Document Title	Specification of GPT Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	030
Document Classification	Standard

Document Version	3.2.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Change Description
13.10.2011	3.2.0	AUTOSAR Administration	• Range added to GPT331_Conf • "module short name" replaced by "module abbreviation" • Chapter 6 revised and chapter 13 added due to new traceability mechanism
14.10.2010	3.1.0	AUTOSAR Administration	• GPT208, GPT376 and GPT378 removed • Multiplicity changed in GPT312_Conf (chapter 10.2.6 updated) • GPT256 rephrased • GPT256 changed according to changed BSW004
30.11.2009	3.0.0	AUTOSAR Administration	• 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
23.06.2008	2.2.1	AUTOSAR Administration	Legal disclaimer revised

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
11.12.2007	2.2.0	AUTOSAR Administration	• 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
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

Disclaimer

Advice for users

AUTOSAR specifications 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 specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, 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.

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
- 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 (OS) or Complex Device Drivers (CDD) directly. The number of timer channels controlled by the GPT driver depends on hardware, implementation and system configuration.

Beside the GPT there are other possibilities for time measurements:
Software Free Running Timer (SWFRT) functionality, implemented in the AUTOSAR OS. The SWFRT especially is designed for reuse of timers with "short" tick durations, only realizable on base of hardware timers (typically 100ns ... 1ms).
An other possibility for reusing timers is to create software timers based on cyclic OS tasks or cyclic interrupts. The tick duration of such a timer is equal to the respective cycle time (typically ≥ 1ms).

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 [11]).

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
DEM	Diagnostic Event Manager
DET	Development 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
SWFRT	Software Free Running Timer

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

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 Development Error Tracer,
AUTOSAR_SWS_DevelopmentErrorTracer.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.doc

[13] Glossary,
AUTOSAR_TR_Glossary.pdf

[14] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

3.2 Related standards and norms

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

4 Constraints and assumptions

4.1 Assumptions

Each timer channel is able to trigger an interrupt.

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 DEM [7]
Production errors will be reported to the Diagnostic Event Manager

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.

5.1 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:

[GPT365] ⌈Gpt.h: Module header file (interface file) ⌋ (BSW00370)

[GPT367] ⌈Gpt.c: Module source file⌋( )

[GPT368] ⌈Gpt_Cfg.c: Module configuration file, if pre-compile const are used⌋ (BSW00345, BSW00419)

[GPT369] ⌈Gpt_Cfg.h: Module configuration file, if pre-compile #defines are used⌋ (BSW00345, BSW00381, BSW412)

[GPT371] ⌈Gpt_PBcfg.c: Module config. parameters, if post-build time config. parameters are used⌋ (BSW00380)

[GPT372] ⌈Gpt_Irq.c: File for implementation of interrupt frame (not interrupt service routine) ⌋ (BSW00314)

Users of the GPT driver shall only include the `Gpt.h` file.

[GPT373] ⌈`Gpt.c` shall include `SchM_Gpt.h` in order to access the module specific functionality provided by the BSW scheduler.⌋ (BSW00435)

[GPT172] ⌈The module shall optionally include the `Dem.h` file if any production error will be issued by the implementation.
Comment: 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. ⌋(BSW00409)

[GPT259] ⌈`Gpt.h` shall include `Gpt_Cfg.h` for the API pre-compiler switches. ⌋( )

[GPT293] ⌈`Gpt.c` shall include `Gpt.h`.
Comment: `Gpt.c` has implicit access to the `Gpt_Cfg.h` through the `Gpt.h` file. ⌋( )

[GPT261] ⌈`Gpt_Irq.c` shall include `Gpt.h` for the prototype declaration of the notification functions. ⌋(BSW164)

[GPT271] ⌈`Gpt.h` shall include `EcuM_Cbk.h`, if wakeup functionality is configured. ⌋( )

[GPT374] ⌈All files of GPT module with extension ".c" (`Gpt*.c`) shall include `MemMap.h`. ⌋(BSW00436)

[GPT375] ⌈If development error detection for the GPT module is enabled: `Gpt.c` shall include `Det.h`⌋( )

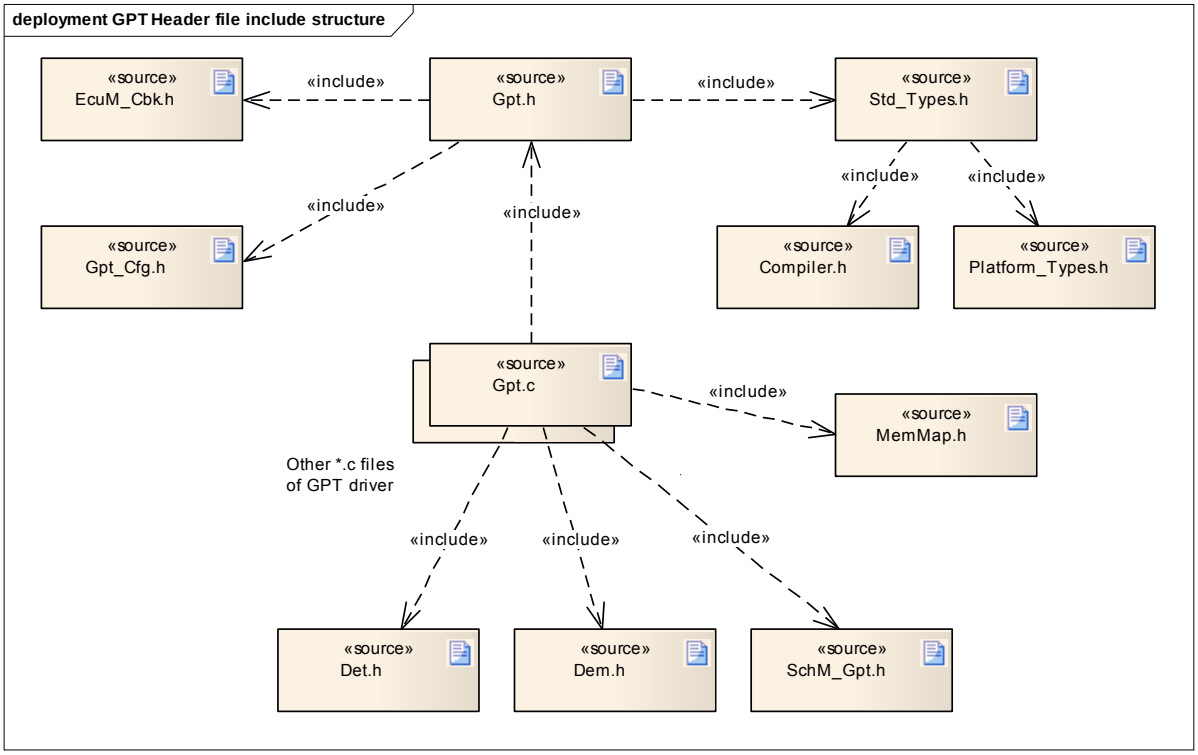

Figure 1: Header file include structure

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	Satisfied by
-	GPT338
-	GPT361
-	GPT367
-	GPT258
-	GPT329
-	GPT324
-	GPT294
-	GPT323
-	GPT331
-	GPT214
-	GPT157
-	GPT336
-	GPT215
-	GPT293
-	GPT231
-	GPT297
-	GPT113
-	GPT328
-	GPT216
-	GPT295
-	GPT309
-	GPT330
-	GPT185
-	GPT099
-	GPT115
-	GPT322
-	GPT118
-	GPT086
-	GPT210
-	GPT176

-	GPT217
-	GPT165
-	GPT379
-	GPT116
-	GPT175
-	GPT218
-	GPT339
-	GPT186
-	GPT271
-	GPT363
-	GPT334
-	GPT107
-	GPT362
-	GPT084
-	GPT177
-	GPT343
-	GPT117
-	GPT211
-	GPT344
-	GPT364
-	GPT213
-	GPT114
-	GPT105
-	GPT340
-	GPT341
-	GPT303
-	GPT333
-	GPT212
-	GPT321
-	GPT158
-	GPT377
-	GPT326
-	GPT307
-	GPT335
-	GPT155
-	GPT375
-	GPT259
-	GPT305
-	GPT164
-	GPT273
-	GPT093
-	GPT156

-	GPT301
-	GPT255
-	GPT234
-	GPT337
-	GPT299
BSW00304	GPT174
BSW00305	GPT359, GPT357, GPT358, GPT360
BSW00306	GPT381
BSW00307	GPT381
BSW00308	GPT381
BSW00309	GPT381
BSW00314	GPT372
BSW00321	GPT381
BSW00325	GPT381
BSW00326	GPT381
BSW00328	GPT381
BSW00330	GPT381
BSW00331	GPT381
BSW00333	GPT381
BSW00334	GPT381
BSW00335	GPT381
BSW00336	GPT281, GPT008
BSW00337	GPT004
BSW00338	GPT178
BSW00339	GPT179
BSW00341	GPT381
BSW00342	GPT381
BSW00344	GPT381
BSW00345	GPT368, GPT369
BSW00347	GPT381
BSW00348	GPT381, GPT278
BSW00353	GPT381
BSW00357	GPT381
BSW00358	GPT280
BSW00359	GPT381
BSW00360	GPT381
BSW00361	GPT381
BSW00369	GPT179, GPT178
BSW00370	GPT365
BSW00373	GPT381
BSW00375	GPT209, GPT292
BSW00376	GPT381

