Document Title	Specification of PWM Driver
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	2.1.0
Document Status	Final
Part of Release	2.1
Revision	0014

Document Change History			
Date	Version	Changed by	Change Description
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 PWM_DUTY_PERIOD_UPDATED_ENDPERIOD • Updated PWM signal description figure • Legal disclaimer revised • "Advice for users" revised • "Revision Information" added
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

Page left intentionally blank

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).
Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

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 services 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.

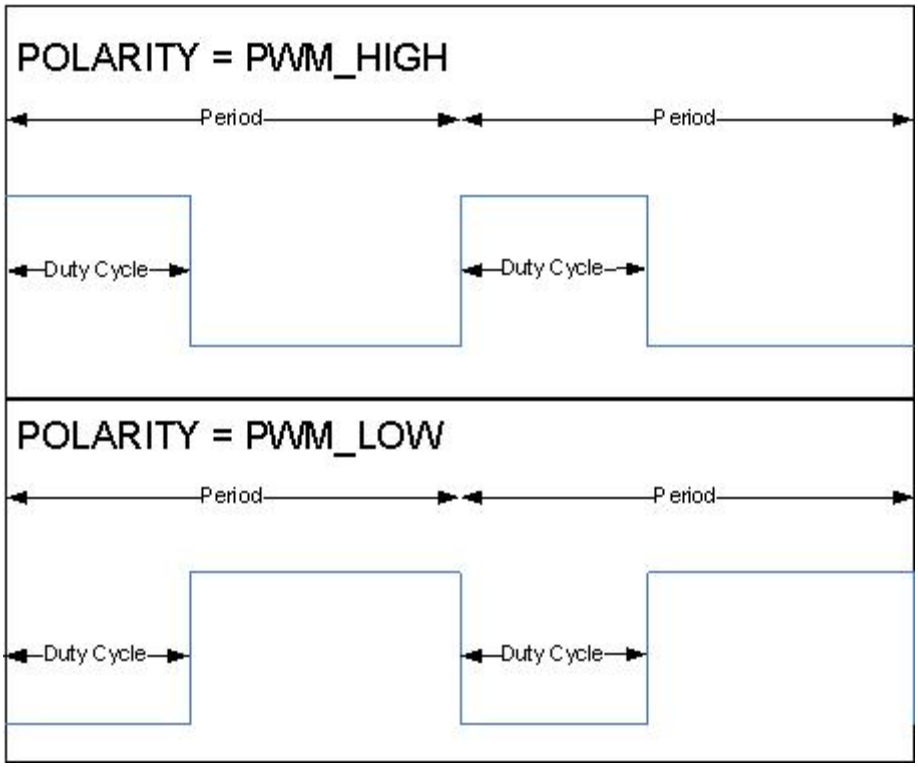


Figure 1: PWM signal description

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
https://svn.autosar.org/repos/10Releases/
AUTOSAR_LayeredSoftwareArchitecture.pdf

[2] General Requirements on SPAL
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_SPAL_General.pdf

[3] General Requirements on Basic Software Modules
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_General.pdf

[4] Specification of Development Error Tracer
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[5] Specification of MCU Driver
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_MCU_Driver.pdf

3.2 Related standards and norms

None

4 Constraints and assumptions

4.1 Limitations

PWM040: All PWM channels which work with the same microcontroller timer shall have the same frequency.

PWM001 The PWM Driver doesn’t cover a 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 code file structure shall not be defined within this specification.

5.1.2 Header file structure

PWM075: The include file structure shall be as follows:

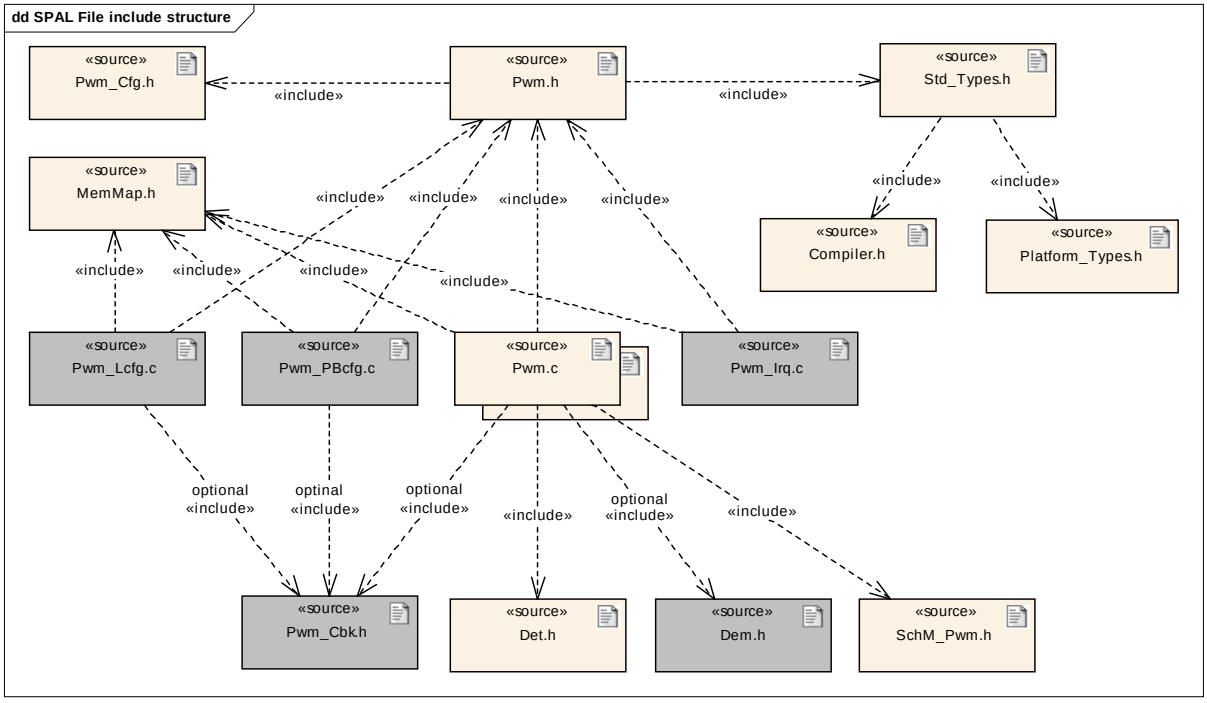


Figure 2: Header file structure

PWM066: The module shall include the Dem.h file. 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	PWM073 PWM074
[BSW00387] Specify the configuration class of callback function	PWM027 parameter Pwm_Notification
[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

