 5: Summary: Return values and DET errors of Gpt_GetTimeElapsed

8.4.5 Gpt_GetTimeRemaining

[SWS_Gpt_00283] [

| | | |
|----------------------------|---|--|
| 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. | |

] (SRS_Gpt_12117)

[SWS_Gpt_00083] [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.] (SRS_Gpt_12117)

[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] [The function Gpt_GetTimeRemaining shall be pre compile time configurable On/Off by the configuration parameter: GptTimeRemainingApi.] (SRS_BSW_00171)

[SWS_Gpt_00223] [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.] (SRS_BSW_00406)

[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 6: Summary: Return values and DET errors of Gpt_GetTimeRemaining

8.4.6 Gpt_StartTimer

[SWS_Gpt_00284] [

| | | | |
|----------------------------|--|--|--|
| 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. | | |

] (SRS_Gpt_12128)

[SWS_Gpt_00274] [The function Gpt_StartTimer shall start the selected timer channel with a defined target time.] (SRS_Gpt_12128)

[SWS_Gpt_00275] [If configured and enabled, an interrupt notification or a wakeup interrupt occurs, when the target time is reached.] (SRS_Gpt_12128)

[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] [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).] (SRS_BSW_00323)

[SWS_Gpt_00224] [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`.] (SRS_BSW_00406)

[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`.] ()

8.4.7 Gpt_StopTimer

[SWS_Gpt_00285] [

| | |
|----------------------------|---|
| 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. |

] (SRS_Gpt_12119)

[SWS_Gpt_00013] [The function `Gpt_StopTimer` shall stop the selected timer channel.] (SRS_Gpt_12119)

[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 channels 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] [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.`] (SRS_BSW_00406)

8.4.8 Gpt_EnableNotification

[SWS_Gpt_00286] [

| | |
|----------------------------|---|
| 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). |

] (SRS_Gpt_12121)

[SWS_Gpt_00014] [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.] (SRS_SPAL_00157, SRS_SPAL_12067, SRS_Gpt_12121)

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] [The function `Gpt_EnableNotification` shall be pre compile time configurable On/Off by the configuration parameter:
`GptEnableDisableNotificationApi.`] (SRS_BSW_00171)

[SWS_Gpt_00226] [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.`] (SRS_BSW_00406)

[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.`] ()

8.4.9 Gpt_DisableNotification

[SWS_Gpt_00287] [

| | |
|----------------------------|--|
| 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). |

] (SRS_Gpt_12122)

[SWS_Gpt_00015] [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.] (SRS_SPAL_00157, SRS_Gpt_12122, SRS_SPAL_12067)

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] [The function `Gpt_DisableNotification` shall be pre compile time configurable On/Off by the configuration parameter: `GptEnableDisableNotificationApi`.] (SRS_BSW_00171)

[SWS_Gpt_00227] [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`.] (SRS_BSW_00406)

[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`.] ()

8.4.10 Gpt_SetMode

[SWS_Gpt_00288] [

| | |
|----------------------|-------------|
| Service name: | Gpt_SetMode |
|----------------------|-------------|

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

└ (SRS_SPAL_12169, SRS_Gpt_13603)

[SWS_Gpt_00151] └ The function `Gpt_SetMode` shall set the operation mode of the GPT driver to the given mode parameter. └ (SRS_SPAL_12169, SRS_Gpt_13603)

[SWS_Gpt_00255] └ The function `Gpt_SetMode` is only available if the configuration parameter `GptReportWakeupSource` is enabled. └ ()

[SWS_Gpt_00152] └ 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. └ (SRS_Gpt_13603)

[SWS_Gpt_00153] └ 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. └ (SRS_Gpt_13603)

[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] [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.`] (SRS_BSW_00406)

[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] [The function `Gpt_SetMode` shall be pre compile time
configurable On/Off by the configuration parameter:
`GptWakeupFunctionalityApi.`] (SRS_BSW_00171)

[SWS_Gpt_00392] [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".] (SRS_Gpt_13607)

[SWS_Gpt_00393] [If the parameter `Mode` has the value `GPT_MODE_SLEEP`:
The function `Gpt_SetMode` shall stop all enabled GPT Predef Timers.]
(SRS_Gpt_13607)

8.4.11 Gpt_DisableWakeup

[SWS_Gpt_00289] [

| | |
|----------------------------|--|
| Service name: | <code>Gpt_DisableWakeup</code> |
| Syntax: | <code>void Gpt_DisableWakeup(Gpt_ChannelType Channel)</code> |
| 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). |

] (SRS_Gpt_13602)

[SWS_Gpt_00159] [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.] (SRS_Gpt_13602)

[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] [The function `Gpt_DisableWakeup` shall be pre compile time configurable On/Off by the configuration parameter: `GptWakeupFunctionalityApi.`] (SRS_BSW_00171)

[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] [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.`] (SRS_BSW_00406)

