MICROSAR GPT

Technical Reference

MCAL Emulation in VTT

Version 1.1.0

Authors	Peter Lang, Christian Leder
Status	Released

Document Information

History

Author	Date	Version	Remarks
Peter Lang	2013-09-17	1.00.00	Initial Creation
Christian Leder	2015-02-09	1.01.00	> Global renaming of Vip to Vtt > Usage of template 5.11.0 for the Technical reference

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_GPTDriver.pdf	V3.2.0
[2]	AUTOSAR	AUTOSAR_SWS_DevelopmentErrorTracer.pdf	V3.2.0
[3]	AUTOSAR	AUTOSAR_SWS_DiagnosticEventManager.pdf	V4.2.0
[4]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	V1.6.0
[5]	AUTOSAR	AUTOSAR_SWS_ECUStateManager.pdf	V3.0.0

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

Component Version	New Features
1.0.x	Initial version of the Vip GPT driver
1.1.x	Modification of interrupt handling
2.0.x	Global renaming of Vip to Vtt

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module GPT as specified in [1].

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	GPT_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	GPT_MODULE_ID	100 decimal (according to ref. [4])

* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

This document describes the functionalities and the API of the GPT driver emulation in Vector's VTT framework.

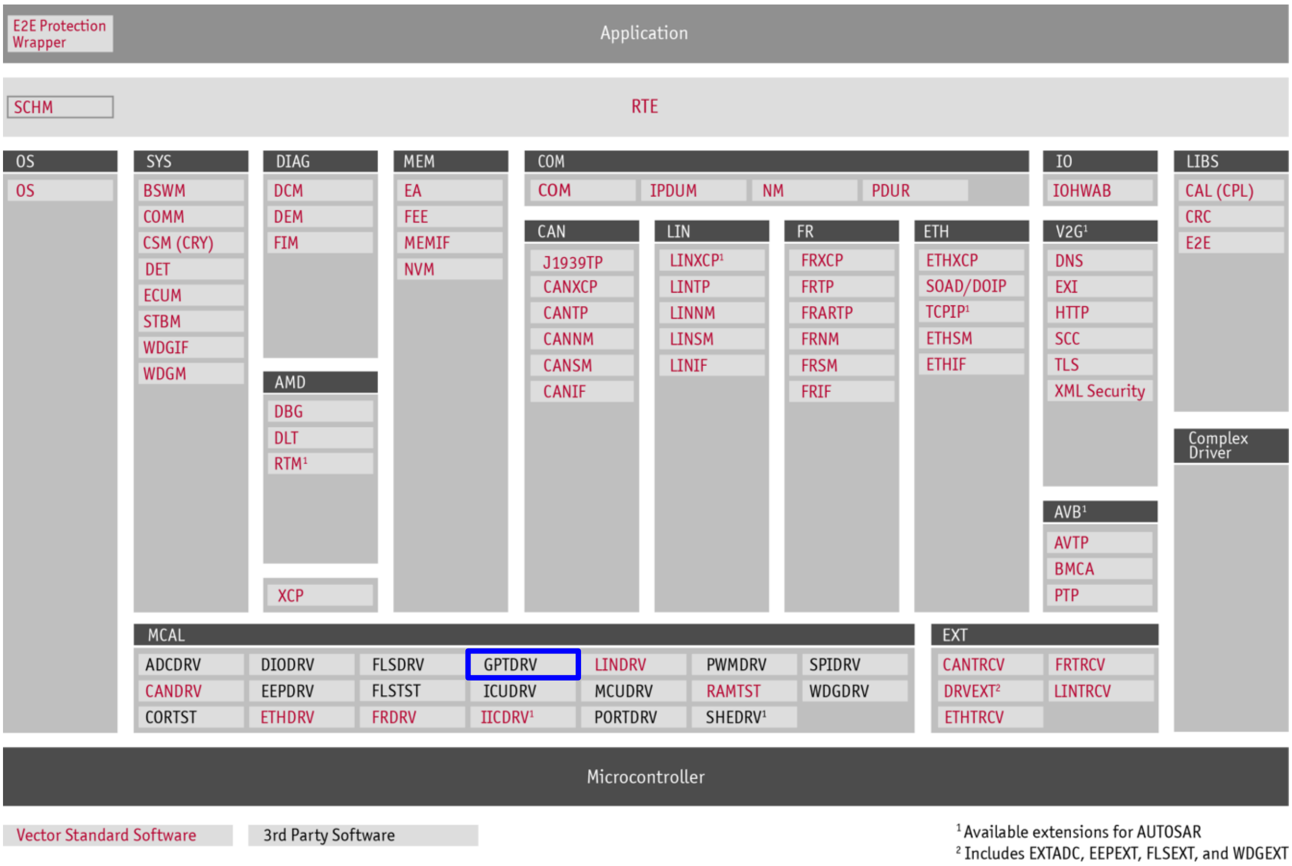
The GPT driver provides services for timer functionalities like free running timers, e.g. for cyclic and single events including notification handling, measurements for elapsed or remaining time and for wakeup events.

The tick duration of a timer channel depends on channel specific settings (part of the GPT driver) as well as on system clock and settings of the clock tree controlled by the MCU module.

