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

Document Version	2.2.3
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change Description
28.02.2014	2.2.3	AUTOSAR Release Management	Editorial changes Removed chapter(s) on change documentation
23.03.2011	2.2.2	AUTOSAR Administration	Legal disclaimer revised
23.06.2008	2.2.1	AUTOSAR Administration	Legal disclaimer revised
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

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

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
Timer tick	Defines the timer resolution

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,
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture,
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules,
AUTOSAR_SRS_General.pdf

[4] Specification of Development Error Tracer,
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[5] Specification of ECU Configuration,
AUTOSAR_ECU_Configuration.pdf

[6] Specification of Diagnostic Event Manager,
AUTOSAR_SWS_DEM.pdf

[7] Specification of ECU State Manager,
AUTOSAR_SWS_ECU_StateManager.pdf

[8] General Requirements on SPAL,
AUTOSAR_SRS_SPAL_General.pdf

[9] Requirements on GPT Driver,
AUTOSAR_SRS_GPT_Driver.pdf

[10] Specification of ICU Driver,
AUTOSAR_SWS_ICU_Driver.pdf

[11] Specification of MCU Driver,
AUTOSAR_SWS_MCU_Driver.doc

[12] Specification of I/O Hardware Abstraction,
AUTOSAR_SWS_IOHW_Abstraction.pdf

[13] Glossary,
AUTOSAR_Glossary.pdf

[14] AUTOSAR Basic Software Module Description Template,
AUTOSAR_BSW_Module_Description.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 completely. At this point it shall be pointed out that the code-file structure shall include the file named "`Gpt_PBcfg.c`" for post build time configurable parameters. This file shall contain all post-build time configurable parameters.

5.1.2 Header file structure

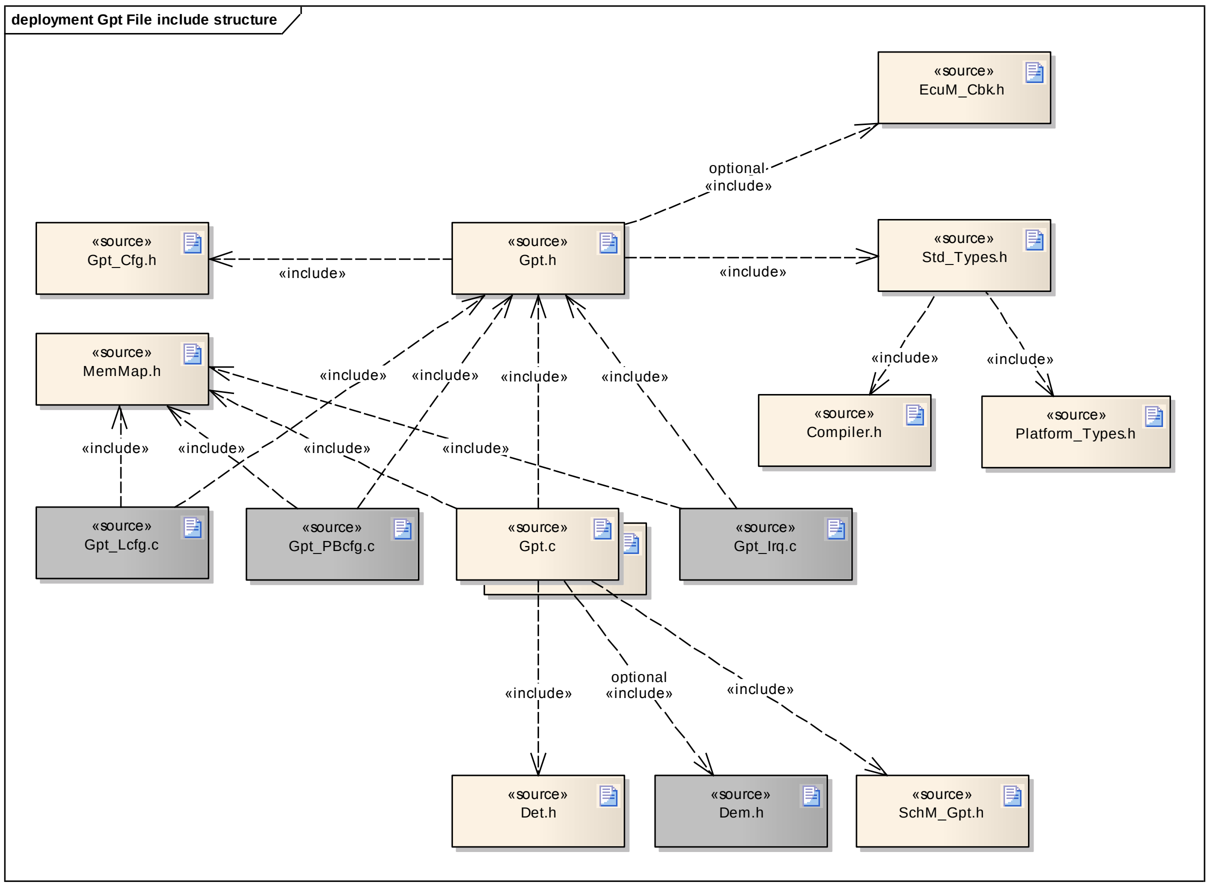

Figure 1: Header file structure

GPT172: The module shall optionally include the Dem.h file if any production error will be issued by the implementation. 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

GPT293: `Gpt.c` shall include `Gpt.h`

`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.

GPT262: The file `Gpt.h` shall contain the type definitions for `Gpt_Lcfg.c` and `Gpt_PBcfg.c`.

GPT271: `Gpt.h` shall include `EcuM_Cbk.h`, 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 section 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)
[BSW00438] Post Build Configuration Data Structure	Gpt_Init function is called by Ecu State Manager directly
[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
[BSW00437] NoInit--Area in RAM	Not applicable (requirement on implementation)
[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)

Requirement	***Satisfied by***
[BSW00336] Shutdown interface	GPT008
[BSW00337] Classification of errors	GPT001, GPT004
[BSW00338] Detection and Reporting of development errors	GPT178, GPT204
[BSW00369] Do not return development error codes via API	GPT178, GPT179, GPT204
[BSW00339] Reporting of production relevant error status	GPT179
[BSW00422] Pre--de--bouncing of production relevant error status	Not applicable (requirement on module "Diagnostic Event Manager")
[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)
[BSW00441] Enumeration literals and #define naming convention	GPT189, GPT001, GPT178, section 8.2.4, GPT185, GPT186
[BSW00305] Self-defined data types naming convention	See section 8.2

Requirement	*Satisfied by*
[BSW00307] Global variables naming convention	Not applicable (requirement on implementation)
[BSW00310] API naming convention	See section 8.3
[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 10.2.5
[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
[BSW00435] Module 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
[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 0, 0, 8.3.6, 8.3.7, 0, 0, 0, 0
[BSW006] Platform independency	Not applicable (module is not above MCAL)
[BSW00439] Declaration of interrupt handlers and ISRs	Not applicable (requirement on implementation)
[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 10.2.2
[BSW00306] Avoid direct use of compiler and platform specific keywords	See Figure 1

Requirement	***Satisfied by***
[BSW00308] Definition of global data	Not applicable (requirement for the implementer)
[BSW00309] Global data with read-only constraint	See section 0
[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 0
[BSW00414] Parameter of init function	GPT257
[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
[BSW00360] Parameters of callback functions	See section 8.6
[BSW00440] Function prototype for callback functions of AUTOSAR Services	Not applicable (no AUTOSAR RTE callback services provided)
[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)

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 0
[BSW12448] Behavior after development error detection	GPT178, GPT234, GPT296, GPT302, 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, GPT308
[BSW12117] Read timer value	GPT083, GPT010
[BSW12128] Start timer	GPT274, GPT275, 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.

