

PWM driver user guide

TRAVEO™ T2G family

About this document

Scope and purpose

This guide describes the architecture, configuration and use of the pulse width modulation (PWM) driver. It will help you to understand the functionality of the driver and will provide a reference to the driver's API.

The installation, build process, and general information about the use of EB tresos are not within the scope of this document.

Intended audience

This document is intended for anyone who uses the PWM driver of the TRAVEO™ T2G family.

Document structure

Chapter [1 General overview](#) gives a brief introduction to the PWM driver, explains the embedding in the AUTOSAR environment and describes the supported hardware and development environment.

Chapter [2 Using the PWM driver](#) details the steps required to use the PWM driver in your application.

Chapter [3 Structure and dependencies](#) describes the file structure and the dependencies for the PWM driver.

Chapter [4 EB tresos Studio configuration interface](#) describes the driver's configuration.

Chapter [5 Functional description](#) gives a functional description of all services offered by the PWM driver.

Chapter [6 Hardware resources](#) gives a description of all hardware resources used.

The [Appendix A](#) and [Appendix B](#) provides a complete API reference and access register table.

Abbreviations and definitions

Words and terms	Description
ADC	Analog Digital Converter
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ASIL	Automotive Safety Integrity Level
AUTOSAR	Automotive Open System Architecture
BSW	Basic Software. Standardized part of software which does not fulfill a vehicle functional job.
CDD	Complex Device Driver
DEM	Diagnostic Event Manager
DET	Default Error Tracer
EcuM	ECU State Manager
GCE	Generic Configuration Editor
IoHwAbs	Input/Output Hardware Abstraction

About this document

Words and terms	Description
ISR	Interrupt Service Routine
μC	Microcontroller
MCU	Microcontroller Unit
MCAL	Microcontroller Abstraction Layer
OS	Operating System
PWM	Pulse Width Modulation
TCPWM	Timer Counter Pulse Width Modulation
EB tresos Studio	Elektrobit Automotive configuration framework
UART	Universal Asynchronous Receiver-Transmitter
UTF-8	8-Bit Universal Character Set Transformation Format
Tick	Defines the timer resolution, the duration of a timer increment

Related documents**AUTOSAR requirements and specifications****Bibliography**

- [1] General specification of basic software modules, AUTOSAR release 4.2.2.
- [2] Specification of PWM driver, AUTOSAR release 4.2.2.
- [3] Specification of standard types, AUTOSAR release 4.2.2.
- [4] Specification of ECU configuration parameters, AUTOSAR release 4.2.2.
- [5] Specification of default error tracer, AUTOSAR release 4.2.2.

Elektrobit automotive documentation**Bibliography**

- [6] EB tresos Studio for ACG8 user's guide.

Hardware documentation**Bibliography**

- [7] The hardware documents are listed in the delivery notes.

Related standards and norms**Bibliography**

- [8] Layered software architecture, AUTOSAR release 4.2.2.

Table of contents

About this document.....	1
Table of contents.....	3
1 General overview	7
1.1 Introduction to the PWM driver	7
1.2 User profile	7
1.3 Embedding in the AUTOSAR environment.....	7
1.4 Supported hardware	8
1.5 Development environment.....	8
1.6 Character set and encoding.....	8
2 Using the PWM driver.....	9
2.1 Installation and prerequisites.....	9
2.2 Configuring the PWM driver	9
2.2.1 Architecture specifics.....	9
2.3 Adapting your application	9
2.4 Starting the build process.....	11
2.5 Measuring stack consumption.....	11
2.6 Memory mapping	11
2.6.1 Memory allocation keyword	12
3 Structure and dependencies.....	13
3.1 Static files	13
3.2 Configuration files	13
3.3 Generated files	13
3.4 Dependencies.....	14
3.4.1 PORT driver	14
3.4.2 MCU driver	14
3.4.3 AUTOSAR OS.....	14
3.4.4 DET.....	14
3.4.5 Error callout handler	14
3.4.6 BSW scheduler.....	14
4 EB tresos Studio configuration interface.....	15
4.1 General configuration	15
4.2 Standard parameters.....	15
4.2.1 Container PwmGeneral.....	15
4.2.1.1 PwmDevErrorDetect	15
4.2.1.2 PwmDutycycleUpdatedEndperiod.....	15
4.2.1.3 PwmIndex.....	16
4.2.1.4 PwmNotificationSupported	16
4.2.1.5 PwmPeriodUpdatedEndperiod	16
4.2.1.6 PwmLowPowerStatesSupport	16
4.2.1.7 PwmPowerStateAsynchTransitionMode	16
4.2.2 Container PwmChannel.....	17
4.2.2.1 PwmChannelClass	17
4.2.2.2 PwmChannelId.....	17
4.2.2.3 PwmDutycycleDefault	17
4.2.2.4 PwmIdleState.....	17
4.2.2.5 PwmMcuClockReferencePoint	18
4.2.2.6 PwmNotification	18

Table of contents

4.2.2.7	PwmPeriodDefault.....	18
4.2.2.8	PwmPolarity.....	18
4.2.3	Container PwmConfigurationOfOptApiServices.....	19
4.2.3.1	PwmDelInitApi	19
4.2.3.2	PwmGetOutputState	19
4.2.3.3	PwmSetDutyCycle.....	19
4.2.3.4	PwmSetOutputToldle	19
4.2.3.5	PwmSetPeriodAndDuty	19
4.2.3.6	PwmVersionInfoApi.....	20
4.2.4	Container PwmPowerStateConfig	20
4.2.4.1	PwmPowerState	20
4.2.4.2	PwmPowerStateReadyCbRef	20
4.3	Vendor and driver specific parameters	20
4.3.1	Container PwmGeneral.....	20
4.3.1.1	PwmHwTriggerOutputUnitType	20
4.3.1.2	PwmMultipleUpdateInPeriod.....	21
4.3.1.3	PwmErrorCalloutFunction.....	21
4.3.1.4	PwmIncludeFile	22
4.3.1.5	PwmStartTriggerSelect0.....	22
4.3.1.6	PwmStartTriggerSelect1.....	23
4.3.2	Container PwmChannel.....	24
4.3.2.1	PwmTimer	24
4.3.2.2	PwmAlignment.....	24
4.3.2.3	PwmHwTriggerOutputLine.....	25
4.3.2.4	PwmHwTriggerOutputFactor	25
4.3.2.5	PwmChannelStartDelay.....	26
4.3.2.6	PwmChannelPrescale	26
4.3.2.7	PwmStartDelayTimer.....	26
4.3.2.8	PwmStartDelayMcuClockReferencePoint.....	27
4.3.2.9	PwmStartDelayTrigger	27
4.3.2.10	PwmHwTriggerOutputDefaultTime	28
4.3.2.11	PwmHwTriggerOutputDefaultTick.....	29
4.3.2.12	PwmHwTriggerOutputScaled	29
4.3.2.13	PwmSetOutputEnable	29
4.3.2.14	PwmOutputOffset.....	30
4.3.2.15	PwmOutputOffsetScaled	30
4.3.2.16	PwmStartAtInit.....	31
4.3.2.17	PwmDebugMode.....	31
4.3.2.18	PwmUpdateOutputAtInitEnable	32
4.3.2.19	PwmOutputAtInitSelect.....	32
4.3.2.20	PwmCompOutputAtInitSelect.....	32
4.3.3	Container PwmChannelGroup.....	33
4.3.3.1	PwmChannelGroupId	33
4.3.3.2	PwmChannelGroupStartTrigger.....	33
4.3.3.3	PwmChannelGroupStopTrigger	34
4.3.3.4	PwmChannelRef.....	36
4.3.3.5	PwmChannelGroupSwitchEventTrigger	36
4.3.4	Container PwmGroupStartDelay.....	37
4.3.4.1	PwmGroupStartDelayTimer	37
4.3.4.2	PwmGroupStartDelayMcuClockReferencePoint	38
4.3.4.3	PwmGroupStartDelayTrigger	38

