Document Title	Specification of GPT Driver
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	2.1.0
Document Status	Draft
Part of Release	2.1
Revision	0014

Document Change History			
Date	Version	Changed by	Change Description
31.01.2007	2.1.0	AUTOSAR Administration	• 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
28.04.2006	2.0.0	AUTOSAR Administration	Document structure adapted to common Release 2.0 SWS Template. • Added wake-up functionality For more details see chapter 11
03.06.2005	1.0.0	AUTOSAR Administration	1.0.0 Initial release

Release Notes

Errata and known deficiencies

The wakeup concept is currently neither harmonized nor consistent throughout all wakeup related specification documents and therefore subject to change.

Known and potential problems resulting from known deficiencies

The Timer Wakeup can currently only be realized with proprietary implementation extensions.

Due to the fact that the wakeup concept is not harmonized, inconsistent assumptions may lead to

- The duplication of functionalities across multiple modules
- Proprietary implementation extensions
- Difficulties during integration

Changes planned for next release

The following changes are planned for the next releases:

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.

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).
Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module GPT Driver.

There are 2 categories of timers within AUTOSAR:

1. Operating system timers (AUTOSAR OS Alarms)
2. Hardware timers,

but only hardware timers are within the scope of this document.

Type of Timer	***Specification/ Implementation within Autosar***	***Usage within Autosar***	***Use case***
Operating system timers	Operating system	*BSW:* directly *Application:* via RTE	Activation of periodic tasks.
			Interior light time-out after door close
Hardware timers	**GPT Driver**	**BSW only**	**Valve timing** **Stepper motor control**

Table 1: Scope of the GPT Driver specification

The module only uses the hardware timer channels of the general-purpose timer unit and thus provides exact and short-term timings for use in the Operating system or within other basic software modules where an OS Alarm service has too much overhead.

An example of a typical time period range is 50µs … 5ms.

2 Acronyms and abbreviations

Acronym:	*Description:*
Timer channel	Each channel represents one instance of GPT hardware. It identifies a resource that provides a timing value and/or a notification.
Timeout period	Number of ticks, after the timer will expire.
One shot mode	Timer channel stops after reaching its timeout period value
Continuous mode	Timer channel is restarted automatically after reaching timeout period value

Table 2: Acronyms

Abbreviation:	*Description:*
DEM	Diagnostic Event Manager
DET	Development Error Tracer
GPT	General Purpose Timer
ICU	Input Capture Unit
ECU	Electronic Control Unit
MCU	Micro Controller Unit
BSW	Basic SoftWare
OS	Operating System

Table 3: Abbreviations

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_General.pdf

[4] Specification of Development Error Tracer,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[5] Specification of ECU Configuration,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_ECU_Configuration.pdf

[6] Specification of Diagnostic Event Manager,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_DEM.pdf

[7] Specification of ECU State Manager,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_ECU_StateManager.pdf

[8] General Requirements on SPAL,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_SPAL_General.pdf

[9] Requirements on GPT Driver,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_GPT_Driver.pdf

[10] Specification of ICU Driver,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_ICU_Driver.pdf

[11] Specification of MCU Driver,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_MCU_Driver.doc

[12] Specification of I/O Hardware Abstraction,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_IOHW_Abstraction.pdf

[13] Glossary,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_Glossary.pdf

3.2 Related standards and norms

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

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

Module DET [4]
In development mode the Error hook-function of module DET [4] will be called.

Module DEM [6]
Production errors will be reported to the Diagnostic Event Manager

Module MCU [11]
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 setting the registers, which configure the clock, prescaler(s) and PLL in its init function. This has to be done by the MCU module [11].

Module EcuM [7]
This module processes the wakeup notifications of the GPT.

5.1 File structure

5.1.1 Code file structure

GPT171: The code file structure shall not be defined within this specification.

5.1.2 Header file structure

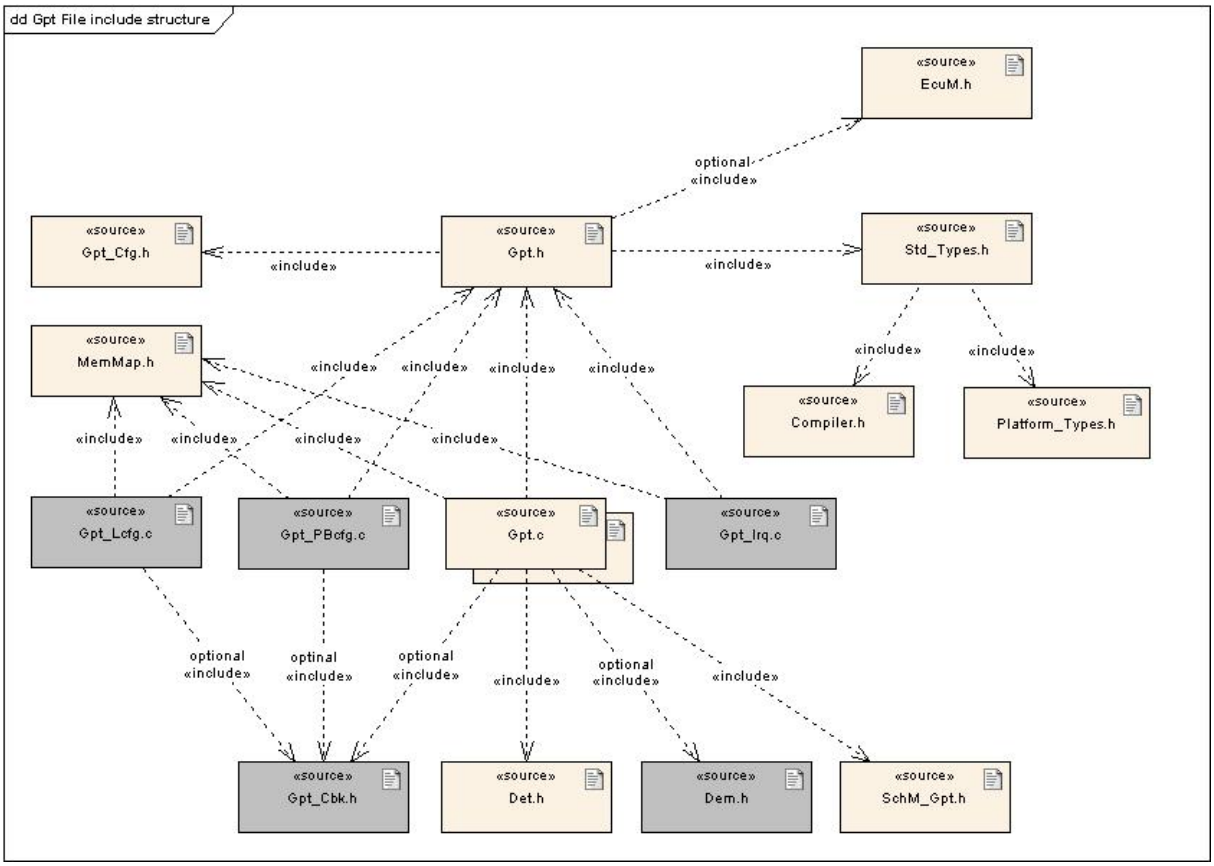

Figure 1: Header file structure

GPT172: The module shall include the Dem.h file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols, which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

The gray boxes are optional:

GPT259: `Gpt.h` shall include `Gpt_Cfg.h` for the API pre-compiler switches

GPT260: `Gpt.c` has access to the `Gpt_Cfg.h` via the implicitly include through the `Gpt.h` file.

GPT261: `Gpt_Irq.c` shall include `Gpt.h` for the function, which shall be called in the interrupt function.

GPT262: The Type definitions for `Gpt_Lcfg.c` and `Gpt_PBcfg.c` are located in the file `Gpt_Cfg.h`. or `Gpt.h`.

GPT264: `Gpt_Lcfg.c` shall include `Gpt_Cbk.h` for a link time configuration, if the call back function is linked to the module via the ROM structure.

GPT265: `Gpt_PBcfg.c` shall include `Gpt_Cbk.h` for a post build time configuration, if the call back function is linked to the module via the ROM structure.

GPT266: `Gpt.c` shall include `Gpt_Cbk.h` for a pre-compile time configuration.

GPT271: The inclusion of EcuM.h is required, if wakeup functionality is configured.

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.

Document: General Requirements on Basic Software Modules [3]

Requirement	***Satisfied by***
[BSW00344] Reference to link-time configuration	GPT184
[BSW00404] Reference to post build-time configuration	Not applicable (no post-build time configurable parameters specified)
[BSW00405] Reference to multiple configuration sets	Not applicable (no post-build time configurable parameters specified)
[BSW00345] Pre-compile-time configuration	GPT183
[BSW159] Tool-based configuration	See Figure 1
[BSW167] Static configuration checking	Not applicable (requirement on configuration tool)
[BSW171] Configurability of optional functionality	GPT182, GPT193, GPT194, GPT195, GPT196, GPT199, GPT200, GPT201, GPT202, GPT203
[BSW170] Data for reconfiguration of AUTOSAR SW-components	Not applicable (requirement on SW component)
[BSW00380] Separate C-File for configuration parameters	See Figure 1
[BSW00419] Separate C-Files for pre-compile time configuration parameters	See Figure 1
[BSW00381] Separate configuration header file for pre-compile time parameters	See Figure 1
[BSW412] Separate H-File for configuration parameters	See Figure 1
[BSW00383] List dependencies of configuration files	See section 10.2
[BSW00384] List dependencies to other modules	See section 10.2
[BSW00387] Specify the configuration class of callback function	See sections 8.6.4 and 10.2
[BSW00388] Introduce containers	GPT183, GPT184, GPT193, GPT235, GPT269
[BSW00389] Containers shall have names	GPT183, GPT184, GPT193, GPT235, GPT269
[BSW00390] Parameter content shall be unique within the module	GPT183, GPT184, GPT193, GPT235, GPT189, GPT269
[BSW00391] Parameter shall have unique names	GPT183, GPT184, GPT193, GPT235, , GPT189
[BSW00392] Parameters shall have a type	GPT183, GPT184, GPT193, GPT235, , GPT189
[BSW00393] Parameters shall have a range	GPT183, GPT184, GPT193, GPT235
[BSW00394] Specify the scope of the parameters	GPT183, GPT184, GPT193, GPT235
[BSW00395] List the required parameters (per parameter)	GPT240, GPT241, GPT242, see also Figure 10

Requirement	*Satisfied by*
[BSW00396] Configuration classes	GPT183, GPT184, GPT193, GPT235
[BSW00397] Pre-compile-time parameters	GPT183
[BSW00398] Link-time parameters	GPT184
[BSW00399] Loadable Post-build time parameters	Not applicable (no post-build time configurable parameters specified)
[BSW00400] Selectable Post-build time parameters	Not applicable (no post-build time configurable parameters specified)
[BSW00402] Published information	GPT189
[BSW00375] Notification of wake-up reason	GPT188, GPT235
[BSW101] Initialization interface	GPT006
[BSW00416] Sequence of initialization	Nor applicable (GPT is not responsible for overall Autosar modules initialization)
[BSW00406] check module initialization	GPT220, GPT221, GPT222, GPT223, GPT224, GPT225, GPT226, GPT227, GPT228, GPT229, GPT230
[BSW168] Diagnostic Interface of SW components	Not applicable (requirement on SW components)
[BSW00407] Function to read out published parameters	GPT189, GPT181
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (GPT driver has no Autosar Interface)
[BSW00424] BSW main processing function task allocation	Not applicable (no main processing function specified)
[BSW00425] Trigger conditions for schedulable objects	Not applicable (requirement for the implementer)
[BSW00426] Exclusive areas in BSW modules	Not applicable (applies only for the module description template)
[BSW00427] ISR description for BSW modules	Not applicable (applies only for the module description template)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (no main processing function specified)
[BSW00429] Restricted BSW OS functionality access	Not applicable (requirement for the implementer)
[BSW00431]The BSW Scheduler module implements task bodies	Not applicable (no scheduling functionality in the GPT module)
[BSW00432] Modules should have separate main processinf functions for read/receive and write/transmit data path	Not applicable (no main processing function specified)
[BSW00433] Calling of main processing functions	Not applicable (no main processing function specified)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (no scheduling functionality in the GPT module)
[BSW00336] Shutdown interface	GPT008
[BSW00337] Classification of errors	GPT001, GPT004
[BSW00338] Detection and Reporting of development errors	GPT178, GPT204