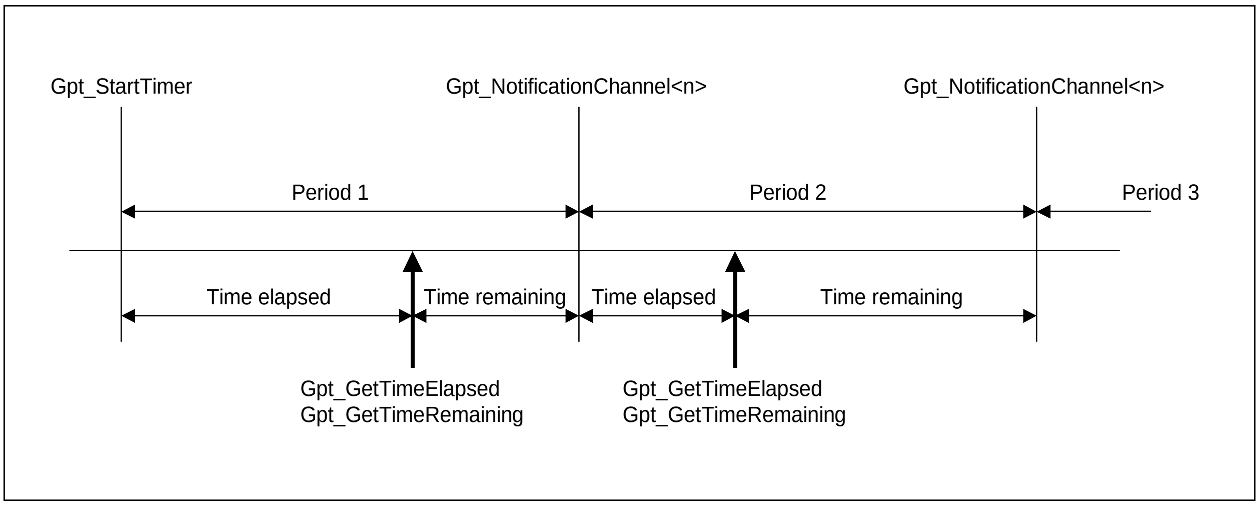

Figure 2: Querying the elapsed/remaining time

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.

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

For a detailled description on wakeup handling please refer to the ECU State Manager specification [7].

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 State transitions

The state chart below (Figure 3) shows the behavior when calling GPT services on channels in different operational states.

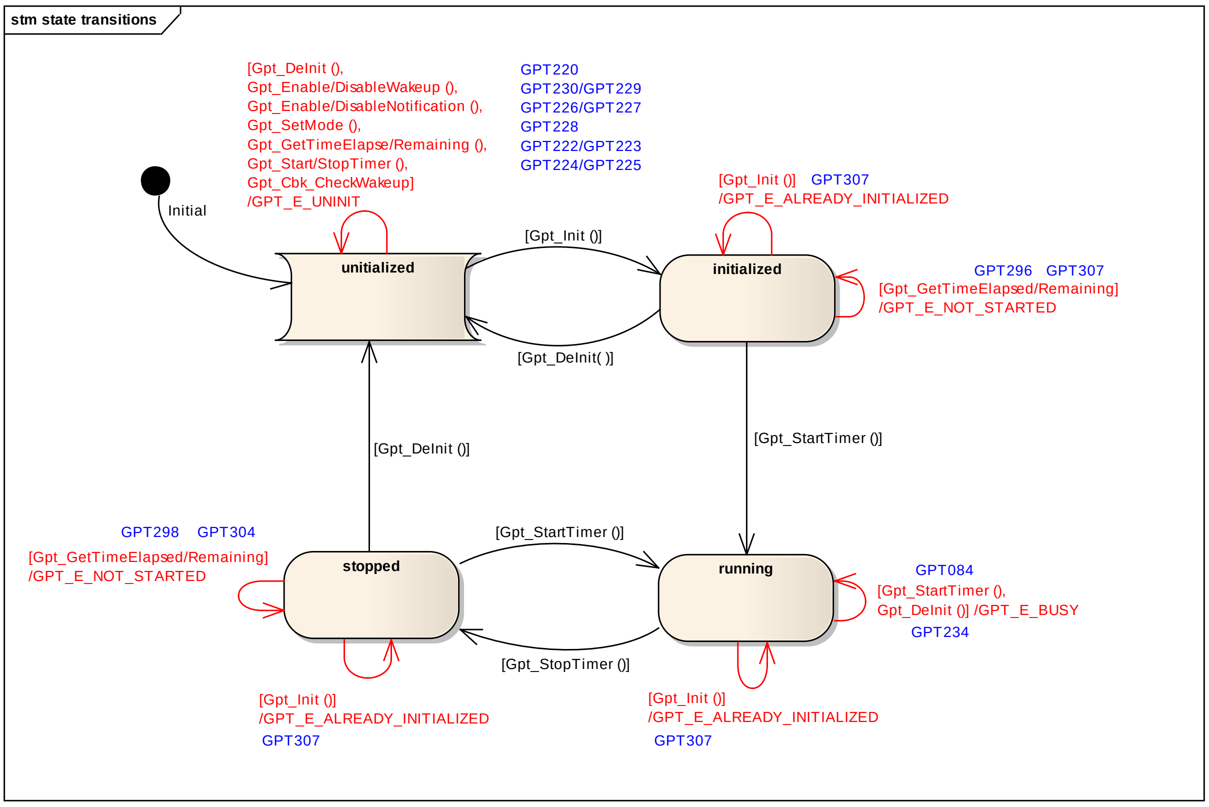

Figure 3: Channel state transitions

Notes:
- notation: `[function call]/error code`
- transitions in red font show the bahavior with development (DET) error detection turned on.
- the figure only shows the development error detection for functions called in the wrong sequence. It is not refelcted in this figure, if i.e. `Gpt_StartTimer()` is callled with an invalid parameter.
- if a function call is not explicitly shown in a state, it is assumed that this function call neither leads to a DET report nor to a state transition of the specific channel

7.1.3 Version checking

GPT256: The Gpt module shall implement compile-time mechanisms that check the following:

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, check wakeup) 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.	Development	`GPT_E_NOT_STARTED`	`0x0C`

Type of error	***Relevance***	***Related error code***	***Value [hex]***
API Gpt_Init service called while the GPT driver has already been initialized	Development	`GPT_E_ALREADY_INITIALIZED`	`0x0D`
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 `GptDevErrorDetect` (see chapter 10) shall activate or deactivate the detection of all development errors.

GPT176: If the `GptDevErrorDetect` 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	None	`None`
Init	ConfigPtr = NULL called again	`GPT_E_PARAM_CONFIG` `GPT_E_ALREADY_INITIALIZED`
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	`GPT_E_UNINIT` `GPT_E_NOT_STARTED`

Function Gpt_	Criteria of detection	Related error code
	called on a stopped channel 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`

Cbk_CheckWakeup	called prior to initialization	`GPT_E_UNINIT`

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 `GptDevErrorDetect` 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:

GPT278:

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

8.2 Type Definitions

8.2.1 Gpt_ConfigType

Name:	`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

Name:	`Gpt_ChannelType`		
Type:	`Unsigned Integer`		
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

Name:	`Gpt_ValueType`		
Type:	`Unsigned Integer`		
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

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 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.

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

GPT279:

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 (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Returns the version information of this module.	

GPT181: The function `Gpt_GetVersionInfo` shall return the version information of this module. The version information includes:
- Module Id
- Vendor Id
- Vendor specific version numbers (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`

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.	

GPT006: The function `Gpt_Init` shall initialize the hardware timer module according to a configuration set referenced by `ConfigPtr`.

GPT272: For variants with no postbuild multiple selectable configuration parameters (Variant PC), the GPT module’s environment shall pass a `NULL` pointer to the function Gpt_Init (see also GPT257).

GPT107: The function Gpt_Init shall disable all notifications.

GPT068: The function Gpt_Init shall only initialize the configured resources. Resources that are not configured in the configuration file shall not be touched.

GPT205: The GPT Driver shall apply the following rules regarding initialization of controller registers:
[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

GPT307: If development error detection for the GPT module is enabled: if called when the GPT driver and hardware are already initialized, the function `Gpt_Init` shall raise development error `GPT_E_ALREADY_INITIALIZED` and return without any action.

GPT258: The function `Gpt_Init` shall disable the wakeup interrupt invocation of all channels after the function call.

GPT294: If development error detection for the GPT module is enabled: if the function `Gpt_Init` is called with a `NULL configPtr` and if a variant containing postbuild multiple selectable configuration parameters is used (Variant PB), the function `Gpt_Init` shall raise the development error `GPT_E_PARAM_CONFIG` and return without any action.

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(`
	
	`)`

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.

GPT008: The function `Gpt_DeInit` shall deinitialize all hardware timer channels used by the configuration to their power on reset state. It's the responsibility of the hardware design that the state does not lead to undefined activities in the µC.

GPT161: Values of registers which are not writable shall be excluded by the function `Gpt_DeInit`.

GPT105: The function `Gpt_DeInit` shall disable all notifications.