8.4.12 Gpt_EnableWakeup

[SWS_Gpt_00290] [

| | |
|----------------------------|---|
| Service name: | <code>Gpt_EnableWakeup</code> |
| Syntax: | <pre>void Gpt_EnableWakeup(Gpt_ChannelType Channel)</pre> |
| Service ID[hex]: | 0x0b |
| Sync/Async: | Synchronous |
| Reentrancy: | Reentrant (but not for the same timer channel) |
| Parameters (in): | <code>Channel</code> 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). |

] (SRS_Gpt_13602)

[SWS_Gpt_00160] [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.] (SRS_Gpt_13602)

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] [The function `Gpt_EnableWakeup` shall be pre compile time configurable On/Off by the configuration parameter: `GptWakeupFunctionalityApi.`] (SRS_BSW_00171)

[SWS_Gpt_00230] [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.`] (SRS_BSW_00406)

[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.`] ()

8.4.13 Gpt_CheckWakeup

[SWS_Gpt_00328] [

| | | |
|----------------------------|---|---|
| Service name: | Gpt_CheckWakeup | |
| Syntax: | <pre>void Gpt_CheckWakeup(Ecum_WakeupSourceType WakeupSource)</pre> | |
| 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 <code>Ecum_SetWakeupEvent</code> in case of a valid GPT channel wakeup event. | |

] ()

[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 [8].] ()

[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] [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.`] (SRS_BSW_00406)

8.4.14 Gpt_GetPredefTimerValue

[SWS_Gpt_00394] [

| | | |
|----------------------------|--|---|
| Service name: | Gpt_GetPredefTimerValue | |
| Syntax: | <pre>Std_ReturnType Gpt_GetPredefTimerValue(Gpt_PredefTimerType PredefTimer, uint32* TimeValuePtr)</pre> | |
| 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. | |

] (SRS_Gpt_13608)

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] [The function Gpt_GetPredefTimerValue shall return the current value of the GPT Predef Timer passed by PredefTimer.]
(SRS_Gpt_13608)

[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] [The function Gpt_GetPredefTimerValue shall be fully reentrant, this means even for the same GPT Predef Timer.] (SRS_Gpt_13608)

[SWS_Gpt_00402] [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.] (SRS_BSW_00406)

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] [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.] (SRS_BSW_00406)

[SWS_Gpt_00399] [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.`] (SRS_BSW_00323)

[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] [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.`] (SRS_BSW_00369, SRS_BSW_00323)

8.5 Call-back Notifications

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

8.6 Scheduled functions

None.

8.7 Expected Interfaces

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

8.7.1 Mandatory Interfaces

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

[SWS_Gpt_00405]

| API function | Description |
|------------------------|-----------------------------------|
| Det_ReportRuntimeError | Service to report runtime errors. |

] (SRS_SPAL_00157 , SRS_SPAL_12064)

8.7.2 Optional Interfaces

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

[SWS_Gpt_00406][

| API function | Description |
|---------------------|---|
| Det_ReportError | Service to report development errors. |
| EcuM_CheckWakeup | This callout is called by the EcuM to poll a wakeup source. It shall 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 | Sets the wakeup event. |

] (SRS_SPAL_00157)

[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] [The ISR's, providing the wakeup events, shall be responsible for resetting the interrupt flags (if needed by hardware).] (SRS_SPAL_12129)

8.7.3 Configurable Interfaces

In this chapter all interfaces are listed where the target function could be configured. The target function is usually a call-back function. The names of these kinds of interfaces is not fixed because they are configurable.

8.7.3.1 GPT Notification

[SWS_Gpt_00292] [

| | |
|----------------------------|---|
| 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: | -- |

] (SRS_BSW_00375, SRS_SPAL_12069)

The notification prototype `Gpt_Notification_<channel>` is for the notification callback function and shall be implemented by the user.

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] [Each channel shall provide its own notification if configured.] (SRS_BSW_00375, SRS_SPAL_12069)

[SWS_Gpt_00093] [When disabled, the GPT Driver will send no notification.] ()

[SWS_Gpt_00233] [The GPT Driver shall invoke a notification whenever the defined target time of the channel is reached.] (SRS_SPAL_12067, SRS_Gpt_12120)

[SWS_Gpt_00206] [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.] (SRS_SPAL_12129)