BSW00377	GPT381
BSW00378	GPT381
BSW00380	GPT371
BSW00381	GPT369
BSW00398	GPT381
BSW004	GPT256
BSW00404	GPT280, GPT357
BSW00405	GPT280, GPT357
BSW00406	GPT230, GPT229, GPT228, GPT227, GPT226, GPT225, GPT224, GPT223, GPT222, GPT220, GPT325
BSW00407	GPT181, GPT279
BSW00409	GPT172
BSW00413	GPT381
BSW00414	GPT280, GPT357
BSW00415	GPT381
BSW00416	GPT381
BSW00417	GPT381
BSW00419	GPT368
BSW00422	GPT381
BSW00423	GPT381
BSW00424	GPT381
BSW00425	GPT381
BSW00426	GPT381
BSW00427	GPT381
BSW00428	GPT381
BSW00429	GPT381
BSW00432	GPT381
BSW00433	GPT381
BSW00435	GPT373
BSW00436	GPT374
BSW00437	GPT381
BSW00438	GPT280, GPT357
BSW00439	GPT381
BSW00440	GPT381
BSW00441	GPT360
BSW005	GPT381
BSW006	GPT381
BSW007	GPT381
BSW009	GPT381
BSW010	GPT381
BSW101	GPT280, GPT006
BSW12057	GPT280, GPT006

BSW12063	GPT359
BSW12064	GPT381
BSW12067	GPT233, GPT015, GPT014
BSW12068	GPT381
BSW12069	GPT209, GPT292
BSW12075	GPT381
BSW12077	GPT381
BSW12078	GPT381
BSW12092	GPT381
BSW12116	GPT308, GPT162, GPT281, GPT008
BSW12117	GPT083, GPT282, GPT283, GPT010
BSW12119	GPT285, GPT013
BSW12120	GPT233
BSW12121	GPT014, GPT286
BSW12122	GPT015, GPT287
BSW12125	GPT068
BSW12128	GPT274, GPT275, GPT284
BSW12129	GPT327, GPT206
BSW12163	GPT281, GPT008
BSW12169	GPT288, GPT151
BSW12263	GPT357
BSW12265	GPT381
BSW12328	GPT359
BSW12448	GPT332, GPT178
BSW12461	GPT353, GPT354, GPT352, GPT355, GPT356
BSW12462	GPT381
BSW12463	GPT381
BSW13601	GPT127
BSW13602	GPT160, GPT290, GPT289, GPT159
BSW13603	GPT288, GPT151, GPT152, GPT153
BSW157	GPT015, GPT014
BSW159	GPT381
BSW160	GPT381
BSW161	GPT381
BSW162	GPT381
BSW164	GPT261
BSW167	GPT381
BSW168	GPT381
BSW170	GPT381
BSW171	GPT182, GPT196, GPT195, GPT194, GPT199, GPT200, GPT203, GPT201, GPT202
BSW172	GPT381
BSW412	GPT369

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

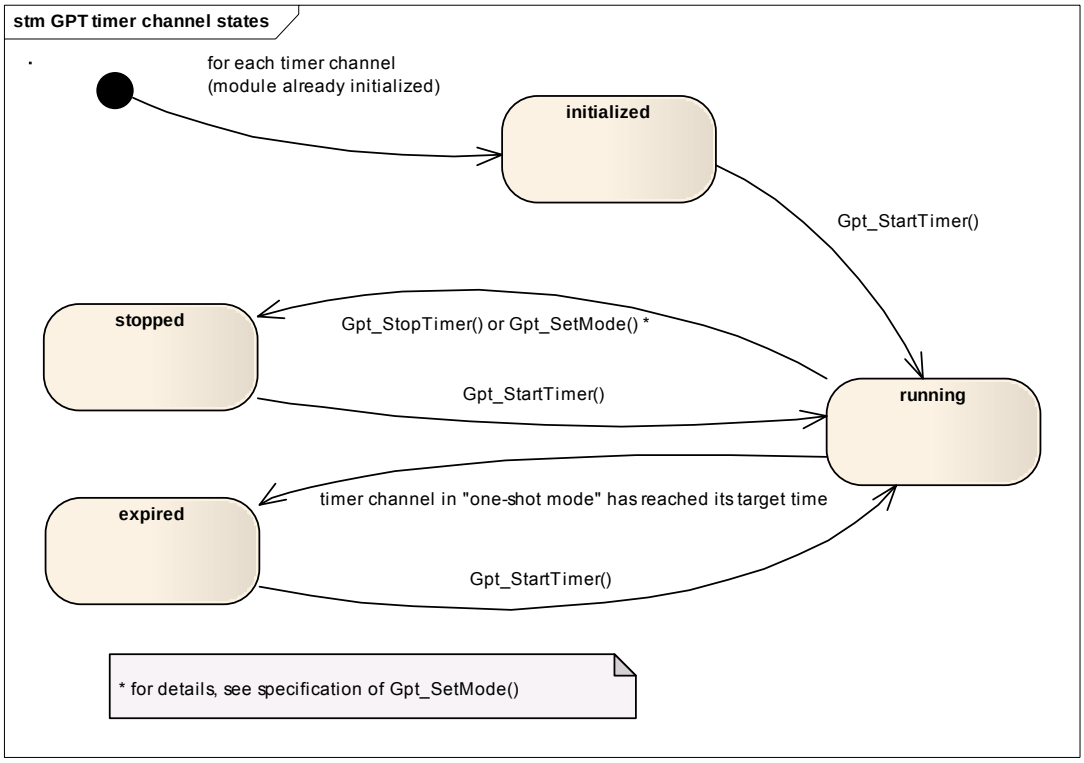

Figure 2: Channel states and state transitions

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

[GPT329] ⌈A timer channel starts counting at value zero.⌋ ( )

[GPT185] ⌈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".⌋ ( )

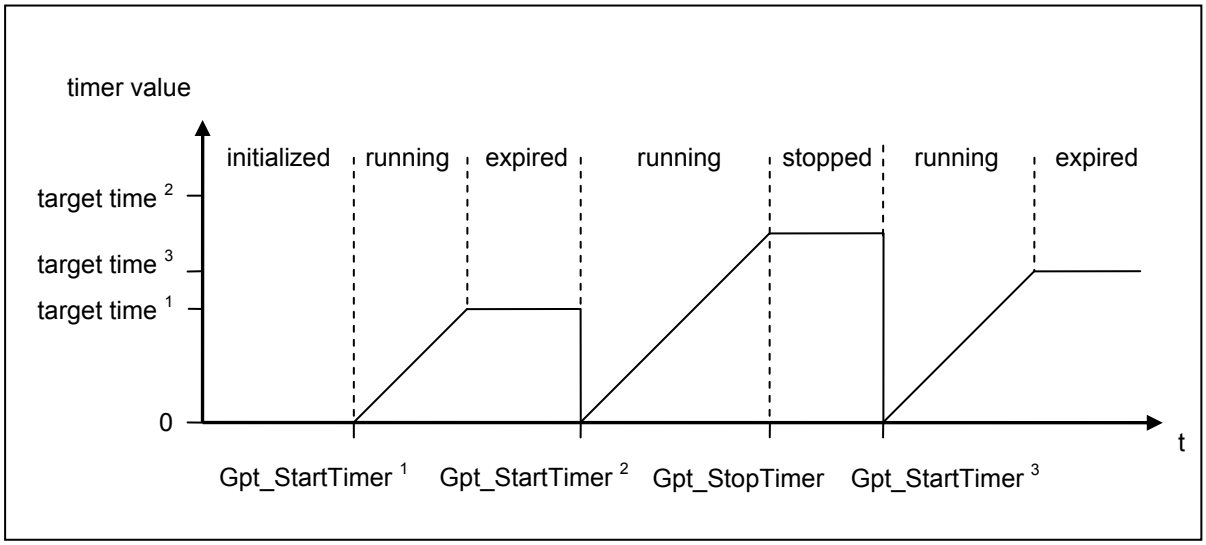

Figure 3: Timer channel in "one-shot mode"

[GPT186] ⌈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. ⌋ ( )

[GPT330] ⌈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).

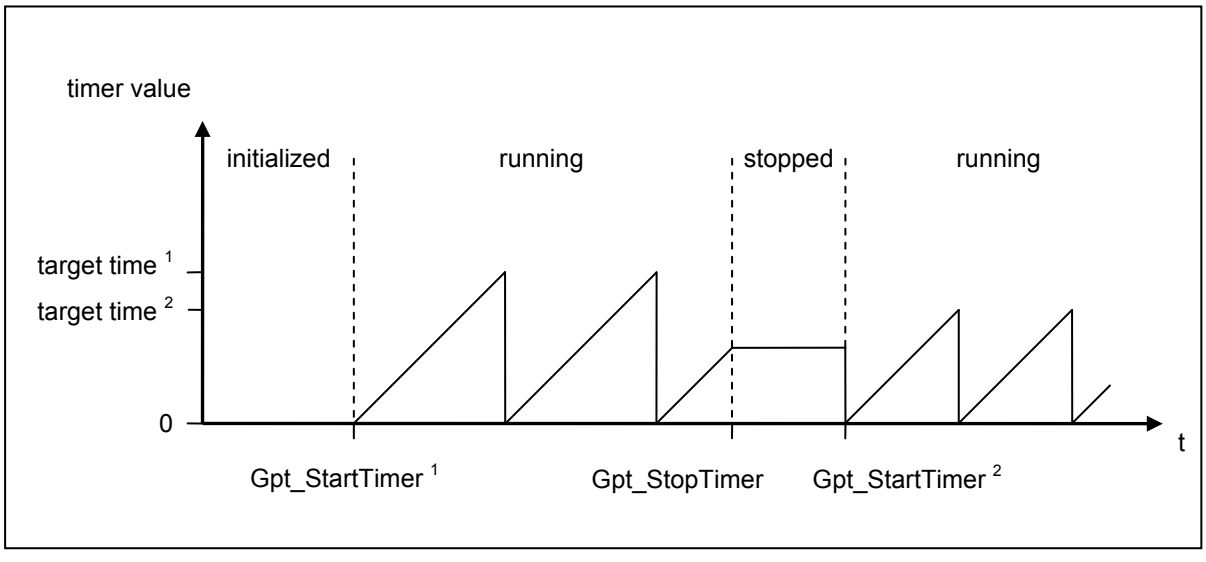

Figure 4: Timer channel in "continuous mode"