GPT162: The function `Gpt_DeInit` shall influence only the peripherals, which are allocated by the static configuration.

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()`.

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

GPT194: The function `Gpt_DeInit` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptDeinitApi`.

GPT234: If development error detection for the GPT module is enabled: if the function `Gpt_DeInit` is called and if any channel is in state running, the function `Gpt_DeInit` shall raise the development error `GPT_E_BUSY` and leave the desired deinitialization functionality without any action.

GPT220: If development error detection for the GPT module is enabled: if the function `Gpt_DeInit` is called before the GPT module was initialized, the function `Gpt_DeInit` shall raise the development error `GPT_E_UNINIT` and leave the desired deinitialization functionality without any action.

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:	Gets the time already elapsed.	

GPT010: The function `Gpt_GetTimeElapsed` shall query the time already elapsed. When the channel is in the mode “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). When the channel is in the mode “continuous mode”, the function `Gpt_GetTimeElapsed` shall return the timer value relative to the last timeout / the start of the channel .

GPT295: If the function `Gpt_GetTimeElapsed` is called prior to starting the specified timer channel, the function `Gpt_GetTimeElapsed` shall return the value “`0`”.

GPT297: If the function `Gpt_GetTimeElapsed` is called on a timer channel in stopped state (channel has been initialized, started and stopped by `Gpt_StopTimer`), the function `Gpt_GetTimeElapsed` shall return the value “`0`”.

The rationale of GPT295 and GPT297 is to have the same behaviour for a stopped channel like a never started channel.

GPT299: If the function `Gpt_GetTimeElapsed` is called on a channel configured for one shot mode after the timeout period has already expired, the function `Gpt_GetTimeElapsed` shall return the value "`0`".

GPT113: The GPT module's environment shall only use the re-entrant capability of the function `Gpt_GetTimeElapsed` if the GPT module's environment takes care that there is no simultaneous usage of the same channel.

GPT195: The function `Gpt_GetTimeElapsed` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptTimeElapsedApi`.

GPT222: If development error detection for the GPT module is enabled: if the function `Gpt_GetTimeElapsed` is called before the GPT module was initialized, the function `Gpt_GetTimeElapsed` shall raise the development error `GPT_E_UNINIT` and return with the value "`0`".

GPT210: If development error detection for the GPT module is enabled: if the parameter "`channel`" is invalid, the function `Gpt_GetTimeElapsed` shall raise the development error `GPT_E_PARAM_CHANNEL` and return with the value "`0`".

GPT296: If development error detection for the GPT module is enabled: if the function `Gpt_GetTimeElapsed` is called prior to starting the specified timer channel, the function `Gpt_GetTimeElapsed` shall raise the development error `GPT_E_NOT_STARTED` and behave according to GPT295.

GPT298: If development error detection for the GPT module is enabled: if the function `Gpt_GetTimeElapsed` is called on a stopped timer channel (channel has been initialized and started before), the function `Gpt_GetTimeElapsed` shall raise the development error `GPT_E_NOT_STARTED` and behave according to GPT297.

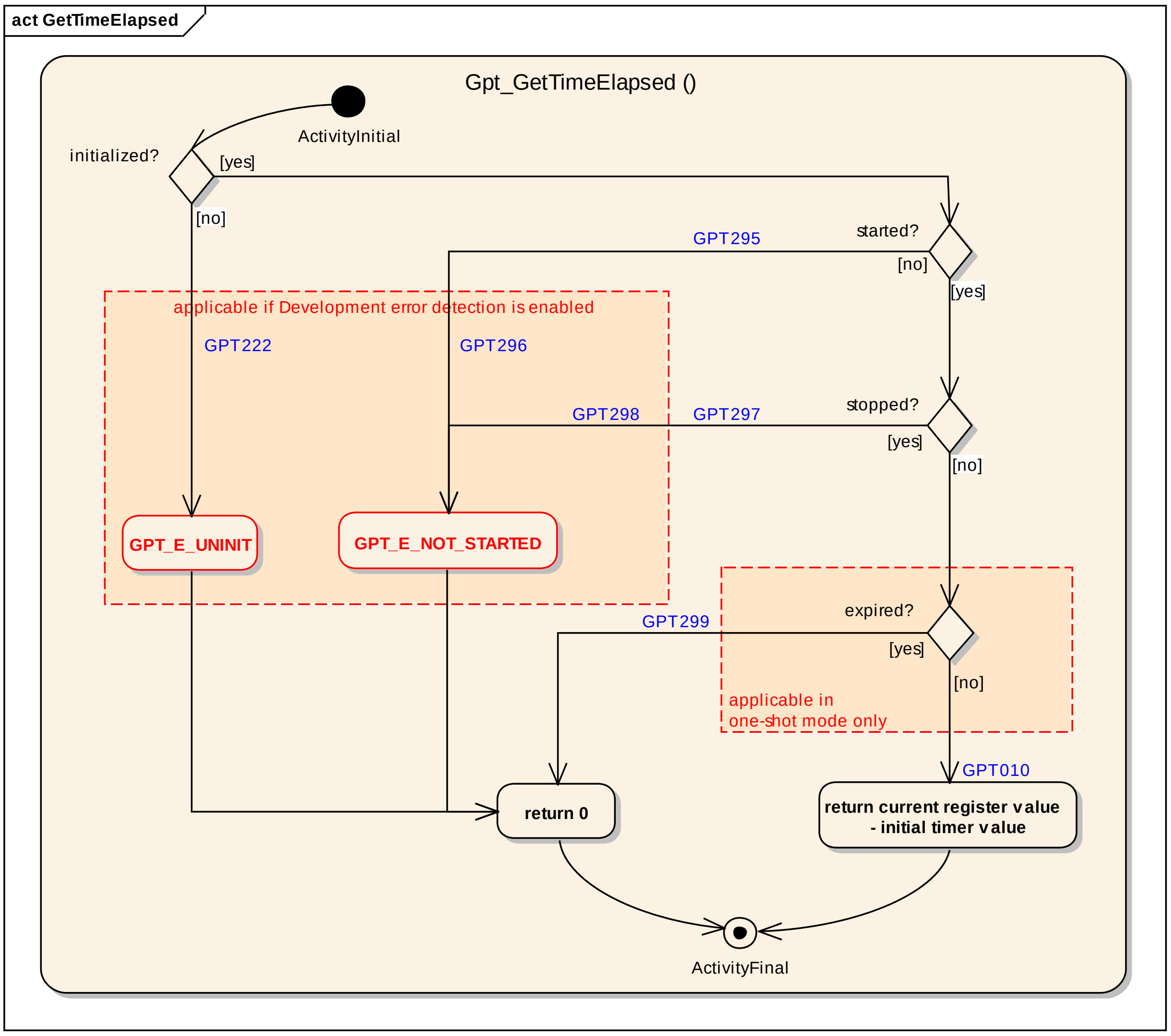

Figure 4: Overview on return and error values of Gpt_GetTimeElapsed

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

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:	Gets the time remaining until the next timeout period will expire.

GPT083: The function `Gpt_GetTimeRemaining` shall return 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).

GPT301: If the function `Gpt_GetTimeRemaining` is called prior to starting the specified timer channel, the function `Gpt_GetTimeRemaining` shall return the value “`0`”.

GPT303: If the function `Gpt_GetTimeRemaining` is called on a stopped timer channel (channel has been initialized and started before), the function `Gpt_GetTimeRemaining` shall return the value “`0`”.

The rationale of GPT301 and GPT303 is to have the same behaviour for a stopped channel like a never started channel.

GPT305: If the function `Gpt_GetTimeRemaining` is called on a channel configured for one shot mode, after the timeout period has already expired, the function `Gpt_GetTimeRemaining` shall return the value “`0`”.

GPT114: The GPT module’s environment shall only use the re-entrant capability of the function `Gpt_GetTimeRemaining` if the GPT module’s environment takes care that there is no simultaneous usage of the same channel.

GPT196: The function `Gpt_GetTimeRemaining` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptTimeRemainingApi`.

GPT223: If development error detection for the GPT module is enabled: if the function `Gpt_GetTimeRemaining` is called before the GPT module was initialized, the function `Gpt_GetTimeRemaining` shall raise the development error `GPT_E_UNINIT` and return with the value “`0`”.

GPT211: If development error detection for the GPT module is enabled: if the parameter `channel` is not within the allowed range (as specified by configuration), the function `Gpt_GetTimeRemaining` shall raise the development error `GPT_E_PARAM_CHANNEL` and return with the value “`0`”.

