Document Title	Specification of PWM Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	037
Document Classification	Standard

Document Version	2.3.1
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change Description
28.02.2014	2.3.1	AUTOSAR Release Management	• Editorial changes • Removed chapter(s) on change documentation
27.04.2011	2.3.0	AUTOSAR Administration	• Updated and improved descriptions • PWM_FIXED_PERIOD_SHIFTED is just optional (no more mandatory) • Legal disclaimer revised
23.06.2008	2.2.1	AUTOSAR Administration	Legal disclaimer revised
20.12.2007	2.2.0	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
30.01.2007	2.1.0	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

Document Change History

Date	Version	Changed by	Change Description
25.04.2006	2.0.0	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
23.06.2005	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

Table of 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.

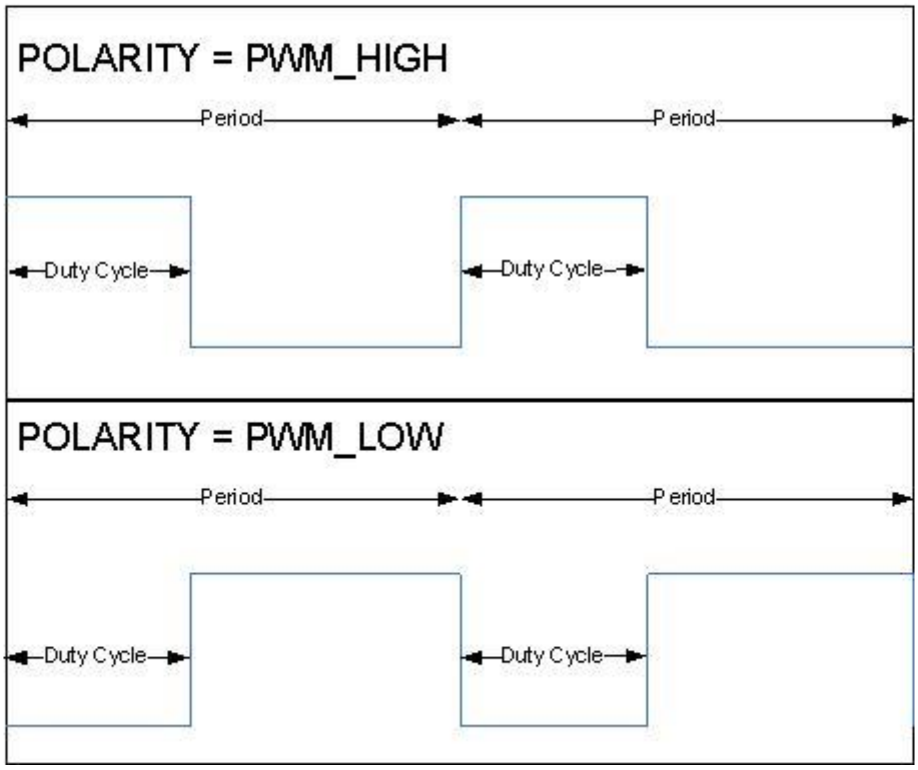


Figure 1: PWM signal description

2 Acronyms and abbreviations

Acronyms and abbreviations that have a local scope are not contained in the AUTOSAR glossary. These must appear in a local glossary.

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.

Abbreviation:	***Description:***
PWM	Pulse Width Modulation.
DEM	Diagnostic Event Manager.
DET	Development Error Tracer.
MCU	Microcontroller Unit.
PLL	Phase Locked Loop.
ISR	Interrupt Service Routine.

3 Related documentation

3.1 Input documents

[1] Layered Software Architecture
AUTOSAR_LayeredSoftwareArchitecture.pdf

[2] General Requirements on SPAL
AUTOSAR_SRS_SPAL_General.pdf

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

[4] Specification of Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[5] Specification of MCU Driver
AUTOSAR_SWS_MCU_Driver.pdf

[6] Specification of ECU Configuration,
AUTOSAR_ECU_Configuration.pdf

[7] AUTOSAR Basic Software Module Description Template,
AUTOSAR_BSW_Module_Description.pdf

4 Constraints and assumptions

4.1 Limitations

PWM001 The Pwm SWS does not cover PWM emulation on general purpose I/O.

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 → PLL off) also affect the clock settings of the PWM hardware.

The PWM Driver depends on the following modules:
- PORT Driver: To set the port pin functionality.
- MCU Driver: To set prescaler, system clock and PLL.
- DET: Development Error Tracer in Development mode.

5.1 File structure

5.1.1 Code file structure

PWM065: The Pwm SWS shall not define the code file structure.

5.1.2 Header file structure

PWM075: The Pwm module shall adhere to the following include file structure:

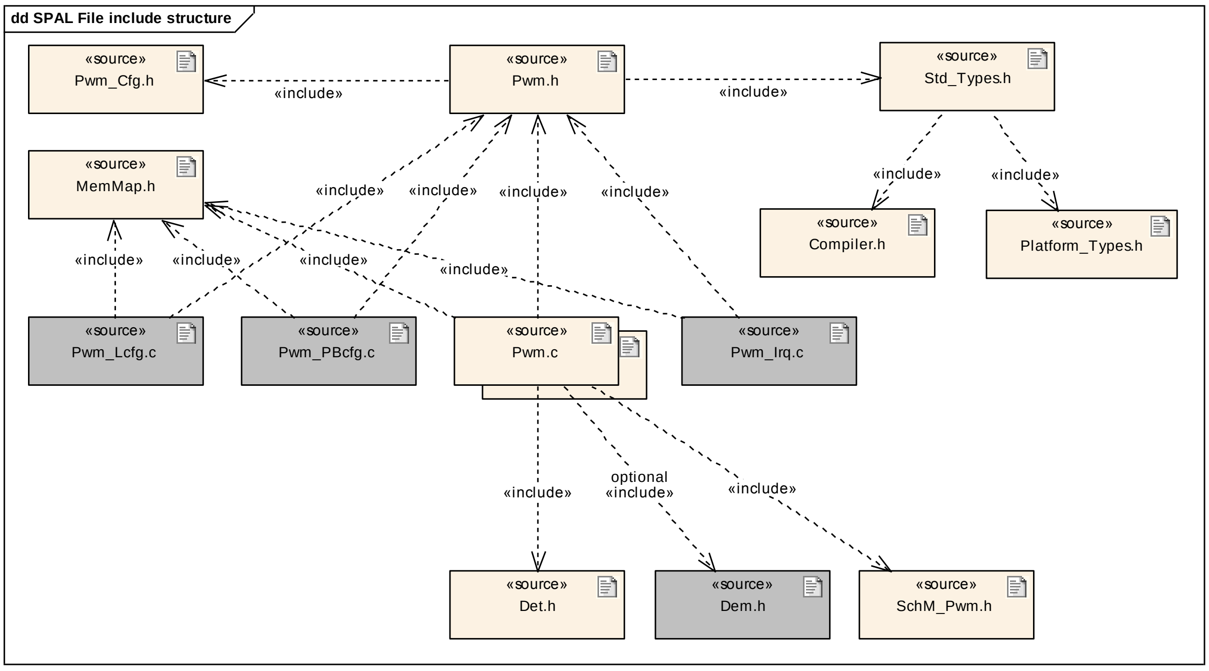


Figure 2: Header file structure

PWM066: The Pwm module shall optionally include the Dem.h file if any production error will be issued by the implementation.

By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns

ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

6 Requirements traceability

Document: General Requirements on Basic Software Modules.

Requirement	Satisfied by
[BSW00344] Reference to link-time configuration	PWM027
[BSW00404] Reference to post build time configuration	PWM027
[BSW00405] Reference to multiple configuration sets	PWM027
[BSW00345] Configuration at Compile time	PWM004
[BSW159] Tool-based configuration	Not applicable (Both static and runtime configuration parameters are located outside the source code of the module. This is the prerequisite for automatic configuration.)
[BSW167] Static configuration checking	Not Applicable (requirement on configuration tool)
[BSW171] Configurability of optional functionality	PWM004 PWM080 PWM082 PWM083 PWM084 PWM085
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable (no reconfiguration and not a SWC)
[BSW00380] Separate C-File for configuration parameters	PWM065
[BSW00419] Separate C-Files for pre-compile time configuration parameters	Not applicable (Implementation specific, the code file structure is not defined within this specification and is left up to the implementer)
[BSW00381] Separate configuration header file for pre-compile time parameters	PWM075
[BSW00412] Separate H-File for configuration parameters	PWM075
[BSW00383] List dependencies of configuration files	Not applicable (Requirement to be taken into account during implementation)
[BSW00384] List dependencies to other modules	CHECK WITH OTHER SWS!
[BSW00387] Specify the configuration class of callback function	PWM027 parameter PwmNotification
[BSW00388] Introduce containers	PWM004 PWM027
[BSW00389] Containers shall have names	PWM004 PWM027
[BSW00390] Parameter content shall be unique within the module	PWM004 PWM027
[BSW00391] Parameter shall have unique names	PWM004 PWM027
[BSW00392] Parameters shall have a type	PWM004 PWM027
[BSW00393] Parameters shall have a range	PWM004 PWM027
[BSW00394] Specify the scope of the parameters	PWM004 PWM027
[BSW00395] List the required parameters (per parameter)	PWM004 PWM027
[BSW00396] Configuration classes	PWM004 PWM027
[BSW00397] Pre-compile-time parameters	PWM004 PWM027
[BSW00398] Link-time parameters	PWM004 PWM027
[BSW00399] Loadable Post-build time parameters	PWM004 PWM027
[BSW00400] Selectable Post-build time parameters	PWM004 PWM027
[BSW00402] Published information	PWM054
[BSW00375] Notification of wake-up reason	Not applicable (No wakeup functionality in this BSW)
[BSW101] Initialization interface	PWM007
[BSW00416] Sequence of Initialization	Not Applicable