Both, the relative time elapsed and the time remaining can be queried by calling:
- `Gpt_GetTimeElapsed`
- `Gpt_GetTimeRemaining`

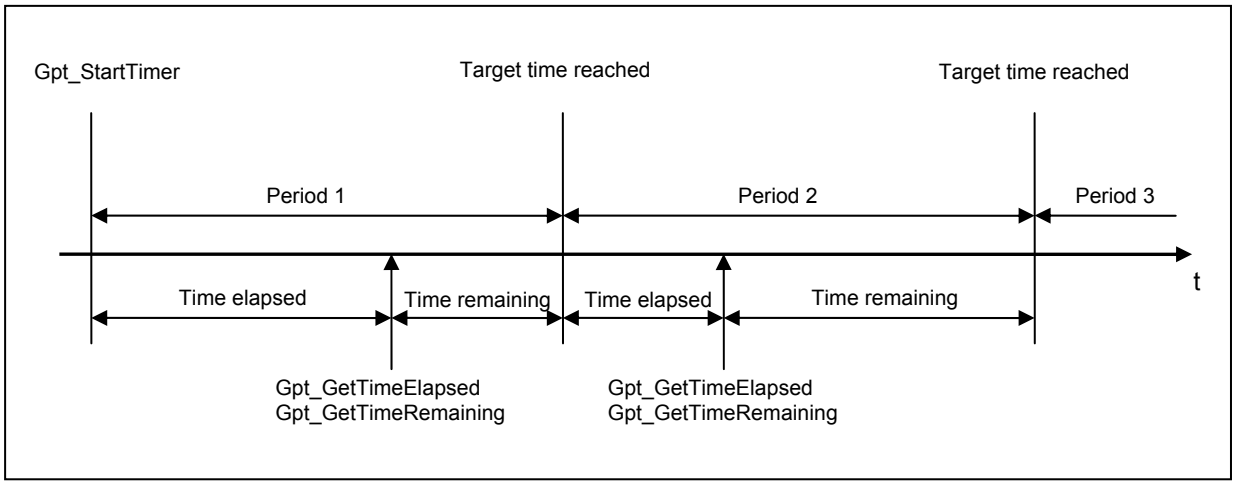

Figure 5: Querying of time elapsed / time remaining for a timer channel in "continuous mode"

⌋( )

[GPT331] ⌈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`⌋( )

[GPT127] ⌈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). ⌋(BSW13601)

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

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

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

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

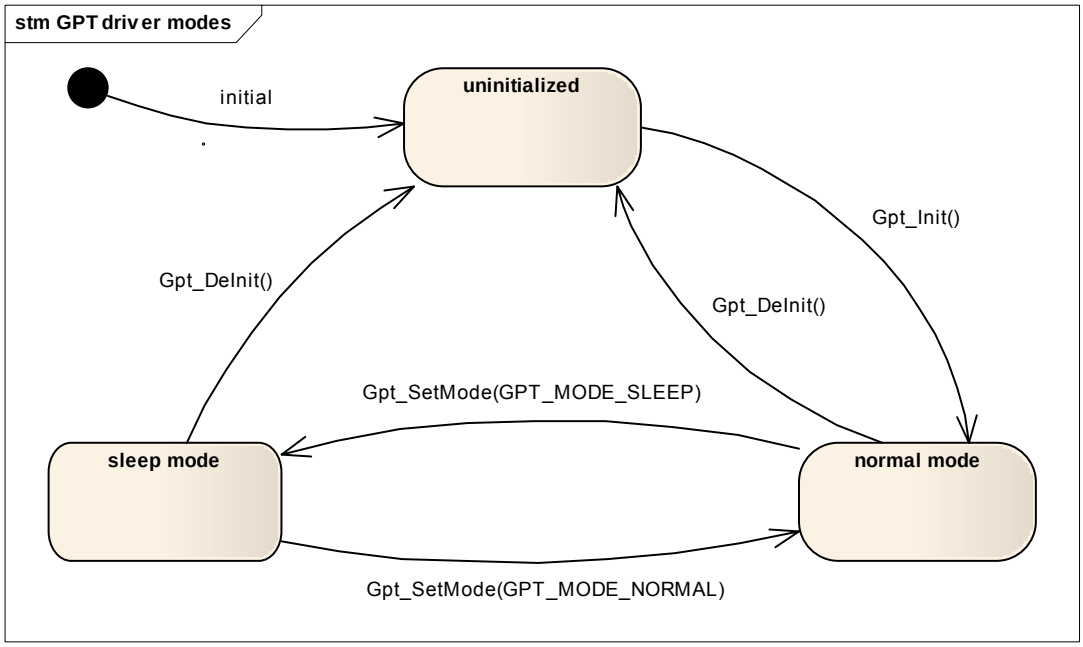

Figure 6: GPT driver modes

7.2 Version checking

[GPT256] ⌈The GPT module shall perform Inter Module Checks to avoid integration of incompatible files. The imported included files shall be checked by preprocessing directives. ⌋(BSW004)

The following version numbers shall be verified:

- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_RELEASE_MINOR_VERSION

Where <MODULENAME> is the module abbreviation of the other (external) modules which provide header files included by the GPT module.

If the values are not identical to the expected values, an error shall be reported.

7.3 Error classification

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`.⌋(BSW00304)

The following errors shall be detectable by the GPT driver depending on its build version (development / production).

ID	Type of error	Relevance	Related error code	Value [hex]
GPT345	API service called without module initialization	Development	`GPT_E_UNINIT`	`0x0A`
GPT346	API service called when timer channel is still busy (running)	Development	`GPT_E_BUSY`	`0x0B`
GPT347	API service for initialization called when already initialized	Development	`GPT_E_ALREADY_INITIALIZED`	`0x0D`
GPT348	API parameter checking: invalid channel	Development	`GPT_E_PARAM_CHANNEL`	`0x14`
GPT349	API parameter checking: invalid value	Development	`GPT_E_PARAM_VALUE`	`0x15`
GPT350	API parameter checking: invalid pointer	Development	`GPT_E_PARAM_POINTER`	`0x16`
GPT351	API parameter checking: invalid mode	Development	`GPT_E_PARAM_MODE`	`0x1F`
	No production errors assigned	Production	-	

Table 3: 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.⌋(BSW00337)

7.4 Error detection

[GPT175] ⌈The detection of development errors is configurable (`STD_ON/STD_OFF`) at pre-compile time. The switch `GptDevErrorDetect` (see chapter 10) shall activate or deactivate the detection of all development errors.⌋( )

[GPT176] ⌈If the `GptDevErrorDetect` switch is enabled, API parameter checking and call sequence checking is enabled. The detailed description of the detected errors can be found in chapter 7.3 and chapter 8.⌋( )

[GPT332] ⌈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).⌋(BSW12448)

[GPT177] ⌈If production errors are specified for GPT module:
The detection of production code errors cannot be switched off.⌋( )

7.5 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 `GptDevErrorDetect` is set (see chapter 10). ⌋ (BSW00338, BSW00369, BSW12448)

[GPT179] ⌈If production errors are specified for GPT module:
Production errors shall be reported to Diagnostic Event Manager[7]. ⌋ (BSW00369, BSW00339)

7.6 Debugging

The following requirements deal with the definition of variables and the description of debug information.

[GPT333] ⌈Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable. ⌋ ( )

[GPT334] ⌈All type definitions of variables which shall be debugged, shall be accessible by the header file Gpt.h. ⌋ ( )

[GPT335] ⌈The declaration of variables in the header file shall allow to calculate the size of the variables by C-"sizeof".⌋ ( )

[GPT336] ⌈Variables available for debugging shall be described in the respective Basic Software Module Description⌋ ( )

[GPT337] ⌈The operation mode of the GPT driver and the state of each timer channel shall be available for debugging. ⌋ ( )

8 API specification

8.1 Imported types

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

[GPT278] ⌈

Module	*Imported Type*
Dem	Dem_EventIdType
	Dem_EventStatusType
EcuM	EcuM_WakeupSourceType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋ (BSW00348)

8.2 Type Definitions

8.2.1 Gpt_ConfigType

[GPT357] ⌈

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.	

⌋ (BSW00404, BSW00405, BSW00438, BSW00305, BSW00414, BSW12263)

8.2.2 Gpt_ChannelType

[GPT358] ⌈

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.		

⌋ (BSW00305)

8.2.3 Gpt_ValueType

[GPT359] ⌈

Name:	`Gpt_ValueType`
Type:	`uint`

Range:	- -	- -	The range of this type is µC dependent (width of the timer register) and has to be described by the supplier.
Description:	Type for reading and setting the timer values (in number of ticks).		

⌋ (BSW00305, BSW12063, BSW12328)

8.2.4 Gpt_ModeType

[GPT360] ⌈

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

⌋ (BSW00441, BSW00305)

8.3 Function definitions

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

8.3.1 Gpt_GetVersionInfo

[GPT279] ⌈

Service name:	Gpt_GetVersionInfo	
Syntax:	`void Gpt_GetVersionInfo( Std_VersionInfoType* VersionInfoPtr )`	
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.	

⌋ (BSW00407)

[GPT181] ⌈The function `Gpt_GetVersionInfo` shall return the version information of this module according to the definition of `Std_VersionInfoType` [4]. ⌋ (BSW00407)

[GPT273] ⌈If source code for caller and callee of `Gpt_GetVersionInfo` is available, the GPT module should realize `Gpt_GetVersionInfo` as a macro, defined in the module's header file. ⌋ ( )

[GPT182] ⌈The function `Gpt_GetVersionInfo` shall be pre compile time configurable On/Off by the configuration parameter: `GptVersionInfoApi`⌋ (BSW171)

[GPT338] ⌈If development error detection for the GPT module is enabled:
If the parameter `VersionInfoPtr` is a null pointer, the function `Gpt_GetVersionInfo` shall raise the error `GPT_E_PARAM_POINTER`. ⌋( )

8.3.2 Gpt_Init

[GPT280] ⌈

Service name:	Gpt_Init	

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

Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfigPtr	Pointer to a selected configuration structure
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the hardware timer module.	