GPT302: If development error detection for the GPT module is enabled: if the function `Gpt_GetTimeRemaining` is called prior to starting the specified timer channel, the function `Gpt_GetTimeRemaining` shall raise the development error `GPT_E_NOT_STARTED` and behave according to GPT301.

GPT304: If development error detection for the GPT module is enabled: if the function `Gpt_GetTimeRemaining` is called on a stopped timer channel (channel has been initialized and started before), the function `Gpt_GetTimeRemaining` shall raise the development error `GPT_E_NOT_STARTED` and behave according to GPT303.

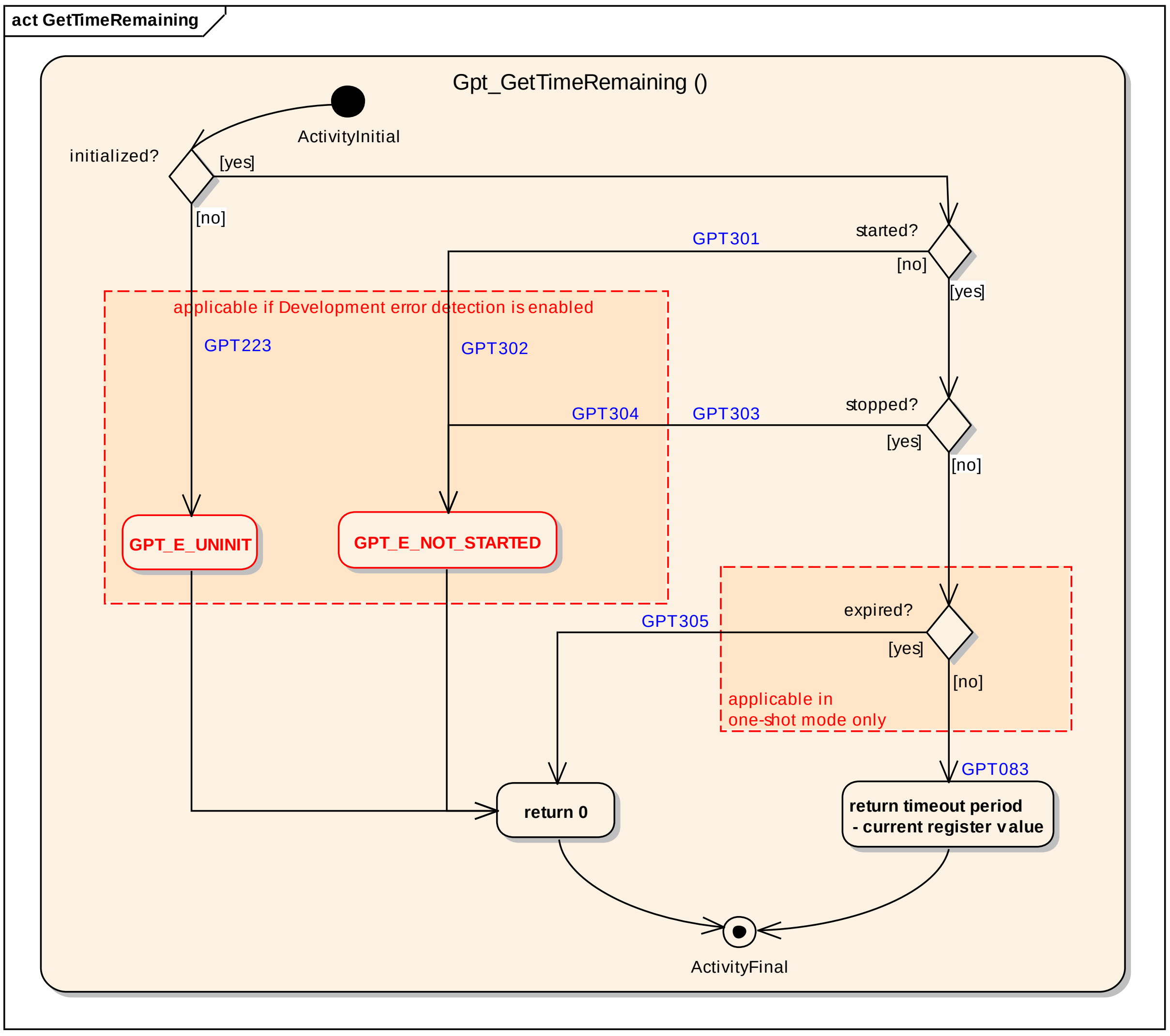

Figure 5: Overview on return and error values of Gpt_GetTimeRemaining

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

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	Timeout period (in number of ticks) after a notification shall occur.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Starts a timer channel.	

GPT274: The function `Gpt_StartTimer` shall start the selected timer channel with a defined timeout period.

GPT275: The function `Gpt_StartTimer` shall invoke the configured notification for that channel (see also GPT292) after the timeout period referenced via the parameter `value` (if enabled).

GPT115: The GPT module's environment shall only use the re-entrant capability of the function `Gpt_StartTimer` if the GPT module's environment takes care that there is no simultaneous usage of the same channel.

GPT212: If development error detection for the GPT module is enabled: the function `Gpt_StartTimer` shall raise the development error `GPT_E_PARAM_CHANNEL` if the parameter `channel` is not within the allowed range (as specified by configuration)..

GPT218: If development error detection for the GPT module is enabled: the function `Gpt_StartTimer` shall raise the development error `GPT_E_PARAM_VALUE` if the parameter `value` is not within the allowed range (exceeding the maximum timer resolution).

GPT224: If development error detection for the GPT module is enabled: if the function `Gpt_StartTimer` is called before the GPT module was initialized, the function `Gpt_StartTimer` shall raise the development error `GPT_E_UNINIT`.

GPT084: If development error detection for the GPT module is enabled: if the function `Gpt_StartTimer` is called on a channel, which is already started and still running, the function `Gpt_StartTimer` shall raise the development error `GPT_E_BUSY` and return without any action.

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.

GPT013: The function `Gpt_StopTimer` shall stop the selected timer channel .

GPT099: The function `Gpt_StopTimer` shall not raise a development error when the function `Gpt_StopTimer` stops a timer channel, which has not been started before.

GPT103: Timer channels configured in one shot mode have to be stopped explicitly by the user, when the timeout period has expired.

GPT116: The GPT module's environment shall only use the re-entrant capability of the function `Gpt_StopTimer` if the GPT module's environment takes care that there is no simultaneous usage of the same channel.

GPT213: If development error detection for the GPT module is enabled: if the parameter `channel` is not within the allowed range (as specified by configuration) , the function `Gpt_StopTimer` shall raise the development error `GPT_E_PARAM_CHANNEL`.

GPT225: If development error detection for the GPT module is enabled: if the function `Gpt_StopTimer` is called before the GPT module was initialized, the function `Gpt_StopTimer` shall raise the development error `GPT_E_UNINIT`.

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 (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Enables the notification for a channel.	

GPT014: The function `Gpt_EnableNotification` shall enable the invocation of the configured notification function (see also GPT233) for a channel.

GPT117: The GPT module's environment shall only use the re-entrant capability of the function `Gpt_EnableNotification` if the GPT module's environment takes care that there is no simultaneous usage of the same channel.

GPT199: The function `Gpt_EnableNotification` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptEnableDisableNotificationApi`

GPT226: If development error detection for the GPT module is enabled: if the function `Gpt_EnableNotification` is called before the GPT module was initialized, the function `Gpt_EnableNotification` shall raise the development error `GPT_E_UNINIT`.

GPT214: If development error detection for the GPT module is enabled: if the parameter `channel` is not within the allowed range (as specified by configuration), the function `Gpt_EnableNotification` shall raise the development 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 notification for a channel.	

GPT015: The function `Gpt_DisableNotification` shall disable the invocation of the configured notification function (see also GPT233) for a channel.

GPT118: The GPT module's environment shall only use the re-entrant capability of the function `Gpt_DisableNotification` if the GPT module's environment takes care that there is no simultaneous usage of the same channel.

GPT200: The function `Gpt_DisableNotification` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptEnableDisableNotificationApi`.

GPT227: If development error detection for the GPT module is enabled: if the function `Gpt_DisableNotification` is called before the GPT module was initialized, the function `Gpt_DisableNotification` shall raise the development error `GPT_E_UNINIT`.