Requirement	***Satisfied by***
	(SW Integration requirement)
[BSW00406] Check module initialization	PWM117
[BSW168] Diagnostic Interface of SW components	Not applicable (Not a SWC)
[BSW00407] Function to read out published parameters	PWM068 PWM069
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (Module is part of MCAL)
[BSW00424] BSW main processing function task allocation	Not applicable (No Main function in this module and requirement for software integration)
[BSW00425] Trigger conditions for schedulable objects	Not applicable (Requirement to be taken into account during implementation and integration)
[BSW00426] Exclusive areas in BSW modules	Not applicable (Requirement to be taken into account during implementation and integration)
[BSW00427] ISR description for BSW modules	Not applicable (Requirement to be taken into account during implementation and integration)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (Requirement to be taken into account during implementation and integration)
[BSW00429] Restricted BSW OS functionality access	Not applicable (Requirement to be taken into account during implementation and integration)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (Requirement to be taken into account during implementation and integration)
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (Requirement to be taken into account during implementation and integration)
[BSW00433] Calling of main processing functions	Not applicable (Requirement to be taken into account during implementation and integration)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (Requirement to be taken into account during implementation and integration)
[BSW00435] Module Header File Structure for the Basic Software Scheduler	PWM075
[BSW00436] Module Header File Structure for the Memory Mapping	PWM075
[BSW00336] Shutdown interface	PWM010
[BSW00337] Classification of errors	PWM002
[BSW00338] Detection and Reporting of development errors	PWM003
[BSW00369] Do not return development error codes via API	PWM003
[BSW00339] Reporting of production relevant error status	PWM005 PWM006 PWM066
[BSW00421] Reporting of production relevant error events	PWM005 PWM006 PWM066
[BSW00422] Debouncing of production relevant error status	PWM005 PWM006
[BSW00420] Production relevant error event rate detection	PWM005 PWM006

Requirement	***Satisfied by***
[BSW00417] Reporting of Error Events by Non-Basic Software	Not Applicable (Module is a BSW)
[BSW00323] API parameter checking	PWM117 PWM045 PWM046 PWM047 PWM051
[BSW004] Version check	PWM029
[BSW00409] Header files for production code error IDs	PWM066
[BSW00385] List possible error notifications	PWM002
[BSW00386] Configuration for detecting an error	PWM051 PWM117 PWM045 PWM046 PWM047 PWM003 PWM005 PWM006 PWM064 PWM002
[BSW161] Microcontroller abstraction	Not Applicable (Requirement on software architecture, not for a single module)
[BSW162] ECU layout abstraction	Not Applicable (Requirement on software architecture, not for a single module)
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (Requirement to be taken into account during implementation)
[BSW00415] User dependent include files	Not applicable (Requirement to be taken into account during implementation)
[BSW164] Implementation of interrupt service routines	Not applicable (Requirement to be taken into account during implementation)
[BSW00325] Runtime of interrupt service routines	Not applicable (Requirement to be taken into account during implementation)
[BSW00326] Transition from ISRs to OS tasks	Not applicable (Requirement to be taken into account during implementation/Integration)
[BSW00342] Usage of source code and object code	Not applicable (Requirement to be taken into account during implementation)
[BSW00343] Specification and configuration of time	PWM070
[BSW160] Human-readable configuration data	Not applicable (Requirement to be taken into account during implementation)
[BSW007] HIS MISRA C	Not applicable (Requirement to be taken into account during implementation)
[BSW00300] Module naming convention	Not applicable (Requirement to be taken into account during implementation)
[BSW00413] Accessing instances of BSW modules	Not applicable (Requirement to be taken into account during implementation)
[BSW00347] Naming separation of different instances of BSW drivers	Not applicable (Requirement to be taken into account during implementation)
[BSW00305] Self-defined data types naming convention	Not applicable (Requirement to be taken into account during implementation)
[BSW00307] Global variables naming convention	Not applicable (Requirement to be taken into account during implementation)

Requirement	***Satisfied by***
[BSW00310] API naming convention	Not applicable (Requirement to be taken into account during implementation)
[BSW00373] Main processing function naming convention	Not applicable (Requirement to be taken into account during implementation)
[BSW00327] Error values naming convention	Not applicable (Requirement to be taken into account during implementation)
[BSW00335] Status values naming convention	Not applicable (Requirement to be taken into account during implementation)
[BSW00350] Development error detection keyword	Not applicable (Requirement to be taken into account during implementation)
[BSW00408] Configuration parameter naming convention	Not applicable (Requirement to be taken into account during implementation)
[BSW00410] Compiler switches shall have defined values	Not applicable (Requirement to be taken into account during implementation)
[BSW00411] Get version info keyword	PWM004
[BSW00346] Basic set of module files	PWM065
[BSW158] Separation of configuration from implementation	PWM065
[BSW00314] Separation of interrupt frames and service routines	PWM065
[BSW00370] Separation of callback interface from API	PWM065
[BSW00348] Standard type header	Not applicable (Requirement to be taken into account during implementation)
[BSW00353] Platform specific type header	Not applicable (Requirement to be taken into account during implementation)
[BSW00361] Compiler specific language extension header	Not applicable (Requirement to be taken into account during implementation)
[BSW00301] Limit imported information	Not applicable (Requirement to be taken into account during implementation)
[BSW00302] Limit exported information	Not applicable (Requirement to be taken into account during implementation)
[BSW00328] Avoid duplication of code	Not applicable (Requirement to be taken into account during implementation)
[BSW00312] Shared code shall be reentrant	Not applicable (Requirement to be taken into account during implementation)
[BSW006] Platform independency	Not applicable (Requirement to be taken into account during implementation)
[BSW00357] Standard API return type	Not applicable (Requirement to be taken into account during implementation)
[BSW00377] Module specific API return types	Not applicable (Requirement to be taken into account during implementation)
[BSW00304] AUTOSAR integer data types	Not applicable

Requirement	***Satisfied by***
	(Requirement to be taken into account during implementation)
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable (Requirement to be taken into account during implementation)
[BSW00378] AUTOSAR boolean type	Not applicable (Requirement to be taken into account during implementation)
[BSW00306] Avoid direct use of compiler and platform specific keywords	Not applicable (Requirement to be taken into account during implementation)
[BSW00308] Definition of global data	Not applicable (Requirement to be taken into account during implementation)
[BSW00309] Global data with read-only constraint	Not applicable (Requirement to be taken into account during implementation)
[BSW00371] Do not pass function pointers via API	Not applicable (Requirement to be taken into account during implementation)
[BSW00358] Return type of `init()` functions	Not applicable (Requirement to be taken into account during implementation)
[BSW00414] Parameter of init function	Not applicable (Requirement to be taken into account during implementation)
[BSW00376] Return type and parameters of main processing functions	Not Applicable: (No Main Function)
[BSW00359] Return type of callback functions	Not applicable (Requirement to be taken into account during implementation)
[BSW00360] Parameters of callback functions	Not applicable (Requirement to be taken into account during implementation)
[BSW00329] Avoidance of generic interfaces	Not Applicable (Requirement on software architecture, not for a single module)
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable (Requirement to be taken into account during implementation)
[BSW00331] Separation of error and status values	Not applicable (Requirement to be taken into account during implementation)
[BSW009] Module User Documentation	Not applicable (Requirement to be taken into account during implementation)
[BSW00401] Documentation of multiple instances of configuration parameters	Not applicable (Requirement to be taken into account during implementation)
[BSW172] Compatibility and documentation of scheduling strategy	Not applicable (Requirement to be taken into account during implementation)
[BSW010] Memory resource documentation	Not applicable (Requirement to be taken into account during implementation)
[BSW00333] Documentation of callback function context	Not applicable (Requirement to be taken into account during implementation)

Requirement	***Satisfied by***
[BSW00374] Module vendor identification	PWM054
[BSW00379] Module identification	PWM054
[BSW003] Version identification	Not applicable (Requirement to be taken into account during implementation)
[BSW00318] Format of module version numbers	PWM054
[BSW00321] Enumeration of module version numbers	PWM054
[BSW00341] Microcontroller compatibility documentation	Not applicable (Requirement to be taken into account during implementation)
[BSW00334] Provision of XML file	Not applicable (Requirement to be taken into account during implementation)

Document: General Requirements on SPAL.