⌋ (BSW00404, BSW00405, BSW00438, BSW101, BSW00358, BSW00414, BSW12057)

[GPT006] ⌈The function `Gpt_Init` shall initialize the hardware timer module according to a configuration set referenced by `ConfigPtr`. ⌋(BSW101, BSW12057)

[GPT107] ⌈The function `Gpt_Init` shall disable all interrupt notifications, controlled by the GPT driver. ⌋( )

[GPT068] ⌈The function `Gpt_Init` shall only initialize the configured resources. Resources that are not configured in the configuration file shall not be touched. ⌋ (BSW12125)

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

- **[GPT352]** ⌈If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register⌋ (BSW12461)
- **[GPT353]** ⌈If the register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver⌋ (BSW12461)

- **[GPT354]** ⌈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⌋ (BSW12461)
- **[GPT355]** ⌈One-time writable registers that require initialization directly after reset shall be initialized by the startup code⌋ (BSW12461)
- **[GPT356]** ⌈All other registers shall be initialized by the startup code⌋ (BSW12461)

[GPT307] ⌈If development error detection for the GPT module is enabled: If the GPT driver is not in operation mode "uninitialized", the function `Gpt_Init` shall raise the error `GPT_E_ALREADY_INITIALIZED`.⌋ ( )

[GPT258] ⌈The function `Gpt_Init` shall disable all wakeup interrupts, controlled by the GPT driver.⌋ ( )

[GPT339] ⌈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`.⌋ ( )

[GPT294] ⌈If development error detection for the GPT module is enabled:
If VARIANT-POST-BUILD is supported by implementation and the parameter `ConfigPtr` is a null pointer, the function `Gpt_Init` shall raise the error `GPT_E_PARAM_POINTER`.⌋ ( )

[GPT340] ⌈If development error detection for the GPT module is enabled:
If VARIANT-POST-BUILD is **not** supported by implementation and the parameter `ConfigPtr` is **not** a null pointer, the function `Gpt_Init` shall raise the error `GPT_E_PARAM_POINTER`.⌋ ( )

[GPT309] ⌈A re-initialization of the GPT driver by executing the `Gpt_Init` function requires a de-initialization before by executing a `Gpt_DeInit`.⌋ ( )

8.3.3 Gpt_DeInit

[GPT281] ⌈

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 all hardware timer channels.

⌋(BSW00336, BSW12163, BSW12116)

[GPT008] ⌈The function `Gpt_DeInit` shall deinitialize the hardware used by the GPT driver (depending on configuration) to the power on reset state. Values of registers which are not writeable are excluded. It's the responsibility of the hardware design that the state does not lead to undefined activities in the µC. ⌋(BSW00336**,** BSW12163, BSW12116)

[GPT105] ⌈The function `Gpt_DeInit` shall disable all interrupt notifications and wakeup interrupts, controlled by the GPT driver. ⌋( )

[GPT162] ⌈The function `Gpt_DeInit` shall influence only the peripherals, which are allocated by the static configuration. ⌋(BSW12116)

[GPT308] ⌈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`. ⌋(BSW12116)

[GPT194] ⌈The function `Gpt_DeInit` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptDeInitApi`. ⌋(BSW171)

[GPT363] ⌈The function `Gpt_DeInit` shall set the operation mode of the GPT driver to "uninitialized".⌋( )

[GPT234] ⌈If development error detection for the GPT module is enabled:
If any timer channel is in state "running", the function `Gpt_DeInit` shall raise the error `GPT_E_BUSY`. ⌋( )

[GPT220] ⌈If development error detection for the GPT module is enabled:
If the driver is not initialized, the function `Gpt_DeInit` shall raise the error `GPT_E_UNINIT`. ⌋(BSW00406)

8.3.4 Gpt_GetTimeElapsed

[GPT282] ⌈

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.	

⌋ (BSW12117)

[GPT010] ⌈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. ⌋ (BSW12117)

[GPT361] ⌈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. ⌋ ( )

[GPT295] ⌈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`".⌋ ( )

[GPT297] ⌈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. ⌋ ( )

[GPT299] ⌈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. ⌋ ( )

[GPT113] ⌈The function `Gpt_GetTimeElapsed` shall be reentrant, if the timer channels used in concurrent calls are different. ⌋ ( )

[GPT195] ⌈The function `Gpt_GetTimeElapsed` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptTimeElapsedApi`. ⌋ (BSW171)

[GPT222] ⌈If development error detection for the GPT module is enabled:
If the driver is not initialized, the function `Gpt_GetTimeElapsed` shall raise the error `GPT_E_UNINIT` and shall return the value “`0`”. ⌋ (BSW00406)

[GPT210] ⌈If development error detection for the GPT module is enabled:
If the parameter `Channel` is invalid (not within the range specified by configuration),

the function `Gpt_GetTimeElapsed` shall raise the development error `GPT_E_PARAM_CHANNEL` and shall return the value "0". ⌋( )

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

Table 4: Summary: Return values and DET errors of Gpt_GetTimeElapsed

8.3.5 Gpt_GetTimeRemaining

[GPT283] ⌈

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.	

⌋(BSW12117)

[GPT083] ⌈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. ⌋(BSW12117)

[GPT301] ⌈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".⌋( )

[GPT303] ⌈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. ⌋( )

[GPT305] ⌈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`". ⌋( )

[GPT114] ⌈The function `Gpt_GetTimeRemaining` shall be reentrant, if the timer channels used in concurrent calls are different. ⌋( )

[GPT196] ⌈The function Gpt_GetTimeRemaining shall be pre compile time configurable On/Off by the configuration parameter: `GptTimeRemainingApi`. ⌋ (BSW171)

[GPT223] ⌈If development error detection for the GPT module is enabled:
If the driver is not initialized, the function `Gpt_GetTimeRemaining` shall raise the error `GPT_E_UNINIT` and shall return the value "`0`". ⌋(BSW00406)

[GPT211] ⌈If development error detection for the GPT module is enabled:
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` and shall return the value "`0`". ⌋( )

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 5: Summary: Return values and DET errors of Gpt_GetTimeRemaining

8.3.6 Gpt_StartTimer

[GPT284] ⌈

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.

	Value	Target time in number of ticks.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Starts a timer channel.	

⌋(BSW12128)

[GPT274] ⌈The function `Gpt_StartTimer` shall start the selected timer channel with a defined target time.⌋(BSW12128)

[GPT275] ⌈If configured and enabled, an interrupt notification or a wakeup interrupt occurs, when the target time is reached.⌋(BSW12128)

[GPT115] ⌈The function `Gpt_StartTimer` shall be reentrant, if the timer channels used in concurrent calls are different.⌋( )

[GPT364] ⌈The state of the selected timer channel shall be changed to "running" if `Gpt_StartTimer` is called.⌋( )

[GPT212] ⌈If development error detection for the GPT module is enabled:
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`.⌋( )

[GPT218] ⌈If development error detection for the GPT module is enabled:
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).⌋( )

[GPT224] ⌈If development error detection for the GPT module is enabled:
If the driver is not initialized, the function `Gpt_StartTimer` shall raise the error `GPT_E_UNINIT`.⌋(BSW00406)

[GPT084] ⌈If development error detection for the GPT module is enabled:
If the function `Gpt_StartTimer` is called on a channel in state "running", the function shall raise the error `GPT_E_BUSY`.⌋( )

8.3.7 Gpt_StopTimer

[GPT285] ⌈

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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Stops a timer channel.	

⌋(BSW12119)

[GPT013] ⌈The function `Gpt_StopTimer` shall stop the selected timer channel. ⌋ (BSW12119)

[GPT343] ⌈The state of the selected timer channel shall be changed to "stopped" if `Gpt_StopTimer` is called. ⌋( )

[GPT099] ⌈If development error detection for the GPT module is enabled:
If the function `Gpt_StopTimer` is called on a channel in state "initialized", "stopped" or "expired", the function shall not raise a development error. ⌋( )

[GPT344] ⌈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). ⌋( )

[GPT116] ⌈The function `Gpt_StopTimer` shall be reentrant, if the timer channels used in concurrent calls are different. ⌋( )

[GPT213] ⌈If development error detection for the GPT module is enabled:
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`. ⌋( )

[GPT225] ⌈If development error detection for the GPT module is enabled:
If the driver is not initialized, the function `Gpt_StopTimer` shall raise the error `GPT_E_UNINIT`. ⌋(BSW00406)

8.3.8 Gpt_EnableNotification

[GPT286] ⌈

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.
Parameters	None	

(inout):	
Parameters (out):	None
Return value:	None
Description:	Enables the interrupt notification for a channel (relevant in normal mode).

⌋(BSW12121)

[GPT014] ⌈The function `Gpt_EnableNotification` shall enable the interrupt notification of the referenced channel configured for notification (see also GPT233). The function shall save an attribute like "notification enabled" of the channel. Comment: This attribute affects the interrupt notification always when the driver is in "normal mode". In "sleep mode" the attribute has no influence. ⌋(BSW157, BSW12067, BSW12121)

[GPT117] ⌈The function `Gpt_EnableNotification` shall be reentrant, if the timer channels used in concurrent calls are different. ⌋( )

[GPT199] ⌈The function `Gpt_EnableNotification` shall be pre compile time configurable `On/Off` by the configuration parameter:
`GptEnableDisableNotificationApi`⌋(BSW171)

[GPT226] ⌈If development error detection for the GPT module is enabled:
If the driver is not initialized, the function `Gpt_EnableNotification` shall raise the error `GPT_E_UNINIT`. ⌋(BSW00406)

[GPT214] ⌈If development error detection for the GPT module is enabled:
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`. ⌋( )

[GPT377] ⌈If development error detection for the GPT module is enabled:
If no valid notification function is configured (GptNotification), the function `Gpt_EnableNotification` shall raise the error `GPT_E_PARAM_CHANNEL`. ⌋( )

8.3.9 Gpt_DisableNotification

[GPT287] ⌈

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.
Parameters (inout):	None	
Parameters (out):	None	

Return value:	None
Description:	Disables the interrupt notification for a channel (relevant in normal mode).