GPT217: If development error detection for the GPT module is enabled: if the parameter `channel` is not within the allowed range (as specified by configuration), the function `Gpt_DisableNotification` shall raise the development 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 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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Sets the operation mode of the GPT.	

GPT151: The function `Gpt_SetMode` shall set the operation mode to the given mode parameter .

GPT255: The function `Gpt_SetMode` is only feasible if `GptReportWakeupSource` is statically configured available.

GPT152: If the parameter `mode` has the value `GPT_MODE_NORMAL`, the function `Gpt_SetMode` shall not affect the notifications as configured and selected by the `Gpt_DisableNotification` and `Gpt_EnableNotification`.

GPT153: If the parameter `mode` has the value `GPT_MODE_SLEEP`, the function `Gpt_SetMode` shall only enable the interrupts for those channels which are configured as wakeup capable and which are not disabled via the function `Gpt_DisableWakeup`.The function `Gpt_SetMode` shall disable all other interrupts 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: If the parameter `mode` has the value `GPT_MODE_SLEEP`, the function `Gpt_SetMode` shall stop all non-wakeup capable timer channels. Only those channels, which can serve as a wakeup source are running.

GPT165: If the parameter `mode` has the value `GPT_MODE_NORMAL` and the current mode is `GPT_MODE_SLEEP`, the function `Gpt_SetMode` shall not restart automatically the timer channels which have been stopped by entering the sleep mode.

GPT228: If development error detection for the GPT module is enabled: if the function `Gpt_SetMode` is called before the GPT module was initialized, the function `Gpt_SetMode` shall raise the development error `GPT_E_UNINIT`.

GPT231: If development error detection for the GPT module is enabled: the function `Gpt_SetMode` shall raise the development 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: `GPT_WAKEUP_FUNCTIONALITY _API` (see 10.2.5)

The function `Gpt_SetMode` influences the functionality of the GPT channels. Therefore the mode switching of the module shall be compatible to the overall state of the ECU.

The function `Gpt_SetMode` is affected by the configuration parameter `GptReportWakeupSource`.

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 invocation of a channel.	

GPT159: The function `Gpt_DisableWakeup` shall disable the wakeup interrupt invocation of a single GPT channel , referenced by the parameter `channel`.

GPT157: The function `Gpt_DisableWakeup` is only feasible, if `GptReportWakeupSource` is statically configured available.

GPT155: The GPT module’s environment shall only use the re-entrant capability of the function `Gpt_DisableNotification` if the GPT module’s environment takes care that there is no simultaneous usage of the same channel.

GPT202: The function `Gpt_DisableWakeup` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptWakeupFunctionalityApi`

GPT215: If development error detection for the GPT module is enabled: the function `Gpt_DisableWakeup` shall raise the development error `GPT_E_PARAM_CHANNEL` if the parameter `channel` is not within the allowed range (as specified by configuration) or on a non-wakeup capable channel.

GPT229: If development error detection for the GPT module is enabled: if the function `Gpt_DisableWakeup` is called before the GPT module was initialized, the function `Gpt_DisableWakeup` shall raise the development error `GPT_E_UNINIT`.

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 invocation of a channel.	

GPT160: The function `Gpt_EnableWakeup` shall re-enable the wakeup interrupt invocation of a single GPT channel, referenced by the parameter `channel`.

GPT158: The function `Gpt_EnableWakeup` is only feasible, if `GptReportWakeupSource` is statically configured available.

GPT156: The GPT module's environment shall only use the re-entrant capability of the function `Gpt_EnableWakeup` if the GPT module's environment takes care that there is no simultaneous usage of the same channel.

GPT203: The function `Gpt_EnableWakeup` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptWakeupFunctionalityApi` (see 10.2.5)

GPT230: If development error detection for the GPT module is enabled: if the function `Gpt_EnableWakeup` is called before the GPT module was initialized, the function `Gpt_EnableWakeup` shall raise the development error `GPT_E_UNINIT`.

GPT216: If development error detection for the GPT module is enabled: the function `Gpt_EnableWakeup` shall raise the development error `GPT_E_PARAM_CHANNEL` if the parameter `channel` is not within the allowed range (as specified by configuration) or on a non-wakeup capable channel.

8.3.13 Gpt_Cbk_CheckWakeup

GPT328:

Service name:	Gpt_Cbk_CheckWakeup	

Syntax:	`void Gpt_Cbk_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_Cbk_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 [7].

GPT322: The function `Gpt_Cbk_CheckWakeup` is only feasible, if `GptReportWakeupSource` is statically configured available.

GPT323: The GPT module's environment shall only use the re-entrant capability of the function `Gpt_Cbk_CheckWakeup` if the GPT module's environment takes care that there is no simultaneous usage of the same channel.

GPT324: The function `Gpt_Cbk_CheckWakeup` shall be pre compile time configurable `On/Off` by the configuration parameter: `GptWakeupFunctionalityApi` (see 10.2.5)

GPT325: If development error detection for the GPT module is enabled: if the function `Gpt_Cbk_CheckWakeup` is called before the GPT module was initialized, the function `Gpt_Cbk_CheckWakeup` shall raise the development error `GPT_E_UNINIT`.

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.

GPT291:

API function	***Description***
Dem_ReportErrorStatus	Reports errors to the DEM.
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).

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 GptNotification

GPT292:

Service name:	Gpt_Notification_<channel>
Syntax:	`void Gpt_Notification_<channel>(`

	)
Sync/Async:	Synchronous
Reentrancy:	GPT user implementation dependant.
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	--

GPT232notification prototype `GptNotification_<channel>` is for the notification callback function and shall be implemented by the user.

GPT207: The callback notifications shall be configurable as function pointers within the initialization data structure (`Gpt_ConfigType`).

GPT086: The callback notifications `GptNotification_<channel>` shall be configurable as pointers to user defined functions within the configuration structure .

GPT209: Each channel shall provide its own notification if configured.

GPT087: The GPT module's environment shall declare a separate notification for each channel to avoid parameter values and to improve runtime efficiency.

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

GPT093: When disabled, the GPT Driver will send no notification. When re-enabled again, the user will not be notified of events, occurred while notifications have been disabled.

GPT233 The GPT Driver shall invoke a notification whenever the defined time period of the channel has expired.

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.

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 synchronous and re-entrant.

9.1 Gpt_Init

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

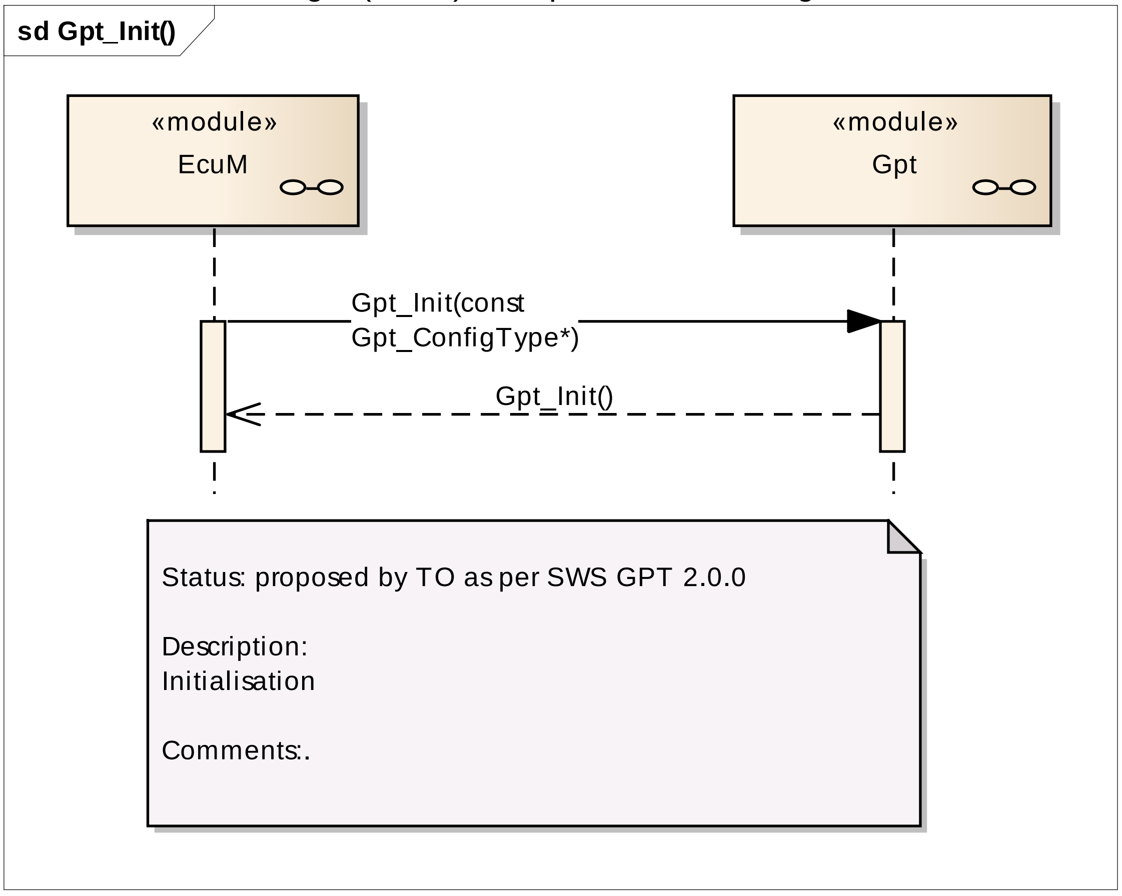

Figure 6: Sequence Diagram - Gpt_Init

9.2 GPT continuous mode

Channel 2 is configured as “Continuous Mode”