Requirement	*Satisfied by*
[BSW00369] Do not return development error codes via API	GPT178, GPT179, GPT204
[BSW00339] Reporting of production relevant error status	GPT179
[BSW00421] Reporting of production relevant error events	GPT179, GPT245
[BSW00420] Production relevant error event rate detection	Not applicable (applies only for DEM)
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable (applies only for non BSW modules)
[BSW00323] API Parameter checking	GPT001, GPT204, GPT210, GPT211, GPT212, GPT213, GPT214, GPT215, GPT216, GPT217, GPT218
[BSW004] Version check	GPT256
[BSW00409] Header files for production code error IDs	GPT172
[BSW00385] List possible error notifications	GPT001, GPT004, see also Table 4
[BSW00386] Configuration for detecting an error	GPT204, see also Table 5
[BSW161] Microcontroller abstraction	Not applicable (architectural AUTOSAR concept is the basis for this driver)
[BSW162] ECU layout abstraction	Not applicable (architectural AUTOSAR concept is the basis for this driver)
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (architectural AUTOSAR concept is the basis for this driver)
[BSW00415] User dependent include files	See Figure 1
[BSW164] Implementation of interrupt service routines	Not applicable (GPT is part of the MCAL and thus is allowed to implement ISRs).
[BSW00325] Runtime of interrupt service routines	Not applicable (requirement on implementation)
[BSW00326] Transition from ISRs to OS tasks	Not applicable (requirement on implementation)
[BSW00342] Usage of source code and object code	Not applicable (requirement on implementation)
[BSW00343] Specification and configuration of time	GPT055, GPT192
[BSW160] Human-readable configuration data	Not applicable (requirement on implementation)
[BSW007] HIS MISRA C	Not applicable (requirement on implementation)
[BSW00300] Module naming convention	See Figure 1
[BSW00413] Accessing instances of BSW modules	Not applicable (requirement on implementation)
[BSW00347] Naming separation of different instances of BSW drivers	Not applicable (requirement on implementation)
[BSW00305] Self-defined data types naming convention	See section 8.2
[BSW00307] Global variables naming convention	Not applicable (requirement on implementation)
[BSW00310] API naming convention	See section 8.3

Requirement	*Satisfied by*
[BSW00373] Main processing function naming convention	Not applicable (no main processing function specified)
[BSW00327] Error values naming convention	See Table 4
[BSW00335] Status values naming convention	Not applicable (no status values specified within this SWS)
[BSW00350] Development error detection keyword	GPT183
[BSW00408] Configuration parameter naming convention	See section 10.2
[BSW00410] Compiler switches shall have defined values	See section 10.2
[BSW00411] Get version info keyword	See sections 8.3.1 and 0
[BSW00346] Basic set of module files	See Figure 1
[BSW158] Separation of configuration from implementation	See Figure 1, GPT183, GPT184
[BSW00314] Separation of interrupt frames and service routines	See Figure 1
[BSW00370] Separation of callback interface from API	See Figure 1
[BSW00348] Standard type header	See Figure 1
[BSW00353] Platform specific type header	See Figure 1
[BSW00361] Compiler specific language extension header	See Figure 1
[BSW00301] Limit imported information	See Figure 1
[BSW00302] Limit exported information	See Figure 1
[BSW00328] Avoid duplication of code	Not applicable (requirement for the implementer)
[BSW00312] Shared code shall be reentrant	See sections 8.3.4, 8.3.5, 8.3.6, 8.3.7, 8.3.8, 8.3.9, 8.3.11, 8.3.12
[BSW006] Platform independency	Not applicable (module is not above MCAL)
[BSW00357] Standard API return type	Not applicable (this type is not used within this SWS)
[BSW00377] Module specific API return types	Not applicable (this type is not used within this SWS)
[BSW00304] AUTOSAR integer data types	GPT174, sections 8.2 and 0
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable (no integer data types redefined in this specification)
[BSW00378] AUTOSAR boolean type	See section 0
[BSW00306] Avoid direct use of compiler and platform specific keywords	See Figure 1
[BSW00308] Definition of global data	Not applicable (requirement for the implementer)
[BSW00309] Global data with read-only constraint	See section 8.3.2
[BSW00371] Do not pass function pointers via API	Not applicable (no function pointers are passed via API in this SWS.)
[BSW00358] Return type of `init()` functions	See section 8.3.2
[BSW00414] Parameter of init function	GPT257

Requirement	***Satisfied by***
[BSW00376] Return type and parameters of main processing functions	Not applicable (no main processing function specified)
[BSW00359] Return type of callback functions	See section 8.6.4
[BSW00360] Parameters of callback functions	See section 8.6.4
[BSW00329] Avoidance of generic interfaces	Not applicable (no generic interfaces specified within this SWS)
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable (requirement for the implementer)
[BSW00331] Separation of error and status values	Not applicable (no status values specified within this SWS)
[BSW009] Module User Documentation	See this SWS
[BSW00401] Documentation of multiple instances of configuration parameters	See section 10.2
[BSW172] Compatibility and documentation of scheduling strategy	See section 8.3
[BSW010] Memory resource documentation	Not applicable (requirement for the implementer)
[BSW00333] Documentation of callback function context	See section 0
[BSW00374] Module vendor identification	GPT189
[BSW00379] Module identification	GPT189
[BSW003] Version identification	GPT189
[BSW00318] Format of module version numbers	GPT189
[BSW00321] Enumeration of module version numbers	Not applicable (requirement for the implementer)
[BSW00341] Microcontroller compatibility documentation	Not applicable (requirement for the implementer)
[BSW00334] Provision of XML file	Not applicable (specified by WP4.1.1.2)
[BSW00435] Header File Structure for the Basic Software Scheduler	See figure 1
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	See figure 1

Document: General Requirements on SPAL [8]

Requirement	***Satisfied by***
[BSW12263] Object code compatible configuration concept	GPT184
[BSW12056] Configuration of notification mechanism	GPT208
[BSW12267] Configuration of wakeup sources	GPT188, GPT235
[BSW12057] Driver module initialization	GPT006
[BSW12125] Initialization of hardware resources	GPT068
[BSW12163] Driver module deinitialization	GPT008
[BSW12461] Responsibility for register initialization	GPT205, GPT006
[BSW12462] Provide settings for register initialization	See section 0
[BSW12463] Combine and forward settings for register initialization	Not applicable (applies only for configurator)
[BSW12062] Selection of static configuration sets	GPT006
[BSW12068] MCAL initialization sequence	Not applicable (overall module initialization is not triggered by this module)
[BSW12069] Wake-up notification of ECU State Manager	GPT188
[BSW157] Notification mechanisms of drivers and handlers	GPT014, GPT015, GPT232
[BSW12155] Prototypes of callback functions	GPT232
[BSW12169] Control of operation mode	GPT151
[BSW12063] Raw value mode	GPT167, GPT168
[BSW12075] Use of application buffers	Not applicable (no random streaming capability
[BSW12129] Resetting of interrupt flags	GPT206
[BSW12064] Change of operation mode during running operation	Not applicable see section 8.3.10
[BSW12448] Behavior after development error detection	GPT178, GPT234, GPT097, GPT101, GPT084, GPT204
[BSW12067] Setting of wake-up conditions	The GPT HW module has no external interrupt source. The only wakeup condition is the internal timeout. The appropriate notification can be enabled/disabled. GPT014, GPT015, GPT232, GPT233
[BSW12077] Non-blocking implementation	Not applicable (requirement for the implementer)
[BSW12078] Runtime and memory efficiency	Not applicable (requirement for the implementer)
[BSW12092] Access to drivers	Not applicable (no handler or manager above)
[BSW12265] Configuration data shall be kept constant	Not applicable (requirement for the implementer)
[BSW12264] Specification of configuration items	GPT183, GPT184, GPT193, GPT235

Document: Requirements on GPT Driver [9]

Requirements (module specific)	Satisfied by
[BSW12328] GPT driver time unit	GPT055, GPT192
[BSW12404] Configuration of one-shot/continuous mode	GPT185, GPT186
[BSW12114] Configuration of timer clock source	GPT187
[BSW12460] Configuration of symbolic names for time values	Not applicable (Requirement for configuration tool
[BSW12116] GPT Deinitialization	GPT008, GPT161, GPT162
[BSW12117] Read timer value	GPT083, GPT010
[BSW12128] Start timer	GPT012, GPT060
[BSW12119] Stop timer	GPT013
[BSW12120] Provide notification	GPT232, GPT233
[BSW12121] Enable notification	GPT014
[BSW12122] Disable notification	GPT015
[BSW13601] Wakeup functionality	GPT184, GPT151, GPT159, GPT160
[BSW13602] Enable/Disable Wakeup	GPT159, GPT160
[BSW13603] Wake-up mode selection service	GPT151, GPT152, GPT153

7 Functional specification

7.1 General behavior

7.1.1 Functional overview

The GPT driver provides services for starting and stopping a functional timer instance (channel) within the hardware timer module. Individual timeout periods (one shot mode) as well as repeating timeout periods (continuous mode) can be generated. The user can configure, if a notification shall be invoked, when the requested timeout period has expired. Notifications can be enabled and disabled at runtime.

Both, the relative time elapsed since the last notification occurred (respectively the channel has been started) and the time remaining until the next notification will occur, can be queried.

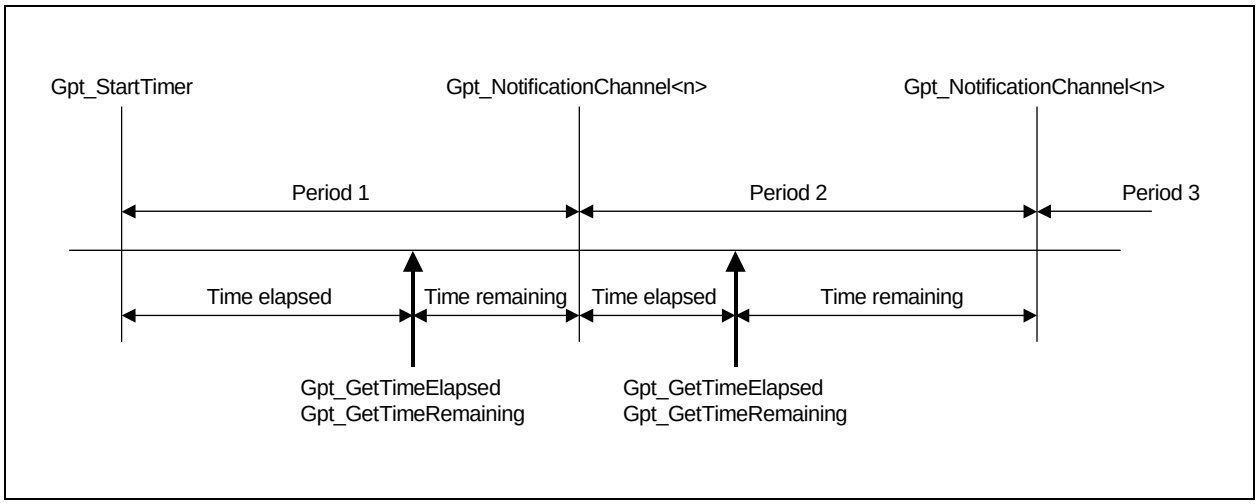

Figure 2: Querying the elapsed/remaining time

Note: The GPT driver only generates time bases, and does not serve as an event counter. This functionality is provided by another driver module (➔ICU driver, see [10]).