⌋ (BSW12122)

[GPT015] ⌈The function `Gpt_DisableNotification` shall disable the interrupt notification of the referenced channel configured for notification (see also GPT233). The function shall save an attribute like "notification disabled" of the channel.
Comment: This attribute affects the interrupt notification always when the driver is in "normal mode". In "sleep mode" the attribute has no influence. ⌋ (BSW157, BSW12122, BSW12067)

[GPT118] ⌈The function `Gpt_DisableNotification` shall be reentrant, if the timer channels used in concurrent calls are different. ⌋ ( )

[GPT200] ⌈The function `Gpt_DisableNotification` shall be pre compile time configurable `On/Off` by the configuration parameter:
`GptEnableDisableNotificationApi`. ⌋ (BSW171)

[GPT227] ⌈If development error detection for the GPT module is enabled:
If the driver is not initialized, the function `Gpt_DisableNotification` shall raise the error `GPT_E_UNINIT`. ⌋ (BSW00406)

[GPT217] ⌈If development error detection for the GPT module is enabled:
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`. ⌋ ( )

[GPT379] ⌈If development error detection for the GPT module is enabled:
If no valid notification function is configured (GptNotification), the function `Gpt_DisableNotification` shall raise the error `GPT_E_PARAM_CHANNEL`. ⌋ ( )

8.3.10 Gpt_SetMode

[GPT288] ⌈

Service name:	Gpt_SetMode	

Syntax:	`void Gpt_SetMode(`
	`  Gpt_ModeType Mode`
	`)`	

Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Mode	GPT_MODE_NORMAL: Normal operation mode of the GPT driver. GPT_MODE_SLEEP: Sleep mode of the GPT driver (wakeup capable). See also Gpt_ModeType.

Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Sets the operation mode of the GPT.

⌋ (BSW12169, BSW13603)

[GPT151] ⌈The function `Gpt_SetMode` shall set the operation mode of the GPT driver to the given mode parameter. ⌋ (BSW12169, BSW13603)

[GPT255] ⌈The function `Gpt_SetMode` is only available if the configuration parameter `GptReportWakeupSource` is enabled. ⌋ ( )

[GPT152] ⌈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. ⌋ (BSW13603)

[GPT153] ⌈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. ⌋ (BSW13603)

[GPT164] ⌈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".⌋ ( )

[GPT165] ⌈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. ⌋ ( )

[GPT341] ⌈If the parameter has the value `GPT_MODE_SLEEP` the function `Gpt_SetMode` shall not start a wakeup timer automatically. First, the application shall call `Gpt_StartTimer` to start a wakeup timer, after this the application shall call `Gpt_SetMode` with parameter `GPT_MODE_SLEEP`. ⌋ ( )

[GPT228] ⌈If development error detection for the GPT module is enabled:
If the driver is not initialized, the function `Gpt_SetMode` shall raise the error `GPT_E_UNINIT`. ⌋ (BSW00406)

[GPT231] ⌈If development error detection for the GPT module is enabled:
The function `Gpt_SetMode` shall raise the error `GPT_E_PARAM_MODE` if the parameter `Mode` is invalid. ⌋ ( )

[GPT201] ⌈The function `Gpt_SetMode` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptWakeupFunctionalityApi`. ⌋ (BSW171)

8.3.11 Gpt_DisableWakeup

[GPT289] ⌈

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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Disables the wakeup interrupt of a channel (relevant in sleep mode).	

⌋ (BSW13602)

[GPT159] ⌈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. ⌋ (BSW13602)

[GPT157] ⌈The function `Gpt_DisableWakeup` is only feasible, if `GptReportWakeupSource` is statically configured available. ⌋ ( )

[GPT155] ⌈The function `Gpt_DisableWakeup` shall be reentrant, if the timer channels used in concurrent calls are different. ⌋ ( )

[GPT202] ⌈The function `Gpt_DisableWakeup` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptWakeupFunctionalityApi`⌋ (BSW171)

[GPT215] ⌈If development error detection for the GPT module is enabled:
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.` ⌋ ( )

[GPT229] ⌈If development error detection for the GPT module is enabled:
If the driver is not initialized, the function `Gpt_DisableWakeup` shall raise the error `GPT_E_UNINIT`. ⌋(BSW00406)

8.3.12 Gpt_EnableWakeup

[GPT290] ⌈

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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Enables the wakeup interrupt of a channel (relevant in sleep mode).	

⌋(BSW13602)

[GPT160] ⌈The function `Gpt_EnableWakeup` shall enable the wakeup interrupt of the referenced channel configured for wakeup. The function shall save an attribute like "wakeup enabled" of the channel.
Comment: This attribute affects the wakeup interrupt always when the driver is in "sleep mode". In "normal mode" the attribute has no influence. ⌋(BSW13602)

[GPT158] ⌈The function `Gpt_EnableWakeup` is only feasible, if `GptReportWakeupSource` is statically configured available. ⌋( )

[GPT156] ⌈The function `Gpt_EnableWakeup` shall be reentrant, if the timer channels used in concurrent calls are different. ⌋( )

[GPT203] ⌈The function `Gpt_EnableWakeup` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptWakeupFunctionalityApi` . ⌋(BSW171)

[GPT230] ⌈If development error detection for the GPT module is enabled:
If the driver is not initialized, the function `Gpt_EnableWakeup` shall raise the error `GPT_E_UNINIT`. ⌋(BSW00406)

[GPT216] ⌈If development error detection for the GPT module is enabled:
If the parameter `Channel` is invalid (not within the range specified by configuration)

or channel wakeup is not enabled by configuration (GptEnableWakeup), the function `Gpt_EnableWakeup` shall raise the error `GPT_E_PARAM_CHANNEL.` ⌋( )

8.3.13 Gpt_CheckWakeup

[GPT328] ⌈

Service name:	Gpt_CheckWakeup	
Syntax:	`void Gpt_CheckWakeup( EcuM_WakeupSourceType WakeupSource )`	
Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	WakeupSource	Information on wakeup source to be checked. The associated GPT channel can be determined from configuration data.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Checks if a wakeup capable GPT channel is the source for a wakeup event and calls the ECU state manager service EcuM_SetWakeupEvent in case of a valid GPT channel wakeup event.	

⌋( )

[GPT321] ⌈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]. ⌋( )

[GPT322] ⌈The function `Gpt_CheckWakeup` is only feasible, if `GptReportWakeupSource` is statically configured available. ⌋( )

[GPT323] ⌈The function `Gpt_CheckWakeup` shall be reentrant, by reason of possible usage in concurrent interrupt service routines. ⌋( )

[GPT324] ⌈The function `Gpt_CheckWakeup` shall be pre compile time configurable `On/Off` by the configuration parameter:
`GptWakeupFunctionalityApi`⌋( )

[GPT325] ⌈If development error detection for the GPT module is enabled:
If the driver is not initialized, the function `Gpt_CheckWakeup` shall raise the error `GPT_E_UNINIT.` ⌋(BSW00406)

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	*Description*
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function.
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.

[GPT326] ⌈EcuM_CheckWakeup shall be called within the Interrupt Service Routine, servicing the GPT channel wakeup event on wakeup-capable channels.⌋( )

[GPT327] ⌈The ISR´s, providing the wakeup events, shall be responsible for resetting the interrupt flags (if needed by hardware).⌋(BSW12129)

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.

8.6.3.1 GPT Notification

[GPT292] ⌈

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:	--

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.⌋ (BSW00375, BSW12069)

[GPT086] ⌈The callback notifications `Gpt_Notification_<channel>` shall be configurable as pointers to user defined functions within the configuration structure. ⌋ ( )

[GPT209] ⌈Each channel shall provide its own notification if configured. ⌋ (BSW00375, BSW12069)

[GPT093] ⌈When disabled, the GPT Driver will send no notification. ⌋ ( )

[GPT233] ⌈The GPT Driver shall invoke a notification whenever the defined target time of the channel is reached. ⌋ (BSW12067, BSW12120)

[GPT206] ⌈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. ⌋ (BSW12129)

[GPT362] ⌈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.