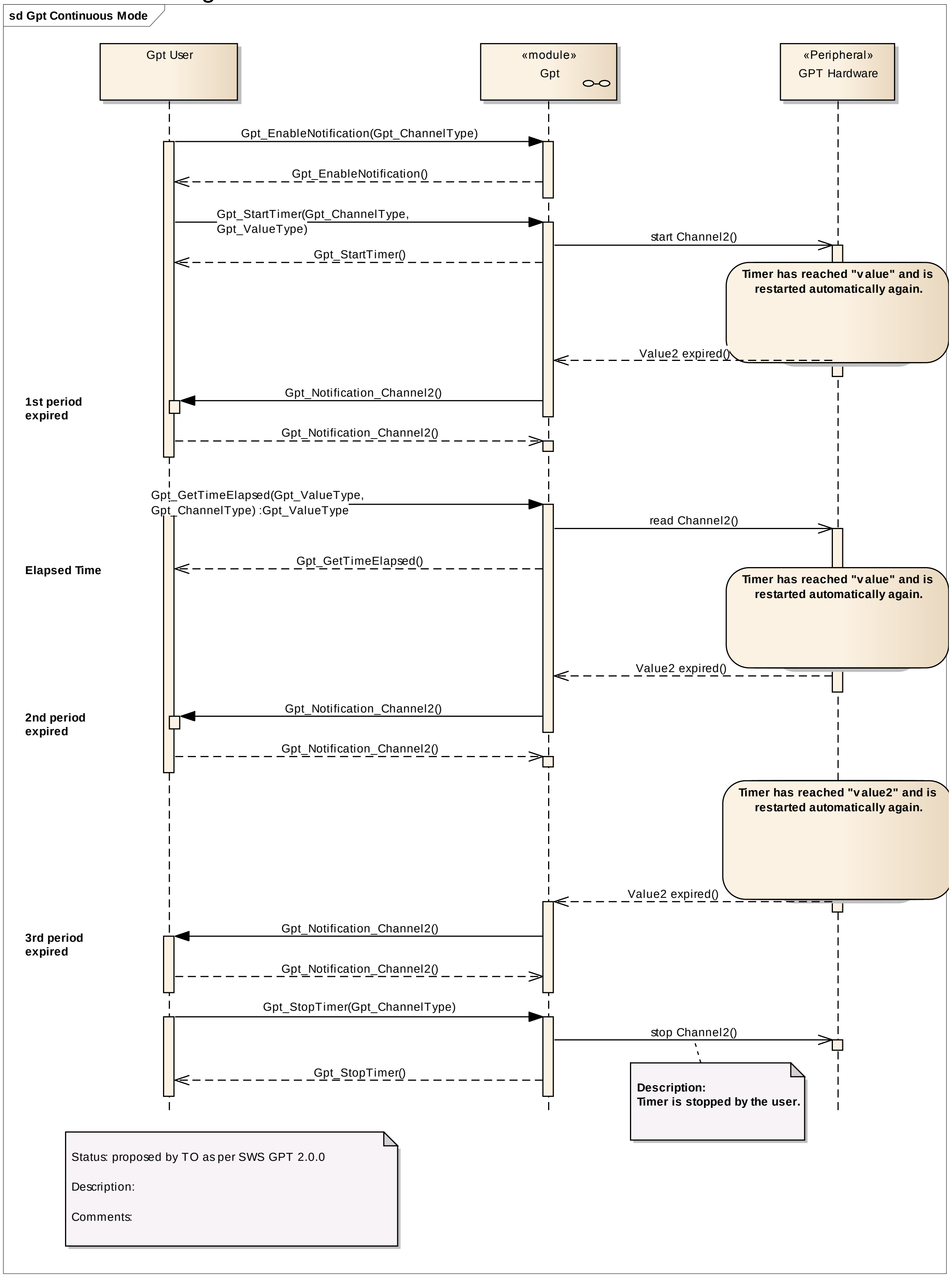
sd Gpt Continuous Mode
Gpt User
«module»
Gpt
«Peripheral»
GPT Hardware
Gpt_EnableNotification(Gpt_ChannelType)
Gpt_EnableNotification()
Gpt_StartTimer(Gpt_ChannelType, Gpt_ValueType)
start Channel2()
Gpt_StartTimer()
Timer has reached "value" and is restarted automatically again.
Value2 expired()
Gpt_Notification_Channel2()
1st period expired
Gpt_Notification_Channel2()
Gpt_GetTimeElapsed(Gpt_ValueType, Gpt_ChannelType) :Gpt_ValueType
read Channel2()
Gpt_GetTimeElapsed()
Elapsed Time
Timer has reached "value" and is restarted automatically again.
Value2 expired()
Gpt_Notification_Channel2()
2nd period expired
Gpt_Notification_Channel2()
Timer has reached "value2" and is restarted automatically again.
Value2 expired()
Gpt_Notification_Channel2()
3rd period expired
Gpt_Notification_Channel2()
Gpt_StopTimer(Gpt_ChannelType)
stop Channel2()
Gpt_StopTimer()
Description:
Timer is stopped by the user.
Status: proposed by TO as per SWS GPT 2.0.0
Description:
Comments:

Figure 7: Sequence Diagram - GPT continuous mode

9.3 GPT one shot mode

Channel 1 is configured for “One shot Mode”