Requirements	***Satisfied by***
[BSW12263] Object code compatible configuration concept	PWM027
[BSW12056] Configuration of notification mechanisms	PWM027
[BSW12267] Configuration of wake-up sources	Not applicable (No wakeup functionality in this BSW)
[BSW12057] Driver module initialization	PWM007 PWM062 PWM009 PWM052
[BSW12125] Initialization of hardware resources	PWM062
[BSW12163] Driver module deinitialization	PWM010 PWM011 PWM012
[BSW12461] Responsibility for register initialization	Not applicable (Requirement to be taken into account during implementation)
[BSW12462] Provide settings for register initialization	Not applicable (Requirement to be taken into account during implementation)
[BSW12463] Combine and forward settings for register initialization	Not Applicable (Requirement on configuration tool)
[BSW12068] MCAL initialization sequence	Not applicable (this is a general software integration requirement)
[BSW12069] Wake-up notification of ECU State Manager	Not Applicable (No wakeup functionality in this BSW)
[BSW157] Notification mechanisms of drivers and handlers	PWM025 PWM025
[BSW12169] Control of operation mode	Not Applicable (No mode used)
[BSW12063] Raw value mode	Conflicts with BSW12459
[BSW12075] Use of application buffers	Not Applicable (No buffers used)
[BSW12129] Resetting of interrupt flags	PWM026 PWM026
[BSW12064] Change of operation mode during running operation	Not Applicable (No mode used)
[BSW12448] Behavior after development error detection	PWM051
[BSW12067] Setting of wake-up conditions	Not Applicable (No wakeup functionality)
[BSW12077] Non-blocking implementation	Not applicable (Requirement to be taken into account during implementation)
[BSW12078] Runtime and memory efficiency	Not applicable (Requirement to be taken into

Requirements	***Satisfied by***
	account during implementation)
[BSW12092] Access to drivers	Not applicable (this is a driver)
[BSW12265] Configuration data shall be kept constant	Not applicable (Requirement to be taken into account during implementation)
[BSW12264]Specification of configuration items	PWM004 PWM027
Requirements (module specific)	***Satisfied by***
[BSW12459] PWM duty cycle scaling	PWM059
[BSW12383] Resolution of duty cycle	PWM058
[BSW12375] PWM global configuration	PWM004
[BSW12293] Configuration of PWM channel properties	PWM061
[BSW12378] Assign notification to edges	PWM023 PWM024 PWM061
[BSW12379] Frequency of PWM channel groups	Not applicable (Requirement to be taken into account during implementation)
[BSW12389] Frequency of PWM channels	PWM041
[BSW12380] Initialization of PWM driver	PWM009
[BSW12381] De-Initialization of PWM driver	PWM010
[BSW12295] Set PWM duty cycle	PWM013
[BSW12382] Update of PWM duty cycle	PWM017
[BSW12358] Set PWM output to idle level	PWM021
[BSW12385] Get current state of PWM Channel	PWM022
[BSW12297] Set PWM period	PWM019 Pwm_SetPeriodAndDuty
[BSW12299] Activation of PWM edge notification	PWM023 PWM024 PWM025
[BSW12386] No PWM emulation	PWM001

7 Functional specification

7.1 General behavior

PWM088: 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 PWM088 must be performed by the module.

PWM089: 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

PWM070: All time units used within the API services of the PWM module shall be of the unit ticks.

7.3 Error classification

PWM002: Development error values are of type uint8.

Type or error	*Relevance*	*Related error code*	*Value [hex]*
API Pwm_Init service called with wrong parameter	Development	`PWM_E_PARAM_CONFIG`	`0x10`
API service used without module initialization	Development	`PWM_E_UNINIT`	`0x11`
API service used with an invalid channel Identifier	Development	`PWM_E_PARAM_CHANNEL`	`0x12`
Usage of unauthorized PWM service on PWM channel configured a fixed period	Development	`PWM_E_PERIOD_UNCHANGEABLE`	`0x13`
API Pwm_Init service called while the PWM driver has already been initialised	Development	`PWM_E_ALREADY_INITIALIZED`	`0x14`
--	Production	`--`	`Assigned externally`

To get more details concerning error detection, refer to chapter API parameter checking.

7.4 Error Detection

PWM003: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch `PwmDevErorDetect` shall activate or deactivate the detection of all development errors.

PWM064: `If the PwmDevErorDetect` switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter Error classification and chapter API specification.

PWM067: The detection of production code errors cannot be switched off.

PWM006: Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the PWM device specific implementation specification. The classification and enumeration shall be compatible to the errors listed above (refer to PWM002).

7.5 Error Notification

PWM078: Detected development errors shall be reported to the *`Det_ReportError`* service of the Development Error Tracer (DET) if the pre-processor switch *`PwmDevErorDetect`* is set.

PWM005: Production errors shall be reported to Diagnostic Event Manager.

7.6 Duty Cycle Resolution and scaling

PWM058: The width of the duty cycle parameter is 16 Bits.