Requirement	Satisfied by
[BSW00416] Sequence of Initialization	Not Applicable (SW Integration requirement)
[BSW00406] Check module initialization	PWM044
[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

Requirement	Satisfied by
[BSW00420] Production relevant error event rate detection	PWM005 PWM006
[BSW00417] Reporting of Error Events by Non-Basic Software	Not Applicable (Module is a BSW)
[BSW00323] API parameter checking	PWM044 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 PWM044 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

Requirement	Satisfied by
	account during implementation)
[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)

Requirement	Satisfied by
[BSW00304] AUTOSAR integer data types	Not applicable (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

Requirement	Satisfied by
	account during implementation)
[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) PWM071
[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)PWM038
[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)

Requirements	Satisfied by
[BSW12078] Runtime and memory efficiency	Not applicable (Requirement to be taken into 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	PWM040
[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 PWM060
[BSW12299] Activation of PWM edge notification	PWM023 PWM024 PWM025
[BSW12386] No PWM emulation	PWM001

7 Functional specification

7.1 General behavior

PWM034: All services except Pwm_Init, Pwm_DeInit and Pwm_GetVersionInfo are re-entrant for different PWM channel numbers. However in order to keep a simple driver implementation, no check will be performed by the driver and this is the user responsibility to check the integrity if several calls are made during run time in different tasks or ISRs for a 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 driver shall be of the unit ticks.

7.3 Error classification

PWM002: The following errors and exceptions shall be detectable by the PWM driver depending on its build version (development/production mode):
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`
--	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 `PWM_DEV_ERROR_DETECT` (see chapter 10) shall activate or deactivate the detection of all development errors.

PWM064: If the `PWM_DEV_ERROR_DETECT` 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 *`PWM_DEV_ERROR_DETECT`* is set (see chapter 10).

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 shall be 16 Bits.

PWM059: The PWM Driver shall provide 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

8.1.1 Standard types

In this chapter all types included from the Std_Types.h are listed:

- Std_VersionInfoType

8.2 Type definitions

8.2.1 Pwm_ChannelType

Type:	uint8 … uint32	
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

Type:	uint8 … uint32	
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

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

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

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`	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	

8.2.6 Pwm_ConfigType

Type:	structure	
Range:	Hardware dependent structure.	The contents of the initialization data structure are hardware specific.
Description:	**PWM061**: This is the 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: - 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

Service name:	Pwm_Init
Syntax:	`void Pwm_Init ( const Pwm_ConfigType *ConfigPtr )`
Service ID [hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	non re-entrant PWM034
Parameters (in):	`ConfigPtr` Pointer to configuration set

Parameters (out):	None --
Return value:	None --
Description:	**PWM007**: Service for PWM initialization. This function shall initialize all internals variables and the used PWM structure of the microcontroller according to the parameters specified in `ConfigPtr`. **PWM062**: This function shall only initialize the configured resources. Resources that are not configured in the configuration file shall not be touched. **PWM009**: After having finished the module initialization, all PWM channels are started with configured default values If duty cycle parameter equals: ▪ 0% or 100% : Then the PWM output signal will be in the state according to the configured polarity parameter ▪ >0% and <100%: Then the PWM output signal will be modulated according to parameters period, duty cycle and configured polarity. **PWM052**: If some notifications are configured, the notifications are disabled. The users of these notifications may not be ready and have to call Pwm_EnableNotification. Error Detection: PWM046 PWM051.
Caveats:	This service shall not be called during a running operation.
Configuration:	None

8.3.2 Pwm_DeInit

Service name:	Pwm_DeInit
Syntax:	`void Pwm_DeInit ( void )`
Service ID [hex]:	0x01
Sync/Async:	Synchronous
Reentrancy:	non re-entrant PWM034
Parameters (in):	None --
Parameters (out):	None --
Return value:	None --
Description:	**PWM010**: Service for PWM De-Initialization. **PWM011**: After the call of this service, the state of the PWM output signals shall be in idle state. **PWM012**: This service shall disable PWM interrupts and notifications. Error Detection: PWM044 PWM051.
Caveats:	None
Configuration:	**PWM080:** This function shall be pre compile time configurable `On/Off` by the configuration parameter: PWM_DE_INIT_API

8.3.3 Pwm_SetDutyCycle

Service name:	Pwm_SetDutyCycle	
Syntax:	`void Pwm_SetDutyCycle ( Pwm_ChannelType ChannelNumber, uint16 DutyCycle )`	
Service ID [hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	re-entrant PWM034	
Parameters (in):	`ChannelNumber`	Numeric identifier of the PWM
	`DutyCycle`	`Min=0x0000 Max=0x8000`
Parameters (out):	None	--
Return value:	None	--
Description:	**PWM013**: This service shall allow setting the duty cycle of the PWM channel. **PWM014**: If the duty cycle = 0% or 100%, then the PWM output state shall be in the state according to the configured polarity parameter. **PWM016**: If the duty cycle > 0 %and < 100%, then the PWM output signal will be modulated according to parameters period, duty cycle and configured polarity. . **PWM017**: The update of the duty cycle shall be performed always at the end of the period if supported by the implementation and configured. PWM058 Format definition of duty cycle parameter. PWM059 Scaling definition of duty cycle parameter. **PWM018**: The driver shall forbid the spike on the PWM output signal. Error Detection: PWM044 PWM047 PWM051.	
Caveats:	None	
Configuration:	`PWM_DUTYCYCLE_UPDATED_ENDPERIOD` **PWM082:** This function shall be pre compile time configurable On/Off by the configuration parameter: PWM_SET_DUTY_CYCLE_API	