Table of contents

4.3.5	Container PwmConfigurationOfOptApiServices.....	39
4.3.5.1	PwmSetOutputState.....	39
4.3.5.2	PwmSetPrescaler	39
4.3.5.3	PwmSetTriggerDelay	39
4.3.5.4	PwmStopGroup.....	39
4.3.5.5	PwmSetOutputOffset.....	40
4.3.5.6	PwmEnableDisableTriggerApi.....	40
4.3.5.7	PwmSetDutyCycleBufferApi	40
4.3.5.8	PwmSetDutyAndChannelOutputBufferApi.....	40
4.3.5.9	PwmSetChannelOutputBufferApi	40
5	Functional description	41
5.1	Initialization.....	41
5.2	Runtime reconfiguration.....	41
5.3	PWM channel	41
5.4	PWM channel group	41
5.5	Notification	41
5.6	Stopping a channel	42
5.7	Starting a channel	42
5.8	Updating a channel duty and period.....	42
5.9	Delaying PWM output.....	42
5.10	Triggering input to a channel group.....	43
5.11	Output trigger to peripherals.....	43
5.12	Prescaler setting function	44
5.13	Retrieving status of a channel.....	44
5.14	Sleep mode.....	44
5.15	API parameter checking.....	44
5.16	Configuration checking.....	46
5.17	Reentrancy.....	46
5.18	Debugging support.....	46
5.19	Execution time dependencies	47
6	Hardware resources	48
6.1	Ports and pins.....	48
6.2	Timer	48
6.3	Interrupts.....	48
6.4	Triggers	49
7	Appendix A – API reference	51
7.1	Include files.....	51
7.2	Data types	51
7.2.1	Pwm_ChannelType	51
7.2.2	Pwm_PeriodType.....	51
7.2.3	Pwm_OutputStateType	51
7.2.4	Pwm_EdgeNotificationType.....	52
7.2.5	Pwm_ClkFrequencyType	52
7.2.6	Pwm_ConfigType	52
7.2.7	Pwm_ChannelGroupType.....	52
7.2.8	Pwm_DriverStatusType	53
7.2.9	Pwm_ChannelStatusType	53
7.2.10	Pwm_OutputLineStateType	54
7.3	Constants.....	54
7.3.1	Error codes	54

Table of contents

7.3.2	Version information	55
7.3.3	Module information	55
7.3.4	API service IDs	55
7.3.5	Symbolic names	56
7.3.6	Channel status values	57
7.4	Functions	57
7.4.1	Pwm_Init	57
7.4.2	Pwm_DeInit	58
7.4.3	Pwm_SetDutyCycle	58
7.4.4	Pwm_SetPeriodAndDuty	59
7.4.5	Pwm_SetOutputTolde	61
7.4.6	Pwm_GetOutputState	61
7.4.7	Pwm_DisableNotification	62
7.4.8	Pwm_EnableNotification	63
7.4.9	Pwm_GetVersionInfo	64
7.4.10	Pwm_GetChannelStatus	64
7.4.11	Pwm_StartGroupTrigger	65
7.4.12	Pwm_StopGroupTrigger	66
7.4.13	Pwm_SetChannelOutput	67
7.4.14	Pwm_SetPrescaler	68
7.4.15	Pwm_SetOutputOffset	69
7.4.16	Pwm_SetTriggerDelay	70
7.4.17	Pwm_EnableTrigger	71
7.4.18	Pwm_DisableTrigger	72
7.4.19	Pwm_SetDutyCycleBuffer	73
7.4.20	Pwm_SetChannelOutputBuffer	74
7.4.21	Pwm_SetDutyAndChannelOutputBuffer	75
7.5	Required callback functions	76
7.5.1	DET	76
7.5.2	DEM	76
7.5.3	PWM notifications	77
7.5.4	Callout functions	77
8	Appendix B – Access register table	79
8.1	TCPWM	79
	Revision history	86

1 General overview

1.1 Introduction to the PWM driver

The PWM driver is a set of software routines, which enables you to create PWM signals on dedicated output pins of the CPU.

For this purpose, the PWM driver provides services for modifying the period and the duty cycle of a PWM signal. In addition, the PWM driver provides services for initializing and deinitializing the PWM module, setting of the output to idle level, and enabling or disabling of a notification callback function on a specified edge.

The PWM driver is not responsible for initializing or configuring hardware ports. This task is performed by the PORT driver.

The driver conforms to the AUTOSAR standard and is implemented according to the *specification of PWM driver* [2].

The PWM driver is delivered with a plugin for the EB tresos Studio, which allows you to statically configure the driver. The driver provides an interface to enable PWM channels and to configure period, duty cycle and other parameters.

1.2 User profile

This guide is intended for users with a least basic knowledge of the following:

- Automotive embedded systems
- C programming language
- AUTOSAR standard
- Target hardware architecture