The GPT Driver can be used to wakeup the ECU, whenever a predefined timeout period has expired. A mode switching service is provided to switch the GPT Driver between normal operation and sleep mode.

The driver does not support timeout periods, which exceed the maximum value restricted by the clock source, prescaler and width of the timer register. The user must handle this.

7.1.2 Version checking

GPT256: The integration of incompatible files shall be avoided. Minimum implementation is the version check of the header file.

For included header files:
- GPT_AR_MAJOR_VERSION
- GPT_AR_MINOR_VERSION

shall be identical.

For the module internal c and h files:
- GPT_SW_MAJOR_VERSION
- GPT_SW_MINOR_VERSION
- GPT_AR_MAJOR_VERSION
- GPT_AR_MINOR_VERSION
- GPT_AR_PATCH_VERSION

shall be identical.

7.2 Error classification

GPT173: Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file Dem_IntErrId.h and included via Dem.h.

GPT174: Development error values are of type uint8.

GPT001: The following development errors shall be detectable by the GPT driver depending on its build version (development/production mode). This checking shall be statically configurable (on/off) for those errors that only can occur during development .

Type of error	*Relevance*	*Related error code*	*Value [hex]*
Operational function (getTimeElapsed/ GetTimeRemaining, start/stop timer, enable/disable notification, enable/disable wakeup, set mode, getVersionInfo) or Gpt_DeInit called prior to init function	Development	GPT_E_UNINIT	0x0A
Function startTimer or deInit called while timer is already running	Development	GPT_E_BUSY	0x0B
Operational function (getTimeElapsed/ getTimeRemaining) called prior to start timer function or after timer has been stopped or after timer has been expired in one-	Development	GPT_E_NOT_STARTED	0x0C

Type of error	Relevance	Related error code	Value [hex]
shot mode.			
Operational function (getTimeElapsed/ GetTimeRemaining, start/stop timer, enable/disable notification, enable/disable wakeup) called with invalid channel ID. Enable/disable wakeup called on a non-wakeup capable channel.	Development	GPT_E_PARAM_CHANNEL	0x14
Function startTimer called with invalid value	Development	GPT_E_PARAM_VALUE	0x15
Gpt_setMode called with invalid mode parameter	Development	GPT_E_PARAM_MODE	0x1F
No production errors assigned	Production	-	

Table 4: Error classification

GPT004: Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the GPT device specific implementation specification. The classification and enumeration shall be compatible to the errors listed above.

7.3 Error detection

GPT175: The detection of development errors is configurable (*STD_ON* / *STD_OFF*) at pre-compile time. The switch GPT_DEV_ERROR_DETECT (see chapter 10) shall activate or deactivate the detection of all development errors.

GPT176: If the GPT_DEV_ERROR_DETECT switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.2 and chapter 8.

GPT177: The detection of production code errors cannot be switched off.

GPT204: If development error detection is enabled for the GPT Driver, the following API parameter checking shall be performed according to the respective functions (see table below). The error shall be reported to the Development Error Tracer.

Function Gpt_	Criteria of detection	Related error code
GetVersionInfo	called prior to initialization	GPT_E_UNINIT
Init	ConfigPtr = NULL	GPT_E_PARAM_CONFIG
DeInit	called prior to initialization called while timer is running	GPT_E_UNINIT GPT_E_BUSY
GetTimeElapsed	called prior to initialization called prior to starting the timer channel called on a stopped channel	GPT_E_UNINIT GPT_E_NOT_STARTED GPT_E_NOT_STARTED

Function Gpt_	Criteria of detection	Related error code
	called on an expired ch. in oneshot mode channel out of range	GPT_E_NOT_STARTED GPT_E_PARAM_CHANNEL
GetTimeRemaining	called prior to initialization called prior to start. timer channel called on a stopped channel channel out of range	GPT_E_UNINIT GPT_E_NOT_STARTED GPT_E_NOT_STARTED GPT_E_PARAM_CHANNEL
StartTimer	called prior to initialization called while channel is already running Channel out of range Passed timer value out of range	GPT_E_UNINIT GPT_E_BUSY GPT_E_PARAM_CHANNEL GPT_E_PARAM_VALUE
StopTimer	called prior to initialization channel out of range	GPT_E_UNINIT GPT_E_PARAM_CHANNEL
EnableNotification	called prior to initialization channel out of range	GPT_E_UNINIT GPT_E_PARAM_CHANNEL
DisableNotification	called prior to initialization channel out of range	GPT_E_UNINIT GPT_E_PARAM_CHANNEL
SetMode	called prior to initialization called with invalid mode param.	GPT_E_UNINIT GPT_E_PARAM_MODE
EnableWakeup	called prior to initialization channel out of range called on a non-wakeup capable channel	GPT_E_UNINIT GPT_E_PARAM_CHANNEL GPT_E_PARAM_CHANNEL
DisableWakeup	called prior to initialization channel out of range called on a non-wakeup capable channel	GPT_E_UNINIT GPT_E_PARAM_CHANNEL GPT_E_PARAM_CHANNEL

Table 5: Error detection

7.4 Error notification

GPT178: Detected development errors shall be reported to the *Det_ReportError* service of the Development Error Tracer (DET[4]) if the preprocessor switch GPT_DEV_ERROR_DETECT is set (see chapter 10).

GPT179: Production errors shall be reported to Diagnostic Event Manager[6].

8 API specification

8.1 Imported types

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

8.1.1 Standard types

- Std_Types.h
  - Std_VersionInfoType

8.1.2 EcuM types

- EcuM_WakeupSourceType

8.2 Type Definitions

8.2.1 Gpt_ConfigType

Type:	`structure`
Range:	Implementation specific configuration data structure, see 10 for configurable parameters
Description:	This is the type of the data structure including the configuration set required for initializing the GPT timer unit

8.2.2 Gpt_ChannelType

Type:	`uint8...uint32`
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.

8.2.3 Gpt_ValueType

Type:	`uint8...uint32`
Range:	The range of this type is µC dependent (width of the timer register) and has to be described by the supplier.
Description:	Used for reading the current timer value/setting periodic timer values (in number of ticks) up to hours.

8.2.4 Gpt_ModeType

Type:	enumeration	
Range:	`GPT_MODE_NORMAL`	Normal operation mode of the GPT
	`GPT_MODE_SLEEP`	Operation for reduced power operation mode. In Wakeup mode only wakeup capable channels are available.
Description:	Allows the selection of different power modes.	

8.3 Function definitions

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

GPT055: All time units used within the API services of the GPT driver shall be of the unit ticks.

GPT192: To get times out of register values it is necessary to know the oscillator frequency, prescalers and so on. Since these settings are made in MCU and/or in other modules it is not possible to calculate such times.
Hence the conversions between time and ticks shall be part of an upper layer.

8.3.1 Gpt_GetVersionInfo

Service name:	Gpt_GetVersionInfo	
Syntax:	`void Gpt_GetVersionInfo ( Std_VersionInfoType* versioninfo )`	
Service ID [hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non reentrant	
Parameters (in):	None	--
Parameters (out):	`versioninfo`	Pointer to where to store the version information of this module.
Return value:	None	--
Description:	**GPT181:** This service returns the version information of this module. The version information includes: – Module Id – Vendor Id – Vendor specific version numbers (BSW00407). **GPT221:** If development error detection is enabled and `Gpt_GetVersionInfo` is called prior to initializing the driver, the development error code `GPT_E_UNINIT` will be reported. Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.	
Caveats:	None	
Configuration:	**GPT182:** This function shall be pre compile time configurable `On/Off` by the configuration parameter: GPT_VERSION_INFO_API (see 10.2.6)	

8.3.2 Gpt_Init

Service name:	Gpt_Init

Syntax:	`void Gpt_Init`
	`(`
	`  const Gpt_ConfigType *configPtr`
	`)`

Service ID [hex]:	0x01
Sync/Async:	Synchronous
Reentrancy:	Non reentrant
Parameters (in):	`configPtr` Pointer to a selected configuration structure
Parameters (out):	None
Return value:	None
Description:	**GPT006:** Service for initializing the hardware timer module according to a configuration set referenced by `ConfigPtr.` **GPT107:** By default, notifications shall be disabled after module initialization . **GPT068:** This function shall only initialize the configured resources. Resources that are not configured in the configuration file shall not be touched. **GPT205:** The following rules regarding initialization of controller registers shall apply to the GPT Driver implementation: [1] If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register [2] If the register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver [3] If the register can affect several hardware modules and if it is not an I/O register it shall be initialized by the MCU driver [4] One-time writable registers that require initialization directly after reset shall be initialized by the startup code [5] All other registers shall be initialized by the startup code **GPT148:** If this function is called, while the driver has already been initialized all channels are stopped and the driver is reinitialized. GPT258: The wakeup –interrupt invocation of a channel shall be disabled after the call of this service.
Caveats:	None
Configuration:	See chapter 10

8.3.3 Gpt_DeInit

Service name:	Gpt_DeInit

Syntax:	`void Gpt_DeInit`
	`(`
	`  void`
	`)`

Service ID [hex]:	0x02
Sync/Async:	Synchronous
Reentrancy:	Non reentrant
Parameters (in):	None

Parameters (out):	None
Return value:	None
Description:	**GPT008**: Service for deinitializing all hardware timer channels to their power on reset state . **GPT088:** The state of the peripheral after DeInit shall be the same as after power on reset. It's the responsibility of the hardware design that the state does not lead to undefined activities in the µC **GPT105:** The service shall disable notifications. **GPT234:** If development error detection is enabled and `Gpt_DeInit` is called, while the timer is running, the driver shall report `GPT_E_BUSY` and the desired deinitialization functionality shall be left without any action . **GPT220:** If development error detection is enabled and `Gpt_DeInit` is called prior to initializing the driver, the development error code `GPT_E_UNINIT` will be raised. **GPT161:** After the call of this service, the state of the peripherals used by configuration shall be the same as after power on reset. Values of registers, which are not writable, are excluded. **GPT162:** The service influences only the peripherals, which are allocated by static configuration and the runtime configuration set passed by the previous call of Gpt_Init().
Caveats:	None
Configuration:	**GPT194:** This function shall be pre compile time configurable `On/Off` by the configuration parameter: GPT_DEINIT_API (see 10.2.6)

8.3.4 Gpt_GetTimeElapsed

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 **GPT210:** If development error detection is enabled and `Gpt_GetTimeElapsed` is called with an invalid `channel` parameter, the development error code `GPT_E_PARAM_CHANNEL` will be raised. In this case, value "0" is returned.
Parameters (out):	None	
Return value:	`Gpt_ValueType`	**GPT167:** Elapsed timer value (in number of ticks)
Description:	**GPT010:** Service for querying the time already elapsed. In one shot mode, this is the value relative to the point in time, the channel has been started with Gpt_StartTimer (calculated by the normal operation function by subtracting the current minus the initial timer value and returning the absolute value). In	

	continuous mode, the function returns the timer value relative to the last timeout / the start of the channel . **GPT222:** If development error detection is enabled and `Gpt_GetTimeElapsed` is called prior to initializing the driver, the development error code `GPT_E_UNINIT` will be raised. In this case, value “0” is returned. **GPT097:** When calling this function prior to starting the specified timer channel, value “0” is returned. If development error detection is enabled, `GPT_E_NOT_STARTED` shall be raised (see section 7.2). **GPT267:** When Calling this function on a stopped timer channel (channel has been initialized and started before), value “0” is returned. If development error detection is enabled, `GPT_E_NOT_STARTED` shall be raised (see section 7.2) The rationale is to have the same behaviour for a stopped channel like a never started channel. **GPT100:** When calling this function on a channel configured for one shot mode, after the timeout period has already expired, value “0” is returned. If development error detection is enabled, GPT_E_NOT_STARTED shall be raised in (see section 7.2).
Caveats:	**GPT113:** Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel .
Configuration:	**GPT195:** This function shall be pre compile time configurable `On/Off` by the configuration parameter: GPT_TIME_ELAPSED_API (see 10.2.6)

8.3.5 Gpt_GetTimeRemaining

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 **GPT211:** If development error detection is enabled and `Gpt_GetTimeRemaining` is called with an invalid `channel` parameter, the development error code `GPT_E_PARAM_CHANNEL` will be raised. In this case, value “0” is returned.
Parameters (out):	None	
Return value:	`Gpt_ValueType`	**GPT168:** Remaining timer value (in number of ticks)
Description:	**GPT083:** This function returns the timer value remaining until the next timeout period will expire (calculated by the normal operation function by subtracting the timeout minus the current timer value and returning the absolute value) . **GPT223:** If development error detection is enabled and `Gpt_GetTimeRemaining` is called prior to initializing the driver, the development error code `GPT_E_UNINIT` will be raised. In this case, value “0” is returned.	