8.3.4 Pwm_SetPeriodAndDuty

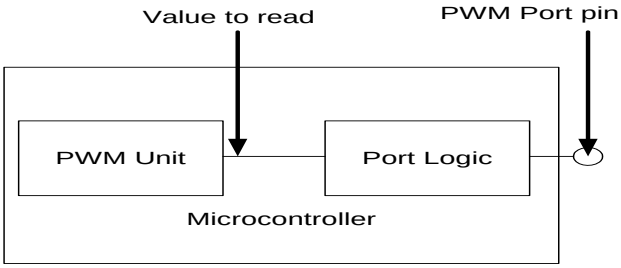
Service name:	Pwm_SetPeriodAndDuty	
Syntax:	`void Pwm_SetPeriodAndDuty ( Pwm_ChannelType ChannelNumber, Pwm_PeriodType Period, uint16 DutyCycle )`	
Service ID [hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	re-entrant PWM034	
Parameters	`ChannelNumber`	Numeric identifier of the PWM

(in):	`Period`	Period of the PWM signal
	`DutyCycle`	Min=0x0000 Max=0x8000
Parameters (out):	None	--
Return value:	None	--
Description:	**PWM019**: Service to set the period and the duty cycle of a PWM channel **PWM076**: The update of the period shall be performed always at the end of the current period if supported by the implementation and configured **PWM020**: The driver shall avoid spikes on the PWM output signal when updating the PWM period and duty. **PWM060**: 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 Error Detection: PWM044 PWM045 PWM047 PWM051.	
Caveats:	**PWM041**: The PWM driver shall allow changing the period only for the PWM channel declared as variable period type.	
Configuration:	`PWM_DUTY_PERIOD_UPDATED_ENDPERIOD` **PWM083:** This function shall be pre compile time configurable On/Off by the configuration parameter: PWM_SET_PERIOD_AND_DUTY_API	

8.3.5 Pwm_SetOutputToIdle

Service name:	Pwm_SetOutputToIdle	
Syntax:	`void Pwm_SetOutputToIdle ( Pwm_ChannelType ChannelNumber )`	
Service ID [hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	re-entrant PWM034	
Parameters (in):	`ChannelNumber`	Numeric identifier of the PWM
Parameters (out):	None	--
Return value:	None	--
Description:	**PWM021**: This service shall set immediately the PWM output to the configured Idle state. Error Detection: PWM045 PWM044 PWM047 PWM051.	
Caveats:	After the call of this function, the PWM channel may be activated again using only the function Pwm_SetPeriodAndDuty.	
Configuration:	**PWM084:** This function shall be pre compile time configurable On/Off by the configuration parameter: PWM_SET_OUTPUT_TO_IDLE_API	

8.3.6 Pwm_GetOutputState

Service name:	Pwm_GetOutputState	
Syntax:	`Pwm_OutputStateType Pwm_GetOutputState ( Pwm_ChannelType ChannelNumber )`	
Service ID [hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	re-entrant PWM034	
Parameters (in):	`ChannelNumber`	Numeric identifier of the PWM
Parameters (out):	None	--
Return value:	PWM_HIGH	The PWM output state is high
	PWM_LOW	The PWM output state is low
Description:	**PWM022**: This service shall read the internal state of the PWM output signal and return it as defined in the diagram below  Error Detection: PWM044 PWM047 PWM051.	
Caveats:	Due to real time constraint and setting of the PWM channel (project dependant), the output state can be modified just after the call of this service.	
Configuration:	**PWM085:** This function shall be pre compile time configurable On/Off by the configuration parameter: PWM_GET_OUTPUT_STATE_API	

8.3.7 Pwm_DisableNotification

Service name:	Pwm_DisableNotification	
Syntax:	`void Pwm_DisableNotification ( Pwm_ChannelType ChannelNumber )`	
Service ID [hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	re-entrant PWM034	
Parameters (in):	`ChannelNumber`	Numeric identifier of the PWM
Parameters (out):	None	--
Return value:	None	--

Description:	**PWM023**: This service shall disable the PWM signal edge notification. Error Detection: PWM044 PWM047 PWM051.
Caveats:	None
Configuration:	`PWM_NOTIFICATION_SUPPORTED`

8.3.8 Pwm_EnableNotification

Service name:	Pwm_EnableNotification	
Syntax:	`void Pwm_EnableNotification ( Pwm_ChannelType ChannelNumber, Pwm_EdgeNotificationType Notification )`	
Service ID [hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	re-entrant PWM034	
Parameters (in):	`ChannelNumber`	Numeric identifier of the PWM
	`Notification`	Type of notification PWM_RISING_EDGE or PWM_FALLING_EDGE or PWM_BOTH_EDGES
Parameters (out):	None	--
Return value:	None	--
Description:	**PWM024**: This service shall enable the PWM signal edge notification according to notification parameter. **PWM081**: This service shall cancel pending interrupts. Error Detection: PWM044 PWM047 PWM051.	
Caveats:	None	
Configuration:	`PWM_NOTIFICATION_SUPPORTED`	

8.3.9 Pwm_GetVersionInfo

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 (out):	`versioninfo`	Pointer to where to store the version information of this module.

