Document Title	Specification of PWM Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	37

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• No content changes
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• Changed ID from [SWS_Pwm_00153] to [SWS_Pwm_NA_00153]
2021-11-25	R21-11	AUTOSAR Release Management	• [SWS_Pwm_00061] removed as <Module>_ConfigTypes are introduced as implementation specific and therefore additional requirements did not make sense anymore • Runtime error added on [SWS_Pwm_00200]
2020-11-30	R20-11	AUTOSAR Release Management	• Minor corrections in section Error Classification
2019-11-28	R19-11	AUTOSAR Release Management	• Introduced MCAL Multicore Distribution • Changed Document Status from Final to published

△

2018-10-31	4.4.0	AUTOSAR Release Management	• Incporporated concept MCAL Multicore Distribution (Draft) • Removal of obsolete elements • Header File Cleanup • Fixed document structure for automated document processing
2017-12-08	4.3.1	AUTOSAR Release Management	• Added classification for Runtime error • Removed [SWS_Pwm_20069], [SWS_Pwm_10120] and [SWS_Pwm_20120]
2016-11-30	4.3.0	AUTOSAR Release Management	• Updated `Pwm_GetOutputState` return value requirement [SWS_Pwm_30051] and its references • Updated Configuration Class for `PwmChannelId` • Removed definition of Configuration variants • Removed Unresolved References of BSW requirements • Updated Header file structure diagram
2015-07-31	4.2.2	AUTOSAR Release Management	• Removed requirements with respect to NULL_PTR check • DET has been renamed
2014-10-31	4.2.1	AUTOSAR Release Management	• Updated trace reference for code file structure requirement
2014-03-31	4.1.3	AUTOSAR Release Management	• Introduction of McuClockReferencePoint • Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Updated requirements related to `PwmPowerStateAsynchTransitionMode` • Updated Scheduled Functions chapter • Editorial changes • Removed chapter(s) on change documentation

▽

△

2013-03-15	4.1.1	AUTOSAR Administration	• Added ECU degradation concept • Adapted to new SWS BSW General • Split memory map header
2011-12-22	4.0.3	AUTOSAR Administration	• Re-formulated [SWS_Pwm_00045]
2010-09-30	3.1.5	AUTOSAR Administration	• New Error symbol: PWM_E_PARAM_POINTER, shall be reported if API Pwm_GetVersionInfo service is called with a NULL parameter • Updated the chapter Version Check • Maintenance in phrasing and explaining
2010-02-02	3.1.4	AUTOSAR Administration	• The behavior of the function Pwm_SetPeriodAndDuty is explained in case of an input value of zero period. • Added the chapter Debug Support • Splitted some requirements so each ID is unique. • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• Tables generated from UML-models and UML-diagrams linked to UML-model • General improvements of requirements in preparation of CT-development • Reactivation concept for IDLE PWM channels adapted • Development error in case of already initialized module added • Document meta information extended • Small layout adaptations made

▽

△

2007-01-24	2.1.15	AUTOSAR Administration	• Updated file include structure • Added configuration macros ON/OFF for PWM APIs • Renamed configuration parameter PWM_PERIOD_UPDATED_ENDPERIOD to `PwmPeriodUpdatedEndperiod` • Updated PWM signal description figure • Legal disclaimer revised • "Advice for users" revised • "Revision Information" added
2006-05-16	2.0	AUTOSAR Administration	• Document structure adapted to common Release 2.0 SWS Template. • Modify abstraction level of PWM channel • Notifications are configurable • Update the configuration of the module
2005-05-31	1.0	AUTOSAR Administration	• Initial Release

Disclaimer

Contents

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module PWM driver.

Each PWM channel is linked to a hardware PWM which belongs to the microcontroller. The type of the PWM signal (for example center Align, left Align, Etc..) is not defined within this specification and is left up to the implementation.

The driver provides functions for initialization and control of the microcontroller internal PWM stage (pulse width modulation). The PWM module generates pulses with variable pulse width. It allows the selection of the duty cycle and the signal period time.

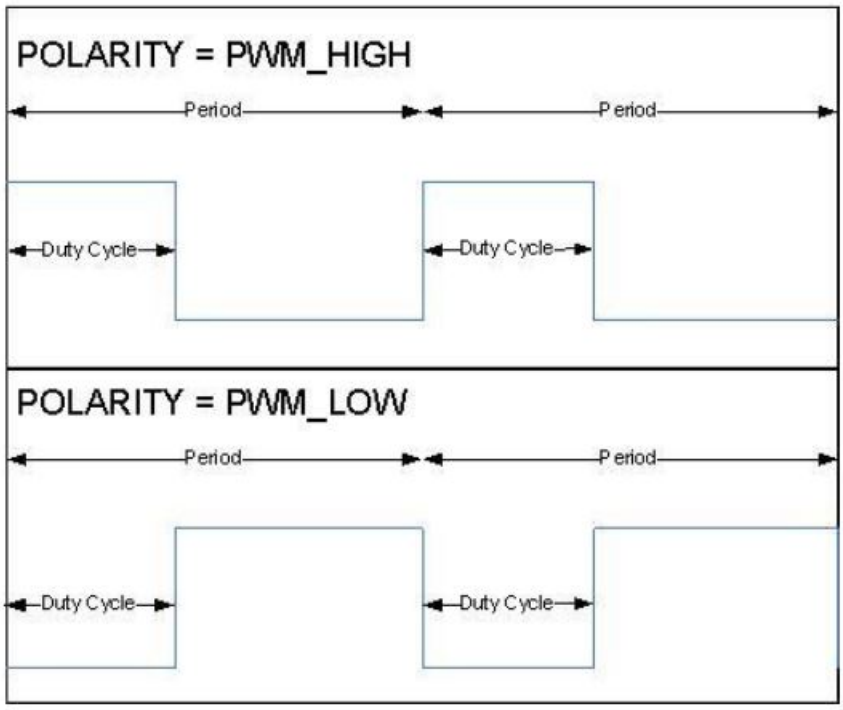


Figure 1.1: PWM Signal Description

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the PWM Driver module that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
PWM Channel	Numeric identifier linked to a hardware PWM.
PWM Output State	Defines the output state for a PWM signal. It could be: • High. • Low.
PWM Idle State	The idle state represents the output state of the PWM channel after the call of `Pwm_SetOutputToIdle` or `Pwm_DeInit`.
PWM Polarity	Defines the starting output state of each PWM channel.
PWM Duty cycle	Defines a percentage of the starting level (could be high or low) related to the period.
PWM period	Defines the period of the PWM signal.
PWM	Pulse Width Modulation.
DEM	Diagnostic Event Manager.
DET	Default Error Tracer.
MCU	Microcontroller Unit.
PLL	Phase Locked Loop.
ISR	Interrupt Service Routine.

Table 2.1: Acronyms and abbreviations used in the scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[3] Specification of Port Driver
AUTOSAR_CP_SWS_PortDriver

[4] Specification of MCU Driver
AUTOSAR_CP_SWS_MCUDriver

[5] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer

[6] Specification of ECU Configuration
AUTOSAR_CP_TPS_ECUConfiguration

[7] Requirements on PWM Driver
AUTOSAR_CP_RS_PWMDriver

[8] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[9] General Requirements on SPAL
AUTOSAR_CP_RS_SPALGeneral

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for PWM Driver.

Thus, the specification SWS BSW General shall be considered as additional and required specification for PWM Driver.

4 Constraints and assumptions

4.1 Limitations

[SWS_Pwm_00001]

Upstream requirements: SRS_Pwm_12386

⌈The Pwm SWS does not cover PWM emulation on general purpose I/O.⌋

- Power State Control APIs are implementable only if the MCAL driver owns the complete underlying HW peripheral i.e. the HW peripheral is not accessed by other MCAL modules.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

The PWM depends on the system clock. Thus, changes of the system clock (e.g. PLL on to PLL off) also affect the clock settings of the PWM hardware.

The PWM Driver depends on the following modules:

- PORT Driver [3]: To set the port pin functionality. **PWM141**
- MCU Driver [4]: To set prescaler, system clock and PLL. **PWM142**
- DET [5]: Default Error Tracer in Development mode. **PWM143**

The document [6] contains a chapter 4.6 - Clock Tree Configuration, which details the mechanism to deliver reference clock signals to peripherals.

5.1 File structure

5.1.1 Code file structure

[SWS_Pwm_00065]

Upstream requirements: SRS_BSW_00346, SRS_BSW_00314

⌈The Pwm SWS shall not define the code file structure.⌋

5.1.2 Header file structure

[SWS_Pwm_50075] ⌈Pwm.c shall include Pwm.h, Det.h and .⌋

[SWS_Pwm_70075] ⌈Pwm_Irq.c shall include Pwm.h.⌋

6 Requirements Tracing

The following tables reference the requirements specified in [7], [8], and [9], and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_Pwm_00007]
[SRS_BSW_00171]	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	[SWS_Pwm_10080] [SWS_Pwm_10082] [SWS_Pwm_10083] [SWS_Pwm_10084] [SWS_Pwm_10085] [SWS_Pwm_20080] [SWS_Pwm_20082] [SWS_Pwm_20083] [SWS_Pwm_20084] [SWS_Pwm_20085]
[SRS_BSW_00314]	All internal driver modules shall separate the interrupt frame definition from the service routine	[SWS_Pwm_00065]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_Pwm_00045] [SWS_Pwm_00047] [SWS_Pwm_00117] [SWS_Pwm_10051] [SWS_Pwm_20051] [SWS_Pwm_30051]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_Pwm_00010]
[SRS_BSW_00337]	Classification of development errors	[SWS_Pwm_20002] [SWS_Pwm_30002] [SWS_Pwm_40002] [SWS_Pwm_50002]
[SRS_BSW_00343]	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	[SWS_Pwm_00070]
[SRS_BSW_00346]	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	[SWS_Pwm_00065]
[SRS_BSW_00385]	List possible error notifications	[SWS_Pwm_20002] [SWS_Pwm_30002] [SWS_Pwm_40002] [SWS_Pwm_50002]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_Pwm_00045] [SWS_Pwm_00047] [SWS_Pwm_00117] [SWS_Pwm_10051] [SWS_Pwm_20002] [SWS_Pwm_20051] [SWS_Pwm_30002] [SWS_Pwm_30051] [SWS_Pwm_40002] [SWS_Pwm_50002]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_Pwm_00117]
[SRS_Pwm_12293]	The PWM driver shall allow the static configuration of PWM channel properties	[SWS_Pwm_00197]
[SRS_Pwm_12295]	The PWM driver shall provide a service for setting the duty cycle of a selected channel	[SWS_Pwm_00013]
[SRS_Pwm_12297]	The PWM driver shall provide a service for setting the period of a selected channel	[SWS_Pwm_00019]
[SRS_Pwm_12299]	The PWM driver shall allow to enable/ disable the PWM edges notification during runtime	[SWS_Pwm_00023] [SWS_Pwm_00024]
[SRS_Pwm_12358]	The PWM driver shall be capable to set the output of selected channel to a given state immediately	[SWS_Pwm_00021]