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

9 Sequence diagrams

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

9.1 Gpt_Init

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

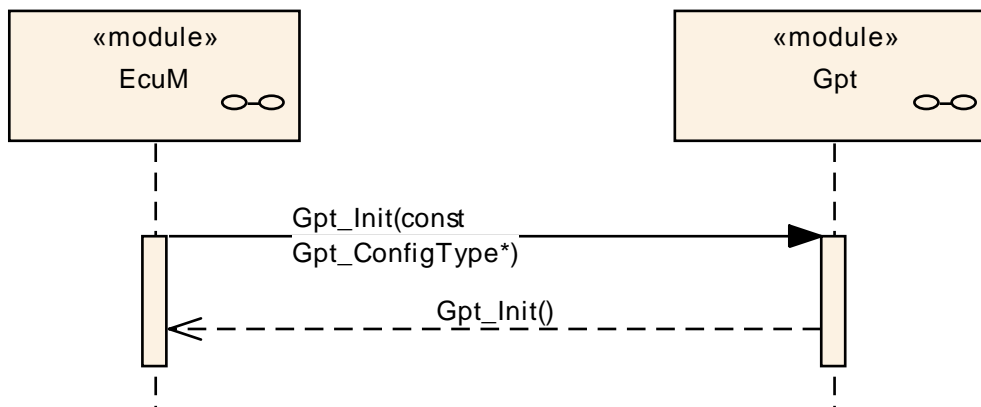


Figure 7: Sequence Diagram - Gpt_Init

9.2 GPT continuous mode

Channel 2 is configured as “Continuous Mode”

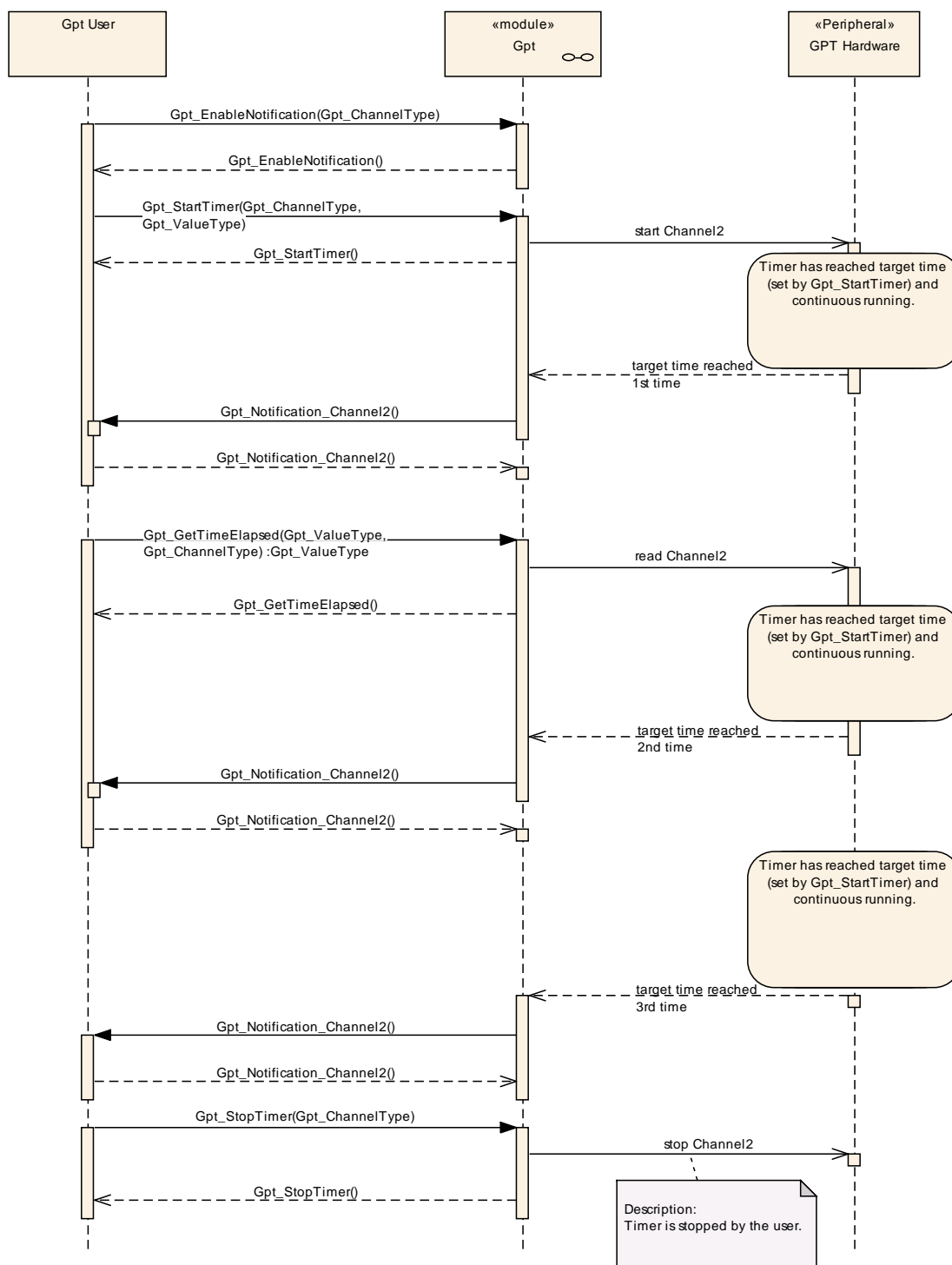


Figure 8: Sequence Diagram - GPT continuous mode

9.3 GPT one-shot mode

Channel 1 is configured for “One-shot Mode”