Return value:	None --
Description:	**PWM068**: This service returns the version information of this module. The version information includes: - Module Id - Vendor Id - Vendor specific version numbers (BSW00407). **PWM069**: This function shall be pre compile time configurable `On/Off` by the configuration parameter: `PWM_VERSION_INFO_API` Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.
Caveats:	None
Configuration:	`PWM_VERSION_INFO_API`

8.4 Callback notifications

PWM031: The Callback notifications shall be configurable as function pointers within the initialization data structure.

PWM032: The callback notifications shall have no parameters and no return value.

PWM033: If a callback notification is configured as null pointer, no callback shall be executed.

As the name of the PWM driver notifications are configurable by the user, refer to chapter Configurable interfaces for further details.

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

PWM073: 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

PWM074: This chapter defines all interfaces which are required to fulfill an optional functionality of the module.

API function	*Module*	*Description*	*Configuration parameter (description see chapter 10)*
Det_ReportError	Det	Development error notification	PWM_DEV_ERROR_DETECT

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.

Service name:	Pwm_Notification_<#Channel>

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

Service ID [hex]:	NA
Sync/Async:	Synchronous
Reentrancy:	PWM user implementation dependant
Parameters (in):	None --
Parameters (out):	None --
Return value:	None --
Description:	**PWM025**: This service shall be called by the PWM Driver accordingly to the last call of Pwm_EnableNotification for channel <#Channel>. **PWM026**: The PWM driver shall reset the interrupt flag associated to the notification. **PWM035**: No parameters are provided by this notification.
Caveats:	None
Configuration:	`PWM_NOTIFICATION_SUPPORTED`

8.7 API parameter checking

PWM051: if development error detection is enabled and when an error occurs, then the PWM 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 service without any actions.
- Return PWM_LOW for service Pwm_GetOutputState.

PWM044: The service Pwm_Init shall be called first before calling any other PWM services. If development error detection is enabled and if this sequence is not respected, then the following error code `PWM_E_UNINIT` will be reported to the Development Error Tracer.

PWM045: If development error detection is enabled, the PWM services shall check the channel class type and report the error `PWM_E_PERIOD_UNCHANGEABLE` to the Development Error Tracer if PWM channel is not declared as a variable period type.

PWM046: If development error detection is enabled, the parameter `ConfigPtr` shall be checked for not being a NULL pointer. This error shall be reported to the Development Error Tracer with error value `PWM_E_PARAM_CONFIG`.

PWM047: If development error detection is enabled, the parameter ChannelNumber shall be checked by PWM services. If ChannelNumber is invalid, then the error `PWM_E_PARAM_CHANNEL` shall be reported to the Development Error Tracer.