△

Requirement	Description	Satisfied by
[SRS_Pwm_12378]	The PWM driver shall be able to assign notification to each edges of the PWM-signal	[SWS_Pwm_00023] [SWS_Pwm_00024] [SWS_Pwm_00197]
[SRS_Pwm_12381]	By de-initializing the PWM driver, all PWM-channels shall be stop	[SWS_Pwm_00010]
[SRS_Pwm_12382]	The PWM Driver shall wait to the end of the signal period to update the duty cycle of a PWM signal	[SWS_Pwm_00017]
[SRS_Pwm_12383]	The PWM driver shall provide a 16 bit interface to set the duty cycle	[SWS_Pwm_00058]
[SRS_Pwm_12385]	The PWM driver shall provide a service to get the state of a PWM channel output	[SWS_Pwm_00022]
[SRS_Pwm_12386]	The PWM driver shall not cover a PWM emulation on general purpose I/O	[SWS_Pwm_00001]
[SRS_Pwm_12389]	The PWM driver shall allow only static configuration of the frequency for some PWM channels	[SWS_Pwm_00041]
[SRS_Pwm_12459]	The PWM Driver shall provide a scaling scheme for duty cycle	[SWS_Pwm_00059]
[SRS_SPAL_00157]	All drivers and handlers of the AUTOSAR Basic Software shall implement notification mechanisms of drivers and handlers	[SWS_Pwm_00025]
[SRS_SPAL_12057]	All driver modules shall implement an interface for initialization	[SWS_Pwm_00007] [SWS_Pwm_00052] [SWS_Pwm_00062] [SWS_Pwm_10009] [SWS_Pwm_20009] [SWS_Pwm_30009]
[SRS_SPAL_12125]	All driver modules shall only initialize the configured resources	[SWS_Pwm_00062]
[SRS_SPAL_12129]	The ISRs shall be responsible for resetting the interrupt flags and calling the according notification function	[SWS_Pwm_00026]
[SRS_SPAL_12163]	All driver modules shall implement an interface for de-initialization	[SWS_Pwm_00010] [SWS_Pwm_00011] [SWS_Pwm_00012]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 General behavior

[SWS_Pwm_00088] ⌈All functions from the PWM module except `Pwm_Init`, `Pwm_DeInit` and `Pwm_GetVersionInfo` shall be re-entrant for different PWM channel numbers.

In order to keep a simple module implementation, no check of [SWS_Pwm_00088] must be performed by the module.⌋

[SWS_Pwm_00089] ⌈The Pwm module's user shall ensure the integrity if several function calls are made during run time in different tasks or ISRs for the same PWM channel.⌋

7.2 Time Unit Ticks

7.2.1 Background & Rationale

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.

7.2.2 Requirements

[SWS_Pwm_00070]

Upstream requirements: SRS_BSW_00343

⌈All time units used within the API services of the PWM module shall be of the unit ticks.⌋

7.3 Support and management of HW low power states

Some PWM HW Module allow to be set in some operation modes which reduce the power consumption, eventually at the cost of a slower reaction time, a lower performance or eventually complete unavailability. Each PWM module could support one or

more low power operation modes, considering the Full Power Mode as always present and set per default at startup.

7.3.1 Background

The PWM Driver offers power state control APIs and a background elaboration mechanism to handle asynchronous power state change processes (i.e. power state changes which are not immediately complete as the they are requested, but need some longer operations).

It is assumed that all constraints deriving from ECU and SW architecture are already satisfied by the upper layers (Application, Mode Management in the service layer, IoHwAbstraction components dealing with peripheral control), thus the scope of control is limited to the PWM HW peripheral.

A check on the operation sequence is executed by the PWM Driver in order to avoid requesting a different power state before the previous request is still being processed or activating a power state when no preparation for the same has been requested.

The PWM module shall support power control capabilities as an optional function. This module neither mandates to use only power control enabled MCUs nor to configure the same. Rather it proposes a way to handle power states if this is supported by the suppliers.

7.3.2 Requirements

[SWS_Pwm_00154] ⌈The PwmDriver shall support power state changes and its APIs when the corresponding configuration parameter `PwmLowPowerStatesSupport` is set to TRUE.⌋

[SWS_Pwm_00155] ⌈If the parameter `PwmLowPowerStatesSupport` is enabled then the APIs `Pwm_PreparePowerState`, `Pwm_SetPowerState`, `Pwm_GetCurrentPowerState`, `Pwm_GetTargetPowerState` shall be generated and shall be used to manage and get informations on power state transitions.⌋

[SWS_Pwm_00156] ⌈The APIs `Pwm_GetTargetPowerState` and `Pwm_GetCurrentPowerState` shall be respectively used to gather information on the requested and the target Pwm power states.⌋

[SWS_Pwm_00157] ⌈The API `Pwm_PreparePowerState` shall be used to start a power state transition.⌋

[SWS_Pwm_00158] ⌈After preparation for a power state is achieved by ([SWS_Pwm_00157]) then the API `Pwm_SetPowerState` shall be used to achieve the requested power state of the Pwm module.

In order to avoid incoherent power state conditions, some APIs (`Pwm_SetPowerState`, `Pwm_PreparePowerState`) have to be called in a given sequence, otherwise an error (if DET tracing is enabled) is stored and the action is interrupted. The Pwm Driver keeps track of the call sequence.⌋

[SWS_Pwm_00159] ⌈The Pwm Driver shall keep track of the call order of the APIs `Pwm_SetPowerState` and `Pwm_PreparePowerState`. In case the first one is called before the second one is called, a DET entry shall be stored and the action shall not be executed.⌋

[SWS_Pwm_00160] ⌈The Pwm Module shall keep track of the current and of the target powerstate if the parameter `PwmLowPowerStatesSupport` is set to `TRUE`.⌋

[SWS_Pwm_00161] ⌈After the Initiliazation the power state of the module shall be always `FULL POWER` if the `PwmLowPowerStatesSupport` is set to `TRUE`.⌋

[SWS_Pwm_00162] ⌈The Pwm Driver shall support synchronuous and asynchronous power state transitions, depending on the value of the configuration parameter `PwmPowerStateAsynchTransitionMode`.⌋

[SWS_Pwm_00163] ⌈In case the configuration parameter `PwmPowerStateAsynchTransitionMode` is set to `FALSE`, the preparation process and the setting process shall be considered concluded as soon as the respective APIs return.⌋

[SWS_Pwm_00164] ⌈In case the configuration parameter `PwmPowerStateAsynchTransitionMode` is set to `TRUE`, the preparation process shall continue in background after the relative API returns and its completion shall be notified by means of the configured callback.⌋

7.4 Duty Cycle Resolution and scaling

[SWS_Pwm_00058]

Upstream requirements: SRS_Pwm_12383

⌈The width of the duty cycle parameter is 16 Bits.⌋

[SWS_Pwm_00059]

Upstream requirements: SRS_Pwm_12459

⌈The Pwm module shall comply with the following scaling scheme for the duty cycle:

- 0x0000 means 0%.
- 0x8000 means 100%. 0x8000 gives the highest resolution while allowing 100% duty cycle to be represented with a 16 bit value.

As an implementation guide, the following source code example is given:

```
AbsoluteDutyCycle = ((uint32)AbsolutePeriodTime * RelativeDutyCycle) >> 15;
```

⌋

7.5 Version check

For details refer to the chapter 5.1.8 "Version Check" in [2].

7.6 Error Classification

Section "Error Handling" of the document [2] "General Specification of Basic Software Modules" describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.6.1 Development Errors

[SWS_Pwm_00201] Definiton of development errors in module Pwm ⌈

Type of error	*Related error code*	*Error value*
API Pwm_Init service called with wrong parameter	PWM_E_INIT_FAILED	0x10
API service used without module initialization	PWM_E_UNINIT	0x11
API service used with an invalid channel Identifier	PWM_E_PARAM_CHANNEL	0x12
Usage of unauthorized PWM service on PWM channel configured a fixed period	PWM_E_PERIOD_UNCHANGEABLE	0x13
API Pwm_Init service called while the PWM driver has already been initialised	PWM_E_ALREADY_INITIALIZED	0x14

▽

△

Type of error	*Related error code*	*Error value*
API Pwm_GetVersionInfo is called with a NULL parameter.	PWM_E_PARAM_POINTER	0x15
The requested power state is not supported by the PWM module.	PWM_E_POWER_STATE_NOT_SUPPORTED	0x17
The requested power state is not reachable from the current one	PWM_E_TRANSITION_NOT_POSSIBLE	0x18
API Pwm_SetPowerState has been called without having called the API Pwm_PreparePowerState before.	PWM_E_PERIPHERAL_NOT_PREPARED	0x19

⌋

[SWS_Pwm_20002]

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00386

⌈The PWM Driver module shall report the development error PWM_E_UNINIT, when API service is used without module initialization.⌋

[SWS_Pwm_30002]

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00386

⌈The PWM Driver module shall report the development error PWM_E_PARAM_CHANNEL, when API service is used with an invalid channel Identifier.⌋

[SWS_Pwm_40002]

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00386

⌈The PWM Driver module shall report the development error PWM_E_PERIOD_UNCHANGEABLE, on usage of unauthorized PWM service on PWM channel configured a fixed period.⌋

[SWS_Pwm_50002]

Upstream requirements: SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00386

⌈The PWM Driver module shall report the development error PWM_E_ALREADY_INITIALIZED, when API Pwm_Init service is called while the PWM driver has already been initialized.⌋

[SWS_Pwm_00174] ⌈The API shall report the development error PWM_E_POWER_STATE_NOT_SUPPORTED in case this API is called with an unsupported power state or the peripheral does not support low power states at all.⌋

[SWS_Pwm_00175] ⌈The API shall report the development error PWM_E_TRANSITION_NOT_POSSIBLE in case the requested power state cannot be directly reached from the current power state.⌋

[SWS_Pwm_00176] ⌈The API shall report the development error PWM_E_PERIPHERAL_NOT_PREPARED in case the HW unit has not been previously prepared for the target power state by use of the API Pwm_PreparePowerState.⌋

To get more details concerning error detection, refer to Section 8.8 "API parameter checking".

7.6.2 Runtime Errors

[SWS_Pwm_00202] Definiton of runtime errors in module Pwm ⌈

Type of error	*Related error code*	*Error value*
API Pwm_SetPowerState is called while the PWM module is still in use.	PWM_E_NOT_DISENGAGED	0x16

⌋

[SWS_Pwm_00200] ⌈The API shall report the runtime error PWM_E_NOT_DISENGAGED in case this API is called when one or more HW channels (where applicable) are in a state different than IDLE (or similar non-operational states) and/or there are still notification registered for the HW module channels.⌋

7.6.3 Production Errors

There are no production errors.

7.6.4 Extended Production Errors

There are no extended production errors.

7.7 Security Events

The module does not report security events.