	GPT101: When calling this function prior to starting the specified timer channel, value “0” is returned. If development error detection is enabled, `GPT_E_NOT_STARTED` shall be raised (see section 7.2) **GPT268:** When Calling this function on a stopped timer channel (channel has been initialized and started before), value “0” is returned. If development error detection is enabled, `GPT_E_NOT_STARTED` shall be raised (see section 7.2) The rationale is to have the same behaviour for a stopped channel like a never started channel. **GPT102:** When calling this function on a channel configured for one shot mode, after the timeout period has already expired, value `“0”` is returned. If development error detection is enabled, GPT_E_NOT_STARTED shall be raised in (see section 7.2).
Caveats:	**GPT114:** Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel.
Configuration:	**GPT196:** This function shall be pre compile time configurable `On/Off` by the configuration parameter: GPT_TIME_REMAINING_API (see 10.2.6)

The table below (Table 6) sumarizes the return and error values for the functions Gpt_GetTimeElapsed and Gpt_GetTimeRemaining, when called in different points of time.

Example how to read the table: When Gpt_GetTimeRemaining is called on a channel in continuous mode after Gpt_StopTimer has been called before, value “0” is returned and in case of enabled development error detection, GPT_E_NOT_STARTED is reported additionally. This is decribed in GPT268.

Note that a continuous channel never expires and thus no return/error values are defined for that special case.

		Gpt_Init(		Gpt_StartTimer()		Gpt_StopTimer() (before expiry)		Timer expired	
GetTimeElapsed									
one-shot	Return “0” UNINIT (GPT222)		Return "0" NOT_STARTED (GPT097)		Return current register value - initial timer value (GPT010)		Return "0" NOT_STARTED (GPT267)		Return "0" NOT_STARTED (GPT100)
continuous									-
GetTimeRemaining									
one-shot	Return “0” UNINIT (GPT223)		Return "0" NOT_STARTED (GPT101)		Return timeout period - current register value (GPT083)		Return "0" NOT_STARTED (GPT268)		Return "0" NOT_STARTED (GPT102)
continuous									-

Table 6: Overview on return and error values of Gpt_GetTimeElapsed/Remaining

8.3.6 Gpt_StartTimer

Service name:	Gpt_StartTimer	
Syntax:	`void Gpt_StartTimer ( Gpt_ChannelType channel Gpt_ValueType value )`	
Service ID [hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`channel`	Numeric identifier of the GPT channel **GPT212:** If development error detection is enabled and `Gpt_StartTimer` is called with an invalid `channel` parameter, the development error code `GPT_E_PARAM_CHANNEL` will be raised.

	`value`	**GPT060:** timeout period (in number of ticks) after a notification shall occur. **GPT218:** If development error detection is enabled and `Gpt_StartTimer` is called with an invalid value, the development error code `GPT_E_PARAM_VALUE` will be raised.
Parameters (out):	None	
Return value:	None	
Description:	**GPT012:** Service for starting the selected timer channel with a defined timeout period. After this timeout period, a notification can be invoked (if enabled). **GPT224:** If development error detection is enabled and `Gpt_StartTimer` is called prior to initializing the driver, the development error code `GPT_E_UNINIT` will be raised. **GPT084:** Calling `Gpt_StartTimer` on a channel, which is already started and still running, will raise a development error (`GPT_E_BUSY`) and the start service shall be left without any action. **GPT085:** In production mode no error is generated. The rational is that it adds no additional functionality to the driver . In this case the timer will be restarted with the timeout value, given as a parameter to the service.	
Caveats:	**GPT115:** Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel.	
Configuration:	**None**	

8.3.7 Gpt_StopTimer

Service name:	Gpt_StopTimer	

Syntax:	`void Gpt_StopTimer`
	`(`
	`  Gpt_ChannelType channel`
	`)`	

Service ID [hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`channel`	Numeric identifier of the GPT channel **GPT213:** If development error detection is enabled and `Gpt_StopTimer` is called with an invalid `channel` parameter, the development error code `GPT_E_PARAM_CHANNEL` will be raised.
Parameters (out):	None	
Return value:	None	
Description:	**GPT013:** Service for stopping the selected timer channel . **GPT099:** Stopping a timer channel, not been started before will not return a development error . **GPT225:** If development error detection is enabled and `Gpt_StopTimer` is called prior to initializing the driver, the development error code `GPT_E_UNINIT` will be raised.	

	GPT103: Timer channels configured in one shot mode are stopped automatically, when the timeout period has expired.
Caveats:	**GPT116:** Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel .
Configuration:	**None**

8.3.8 Gpt_EnableNotification

Service name:	Gpt_EnableNotification	
Syntax:	`void Gpt_EnableNotification ( Gpt_ChannelType channel )`	
Service ID [hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`channel`	Numeric identifier of the GPT channel **GPT214:** If development error detection is enabled and `Gpt_EnableNotification` is called with an invalid `channel` parameter, the development error code `GPT_E_PARAM_CHANNEL` will be raised.
Parameters (out):	None	
Return value:	None	
Description:	**GPT014:** Service for enabling the notification for a channel during runtime. **GPT091:** This function can be called, while the timer is already running. **GPT226:** If development error detection is enabled and `Gpt_EnableNotification` is called prior to initializing the driver, the development error code `GPT_E_UNINIT` will be raised.	
Caveats:	**GPT117:** Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel .	
Configuration:	**GPT199:** This function shall be pre compile time configurable `On/Off` by the configuration parameter: GPT_ENABLE_DISABLE_NOTIFICATION_API (see 10.2.6)	

8.3.9 Gpt_DisableNotification

Service name:	Gpt_DisableNotification	
Syntax:	`void Gpt_DisableNotification ( Gpt_ChannelType channel )`	
Service ID [hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`channel`	Numeric identifier of the GPT channel **GPT217:** If development error detection is enabled and `Gpt_DisableNotification` is called with an invalid `channel` parameter, the development error code `GPT_E_PARAM_CHANNEL` will be raised.

Parameters (out):	None
Return value:	None
Description:	**GPT015:** Service for disabling the notification for a channel during runtime.. **GPT092:** This function can be called, while the timer is already running . **GPT227:** If development error detection is enabled and `Gpt_DisableNotification` is called prior to initializing the driver, the development error code `GPT_E_UNINIT` will be raised.
Caveats:	**GPT118:** Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel .
Configuration:	**GPT200:** This function shall be pre compile time configurable `On/Off` by the configuration parameter: GPT_ENABLE_DISABLE_NOTIFICATION_API (see 10.2.6)

8.3.10 Gpt_SetMode

Service name:	Gpt_SetMode	
Syntax:	`void Gpt_SetMode ( Gpt_ModeType mode )`	
Service ID [hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Non reentrant	
Parameters (in):	`mode`	**`GPT_MODE_NORMAL`:** Normal operation mode of the GPT enabled. **`GPT_MODE_SLEEP`:** Operation for reduced power operation mode. In Wakeup mode only wakeup capable channels are capable of generating interrupts. See also Gpt_ModeType **GPT231:** If development error detection is enabled and `Gpt_SetMode` is called with an invalid `mode` parameter, the development error code `GPT_E_PARAM_MODE` will be raised.
Parameters (out):	None	--
Return value:	None	
Description:	**GPT151:** Service for GPT mode selection. This service shall set the operation mode to the given mode parameter . **GPT255:** This service is only feasible, if GPT_REPORT_WAKEUP_SOURCE is statically configured available. **GPT152:** In `GPT_MODE_NORMAL` mode all notifications are available as configured and selected by the `Gpt_DisableNotification()` and `Gpt_EnableNotification()` services before or after the call of Gpt_SetMode . **GPT153:** In `GPT_MODE_SLEEP` mode interrupts are only enabled for those channels which are configured as wakeup capable and which are not disabled via service Gpt_DisableWakeup. All other interrupts are disabled and must not lead to an exit from the reduced power mode state (e.g. idle, halt) of the MCU if the wakeup timer expires.	

	GPT164: In GPT_MODE_SLEEP, all non-wakeup capable timer channels shall be stopped. Only those channels, which can serve as a wakeup source are running. **GPT165:** When waking up form sleep mode and the mode is set back to GPT_MODE_NORMAL, the timer channels which have been stopped by entering the sleep mode are not restarted automatically. **GPT228:** If development error detection is enabled and `Gpt_SetMode` is called prior to initializing the driver, the development error code `GPT_E_UNINIT` will be raised.
Caveats:	This service influences the functionality of the GPT channels. Therefore the mode switching of the module shall be compatible to the overall state of the ECU.
Configuration:	GPT_REPORT_WAKEUP_SOURCE **GPT201:** This function shall be pre compile time configurable `On/Off` by the configuration parameter: GPT_WAKEUP_FUNCTIONALITY _API (see 10.2.6)

8.3.11 Gpt_DisableWakeup

Service name:	Gpt_DisableWakeup	
Syntax:	`void Gpt_DisableWakeup ( Gpt_ChannelType channel )`	
Service ID [hex]:	0x0A	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`channel`	Numeric identifier of the GPT channel **GPT215**: If development error detection is enabled and `Gpt_DisableWakeup` is called with an invalid `channel` parameter or on a non-wakeup capable channel, the development error code `GPT_E_PARAM_CHANNEL` will be raised.
Parameters (out):	None	--
Return value:	None	
Description:	**GPT159:** This service shall disable the wakeup interrupt invocation of a single GPT channel. **GPT157:** This service is only feasible, if GPT_REPORT_WAKEUP_SOURCE is statically configured available. **GPT229:** If development error detection is enabled and `Gpt_DisableWakeup` is called prior to initializing the driver, the development error code `GPT_E_UNINIT` will be raised.	
Caveats:	**GPT155:** Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel.	
Configuration:	GPT_REPORT_WAKEUP_SOURCE **GPT202:** This function shall be pre compile time configurable `On/Off` by the configuration parameter: GPT_WAKEUP_FUNCTIONALITY_API (see 10.2.6)	

8.3.12 Gpt_EnableWakeup

Service name:	Gpt_EnableWakeup	
Syntax:	`void Gpt_EnableWakeup ( Gpt_ChannelType channel )`	
Service ID [hex]:	0x0B	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	`channel`	Numeric identifier of the GPT channel **GPT216:** If development error detection is enabled and `Gpt_EnableWakeup` is called with an invalid `channel` parameter or on a non-wakeup capable channel, the development error code `GPT_E_PARAM_CHANNEL` will be raised.
Parameters (out):	None	--
Return value:	None	
Description:	**GPT160:** This service shall re-enable the wakeup interrupt invocation of a single GPT channel. **GPT158:** This service is only feasible, if GPT_REPORT_WAKEUP_SOURCE is statically configured available. **GPT230:** If development error detection is enabled and `Gpt_EnableWakeup` is called prior to initializing the driver, the development error code `GPT_E_UNINIT` will be raised.	
Caveats:	**GPT156:** Usage of re-entrant capability is only allowed if the callers take care that there is no simultaneous usage of the same channel.	
Configuration:	GPT_REPORT_WAKEUP_SOURCE **GPT203:** This function shall be pre compile time configurable `On/Off` by the configuration parameter: GPT_WAKEUP_FUNCTIONALITY_API (see 10.2.6)	

8.4 Call-back Notifications

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

8.5 Scheduled functions

None.