9 Sequence diagrams

9.1 Initialization

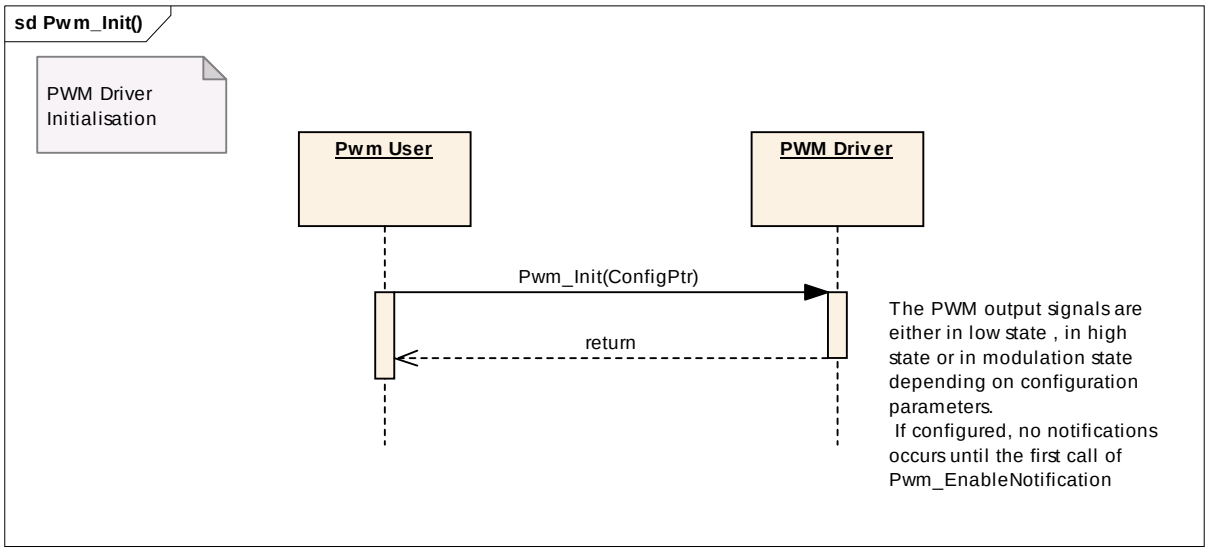


Figure 3: Pwm initialization

9.2 De-initialization

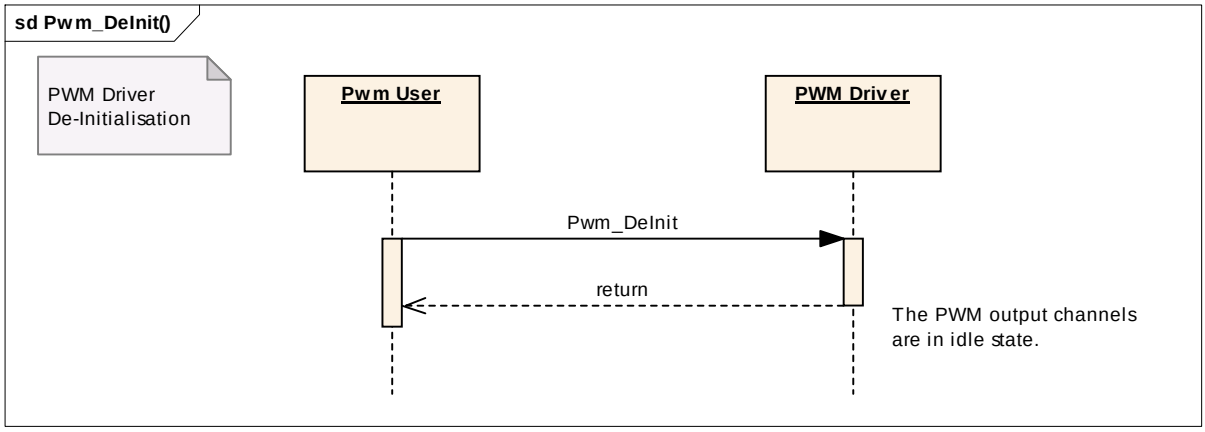


Figure 4: Pwm de-initialization

9.3 Setting the duty cycle

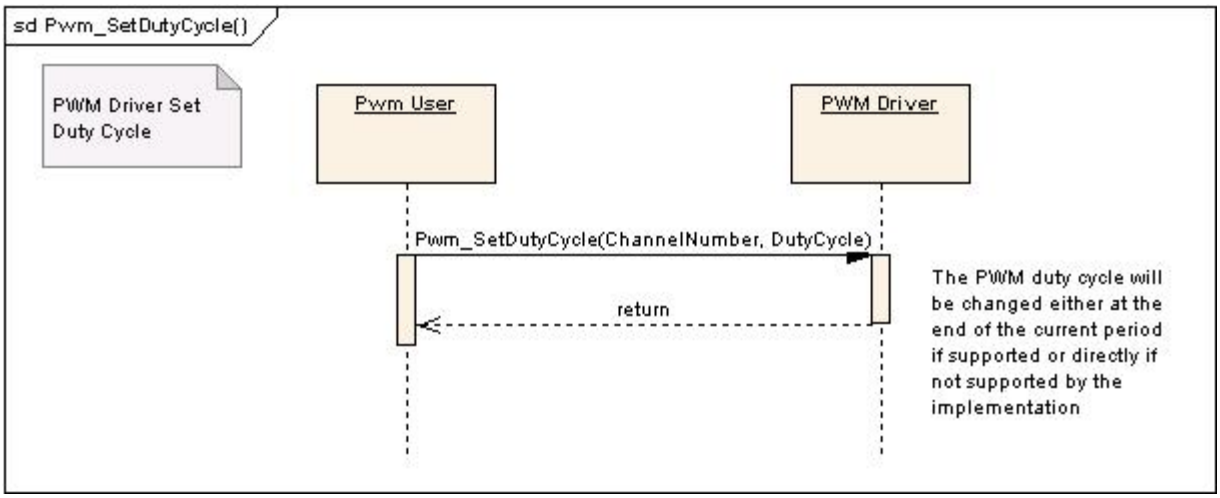


Figure 5: Setting the duty cycle

9.4 Setting the period and the duty

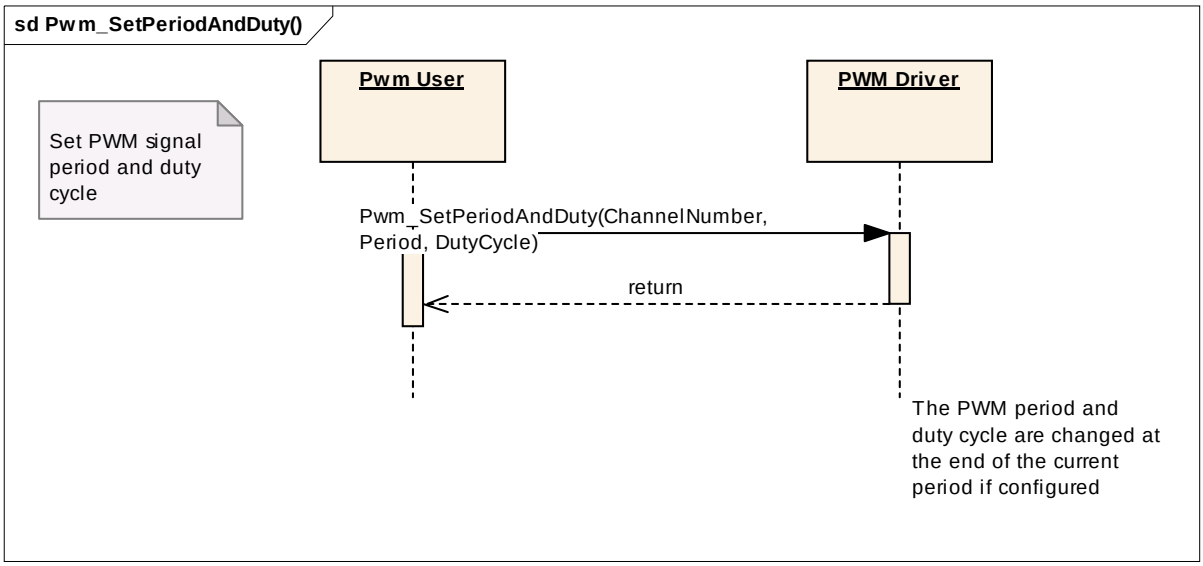


Figure 6: Setting period and duty cycle

9.5 Setting the PWM output to idle

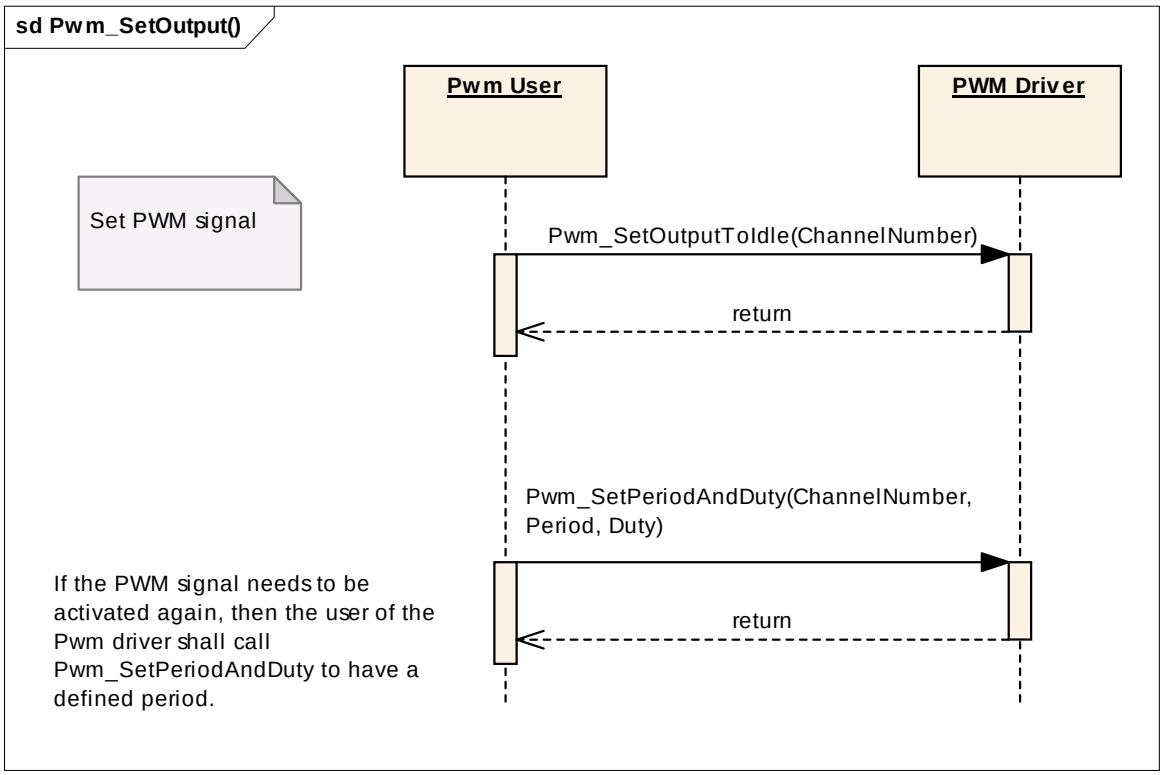


Figure 7: Setting Pwm output to idle

9.6 Getting the PWM Output state

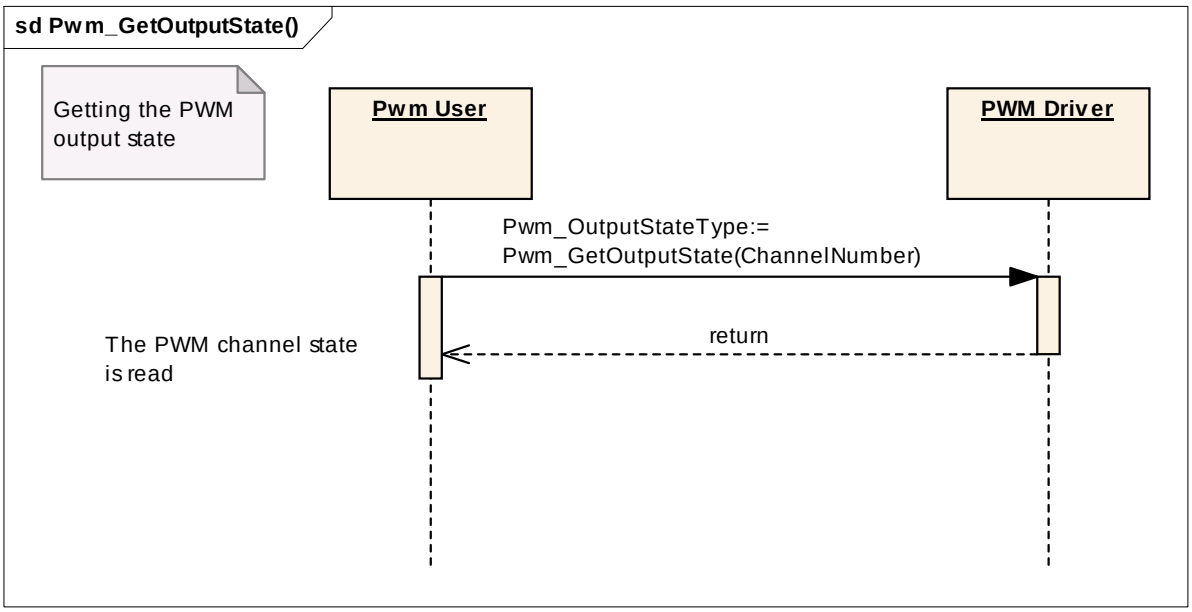


Figure 8: Getting Pwm output state

9.7 Using the PWM notifications

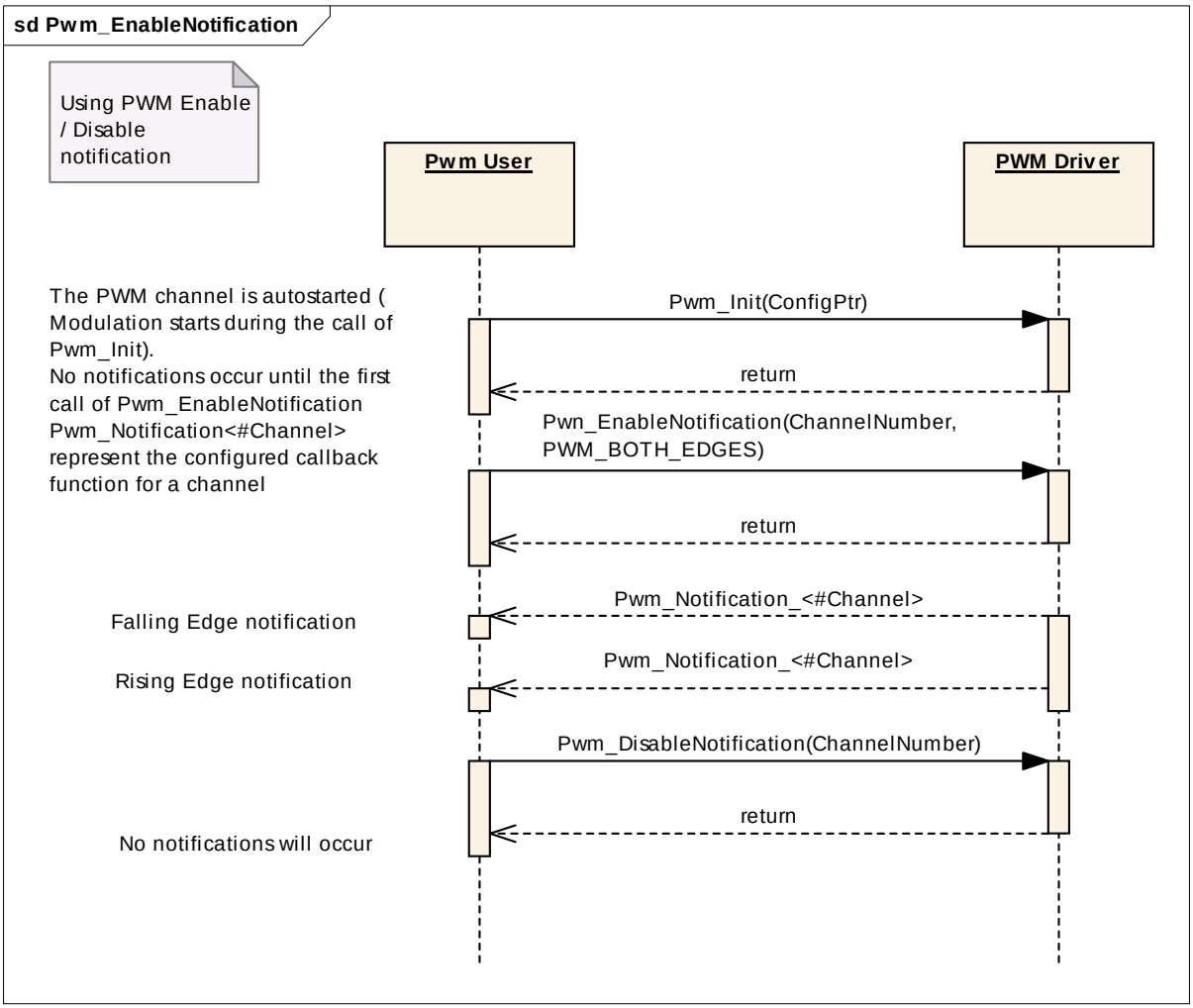


Figure 9: Using Pwm notifications

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
- AUTOSAR ECU Configuration Specification
  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 Configuration

SWS Item	PWM004:
Container Name	PwmModuleConfiguration
Description	Define the general PWM module configuration
Configuration Parameters	

Name	PWM_DEV_ERROR_DETECT		
Description	Preprocessor switch for enabling the development error detection.		