PWM059: 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;
```

7.7 Version check

PWM029: The PWM C-files shall perform a preprocessor check of the versions of its header files to ensure the files are consistent and compatible between themselves.

8 API specification

8.1 Imported types

This chapter lists all types included from other modules.

PWM094:

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

8.2 Type definitions

8.2.1 Pwm_ChannelType

PWM106:

Name:	`Pwm_ChannelType`		
Type:	`Unsigned Integer`		
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.		

8.2.2 Pwm_PeriodType

PWM107:

Name:	`Pwm_PeriodType`		
Type:	`Unsigned Integer`		
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.		

8.2.3 Pwm_OutputStateType

PWM108:

Name:	`Pwm_OutputStateType`	
Type:	`Enumeration`	
Range:	`PWM_HIGH`	The PWM channel is in high state.
	`PWM_LOW`	The PWM channel is in low state.
Description:	Output state of a PWM channel.	

8.2.4 Pwm_EdgeNotificationType

PWM109:

Name:	`Pwm_EdgeNotificationType`	
Type:	`Enumeration`	
Range:	`PWM_RISING_EDGE`	Notification will be called when a rising edge occurs on the PWM output signal.
	`PWM_FALLING_EDGE`	Notification will be called when a falling edge occurs on the PWM output signal.
	`PWM_BOTH_EDGES`	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.	

8.2.5 Pwm_ChannelClassType

PWM110:

Name:	`Pwm_ChannelClassType`	
Type:	`Enumeration`	
Range:	`PWM_VARIABLE_PERIOD`	The PWM channel has a variable period. The duty cycle and the period can be changed.
	`PWM_FIXED_PERIOD`	The PWM channel has a fixed period. Only the duty cycle can be changed.
	`PWM_FIXED_PERIOD_SHIFTED`	Optional, if supported by the hardware. 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	

For a better understanding of the option PWM_FIXED_PERIOD_SHIFTED please see Figure 3. The start point of each PWM channel is shifted to the next channel by a certain time. This shift is constant, t_a is equal to t_b in Figure 3.
Use cases for this feature are e.g. motor control applications or to prevent EMC issues.

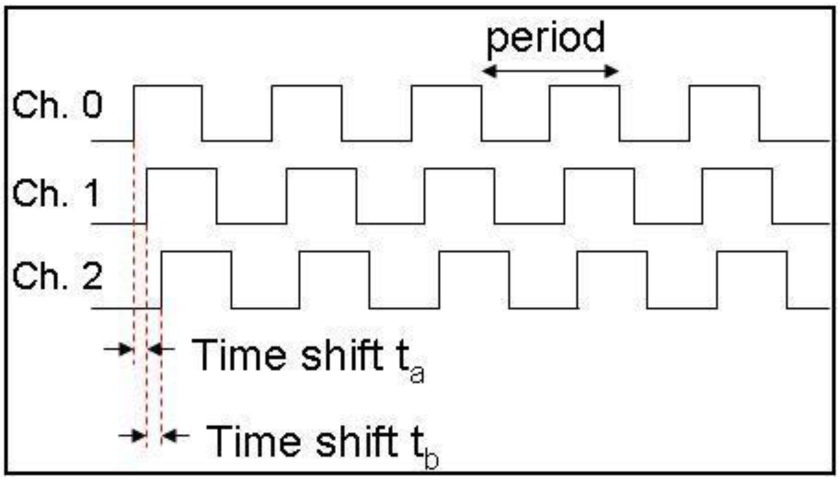


Figure 3: shifted period

8.2.6 Pwm_ConfigType

PWM111:

Name:	`Pwm_ConfigType`	
Type:	`Structure`	
Range:	`Hardware dependent structure.`	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.	

PWM061: Pwm_ConfigType is a type of data structure containing the initialization data for the PWM driver.
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:
    - Variable period
    - Fixed period
    - Fixed period, shifted (optional, if supported by hardware)

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

PWM095:

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.	

PWM007: The function Pwm_Init shall initialize all internals variables and the used PWM structure of the microcontroller according to the parameters specified in ConfigPtr.

PWM062: The function Pwm_Init shall only initialize the configured resources and shall not touch resources that are not configured in the configuration file.

PWM009: The function Pwm_Init shall start all PWM channels with the configured default values. If the duty cycle parameter equals:
- 0% or 100% : Then the PWM output signal shall be in the state according to the configured polarity parameter
- >0% and <100%: Then the PWM output signal shall be modulated according to parameters period, duty cycle and configured polarity.

PWM052: 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.

PWM093: The users of the Pwm module shall not call the function Pwm_Init during a running operation.

PWM046: If development error detection is enabled for the Pwm module, the function Pwm_Init shall raise development error `PWM_E_PARAM_CONFIG` if `ConfigPtr` is a null pointer.

Regarding error detection, the requirement PWM051 is applicable to the function Pwm_Init.

PWM116: The Pwm module's environment shall not call any function of the Pwm module before having called Pwm_Init.

PWM118: 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.

PWM120: For pre-compile and link time configuration variants, a NULL pointer shall be passed to the initialization routine. In this case the check for this NULL pointer has to be omitted.

PWM121: A re-initialization of the Pwm driver by executing the Pwm_Init() function requires a de-initialization before by executing a Pwm_DeInit().

8.3.2 Pwm_DeInit

PWM096:

Service name:	Pwm_DeInit

Syntax:	`void Pwm_DeInit(`
	
	`)`

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.

PWM010: The function Pwm_DeInit shall de-initialize the PWM module.

PWM011: The function Pwm_DeInit shall set the state of the PWM output signals to the idle state.

PWM012: The function Pwm_DeInit shall disable PWM interrupts and PWM signal edge notifications.