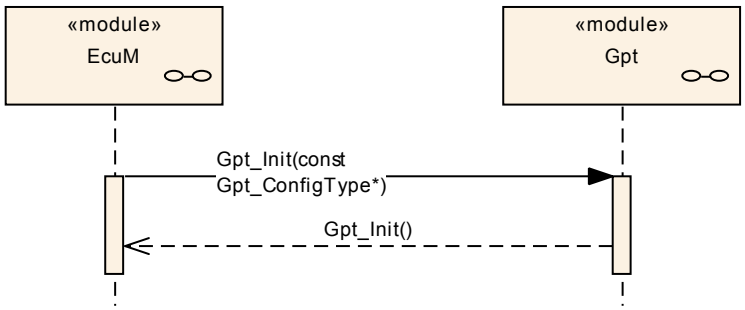

Figure 7: Sequence Diagram - Gpt_Init

9.2 GPT continuous mode

Channel 2 is configured as “Continuous Mode”

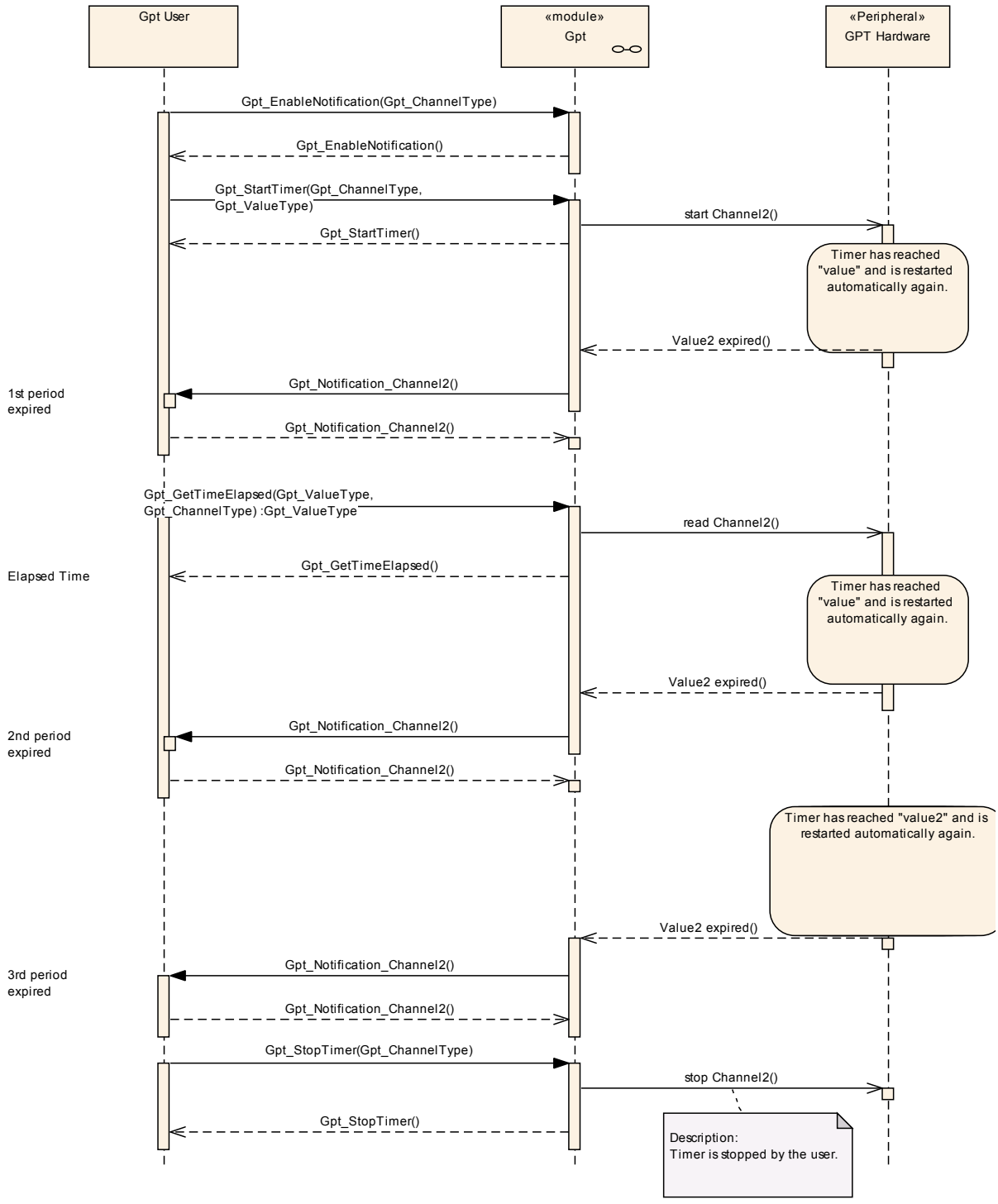

Figure 8: Sequence Diagram - GPT continuous mode

9.3 GPT one-shot mode

Channel 1 is configured for “One-shot Mode”

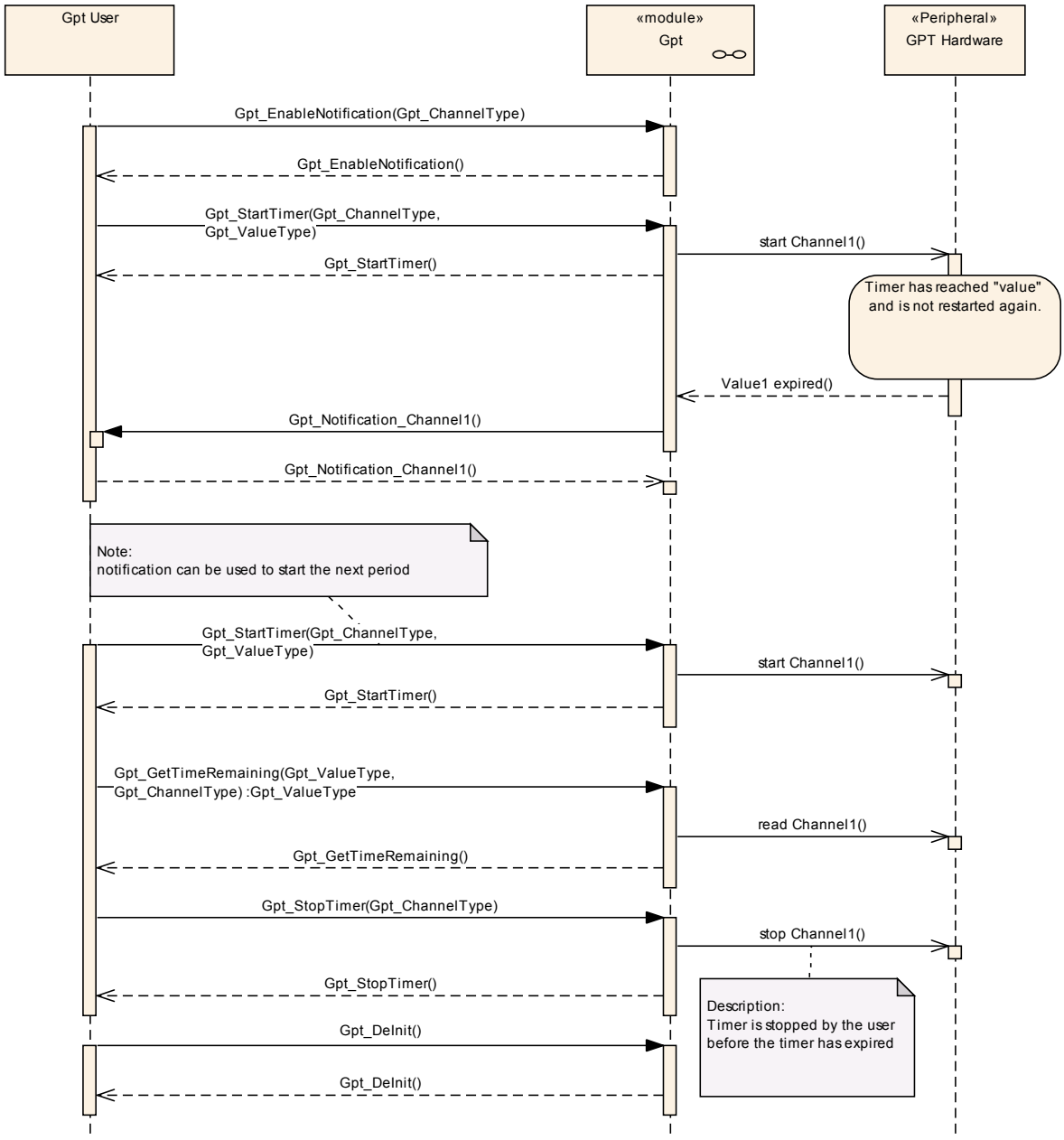

Figure 9: Sequence Diagram - GPT one-shot mode

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.