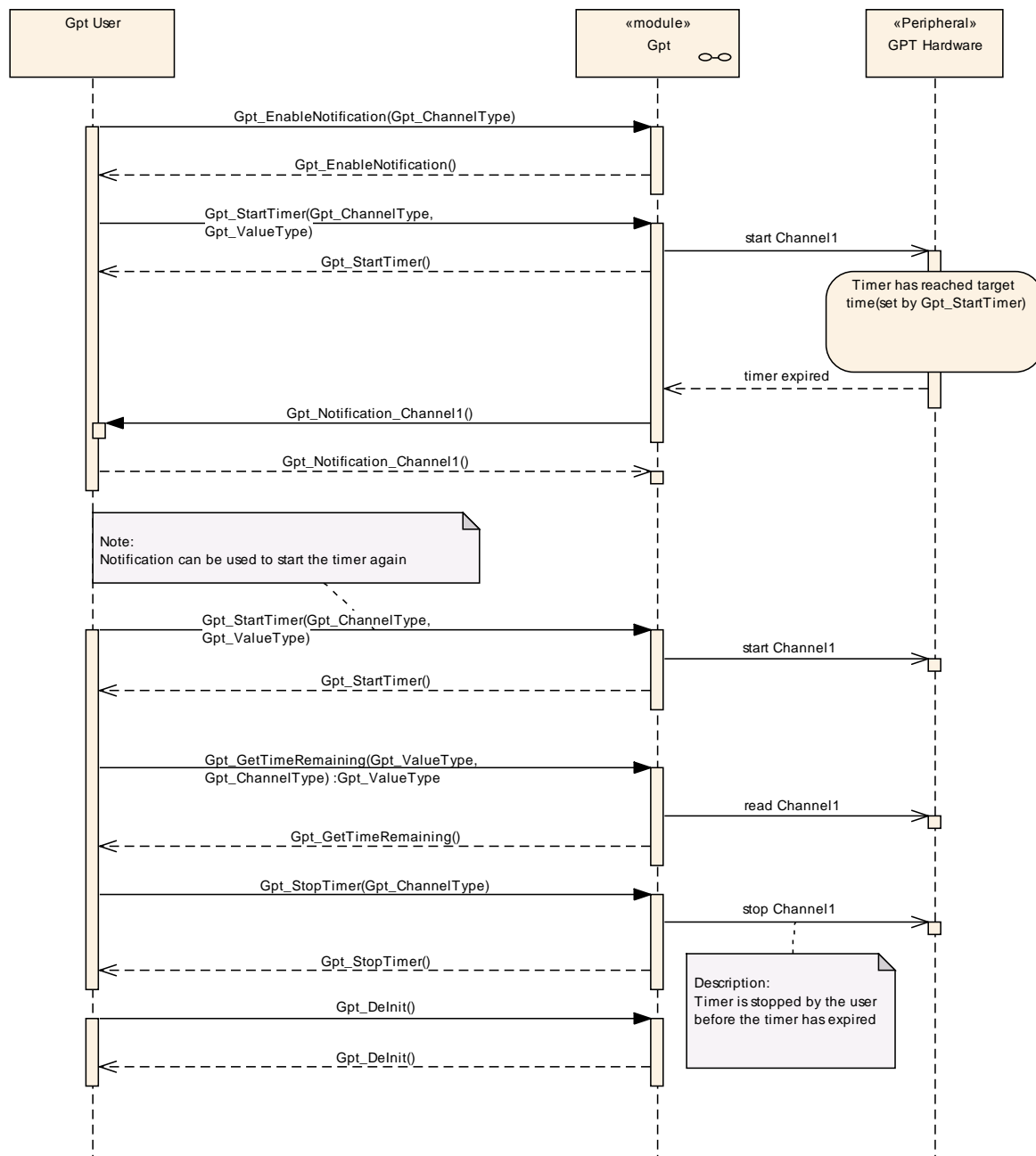


Figure 9: Sequence Diagram - GPT one-shot mode

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.