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

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

None.

8.6.2 Optional Interfaces

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

API function	Det_ReportError
Module	Development Error Tracer (DET, see [4])
Description	Development error notification
Configuration parameter (description see chapter 10)	GPT_DEV_ERROR_DETECT

API function	`Dem_ReportErrorStatus`
Module	Diagnostic Event Manager (DEM, see [6])
Description	**GPT245:** Reporting of production relevant error status

API function	EcuM_SetWakeupEvent
Module	ECU State Manager [7]
Description	This service shall be called if ▪ the static configuration parameter GPT_REPORT_WAKEUP_SOURCE is set “STD_ON” ▪ and the module is in mode GPT_MODE_SLEEP ▪ and a wakeup event occurs on a wakeupc capable GPT channel.
Configuration parameter (description see chapter 10)	GPT_REPORT_WAKEUP_SOURCE

8.6.3 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.

GPT206: The ISR´s, providing the timeout period events, shall be responsible for resetting the interrupt flags (if needed by hardware) and calling the according notification function.
GPT207: The callback notifications shall be configurable as function pointers within the initialization data structure (Gpt_ConfigType).

GPT208: If a callback notification is configured as null pointer, no callback shall be executed.

GPT209: Each channel shall provide its own notification if configured

8.6.4 Gpt_Notification

Service name:	Gpt_Notification_<channel>

Syntax:	`void Gpt_Notification_<channel>`
	`(`
	`  void`
	`)`

Service ID [hex]:	0x0C
Sync/Async:	Synchronous
Reentrancy:	GPT user implementation dependant
Parameters (in):	None
Parameters (out):	None
Return value:	None
Description:	**GPT232**Notification prototype for the notification callback function. It has to be implemented by the user. **GPT233**The GPT Driver shall provide a notification per channel that is called whenever the defined time period is over. **GPT093:** When disabled, no notification will be sent. When re-enabled again, the user will not be notified of events, occurred while notifications have been disabled (see chapter 9.4). **GPT087:** To avoid parameter values and to improve runtime efficiency, for every channel, a particular notification callback has to be declared . **GPT086:**The callback notifications shall be configurable as pointers to user defined functions within the configuration structure .
Caveats:	None
Configuration:	For all available channels, callback functions have to be declared by the configuration tool (see chapter 10).

9 Sequence diagrams

All functions except Gpt_Init, Gpt_DeInit, Gpt_GetVersionInfo and Gpt_SetMode are performed synchronously and re-entrant.

9.1 Gpt_Init

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

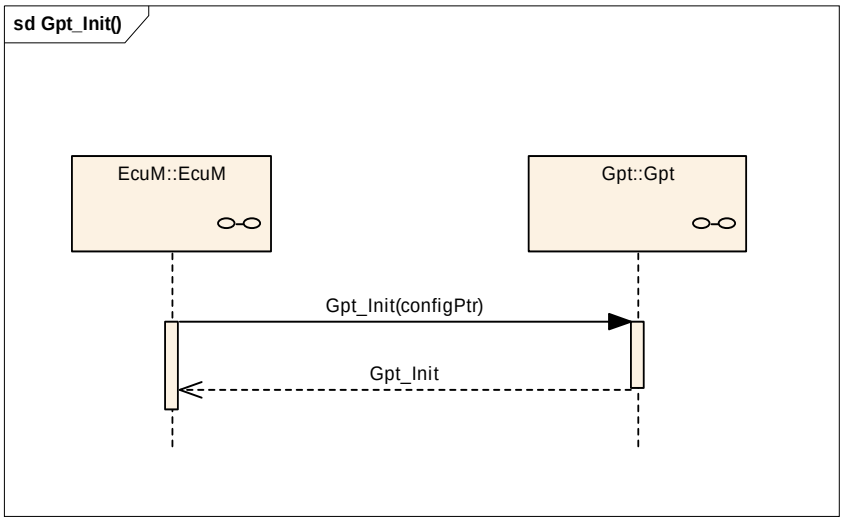

Figure 3: Sequence Diagram - Gpt_Init

9.2 GPT continuous mode

Channel 2 is configured as “Continuous Mode”

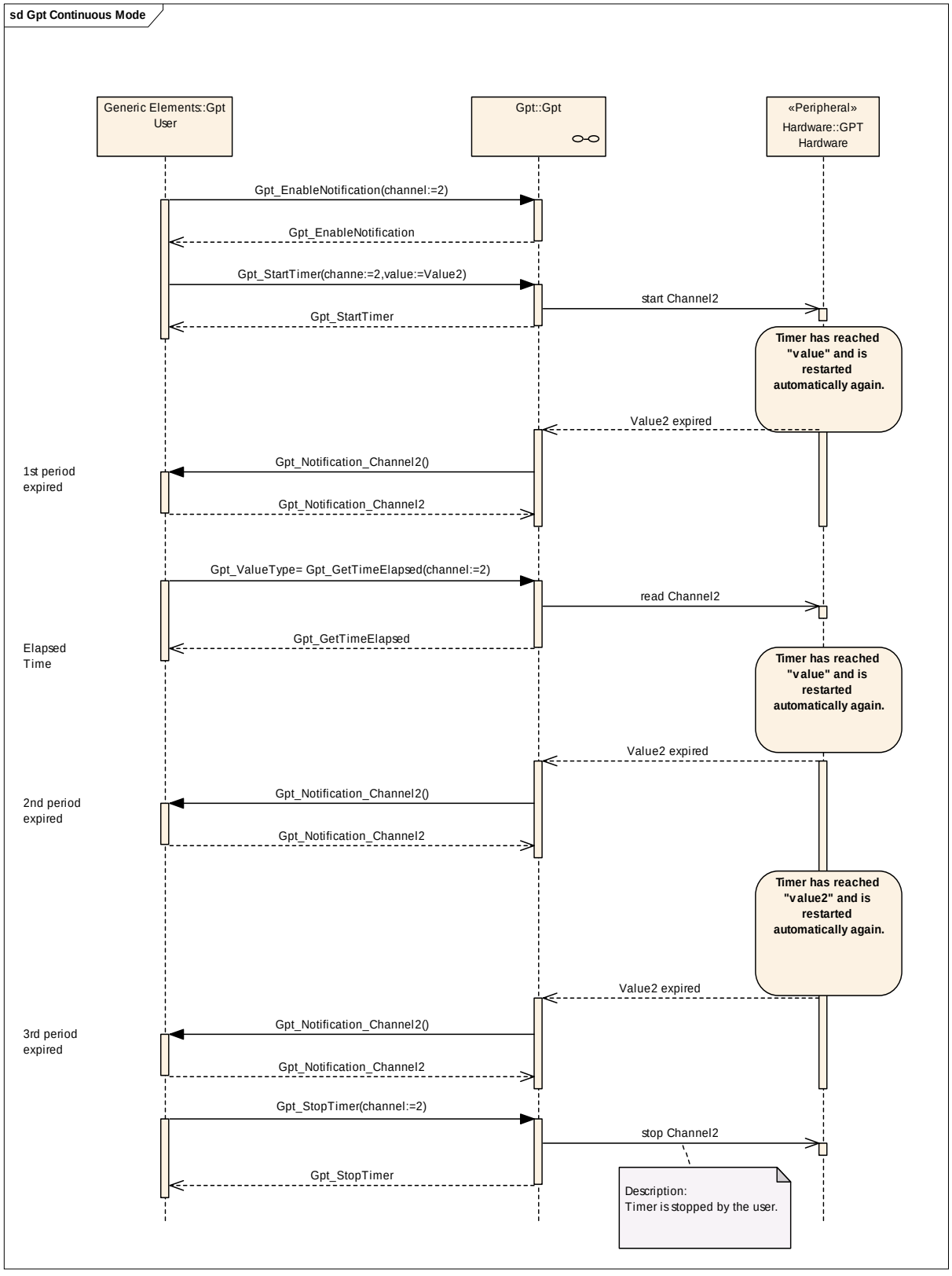

Figure 4: Sequence Diagram - GPT continuous mode

9.3 GPT one shot mode

Channel 1 is configured for “One shot Mode”

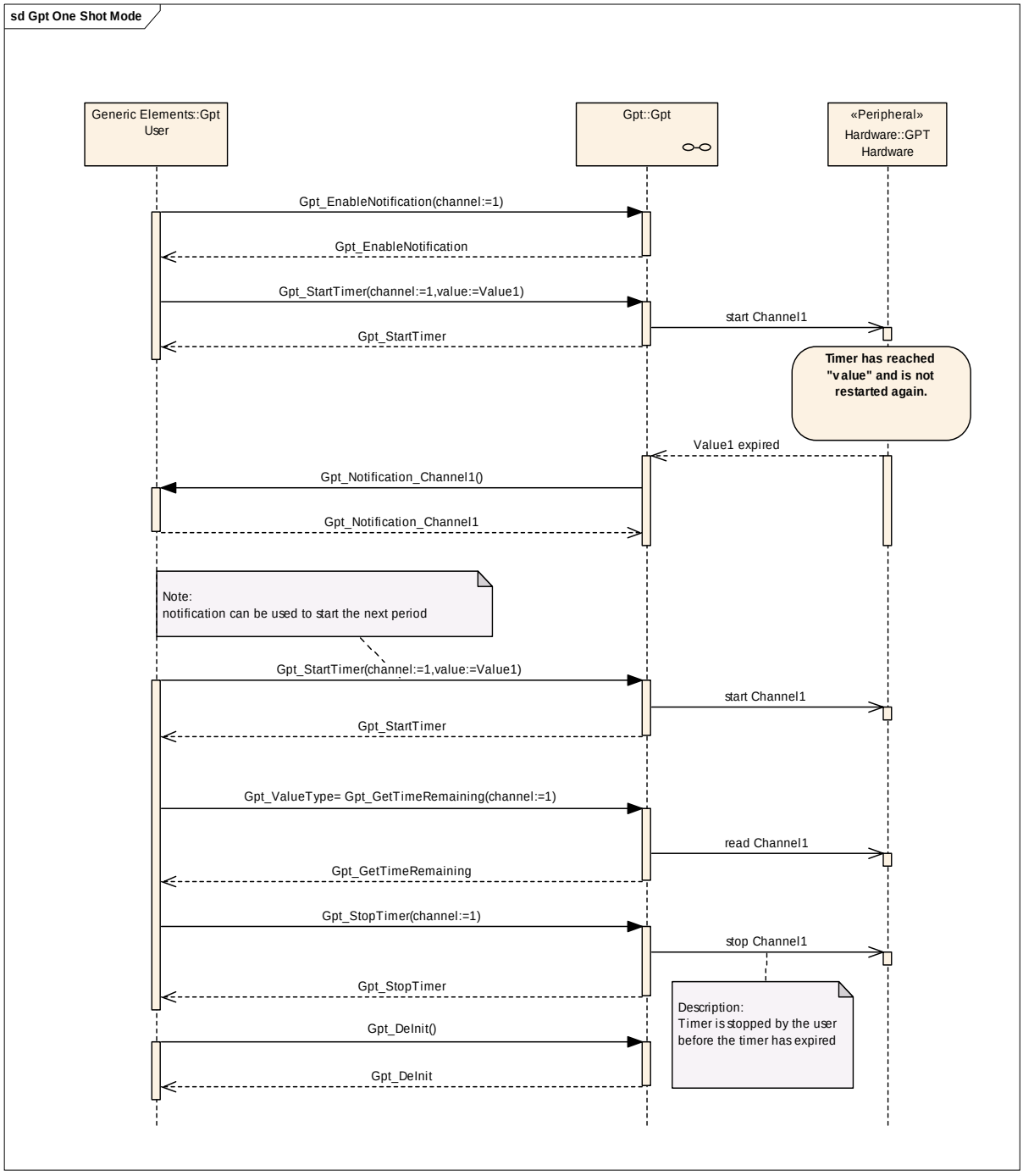

Figure 5: Sequence Diagram - GPT one shot mode

9.4 Disable/Re-enable Notifications

The sequence diagram shown in this chapter explains the behavior of the driver, when the timeout notification is disabled, while the timer is still running.