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

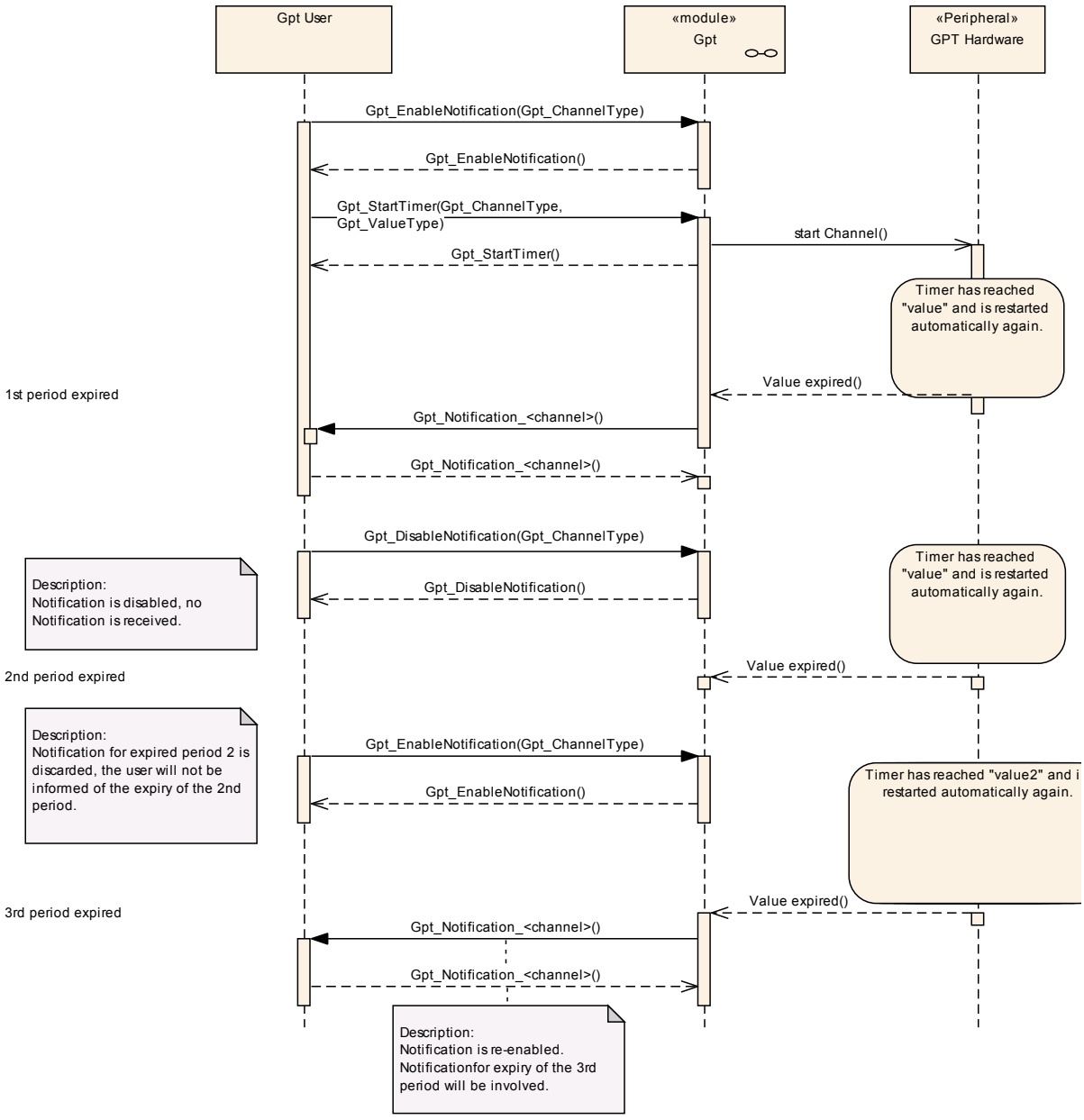

Figure 10: Sequence Diagram - Disable/Enable Notifications

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

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

- AUTOSAR Layered Software Architecture [2].
- AUTOSAR ECU Configuration Specification [6]
  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.
--	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 Chapter 7 and Chapter 8.

10.2.1 Variants

Configuration variants describe sets of configuration parameters:

- VARIANT-PRE-COMPILE (PC)
  Only parameters with "Pre-compile time" configuration are allowed in this variant.

- VARIANT-LINK-TIME (LT)
  Only parameters with "Pre-compile time" and "Link time" are allowed in this variant.

- VARIANT-POST-BUILD (PB)
  Parameters with "Pre-compile time", "Link time" and "Post-build time" are allowed in this variant.

[GPT342] ⌈At least one of the following variants has to be supported by implementation:
- VARIANT-PRE-COMPILE
- VARIANT-POST-BUILD⌋ (BSW00397, BSW00399, BSW00400)

[GPT257] ⌈The initialization function of this module shall always have a pointer as a parameter. For variant "Pre-compile time" (no pointer to configuration is available) a null pointer shall be passed. ⌋ (BSW00414)

[GPT270] ⌈Within one container it shall not be possible to mix parameters assigned to different configuration classes. ⌋ ( )

10.2.2 Gpt

Module Name	*Gpt*
Module Description	Configuration of the Gpt (General Purpose Timer) module.

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	**GPT183_Conf :**
Container Name	GptDriverConfiguration
Description	This container contains the module-wide configuration (parameters) of the GPT Driver
Configuration Parameters	

SWS Item	**GPT321_Conf :**		
Name	GptDevErrorDetect {GPT_DEV_ERROR_DETECT}		
Description	Enables/Disables development error detection.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**GPT322_Conf :**		
Name	GptReportWakeupSource {GPT_REPORT_WAKEUP_SOURCE}		
Description	Enables/Disables wakeup source reporting.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

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).

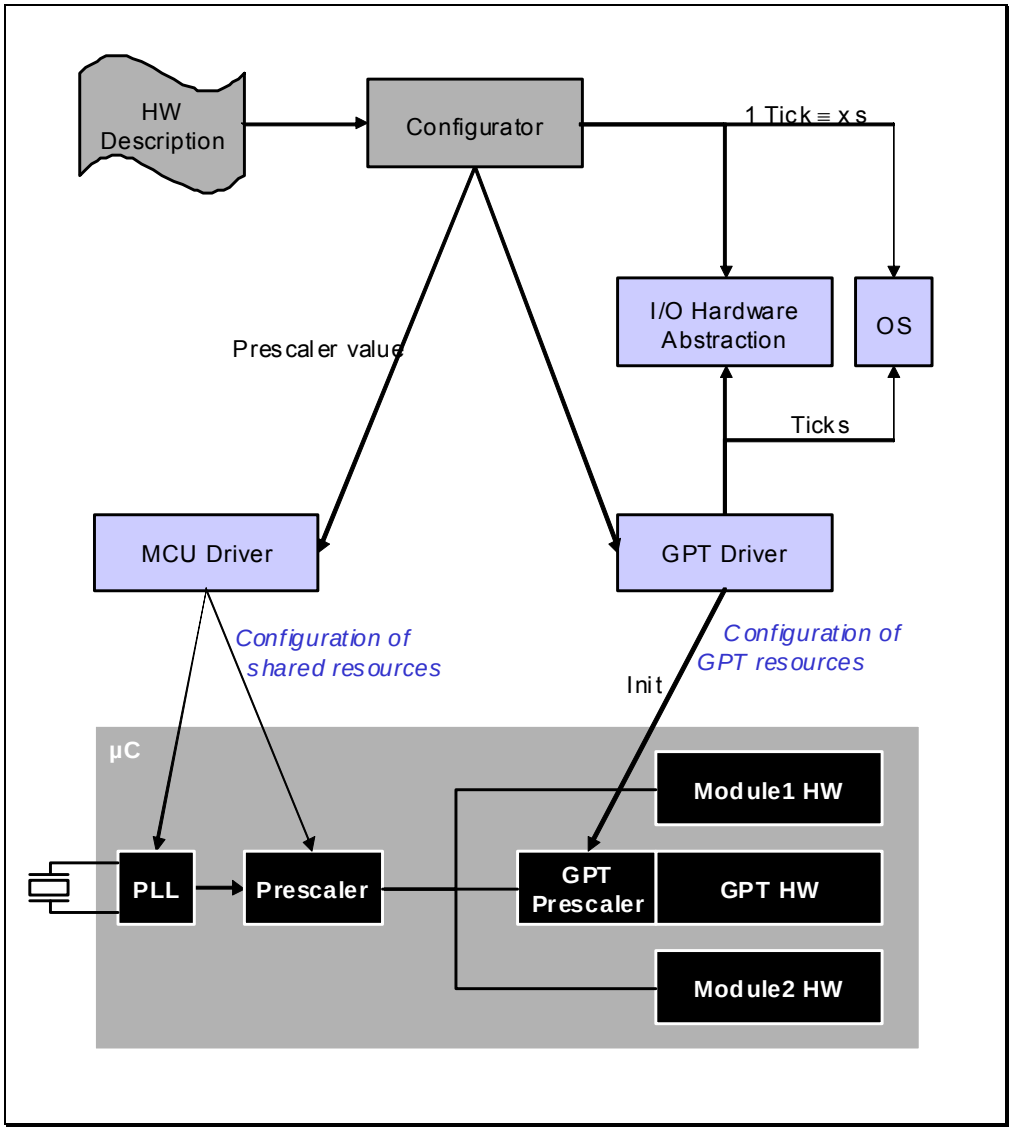

Figure 11: Scope of the GPT Driver configuration

10.2.4 GptClockReferencePoint

SWS Item	**GPT329_Conf :**
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	**GPT330_Conf :**
Name	GptClockReference {GPT_CLOCK_REFERENCE}
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]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.5 GptChannelConfigSet

SWS Item	**GPT269_Conf :**
Container Name	GptChannelConfigSet [Multi Config Container]
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	**GPT184_Conf :**
Container Name	GptChannelConfiguration
Description	Configuration of an individual GPT channel.
Configuration Parameters	

SWS Item	**GPT308_Conf :**		
Name	GptChannelId {GPT_CHANNEL_ID}		
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	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Instance		

SWS Item	**GPT309_Conf :**		
Name	GptChannelMode {GPT_CHANNEL_MODE}		
Description	Specifies the behaviour 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).	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Instance		

SWS Item	**GPT331_Conf :**		
Name	GptChannelTickFrequency		
Description	Specifies the tick frequency of the timer channel in Hz.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0 .. INF`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**GPT332_Conf :**		
Name	GptChannelTickValueMax		
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	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**GPT311_Conf :**
Name	GptEnableWakeup {GPT_ENABLE_WAKEUP}
Description	Enables wakeup capability of MCU for a channel.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	--

ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Instance		

SWS Item	**GPT312_Conf :**		
Name	GptNotification {GPT_NOTIFICATION}		
Description	Function pointer to callback function (for non-wakeup notification)		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Instance		

SWS Item	**GPT333_Conf :**		
Name	GptChannelClkSrcRef {GPT_CHANNEL_CLKSRC_REF}		
Description	Reference to the GptClockReferencePoint from which the channel clock is derived.		
Multiplicity	1		
Type	Reference to [GptClockReferencePoint]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Instance		

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	**GPT235_Conf :**
Container Name	GptWakeupConfiguration{GPT_WAKEUP_CONFIGURATION}
Description	Function pointer to callback function (for non-wakeup notification).
Configuration Parameters	

SWS Item	**GPT313_Conf :**		
Name	GptWakeupSourceRef {GPT_WAKEUP_SOURCE_REF}		
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	Reference to [EcuMWakeupSource]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: instance

No Included Containers

10.2.8 GptConfigurationOfOptApiServices

SWS Item	**GPT193_Conf :**
Container Name	GptConfigurationOfOptApiServices{Configuration of optional API services}
Description	This container contains all configuration switches for configuring optional API services of the GPT driver.
Configuration Parameters	