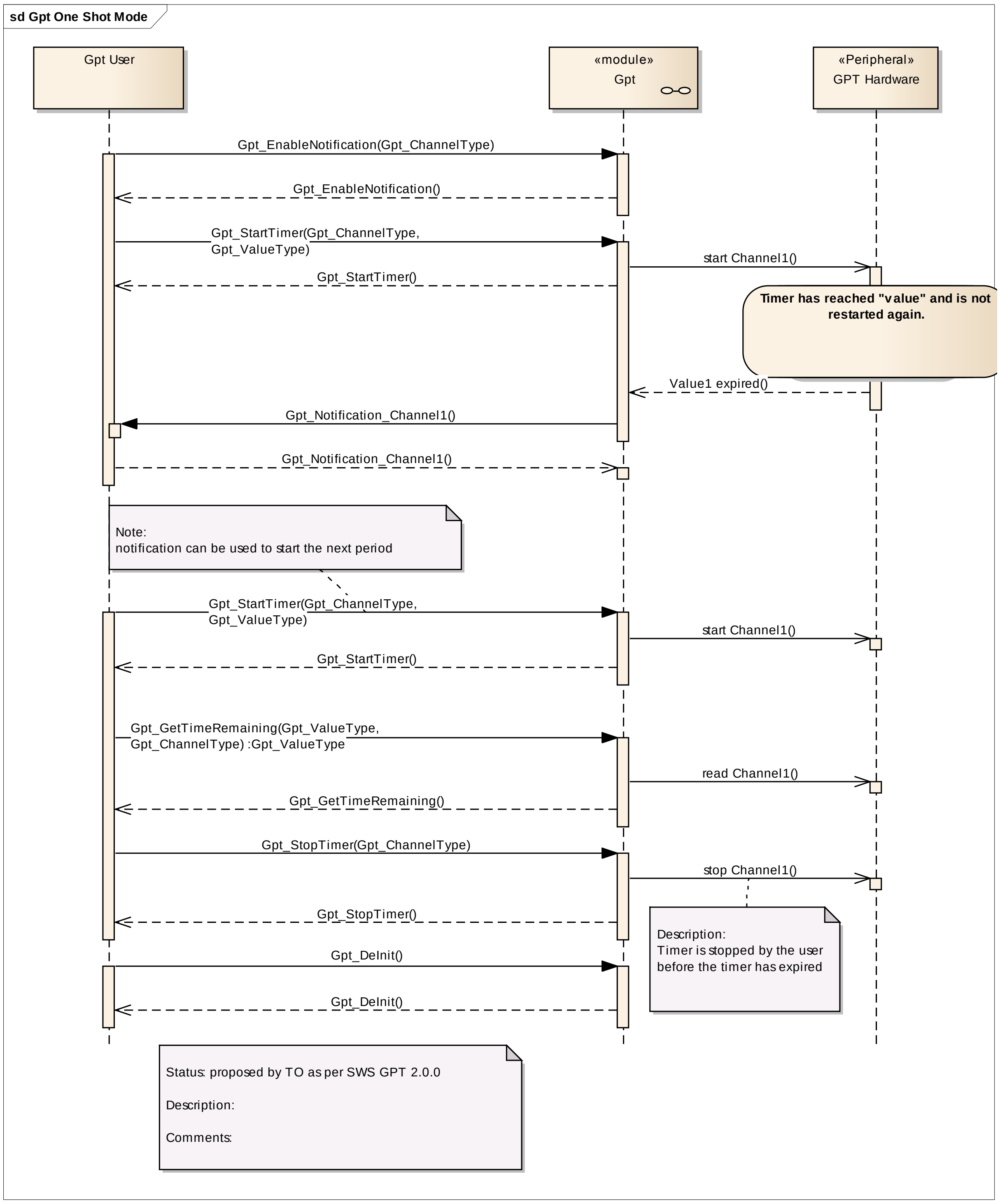
sd Gpt One Shot Mode
Gpt User
«module» Gpt
«Peripheral» GPT Hardware
Gpt_EnableNotification(Gpt_ChannelType)
Gpt_EnableNotification()
Gpt_StartTimer(Gpt_ChannelType, Gpt_ValueType)
start Channel1()
Gpt_StartTimer()
Timer has reached "value" and is not restarted again.
Value1 expired()
Gpt_Notification_Channel1()
Gpt_Notification_Channel1()
Note: notification can be used to start the next period
Gpt_StartTimer(Gpt_ChannelType, Gpt_ValueType)
start Channel1()
Gpt_StartTimer()
Gpt_GetTimeRemaining(Gpt_ValueType, Gpt_ChannelType) :Gpt_ValueType
read Channel1()
Gpt_GetTimeRemaining()
Gpt_StopTimer(Gpt_ChannelType)
stop Channel1()
Gpt_StopTimer()
Description: Timer is stopped by the user before the timer has expired
Gpt_DeInit()
Gpt_DeInit()
Status: proposed by TO as per SWS GPT 2.0.0
Description:
Comments:

Figure 8: 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.

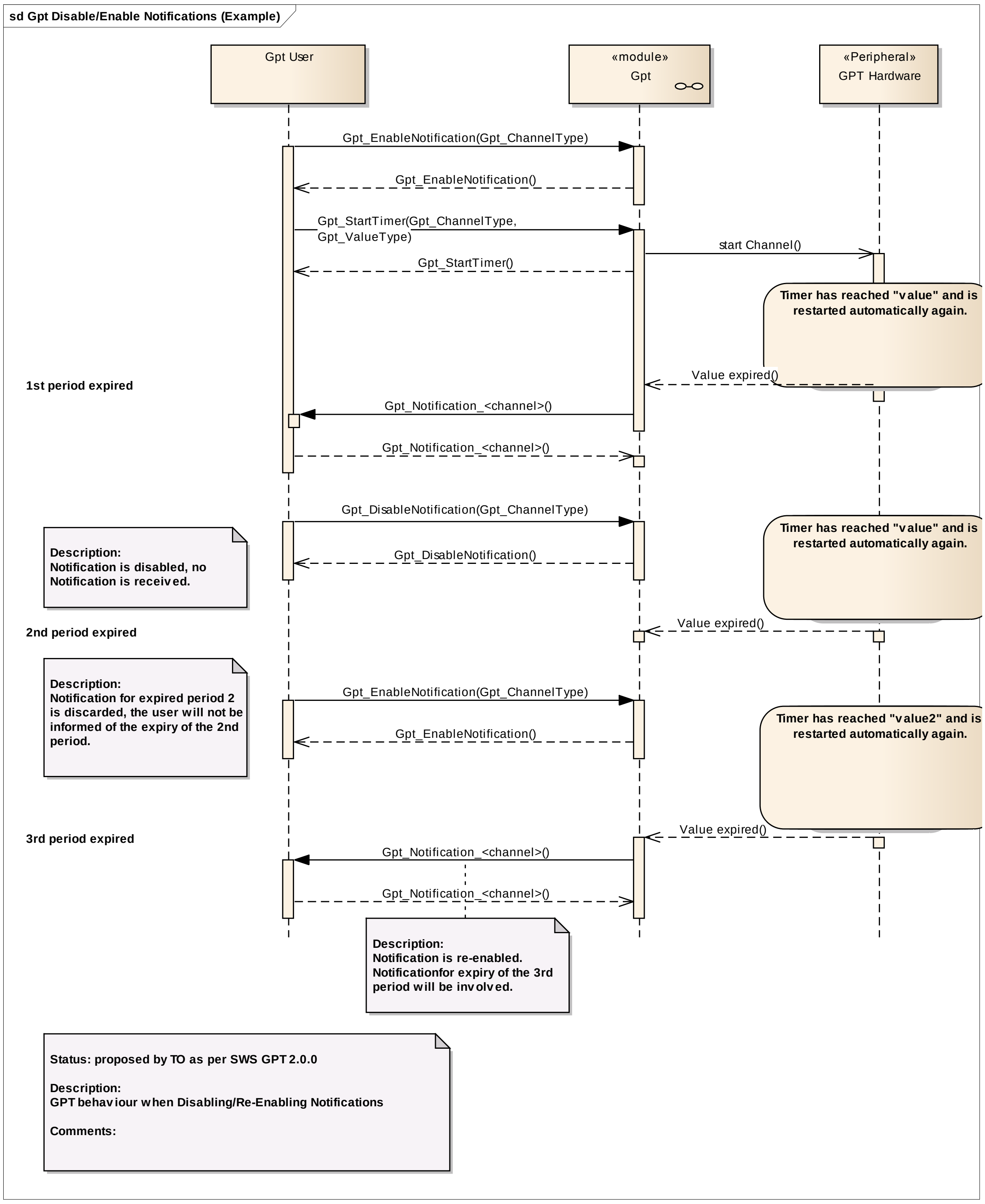
sd Gpt Disable/Enable Notifications (Example)
Gpt User
«module» Gpt
«Peripheral» GPT Hardware
Gpt_EnableNotification(Gpt_ChannelType)
Gpt_EnableNotification()
Gpt_StartTimer(Gpt_ChannelType, Gpt_ValueType)
start Channel()
Gpt_StartTimer()
Timer has reached "value" and is restarted automatically again.
Value expired()
1st period expired
Gpt_Notification_<channel>()
Gpt_Notification_<channel>()
Gpt_DisableNotification(Gpt_ChannelType)
Description: Notification is disabled, no Notification is received.
Gpt_DisableNotification()
Timer has reached "value" and is restarted automatically again.
Value expired()
2nd period expired
Description: Notification for expired period 2 is discarded, the user will not be informed of the expiry of the 2nd period.
Gpt_EnableNotification(Gpt_ChannelType)
Gpt_EnableNotification()
Timer has reached "value2" and is restarted automatically again.
3rd period expired
Value expired()
Gpt_Notification_<channel>()
Gpt_Notification_<channel>()
Description: Notification is re-enabled. Notificationfor expiry of the 3rd period will be involved.
Status: proposed by TO as per SWS GPT 2.0.0
Description: GPT behaviour when Disabling/Re-Enabling Notifications
Comments:

Figure 9: Sequence Diagram - Disable/Re-enable Notifications

9.5 Wakeup

Note: Sequence charts on timer wakeup can be found in the ECU state manager specification [7].

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

GPT276: Variant PC: This variant is limited to pre-compile-configuration parameters only.

GPT277: 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.