Type	#define		
Unit	--		
Range	ON	development error detection enabled	
	OFF	development error detection disabled	
Configuration Class	***Pre-compile***	X	All Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PWM_DUTYCYCLE_UPDATED_ENDPERIOD		
Description	Preprocessor switch for enabling the update of the duty cycle parameter at the end of the current period		
Type	#define		
Unit	--		
Range	ON	Feature supported	
	OFF	Feature not supported	
Configuration Class	***Pre-compile***	X	All Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PWM_DUTY_PERIOD_UPDATED_ENDPERIOD		
Description	Preprocessor switch for enabling the update of the duty cycle and the period parameter at the end of the current period		
Type	#define		
Unit	--		
Range	ON	Feature supported	
	OFF	Feature not supported	
Configuration Class	***Pre-compile***	X	All Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PWM_NOTIFICATION_SUPPORTED		
Description	Preprocessor switch to indicate that the notifications are supported		
Type	#define		
Unit	--		
Range	ON	Notifications are supported	
	OFF	Notifications are not supported	
	Pre-compile	X	All Variants
	Link time	--	--

	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PWM_VERSION_INFO_API		
Description	Preprocessor switch to indicate that the Pwm_ GetVersionInfo is supported		
Type	#define		
Unit	--		
Range	ON	Pwm_GetVersionInfo is supported	
	OFF	Pwm_GetVersionInfo is not supported	
Configuration Class	***Pre-compile***	X	All Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PWM_DE_INIT_API		
Description	Adds / removes the service Pwm_DeInit() from the code.		
Type	#define		
Unit	--		
Range	ON	Pwm_DeInit() can be used	
	OFF	Pwm_DeInit() can not be used	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PWM_SET_DUTY_CYCLE_API		
Description	Adds / removes the service Pwm_SetDutyCycle() from the code.		
Type	#define		
Unit	--		
Range	ON	Pwm_ SetDutyCycle () can be used	
	OFF	Pwm_ SetDutyCycle () can not be used	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PWM_SET_PERIOD_AND_DUTY_API		
Description	Adds / removes the service Pwm_ SetPeriodAndDuty () from the code.		
Type	#define		
Unit	--		
Range	ON	Pwm_SetPeriodAndDuty () can be used	
	OFF	Pwm_SetPeriodAndDuty () can not be used	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PWM_SET_OUTPUT_TO_IDLE_API