When disabled the user will not be informed, when timeout period 2 has expired. This notification is discarded and not made up again, when the timeout notification is re-enabled.

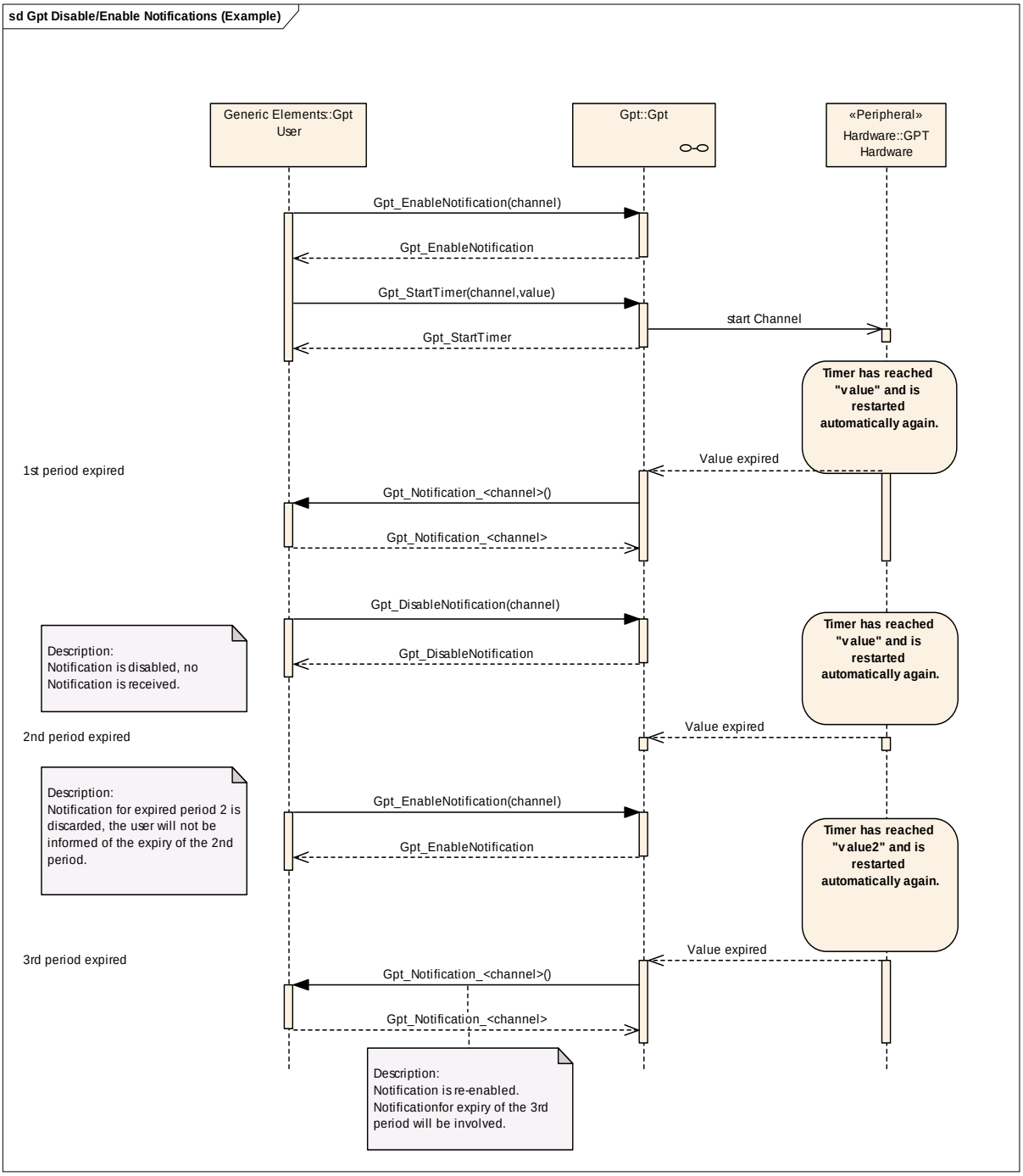

Figure 6: Sequence Diagram - Disable/Re-enable Notifications

9.5 Wakeup

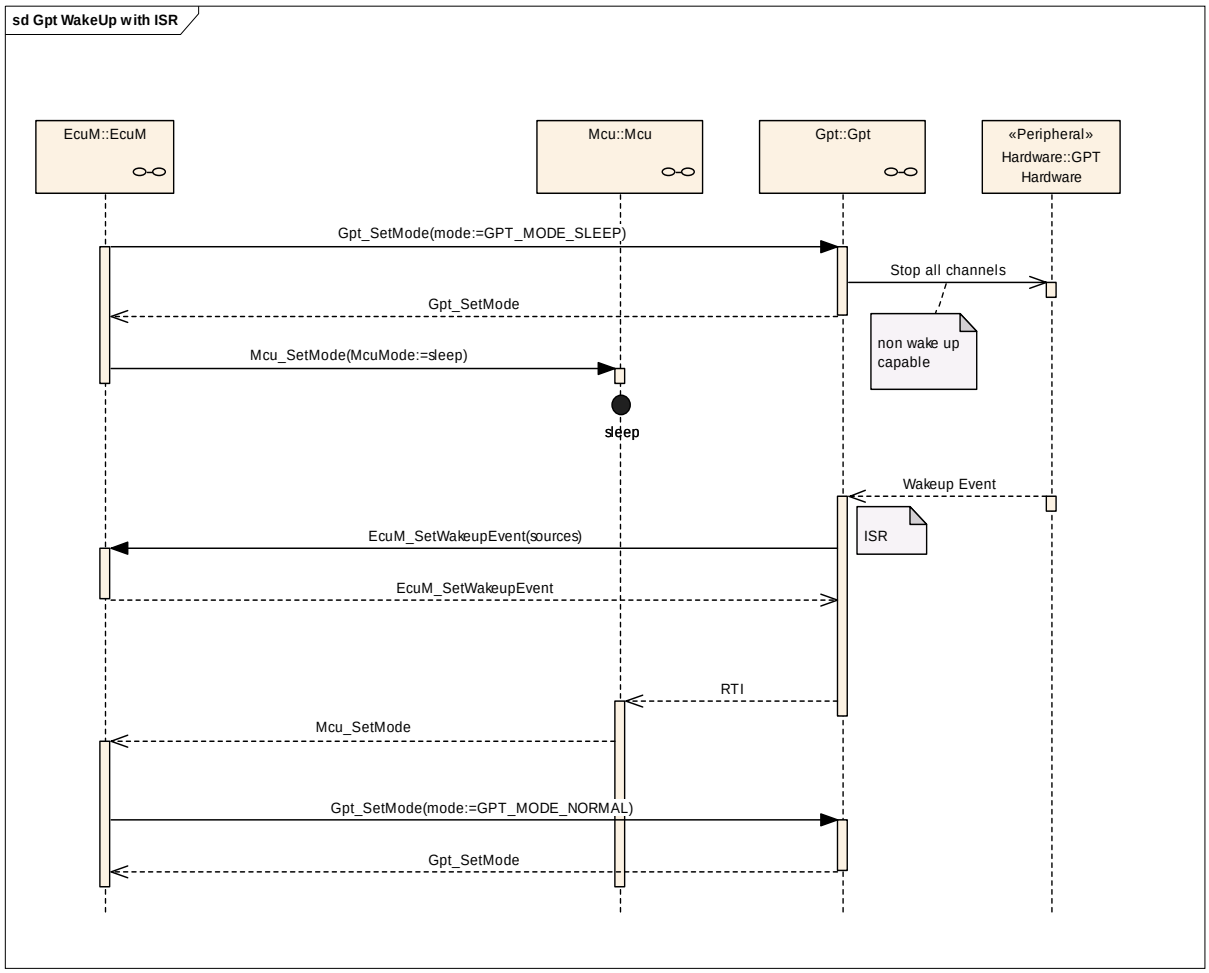

Figure 7: Sequence Diagram – Wakeup with ISR usage

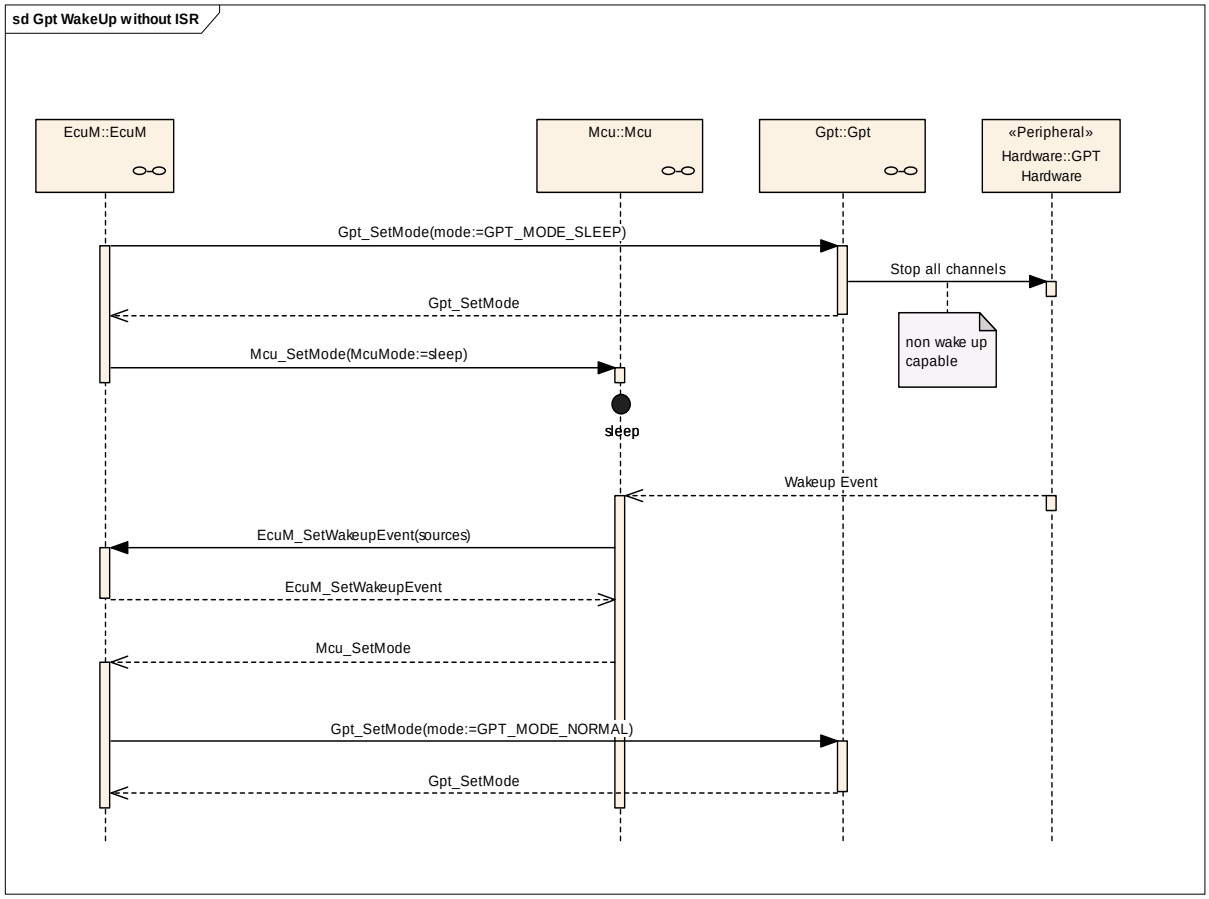

Figure 8: Sequence Diagram – Wakeup without ISR usage

GPT127: If supported by hardware and enabled, an internal hardware timer can serve as a wakeup source .