8 API specification

8.1 Imported types

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

[SWS_Pwm_00094] Definition of imported datatypes of module Pwm ⌈

Module	*Header File*	*Imported Type*
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋

8.2 Type definitions

8.2.1 Pwm_ChannelType

[SWS_Pwm_00106] Definition of datatype Pwm_ChannelType ⌈

Name	Pwm_ChannelType		
Kind	Type		
Derived from	uint		
Range	8..32 bit	–	This is implementation specific but not all values may be valid within the type. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.
Description	Numeric identifier of a PWM channel.		
Available via	Pwm.h		

⌋

8.2.2 Pwm_PeriodType

[SWS_Pwm_00107] Definition of datatype Pwm_PeriodType ⌈

Name	Pwm_PeriodType		
Kind	Type		
Derived from	uint		
Range	8..32 bit	–	Implementation specific. This type shall be chosen in order to have the most efficient implementation on a specific microcontroller platform.
Description	Definition of the period of a PWM channel.		
Available via	Pwm.h		

⌋

8.2.3 Pwm_OutputStateType

[SWS_Pwm_00108] Definition of datatype Pwm_OutputStateType ⌈

Name	Pwm_OutputStateType		
Kind	Enumeration		
Range	PWM_HIGH	0x00	The PWM channel is in high state.
	PWM_LOW	0x01	The PWM channel is in low state.
Description	Output state of a PWM channel.		
Available via	Pwm.h		

⌋

8.2.4 Pwm_EdgeNotificationType

[SWS_Pwm_00109] Definition of datatype Pwm_EdgeNotificationType ⌈

Name	Pwm_EdgeNotificationType		
Kind	Enumeration		
Range	PWM_RISING_EDGE	0x00	Notification will be called when a rising edge occurs on the PWM output signal.
	PWM_FALLING_EDGE	0x01	Notification will be called when a falling edge occurs on the PWM output signal.

▽

△

	PWM_BOTH_EDGES	0x02	Notification will be called when either a rising edge or falling edge occur on the PWM output signal.
Description	Definition of the type of edge notification of a PWM channel.		
Available via	Pwm.h		

⌋

8.2.5 Pwm_ChannelClassType

[SWS_Pwm_00110] Definition of datatype Pwm_ChannelClassType ⌈

Name	Pwm_ChannelClassType		
Kind	Enumeration		
Range	PWM_VARIABLE_PERIOD	0x00	The PWM channel has a variable period. The duty cycle and the period can be changed.
	PWM_FIXED_PERIOD	0x01	The PWM channel has a fixed period. Only the duty cycle can be changed.
	PWM_FIXED_PERIOD_SHIFTED	0x02	The PWM channel has a fixed shifted period. Impossible to change it (only if supported by hardware)
Description	Defines the class of a PWM channel		
Available via	Pwm.h		

⌋

8.2.6 Pwm_ConfigType

[SWS_Pwm_00111] Definition of datatype Pwm_ConfigType ⌈

Name	Pwm_ConfigType	
Kind	Structure	
Elements	Hardware dependent structure.	
	Type	–
	Comment	The contents of the initialization data structure are hardware specific.
Description	This is the type of data structure containing the initialization data for the PWM driver.	
Available via	Pwm.h	

⌋

8.2.7 Pwm_PowerStateRequestResultType

[SWS_Pwm_00165] Definition of datatype Pwm_PowerStateRequestResultType

⌈

Name	Pwm_PowerStateRequestResultType		
Kind	Enumeration		
Range	PWM_SERVICE_ ACCEPTED	0x00	Power state change executed.
	PWM_NOT_INIT	0x01	PWM Module not initialized.
	PWM_SEQUENCE_ ERROR	0x02	Wrong API call sequence.
	PWM_HW_FAILURE	0x03	The HW module has a failure which prevents it to enter the required power state.
	PWM_POWER_STATE_ NOT_SUPP	0x04	PWM Module does not support the requested power state.
	PWM_TRANS_NOT_ POSSIBLE	0x05	PWM Module cannot transition directly from the current power state to the requested power state or the HW peripheral is still busy.
Description	Result of the requests related to power state transitions.		
Available via	Pwm.h		

⌋

8.2.8 Pwm_PowerStateType

[SWS_Pwm_00197] Definition of datatype Pwm_PowerStateType

Upstream requirements: SRS_Pwm_12293, SRS_Pwm_12378

⌈

Name	Pwm_PowerStateType		
Kind	Enumeration		
Range	1..255	–	power modes with decreasing power consumptions.
	PWM_FULL_POWER	0x00	Full Power
Description	Power state currently active or set as target power state.		
Available via	Pwm.h		

⌋

Mandatory parameters:

- Assigned HW channel
- Default value for period

- Default value for duty cycle
- Polarity (high or low)
- Idle state high or low
- Channel class:
  - Fixed period
  - Fixed period, shifted (if supported by hardware)
  - Variable period

Optional parameters (if supported by hardware):

- Channel phase shift
- Reference channel for phase shift
- Microcontroller specific channel properties

8.3 Function definitions

8.3.1 Pwm_Init

[SWS_Pwm_00095] Definition of API function Pwm_Init ⌈

Service Name	Pwm_Init	
Syntax	`void Pwm_Init ( const Pwm_ConfigType* ConfigPtr )`	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to configuration set
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service for PWM initialization.	
Available via	Pwm.h	

⌋

[SWS_Pwm_00007]

Upstream requirements: SRS_BSW_00101, SRS_SPAL_12057

⌈The function `Pwm_Init` shall initialize all internals variables and the used PWM structure of the microcontroller according to the parameters specified in `ConfigPtr`.⌋

[SWS_Pwm_00062]

Upstream requirements: SRS_SPAL_12057, SRS_SPAL_12125

⌈The function `Pwm_Init` shall only initialize the configured resources and shall not touch resources that are not configured in the configuration file.⌋

[SWS_Pwm_10009]

Upstream requirements: SRS_SPAL_12057

⌈The function `Pwm_Init` shall start all PWM channels with the configured default values.⌋

If the duty cycle parameter equals:

- **[SWS_Pwm_20009]**

  Upstream requirements: SRS_SPAL_12057

  ⌈0% or 100% : Then the PWM output signal shall be in the state according to the configured polarity parameter⌋

- **[SWS_Pwm_30009]**

  Upstream requirements: SRS_SPAL_12057

  ⌈>0% and <100%: Then the PWM output signal shall be modulated according to parameters period, duty cycle and configured polarity.⌋

[SWS_Pwm_00052]

Upstream requirements: SRS_SPAL_12057

⌈The function `Pwm_Init` shall disable all notifications.⌋

The reason is that the users of these notifications may not be ready. They can call `Pwm_EnableNotification` to start notifications.

[SWS_Pwm_00093] ⌈The users of the Pwm module shall not call the function `Pwm_Init` during a running operation.⌋

[SWS_Pwm_00116] ⌈The Pwm module's environment shall not call any function of the Pwm module before having called `Pwm_Init`.⌋

[SWS_Pwm_00118] ⌈If development error detection is enabled, calling the routine `Pwm_Init` while the PWM driver and hardware are already initialized will cause a development error `PWM_E_ALREADY_INITIALIZED`. The desired functionality shall be left without any action.⌋

[SWS_Pwm_00121] ⌈A re-initialization of the Pwm driver by executing the `Pwm_Init` function requires a de-initialization before by executing a `Pwm_DeInit`.⌋

Regarding error detection, the requirement [SWS_Pwm_10051] and [SWS_Pwm_20051] are applicable to the function `Pwm_Init`.

8.3.2 Pwm_DeInit

[SWS_Pwm_00096] Definition of API function Pwm_DeInit ⌈

Service Name	Pwm_DeInit
Syntax	`void Pwm_DeInit ( void )`
Service ID [hex]	0x01
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Service for PWM De-Initialization.
Available via	Pwm.h

⌋

[SWS_Pwm_00010]

Upstream requirements: SRS_BSW_00336, SRS_SPAL_12163, SRS_Pwm_12381

⌈The function `Pwm_DeInit` shall de-initialize the PWM module.⌋

[SWS_Pwm_00011]

Upstream requirements: SRS_SPAL_12163

⌈The function `Pwm_DeInit` shall set the state of the PWM output signals to the idle state.⌋

[SWS_Pwm_00012]

Upstream requirements: SRS_SPAL_12163

⌈The function `Pwm_DeInit` shall disable PWM interrupts and PWM signal edge notifications.⌋

[SWS_Pwm_10080]

Upstream requirements: SRS_BSW_00171

⌈The function `Pwm_DeInit` shall be pre compile time configurable On/Off by the configuration parameter: `PwmDeInitApi`.⌋

[SWS_Pwm_20080]

Upstream requirements: SRS_BSW_00171

⌈The function `Pwm_DeInit` shall be configurable On/Off by the configuration parameter `PwmDeInitApi` {`PWM_DE_INIT_API`}.

Regarding error detection, the requirements [SWS_Pwm_00117], [SWS_Pwm_10051], and [SWS_Pwm_20051] are applicable to the function `Pwm_DeInit`.⌋

8.3.3 Pwm_SetDutyCycle

[SWS_Pwm_91000] Definition of API function Pwm_SetDutyCycle ⌈

Service Name	Pwm_SetDutyCycle	
Syntax	`void Pwm_SetDutyCycle ( Pwm_ChannelType ChannelNumber, uint16 DutyCycle )`	
Service ID [hex]	0x02	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the PWM
	DutyCycle	Min=0x0000 Max=0x8000
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service sets the duty cycle of the PWM channel.	
Available via	Pwm.h	

⌋

[SWS_Pwm_00013]

Upstream requirements: SRS_Pwm_12295

⌈The function `Pwm_SetDutyCycle` shall set the duty cycle of the PWM channel.⌋

[SWS_Pwm_00014] ⌈When the requested duty cycle is either 0% or 100%, the function

`Pwm_SetDutyCycle` shall set the PWM output state to either `PWM_HIGH` or `PWM_LOW`, with regard to both the configured polarity parameter and the requested duty cycle.

Thus for 0% requested Duty Cycle the output will be the inverse of the configured polarity parameter, and for 100% Duty Cycle the output will be equal to the configured polarity parameter.⌋

[SWS_Pwm_00016] ⌈The function `Pwm_SetDutyCycle` shall modulate the PWM output signal according to parameters period, duty cycle and configured polarity, when the duty cycle > 0 % and < 100%.⌋

[SWS_Pwm_00017]

Upstream requirements: SRS_Pwm_12382

⌈The function `Pwm_SetDutyCycle` shall update the duty cycle always at the end of the period if supported by the implementation and configured with `PwmDutycycleUpdatedEndperiod`.⌋

Regarding format definition of duty cycle parameter, the requirement [SWS_Pwm_00058] is applicable to the function `Pwm_SetDutyCycle`.

Regarding scaling definition of duty cycle parameter, the requirement [SWS_Pwm_00059] is applicable to the function `Pwm_SetDutyCycle`.

[SWS_Pwm_00018] ⌈The driver shall forbid the spike on the PWM output signal.⌋

Regarding error detection, the requirements [SWS_Pwm_00117], [SWS_Pwm_00047], [SWS_Pwm_10051] and [SWS_Pwm_20051] are applicable to the function `Pwm_SetDutyCycle`.