The GPT driver only generates time bases, and does not serve as an event counter. This functionality is provided by the ICU driver module.

2.1 Architecture Overview

The following figure shows where the GPT is located in the AUTOSAR architecture.



[1] Available extensions for AUTOSAR
[2] Includes EXTADC, EEPEXT, FLSEXT, and WDGEXT

Figure 2-1 AUTOSAR 4.x Architecture Overview

The next figure shows the interfaces to adjacent modules of the GPT. These interfaces are described in chapter 5.

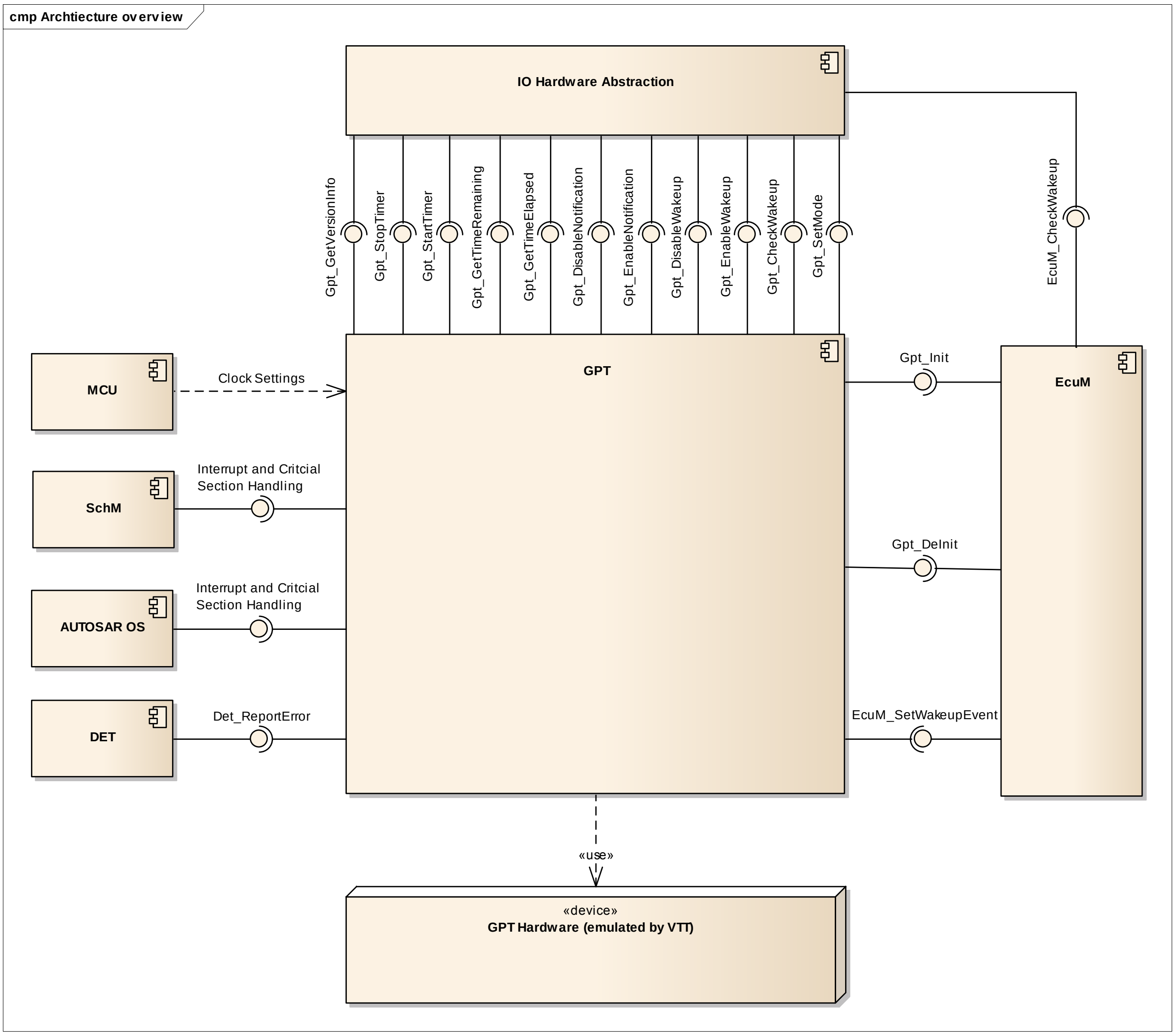


Figure 2-2 Interfaces to adjacent modules of the GPT

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for the GPT.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported AUTOSAR standard conform features

> Table 3-2 Not supported AUTOSAR standard conform features

Vector Informatik provides further GPT functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Functions to initialize and de-initialize the module
Functions to start and stop a particular timer
Functions to acquire the time elapsed or remaining on a particular timer
Functions to enable and disable the timer notification of a particular timer
Functions to enable and disable the wakeup capability and to initiate a wakeup process
Function to adjust the module mode (switches from NORMAL into SLEEP and vice versa)

Table 3-1 Supported AUTOSAR standard conform features

3.1.1 Deviations

The following features specified in [1] are not supported:

Not Supported AUTOSAR Standard Conform Features
None

Table 3-2 Not supported AUTOSAR standard conform features

3.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
In addition to the existing checks required by the AUTOSAR standard, the parameter versioninfo passed to the service `Gpt_GetVersionInfo()` is checked for not referencing `NULL_PTR`. If it does, the error `GPT_E_PARAM_VINFO` is reported to DET instead of `GPT_E_PARAM_POINTER`
In addition, if the parameter passed to the service `Gpt_Init` references `NULL_PTR`, the error `GPT_E_PARAM_CONFIG` is reported to DET instead of `GPT_E_PARAM_POINTER`

Features Provided Beyond The AUTOSAR Standard
The error GPT_E_PARAM_POINTER is never reported to DET. As mentioned above additional error codes are introduced to specify the errors in more detail

Table 3-3 Features provided beyond the AUTOSAR standard

3.1.3 Limitations

There are no limitations within the implementation of the GPT emulation by VTT.

3.2 Initialization

The GPT module is being initialized by calling `Gpt_Init(&GptChannelConfigSet)`. All global variables are initialized by calling `Gpt_InitMemory()`. So, `Gpt_InitMemory()` has to be called prior to `Gpt_Init()`.

3.3 Emulation

This driver uses timers emulated by the VTT framework for simulation of an onboard timer unit. Therefore, the **user has to ensure, that the application does not block timer handling**.

Caution

Be careful using while loops in order to poll any status.

The user has to ensure, that the application does not block the emulation. So, within every while loop the following function call has to be called:

```
while(ANY_STATUS == temp_status)
{
  Schedule();
}
```

Use the function call Schedule() which is available once the header file of the module GPT is included.

3.4 States

3.4.1 Module States

The module GPT provides the following global states:

- *uninitialized / undefined*: GPT is not initialized
- `GPT_MODE_NORMAL`: Module stays in normal operating mode
- `GPT_MODE_SLEEP`: Module was set in sleep mode

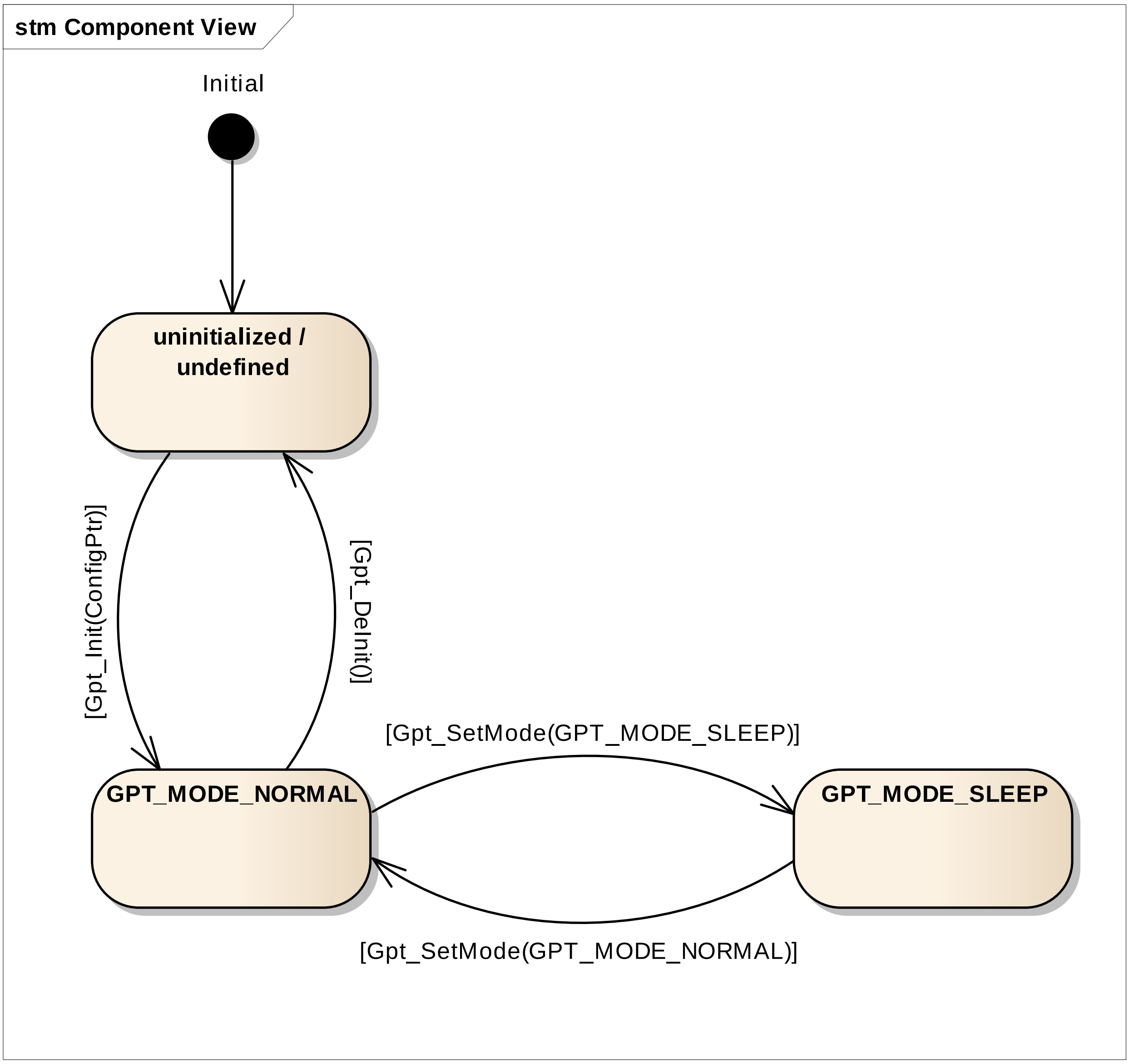


Figure 3-1 Module States

3.4.2 Timer Channel States

Each timer can be in one of the following states:

- *idle / inactive*: timer channel is not running
- *running / active*: timer channel is running

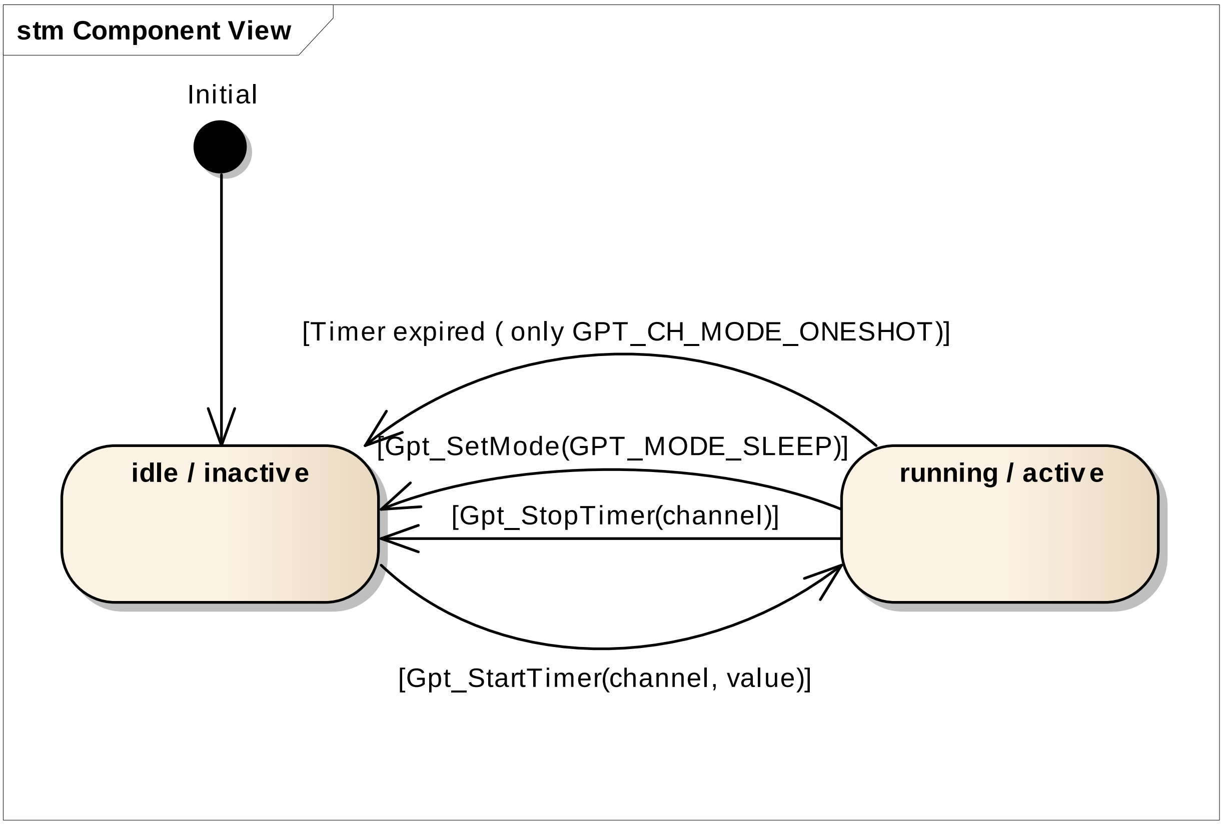

stm Component View
Initial
[Timer expired (only GPT_CH_MODE_ONESHOT)]
[Gpt_SetMode(GPT_MODE_SLEEP)]
idle / inactive
running / active
[Gpt_StopTimer(channel)]
[Gpt_StartTimer(channel, value)]

Figure 3-2 Timer Channel States

3.5 Main Functions

Module GPT does not provide any cyclic main functions.

3.6 Error Handling

3.6.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `GPT_DEV_ERROR_DETECT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported GPT ID is 100.