GPT128: When sleep mode is requested, the ECU State Manager[7] calls Gpt_SetMode with mode parameter “GPT_MODE_SLEEP” and prepares the GPT for sleep mode. With calling Mcu_SetMode, the MCU Driver[11] is then putting the controller into SLEEP mode . The controller is halted.

GPT129: The SLEEP State is exited, when the timer wakeup event occurs (i.e. when the timer expired). Depending on the implementation and the used controller, an ISR is invoked, once the timer wakeup event occurs. Within the ISR, the ECU State Manager[7] is informed about the wakeup event by calling EcuM_SetWakeupEvent (see [7]) with a parameter identifying the GPT as the wakeup source. The ISR returns and the program execution continuous at the MCU Driver[11]. Finally, the `MCU_SetMode` service of the MCU driver[11] returns and the ECU State Manager[7] regains control.

GPT150: By calling Gpt_SetMode with mode parameter “GPT_MODE_NORMAL”, the ECU State Manager[7] switches the GPT back to normal operation mode.

GPT132: Due to the fact, that only MCU-internal timers are in scope, no unintended events are expected and thus no wakeup validation is required.

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 0 specifies published information of the module GPT

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [2].
- AUTOSAR ECU Configuration Specification [5]
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Variants

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.1.4 Specification template for configuration parameters

Pre-compile time - specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* - the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* - the configuration parameter shall be of configuration class *Post Build* and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

10.2 Containers and configuration parameters

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

10.2.1 Variants

Variant PC: This variant is limited to pre-compile-configuration parameters only.

Variant PB: This variant allows a mix of pre-compile time and post-build multiple selectable configurable parameters.

GPT257: The initialization function of this module shall always have a pointer as a parameter, even though for Variant PC no configuration set shall be given. Instead a NULL pointer shall be passed to the initialization function. This means that, in contradiction to BSW00414 only one interface for initialization shall be implemented and it shall not depend on the modules configuration, which interface the calling software module shall use.

GPT270: Within one container it shall not be possible to mix parameters assigned to different configuration classes.

10.2.2 Overview on configuration container structure

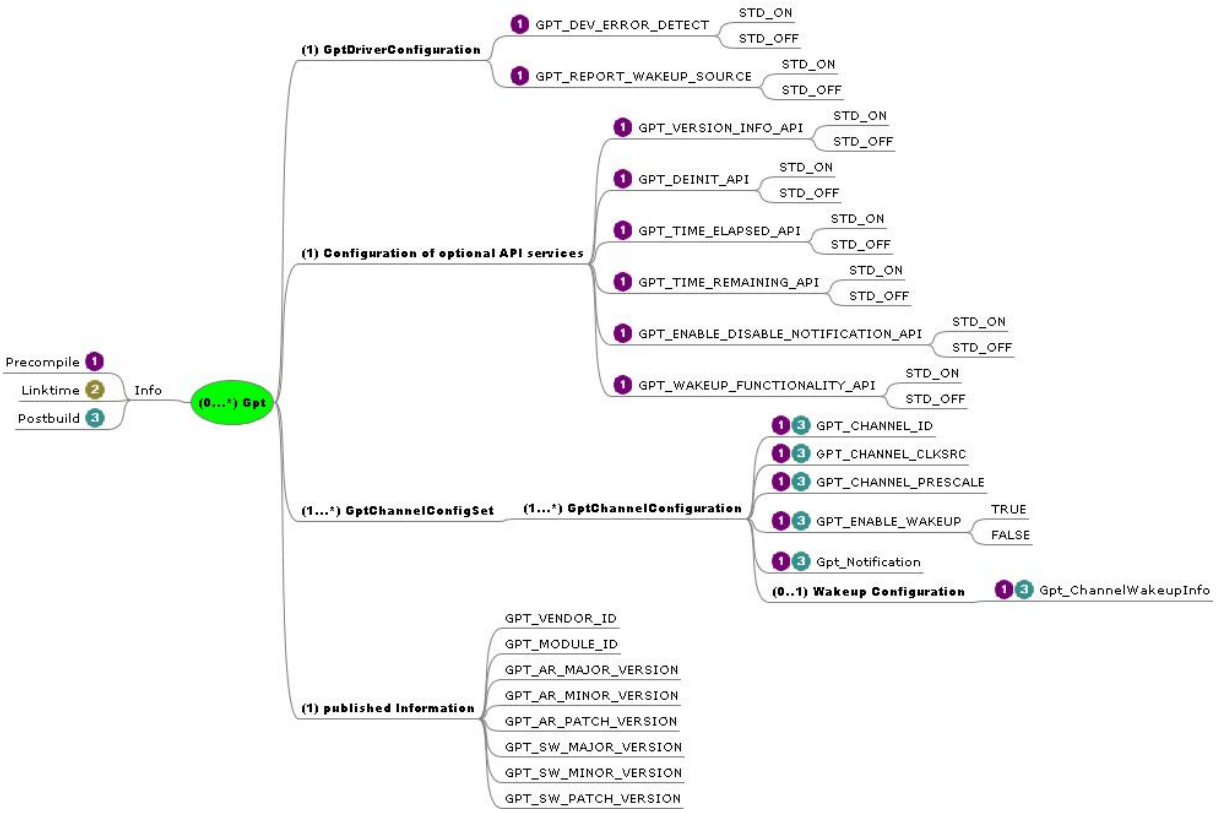

Figure 9: Overview on configuration container structure

10.2.3 GptDriverConfiguration

SWS Item	**GPT183:**
Container Name	GptDriverConfiguration
Description	This container contains the module-wide configuration (parameters) of the GPT Driver.
Configuration Parameters	

Name	GPT_DEV_ERROR_DETECT		
Description	Enables/Disables development error detection		
Type	#define		
Unit	--		
Range	STD_ON	Enable development error detection	
	STD_OFF	Disable development error detection	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	--		

Name	GPT_REPORT_WAKEUP_SOURCE		
Description	Enables/Disables wakeup source reporting		
Type	#define		
Unit	--		
Range	STD_ON	--	
	STD_OFF	--	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

10.2.4 GptChannelConfigSet

SWS Item	**GPT269:**
Container Name	GptChannelConfigSet
Description	This container aggregates different channel configuration sets of the GPT Driver.
Configuration Parameters	

None.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
GptChannelConfiguration	1…*	Module

10.2.5 GptChannelConfiguration

GPT236: It shall not be possible to add or remove GPT channels dynamically at runtime.

SWS Item	**GPT184:**
Container Name	GptChannelConfiguration
Description	This container contains the channel-wide configuration (parameters) of the GPT Driver.
Configuration Parameters	

Name	GPT_CHANNEL_ID		
Description	Channel Id of the numeric GPT channel. This value will be assigned to the symbolic name derived of the GptChannel container short name.		
Type	implementation specific		
Unit	--		
Range	--		
	--		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	instance		
Dependency	--		

Name	GPT_CHANNEL_MODE		
Description	Specifies the behavior of the timer channel after the timeout has expired.		
Type	implementation specific		
Unit	--		
Range	GPT_MODE_ONESHOT	**GPT185:** Timer channel stops after reaching its end value.	
	GPT_MODE_CONTINUOUS	**GPT186:** Timer channel is restarted automatically after reaching its end value.	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	instance		
Dependency	--		

Name	GPT_CHANNEL_CLKSRC		
Description	**GPT187:** The GPT module specific clock input for the timer unit can statically be configured and allows selecting different clock sources (external clock, internal GPT specific clock) per channel.		
Type	implementation specific		
Unit	--		
Range	Implementation specific	--	
	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	instance		
Dependency	Optional parameter, only available if supported by hardware **GPT240:** For dependencies to other modules see section 5		

Name	GPT_CHANNEL_PRESCALE

Description	GPT module specific prescaler factor per channel.		
Type	Gpt_PrescaleType		
Unit	--		
Range	Implementation specific	--	
	--	--	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	instance		
Dependency	Optional parameter, only available if supported by hardware **GPT241:** For dependencies to other modules see section 5 **GPT242:** See Figure 10		

Name	GPT_ENABLE_WAKEUP		
Description	**GPT188:**Enables wakeup feature of CPU for a channel when timeout period expires. This might be different to enabling the notification depending on hardware capabilities.		
Type	boolean		
Unit	--		
Range	TRUE	Wakeup capability on this channel is enabled	
	FALSE	Wakeup capability on this channel is disabled	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	instance		
Dependency	--		

Name	Gpt_Notification		
Description	Function pointer to callback function.		
Type	Function pointer.		
Unit	--		
Range	--	--	
	--	--	
Configuration Class	***Pre-compile***	X	All variants
	Link time	--	--
	Post Build	M	Variante PB
Scope	Instance		
Dependency	--		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
Wakeup configuration	0…1	Instance