[SWS_Pwm_10082]

Upstream requirements: SRS_BSW_00171

⌈The function `Pwm_SetDutyCycle` shall be pre compile time configurable On/Off by the configuration parameter: `PwmSetDutyCycle`.⌋

[SWS_Pwm_20082]

Upstream requirements: SRS_BSW_00171

⌈The function `Pwm_SetDutyCycle` shall be configurable On/Off by the configuration parameter: `PwmSetDutyCycle` {`PWM_SET_DUTY_CYCLE_API`}.⌋

8.3.4 Pwm_SetPeriodAndDuty

[SWS_Pwm_91001] Definition of API function Pwm_SetPeriodAndDuty ⌈

Service Name	Pwm_SetPeriodAndDuty	
Syntax	`void Pwm_SetPeriodAndDuty ( Pwm_ChannelType ChannelNumber, Pwm_PeriodType Period, uint16 DutyCycle )`	
Service ID [hex]	0x03	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the PWM
	Period	Period of the PWM signal
	DutyCycle	Min=0x0000 Max=0x8000
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service sets the period and the duty cycle of a PWM channel	
Available via	Pwm.h	

⌋

[SWS_Pwm_00019]

Upstream requirements: SRS_Pwm_12297

⌈The function `Pwm_SetPeriodAndDuty` shall set the period and the duty cycle of a PWM channel.⌋

[SWS_Pwm_00076] ⌈The function `Pwm_SetPeriodAndDuty` shall update the period always at the end of the current period if supported by the implementation and configured with `PwmPeriodUpdatedEndperiod`.⌋

[SWS_Pwm_00020] ⌈When updating the PWM period and duty, the driver shall repress any spikes on the PWM output signal.⌋

The PWM duty cycle parameter is necessary to maintain the consistency between frequency and duty cycle. Refer to [SWS_Pwm_00058] and [SWS_Pwm_00059] to know the scaling and format definition of duty cycle parameter

Regarding error detection, the requirements [SWS_Pwm_00117], [SWS_Pwm_00045], [SWS_Pwm_00047], [SWS_Pwm_10051] and [SWS_Pwm_20051] are applicable to the function `Pwm_SetPeriodAndDuty`.

[SWS_Pwm_00041]

Upstream requirements: SRS_Pwm_12389

⌈The function `Pwm_SetPeriodAndDuty` shall allow changing the period only for the PWM channel declared as variable `Period` type.⌋

[SWS_Pwm_10083]

Upstream requirements: SRS_BSW_00171

⌈The function `Pwm_SetPeriodAndDuty` shall be pre compile time configurable On/Off by the configuration parameter: `PwmSetPeriodAndDuty`.⌋

[SWS_Pwm_20083]

Upstream requirements: SRS_BSW_00171

⌈The function `Pwm_SetPeriodAndDuty` shall be configurable On/Off by the configuration parameter: `PwmSetPeriodAndDuty` {`PWM_SET_PERIOD_AND_DUTY_API`}.⌋

[SWS_Pwm_00150] ⌈If the period is set to zero the setting of the duty-cycle is not relevant. In this case the output shall be zero (zero percent duty-cycle).⌋

8.3.5 Pwm_SetOutputToIdle

[SWS_Pwm_91002] Definition of API function Pwm_SetOutputToIdle ⌈

Service Name	Pwm_SetOutputToIdle	
Syntax	`void Pwm_SetOutputToIdle ( Pwm_ChannelType ChannelNumber )`	
Service ID [hex]	0x04	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the PWM
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	None
Description	Service sets the PWM output to the configured Idle state.
Available via	Pwm.h

⌋

[SWS_Pwm_00021]

Upstream requirements: SRS_Pwm_12358

⌈The function `Pwm_SetOutputToIdle` shall set immediately the PWM output to the configured Idle state.⌋

Regarding error detection, the requirements [SWS_Pwm_00117], [SWS_Pwm_00047], [SWS_Pwm_10051] and [SWS_Pwm_20051] are applicable to the function `Pwm_SetOutputToIdle`.

[SWS_Pwm_10084]

Upstream requirements: SRS_BSW_00171

⌈The function `Pwm_SetOutputToIdle` shall be pre compile time configurable On/Off by the configuration parameter: `PwmSetOutputToIdle`.⌋

[SWS_Pwm_20084]

Upstream requirements: SRS_BSW_00171

⌈The function `Pwm_SetOutputToIdle` shall be configurable On/Off by the configuration parameter: `PwmSetOutputToIdle` {`PWM_SET_OUTPUT_TO_IDLE_API`}.⌋

[SWS_Pwm_10086] ⌈After the call of the function `Pwm_SetOutputToIdle`, variable period type channels shall be reactivated using the Api `Pwm_SetPeriodAndDuty` to activate the PWM channel with the new passed period.⌋

[SWS_Pwm_20086] ⌈After the call of the function `Pwm_SetOutputToIdle`, channels shall be reactivated using the Api `Pwm_SetDutyCycle` to activate the PWM channel with the old period.⌋

[SWS_Pwm_00119] ⌈After the call of the function `Pwm_SetOutputToIdle`, fixed period type channels shall be reactivated using only the API `Pwm_SetDutyCycle` to activate the PWM channel with the old period.⌋

8.3.6 Pwm_GetOutputState

[SWS_Pwm_00100] Definition of API function Pwm_GetOutputState ⌈

Service Name	Pwm_GetOutputState	
Syntax	`Pwm_OutputStateType Pwm_GetOutputState ( Pwm_ChannelType ChannelNumber )`	
Service ID [hex]	0x05	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the PWM
Parameters (inout)	None	
Parameters (out)	None	
Return value	Pwm_OutputStateType	`PWM_HIGH` The PWM output state is high `PWM_LOW` The PWM output state is low
Description	Service to read the internal state of the PWM output signal.	
Available via	Pwm.h	

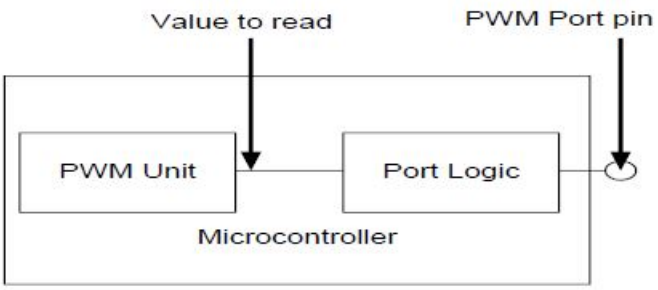
⌋

[SWS_Pwm_00022]

Upstream requirements: SRS_Pwm_12385

⌈The function `Pwm_GetOutputState` shall read the internal state of the PWM output signal and return it as defined in the diagram below.

Regarding error detection, the requirements [SWS_Pwm_00117], [SWS_Pwm_00047], [SWS_Pwm_10051] and [SWS_Pwm_20051] are applicable to the function `Pwm_GetOutputState`.⌋



[SWS_Pwm_10085]

Upstream requirements: SRS_BSW_00171

⌈The function `Pwm_GetOutputState` shall be pre compile time configurable On/Off using the configuration parameter: `PwmGetOutputState`.⌋

[SWS_Pwm_20085]

Upstream requirements: SRS_BSW_00171

⌈The function `Pwm_GetOutputState` shall be configurable On/Off by the configuration parameter: `PwmGetOutputState` {`PWM_GET_OUTPUT_STATE_API`}.

Due to real time constraint and setting of the PWM channel (project dependant), the output state can be modified just after the call of the service `Pwm_GetOutputState`.⌋

[SWS_Pwm_30051]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈If `Pwm_GetOutputState` is called before module initialization, or with an invalid channel, it shall return `PWM_LOW`.⌋

8.3.7 Pwm_DisableNotification

[SWS_Pwm_91003] Definition of API function Pwm_DisableNotification ⌈

Service Name	Pwm_DisableNotification	
Syntax	`void Pwm_DisableNotification ( Pwm_ChannelType ChannelNumber )`	
Service ID [hex]	0x06	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the PWM
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service to disable the PWM signal edge notification.	
Available via	Pwm.h	

⌋

[SWS_Pwm_00023]

Upstream requirements: SRS_Pwm_12378, SRS_Pwm_12299

⌈The function `Pwm_DisableNotification` shall disable the PWM signal edge notification.⌋

[SWS_Pwm_10112] ⌈The function `Pwm_DisableNotification` shall be pre compile time configurable On/Off using the configuration parameter: `PwmNotificationSupported`.⌋

[SWS_Pwm_20112] ⌈The function `Pwm_DisableNotification` shall be configurable On/Off by the configuration parameter: `PwmNotificationSupported` {`PWM_NOTIFICATION_SUPPORTED`}.

Regarding error detection, the requirements [SWS_Pwm_00117], [SWS_Pwm_00047], [SWS_Pwm_10051] and [SWS_Pwm_20051] are applicable to the function `Pwm_DisableNotification`.⌋

8.3.8 Pwm_EnableNotification

[SWS_Pwm_91004] Definition of API function Pwm_EnableNotification ⌈

Service Name	Pwm_EnableNotification	
Syntax	`void Pwm_EnableNotification ( Pwm_ChannelType ChannelNumber, Pwm_EdgeNotificationType Notification )`	
Service ID [hex]	0x07	
Sync/Async	Asynchronous	
Reentrancy	Reentrant for different channel numbers	
Parameters (in)	ChannelNumber	Numeric identifier of the PWM
	Notification	Type of notification PWM_RISING_EDGE or PWM_FALLING_EDGE or PWM_BOTH_EDGES
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service to enable the PWM signal edge notification according to notification parameter.	
Available via	Pwm.h	

⌋

[SWS_Pwm_00024]

Upstream requirements: SRS_Pwm_12378, SRS_Pwm_12299

⌈The function `Pwm_EnableNotification` shall enable the PWM signal edge notification according to notification parameter.⌋

[SWS_Pwm_00081] ⌈The function `Pwm_EnableNotification` shall cancel pending interrupts.⌋

[SWS_Pwm_10113] ⌈The function `Pwm_EnableNotification` shall be pre compile time configurable On/Off using the configuration parameter: `PwmNotificationSupported`.⌋

[SWS_Pwm_20113] ⌈The function `Pwm_EnableNotification` shall be configurable On/Off by the configuration parameter: `PwmNotificationSupported` {`PWM_NOTIFICATION_SUPPORTED`}.

Regarding error detection, the requirements [SWS_Pwm_00117], [SWS_Pwm_00047], [SWS_Pwm_10051] and [SWS_Pwm_20051] are applicable to the function `Pwm_EnableNotification`.⌋

8.3.9 Pwm_SetPowerState

[SWS_Pwm_00166] Definition of API function Pwm_SetPowerState ⌈

Service Name	Pwm_SetPowerState	
Syntax	`Std_ReturnType Pwm_SetPowerState ( Pwm_PowerStateRequestResultType* Result )`	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	Result	If the API returns E_OK: PWM_SERVICE_ACCEPTED:Power state change executed. If the API returns E_NOT_OK: PWM_NOT_INIT: PWM Module not initialized. PWM_SEQUENCE_ERROR: wrong API call sequence. PWM_HW_FAILURE: the HW module has a failure which prevents it to enter the required power state.
Return value	Std_ReturnType	`E_OK`: Power Mode changed `E_NOT_OK`: request rejected
Description	This API configures the Pwm module so that it enters the already prepared power state, chosen between a predefined set of configured ones.	
Available via	Pwm.h	

⌋

[SWS_Pwm_00167] ⌈The API configures the HW in order to enter the given Power State. All preliminary actions to enable this transition (e.g. setting all channels in `IDLE` status, de-registering of all notifications and so on) must already have been taken by the responsible SWCs (e.g. IoHwAbs).