PWM080: The function Pwm_DeInit shall be pre compile time configurable On/Off by the configuration parameter: PwmDeInitApi.

Regarding error detection, the requirements PWM117 and PWM051 are applicable to the function Pwm_DeInit.

8.3.3 Pwm_SetDutyCycle

PWM097:

Service name:	Pwm_SetDutyCycle	
Syntax:	`void Pwm_SetDutyCycle( Pwm_ChannelType ChannelNumber, uint16 DutyCycle )`	
Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
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.	

PWM013: The function Pwm_SetDutyCycle shall set the duty cycle of the PWM channel.

PWM014: The function Pwm_SetDutyCycle shall set the PWM output state according to the configured polarity parameter, when the duty cycle = 0% or 100%.

PWM016: 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%.

PWM017: 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 PWM058 is applicable to the function Pwm_SetDutyCycle.

Regarding scaling definition of duty cycle parameter, the requirement PWM059 is applicable to the function Pwm_SetDutyCycle.

PWM018: The driver shall forbid the spike on the PWM output signal.

Regarding error detection, the requirements PWM117, PWM047 and PWM051 are applicable to the function Pwm_SetDutyCycle.

PWM082: The function Pwm_SetDutyCycle shall be pre compile time configurable On/Off by the configuration parameter: PwmSetDutyCycle.

8.3.4 Pwm_SetPeriodAndDuty

PWM098:

Service name:	Pwm_SetPeriodAndDuty	

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

Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
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	

PWM019: The function Pwm_SetPeriodAndDuty shall set the period and the duty cycle of a PWM channel.

PWM076: 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.

PWM020: The driver shall avoid spikes on the PWM output signal when updating the PWM period and duty.

The PWM duty cycle parameter is necessary to maintain the consistency between frequency and duty cycle. Refer to PWM058: and PWM059 : to know the scaling and format definition of duty cycle parameter

Regarding error detection, the requirements PWM117, PWM045, PWM047 and PWM051 are applicable to the function Pwm_SetPeriodAndDuty.

PWM041: The function Pwm_SetPeriodAndDuty shall allow changing the period only for those PWM channels which have been configured with Pwm_ChannelClassType PWM_VARIABLE_PERIOD (see PWM110).

PWM083: The function Pwm_SetPeriodAndDuty shall be pre compile time configurable On/Off by the configuration parameter: PwmSetPeriodAndDuty.

8.3.5 Pwm_SetOutputToIdle

PWM099:

Service name:	Pwm_SetOutputToIdle	

Syntax:	`void Pwm_SetOutputToIdle(`
	`    Pwm_ChannelType ChannelNumber`
	`)`	

Service ID[hex]:	0x04	
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:	None	
Description:	Service sets the PWM output to the configured Idle state.	

PWM021: The function Pwm_SetOutputToIdle shall set immediately the PWM output to the configured Idle state.

Regarding error detection, the requirements PWM117, PWM047 and PWM051 are applicable to the function Pwm_SetOutputToIdle.

PWM084: The function Pwm_SetOutputToIdle shall be pre compile time configurable On/Off by the configuration parameter: PwmSetOutputToIdle.

PWM086: After the call of the function Pwm_SetOutputToIdle, variable period type channels shall be reactivated either using the Api Pwm_SetPeriodAndDuty() to activate the PWM channel with the new passed period or Api Pwm_SetDutyCycle() to activate the PWM channel with the old period.

PWM119: After the call of the function Pwm_SetOutputToIdle, fixed period type channels shall be reactivated using only the API Api Pwm_SetDutyCycle() to activate the PWM channel with the old period.

8.3.6 Pwm_GetOutputState

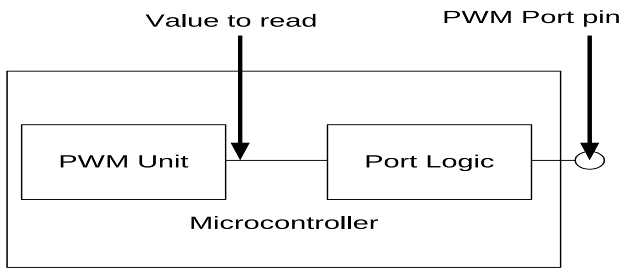
PWM100:

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.	

PWM022: 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 PWM117, PWM047 and PWM051 are applicable to the function Pwm_GetOutputState.

PWM085: The function Pwm_GetOutputState shall be pre compile time configurable On/Off by the configuration parameter: PwmGetOutputState.

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.

8.3.7 Pwm_DisableNotification

PWM101:

Service name:	Pwm_DisableNotification	
Syntax:	`void Pwm_DisableNotification( Pwm_ChannelType ChannelNumber )`	
Service ID[hex]:	0x06	
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:	None	
Description:	Service to disable the PWM signal edge notification.	

PWM023: The function Pwm_DisableNotification shall disable the PWM signal edge notification.

PWM112: The function Pwm_DisableNotification shall be pre compile time configurable On/Off by the configuration parameter: PwmNotificationSupported.

