MICROSAR PWM

Technical Reference

MCAL Emulation in VTT

Version 1.1.0

Authors	Christian Leder
Status	Released

Document Information

History

Author	Date	Version	Remarks
Christian Leder	2014-02-20	1.00.00	Creation of document
Christian Leder	2015-02-18	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_PWMDriver.pdf	V2.5.0
[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

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 PWM driver
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 PWM as specified in [1].

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	PWM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	PWM_MODULE_ID	121 decimal (according to ref. [4])

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

The MICROSAR module PWM implements an interface in C programming language for handling the PWM functionality of the emulated microcontroller. This PWM driver takes care of initializing and de-initializing the PWM unit and offers services to:

- Start output of a PWM signal
- Stop output of a PWM signal
- Set parameters of a PWM channel’s waveform
- Enable/disable notifications

2.1 Architecture Overview

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

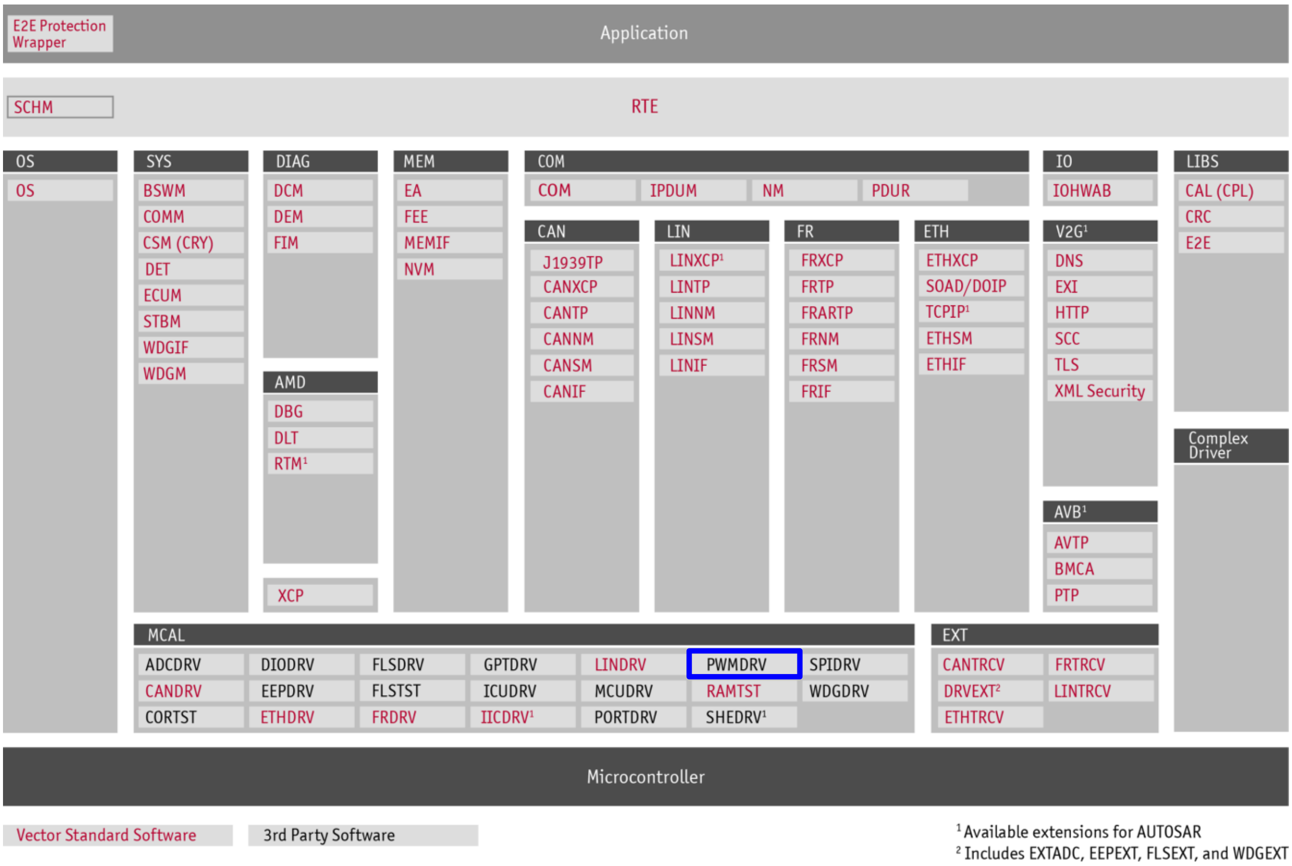

Figure 2-1 AUTOSAR 4.x Architecture Overview

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

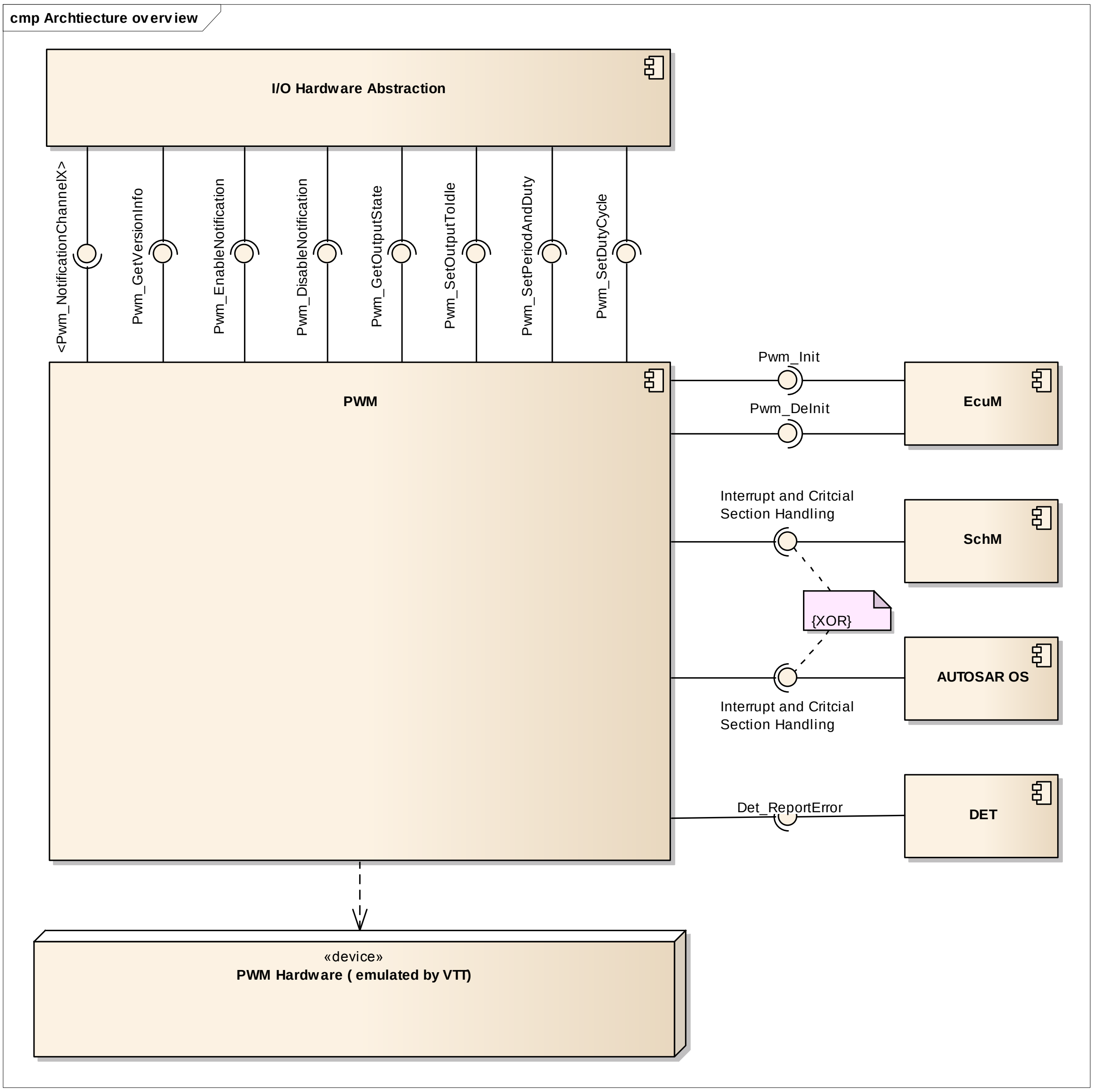

Figure 2-2 Interfaces to adjacent modules of the PWM

3 Functional Description

3.1 Features

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

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 PWM 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
Synchronous services to start and stop the output of pulse-width-modulated wave forms. These outputs are mirrored in system variables in CANoe
Changing of frequency and duty cycle for a PWM channel at runtime besides the default configuration

Table 3-1 Supported AUTOSAR standard conform features

Figure 3-1 shows the two different possibilities how the PWM signal can be interpreted. If the polarity is `PWM_HIGH`, the duty cycle is the relation of the high time to the period time. If the polarity is `PWM_LOW`, the duty cycle is the relation of the low time to the period time.