The API shall not execute preliminary, implicit power state changes (i.e. if a requested power state is not reachable starting from the current one, no intermediate power state change shall be executed and the request shall be rejected).⌋

[SWS_Pwm_00168] ⌈In case the target power state is the same as the current one, no action is executed and the API returns immediately with an `E_OK` result.⌋

[SWS_Pwm_00169] ⌈In case the normal Power State is requested, the API shall refer to the necessary parameters contained in the same containers used by `Pwm_Init`.

No separate container or hard coded data shall be used for the normal (i.e. full) power mode, in order to avoid misalignments between initialization parameters used during the init phase and during a power state change.⌋

[SWS_Pwm_00170] ⌈For the other power states, only power state transition specific reconfigurations shall be executed in the context of this API (i.e. the API cannot be used to apply a completely new configuration to the Pwm module). Any other reconfiguration not strictly related to the power state transition shall not take place.⌋

[SWS_Pwm_00171] ⌈The API shall refer to the configuration container related to the required Power State in order to derive some specific features of the state (e.g support of Power States).⌋

In case development error reporting is activated:

[SWS_Pwm_00172] ⌈The API shall report the development error `PWM_E_UNINIT` in case this API is called before having initialized the HW unit.⌋

[SWS_Pwm_00173] ⌈The API shall report the runtime error `PWM_E_NOT_DISENGAGED` in case this API is called when one or more HW channels (where applicable) are in a state different than IDLE (or similar non-operational states) and/or there are still notification registered for the HW module channels.⌋

[SWS_Pwm_00194] ⌈The API shall report the development error `PWM_E_POWER_STATE_NOT_SUPPORTED` in case this API is called with an unsupported power state or the peripheral does not support low power states at all.⌋

[SWS_Pwm_00195] ⌈The API shall report the development error `PWM_E_TRANSITION_NOT_POSSIBLE` in case the requested power state cannot be directly reached from the current power state.⌋

[SWS_Pwm_00196] ⌈The API shall report the development error `PWM_E_PERIPHERAL_NOT_PREPARED` in case the HW unit has not been previously prepared for the target power state by use of the API `Pwm_PreparePowerState`.⌋

8.3.10 Pwm_GetCurrentPowerState

[SWS_Pwm_00177] Definition of API function Pwm_GetCurrentPowerState ⌈

Service Name	Pwm_GetCurrentPowerState	
Syntax	`Std_ReturnType Pwm_GetCurrentPowerState ( Pwm_PowerStateType* CurrentPowerState, Pwm_PowerStateRequestResultType* Result )`	
Service ID [hex]	0x0a	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	CurrentPowerState	The current power mode of the PWM HW Unit is returned in this parameter
	Result	If the API returns E_OK: PWM_SERVICE_ACCEPTED: Current power mode was returned. If the API returns E_NOT_OK: PWM_NOT_INIT: PWM Module not initialized.
Return value	Std_ReturnType	`E_OK`: Mode could be read `E_NOT_OK`: Service is rejected
Description	This API returns the current power state of the PWM HW unit.	
Available via	Pwm.h	

⌋

[SWS_Pwm_00178] ⌈The API returns the power state of the HW unit.⌋

In case development error reporting is activated:

[SWS_Pwm_00179] ⌈The API shall report the development error `PWM_E_UNINIT` in case this API is called before having initialized the HW unit.⌋

8.3.11 Pwm_GetTargetPowerState

[SWS_Pwm_00180] Definition of API function Pwm_GetTargetPowerState ⌈

Service Name	Pwm_GetTargetPowerState
Syntax	`Std_ReturnType Pwm_GetTargetPowerState ( Pwm_PowerStateType* TargetPowerState, Pwm_PowerStateRequestResultType* Result )`
Service ID [hex]	0x0b

▽

△

Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	TargetPowerState	The Target power mode of the PWM HW Unit is returned in this parameter
	Result	If the API returns E_OK: PWM_SERVICE_ACCEPTED:Target power mode was returned. If the API returns E_NOT_OK: PWM_NOT_INIT: PWM Module not initialized.
Return value	Std_ReturnType	`E_OK`: Mode could be read `E_NOT_OK`: Service is rejected
Description	This API returns the Target power state of the PWM HW unit.	
Available via	Pwm.h	

⌋

[SWS_Pwm_00181] ⌈The API returns the requested power state of the HW unit. This shall coincide with the current power state if no transition is ongoing.

The API is considered to always succeed except in case of HW failures.⌋

In case development error reporting is activated:

[SWS_Pwm_00182] ⌈The API shall report the development error `PWM_E_UNINIT` in case this API is called before having initialized the HW unit.⌋

8.3.12 Pwm_PreparePowerState

[SWS_Pwm_00183] Definition of API function Pwm_PreparePowerState ⌈

Service Name	Pwm_PreparePowerState	

Syntax	`Std_ReturnType Pwm_PreparePowerState (`
	`  Pwm_PowerStateType PowerState,`
	`  Pwm_PowerStateRequestResultType* Result`
	`)`	

Service ID [hex]	0x0c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	PowerState	The target power state intended to be attained
Parameters (inout)	None	

▽

△

Parameters (out)	Result	If the API returns E_OK: PWM_SERVICE_ACCEPTED: PWM Module power state preparation was started. If the API returns E_NOT_OK: PWM_NOT_INIT: PWM Module not initialized. PWM_SEQUENCE_ERROR: wrong API call sequence (Current Power State = Target Power State). PWM_POWER_STATE_NOT_SUPP: PWM Module does not support the requested power state. PWM_TRANS_NOT_POSSIBLE: PWM Module cannot transition directly from the current power state to the requested power state or the HW peripheral is still busy.
Return value	Std_ReturnType	E_OK: Preparation process started E_NOT_OK: Service is rejected
Description	This API starts the needed process to allow the PWM HW module to enter the requested power state.	
Available via	Pwm.h	

⌋

[SWS_Pwm_00184] ⌈This API initiates all actions needed to enable a HW module to enter the target power state.

The possibility to operate the periphery depends on the power state and the HW features. These properties should be known to the integrator and the decision whether to use the periphery or not is in his responsibility.⌋

[SWS_Pwm_00185] ⌈In case the target power state is the same as the current one, no action is executed and the API returns immediately with an E_OK result.

The responsibility of the preconditions is left to the environment.⌋

In case development error reporting is activated:

[SWS_Pwm_00186] ⌈The API shall report the development error PWM_E_UNINIT in case this API is called before having initialized the HW unit.⌋

[SWS_Pwm_00187] ⌈The API shall report the development error PWM_E_POWER_STATE_NOT_SUPPORTED in case this API is called with an unsupported power state is requested or the peripheral does not support low power states at all.⌋

[SWS_Pwm_00188] ⌈The API shall report the development error PWM_E_TRANSITION_NOT_POSSIBLE in case the requested power state cannot be directly reached from the current power state.

All asynchronous operation needed to reach the target power state can be executed in background in the context of Pwm_Main_PowerTransitionManager.⌋

8.3.13 Pwm_GetVersionInfo

[SWS_Pwm_00103] Definition of API function Pwm_GetVersionInfo ⌈

Service Name	Pwm_GetVersionInfo	
Syntax	`void Pwm_GetVersionInfo ( Std_VersionInfoType* versioninfo )`	
Service ID [hex]	0x08	
Sync/Async	Synchronous	
Reentrancy	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	Service returns the version information of this module.	
Available via	Pwm.h	

⌋

8.4 Callback notifications

Since the PWM Driver is a module on the lowest architectural layer it doesn't provide any call-back functions for lower layer modules.

8.5 Scheduled functions

All services offered by the PWM Driver are of synchronous nature, with the exception of the asynchronous power transition management, if so configured.

In case the synchronous power transition management is configured, no scheduled API is generated.

8.5.1 Pwm_Main_PowerTransitionManager

[SWS_Pwm_00189] Definition of scheduled function Pwm_Main_PowerTransitionManager ⌈

Service Name	Pwm_Main_PowerTransitionManager

Syntax	`void Pwm_Main_PowerTransitionManager (`
	`  void`
	`)`

Service ID [hex]	0x0d
Description	This API is cyclically called and supervises the power state transitions, checking for the readiness of the module and issuing the callbacks IoHwAb_Pwm_NotifyReadyForPower State<Mode> (see PwmPowerStateReadyCbkRef configuration parameter).
Available via	SchM_Pwm.h

⌋

[SWS_Pwm_00190] ⌈This API executes any non-immediate action needed to finalize a power state transition requested by `Pwm_PreparePowerState`.⌋

[SWS_Pwm_00191] ⌈The rate of scheduling shall be defined by Pwm MainSchedulePeriod and shall be variable, as the function only needs to be called if a transition has been requested.⌋

[SWS_Pwm_00192] ⌈This API shall also issue callback notifications to the eventually registered users (IoHwAbs) as configured, only in case the asynch mode is chosen.⌋

[SWS_Pwm_00193] ⌈In case the PWM module is not initialized, this function shall simply return without any further elaboration. This is needed to avoid to elaborate uninitialized variables. No development error shall be entered, because this condition can easily be verified during the startup phase (tasks started before the initialization is complete).

Rationale: during the startup phase it can happen that the OS already schedules tasks, which call main functions, while some modules are not initialised yet. This is no real error condition, although need handling, i.e. returning without execution.

Although the transition state monitoring functionality is mandatory, the implementation of this API is optional, meaning that if the HW allows for other ways to deliver notification and watch the transition state the implementation of this function can be skipped.⌋

8.6 Expected interfaces

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

8.6.1 Mandatory interfaces

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

As this module is part of the MCAL layer, it access directly to the microcontroller registers and therefore doesn't need any lower interfaces.

[SWS_Pwm_91006] Definition of mandatory interfaces required by module Pwm ⌈

API Function	*Header File*	*Description*
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.

⌋

8.6.2 Optional interfaces

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

[SWS_Pwm_00104] Definition of optional interfaces requested by module Pwm ⌈

API Function	*Header File*	*Description*
Det_ReportError	Det.h	Service to report development errors.