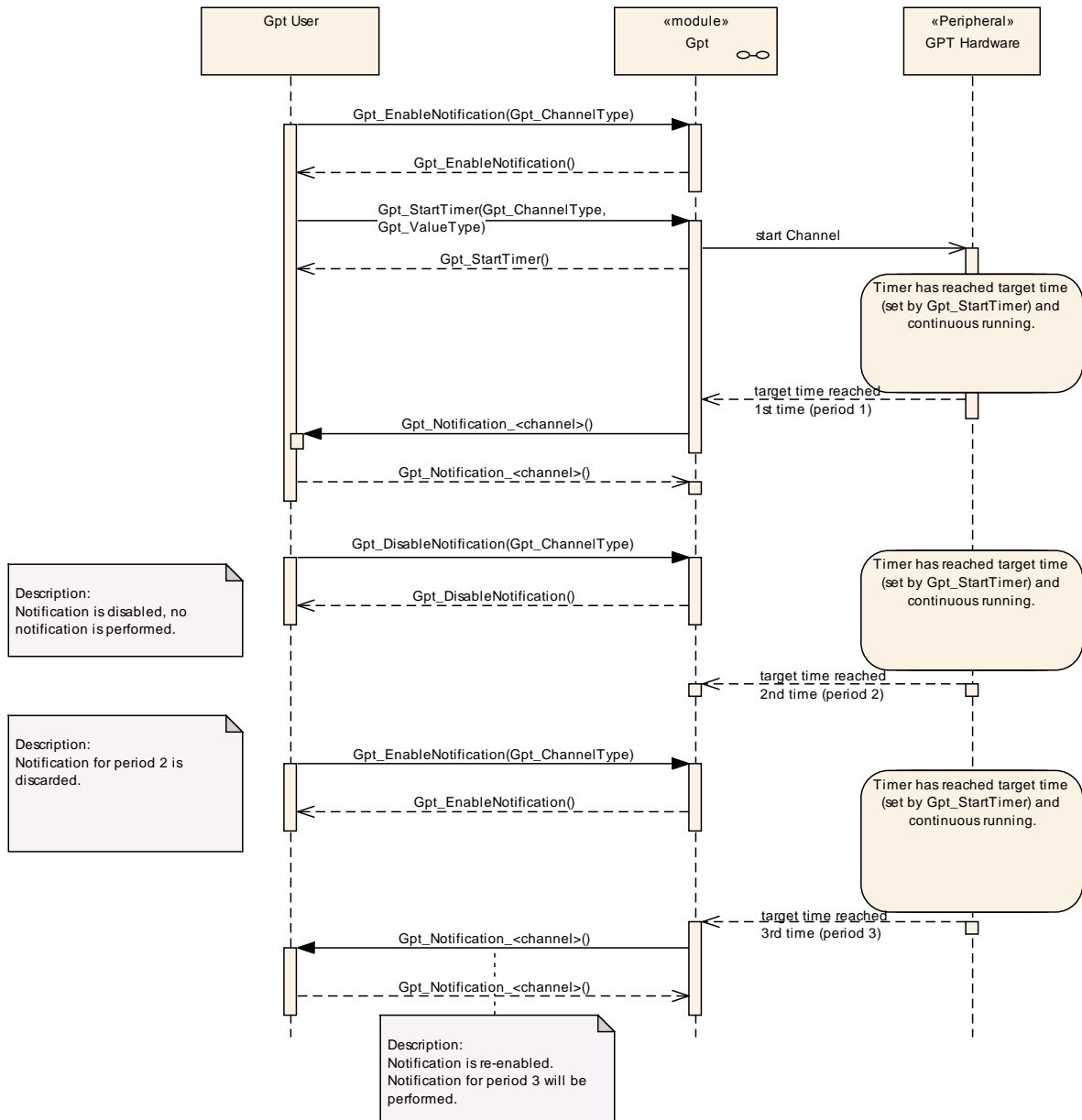


Figure 10: Sequence Diagram - Disable/Enable Notifications

9.5 Wakeup

Note: Sequence charts on timer wakeup can be found in the ECU state manager specification [8].

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 the chapter 10.1 "Introduction to configuration specification" in *SWS_BSWGeneral*.

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.

10.2.1 Variants

[SWS_Gpt_00270] [Within one container it shall not be possible to mix parameters assigned to different configuration classes.] ()

10.2.2 Gpt

| | |
|-----------------------------------|--|
| SWS Item | ECUC_Gpt_00336 : |
| Module Name | <i>Gpt</i> |
| Module Description | Configuration of the Gpt (General Purpose Timer) module. |
| Post-Build Variant Support | true |
| Supported Config Variants | VARIANT-POST-BUILD, VARIANT-PRE-COMPILE |

| Included Containers | | |
|----------------------------------|---------------------|---|
| Container Name | Multiplicity | Scope / Dependency |
| 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 |

10.2.3 GptDriverConfiguration

| | |
|---------------------------------|--|
| SWS Item | ECUC_Gpt_00183 : |
| Container Name | GptDriverConfiguration |
| Description | This container contains the module-wide configuration (parameters) of the GPT Driver |
| Configuration Parameters | |

| | | | |
|----------------------------------|---|----|--------------|
| SWS Item | ECUC_Gpt_00321 : | | |
| Name | GptDevErrorDetect | | |
| Parent Container | GptDriverConfiguration | | |
| Description | Switches the development error detection and notification on or off. <ul style="list-style-type: none"> true: detection and notification is enabled. false: detection and notification is disabled. | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | false | | |
| Post-Build Variant Value | false | | |
| Value Configuration Class | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