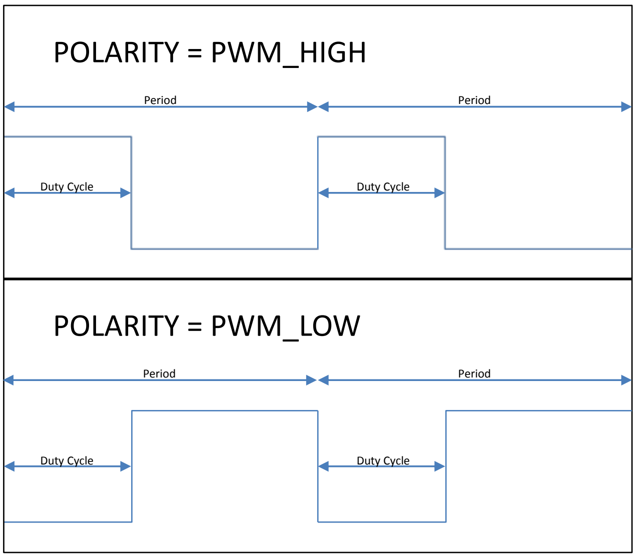

Figure 3-1 PWM signal description [1]

The following figure shows two different options of the PWM module's idle level:

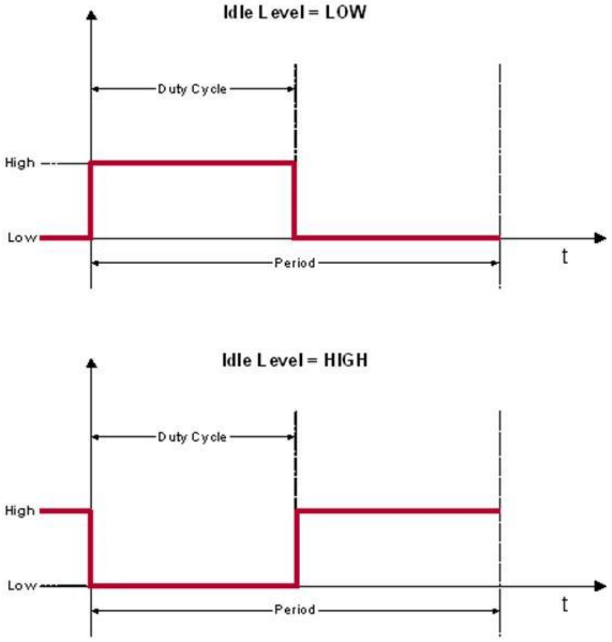

Figure 3-2 Standard alignments left and right, expressed by high and low polarity level

3.1.1 Deviations

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

Not Supported AUTOSAR Standard Conform Features
The channel class `PWM_FIXED_PERIOD_SHIFTED` is not supported

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 `Pwm_GetVersionInfo` is checked for not referencing `NULL_PTR`. If it does, the error `PWM_E_PARAM_VINFO` is reported to DET instead of `PWM_E_PARAM_POINTER`
In addition, if a PWM channel is configured with period zero, an error is reported to DET with the Error ID `PWM_E_PERIOD_ZERO` within `Pwm_Init`

Table 3-3 Features provided beyond the AUTOSAR standard

3.1.3 Limitations

3.1.3.1 Diagnostic Event Manager

Due to the fact that the PWM is emulated, reporting of hardware errors to the DEM is not supported. Because of compatibility reasons, the DEM has to be configured in DaVinci Configurator.