⌋

8.6.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

[SWS_Pwm_00105] Definition of configurable interface Pwm_Notification_<#Channel> ⌈

Service Name	Pwm_Notification_<#Channel>
Syntax	`void Pwm_Notification_<#Channel> ( void )`
Sync/Async	Synchronous
Reentrancy	PWM user implementation dependant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	The Pwm module shall call the function Pwm_Notification_<#Channel> accordingly to the last call of Pwm_EnableNotification for channel <#Channel>.
Available via	Pwm_Externals.h

⌋

[SWS_Pwm_00025]

Upstream requirements: SRS_SPAL_00157

⌈The Pwm module shall call the function `Pwm_Notification_<#Channel>` accordingly to the last call of `Pwm_EnableNotification` and `Pwm_DisableNotification` for channel <#Channel>.⌋

[SWS_Pwm_00026]

Upstream requirements: SRS_SPAL_12129

⌈The Pwm module shall reset the interrupt flag associated to the notification `Pwm_Notification_<#Channel>`⌋

[SWS_Pwm_10115] ⌈The Pwm module shall provide the functionality of `Pwm_EnableNotification` only when the configuration parameter `PwmNotificationSupported` is ON.⌋

[SWS_Pwm_20115] ⌈The Pwm module shall provide the functionality of `Pwm_DisableNotification` only when the configuration parameter `PwmNotificationSupported` is ON.⌋

[SWS_Pwm_30115] ⌈The Pwm module shall reset the interrupt flag associated to the notification only when the configuration parameter `PwmNotificationSupported` is ON.⌋

[SWS_Pwm_91005] Definition of configurable interface IoHwAb_Pwm_NotifyReadyForPowerState<#Mode> ⌈

Service Name	IoHwAb_Pwm_NotifyReadyForPowerState<#Mode>
Syntax	`void IoHwAb_Pwm_NotifyReadyForPowerState<#Mode> ( void )`
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	The API shall be invoked by the PWM Driver when the requested power state preparation for mode <#Mode> is completed.
Available via	IoHwAb_Pwm.h

⌋

[SWS_Pwm_00199] ⌈In case the PWM Driver is configured to support power state management with asynchronous transitions, this API shall be called to signal completion of the power transition preparation phase to the IoHwAbs module.

This is a callback, this API is to be implemented in the IoHwAbs component.⌋

8.7 Service Interfaces

This module does not provide any service interfaces.

8.8 API parameter checking

[SWS_Pwm_10051]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈If development error detection for the Pwm module is enabled, and a development error occurs, then the corresponding PWM function shall report the error to the Default Error Tracer.⌋

[SWS_Pwm_20051]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈If development error detection for the Pwm module is enabled, and a development error occurs, then the corresponding PWM function shall skip the desired functionality in order to avoid any corruptions of data or hardware registers leaving the function without any actions.⌋

[SWS_Pwm_00117]

Upstream requirements: SRS_BSW_00406, SRS_BSW_00323, SRS_BSW_00386

⌈If development error detection for the Pwm module is enabled: if any function (except `Pwm_Init`) is called before `Pwm_Init` has been called, the called function shall raise development error `PWM_E_UNINIT`.⌋

[SWS_Pwm_00045]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈If development error detection for the Pwm module is enabled: The API `Pwm_SetPeriodAndDuty` shall check if the given PWM channel is of the channel class type `PWM_VARIABLE_PERIOD`. If this is not the case the development error `PWM_E_PERIOD_UNCHANGEABLE` shall be called.⌋

[SWS_Pwm_00047]

Upstream requirements: SRS_BSW_00323, SRS_BSW_00386

⌈If development error detection for the Pwm module is enabled: the PWM functions shall check the parameter ChannelNumber and raise development error `PWM_E_PARAM_CHANNEL` if the parameter ChannelNumber is invalid.⌋