1.3 Embedding in the AUTOSAR environment

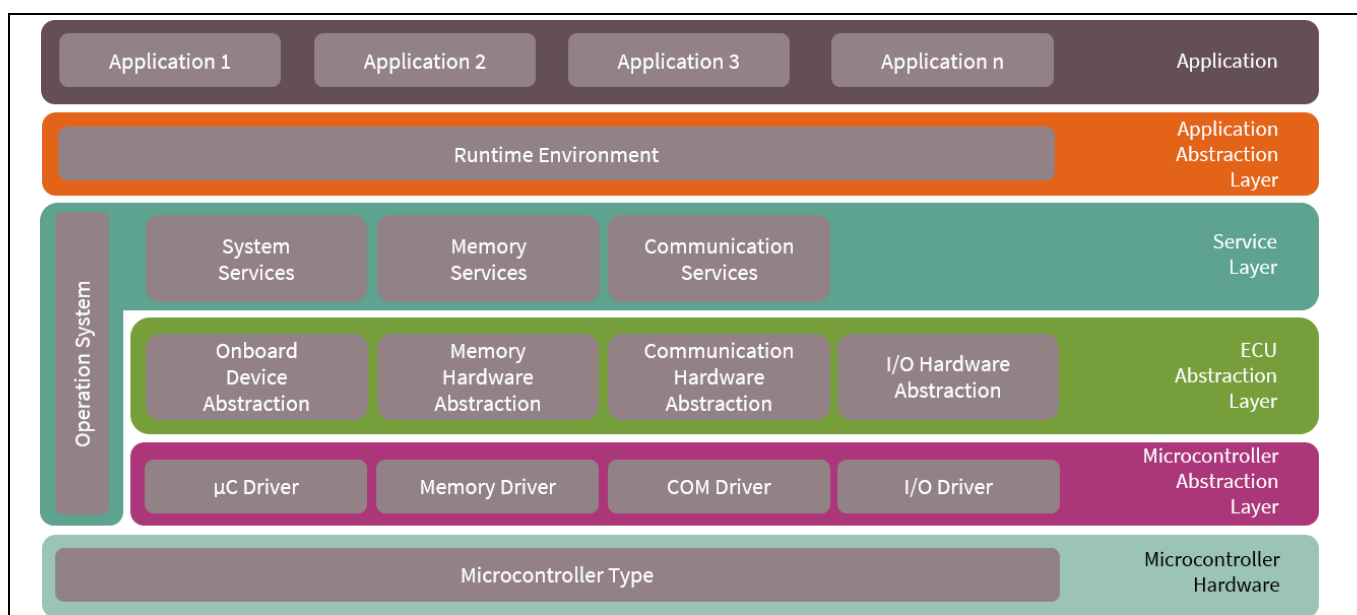


Figure 1 Overview of AUTOSAR software layers

General overview

Figure 1 depicts the layered AUTOSAR software architecture. The PWM driver (**Figure 2**) is part of group of I/O drivers within the microcontroller abstraction layer (MCAL), the lowest layer of basic software in the AUTOSAR environment.

For an exact overview of the AUTOSAR layered software architecture, see the *layered software architecture* [8].

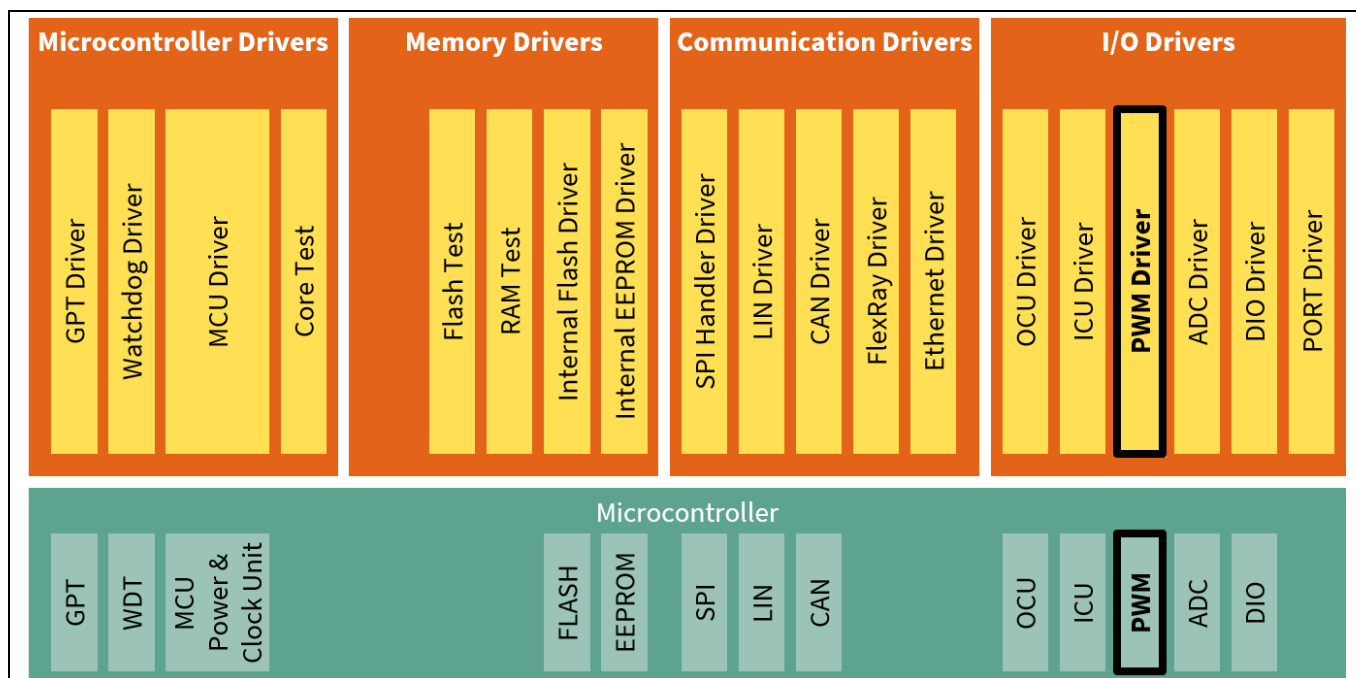


Figure 2 PWM driver in MCAL layer

1.4 Supported hardware

This version of the PWM driver supports the TRAVEO™ T2G family. No special external hardware devices are required.

The supported derivatives are listed in the release notes.

1.5 Development environment

The development environment corresponds to AUTOSAR release 4.2.2. The modules BASE, MAKE, MCU, PORT and RESOURCE are needed for proper functionality of the PWM driver.

1.6 Character set and encoding

All source code files of the PWM driver are restricted to the ASCII character set. The files are encoded in UTF-8 format, with only the 7-bit subset (values 0x00 ... 0x7F) being used.

2 Using the PWM driver

This chapter describes all necessary steps to incorporate the PWM driver into your application.

2.1 Installation and prerequisites

Note: Before you start, see the *EB tresos Studio for ACG8 user's guide* [6] for the following information.

1. The installation procedure of EB tresos ECU AUTOSAR components
2. The usage of the EB tresos Studio
3. The usage of the EB tresos ECU AUTOSAR build environment (It includes the steps to setup and integrate the own application within the EB tresos ECU AUTOSAR build environment)

The installation of the PWM driver corresponds with the general installation procedure for EB tresos AUTOSAR components given in the documents mentioned above. If the driver has been successfully installed, the driver will appear in the module list of the EB tresos Studio (see the *EB tresos Studio for ACG8 user's guide* [6]).

This document assumes that the project is properly set up and is using the application template as described in the *EB tresos Studio for ACG8 user's guide* [6]. This template provides the necessary folder structure, project and makefiles needed to configure and compile your application within the build environment. You also must be familiar with the use of the command shell.

2.2 Configuring the PWM driver

The PWM driver can be configured with any AUTOSAR compliant GCE tool. Save the configuration in a separate file named for example, *Pwm.epc*. More information about the PWM driver configuration can be found in [EB tresos Studio configuration interface](#).

2.2.1 Architecture specifics

See [Vendor and driver specific parameters](#), where all vendor- and driver-specific configuration parameters are described.

2.3 Adapting your application

To use the PWM driver in your application, you must include the header files of PWM, MCU, and PORT driver by adding the following lines of code to your source file:

```
#include "Mcu.h"
#include "Port.h"
#include "Pwm.h"
```

This publishes all needed function and data prototypes and symbolic names of the configuration into the application. In addition, you must implement the error callout function for ASIL safety extension.

To use the PWM driver, the appropriate port pins and PWM interrupts must be configured in MCU driver, PORT driver, and OS. For detailed information see [Hardware resources](#).