3.2 Emulation

This driver is an emulation of an PWM module.

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 PWM is included.

3.3 Initialization

The PWM module is being initialized by calling `Pwm_Init(&PwmConfigSet)`. All global variables are initialized by calling `Pwm_InitMemory()`. So, `Pwm_InitMemory()` has to be called prior to `Pwm_Init()`.

3.4 States

The PWM module does not implement a state machine.

3.5 Main Functions

The PWM module 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 `PWM_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 PWM ID is 121.

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	Pwm_Init
0x01	Pwm_DeInit
0x02	Pwm_SetDutyCycle
0x03	Pwm_SetPeriodAndDuty
0x04	Pwm_SetOutputToIdle
0x05	Pwm_GetOutputState
0x06	Pwm_DisableNotification
0x07	Pwm_Enable_Notification
0x08	Pwm_GetVersionInfo

Table 3-4 Service IDs

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

Error Code		Description
0x10	PWM_E_PARAM_CONFIG	API Pwm_Init service called with wrong parameter
0x11	PWM_E_UNINIT	API service used without module initialization
0x12	PWM_E_PARAM_CHANNEL	API service used with an invalid channel Identifier
0x13	PWM_E_PERIOD_UNCHANGEABLE	Usage of unauthorized PWM service on PWM channel configured a fixed period
0x14	PWM_E_ALREADY_INITIALIZED	API Pwm_Init service called while the PWM driver has already been initialized
0x15	PWM_E_PARAM_POINTER	API Pwm_EnableNotification is called and no notification function is configured for this channel
0x16	PWM_E_PARAM_VINFO	API Pwm_GetVersionInfo is called with a NULL_PTR parameter.
0x17	PWM_E_PERIOD_ZERO	API Pwm_Init is called with and period time is configured with 0 for this channel

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	PWM_E_PARAM_CONFIG	PWM_E_UNINIT	PWM_E_PARAM_CHANNEL	PWM_E_PERIOD_UNCHANGEABLE	PWM_E_ALREADY_INITIALIZED	PWM_E_NULL_POINTER	PWM_E_PARAM_VINFO	PWM_E_PERIOD_ZERO
Pwm_Init	■				■			■
Pwm_DeInit		■						
Pwm_SetDutyCycle		■	■	■				■
Pwm_SetPeriodAndDuty		■	■					
Pwm_SetOutputToIdle		■	■					
Pwm_GetOutputState		■	■					
Pwm_DisableNotification		■	■					
Pwm_EnableNotification		■	■			■		
Pwm_GetVersionInfo							■	

Table 3-6 Development Error Reporting: Assignment of checks 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 PWM into an application environment of an ECU.

4.1 Scope of Delivery

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

4.1.1 Static Files

File Name	Description
Pwm.h	The module header defines the interface of the PWM. This file must be included by upper layer software components
Pwm.c	This C-source contains the implementation of the module's functionalities
Pwm_Irq.h	The Pwm_Irq header defines the interfaces for the emulated hardware
Pwm_Irq.c	This C-Source contains the implementation of the interfaces for the emulated hardware
DrvPwm_VttCanoe01Asr.jar	This jar-file contains the generator and the validator for the DaVinci Configurator
VTTPwm_bswmd.arxml	Basic Software Module Description according to AUTOSAR for VTT Emulation
Pwm_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
Pwm_Cfg.h	The configuration-header contains the static configuration part of this module
Pwm_PBcfg.c	The configuration-source contains the object independent part of the runtime configuration