Description	Adds / removes the service Pwm_SetOutputToIdle () from the code.		
Type	#define		
Unit	--		
Range	ON	Pwm_SetOutputToIdle () can be used	
	OFF	Pwm_SetOutputToIdle () can not be used	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PWM_GET_OUTPUT_STATE_API		
Description	Adds / removes the service Pwm_GetOutputState () from the code.		
Type	#define		
Unit	--		
Range	ON	Pwm_GetOutputState () can be used	
	OFF	Pwm_GetOutputState () can not be used	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
PwmChannelConfiguration	1…*	ECU

10.2.3 PWM Channel Configuration

SWS Item	PWM027:
Container Name	PwmChannelConfiguration
Description	Define the PWM Channel configuration
Configuration Parameters	

Name	PWM_CHANNEL_ID		
Description	Channel Id of the PWM channel. This value will be assigned to the symbolic name derived of the PwmChannel container short name.		
Type	Integer (Symbolic Name generated for this parameter)		
Unit	--		
Range	--		
	--		
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	PWM_PERIOD_DEFAULT
Description	Value of period used for Initialization.
Type	Pwm_PeriodType
Unit	--

Range	8 bits	Implementation specific	
	32 bits	Implementation specific	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	ECU		
Dependency	None		

Name	PWM_DUTYCYLE_DEFAULT		
Description	Value of duty cycle used for initialization		
Type	`uint16`		
Unit	%		
Range	0x0	0%	
	`0x8000`	100%	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	ECU		
Dependency	None		

Name	PWM_POLARITY		
Description	Defines the starting polarity of each PWM channel		
Type	Pwm_OutputStateType		
Unit	--		
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	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	ECU		
Dependency	None		

Name	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).		
Type	Pwm_OutputStateType		
Unit	--		
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	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	ECU		
Dependency	None		

Name	PWM_CHANNEL_CLASS
Description	Class of PWM

Type	Pwm_ChannelClassType		
Unit	--		
Range	PWM_VARIABLE_PERIOD	Duty Cycle and period can be changed.	
	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)	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	ECU		
Dependency	None		

Name	Pwm_Notification_<#Channel>		
Description	Definition of the Callback function.		
Type	Function pointer		
Unit	--		
Range	NULL	No function pointer	
	Not NULL	Function to be called	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	ECU		
Dependency	None		

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.

SWS Item	PWM054:	
Information elements		
Information element name	***Type / Range***	***Information element description***

PWM_VENDOR_ID	uint16 / --	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
PWM_MODULE_ID	uint8 / --	Module ID of this module from Module List
PWM_AR_MAJOR_VERSION	uint8 / --	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
PWM_AR_MINOR_VERSION	uint8 / --	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
PWM_AR_PATCH_VERSION	uint8 / --	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
PWM_SW_MAJOR_VERSION	uint8 / --	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
PWM_SW_MINOR_VERSION	uint8 / --	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
PWM_SW_PATCH_VERSION	uint8 / --	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

11 Changes

11.1 Deleted SWS Items

SWS Item	Rationale
PWM028	Take in account the new template for configuration
PWM030	Take in account the new template for configuration and bug #5523
PWM042	Correction of bug #4007
PWM036	Remove time resolution from the published parameter
PWM055	Remove (due to SRS publication)
PWM056	Remove (due to SRS publication)
PWM057	Remove (due to SRS publication)
PWM015	Review comments of SVDO
PWM048	Correction of Bug #11802 and RFC #13823

11.2 Changed SWS Items

SWS Item	Rationale
PWM002	Take in account the new template sentence to describe requirement
PWM003	Take in account the new template sentence to describe requirement
PWM004	Take in account the new template for configuration
PWM015	Correct bugzilla #4171
PWM027	Take in account the new template for configuration and bug #4055
PWM029	Take in account the new template
PWM054	Implementation of Rfc 4060
PWM022	Correction of bug #4052
PWM021	Remove abstraction level #4962 and correct # 9607
PWM009	Remove abstraction level #4962
PWM025	Clarify the name of the notification
PWM011	Remove abstraction level #4962
PWM014	Remove abstraction level #4962
PWM015	Remove abstraction level #4962
PWM016	Remove abstraction level #4962
PWM051	Remove abstraction level #4962
PWM009	Review comments of SVDO
PWM014	Review comments of SVDO

11.3 Added SWS Items

SWS Item	Rationale
PWM064	Take in account the new template sentence to describe requirement
PWM065	Take in account the new template sentence to describe requirement
PWM066	Take in account the new template sentence to describe requirement
PWM067	Take in account the new template sentence to describe requirement
PWM068	Implementation of Pwm_GetVersionInfo
PWM069	Implementation of Pwm_GetVersionInfo
PWM070	Take in account configuration of time
PWM073	Take in account new SRS of Autosar Srs_General BSW00384
PWM074	Take in account new SRS of Autosar_Srs_General BSW00384
PWM075	Take in account new SRS of Autosar_Srs_General BSW00381, BSW00412, BSW00435, BSW00436

PWM076	Bugzilla 8485
PWM077	Variant definition
PWM078	Add this requirement to comply with chapter Error notification of the SWS template
PWM079	Add PC variant
PWM080	Bug #11767 RFC #13627: Configuration of optional API service “Pwm_DeInit()“
PWM081	Bug #11800 RFC #13633: Cancel pending interrupts
PWM082	Bug #14468 RFC #15495: Configuration of optional API service “Pwm_SetDutyCycle()“
PWM083	Bug #14468 RFC #15495: Configuration of optional API service “Pwm_SetPeriodAndDuty()“
PWM084	Bug #14468 RFC #15495: Configuration of optional API service “Pwm_SetOutputToIdle()“
PWM085	Bug #14468 RFC #15495: Configuration of optional API service “Pwm_GetOutputState()“