Using the PWM driver

Initialization of MCU, PORT, and PWM driver needs to be done in the following order:

```
Mcu_Init(&Mcu_Config[0]);
Port_Init(&Port_Config[0]);
Pwm_Init(&Pwm_Config[0]);
```

The function `Mcu_Init()` is called with a pointer to a structure of type `Mcu_ConfigType`, which is published by the MCU driver itself.

The function `Port_Init()` is called with a pointer to a structure of type `Port_ConfigType`, which is published by the PORT driver itself.

The function `Pwm_Init()` is called with a pointer to a structure of type `Pwm_ConfigType`, which is published by the PWM driver itself.

After initialization of the PWM driver, its API functions can be used. Each configured PWM channel operates, but no callback functions are called until you enable the functions.

`PwmStartTriggerSelect0` specifies the input trigger of TCPWM instance 0 to start all PWM channels synchronously except `PwmStartAtInit` is set to FALSE. When this parameter is configured, a trigger signal is required to start those channels. It is necessary to call `Port_ActTrigger()` to issue a trigger signal after calling `Pwm_Init()`. In this case, the group trigger configuration is also required in the PORT module. When this parameter is not configured, channels are started sequentially by `Pwm_Init()`.

`PwmStartTriggerSelect1` specifies the input trigger of TCPWM instance 1 to start all PWM channels synchronously except `PwmStartAtInit` is set to FALSE. When this parameter is configured, a trigger signal is required to start those channels. It is necessary to call `Port_ActTrigger()` to issue a trigger signal after calling `Pwm_Init()`. In this case, the group trigger configuration is also required in the PORT module. When this parameter is not configured, channels are started sequentially by `Pwm_Init()`.

`PwmChannelGroupStartTrigger` specifies the input trigger to start the PWM channel group synchronously. When this parameter is configured, a trigger signal is required to start all channels in the group. It is necessary to call `Port_ActTrigger()` to issue a trigger signal after calling `Pwm_StartGroupTrigger()`. In this case, the group trigger configuration is also required in the PORT module. When this parameter is not configured, the channels in the group are started sequentially by `Pwm_StartGroupTrigger()`.

`PwmChannelGroupStopTrigger` specifies the input trigger to stop the PWM channel group synchronously. When this parameter is configured, a trigger signal is required to stop all channels in the group. It is necessary to call `Port_ActTrigger()` to issue a trigger signal after calling `Pwm_StopGroupTrigger()`. In this case, the group trigger configuration is also required in PORT module. When this parameter is not configured, the channels in the group are stopped sequentially by `Pwm_StopGroupTrigger()`.

For a configured channel with the name `MY_PWM_CHANNEL` (`PwmChannelClass = PWM_VARIABLE_PERIOD`, `PwmPolarity = PWM_HIGH`, `PwmIdleState = PWM_LOW`), following functions can be called:

```
/* set 50% duty */
Pwm_SetDutyCycle(MY_PWM_CHANNEL, 0x4000);
/* period = 0x6000 ticks with 25% duty */
Pwm_SetPeriodAndDuty(MY_PWM_CHANNEL, 0x6000, 0x2000);
/* force polarity level = constant output = HIGH = 100% duty */
Pwm_SetDutyCycle(MY_PWM_CHANNEL, 0x8000);
/* set output into idle state = LOW */
```

Using the PWM driver

```
Pwm_SetOutputToIdle(MY_PWM_CHANNEL);  
/* start PWM channel again after it was idle */  
Pwm_SetDutyCycle(MY_PWM_CHANNEL, 0x0815);
```

2.4 Starting the build process

Do the following to build your application:

Note: For a clean build, use the build command with target `clean_all`. before (make `clean_all`).

1. Type the following in the command shell to generate the necessary configuration dependent files:

```
> make generate
```

2. The details of the generated files are described in [Generated files](#).
3. Type the following command to resolve required file dependencies:

```
> make depend
```

4. Type the following command to compile and link the application:

```
> make (optional target: all)
```

The application is now built. All files are compiled and linked to a binary file, which can be downloaded to the target hardware.

2.5 Measuring stack consumption

Do the following to measure stack consumption. It requires the Base module for proper measurement.

Note: All files (including library files) should be rebuilt with dedicated compiler option. The executable file built in this step must be used only to measure stack consumption.

To measure stack consumption:

1. Add the following compiler option to the Makefile to enable stack consumption measurement.

```
-DSTACK_ANALYSIS_ENABLE
```

2. Type the following command to clean library files.

```
> make clean_lib
```

3. Follow the build process described in [Starting the build process](#).
4. Follow the instructions in the release notes and measure the stack consumption.

2.6 Memory mapping

The `Pwm_MemMap.h` file in the `$(TRESOS_BASE)/plugins/MemMap_TS_T40D13M0I0R0/include` directory is a sample. This file is replaced by the file generated by MEMMAP module. Input to MEMMAP module is generated as `Pwm_Bswmd.arxml` in the `$(PROJECT_ROOT)/output/generated/swcd` directory of your project folder.

2.6.1 Memory allocation keyword

- `PWM_START_SEC_CODE_ASIL_B / PWM_STOP_SEC_CODE_ASIL_B`

The memory section type is CODE. All executable code is allocated in this section.

- `PWM_START_SEC_CONST_ASIL_B_UNSPECIFIED / PWM_STOP_SEC_CONST_ASIL_B_UNSPECIFIED`

The memory section type is CONST. The following constants are allocated in this section:

- PWM channel configuration setting
 - PWM group configuration setting
 - PWM whole configuration setting
 - Lookup table to search a specific PWM channel by logical channel index
- `PWM_START_SEC_VAR_INIT_ASIL_B_UNSPECIFIED / PWM_STOP_SEC_VAR_INIT_ASIL_B_UNSPECIFIED`

The memory section type is VAR. The following constants are allocated in this section:

- Pointer to whole configuration setting
 - Information for PWM driver state
- `PWM_START_SEC_VAR_NO_INIT_ASIL_B_UNSPECIFIED / PWM_STOP_SEC_VAR_NO_INIT_ASIL_B_UNSPECIFIED`

The memory section type is VAR. The following constants are allocated in this section:

- Information for target PWM channel state

3 Structure and dependencies

The PWM driver consists of static, configuration, and generated files.

3.1 Static files

- $\$(PLUGIN_PATH)=\$(TRESOS_BASE)/plugins/Pwm_TS_*$ is the path to the PWM driver plugin.
- $\$(PLUGIN_PATH)/lib_src$ contains all static source files of the PWM driver. These files contain the functionality of the driver, which does not depend on the current configuration. The files are grouped into a static library.
- $\$(PLUGIN_PATH)/src$ comprises configuration dependent source files or special derivative files. Each file will be rebuilt when the configuration is changed.

All necessary source files will automatically be compiled and linked during the build process and all include paths will be set if the PWM driver is enabled.

- $\$(PLUGIN_PATH)/include$ is the basic public include directory needed by the user and should be included in *Pwm.h*.
- $\$(PLUGIN_PATH)/autosar$ directory contains the AUTOSAR ECU parameter definition with vendor, architecture, and derivative specific adaptations to create a correct matching parameter configuration for the PWM driver.

3.2 Configuration files

The configuration of the PWM driver is done via EB tresos Studio. The file containing the PWM driver's configuration is named *Pwm.xdm* and is in the directory $\$(PROJECT_ROOT)/config$. This file serves as input for the generation of the configuration dependent source and header files during the build process.