| | | | |
|-------------------------|---|--|--|
| SWS Item | ECUC_Gpt_00335 : | | |
| Name | GptPredefTimer100us32bitEnable | | |
| Parent Container | GptDriverConfiguration | | |
| Description | Enables/disables the GPT Predef Timer 100µs32bit. | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | -- | | |

| | | | |
|----------------------------------|-------------------------|----|--------------|
| Post-Build Variant Value | false | | |
| Value Configuration Class | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: ECU | | |

| | | | |
|---------------------------|---|---------------------------------|--------------|
| SWS Item | ECUC_Gpt_00334 : | | |
| Name | GptPredefTimer1usEnablingGrade | | |
| Parent Container | GptDriverConfiguration | | |
| 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 | |
| Post-Build Variant Value | false | | |
| Value Configuration Class | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope Dependency | /scope: ECU | | |

| | | | |
|----------------------------------|---|----|--------------|
| SWS Item | ECUC_Gpt_00322 : | | |
| Name | GptReportWakeupSource | | |
| Parent Container | GptDriverConfiguration | | |
| Description | Enables/Disables wakeup source reporting. | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | -- | | |
| Post-Build Variant Value | false | | |
| Value Configuration Class | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

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

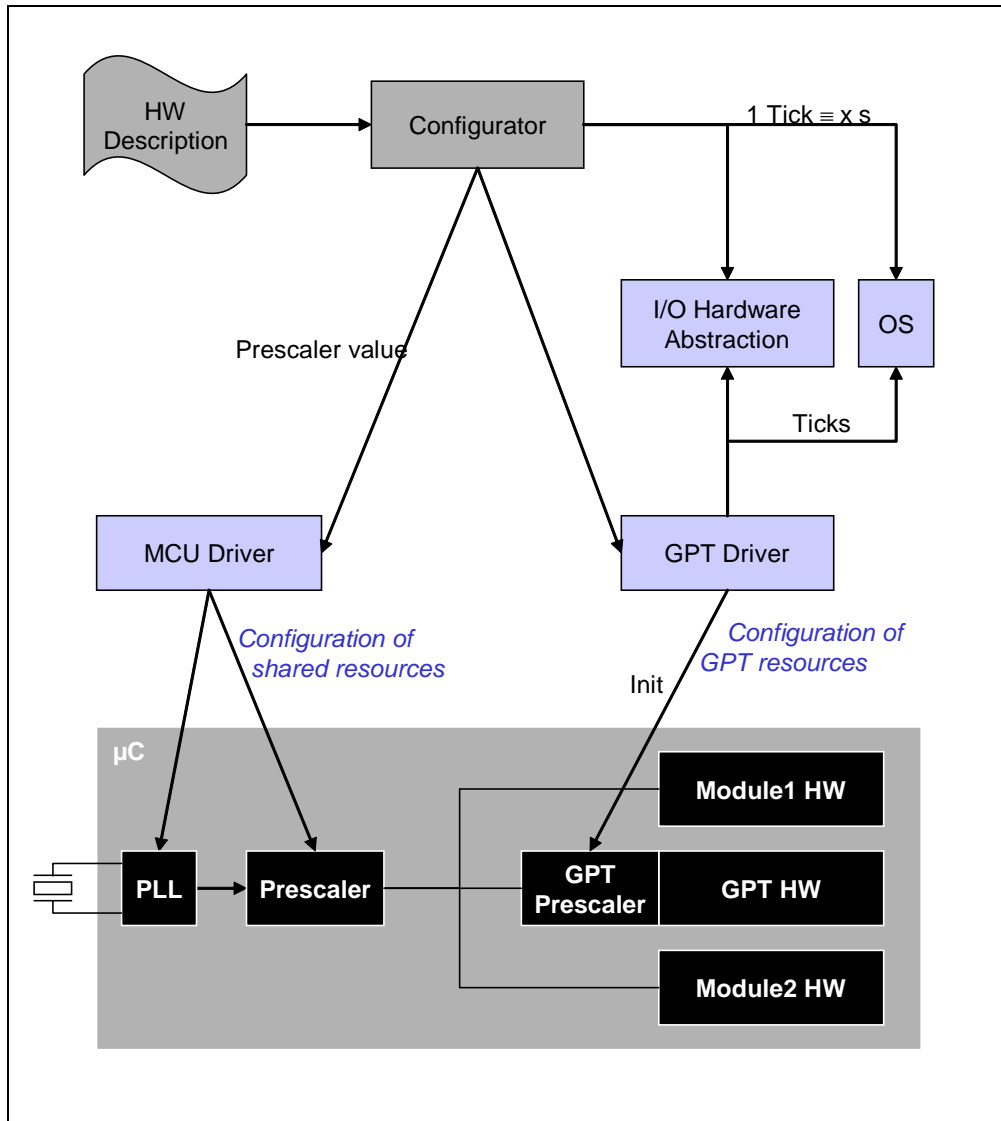


Figure 11: Scope of the GPT Driver configuration

10.2.4 GptClockReferencePoint

| | |
|---------------------------------|---|
| SWS Item | ECUC_Gpt_00329 : |
| Container Name | GptClockReferencePoint |
| 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). |
| Configuration Parameters | |
| SWS Item | ECUC_Gpt_00330 : |
| Name | GptClockReference |
| Parent Container | GptClockReferencePoint |
| 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] | | |
| Post-Build Variant Value | false | | |
| Value Configuration Class | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