SWS Item	**GPT314_Conf :**		
Name	GptDeinitApi {GPT_DEINIT_API}		
Description	Adds / removes the service Gpt_DeInit() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**GPT315_Conf :**		
Name	GptEnableDisableNotificationApi {GPT_ENABLE_DISABLE_NOTIFICATION_API}		
Description	Adds / removes the services Gpt_EnableNotification() and Gpt_DisableNotification from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**GPT317_Conf :**		
Name	GptTimeElapsedApi {GPT_TIME_ELAPSED_API}		
Description	Adds / removes the service Gpt_GetTimeElapsed() from the code		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**GPT318_Conf :**
Name	GptTimeRemainingApi {GPT_TIME_REMAINING_API}
Description	Adds / removes the service Gpt_GetTimeRemaining() from the code.
Multiplicity	1
Type	EcucBooleanParamDef

Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**GPT319_Conf :**		
Name	GptVersionInfoApi {GPT_VERSION_INFO_API}		
Description	Adds / removes the service Gpt_GetVersionInfo() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**GPT320_Conf :**		
Name	GptWakeupFunctionalityApi {GPT_WAKEUP_FUNCTIONALITY_API}		
Description	Adds / removes the services Gpt_SetMode(), Gpt_EnableWakeup() Gpt_DisableWakeup() and Gpt_CheckWakeup() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.3 Published Information

[GPT380] ⌈The standardized common published parameters as required by BSW00402 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 Changes to Release 3.0/3.1

This chapter describes changes between Release 3.0/3.1 and Release 4.0 Rev1.

11.1 Deleted SWS Items

SWS Item	*Rationale*
GPT262	SWS item not necessary
GPT171	First sentence not a requirement
GPT173	Not a requirement on the GPT driver, only a comment.
GPT204	Improvement, already covered by GPT178
GPT055	Already covered by GPT274
GPT272	Already covered by GPT294
GPT161	Not a stand alone requirement (moved to GPT008)
GPT296	Behaviour rectified
GPT298	Behaviour rectified
GPT302	Behaviour rectified
GPT304	Behaviour rectified
GPT103	Behaviour rectified
GPT291	Not a requirement on the GPT driver, only a comment
GPT232	Not a requirement on the GPT driver, only a comment.
GPT207	Already covered by variant post build
GPT087	Not a requirement on the GPT driver, only a comment
GPT276	Not a requirement on the GPT driver
GPT277	Not a requirement on the GPT driver

11.2 Replaced SWS Items

SWS Item	*replaced by SWS Item*	*Rationale*
GPT001	GPT345, GPT346, GPT347, GPT348, GPT349, GPT350, GPT351	Requirement GPT001 splitted
GPT205	GPT352, GPT353, GPT354, GPT355, GPT356	Requirement GPT205 splitted
GPT183	GPT183_Conf	Postfix "_Conf" added to SWS Items in tables generated out of AUTOSAR_MetaModel (for chapter 10). * Note: The SWS Items marked with * are present anymore (outside chapter 10), because they are used duplicate in R3.0/R.3.1.
GPT321 *	GPT321_Conf	
GPT322 *	GPT322_Conf	
GPT269	GPT269_Conf	
GPT184	GPT184_Conf	
GPT307 *	GPT307_Conf	
GPT308 *	GPT308_Conf	
GPT309 *	GPT309_Conf	
GPT310*	GPT310_Conf	
GPT311	GPT311_Conf	
GPT312	GPT312_Conf	
GPT235	GPT235_Conf	
GPT313	GPT313_Conf	
GPT193	GPT193_Conf	
GPT314	GPT314_Conf	
GPT315	GPT315_Conf	
GPT317	GPT317_Conf	

GPT318	GPT318_Conf	
GPT319	GPT319_Conf	
GPT320	GPT320_Conf	
GPT307_Conf	GPT333_Conf	Improvement
GPT310_Conf	GPT331_Conf	Improvement

11.3 Changed SWS Items

SWS Item	*Rationale*
GPT172	Improvement
GPT293	Improvement
GPT185	Improvement (moved to chapter 7 and rephrased)
GPT186	Improvement (moved to chapter 7 and rephrased)
GPT127	Improvement
GPT256	Rectified
GPT176	Improvement
GPT177	Improvement
GPT179	Improvement
GPT278	Rectified: EcuM_Types.h -> EcuM.h, Improvement
GPT181	Improvement
GPT107	Improvement
GPT307	Improvement
GPT258	Improvement
GPT294	Improvement
GPT008	Improvement
GPT105	Improvement
GPT234	Improvement
GPT220	Improvement
GPT010	Improvement
GPT295	Improvement
GPT297	Improvement
GPT299	Behaviour rectified
GPT113	Not a requirement on the GPT driver
GPT222	Improvement
GPT210	Improvement
GPT083	Improvement
GPT301	Improvement
GPT303	Behaviour rectified
GPT305	Improvement
GPT114	Not a requirement on the GPT driver
GPT223	Improvement
GPT211	Improvement
GPT274	Improvement
GPT275	Improvement
GPT115	Not a requirement on the GPT driver
GPT212	Improvement
GPT218	Improvement
GPT224	Improvement
GPT084	Improvement
GPT099	Improvement
GPT116	Not a requirement on the GPT driver
GPT213	Improvement
GPT225	Improvement
GPT014	Improvement
GPT117	Not a requirement on the GPT driver

GPT226	Improvement
GPT214	Improvement
GPT287	Improvement of Description
GPT015	Improvement
GPT118	Not a requirement on the GPT driver
GPT227	Improvement
GPT217	Improvement
GPT288	Improvement of Parameter description
GPT152	Improvement
GPT153	Improvement
GPT164	Improvement
GPT165	Improvement
GPT228	Improvement
GPT201	Reference to chapter 10 removed
GPT289	Improvement of Description
GPT159	Improvement
GPT155	Not a requirement on the GPT driver
GPT215	Improvement
GPT229	Improvement
GPT290	Improvement of Description
GPT160	Improvement
GPT156	Not a requirement on the GPT driver
GPT203	Reference to chapter 10 removed
GPT230	Improvement
GPT216	Improvement
GPT323	Improvement
GPT324	Reference to chapter 10 removed
GPT325	Improvement
GPT292	Typo (dependant -> dependent)
GPT086	Typo (GptNotification... -> Gpt_Notification...)
GPT093	Second part of requirement not necessary
GPT233	Improvement
GPT206	Improvement
GPT257	Improvement
GPT255	Improvement
GPT201	Name of configuration parameter corrected
GPT328 GPT321 GPT322 GPT323 GPT324 GPT325	Name of API service changed: Gpt_Cbk_CheckWakeup -> Gpt_CheckWakeup

11.4 Added SWS Items

SWS Item	*Rationale*
GPT365	Improvement
GPT367	Improvement
GPT368	Improvement
GPT369	Improvement
GPT371	Improvement
GPT372	Improvement
GPT373	Improvement
GPT374	Improvement
GPT375	Improvement

GPT329	Improvement
GPT330	Improvement
GPT331	Improvement
GPT332	Improvement
GPT333	Debugging Concept incorporated
GPT334	Debugging Concept incorporated
GPT335	Debugging Concept incorporated
GPT336	Debugging Concept incorporated
GPT337	Debugging Concept incorporated
GPT357	Requirement without SWS Item
GPT358	Requirement without SWS Item
GPT359	Requirement without SWS Item
GPT360	Requirement without SWS Item
GPT338	Improvement
GPT339	Improvement
GPT340	Improvement
GPT363	Improvement
GPT361	Improvement
GPT364	Improvement
GPT343	Improvement
GPT344	Improvement
GPT341	Improvement
GPT362	Not a comment, but a requirement
GPT342	Improvement
GPT329_Conf	Clock Tree Configuration incorporated
GPT330_Conf	Clock Tree Configuration incorporated
GPT376	Improvement
GPT377	Improvement
GPT378	Improvement
GPT379	Improvement
GPT331_Conf	Improvement
GPT332_Conf	Improvement
GPT333_Conf	Improvement
GPT380	Rework of Published Information

12 Changes to Release 4.0 Rev 1

12.1 Deleted SWS Items

SWS Item	*Rationale*
GPT208	Multiplicity of configuration parameter GptNotification set from 1 to 0..1, so this requirement is no longer needed
GPT376	Requirement is needless and not testable.
GPT378	Requirement is needless and not testable

12.2 Replaced SWS Items

SWS Item	*replaced by SWS Item*	*Rationale*

12.3 Changed SWS Items

SWS Item	*Rationale*
GPT256	Updated according to BSW004
GPT312_Conf	New handling of unused notification (multiplicity changed from 1 to 0..1)
GPT331_CONF	Range added

12.4 Added SWS Items

SWS Item	*Rationale*

13 Not applicable requirements

[GPT381] ⌈These requirements are not applicable to this specification.⌋ (BSW00344, BSW159, BSW167, BSW170, BSW00398, BSW00416, BSW00437, BSW168, BSW168, BSW00423, BSW00424, BSW00425, BSW00426, BSW00427, BSW00428, BSW00429, BSW00432, BSW00433, BSW00422, BSW00417, BSW161, BSW162, BSW005, BSW00415, BSW00325, BSW00326, BSW00342, BSW160, BSW007, BSW00413, BSW00347, BSW00307, BSW00373, BSW00335, BSW00348, BSW00353, BSW00361, BSW00328, BSW006, BSW00439, BSW00357, BSW00377, BSW00378, BSW00306, BSW00308, BSW00309, BSW00376, BSW00359, BSW00360, BSW00440, BSW00330, BSW00331, BSW009, BSW172, BSW010, BSW00333, BSW00321, BSW00341, BSW00334, BSW12462, BSW12463, BSW12068, BSW12075, BSW12064, BSW12077, BSW12078, BSW12092, BSW12265)