The reported service IDs identify the services which are described in 5.3. The following table presents the service IDs and the related services:

Service ID	Service
0x00	Gpt_GetVersionInfo
0x01	Gpt_Init
0x02	Gpt_DeInit
0x03	Gpt_GetTimerElapsed

Service ID	Service
0x04	Gpt_GetTimerRemaining
0x05	Gpt_StartTimer
0x06	Gpt_StopTimer
0x07	Gpt_EnableNotification
0x08	Gpt_DisableNotification
0x09	Gpt_SetMode
0x0A	Gpt_DisableWakeup
0x0B	Gpt_EnableWakeup
0x0C	Gpt_CheckWakeup

Table 3-4 Service IDs

The errors reported to DET are described in the following table:

Error Code		Description
0x0A	GPT_E_UNINIT	API service called without module initialization
0x0B	GPT_E_BUSY	API service called when timer channel is still busy (running)
0x0D	GPT_E_ALREADY_INITIALIZED	API service for initialization called when already initialized
0x14	GPT_E_PARAM_CHANNEL	API parameter checking: invalid channel
0x15	GPT_E_PARAM_VALUE	API parameter checking: invalid value
0x1F	GPT_E_PARAM_MODE	API parameter checking: invalid mode
0x21	GPT_E_PARAM_CONFIG	Gpt_Init called with ConfigPtr referencing NULL_PTR
0x22	GPT_E_PARAM_VINFO	Gpt_GetVersionInfo called with VersionInfoPtr referencing NULL_PTR

Table 3-5 Errors reported to DET

3.6.1.1 Parameter Checking

AUTOSAR requires that API functions check the validity of their parameters. The checks in Table 3-6 are internal parameter checks of the API functions. These checks are for development error reporting and can be en-/disabled.

The following table shows which parameter checks are performed on which services:

Check Service	GPT_E_PARAM_CHANNEL	GPT_E_ALREADY_INITIALIZED	GPT_E_PARAM_CONFIG	GPT_E_UNINIT	GPT_E_BUSY	GPT_E_PARAM_MODE	GPT_E_UNINIT	GPT_E_PARAM_VINFO
Gpt_Init		■	■					
Gpt_DeInit				■	■		■	
Gpt_GetTimeElapsed	■	■		■			■	
Gpt_GetTimeRemaining	■	■		■			■	
Gpt_StartTimer	■			■	■		■	
Gpt_StopTimer	■			■			■	
Gpt_EnableNotification	■			■			■	
Gpt_DisableNotification	■			■			■	
Gpt_DisableWakeup	■			■			■	
Gpt_EnableWakeup	■			■			■	
Gpt_SetMode				■		■	■	
Gpt_GetVersionInfo								■

Table 3-6 Development Error Reporting: Assignment of error codes to services

3.6.2 Production Code Error Reporting

Info
Production errors are not supported in this emulation.

4 Integration

This chapter gives necessary information for the integration of the MICROSAR GPT into an application environment of an ECU.

4.1 Scope of Delivery

The delivery of the GPT contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Description
Gpt.h	The module header defines the interface of the GPT. This file must be included by upper layer software components
Gpt.c	This C-source contains the implementation of the module's functionalities
Gpt_Irq.h	The Gpt_Irq header defines the interfaces for the emulated hardware
Gpt_Irq.c	This C-Source contains the implementation of the interfaces for the emulated hardware
DrvGpt_VttCanoe01Asr.jar	This jar-file contains the generator and the validator for the DaVinci Configurator
VTTGpt_bswmd.arxml	Basic Software Module Description according to AUTOSAR for VTT Emulation
Gpt_bswmd.arxml	Optional Basic Software Module Description. Placeholder for real target (semiconductor manufacturer) in VTT only use case

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator.

File Name	Description
Gpt_Cfg.h	The configuration-header contains the static configuration part of this module
Gpt_PBcfg.c	The configuration-source contains the object independent part of the runtime configuration