No Included Containers

10.2.5 GptChannelConfigSet

| | |
|---------------------------------|---|
| SWS Item | ECUC_Gpt_00269 : |
| Container Name | GptChannelConfigSet |
| 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. |
| Configuration Parameters | |

| | | | |
|----------------------------|---------------------|---|--|
| Included Containers | | | |
| Container Name | Multiplicity | Scope / Dependency | |
| GptChannelConfiguration | 1..* | This container contains the channel specific configuration of the GPT Driver. | |

10.2.6 GptChannelConfiguration

| | |
|---------------------------------|---|
| SWS Item | ECUC_Gpt_00184 : |
| Container Name | GptChannelConfiguration |
| Description | Configuration of an individual GPT channel. |
| Configuration Parameters | |

| | | | |
|----------------------------------|--|----|--------------|
| SWS Item | ECUC_Gpt_00308 : | | |
| Name | GptChannelId | | |
| Parent Container | GptChannelConfiguration | | |
| 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 | -- | | |
| Post-Build Variant Value | false | | |
| Value Configuration Class | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: ECU | | |

| | | | |
|----------------------------------|---|--|---------------------|
| SWS Item | ECUC_Gpt_00309 : | | |
| Name | GptChannelMode | | |
| Parent Container | GptChannelConfiguration | | |
| 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 Variant Value | true | | |
| Value Configuration Class | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope Dependency | scope: ECU | | |

| | | | |
|----------------------------------|--|----|---------------------|
| SWS Item | ECUC_Gpt_00331 : | | |
| Name | GptChannelTickFrequency | | |
| Parent Container | GptChannelConfiguration | | |
| Description | Specifies the tick frequency of the timer channel in Hz. | | |
| Multiplicity | 1 | | |
| Type | EcucFloatParamDef | | |
| Range | [0 .. INF] | | |
| Default value | -- | | |
| Post-Build Variant Value | true | | |
| Value Configuration Class | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: ECU | | |

| | | | |
|----------------------------------|---|----|---------------------|
| SWS Item | ECUC_Gpt_00332 : | | |
| Name | GptChannelTickValueMax | | |
| Parent Container | GptChannelConfiguration | | |
| 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 Variant Value | true | | |
| Value Configuration Class | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: ECU | | |

| | | | |
|-------------------------|---|--|--|
| SWS Item | ECUC_Gpt_00311 : | | |
| Name | GptEnableWakeup | | |
| Parent Container | GptChannelConfiguration | | |
| Description | Enables wakeup capability of MCU for a channel. | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |

| | | | |
|----------------------------------|-------------------------|----|---------------------|
| Default value | -- | | |
| Post-Build Variant Value | true | | |
| Value Configuration Class | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| | | | |
|---|---|----|---------------------|
| SWS Item | ECUC_Gpt_00312 : | | |
| Name | GptNotification | | |
| Parent Container | GptChannelConfiguration | | |
| Description | Function pointer to callback function (for non-wakeup notification) | | |
| Multiplicity | 0..1 | | |
| Type | EcucFunctionNameDef | | |
| Default value | -- | | |
| maxLength | -- | | |
| minLength | -- | | |
| regularExpression | -- | | |
| Post-Build Variant Multiplicity | true | | |
| Post-Build Variant Value | true | | |
| Multiplicity Configuration Class | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Value Configuration Class | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| | | | |
|----------------------------------|--|----|---------------------|
| SWS Item | ECUC_Gpt_00333 : | | |
| Name | GptChannelClkSrcRef | | |
| Parent Container | GptChannelConfiguration | | |
| Description | Reference to the GptClockReferencePoint from which the channel clock is derived. | | |
| Multiplicity | 1 | | |
| Type | Reference to [GptClockReferencePoint] | | |
| Post-Build Variant Value | true | | |
| Value Configuration Class | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

| | | |
|----------------------------|---------------------|--|
| Included Containers | | |
| Container Name | Multiplicity | Scope / Dependency |
| GptWakeupConfiguration | 0..1 | Function pointer to callback function (for non-wakeup notification). |