Table 4-2 Generated files

4.2 Include Structure

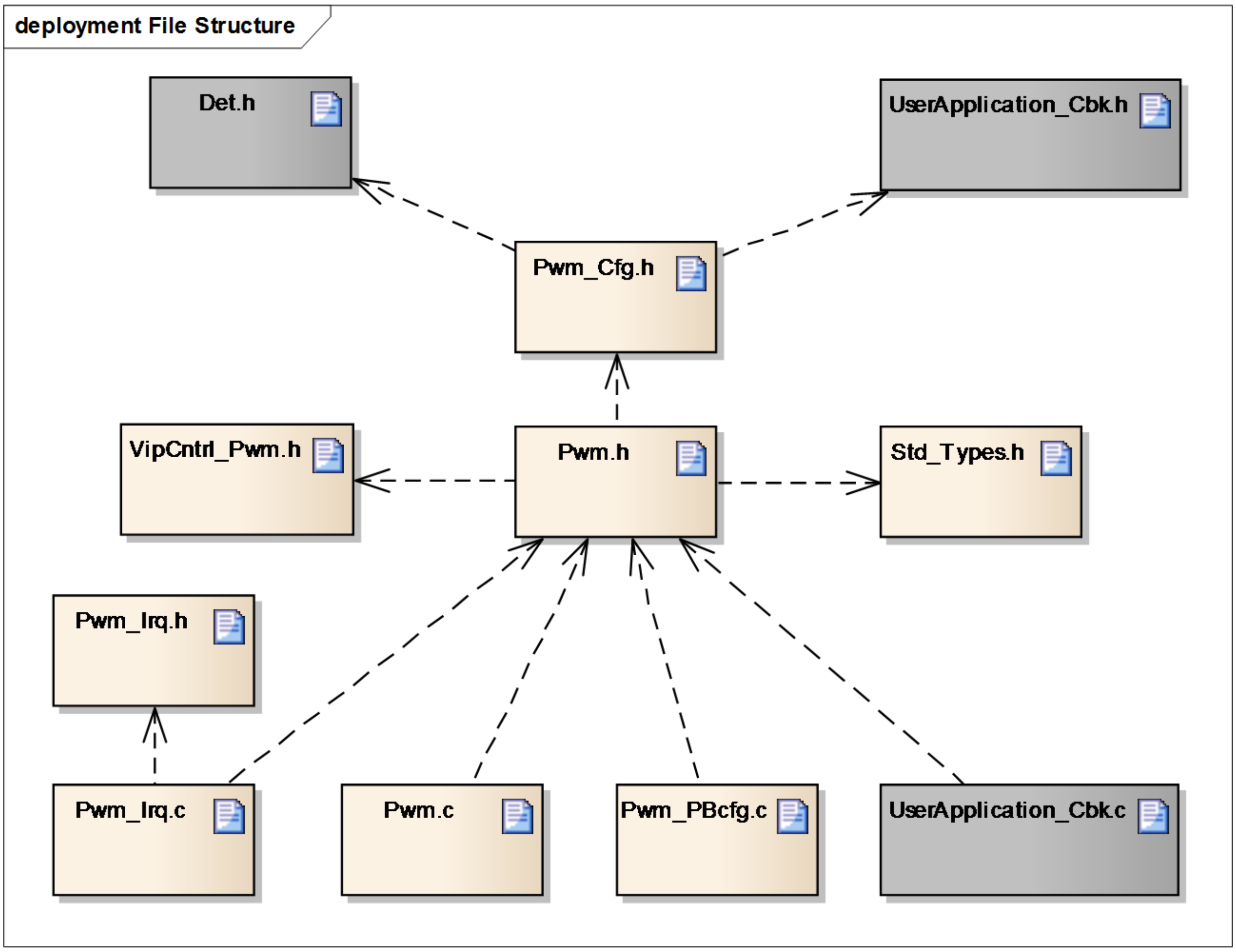

Figure 4-1 Include Structure

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 PWM 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 PWM calls at begin and end of critical sections.