9 Sequence diagrams

9.1 Initialization

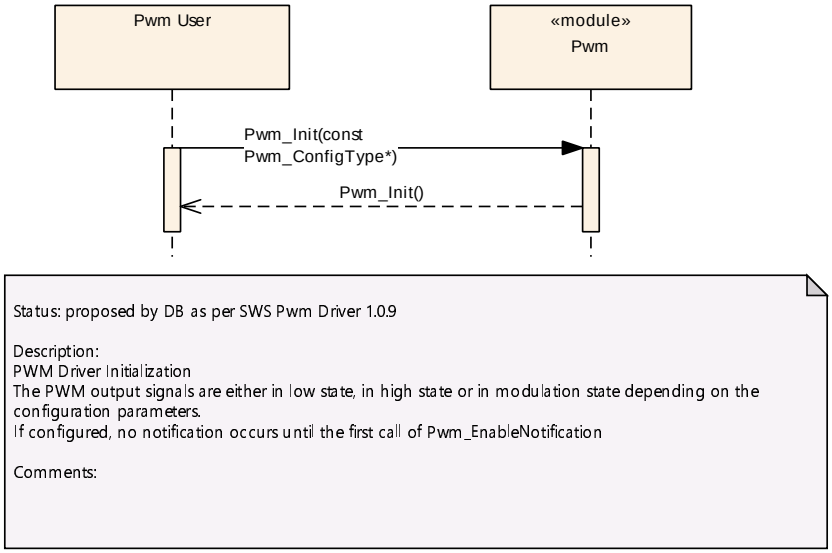


Figure 9.1: PWM initialization

9.2 De-initialization

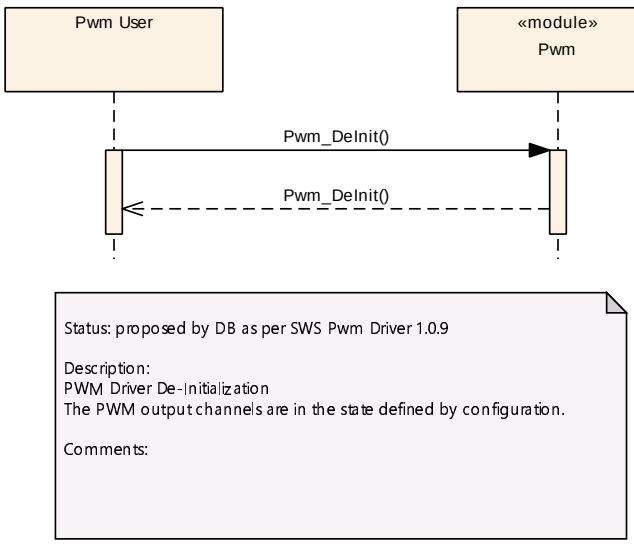


Figure 9.2: PWM de-initialization

9.3 Setting the duty cycle

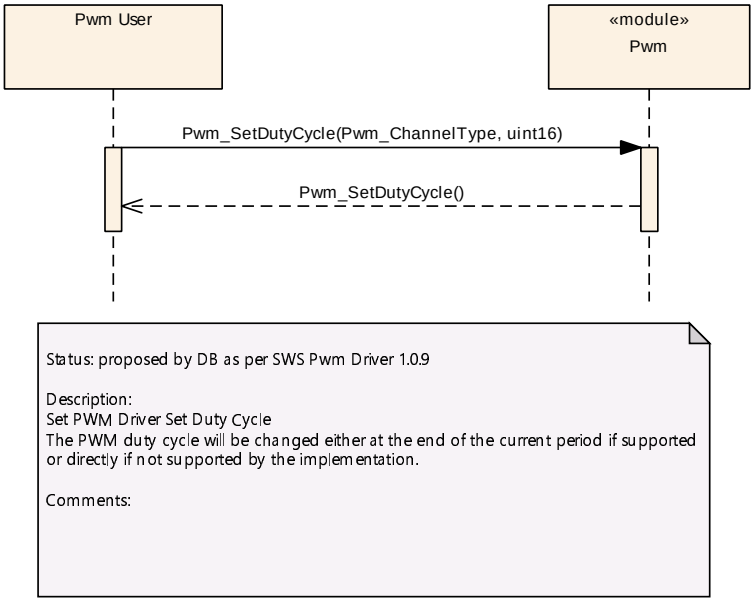


Figure 9.3: Setting the duty cycle

9.4 Setting the period and the duty

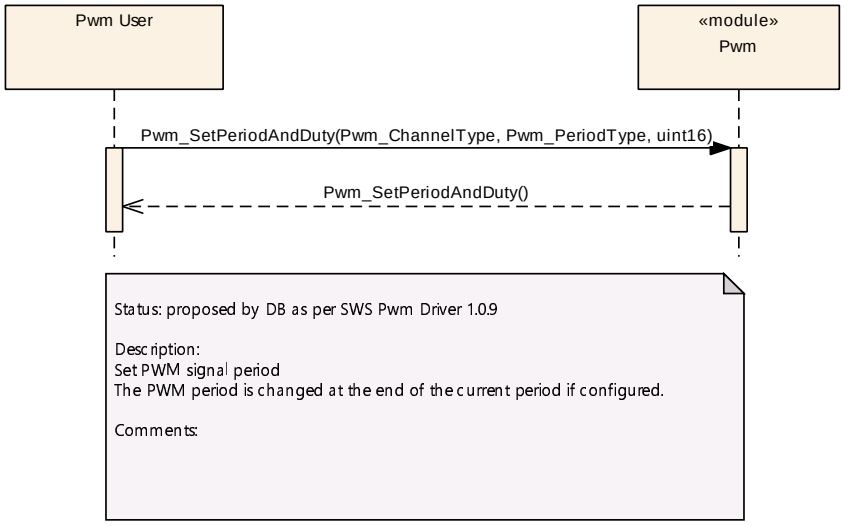


Figure 9.4: Setting the period and duty cycle

9.5 Setting the PWM output to idle

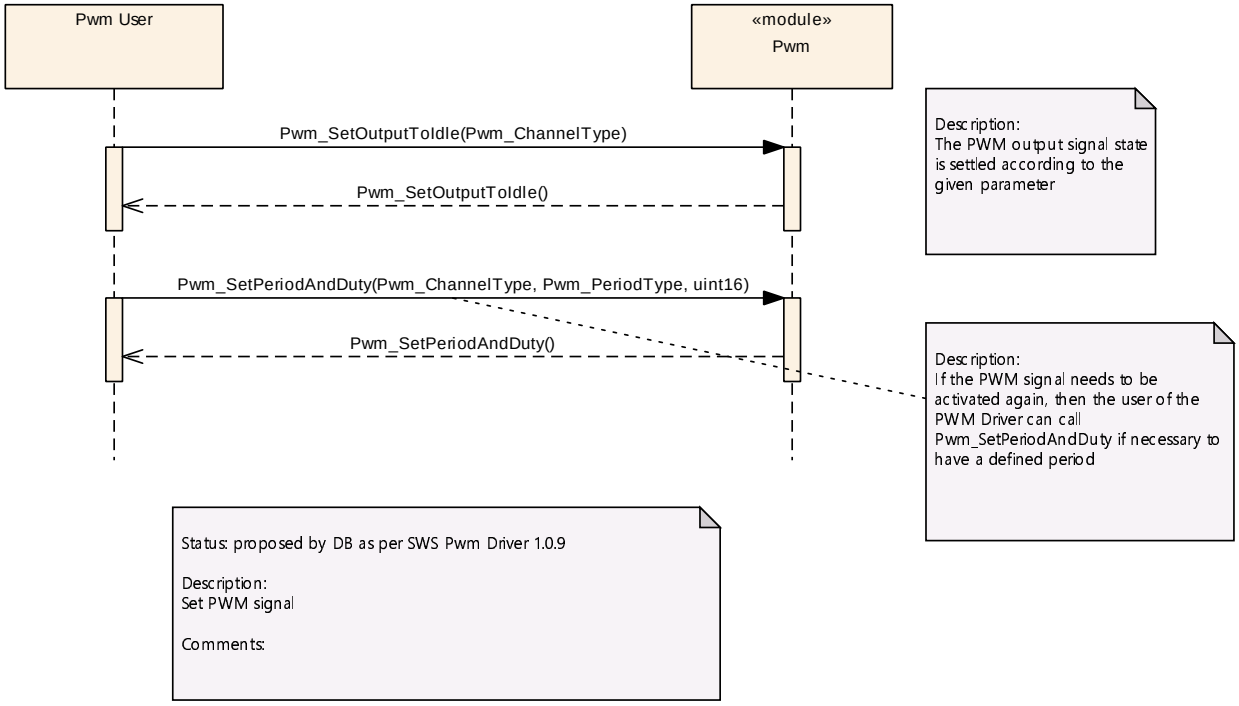


Figure 9.5: Setting Pwm output to idle

9.6 Getting the PWM Output state

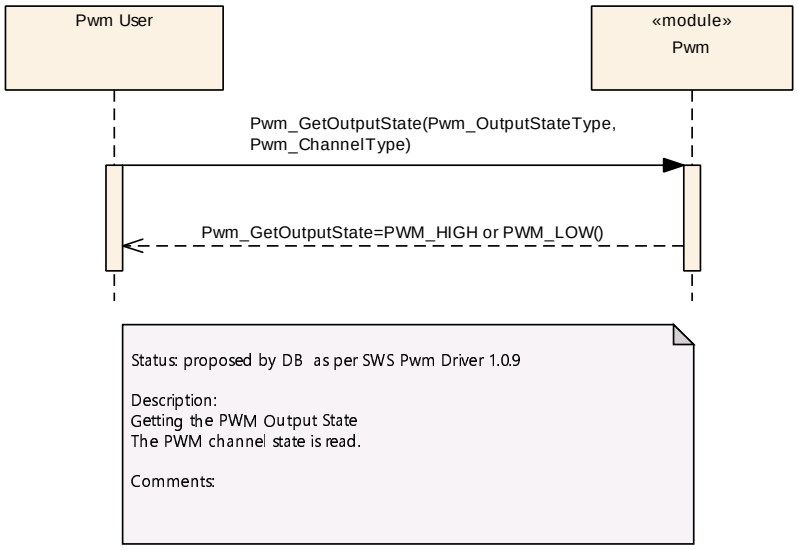


Figure 9.6: Getting Pwm output state

9.7 Using the PWM notifications

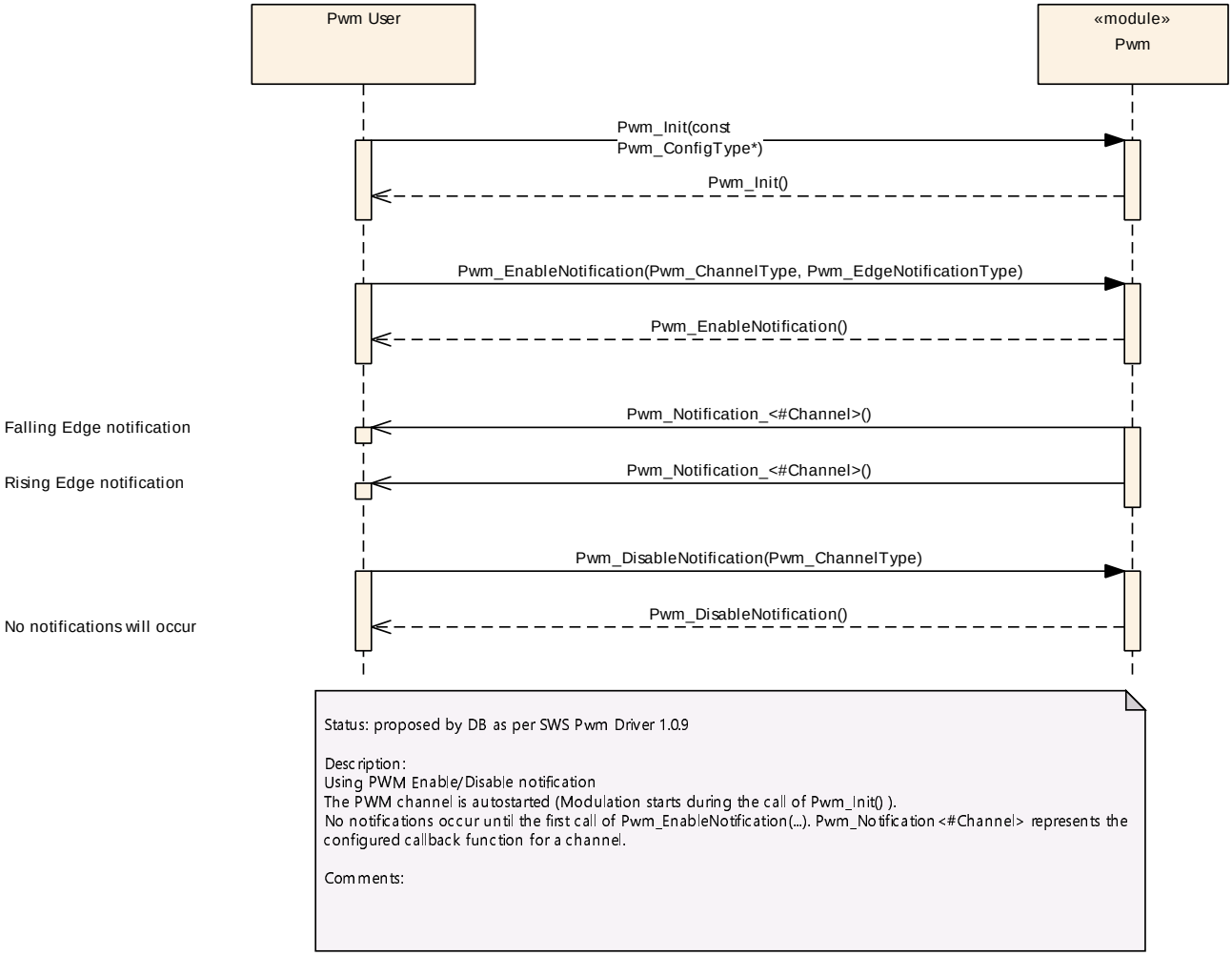


Figure 9.7: Using Pwm notifications

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 PWM Driver.

Chapter 10.3 specifies published information of the module PWM Driver.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in [2].

10.2 Containers and configuration parameters

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

[SWS_Pwm_00203] ⌈The PWM module shall reject configurations with partition mappings which are not supported by the implementation.⌋

10.2.1 Pwm

[ECUC_Pwm_00148] Definition of EcucModuleDef Pwm ⌈

Module Name	Pwm
Description	Configuration of Pwm (Pulse Width Modulation) module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
PwmChannelConfigSet	1	This container contains the configuration parameters and sub containers of the AUTOSAR Pwm module.
PwmConfigurationOfOptApi Services	1	–
PwmGeneral	1	–

⌋

10.2.2 PwmGeneral

[ECUC_Pwm_00004] Definition of EcucParamConfContainerDef PwmGeneral ⌈

Container Name	PwmGeneral
Parent Container	Pwm
Description	–
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
PwmDevErrorDetect	1	[ECUC_Pwm_00131]
PwmDutycycleUpdatedEndperiod	1	[ECUC_Pwm_00132]
PwmIndex	1	[ECUC_Pwm_00139]
PwmLowPowerStatesSupport	0..1	[ECUC_Pwm_00142]
PwmNotificationSupported	1	[ECUC_Pwm_00133]
PwmPeriodUpdatedEndperiod	1	[ECUC_Pwm_00134]
PwmPowerStateAsynchTransitionMode	0..1	[ECUC_Pwm_00143]
PwmEcucPartitionRef	0..*	[ECUC_Pwm_00149]
PwmKernelEcucPartitionRef	0..1	[ECUC_Pwm_00150]

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
PwmPowerStateConfig	0..*	Each instance of this parameter defines a power state and the callback to be called when this power state is reached.

⌋

[ECUC_Pwm_00131] Definition of EcucBooleanParamDef PwmDevErrorDetect ⌈

Parameter Name	PwmDevErrorDetect		
Parent Container	PwmGeneral		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00132] Definition of EcucBooleanParamDef PwmDutycycleUpdatedEndperiod ⌈

Parameter Name	PwmDutycycleUpdatedEndperiod		
Parent Container	PwmGeneral		
Description	Switch for enabling the update of the duty cycle parameter at the end of the current period. TRUE: update of duty cycle is done at the end of period of currently generated waveform (current waveform is finished). FALSE: update of duty cycle is done immediately (just after service call, current waveform is cut).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00139] Definition of EcucIntegerParamDef PwmIndex ⌈

Parameter Name	PwmIndex		
Parent Container	PwmGeneral		
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00142] Definition of EcucBooleanParamDef PwmLowPowerStatesSupport ⌈

Parameter Name	PwmLowPowerStatesSupport
Parent Container	PwmGeneral
Description	Adds / removes all power state management related APIs (PWM_SetPowerState, PWM_GetCurrentPowerState, PWM_GetTargetPowerState, PWM_PreparePowerState, PWM_Main_PowerTransitionManager), indicating if the HW offers low power state management.
Multiplicity	0..1
Type	EcucBooleanParamDef

▽

△

Default value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00133] Definition of EcucBooleanParamDef PwmNotificationSupported ⌈

Parameter Name	PwmNotificationSupported		
Parent Container	PwmGeneral		
Description	Switch to indicate that the notifications are supported		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00134] Definition of EcucBooleanParamDef PwmPeriodUpdatedEndperiod ⌈

Parameter Name	PwmPeriodUpdatedEndperiod		
Parent Container	PwmGeneral		
Description	Switch for enabling the update of the period parameter at the end of the current period. TRUE: update of period/duty cycle is done at the end of period of currently generated waveform (current waveform is finished). FALSE: update of period/duty cycle is done immediately (just after service call, current waveform is cut).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

▽

△

Scope / Dependency	scope: local

⌋

[ECUC_Pwm_00143] Definition of EcucBooleanParamDef PwmPowerStateAsynchTransitionMode ⌈

Parameter Name	PwmPowerStateAsynchTransitionMode		
Parent Container	PwmGeneral		
Description	Enables / disables support of the PWM Driver to the asynchronous power state transition.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local dependency: This parameter shall only be configured if the parameter PwmLowPowerStatesSupport is set to true.		