3.3 Generated files

During the build process the following files are generated based on the current configuration description. They are in the *output/generated* sub folder of your project folder:

- *include/Pwm_Cfg.h*
- *include/Pwm_Cfg_Include.h*
- *include/Pwm_PBcfg.h*
- *src/Pwm_Irq.c*
- *src/Pwm_PBcfg.c*

Note: *Generated source files need not to be added to your application make file. These files will be compiled and linked automatically during the build process.*

- *swcd/Pwm_Bswmd.arxml*

Note: *Additional steps are required for the generation of BSW module description. In EB tresos Studio, follow the menu path **Project > Build Project** and click **generate_swcd**.*

3.4 Dependencies

3.4.1 PORT driver

Although the PWM driver can successfully be compiled and linked without an AUTOSAR compliant PORT driver, the latter is required to configure and initialize all ports. Also, the configuration of triggers is required and it is necessary to call `Port_ActTrigger()` to issue a trigger signal after either of the following case:

- Calling `Pwm_Init()` if the `PwmStartTriggerSelect0` or `PwmStartTriggerSelect1` parameter is configured.
- Calling `Pwm_StartGroupTrigger()` if the `PwmChannelGroupStartTrigger` parameter is configured.
- Calling `Pwm_StopGroupTrigger()` if the `PwmChannelGroupStopTrigger` parameter is configured.

If `Port_ActTrigger()` is not called, channels or channels in the group are not started/stopped. The PORT driver needs to be initialized before the PWM driver is initialized. See the *PORT driver's user guide* for details.

The configuration of triggers is required in next case:

- Calling `Pwm_Init()` if the `PwmStartDelayTrigger` parameter is configured.

If `Port_ActTrigger()` is called with the `PwmStartDelayTrigger`, the PWM driver will show undefined behavior.

3.4.2 MCU driver

The MCU driver needs to be initialized and all MCU clock reference points referenced by the MCU driver channels via configuration parameter `PwmMcuClockReferencePoint` must have been activated (via calls of MCU API functions) before initializing the MCU driver. See the *MCU driver's user guide* for details.

3.4.3 AUTOSAR OS

The OS must be used to configure and create the ISR vector table entries for the PWM used channels. See the hardware and derivatives specified in [Interrupts](#).

3.4.4 DET

If the development error detection is enabled in the PWM driver configuration, the DET module needs to be installed, configured and integrated into the application also.

3.4.5 Error callout handler

The error callout handler is called on every error that is detected, regardless of whether development error detection is enabled or disabled. The error callout handler is an ASIL safety extension that is not specified by AUTOSAR. It is configured via the configuration parameter `PwmErrorCalloutFunction`.

3.4.6 BSW scheduler

The PWM driver uses the following services of the BSW scheduler to enter and leave critical sections:

- `SchM_Enter_Pwm_PWM_EXCLUSIVE_AREA_0(void)`
- `SchM_Exit_Pwm_PWM_EXCLUSIVE_AREA_0(void)`

You must ensure that the BSW scheduler is properly configured and initialized before using the PWM driver.

You must ensure that all interrupts do not occur during enter critical section.

4 EB tresos Studio configuration interface

Use an AUTOSAR compliant GCE to adapt the configuration files as desired.

Note: The ECU parameter description of the PWM driver basically corresponds to the one defined by AUTOSAR. However, as there are some vendor specific extensions, you must use the ECU parameter description file that is delivered with the PWM driver. There are two types of the file:

- The EB tresos Studio configuration description
(`$(TRESOS_BASE)/plugins/Pwm_TS_<VARIANT>/config/Pwm.xdm`): This file is automatically loaded when the module is selected in EB tresos. The file contains specific support for calculated values and error checks.
- The AUTOSAR compatible configuration description
(`$(TRESOS_BASE)/plugins/Pwm_TS_<VARIANT>/autosar/Pwm.arxml`): This file can be used as a basis for GCEs other than Tresos. The file does not contain specific support for calculated values and error checks, but only general ranges that cover all possible values for all derivatives.

4.1 General configuration

The module comes preconfigured with default settings. These settings must be adapted if necessary.

4.2 Standard parameters

This section lists all configuration parameters required by AUTOSAR. Deviations or hardware specific restrictions from the specification are mentioned where applicable.

4.2.1 Container PwmGeneral

4.2.1.1 PwmDevErrorDetect

Description

Enables or disables development error notification for the PWM driver.

Remarks

Setting this parameter to FALSE will disable the notification of development errors via DET. However, in contrast to AUTOSAR specification, detection of development errors is still enabled and errors will be reported via `PwmErrorCalloutFunction`.

4.2.1.2 PwmDutycycleUpdatedEndperiod

Description

Switch for enabling the update of the duty cycle parameter at the end of the current period.

Remarks

- `TRUE`: Duty cycle is updated at the end of the period of the currently generated waveform (current waveform is finished).
- `FALSE`: Duty cycle is updated immediately (just after service call, current waveform is cut).

4.2.1.3 PwmIndex

Description

Specifies the `InstanceId` of this module instance. If only one instance is present, the `Id` will be 0.

Remarks

This parameter is not used by the PWM driver and therefore not being evaluated.

4.2.1.4 PwmNotificationSupported

Description

Adds or removes the services `Pwm_EnableNotification()` and `Pwm_DisableNotification()` from the code.

Remarks

None

4.2.1.5 PwmPeriodUpdatedEndperiod

Description

Switch for enabling the update of the period parameter at the end of the current period.

Remarks

- `TRUE`: Period or duty cycle is updated at the end of the period of the currently generated waveform (current waveform is finished).
- `FALSE`: Period or duty cycle is updated immediately (just after service call, current waveform is cut).

4.2.1.6 PwmLowPowerStatesSupport

Description

Adds or removes all power state management related APIs (`Pwm_SetPowerState`, `Pwm_GetCurrentPowerState`, `Pwm_GetTargetPowerState`, `Pwm_PreparePowerState`, `Pwm_Main_PowerTransitionManager`), indicating whether the HW offers low power state management.

Remarks

The HW does not offer low power state management, so `PwmLowPowerStatesSupport` is always set to `FALSE`.

4.2.1.7 PwmPowerStateAsynchTransitionMode

Description

Enables or disables PWM driver's support to the asynchronous power state transition.

Remarks

The parameter `PwmLowPowerStatesSupport` is always set to `FALSE`, so this parameter will not be configured.

4.2.2 Container PwmChannel

4.2.2.1 PwmChannelClass

Description

Class of PWM channel: `PWM_FIXED_PERIOD`, `PWM_FIXED_PERIOD_SHIFTED` or `PWM_VARIABLE_PERIOD`.

Remarks

- `PWM_FIXED_PERIOD`: `PwmChannelStartDelay` is disabled. Calling `Pwm_SetPeriodAndDuty()` becomes an error of `PWM_E_PERIOD_UNCHANGEABLE`.
- `PWM_FIXED_PERIOD_SHIFTED`: `PwmChannelStartDelay` is enabled. Calling `Pwm_SetPeriodAndDuty()` becomes an error of `PWM_E_PERIOD_UNCHANGEABLE`.
- `PWM_VARIABLE_PERIOD`: `PwmChannelStartDelay` is enabled. Calling `Pwm_SetPeriodAndDuty()` is effective.