4.3.4 EcuM (Optional)

The EcuM cares for the initialization of the module PWM.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

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

Type Name	C-Type	Description	Value Range
Pwm_OutputStateType	enum	Output states of a PWM channel	PWM_HIGH The output signal is high.
			PWM_LOW The output signal is low.
Pwm_EdgeNotificationType	enum	Definition of the type of notification for a PWM channel	PWM_RISING_EDGE Notification will be called upon a rising edge in the PWM signal.
			PWM_FALLING_EDGE Notification will be called upon a falling edge in the PWM signal.
			PWM_BOTH_EDGES Notification will be called upon both falling and rising edge in the PWM signal.
Pwm_ChannelType	uint8	This is the numeric representative of the symbolic name of a PWM channel	0 … 9 Available channels
Pwm_InterruptSourceType	uint8	Identifier for an PWM interrupt	0 … 9 Represents the interrupt source (similar to Pwm_ChannelType)
Pwm_PeriodType	uint32	Used to specify period values for a PWM channel	1 – 4294967295
Pwm_ChannelClassType	enum	Definition of restricted operations for a channel	Fixed Period Period value of the channel cannot be changed.
			Variable Period Period value of the channel can be changed during runtime.

Table 5-1 Type definitions

5.2 Interrupt Service Routines provided by PWM

5.2.1 PwmIsr_<0…9>

Prototype	
void **PwmIsr_<0…9>** (void)	
Parameter	
-	-
Return code	
-	-
Functional Description	
Interrupt Service Routine for each PWM 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, if configured and enabled.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant.	

Table 5-2 PwmIsr_<0…9>

5.3 Services provided by PWM

5.3.1 Pwm_InitMemory

Prototype	
void **Pwm_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 Pwm_InitMemory

5.3.2 Pwm_Init

Prototype	
void **Pwm_Init** (P2CONST(Pwm_ConfigType, AUTOMATIC, PWM_APPL_CONST) ConfigPtr)	
Parameter	
ConfigPtr	Pointer to configuration set.
Return code	
-	-
Functional Description	
This function initializes the whole PWM unit of the emulated microcontroller according to the parameters specified in ConfigPtr.	
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 Pwm_Init

Note
After having finished the module initialization, all PWM channels are started with configured default values.

5.3.3 Pwm_DeInit

Prototype	
void **Pwm_DeInit** (void)	
Parameter	
-	-
Return code	
-	-
Functional Description	
Service for PWM de-initialization. This service disables notifications. After de-initialization, the output signals of all PWM channels are set to their idle.	
Particularities and Limitations	
> This function is synchronous > This function is non reentrant. > This function may only be called if the module has been initialized before. > This function shall not be called during a running operation. > This function is configurable	

Expected Caller Context
> ECU State Manager or comparable software module, responsible for driver initialization after startup.

Table 5-5 Pwm_DeInit

5.3.4 Pwm_SetDutyCycle

Prototype	
`void` **`Pwm_SetDutyCycle`** `(Pwm_ChannelType ChannelNumber, uint16 DutyCycle)`	
Parameter	
`ChannelNumber`	Identifier of a channel
`DutyCycle`	Duty cycle in proportion to the current period (range 0 (0%) to 0x8000 (100%))
Return code	
-	-
Functional Description	
This service sets the duty cycle of a PWM channel. The update of the duty cycle is performed either immediately or at the end of the period as configured with `PwmDutycycleUpdateEndperiod`.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different channel numbers. > This function may only be called if the module has been initialized before. > This function is configurable	
Expected Caller Context	
> Task Context	

Table 5-6 Pwm_SetDutyCycle

5.3.5 Pwm_SetPeriodAndDuty

Prototype	

`void` **`Pwm_SetPeriodAndDuty`**
`(`
`  Pwm_ChannelType ChannelNumber,`
`  Pwm_PeriodType Period,`
`  uint16 DutyCycle`
`)`	

Parameter	
`ChannelNumber`	Identifier of a channel
`Period`	Period value for the channel in microseconds
`DutyCycle`	Duty cycle in proportion to the period (range 0 (0%) to 0x8000 (100%))
Return code	
-	-

Functional Description
Sets the period time and duty cycle of a PWM channel. The PWM driver only changes the period for PWM channels declared as variable period type. The update of the period is performed either immediately or at the end of the period as configured with `PwmPeriodUpdateEndperiod`.
Particularities and Limitations
> This function is synchronous. > This function is reentrant for different channel numbers. > This function may only be called if the module has been initialized before. > This function is configurable
Expected Caller Context
> Task Context