Regarding error detection, the requirements PWM117, PWM047 and PWM051 are applicable to the function Pwm_DisableNotification.

8.3.8 Pwm_EnableNotification

PWM102:

Service name:	Pwm_EnableNotification	
Syntax:	`void Pwm_EnableNotification( Pwm_ChannelType ChannelNumber, Pwm_EdgeNotificationType Notification )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
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.	

PWM024: The function Pwm_EnableNotification shall enable the PWM signal edge notification according to notification parameter.

PWM081: The function Pwm_EnableNotification shall cancel pending interrupts.

PWM113: The function Pwm_EnableNotification shall be pre compile time configurable On/Off by the configuration parameter: PwmNotificationSupported.

Regarding error detection, the requirements PWM117, PWM047 and PWM051 are applicable to the function Pwm_EnableNotification.

8.3.9 Pwm_GetVersionInfo

PWM103:

Service name:	Pwm_GetVersionInfo	

Syntax:	`void Pwm_GetVersionInfo(`
	`    Std_VersionInfoType* versioninfo`
	`)`	

Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Service returns the version information of this module.	

PWM068: The function Pwm_GetVersionInfo shall return the version information of this module. The version information includes:
- Module Id
- Vendor Id
- Vendor specific version numbers (BSW00407).

PWM069: The function Pwm_GetVersionInfo shall be pre compile time configurable On/Off by the configuration parameter: PwmVersionInfoApi.

PWM114: If source code for caller and callee of Pwm_GetVersionInfo is available, the Pwm module should realize Pwm_GetVersionInfo as a macro, defined in the module’s header file.

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

The PWM driver offers only synchronous services and therefore doesn't need any scheduled functions.

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

This chapter defines all interfaces which are required to fulfill the core functionality of the module.

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

8.6.2 Optional Interfaces

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

PWM104:

API function	***Description***
Dem_ReportErrorStatus	Reports errors to the DEM.
Det_ReportError	Service to report development errors.

8.6.3 Configurable interfaces

In this chapter all interfaces are listed where the target function could be configured. The target function is usually a call-back function. The names of these kinds of interfaces are not fixed because they are configurable.

PWM105:

Service name:	Pwm_Notification_<#Channel>

Syntax:	`void Pwm_Notification_<#Channel>(`
	
	`)`

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>.

PWM025: The Pwm module shall call the function Pwm_Notification_<#Channel> accordingly to the last call of Pwm_EnableNotification and Pwm_DisableNotification for channel <#Channel>.

PWM026: The Pwm module shall shall reset the interrupt flag associated to the notification Pwm_Notification_<#Channel>

PWM115: The Pwm module shall only provide the functionality of PWM025 and PWM026 if the configuration parameter PwmNotificationSupported is ON.

8.7 API parameter checking

PWM051: If development error detection for the Pwm module is enabled: when a development error occurs, the corresponding PWM function shall:

- Report the error to the Development Error Tracer.
- Skip the desired functionality in order to avoid any corruptions of data or hardware registers: This means leave the function without any actions.
- Return PWM_LOW for the function Pwm_GetOutputState.

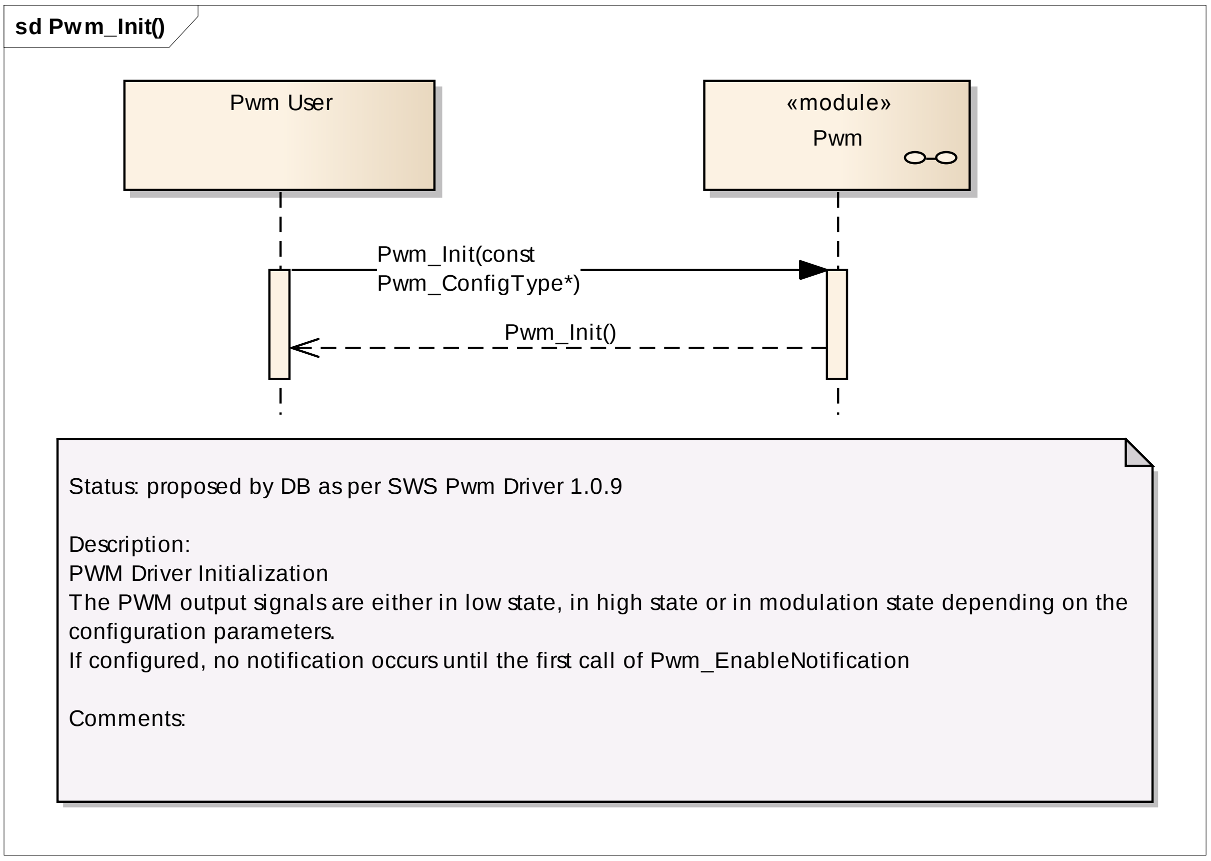
PWM117: If development error detection for the Pwm module is enabled: if any function (except Pwm_Init) is called before Pwm_Init has been called, the called function shall raise development error `PWM_E_UNINIT`.

PWM045: If development error detection for the Pwm module is enabled: the PWM functions shall check the channel class type and raise development error `PWM_E_PERIOD_UNCHANGEABLE` if the PWM channel is not declared as a variable period type.

PWM047: 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



Status: proposed by DB as per SWS Pwm Driver 1.0.9

Description:
PWM Driver Initialization
The PWM output signals are either in low state, in high state or in modulation state depending on the configuration parameters.
If configured, no notification occurs until the first call of Pwm_EnableNotification

Comments:

Figure 4: Pwm initialization

9.2 De-initialization

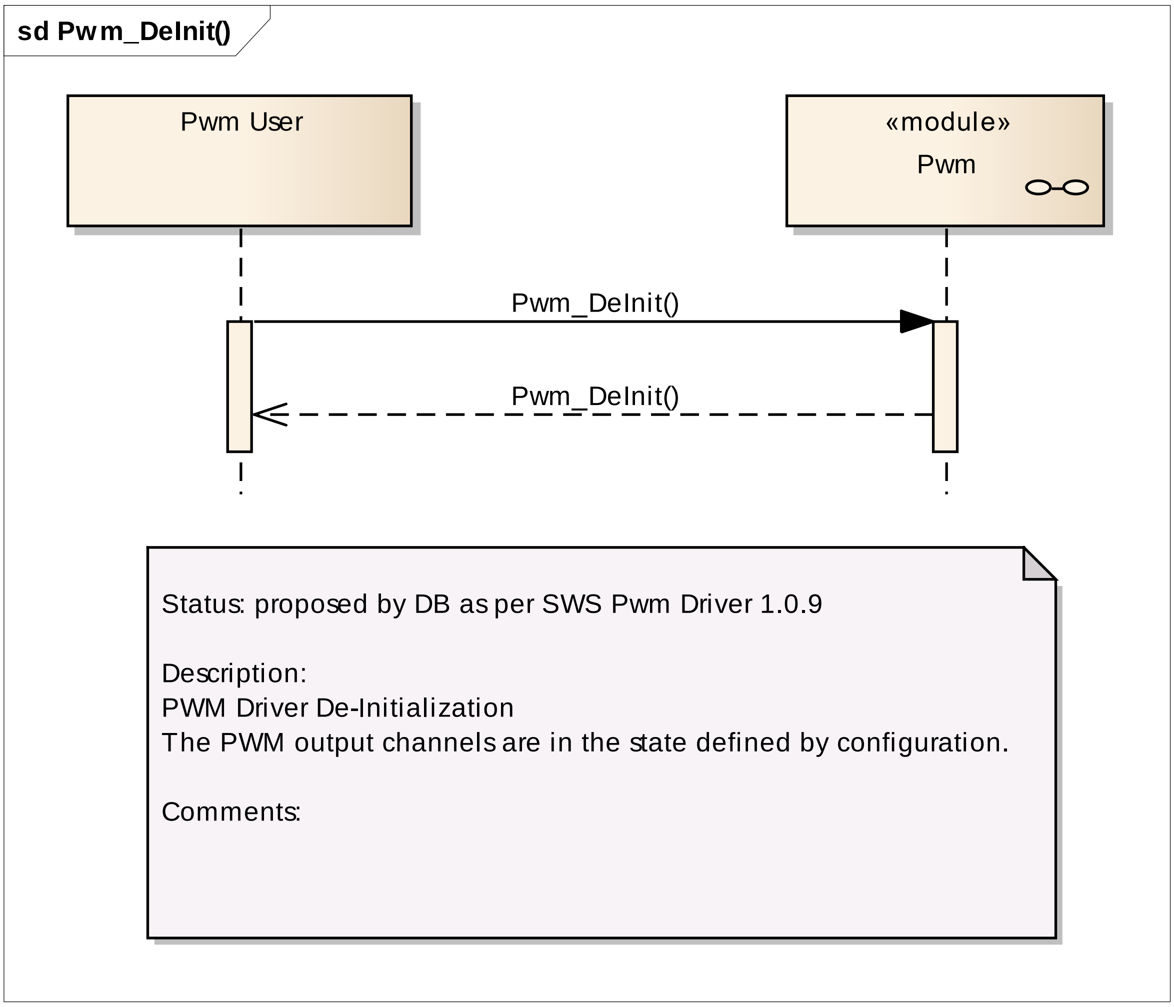
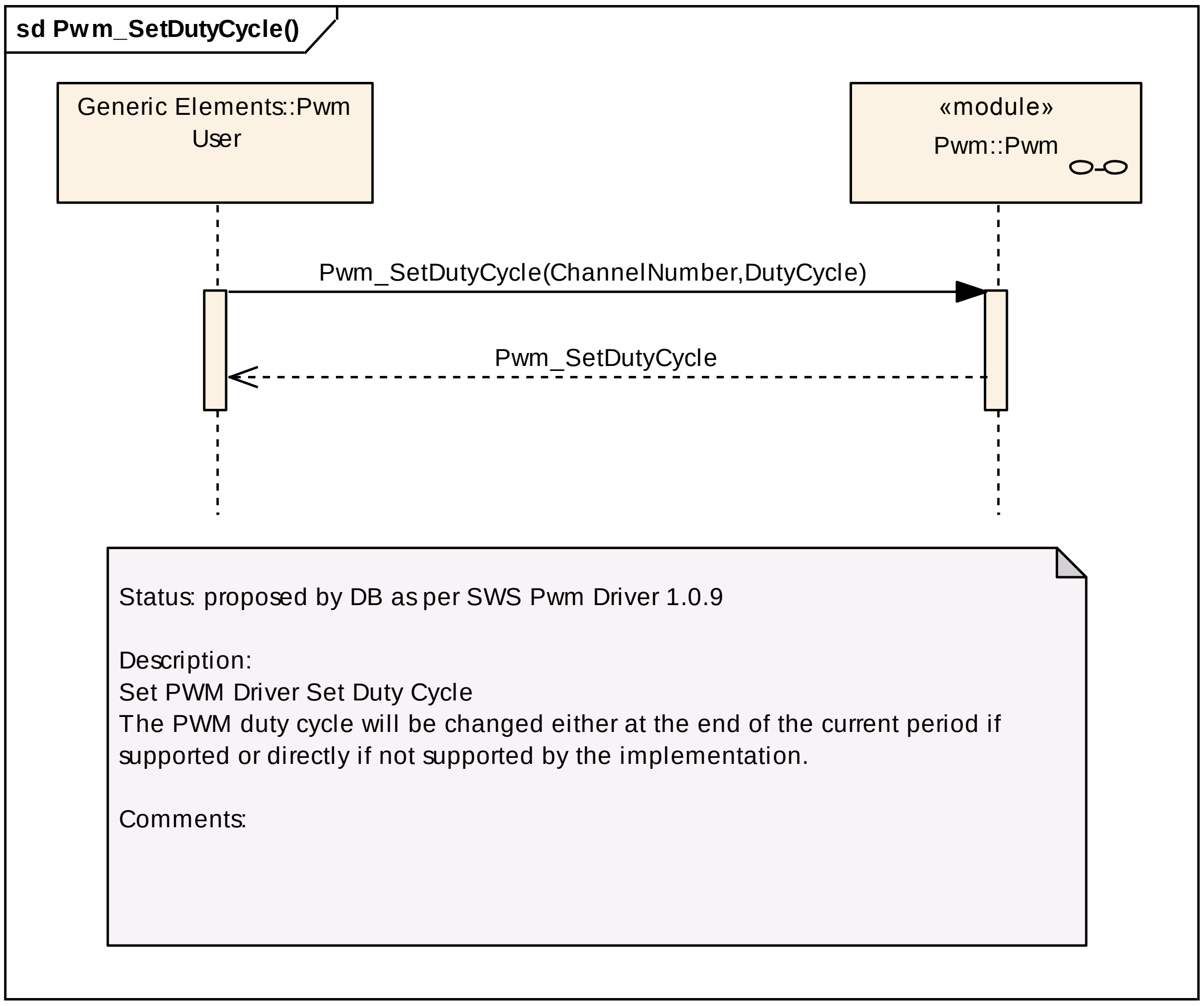

sd Pwm_DeInit()
Pwm User
«module»
Pwm
Pwm_DeInit()
Pwm_DeInit()
Status: proposed by DB as per SWS Pwm Driver 1.0.9
Description:
PWM Driver De-Initialization