Table 4-2 Generated files

4.2 Include Structure

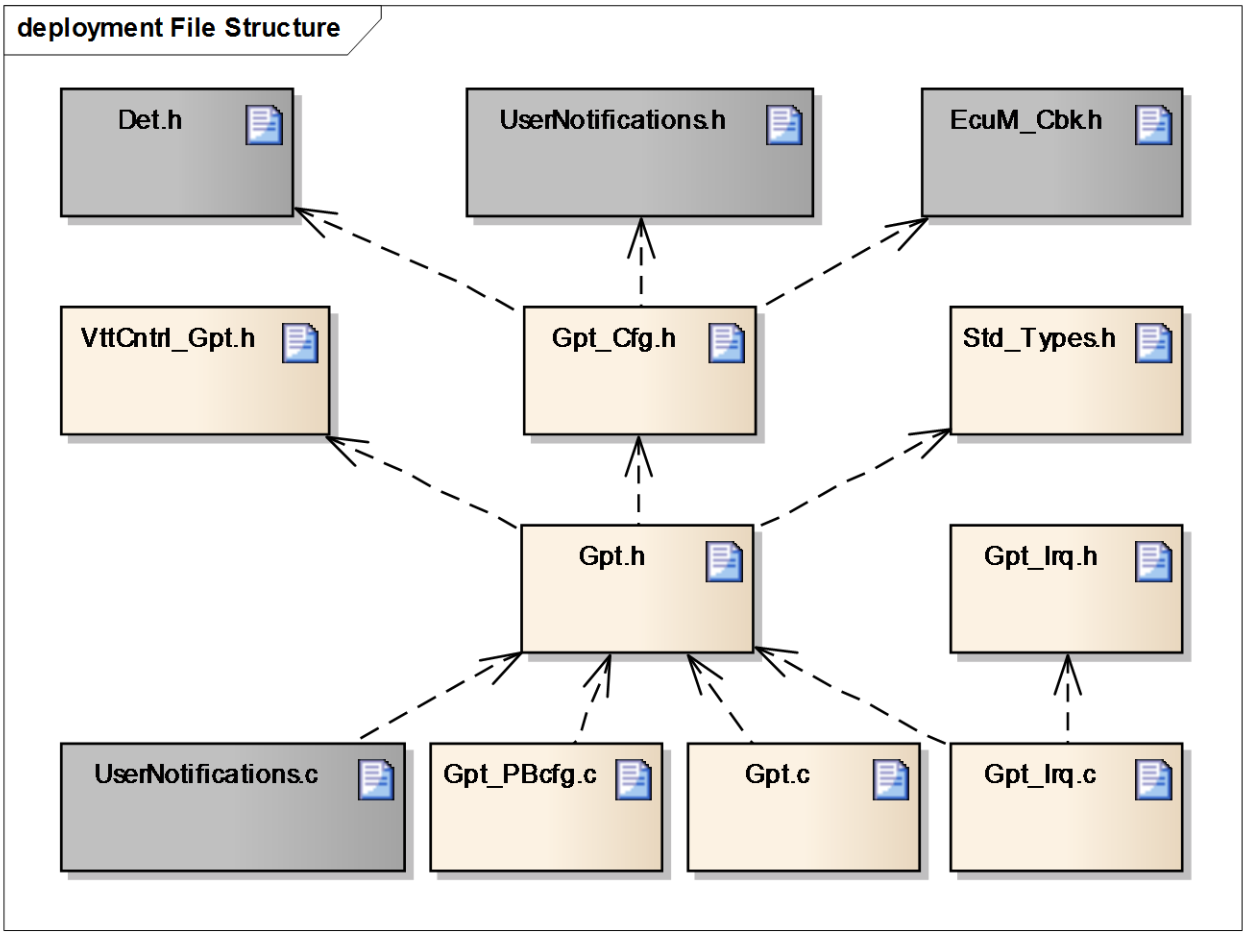


Figure 4-1 Include Structure

4.3 Dependencies on SW Modules

4.3.1 AUTOSAR OS (Optional)

An operating system can be used for task scheduling, interrupt handling, global suspend and restore of interrupts and creating of the Interrupt Vector Table.

4.3.2 DET (Optional)

The GPT module depends on the DET (by default) in order to report development errors. Detection and reporting of development errors can be enabled or disabled by the switch "Enable Development Error Detection".

4.3.3 SchM (Optional)

Beside the AUTOSAR OS the Schedule Manager provides functions that module GPT calls at begin and end of critical sections.

4.3.4 EcuM (Optional)

The module EcuM delivers functionalities to use low-power modes offered by the hardware (e.g. wakeup functionalities). Also the initialization is done by the EcuM module.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

The types defined by the GPT are described in this chapter.

Type Name	C-Type	Description	Value Range
Gpt_ChannelType	uint8	Numeric ID of a GPT channel	0 – 19 Maximum 20 channels can be configured in this emulation
Gpt_ValueType	uint32	Type for reading and setting the timer values (in number of ticks).	0 – 4294967295 According to the register width of 32 bits
Gpt_ModeType	enum	Allows the selection of different power modes	`GPT_MODE_NORMAL` Normal operation mode of the GPT
			`GPT_MODE_SLEEP` Reduced power operation mode

Table 5-1 Type definitions

5.2 Interrupt Service Routines provided by GPT

5.2.1 GptIsr_<0…19>

Prototype	
`void` **`GptIsr_<0…19>`** `( void )`	
Parameter	
-	-
Return code	
-	-
Functional Description	
Interrupt Service Routine for each Timer Unit. The ISRs are called from the VTT Kernel if an ongoing timer has expired. Within the function, a handler is called which do the state transitions and calls the notifications.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant.	

Table 5-2 [ISR name]

5.3 Services provided by GPT

5.3.1 Gpt_InitMemory

Prototype	
`void Gpt_InitMemory (void)`	
Parameter	
-	-
Return code	
-	-
Functional Description	
This service initializes the global variables in case the startup code does not work	
Particularities and Limitations	
> This function is synchronous. > This function is non reentrant. > Module must not be initialized.	
Expected Caller Context	
> Called during startup	

Table 5-3 Gpt_InitMemory

5.3.2 Gpt_Init

Prototype	
`void Gpt_Init (P2CONST(Gpt_ConfigType, AUTOMATIC, GPT_APPL_CONST) ConfigPtr)`	
Parameter	
`ConfigPtr`	Pointer to the configuration struct of the GPT
Return code	
-	-
Functional Description	
The function Gpt_Init initializes the timer module that is emulated in the VTT framework. This function has to be called first in order to initialize the GPT for use. Otherwise no operation can be performed. The function disables all wakeup and notification interrupts. GPT is set to normal mode. This function also has to be called after a reset or for re-initialization after calling the service Gpt_DeInit. The function sets the operation mode of the GPT to `GPT_MODE_NORMAL`.	
Particularities and Limitations	
> This function is synchronous. > This function is non reentrant. > Module must not be initialized.	
Expected Caller Context	
> ECU State Manager or comparable software module, responsible for driver initialization after startup.	