4.2.2.2 PwmChannelId

Description

Channel Id of the PWM channel.

Range

0..[65535 or number of configured channels - 1, whichever is lower]

Value must be unique among all channels.

Remarks

This value is assigned to a symbolic name. Use only the symbolic channel names defined in *Pwm_Cfg.h* for API calls.

4.2.2.3 PwmDutycycleDefault

Description

Value of duty cycle used at initialization.

Remarks

Valid range: 0 (0% duty). 0x8000 (100% duty).

4.2.2.4 PwmIdleState

Description

Specifies the output state of the PWM after the signal is stopped (e.g. by call of `Pwm_SetOutputToIdle`): `PWM_HIGH` or `PWM_LOW`.

Remarks

None

4.2.2.5 PwmMcuClockReferencePoint

Description

This parameter contains reference to `McuClockReferencePoint`.

Remarks

`PwmMcuClockReferencePoint` must have the target's `McuClockReferencePoint` setting.

4.2.2.6 PwmNotification

Description

Definition of the notification callback function.

Remarks

This parameter can be configured as empty string, or NULL can be entered. Both will result in no notification being called. Otherwise, the function name must be unique.

4.2.2.7 PwmPeriodDefault

Description

Period value used at initialization in seconds.

Range

Depends on `PwmMcuClockReferencePoint`.

Remarks

A symbolic name for the tick value of `PwmPeriodDefault` is generated. This symbolic name is defined in *Pwm_PBcfg.h*. The tick value helps to calculate a new period tick value in relation with the existing `PwmPeriodDefault` for usage in `Pwm_SetPeriodAndDuty()`.

`PwmPeriodDefault` must be equal to or more than $1 / (\text{McuClockReferencePointFrequency of PwmMcuClockReferencePoint})$ and less than $1 / (\text{McuClockReferencePointFrequency of PwmMcuClockReferencePoint}) * 128 * 65535$.

4.2.2.8 PwmPolarity

Description

Specifies the polarity of each PWM channel: `PWM_HIGH` or `PWM_LOW`.

Remarks

None

4.2.3 Container PwmConfigurationOfOptApiServices

4.2.3.1 PwmDeInitApi

Description

Adds or removes the service `Pwm_DeInit()` from the code.

Remarks

None

4.2.3.2 PwmGetOutputState

Description

Adds or removes the service `Pwm_GetOutputState()` from the code.

Remarks

None

4.2.3.3 PwmSetDutyCycle

Description

Adds or removes the service `Pwm_SetDutyCycle()` from the code.

Remarks

None

4.2.3.4 PwmSetOutputToIdle

Description

Adds or removes the service `Pwm_SetOutputToIdle()` from the code.

Remarks

None

4.2.3.5 PwmSetPeriodAndDuty

Description

Adds or removes the service `Pwm_SetPeriodAndDuty()` from the code.

Remarks

None

4.2.3.6 PwmVersionInfoApi

Description

Adds or removes the service `Pwm_GetVersionInfo()` from the code.

Remarks

None

4.2.4 Container PwmPowerStateConfig

4.2.4.1 PwmPowerState

Description

Each instance of this parameter describes a different power state supported by the PWM HW. It should be defined by the HW supplier and used by PWM driver to reference specific HW configurations which set the PWM HW module in the referenced power state. At least the power mode corresponding to full power state will be always configured.

Remarks

The parameter `PwmLowPowerStatesSupport` is always set to `FALSE`, so this parameter will not be configured.

4.2.4.2 PwmPowerStateReadyCbRef

Description

Each instance of this parameter contains a reference to a power mode callback defined in a CDD or IoHwAbs component.

Remarks

The parameter `PwmLowPowerStatesSupport` is always set to `FALSE`, so this parameter will not be configured.

4.3 Vendor and driver specific parameters

4.3.1 Container PwmGeneral

4.3.1.1 PwmHwTriggerOutputUnitType

Description

Switches a unit type to second or tick.

Type

`EcucEnumerationParamDef`

Range

`PWM_TRIGGER_TICK`: `PwmHwTriggerOutputDefaultTick` is available.

`PWM_TRIGGER_SECOND`: `PwmHwTriggerOutputDefaultTime` is available.

4.3.1.2 PwmMultipleUpdateInPeriod

Description

Switch for enabling multiple updates within a period. `PwmMultipleUpdateInPeriod` specifies the behavior when the following functions are called while the channel is waiting for an update by the next period (`PWM_CH_COND_WAIT_UPDATE`).

`Pwm_SetPeriodAndDuty`, `Pwm_SetDutyCycle`, `Pwm_SetTriggerDelay`, `Pwm_SetOutputOffset`

Type

`EcucBooleanParamDef`

Range

- `TRUE`: The function overwrites the register with the new value. PWM channel continues the output waveform. There is a risk of abnormal waveform output if API is continuously called at intervals shorter than one period.
- `FALSE`: The PWM channel stops and restarts with the new value.

Remarks

In the following cases, the channel transitions to the wait for update state (`PWM_CH_COND_WAIT_UPDATE`) until the next period:

- Calling `Pwm_SetTriggerDelay`, `Pwm_SetOutputOffset`, and `Pwm_SetChannelOutput`.
- Calling `Pwm_Init` or `Pwm_StartGroupTrigger` with any start delay configured channel.
- Calling `Pwm_SetPeriodAndDuty` or `Pwm_SetDutyCycle` when `PwmPeriodUpdatedEndperiod` is `TRUE`.

4.3.1.3 PwmErrorCalloutFunction

Description

Error callout function.

Syntax:

```
void ErrorCalloutHandler
(
    uint16 ModuleId,
    uint8 InstanceId,
    uint8 ApiId,
    uint8 ErrorId
);
```

An error callout function is called on every error. The ASIL level of this function limits the ASIL level of the PWM driver.

Type

`EcucFunctionNameDef`

Remarks

`PwmErrorCalloutFunction` must be a valid C function name.

4.3.1.4 PwmIncludeFile

Description

Lists the filenames that will be included within the driver. Any application specific symbol that is used by the PWM configuration (e.g. error callout function) should be included by configuring this parameter.

Type

EcucStringParamDef

Remarks

PwmIncludeFile must be an unique filename with an extension *.h*.

4.3.1.5 PwmStartTriggerSelect0

Description

Specifies the input trigger of TCPWM instance 0 to start all PWM channels synchronously except PwmStartAtInit, which is set to FALSE. When this parameter is configured, a trigger signal is required to start those channels. It is necessary to call Port_ActTrigger() to issue a trigger signal after calling Pwm_Init(). In this case, the group trigger configuration is also required in the PORT module. When this parameter is not configured, channels are started sequentially by Pwm_Init().

Type

EcucEnumerationParamDef

Range

- TCPWM_0_TR_ALL_CNT_IN_0: Input trigger 0 of TCPWM instance 0
- TCPWM_0_TR_ALL_CNT_IN_1: Input trigger 1 of TCPWM instance 1
- ...
- TCPWM_0_TR_ALL_CNT_IN_n: Input trigger n of TCPWM instance 0

Note: The number of available input triggers depends on the hardware resources.