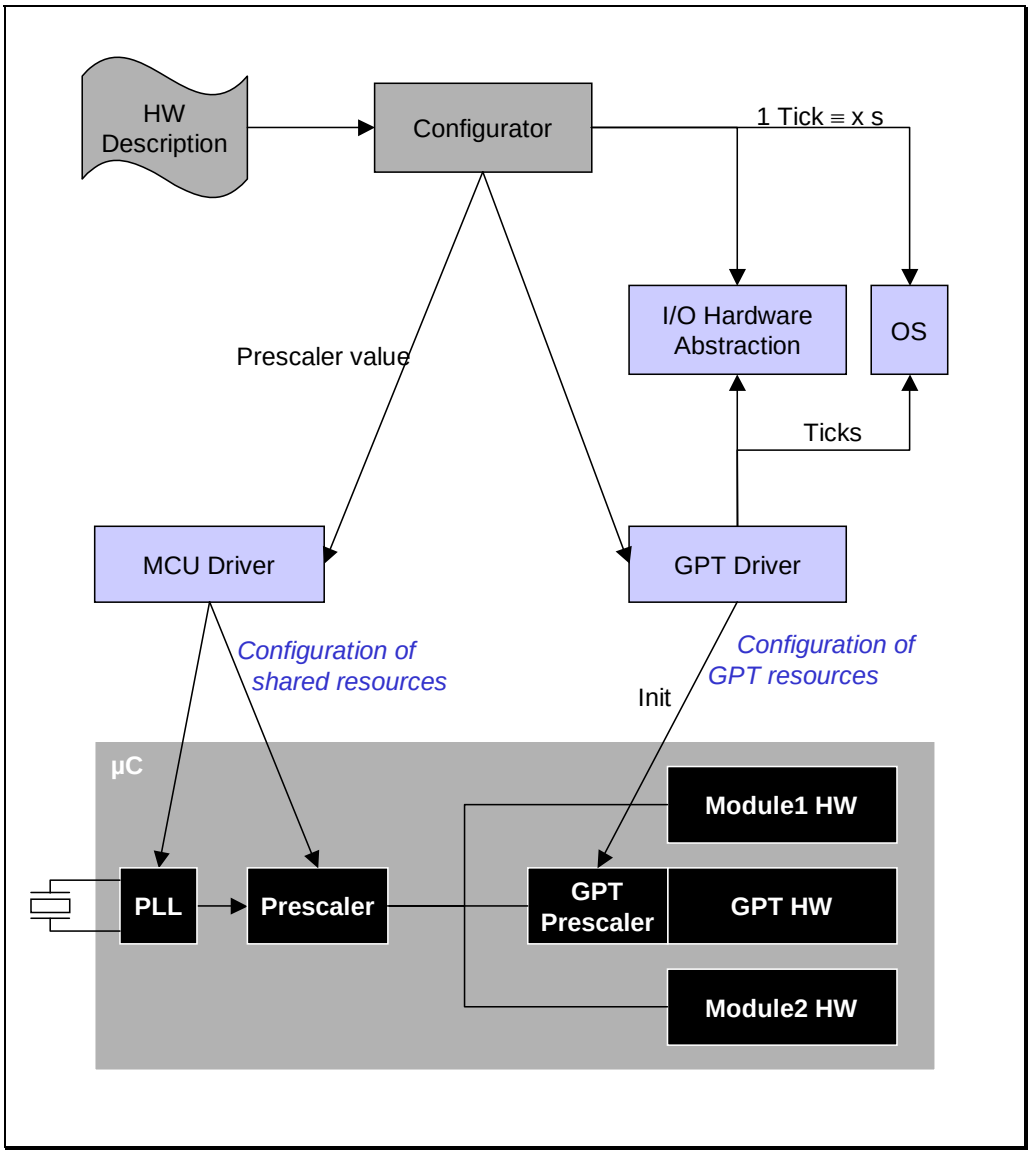

Figure 10: Scope of the GPT Driver configuration

10.2.6 Wakeup Configuration

SWS Item	**GPT235:**
Container Name	Wakeup Configuration
Description	This container contains the configuration (parameters) needed to configure a wakeup capable channel
Configuration Parameters	

Name	Gpt_ChannelWakeupInfo		
Description	In case the wakeup-capability is true this value is transmitted to the Ecu State Manager[7]		
Type	EcuM_WakeupSourceType		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	ECU		
Dependency	**--**		

10.2.7 Configuration of optional API services

SWS Item	**GPT193:**
Container Name	Configuration of optional API services
Description	This container contains all configuration switches for configuring optional API services of the GPT driver.
Configuration Parameters	

Name	GPT_VERSION_INFO_API		
Description	Adds / removes the service Gpt_GetVersionInfo() from the code.		
Type	#define		
Unit	--		
Range	STD_ON	Gpt_GetVersionInfo() can be used	
	STD_OFF	Gpt_GetVersionInfo() can not be used	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	GPT_DEINIT_API		
Description	Adds / removes the service Gpt_DeInit() from the code.		
Type	#define		
Unit	--		
Range	STD_ON	Gpt_DeInit() can be used	
	STD_OFF	Gpt_DeInit() can not be used	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	GPT_TIME_ELAPSED_API		
Description	Adds / removes the service Gpt_GetTimeElapsed() from the code.		
Type	#define		
Unit	--		
Range	STD_ON	Gpt_GetTimeElapsed() can be used	
	STD_OFF	Gpt_GetTimeElapsed () can not be used	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	GPT_TIME_REMAINING_API		
Description	Adds / removes the service Gpt_GetTimeRemaining() from the code.		
Type	#define		
Unit	--		
Range	STD_ON	Gpt_GetTimeRemaining() can be used	
	STD_OFF	Gpt_GetTimeRemaining() can not be used	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		

Dependency	None

Name	GPT_ENABLE_DISABLE_NOTIFICATION_API		
Description	Adds / removes the services Gpt_EnableNotification() and Gpt_DisableNotification from the code.		
Type	#define		
Unit	--		
Range	STD_ON	Gpt_EnableNotification() and Gpt_DisableNotification can be used	
	STD_OFF	Gpt_EnableNotification() and Gpt_DisableNotification can not be used	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	GPT_WAKEUP_FUNCTIONALITY_API		
Description	Adds / removes the services Gpt_SetMode(), Gpt_EnableWakeup() and Gpt_DisableWakeup() from the code.		
Type	#define		
Unit	--		
Range	STD_ON	Gpt_SetMode() can be used	
	STD_OFF	Gpt_SetMode() can not be used	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
None	--	--

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

SWS Item	**GPT189:**	
Information elements		
Information element name	***Type / Range***	***Information element description***
GPT_VENDOR_ID	#define/ uint16	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
GPT_MODULE_ID	#define/ uint8	Module ID of this module from Module List
GPT_AR_MAJOR_VERSION	#define/ uint8	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
GPT_AR_MINOR_VERSION	#define/ uint8	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
GPT_AR_PATCH_VERSION	#define/ uint8	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
GPT_SW_MAJOR_VERSION	#define/ uint8	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
GPT_SW_MINOR_VERSION	#define/ uint8	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
GPT_SW_PATCH_VERSION	#define/ uint8	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

11 Changes to Release 1

11.1 Deleted SWS Items

SWS Item	*Rationale*
GPT027, GPT046, GPT063, GPT064	Not required for new configuration concept
GPT119, GPT120, GPT121	Redundant to GPT205
GPT124	Requirement applicable for configuration tool
GPT169	Not required for new version checking concept
GPT190	Requirement applicable for configuration tool
GPT191	Requirement applicable for configuration tool

11.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
GPT002	GPT178	Redundancy
GPT003	GPT179	Redundancy
GPT016	GPT232	Moved to another chapter
GPT017	GPT233	Moved to another chapter
GPT024	GPT187	Moved to another chapter
GPT049	GPT204	Moved to another chapter
GPT069	GPT206	Moved to another chapter
GPT071	GPT185	Moved to another chapter
GPT072	GPT186	Moved to another chapter
GPT078	GPT207	Moved to another chapter
GPT080	GPT208	Moved to another chapter
GPT096	GPT234	Moved to another chapter
GPT104	GPT188	Moved to another chapter
GPT106	GPT209	Moved to another chapter
GPT122	GPT190	Moved to another chapter
GPT123	GPT191	Moved to another chapter
GPT125	GPT119, GPT120, GPT121	Redundancy
GPT166	GPT192	Reformulation of SWS ID to achieve consistency among SPAL SWS.

11.3 Changed SWS Items

SWS Item	*Rationale*
GPT010	Bug 5541
GPT084, GPT097, GPT101	Improved description.
GPT085	Incorporated review comment
GPT105	Bug 4555
GPT119	Bug 5463
GPT128, GPT129, GPT150, GPT235	Bug #10088

GPT184	Replaced types with “implementation specific” for parameters within GptChannelConfiguration
GPT148	Bug 4549
GPT149	Bug 4550

11.4 Added SWS Items

SWS Item	*Rationale*
GPT127, GPT128, GPT129, GPT132, GPT150	Wakeup feature first introduced in Release 2.0
GPT171, GPT172, GPT173, GPT174, GPT175, GPT176, GPT177, GPT178, GPT179, GPT181, GPT182,	Required for new SWS template.
GPT183	Required field for specifying Container “GptDriverConfiguration”
GPT184	Required field for specifying Container “GptChannelConfiguration”
GPT189	Required field for specifying “Published Information”
GPT193	Required field for specifying “Configuration of optional API services”
GPT194	Configuration of optional API service “Gpt_DeInit()“
GPT195	Configuration of optional API service “Gpt_GetTimeElapsed() “
GPT196	Configuration of optional API service “Gpt_GetTimeRemaining() “
GPT197	Configuration of optional API service “Gpt_StartTimer() “
GPT198	Configuration of optional API service “Gpt_StopTimer() “
GPT199	Configuration of optional API service “Gpt_EnableNotification() “
GPT200	Configuration of optional API service “Gpt_DisableNotification() “
GPT201	Configuration of optional API service “Gpt_SetMode() “
GPT202	Configuration of optional API service “Gpt_DisableWakeup() “
GPT203	Configuration of optional API service “Gpt_EnableWakeup() “
GPT205	Rules concerning register initialization defined
GPT210, GPT222	API parameter checking: “Gpt_GetTimeElapsed”
GPT211, GPT223	API parameter checking: “Gpt_GetTimeRemaining”
GPT212, GPT218, GPT224	API parameter checking: “Gpt_StartTimer”
GPT213, GPT225	API parameter checking: “Gpt_StopTimer”
GPT214, GPT226	API parameter checking: “Gpt_EnableNotification”
GPT215, GPT230	API parameter checking: “Gpt_EnableWakeup”
GPT216, GPT229	API parameter checking: “Gpt_DisableWakeup”
GPT217, GPT227	API parameter checking: “Gpt_DisableNotification”
GPT220	API parameter checking: “Gpt_DeInit”
GPT221	API parameter checking: “Gpt_GetVersionInfo”
GPT228, GPT231	API parameter checking: “Gpt_SetMode”
GPT235	Configuration of wakeup
GPT236	Bug 4940
GPT240, GPT241, GPT242, GPT243,	Dependency to other modules
GPT245, GPT246	Mandatory Interfaces: DEM
GPT255	Review comment #71
GPT256	Review comment #75
GPT257	added (SPAL decision, 42nd meeting, minutes day2, issue 5)
Release 2.1	
GPT258	Bugzilla #11592

12 Changes to Release 2.0

12.1 Deleted SWS Items

SWS Item	*Rationale*
GPT149	Covered by reformulated GPT097
GPT170	Covered by reformulated GPT101
GPT059	Bug **13096**
GPT219	Bug **13902**
GPT246	**Bug 15795**
GPT243	**Bug 14502**
GPT244	**Bug 12379**
GPT263	**Bug 17614**
GPT197	**Bug 17617**
GPT198	**Bug 17617**

12.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
--	--	--

12.3 Changed SWS Items

SWS Item	*Rationale*
GPT216	**Rfc #11978:** Chapter 8.3.12, GPT216, Formal Error Description: SWS requirement GPT216 is for Gpt_EnableWakeUp(). In this requirement Function name is mentioned as Gpt_EnableNotification instead of Gpt_EnableWakeUp.
GPT221	**Rfc #12318:** Chapter 8.3.1, GPT221 Description: 1) There is a wrong mention on Gpt_Deinit.
GPT218	**Rfc #12321:** Chapter 8.3.6, GPT218 Description: It should be "GPT_E_PARAM_VALUE" and not "GPT_E_PARAM_CHANNEL"
GPT210 GPT211 GPT222 GPT223	**Rfc #12327:** Chapter 8.3.4 and 8.3.5 Description: Return value not mentioned when development error detection is enabled and Gpt_GetTimeElapsed\Gpt_GetTimeRemaining is called prior to initializing the driver.
GPT097 GPT100 GPT101 GPT102	**Rfc #14814**
GPT184	**Bug 12661**
GPT153 GPT159 GPT160 GPT258	**Rfc #13708**
GPT188	**Rfc #13743**

GPT183	**Rfc #15740:** container structure changed, “GptDriverConfiguration” includes no subcontainer anymore
GPT184	**Rfc #15740:** Changed multiplicity of “wakeup configuration” from (0…*) → (0…1)
GPT175	**Bug 17616**

12.4 Added SWS Items

SWS Item	*Rationale*
GPT259	**Rfc #12317:** GPT: File Structure not correct Gpt_xCfg.c doesn’t include Gpt.h, then how can it define the configuration structure(Gpt_ConfigType) declared in Gpt.h?
GPT260	
GPT261	
GPT262	
GPT263	
GPT264	
GPT265	
GPT266	
GPT267	**Rfc #14814**
GPT268	
GPT269	**Rfc #15740:** introduced new container “GptChannelConfigSet”
GPT270	**Rfc #15470:** SPAL decission (Meeting 51): it shall not be allowed to mix parameters assigned to different configuration classes within one container
GPT271	RfC #12487 File Include Structure