Table 5-4 Gpt_Init

5.3.3 Gpt_DeInit

Prototype	
void **Gpt_DeInit** (void)	
Parameter	
-	-
Return code	
-	-
Functional Description	
The function Gpt_DeInit de-initializes all timer channels used by the configuration to their power on reset state. The function disables all interrupt notifications and wakeup interrupts, controlled by the GPT driver.	
Particularities and Limitations	
> This function is synchronous. > This function is non reentrant.	
Expected Caller Context	
> Task context	

Table 5-5 Gpt_DeInit

5.3.4 Gpt_GetTimeElapsed

Prototype	
Gpt_ValueType **Gpt_GetTimeElapsed** (Gpt_ChannelType Channel)	
Parameter	
Channel	Numeric identifier of the GPT channel (it is recommended to use the symbolic channel name configured in the configuration tool instead of the numeric ID).
Return code	
Gpt_ValueType	Elapsed timer value (in number of ticks).
Functional Description	
The service queries the time already elapsed. When the channel is in mode GPT_MODE_ONESHOT, this is the value relative to the time the channel has been started with Gpt_StartTimer. When the channel is in mode GPT_MODE_CONTINUOUS, the function returns the timer value relative to the last timeout respectively the start of the channel.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different timer channels. > This service may only be called if the module has been initialized before. > This service may only be called if the timer is active.	
Expected Caller Context	
> Task context	

Table 5-6 Gpt_GetTimeElapsed

5.3.5 Gpt_GetTimeRemaining

Prototype	
`Gpt_ValueType` **`Gpt_GetTimeRemaining`** `(Gpt_ChannelType Channel)`	
Parameter	
`Channel`	Numeric identifier of the GPT channel (it is recommended to use the symbolic channel name configured in the configuration tool instead of the numeric ID).
Return code	
`Gpt_ValueType`	Remaining timer value (in number of ticks).
Functional Description	
The service queries the time remaining until the next timeout period will expire.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant. > This service may only be called if the module has been initialized before. > This service may only be called if the timer is active.	
Expected Caller Context	
> Task context	

Table 5-7 Gpt_GetTimeRemaining

5.3.6 Gpt_StartTimer

Prototype	
`void` **`Gpt_StartTimer`** `( Gpt_ChannelType Channel, Gpt_ValueType Value )`	
Parameter	
`Channel`	Numeric identifier of the GPT channel (it is recommended to use the symbolic channel name configured in the configuration tool instead of the numeric ID)
`Value`	Timeout period (in number of ticks) after that the timer channel expires.
Return code	
-	-
Functional Description	
The service starts the selected timer channel with a defined timeout period (e.g. to invoke the configured notification for that channel after the timeout period).	
Particularities and Limitations	
> This function is synchronous and is reentrant for different timer channels. > This service may only be called if the module has been initialized before. > This service may only be called if the timer is inactive.	
Expected Caller Context	
> Task context	

Table 5-8 Gpt_StartTimer

5.3.7 Gpt_StopTimer

Prototype	
void **Gpt_StopTimer** (Gpt_ChannelType Channel)	
Parameter	
Channel	Numeric identifier of the GPT channel (it is recommended to use the symbolic channel name configured in the configuration tool instead of the numeric ID)
Return code	
-	-
Functional Description	
The service stops the selected timer channel.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant. > This service may only be called if the module has been initialized before.	
Expected Caller Context	
> Task context	

Table 5-9 Gpt_StopTimer

5.3.8 Gpt_EnableNotification

Prototype	
void **Gpt_EnableNotification** (Gpt_ChannelType Channel)	
Parameter	
Channel	Numeric identifier of the GPT channel (it is recommended to use the symbolic channel name configured in the configuration tool instead of the numeric ID)
Return code	
-	-
Functional Description	
The service enables the invocation of the configured notification function for the assigned timer channel independent of the call of the function Gpt_StartTimer.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different timer channels. > This service may only be called if the module has been initialized before. > This service may only be called if the module has a configured notification function. > This function is configurable.	
Expected Caller Context	
> Task context	

Table 5-10 Gpt_EnableNotification

5.3.9 Gpt_DisableNotification

Prototype	
void **Gpt_DisableNotification** (Gpt_ChannelType Channel)	
Parameter	
Channel	Numeric identifier of the GPT channel (it is recommended to use the symbolic channel name configured in the configuration tool instead of the numeric ID)
Return code	
-	-
Functional Description	
The service disables the invocation of the configured notification function for the assigned timer channel independent of the call of the function Gpt_StartTimer.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different timer channels. > This service may only be called if the module has been initialized before. > This service may only be called if the module has a configured notification function. > This function is configurable.	
Expected Caller Context	
> Task context	

Table 5-11 Gpt_DisableNotification

5.3.10 Gpt_SetMode

Prototype	
void **Gpt_SetMode** (Gpt_ModeType Mode)	
Parameter	
Mode	GPT_MODE_NORMAL, normal operating mode GPT_MODE_SLEEP, module should be prepared for sleep mode
Return code	
-	-
Functional Description	
The function Gpt_SetMode sets the operation mode to the given mode parameter. > Parameter mode equals GPT_MODE_NORMAL: The service does not affect the notifications as configured and selected by the functions Gpt_DisableNotification and Gpt_EnableNotification. > Parameter mode equals GPT_MODE_SLEEP: This service stops all non-wakeup capable timer channels. Only those channels, which can serve as a wakeup source, keep on running.	