The PWM output channels are in the state defined by configuration.
Comments:

Figure 5: Pwm de-initialization

9.3 Setting the duty cycle


sd Pwm_SetDutyCycle()
Generic Elements::Pwm User
«module» Pwm::Pwm
Pwm_SetDutyCycle(ChannelNumber,DutyCycle)
Pwm_SetDutyCycle

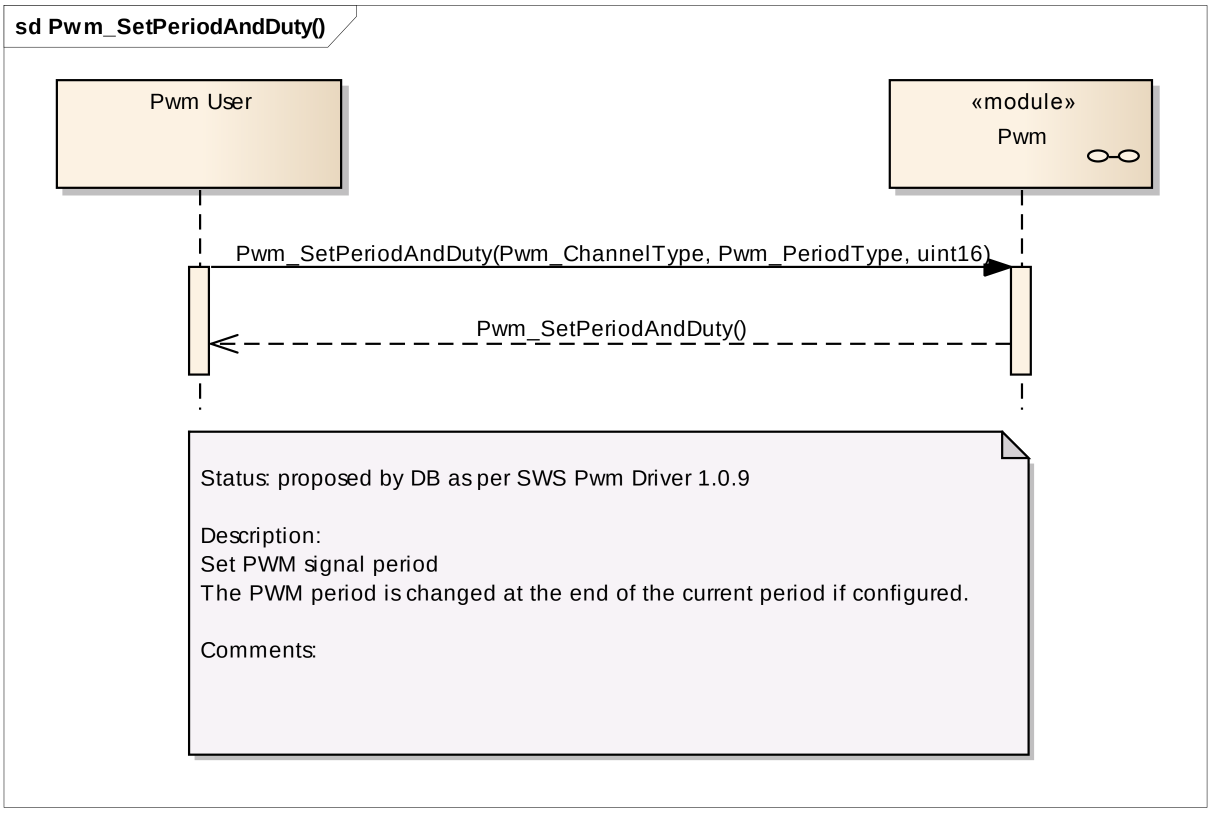
Status: proposed by DB as per SWS Pwm Driver 1.0.9

Description:
Set PWM Driver Set Duty Cycle
The PWM duty cycle will be changed either at the end of the current period if supported or directly if not supported by the implementation.

Comments:

Figure 6: Setting the duty cycle

9.4 Setting the period and the duty


sd Pwm_SetPeriodAndDuty()
Pwm User
«module»
Pwm
Pwm_SetPeriodAndDuty(Pwm_ChannelType, Pwm_PeriodType, uint16)
Pwm_SetPeriodAndDuty()

Status: proposed by DB as per SWS Pwm Driver 1.0.9

Description:
Set PWM signal period
The PWM period is changed at the end of the current period if configured.

Comments:

Figure 7: Setting period and duty cycle

9.5 Setting the PWM output to idle

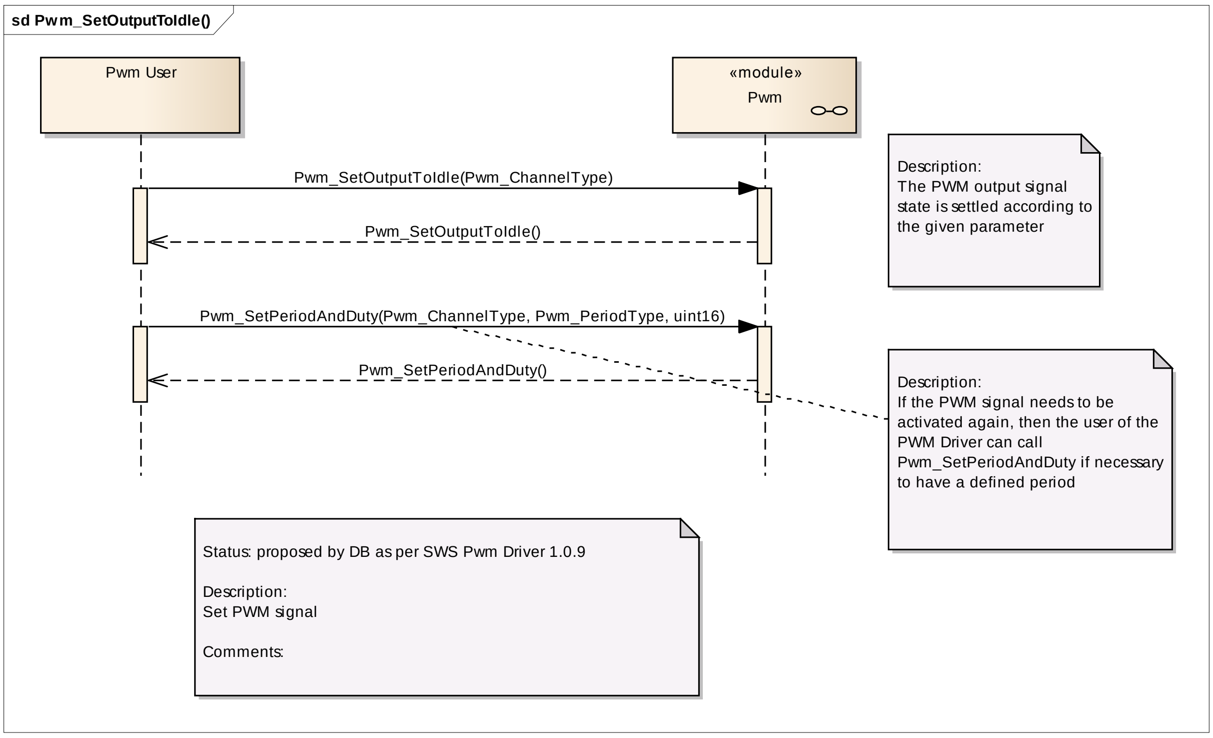

sd Pwm_SetOutputToIdle()
Pwm User
«module»
Pwm
Pwm_SetOutputToIdle(Pwm_ChannelType)
Pwm_SetOutputToIdle()
Pwm_SetPeriodAndDuty(Pwm_ChannelType, Pwm_PeriodType, uint16)
Pwm_SetPeriodAndDuty()
Description:
The PWM output signal state is settled according to the given parameter
Description:
If the PWM signal needs to be activated again, then the user of the PWM Driver can call Pwm_SetPeriodAndDuty if necessary to have a defined period
Status: proposed by DB as per SWS Pwm Driver 1.0.9
Description:
Set PWM signal
Comments:

Figure 8: Setting Pwm output to idle

9.6 Getting the PWM Output state

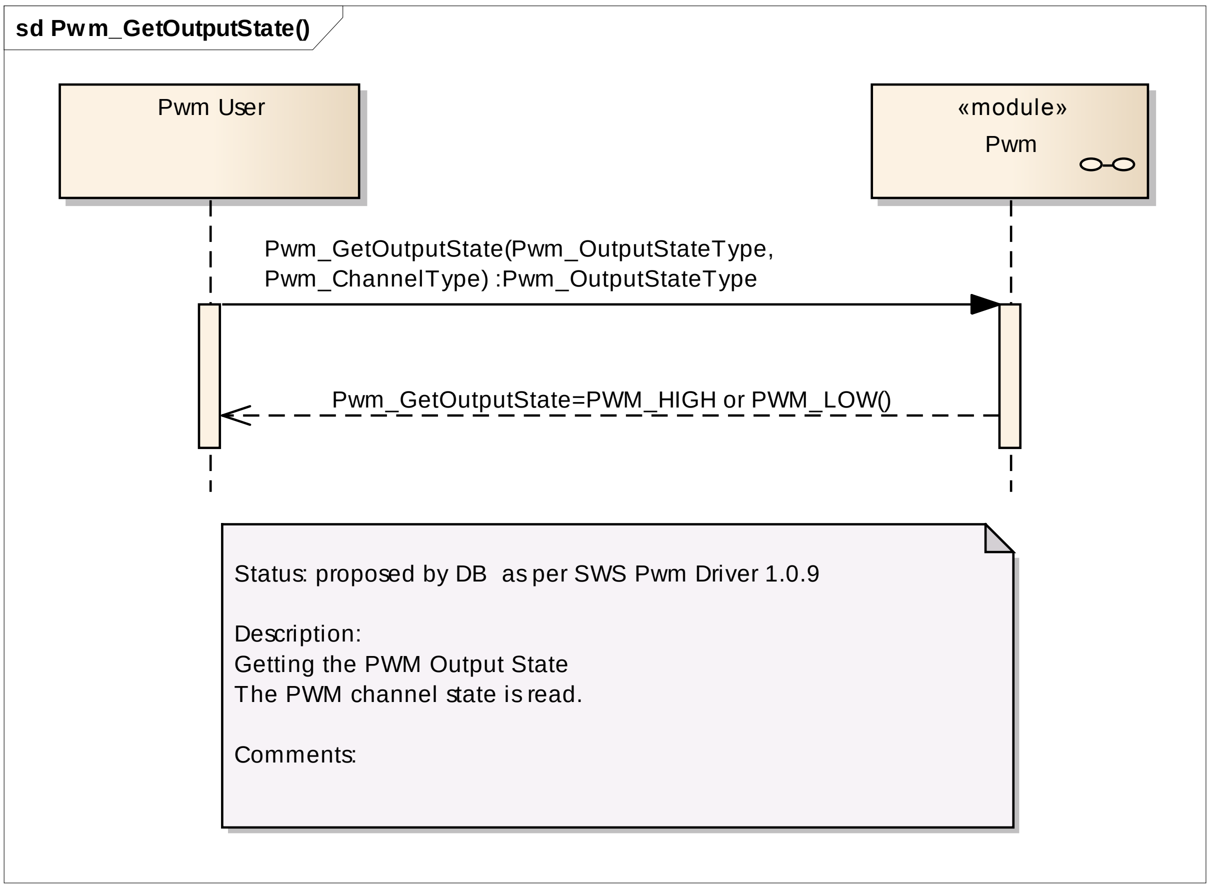
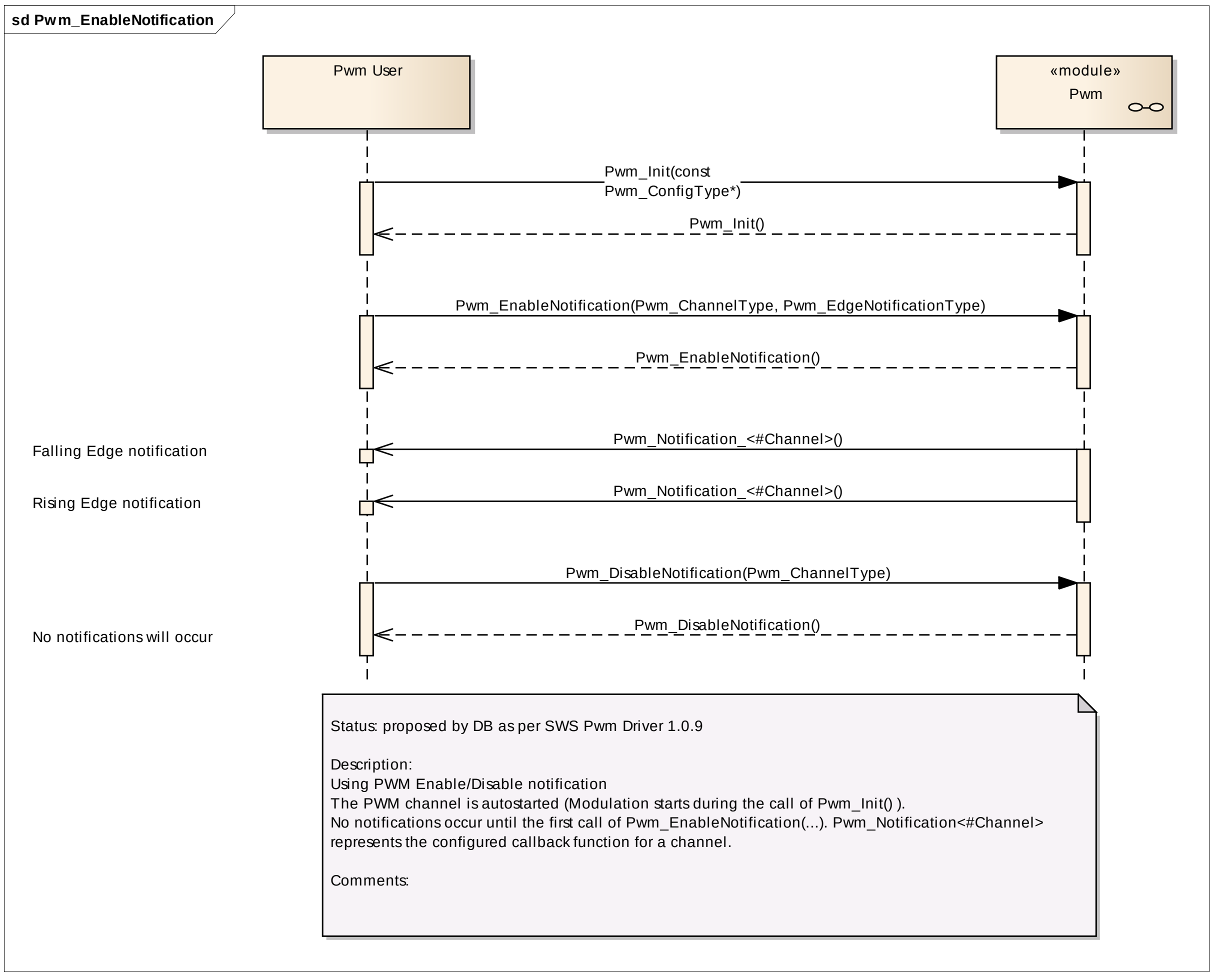

sd Pwm_GetOutputState()
Pwm User
«module»
Pwm
Pwm_GetOutputState(Pwm_OutputStateType,
Pwm_ChannelType) :Pwm_OutputStateType
Pwm_GetOutputState=PWM_HIGH or PWM_LOW()
Status: proposed by DB as per SWS Pwm Driver 1.0.9
Description:
Getting the PWM Output State
The PWM channel state is read.
Comments:

Figure 9: Getting Pwm output state

9.7 Using the PWM notifications


sd Pwm_EnableNotification
Pwm User
«module»
Pwm
Pwm_Init(const Pwm_ConfigType*)
Pwm_Init()
Pwm_EnableNotification(Pwm_ChannelType, Pwm_EdgeNotificationType)
Pwm_EnableNotification()
Falling Edge notification
Pwm_Notification_<#Channel>()
Rising Edge notification
Pwm_Notification_<#Channel>()
Pwm_DisableNotification(Pwm_ChannelType)
No notifications will occur
Pwm_DisableNotification()

Status: proposed by DB as per SWS Pwm Driver 1.0.9

Description:
Using PWM Enable/Disable notification
The PWM channel is autostarted (Modulation starts during the call of Pwm_Init()).
No notifications occur until the first call of Pwm_EnableNotification(...). Pwm_Notification<#Channel> represents the configured callback function for a channel.

Comments:

Figure 10: 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

In addition to this section, it is highly recommended to read the documents:
- AUTOSAR Layered Software Architecture [1]
- AUTOSAR ECU Configuration Specification [6]
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Containers

Containers structure the set of configuration parameters. This means:
- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.1.3 Specification template for configuration parameters

The following tables consist of three sections:
- the general section
- the configuration parameter section
- the section of included/referenced containers

Pre-compile time - specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

Label	*Description*
x	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	*Description*
x	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	*Description*
x	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* - the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* - the configuration parameter shall be of configuration class *Post Build* and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters Functional specification and Chapter API specification.

10.2.1 Variants

PWM079: Variant **PC** is limited to pre-compile configuration parameters only.

PWM077: Variant **PB**: has been defined for this module and allows Mix of precompile and **P**ost **B**uild multiple selectable configurable configurations.

10.2.2 Pwm

Module Name	*Pwm*
Module Description	Configuration of Pwm (Pulse Width Modulation) module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PwmChannelConfigSet	1	Multiple Configuration Set Container
PwmConfigurationOfOptApiServices	1	--
PwmGeneral	1	--

10.2.3 PwmGeneral

SWS Item	**PWM004 :**
Container Name	PwmGeneral{PwmModuleConfiguration}
Description	--
Configuration Parameters	

SWS Item	**PWM131 :**		
Name	PwmDevErorDetect {PWM_DEV_ERROR_DETECT}		