Table 5-7 Pwm_SetPeriodAndDuty

5.3.6 Pwm_SetOutputToIdle

Prototype	
`void` **`Pwm_SetOutputToIdle`** `(Pwm_ChannelType ChannelNumber)`	
Parameter	
`ChannelNumber`	Identifier of a channel
Return code	
-	-
Functional Description	
This service sets the PWM output state immediately to idle level as configured in the DaVinci Configurator.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different channel numbers. > This function may only be called if the module has been initialized before. > This function is configurable	
Expected Caller Context	
> Task Context	

Table 5-8 Pwm_SetOutputToIdle

5.3.7 Pwm_GetOutputState

Prototype	
`Pwm_OutputStateType` **`Pwm_GetOutputState`** `(Pwm_ChannelType ChannelNumber)`	
Parameter	
`ChannelNumber`	Identifier of a channel
Return code	
`Pwm_OutputStateType`	The current level on channel’s output pin

Functional Description
This service returns the internal state of the PWM output signal. I. e. that this value is the state between the PWM Unit and the Port Logic. So even the output is set to idle the internal state is alternating (according to ref. [1])
Particularities and Limitations
> This function is synchronous. > This function is reentrant for different channel numbers. > This function may only be called if the module has been initialized before. > This function is configurable > Due to real time constraints, it is possible that the output state of a channel has already changed, before this function returns.
Expected Caller Context
> Task Context

Table 5-9 Pwm_GetOutputState

5.3.8 Pwm_DisableNotification

Prototype	
`void` **`Pwm_DisableNotification`** `(Pwm_ChannelType ChannelNumber)`	
Parameter	
`ChannelNumber`	Identifier of a channel
Return code	
-	-
Functional Description	
This service disables the PWM signal edge notification.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different channel numbers. > This function may only be called if the module has been initialized before. > This function is configurable	
Expected Caller Context	
> Task Context	

Table 5-10 Pwm_DisableNotification

5.3.9 Pwm_EnableNotification

Prototype

`void` **`Pwm_EnableNotification`**
`(`
`  Pwm_ChannelType ChannelNumber,`
`  Pwm_EdgeNotficationType Notification`
`)`

Parameter	
`ChannelNumber`	Identifier of a channel
`Notification`	Specifies whether notification is performed upon rising or falling edges or both.
Return code	
-	-
Functional Description	
This service enables the PWM signal edge notification according to the `Notification` parameter.	
Particularities and Limitations	
> This function is synchronous. > This function is reentrant for different channel numbers. > This function may only be called if the module has been initialized before. > This function is configurable	
Expected Caller Context	
> Task Context	

Table 5-11 Pwm_EnableNotification

5.3.10 Pwm_GetVersionInfo

Prototype	

`void` **`Pwm_GetVersionInfo`**
`(`
`  P2VAR(Std_VersionInfoType, AUTOMATIC, PWM_APPL_DATA) versioninfo`
`)`	

Parameter	
`versioninfo`	Pointer to where to store 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 service is configurable.	
Expected Caller Context	
> Task Context	

Table 5-12 Pwm_GetVersionInfo

5.4 Services used by PWM

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

Component	API
DET	Det_ReportError

Table 5-13 Services used by the PWM

5.5 Configurable Interfaces

5.5.1 Notifications

At its configurable interfaces the PWM defines notifications that can be mapped to callback functions provided by other modules. The mapping is not statically defined by the PWM 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 PwmNotification

Prototype	
`void` **`<PwmNotification>`** `(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 PWM channel.	
Particularities and Limitations	
> This function is synchronous. > These notification functions are only available, if `Pwm_EnableNotification` was called before for the assigned PWM channel has expired. > The notification functions can be configured in the configuration tool	
Call context	
> Called within simulated ISR.	

Table 5-14 PwmNotification

6 Configuration

6.1 Configuration Variants

The PWM supports the configuration variants

- `VARIANT-PRE-COMPILE`

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

6.2 Configuration with DaVinci Configurator 5

The PWM 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
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
PWM	Pulse Width Modulation
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