Particularities and Limitations
> This function is synchronous. > This function is non reentrant. > This service may only be called if the module has been initialized before. > This function is configurable.
Expected Caller Context
> Task context

Table 5-12 Gpt_SetMode

5.3.11 Gpt_DisableWakeup

Prototype	
void **Gpt_DisableWakeup** (Gpt_ChannelType Channel)	
Parameter	
Channel	Numeric identifier of the GPT channel (it is recommended to use the symbolic channel name configured in the configuration tool instead of the numeric ID)
Return code	
-	-
Functional Description	
The service disables the wakeup functionality of the assigned GPT channel.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different timer channels. > This service may only be called if the module has been initialized before. > This function is configurable.	
Expected Caller Context	
> Task context	

Table 5-13 Gpt_DisableWakeup

5.3.12 Gpt_EnableWakeup

Prototype	
void **Gpt_EnableWakeup** (Gpt_ChannelType Channel)	
Parameter	
Channel	Numeric identifier of the GPT channel (it is recommended to use the symbolic channel name configured in the configuration tool instead of the numeric ID)
Return code	
-	-
Functional Description	
The function Gpt_EnableWakeup enables the wakeup functionality of the assigned GPT channel.	

Particularities and Limitations
> This function is synchronous. > This function is reentrant for different timer channels. > This service may only be called if the module has been initialized before. > This function is configurable.
Expected Caller Context
> Task context

Table 5-14 Gpt_EnableWakeup

5.3.13 Gpt_CheckWakeup

Prototype	
void **Gpt_CheckWakeup** (EcuM_WakeupSourceType WakeupSource)	
Parameter	
WakeupSource	Numeric identifier of the expected GPT wakeup source ID
Return code	
--	--
Functional Description	
The service checks if a wakeup capable GPT channel is the source for a wakeup event and calls EcuM_SetWakeupEvent to indicate a valid wakeup timer event to ECU State Manager.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant. > This service may only be called if the module has been initialized before. > This function is configurable.	
Expected Caller Context	
> Task context	

Table 5-15 Gpt_CheckWakeup

5.3.14 Gpt_GetVersionInfo

Prototype	

`void` **`Gpt_GetVersionInfo`**
`(`
`  P2VAR(Std_VersionInfoType, AUTOMATIC, GPT_APPL_DATA) versioninfo`
`)`	

Parameter	
`versioninfo`	Pointer for storing the version information of this module
Return code	
-	-
Functional Description	
This function returns the version information of the module. The version information includes: > Module Id > Vendor Id > Software version numbers	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant. > This function is configurable.	
Expected Caller Context	
> Task context	

Table 5-16 Gpt_GetVersionInfo

5.4 Services used by GPT

In the following table services provided by other components, which are used by the GPT are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET	Det_ReportError

Table 5-17 Services used by the GPT

5.5 Configurable Interfaces

5.5.1 Notifications

At its configurable interfaces the GPT defines notifications that can be mapped to callback functions provided by other modules. The mapping is not statically defined by the GPT but can be performed at configuration time. The function prototypes that can be used for the configuration have to match the appropriate function prototype signatures, which are described in the following sub-chapters.

5.5.1.1 GptNotification

Prototype	
`void` **`<GptNotification>`** `(void)`	
Parameter	
-	-
Return code	
-	-
Functional Description	
This function is called upon an interrupt caused by a CANoe timer event. An individual notification callback can be associated with each timer channel.	
Particularities and Limitations	
> This function is synchronous. > These notification functions are only available, if `Gpt_EnableNotification` was called before for the assigned timer channel has expired. > The notification functions can be configured in the configuration tool	
Call context	
> Called within simulated ISR.	

Table 5-18 GptNotification

6 Configuration

6.1 Configuration Variants

The GPT supports the configuration variants

- `VARIANT-PRE-COMPILE`

The configuration classes of the GPT parameters depend on the supported configuration variants. For their definitions please see the VTTGpt_bswmd.arxml file.

6.2 Configuration with DaVinci Configurator 5

The GPT module is configured with the help of the configuration tool DaVinci Configurator 5 (CFG5). The definition of each parameter is given in the corresponding BSWMD file.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
CANoe	Tool for simulation and testing of networks and electronic control units.
DaVinci Configurator	Configuration and generation tool for MICROSAR components

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
EcuM	ECU State Manager
GPT	General Purpose Timer
IoHwAb	BSW Module I/O Hardware Abstraction (Connection to RTE)
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
OS	Operating System
RTE	Runtime Environment
SchM	BSW Module Scheduler
VTT	vVIRTUALtarget

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

> News
> Products
> Demo software
> Support
> Training data
> Addresses

www.vector.com