Description	Switch for enabling the development error detection.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**PWM132 :**		
Name	PwmDutycycleUpdatedEndperiod {PWM_DUTYCYCLE_UPDATED_ENDPERIOD}		
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	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**ECUC_Pwm_00139 :**		
Name	PwmIndex		
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.		
Multiplicity	1		
Type	IntegerParamDef		
Range	..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants

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

SWS Item	**PWM133 :**		
Name	PwmNotificationSupported {PWM_NOTIFICATION_SUPPORTED}		
Description	Switch to indicate that the notifications are supported		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**PWM134 :**		
Name	PwmPeriodUpdatedEndperiod {PWM_DUTY_PERIOD_UPDATED_ENDPERIOD}		
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	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.2.4 PwmChannel

SWS Item	**PWM027 :**
Container Name	PwmChannel{PwmChannelConfiguration}
Description	Configuration of an individual PWM channel.
Configuration Parameters	

SWS Item	**PWM119 :**		
Name	PwmChannelClass {PWM_CHANNEL_CLASS}		
Description	Class of PWM Channel. ImplementationType: Pwm_ChannelClassType		
Multiplicity	1		
Type	EnumerationParamDef		
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.	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	

	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**PWM120 :**		
Name	PwmChannelId		
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	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 ..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**PWM121 :**		
Name	PwmDutycycleDefault {PWM_DUTYCYLE_DEFAULT}		
Description	Value of duty cycle used for Initialization 0, represents 0% 0x8000 represents 100%		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 32768		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**PWM122 :**		
Name	PwmIdleState {PWM_IDLE_STATE}		
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	EnumerationParamDef		
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.	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**PWM123 :**		
Name	PwmNotification {Pwm_Notification}		
Description	Definition of the Callback function.		
Multiplicity	1		
Type	FunctionNameDef		
Default value	"NULL"		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build	M	VARIANT-POST-BUILD

	time		
Scope / Dependency	scope: ECU		

SWS Item	**PWM124 :**		
Name	PwmPeriodDefault {PWM_PERIOD_DEFAULT}		
Description	Value of period used for Initialization.(in seconds).		
Multiplicity	1		
Type	FloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**PWM125 :**		
Name	PwmPolarity {PWM_POLARITY}		
Description	Defines the starting polarity of each PWM channel.		
Multiplicity	1		
Type	EnumerationParamDef		
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.	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	M	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.5 PwmChannelConfigSet

SWS Item	**PWM118 :**
Container Name	PwmChannelConfigSet [Multi Config Container]
Description	Multiple Configuration Set Container
Configuration Parameters	

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

10.2.6 PwmConfigurationOfOptApiServices

SWS Item	**PWM126 :**
Container Name	PwmConfigurationOfOptApiServices
Description	--
Configuration Parameters	

SWS Item	**ECUC_Pwm_00141 :**
Name	PwmDeInitApi {PWM_DE_INIT_API}
Description	Adds / removes the service Pwm_DeInit() from the code.
Multiplicity	1
Type	BooleanParamDef
Default value	--

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

SWS Item	**PWM127 :**		
Name	PwmGetOutputState {PWM_GET_OUTPUT_STATE_API}		
Description	--		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**PWM128 :**		
Name	PwmSetDutyCycle {PWM_SET_DUTY_CYCLE_API}		
Description	--		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**PWM129 :**		
Name	PwmSetOutputToIdle {PWM_SET_OUTPUT_TO_IDLE_API}		
Description	--		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**PWM130 :**		
Name	PwmSetPeriodAndDuty {PWM_SET_PERIOD_AND_DUTY_API}		
Description	--		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build	--	

	time		
Scope / Dependency	scope: Module		

SWS Item	**PWM135 :**		
Name	PwmVersionInfoApi {PWM_VERSION_INFO_API}		
Description	Switch to indicate that the Pwm_ GetVersionInfo is supported		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

The standard common published information like

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

is provided in the BSW Module Description Template (see [7] Figure 4.1 and Figure 7.1).

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