⌋

[ECUC_Pwm_00149] Definition of EcucReferenceDef PwmEcucPartitionRef ⌈

Parameter Name	PwmEcucPartitionRef		
Parent Container	PwmGeneral		
Description	Maps the PWM driver to zero or multiple ECUC partitions to make the driver API available in the according partition.		
Multiplicity	0..*		
Type	Reference to EcucPartition		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

⌋

[ECUC_Pwm_00150] Definition of EcucReferenceDef PwmKernelEcucPartition Ref ⌈

Parameter Name	PwmKernelEcucPartitionRef		
Parent Container	PwmGeneral		
Description	Maps the PWM kernel to zero or one ECUC partitions to assign the driver kernel to a certain core. The ECUC partition referenced is a subset of the ECUC partitions where the PWM driver is mapped to.		
Multiplicity	0..1		
Type	Reference to EcucPartition		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

⌋

[SWS_Pwm_CONSTR_00001] ⌈The ECUC partitions referenced by `PwmKernelEcucPartitionRef` shall be a subset of the ECUC partitions referenced by `PwmEcucPartitionRef`.⌋

[SWS_Pwm_CONSTR_00002] ⌈If `PwmEcucPartitionRef` references one or more ECUC partitions, `PwmKernelEcucPartitionRef` shall have a multiplicity of one and reference one of these ECUC partitions as well.⌋

10.2.3 PwmPowerStateConfig

[ECUC_Pwm_00144] Definition of EcucParamConfContainerDef PwmPowerState Config ⌈

Container Name	PwmPowerStateConfig
Parent Container	PwmGeneral
Description	Each instance of this parameter defines a power state and the callback to be called when this power state is reached.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
PwmPowerState	1	[ECUC_Pwm_00146]
PwmPowerStateReadyCbkRef	1	[ECUC_Pwm_00145]

No Included Containers

⌋

[ECUC_Pwm_00146] Definition of EcucIntegerParamDef PwmPowerState ⌈

Parameter Name	PwmPowerState		
Parent Container	PwmPowerStateConfig		
Description	Each instance of this parameter describes a different power state supported by the PWM HW. It should be defined by the HW supplier and used by the PWMDriver to reference specific HW configurations which set the PWM HW module in the referenced power state. At least the power mode corresponding to full power state shall be always configured.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 18446744073709551615		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local dependency: This parameter shall only be configured if the parameter PwmLowPower StatesSupport is set to true.		

⌋

[ECUC_Pwm_00145] Definition of EcucFunctionNameDef PwmPowerStateReady CbkRef ⌈

Parameter Name	PwmPowerStateReadyCbkRef		
Parent Container	PwmPowerStateConfig		
Description	Each instance of this parameter contains a reference to a power mode callback defined in a CDD or IoHwAbs component.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local dependency: This parameter shall only be configured if the parameter PwmLowPower StatesSupport is set to true.		

⌋

10.2.4 PwmChannel

[ECUC_Pwm_00027] Definition of EcucParamConfContainerDef PwmChannel ⌈

Container Name	PwmChannel
Parent Container	PwmChannelConfigSet
Description	Configuration of an individual PWM channel.
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
PwmChannelClass	0..1	[ECUC_Pwm_00136]
PwmChannelId	1	[ECUC_Pwm_00137]
PwmDutycycleDefault	1	[ECUC_Pwm_00138]
PwmIdleState	1	[ECUC_Pwm_00122]
PwmNotification	0..1	[ECUC_Pwm_00123]
PwmPeriodDefault	1	[ECUC_Pwm_00124]
PwmPolarity	1	[ECUC_Pwm_00125]
PwmChannelEcucPartitionRef	0..*	[ECUC_Pwm_00151]
PwmMcuClockReferencePoint	1	[ECUC_Pwm_00147]

No Included Containers

⌋

[ECUC_Pwm_00136] Definition of EcucEnumerationParamDef PwmChannelClass ⌈

Parameter Name	PwmChannelClass		
Parent Container	PwmChannel		
Description	Class of PWM Channel. ImplementationType: Pwm_ChannelClassType		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	PWM_FIXED_PERIOD	Only the duty cycle can be changed.	
	PWM_FIXED_PERIOD_SHIFTED	Only the duty cycle can be changed. The period is shifted (only if supported by hardware)	
	PWM_VARIABLE_PERIOD	Duty Cycle and period can be changed.	
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD

▽

△

Scope / Dependency	scope: local

⌋

[ECUC_Pwm_00137] Definition of EcucIntegerParamDef PwmChannelId ⌈

Parameter Name	PwmChannelId		
Parent Container	PwmChannel		
Description	Channel Id of the PWM channel. This value will be assigned to the symbolic name derived of the PwmChannel container short name.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00138] Definition of EcucIntegerParamDef PwmDutycycleDefault ⌈

Parameter Name	PwmDutycycleDefault		
Parent Container	PwmChannel		
Description	Value of duty cycle used for Initialization 0, represents 0% 0x8000 represents 100%		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 32768		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00122] Definition of EcucEnumerationParamDef PwmIdleState ⌈

Parameter Name	PwmIdleState
Parent Container	PwmChannel
Description	The parameter PWM_IDLE_STATE represents the output state of the PWM after the signal is stopped (e.g. call of Pwm_SetOutputToIdle).
Multiplicity	1

▽

△

Type	EcucEnumerationParamDef		
Range	PWM_HIGH	The PWM channel output will be set to high (3 or 5 V) in idle state.	
	PWM_LOW	The PWM channel output will be set to low (0 V) in idle state.	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00123] Definition of EcucFunctionNameDef PwmNotification ⌈

Parameter Name	PwmNotification		
Parent Container	PwmChannel		
Description	Definition of the Callback function.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	"NULL"		
Regular Expression	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00124] Definition of EcucFloatParamDef PwmPeriodDefault ⌈

Parameter Name	PwmPeriodDefault		
Parent Container	PwmChannel		
Description	Value of period used for Initialization.(in seconds).		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD

▽

△

Scope / Dependency	scope: local

⌋

[ECUC_Pwm_00125] Definition of EcucEnumerationParamDef PwmPolarity ⌈

Parameter Name	PwmPolarity		
Parent Container	PwmChannel		
Description	Defines the starting polarity of each PWM channel.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	PWM_HIGH	The PWM channel output is high at the beginning of the cycle and then goes low when the duty count is reached.	
	PWM_LOW	The PWM channel output is low at the beginning of the cycle and then goes high when the duty count is reached.	
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00151] Definition of EcucReferenceDef PwmChannelEcucPartitionRef ⌈

Parameter Name	PwmChannelEcucPartitionRef		
Parent Container	PwmChannel		
Description	Maps a PWM channel to zero or multiple ECUC partitions to limit the access to this channe. The ECUC partitions referenced are a subset of the ECUC partitions where the PWM driver is mapped to.		
Multiplicity	0..*		
Type	Reference to EcucPartition		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

⌋

[ECUC_Pwm_00147] Definition of EcucReferenceDef PwmMcuClockReference Point ⌈

Parameter Name	PwmMcuClockReferencePoint		
Parent Container	PwmChannel		
Description	This parameter contains reference to the McuClockReferencePoint		
Multiplicity	1		
Type	Reference to McuClockReferencePoint		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	–	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

⌋

[SWS_Pwm_CONSTR_00003] ⌈If `PwmEcucPartitionRef` references one or more ECUC partitions, `PwmChannelEcucPartitionRef` shall have a multiplicity of greater than zero and reference one or several of these ECUC partitions as well.⌋

10.2.5 PwmChannelConfigSet

[ECUC_Pwm_00140] Definition of EcucParamConfContainerDef PwmChannel ConfigSet ⌈

Container Name	PwmChannelConfigSet
Parent Container	Pwm
Description	This container contains the configuration parameters and sub containers of the AUTOSAR Pwm module.
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
PwmChannel	1..*	Configuration of an individual PWM channel.

⌋

10.2.6 PwmConfigurationOfOptApiServices

[ECUC_Pwm_00126] Definition of EcucParamConfContainerDef PwmConfigurationOfOptApiServices ⌈

Container Name	PwmConfigurationOfOptApiServices
Parent Container	Pwm
Description	–
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
PwmDeInitApi	1	[ECUC_Pwm_00141]
PwmGetOutputState	1	[ECUC_Pwm_00127]
PwmSetDutyCycle	1	[ECUC_Pwm_00128]
PwmSetOutputToIdle	1	[ECUC_Pwm_00129]
PwmSetPeriodAndDuty	1	[ECUC_Pwm_00130]
PwmVersionInfoApi	1	[ECUC_Pwm_00135]

No Included Containers

⌋

[ECUC_Pwm_00141] Definition of EcucBooleanParamDef PwmDeInitApi ⌈

Parameter Name	PwmDeInitApi		
Parent Container	PwmConfigurationOfOptApiServices		
Description	Adds / removes the service Pwm_DeInit() from the code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00127] Definition of EcucBooleanParamDef PwmGetOutputState ⌈

Parameter Name	PwmGetOutputState		
Parent Container	PwmConfigurationOfOptApiServices		
Description	–		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	

▽

△

Scope / Dependency	scope: local

⌋

[ECUC_Pwm_00128] Definition of EcucBooleanParamDef PwmSetDutyCycle ⌈

Parameter Name	PwmSetDutyCycle		
Parent Container	PwmConfigurationOfOptApiServices		
Description	–		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00129] Definition of EcucBooleanParamDef PwmSetOutputToIdle ⌈

Parameter Name	PwmSetOutputToIdle		
Parent Container	PwmConfigurationOfOptApiServices		
Description	–		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00130] Definition of EcucBooleanParamDef PwmSetPeriodAndDuty ⌈

Parameter Name	PwmSetPeriodAndDuty
Parent Container	PwmConfigurationOfOptApiServices
Description	–
Multiplicity	1
Type	EcucBooleanParamDef

▽

△

Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_Pwm_00135] Definition of EcucBooleanParamDef PwmVersionInfoApi ⌈

Parameter Name	PwmVersionInfoApi		
Parent Container	PwmConfigurationOfOptApiServices		
Description	Switch to indicate that the Pwm_ GetVersionInfo is supported		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in [2].

A Not applicable requirements

[SWS_Pwm_NA_00153]

Upstream requirements: SRS_BSW_00159, SRS_BSW_00167, SRS_BSW_00170, SRS_BSW_00419, SRS_BSW_00383, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00417, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00415, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00342, SRS_BSW_00160, SRS_BSW_00007, SRS_BSW_00300, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00305, SRS_BSW_00307, SRS_BSW_00310, SRS_BSW_00373, SRS_BSW_00327, SRS_BSW_00335, SRS_BSW_00350, SRS_BSW_00408, SRS_BSW_00410, SRS_BSW_00348, SRS_BSW_00353, SRS_BSW_00301, SRS_BSW_00302, SRS_BSW_00328, SRS_BSW_00312, SRS_BSW_00006, SRS_BSW_00357, SRS_BSW_00377, SRS_BSW_00304, SRS_BSW_00378, SRS_BSW_00306, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00358, SRS_BSW_00414, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00009, SRS_BSW_00401, SRS_BSW_00172, SRS_BSW_00010, SRS_BSW_00333, SRS_BSW_00003, SRS_BSW_00341, SRS_SPAL_12267, SRS_SPAL_12461, SRS_SPAL_12462, SRS_SPAL_12463, SRS_SPAL_12068, SRS_SPAL_12069, SRS_SPAL_12169, SRS_SPAL_12075, SRS_SPAL_12064, SRS_SPAL_12067, SRS_SPAL_12077, SRS_SPAL_12078, SRS_SPAL_12092, SRS_SPAL_12265, SRS_Pwm_12379

⌈These requirements are not applicable to this specification.⌋