Remarks

PWM, ICU, GPT, and OCU drivers use input triggers of TCPWM.

An error occurs, if the value of the PwmStartTriggerSelect0 parameter is configured same, as the following parameters:

- PwmGroupStartDelayTrigger
- PwmStartDelayTrigger
- PwmChannelGroupStartTrigger
- PwmChannelGroupStopTrigger
- PwmChannelGroupSwitchEventTrigger
- IcuInputTriggerSelection
- IcuChannelGroupStartTrigger
- IcuChannelGroupStopTrigger
- GptPredefTimerStartTriggerSelect
- GptInputTriggerSelection

If the value of `PwmStartTriggerSelect0` parameter is configured as same the value of followings parameter, a warning occurs.

- `OcuStartTriggerSelect0`

4.3.1.6 PwmStartTriggerSelect1

Description

Specifies the input trigger of TCPWM instance 1 to start all PWM channels synchronously except `PwmStartAtInit`, which is set to `FALSE`. When this parameter is configured, a trigger signal is required to start those channels. It is necessary to call `Port_ActTrigger()` to issue a trigger signal after calling `Pwm_Init()`. In this case, the group trigger configuration is also required in the PORT module. When this parameter is not configured, channels are started sequentially by `Pwm_Init()`.

Type

`EcucEnumerationParamDef`

Range

- `TCPWM_1_TR_ALL_CNT_IN_0`: Input trigger 0 of TCPWM instance 1
- `TCPWM_1_TR_ALL_CNT_IN_1`: Input trigger 1 of TCPWM instance 1
- ...
- `TCPWM_1_TR_ALL_CNT_IN_n`: Input trigger n of TCPWM instance 1

Note: The number of available input triggers depends on the hardware resources.

Remarks

PWM, ICU, GPT, and OCU drivers use input triggers of TCPWM.

An error occurs, if the value of the `PwmStartTriggerSelect1` parameter is configured same, as the following parameters:

- `PwmGroupStartDelayTrigger`
- `PwmStartDelayTrigger`
- `PwmChannelGroupStartTrigger`
- `PwmChannelGroupStopTrigger`
- `PwmChannelGroupSwitchEventTrigger`
- `IcuInputTriggerSelection`
- `IcuChannelGroupStartTrigger`
- `IcuChannelGroupStopTrigger`
- `GptPredefTimerStartTriggerSelect`
- `GptInputTriggerSelection`

A warning occurs, if the value of the `PwmStartTriggerSelect1` parameter is configured same, as the following parameter:

- `OcuStartTriggerSelect1`

4.3.2 Container PwmChannel

4.3.2.1 PwmTimer

Description

Specifies the physical hardware timer that is assigned to this logical channel.

Type

`EcucEnumerationParamDef`

Range

- `TCPWM_0_0`: Timer counter pulse width modulation instance 0 Channel 0
- `TCPWM_0_1`: Timer counter pulse width modulation instance 0 Channel 1
- ...
- `TCPWM_1_0`: Timer counter pulse width modulation instance 1 Channel 0
- `TCPWM_1_1`: Timer counter pulse width modulation instance 1 Channel 1
- ...
- `TCPWM_m_n`: Timer counter pulse width modulation instance m Channel n

Note: The number of available physical hardware timer channels depends on the configured target derivative.

Remarks

Only one configuration per `PwmTimer` is allowed in the same configuration set.

`PwmTimer` must not use TCPWM channel that is used by `PwmStartDelayTimer` and `PwmGroupStartDelayTimer`.

`PwmTimer` shows all TCPWM resources on the device. See hardware documentation [\[7\]](#) for the TCPWM connected to the pin for `PwmTimer`.

GPT, ICU, OCU drivers, and OS use TCPWM channels. PWM driver must not use TCPWM channel that is used by other modules.

4.3.2.2 PwmAlignment

Description

Defines aligned type of timer channel.

Type

`EcucEnumerationParamDef`

Range

- `PWM_LEFT_ALIGNED`: The PWM channel output is left aligned.
- `PWM_CENTER_ALIGNED`: The PWM channel output is center aligned.
- `PWM_RIGHT_ALIGNED`: The PWM channel output is right aligned.

4.3.2.3 PwmHwTriggerOutputLine

Description

This parameter is used to allow the PWM channel to output trigger. TCPWM has two output triggers; TR_OUT0 and TR_OUT1. The available output trigger depends on the hardware. The PORT module and connecting target module should be configured. Group trigger or One-to-One trigger is available.

Type

EcucEnumerationParamDef

Range

- TR_OUT0: The PWM driver will generate the output triggers to TR_OUT0 after delay time or delay ticks is specified by PwmHwTriggerOutputDefaultTime or PwmHwTriggerOutputDefaultTick from the beginning of period.
- TR_OUT1: The PWM driver will generate the output triggers to TR_OUT1 after delay time or delay ticks is specified by PwmHwTriggerOutputDefaultTime or PwmHwTriggerOutputDefaultTick from the beginning of period.
- BOTH: The PWM driver will generate the output triggers to TR_OUT0 and TR_OUT1 after delay time or delay ticks is specified by PwmHwTriggerOutputDefaultTime or PwmHwTriggerOutputDefaultTick from the beginning of period.

Remarks

The connection target of PwmHwTriggerOutputLine depends on the hardware resource.

4.3.2.4 PwmHwTriggerOutputFactor

Description

Defines the output trigger factor of timer channel.

Type

EcucEnumerationParamDef

Range

- PWM_TRIGGER_SYNC_PERIOD: The PWM driver will generate the output triggers after delay time or delay ticks is specified by PwmHwTriggerOutputDefaultTime or PwmHwTriggerOutputDefaultTick from the beginning of period.
- PWM_TRIGGER_SYNC_DUTY: The PWM driver will generate the output triggers after delay time or delay ticks specified by PwmHwTriggerOutputDefaultTime or PwmHwTriggerOutputDefaultTick from the beginning of duty.

Remarks

When PwmHwTriggerOutputLine is enabled, PwmHwTriggerOutputFactor is enabled.

When duty is 0% with PWM_TRIGGER_SYNC_DUTY, output trigger is no output.

When duty is 100% with PWM_TRIGGER_SYNC_DUTY, output trigger is output with match period.

4.3.2.5 PwmChannelStartDelay

Description

Channel start delay value defines the time in ticks that the period start is delayed from the moment of initialization.

Type

EcucIntegerParamDef

Range

0..65535

Remarks

When `PwmStartDelayTimer` or `PwmGroupStartDelay` is enabled and `PwmChannelClass` is `PWM_FIXED_PERIOD_SHIFTED` or `PWM_VARIABLE_PERIOD`, `PwmChannelStartDelay` is enabled.

4.3.2.6 PwmChannelPrescale

Description

Shows a calculated prescaler value of the `PwmTimer` based on `PwmPeriodDefault` and `McuClockReferencePointFrequency`.

Type

EcucIntegerParamDef

Range

0..7

Remarks

This parameter is enabled when `PwmGroupStartDelay` is enabled in the group that references this channel. `PwmChannelPrescale` is used to check if all channels in the PWM channel group have the same prescaler value.

When `PwmPeriodDefault` is changed, the value of `PwmChannelPrescale` is updated by pressing the **Calculate Value** button.