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

SWS Item	**GPT321 :**		
Name	GptDevErrorDetect {GPT_DEV_ERROR_DETECT}		
Description	Enables/Disables development error detection		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**GPT322 :**		
Name	GptReportWakeupSource {GPT_REPORT_WAKEUP_SOURCE}		
Description	Enables/Disables wakeup source reporting		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.2.4 GptChannelConfiguration

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

SWS Item	**GPT307 :**		
Name	GptChannelClkSrc {GPT_CHANNEL_CLKSRC}		
Description	GPT187: The GPT module specific clock input for the timer unit can statically be configured and allows to select different clock sources (external clock, internal GPT specific clock) per channel		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Instance		

SWS Item	**GPT308 :**		
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	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 ..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Instance		

SWS Item	**GPT309 :**		
Name	GptChannelMode {GPT_CHANNEL_MODE}		
Description	Specifies the behaviour of the timerchannel after the timeout has expired		
Multiplicity	1		
Type	EnumerationParamDef		
Range	GPT_MODE_CONTINOUS	GPT186: Timerchannel is restarted automatically after reaching its end value	
	GPT_MODE_ONESHOT	GPT185: Timerchannel stops after reaching its end value	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Instance		

SWS Item	**GPT310 :**
Name	GptChannelPrescale {GPT_CHANNEL_PRESCALE}

Description	GPT module specific prescaler factor per channel		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Instance		

SWS Item	**GPT311 :**		
Name	GptEnableWakeup {GPT_ENABLE_WAKEUP}		
Description	GPT188: Enables wakeup capability of CPU for a channel when timeout period expires. This might be different to enabling the notification depending on hardware capabilities		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Instance		

SWS Item	**GPT312 :**		
Name	GptNotification {Gpt_Notification}		
Description	Function pointer to callback function		
Multiplicity	1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: Instance		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
GptWakeupConfiguration	0..1	--

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

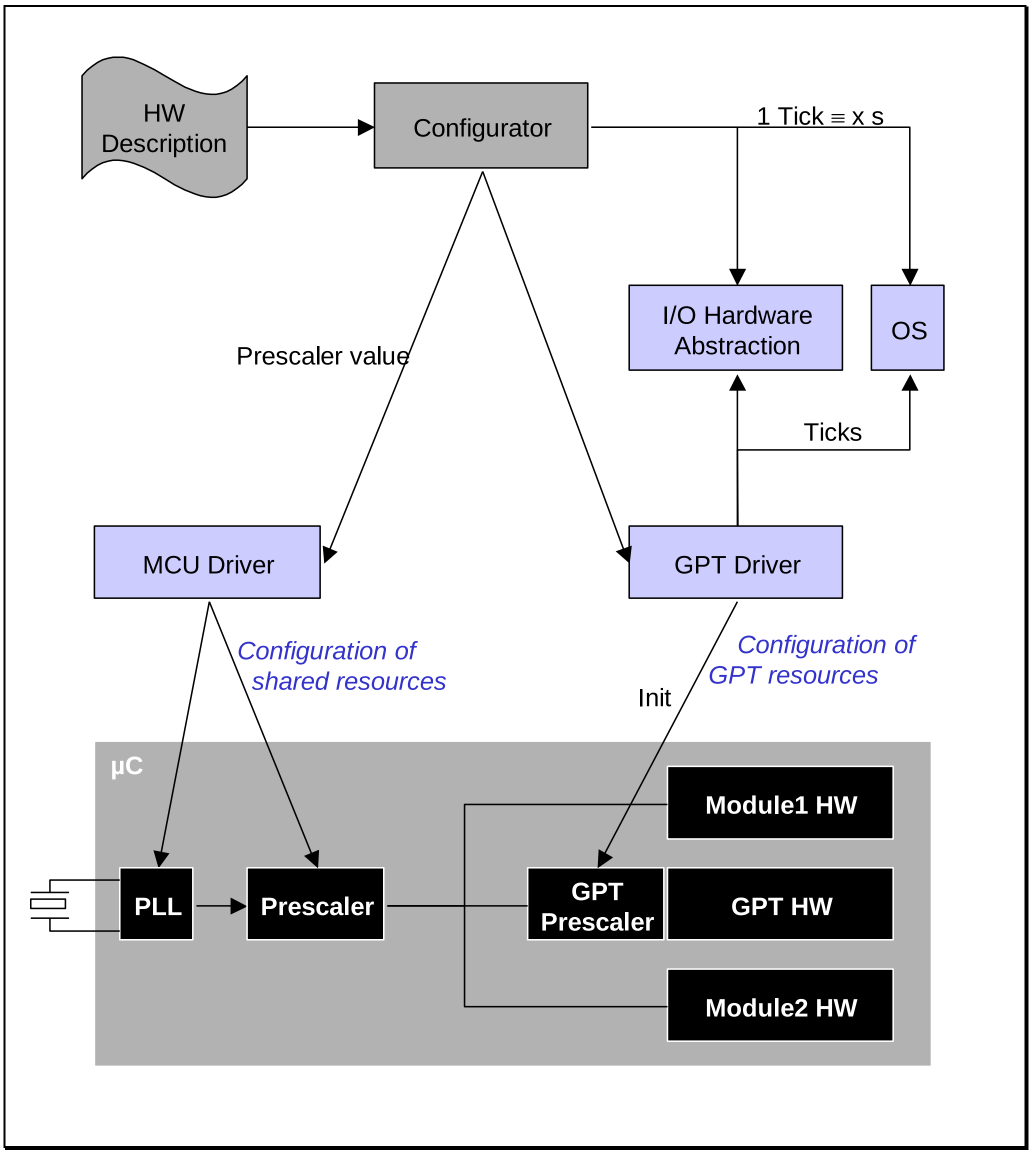

Figure 10: Scope of the GPT Driver configuration

10.2.5 GptChannelConfigSet

SWS Item	**GPT269 :**
Container Name	GptChannelConfigSet [Multi Config Container]
Description	This container is the base of an 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-wide configuration (parameters) of the GPT Driver

10.2.6 GptWakeupConfiguration

SWS Item	**GPT235 :**
Container Name	GptWakeupConfiguration{GPT_WAKEUP_CONFIGURATION}
Description	--
Configuration Parameters	

SWS Item	**GPT313 :**		
Name	GptWakeupSourceRef {Gpt_WakeupSourceRef}		
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	M	VARIANT-POST-BUILD
Scope / Dependency	scope: instance		

No Included Containers

10.2.7 GptConfigurationOfOptApiServices

SWS Item	**GPT193 :**
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 :**		
Name	GptDeinitApi {GPT_DEINIT_API}		
Description	Adds / removes the service Gpt_DeInit() from the code.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**GPT315 :**		
Name	GptEnableDisableNotificationApi {GPT_ENABLE_DISABLE_NOTIFICATION_API}		
Description	Adds / removes the services Gpt_EnableNotification() and Gpt_DisableNotification from the code.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**GPT317 :**		
Name	GptTimeElapsedApi {GPT_TIME_ELAPSED_API}		
Description	Adds / removes the service Gpt_GetTimeElapsed() from the code		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**GPT318 :**		
Name	GptTimeRemainingApi {GPT_TIME_REMAINING_API}		
Description	Adds / removes the service Gpt_GetTimeRemaining() from the code.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**GPT319 :**		
Name	GptVersionInfoApi {GPT_VERSION_INFO_API}		
Description	Adds / removes the service Gpt_GetVersionInfo() from the code.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**GPT320 :**		
Name	GptWakeupFunctionalityApi {GPT_WAKEUP_FUNCTIONALITY_API}		
Description	Adds / removes the services Gpt_SetMode(), Gpt_EnableWakeup() Gpt_DisableWakeup() and Gpt_Cbk_CheckWakeup() from the code.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

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.

The standard common published information like

vendorId (<Module>_VENDOR_ID),
moduleId (<Module>_MODULE_ID),
arMajorVersion (<Module>_AR_MAJOR_VERSION),
arMinorVersion (<Module>_ AR_MINOR_VERSION),
arPatchVersion (<Module>_ AR_PATCH_VERSION),
swMajorVersion (<Module>_SW_MAJOR_VERSION),
swMinorVersion (<Module>_ SW_MINOR_VERSION),
swPatchVersion (<Module>_ SW_PATCH_VERSION),
vendorApiInfix (<Module>_VENDOR_API_INFIX)

is provided in the BSW Module Description Template (see [13] Figure 4.1 and Figure 7.1).

Additional published parameters are listed below if applicable for this module.