10.2.7 GptWakeupConfiguration

| | |
|---------------------------------|--|
| SWS Item | ECUC_Gpt_00235 : |
| Container Name | GptWakeupConfiguration |
| Description | Function pointer to callback function (for wakeup notification). |
| Configuration Parameters | |

| | | | |
|----------------------------------|--|----|---------------------|
| SWS Item | ECUC_Gpt_00313 : | | |
| Name | GptWakeupSourceRef | | |
| Parent Container | GptWakeupConfiguration | | |
| 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 Variant Value | true | | |
| Value Configuration Class | Pre-compile time | X | VARIANT-PRE-COMPILE |
| | Link time | -- | |
| | Post-build time | X | VARIANT-POST-BUILD |
| Scope / Dependency | scope: local | | |

No Included Containers

10.2.8 GptConfigurationOfOptApiServices

| | |
|---------------------------------|---|
| SWS Item | ECUC_Gpt_00193 : |
| Container Name | GptConfigurationOfOptApiServices |
| Description | This container contains all configuration switches for configuring optional API services of the GPT driver. |
| Configuration Parameters | |

| | | | |
|----------------------------------|--|----|--------------|
| SWS Item | ECUC_Gpt_00314 : | | |
| Name | GptDeinitApi | | |
| Parent Container | GptConfigurationOfOptApiServices | | |
| Description | Adds / removes the service Gpt_DeInit() from the code. | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | -- | | |
| Post-Build Variant Value | false | | |
| Value Configuration Class | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

| | | | |
|----------------------------------|---|----|--------------|
| SWS Item | ECUC_Gpt_00315 : | | |
| Name | GptEnableDisableNotificationApi | | |
| Parent Container | GptConfigurationOfOptApiServices | | |
| Description | Adds / removes the services Gpt_EnableNotification() and Gpt_DisableNotification from the code. | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | -- | | |
| Post-Build Variant Value | false | | |
| Value Configuration Class | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

| | | | |
|-----------------|-------------------------|--|--|
| SWS Item | ECUC_Gpt_00317 : | | |
| Name | GptTimeElapsedApi | | |

| | | | |
|----------------------------------|---|----|--------------|
| Parent Container | GptConfigurationOfOptApiServices | | |
| Description | Adds / removes the service Gpt_GetTimeElapsed() from the code | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | -- | | |
| Post-Build Variant Value | false | | |
| Value Configuration Class | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

| | | | |
|----------------------------------|--|----|--------------|
| SWS Item | ECUC_Gpt_00318 : | | |
| Name | GptTimeRemainingApi | | |
| Parent Container | GptConfigurationOfOptApiServices | | |
| Description | Adds / removes the service Gpt_GetTimeRemaining() from the code. | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | -- | | |
| Post-Build Variant Value | false | | |
| Value Configuration Class | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

| | | | |
|----------------------------------|--|----|--------------|
| SWS Item | ECUC_Gpt_00319 : | | |
| Name | GptVersionInfoApi | | |
| Parent Container | GptConfigurationOfOptApiServices | | |
| Description | Adds / removes the service Gpt_GetVersionInfo() from the code. | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | false | | |
| Post-Build Variant Value | false | | |
| Value Configuration Class | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

| | | | |
|----------------------------------|--|----|--------------|
| SWS Item | ECUC_Gpt_00320 : | | |
| Name | GptWakeupFunctionalityApi | | |
| Parent Container | GptConfigurationOfOptApiServices | | |
| Description | Adds / removes the services Gpt_SetMode(), Gpt_EnableWakeup() Gpt_DisableWakeup() and Gpt_CheckWakeup() from the code. | | |
| Multiplicity | 1 | | |
| Type | EcucBooleanParamDef | | |
| Default value | -- | | |
| Post-Build Variant Value | false | | |
| Value Configuration Class | Pre-compile time | X | All Variants |
| | Link time | -- | |
| | Post-build time | -- | |
| Scope / Dependency | scope: local | | |

| | | | |
|-------------------------------|--|--|--|
| No Included Containers | | | |
|-------------------------------|--|--|--|

10.3 Published Information

[SWS_Gpt_00380] [The standardized common published parameters as required by SRS_BSW_00402 in the SRS General on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1].] ()

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

11 Not applicable requirements

[SWS_Gpt_00381] [These requirements are not applicable to this specification.]
 (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_BSW_00161,
 SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00415, SRS_BSW_00325,
 SRS_BSW_00342, SRS_BSW_00160, SRS_BSW_00007, SRS_BSW_00413,
 SRS_BSW_00347, SRS_BSW_00307, SRS_BSW_00373, SRS_BSW_00335,
 SRS_BSW_00348, SRS_BSW_00353, SRS_BSW_00361, SRS_BSW_00328,
 SRS_BSW_00006, SRS_BSW_00439, SRS_BSW_00357, SRS_BSW_00377,
 SRS_BSW_00378, SRS_BSW_00306, SRS_BSW_00308, SRS_BSW_00309,
 SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00440, SRS_BSW_00330,
 SRS_BSW_00331, SRS_BSW_00009, SRS_BSW_00172, SRS_BSW_00010,
 SRS_BSW_00333, SRS_BSW_00321, SRS_BSW_00341, SRS_BSW_00334,
 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)