4.3.2.7 PwmStartDelayTimer

Description

Specifies the physical hardware timer for `PwmChannelStartDelay`.

Type

EcucEnumerationParamDef

Range

- `TCPWM_0_0`: Timer counter pulse width modulation instance 0 Channel 0
- `TCPWM_0_1`: Timer counter pulse width modulation instance 0 Channel 1
- ...
- `TCPWM_1_0`: Timer counter pulse width modulation instance 1 Channel 0

EB tresos Studio configuration interface

- TCPWM_1_1: Timer counter pulse width modulation instance 1 Channel 1
- ...
- TCPWM_m_n: Timer counter pulse width modulation instance m Channel n

Note:

The number of available physical hardware timer channels depends on the configured target derivative.

Remarks

When `PwmChannelClass` is `PWM_FIXED_PERIOD_SHIFTED` or `PWM_VARIABLE_PERIOD`, `PwmStartDelayTimer` is enabled.

Only one configuration per `PwmStartDelayTimer` is allowed in the same configuration set.

`PwmStartDelayTimer` must not use TCPWM channel that is used by `PwmTimer` and `PwmGroupStartDelayTimer`.

GPT, ICU, OCU drivers, and OS use TCPWM channels. PWM driver must not use TCPWM channel that is used by other modules.

4.3.2.8 PwmStartDelayMcuClockReferencePoint

Description

This parameter contains reference to the `McuClockReferencePoint` for `PwmStartDelayTimer`

Type

`EcucReferenceDef`

Remarks

When `PwmStartDelayTimer` is enabled (`PwmChannelClass` is `PWM_FIXED_PERIOD_SHIFTED` or `PWM_VARIABLE_PERIOD`), `PwmStartDelayMcuClockReferencePoint` is enabled.

The clock frequency of `PwmStartDelayTimer` must be the same as `PwmTimer`.

4.3.2.9 PwmStartDelayTrigger

Description

Specifies the input trigger for `PwmChannelStartDelay`.

Type

`EcucEnumerationParamDef`

Range

- TCPWM_0_TR_ALL_CNT_IN_0: Input trigger 0 of TCPWM instance 0
- TCPWM_0_TR_ALL_CNT_IN_1: Input trigger 1 of TCPWM instance 0
- ...
- TCPWM_1_TR_ALL_CNT_IN_0: Input trigger 0 of TCPWM instance 1
- TCPWM_1_TR_ALL_CNT_IN_1: Input trigger 1 of TCPWM instance 1
- ...
- TCPWM_m_TR_ALL_CNT_IN_n: Input trigger n of TCPWM instance m

Note: *The number of available input triggers depend on the hardware resources.*

Remarks

When `PwmStartDelayTimer` is enabled (`PwmChannelClass` is `PWM_FIXED_PERIOD_SHIFTED` or `PWM_VARIABLE_PERIOD`), `PwmStartDelayTrigger` is enabled.

PWM, ICU, GPT, and OCU drivers use input triggers of TCPWM.

An error occurs, if the value of the `PwmStartDelayTrigger` parameter is configured the same, as the following parameters in the same configuration set:

- `PwmGroupStartDelayTrigger`
- `PwmStartDelayTrigger`
- `PwmChannelGroupStartTrigger`
- `PwmChannelGroupStopTrigger`
- `PwmChannelGroupSwitchEventTrigger`

If the value of `PwmStartDelayTrigger` parameter is configured as same the value of followings parameter, an error occurs.

- `PwmStartTriggerSelect0`
- `PwmStartTriggerSelect1`
- `IcuInputTriggerSelection`
- `IcuChannelGroupStartTrigger`
- `IcuChannelGroupStopTrigger`
- `OcuStartTriggerSelect0`
- `OcuStartTriggerSelect1`
- `GptPredefTimerStartTriggerSelect`
- `GptInputTriggerSelection`

4.3.2.10 PwmHwTriggerOutputDefaultTime

Description

Specifies the value of output trigger default delay time in seconds.

This value describes a fixed time from the beginning of the period when the output trigger is set off. The respective register value depending on the current period is calculated and set. If the period is modified by `Pwm_SetPeriodAndDuty()`, the register value is recalculated accordingly.

If `PwmHwTriggerOutputDefaultTime` is greater than `PwmPeriodDefault`, an error will be reported at configuration time.

Type

`EcucFloatParamDef`

Range

`0..PwmPeriodDefault`

Remarks

When `PwmHwTriggerOutputLine` is enabled and the value of `PwmHwTriggerOutputUnitType` is `PWM_TRIGGER_SECOND` and `PwmOutputOffset` is disabled, `PwmHwTriggerOutputDefaultTime` is enabled.

4.3.2.11 PwmHwTriggerOutputDefaultTick

Description

Specifies the value of output trigger default delay in ticks.

This value describes the fixed ticks from the beginning of the period when the output trigger is set off. The respective register value depending on the current period is calculated and set. If the period is modified by `Pwm_SetPeriodAndDuty()`, the register value is recalculated accordingly.

If `PwmHwTriggerDefaultOutputTick` is greater than `PwmPeriodDefault` (converted to ticks), an error will be reported at configuration time.

Type

`EcucIntegerParamDef`

Range

0..65534

Remarks

When `PwmHwTriggerOutputLine` is enabled and the value of `PwmHwTriggerOutputUnitType` is `PWM_TRIGGER_TICK` and `PwmOutputOffset` is disabled, `PwmHwTriggerOutputDefaultTick` is enabled.

4.3.2.12 PwmHwTriggerOutputScaled

Description

Switch for enabling the scaling of the output trigger delay with the period by specified by `Pwm_SetPeriodAndDuty`.

Type

`EcucBooleanParamDef`

Range

- `TRUE`: Scaling with the period change.
- `FALSE`: Not scaling with the period change.

Remarks

`PwmHwTriggerOutputScaled` can be editable only when `PwmHwTriggerOutputLine` is enabled and `PwmOutputOffset` is disabled.

4.3.2.13 PwmSetOutputEnable

Description

Supports set channel output state.

Type

`EcucBooleanParamDef`

Range

- `TRUE`: Supports the set channel output state.
- `FALSE`: Does not support the set channel output state.

Remarks

When `PwmSetOutputState` is `TRUE` and the channel has a function to set the output state, `PwmSetOutputEnable` is enabled.

4.3.2.14 PwmOutputOffset

Description

The offset value for the rising edge in ticks.

Type

`EcucIntegerParamDef`

Range

0..65533

Remarks

`PwmOutputOffset` cannot exceed the end of a period.

`PwmOutputOffset` cannot use when `PwmAlignment` is `PWM_RIGHT_ALIGNED`.

`PwmOutputOffset` can set the value from 0 to (period ticks - duty ticks - 1) when `DutyCycle` is not 0% nor 100%.

`PwmOutputOffset` can set the value from 0 to (period ticks - 1) when `DutyCycle` is 0% or 100%.

4.3.2.15 PwmOutputOffsetScaled

Description

Switch for enabling the scaling of the rising edge offset together with the period by `Pwm_SetPeriodAndDuty`.
True: scaling with the period change / False: not scaling with the period change

Type

`EcucBooleanParamDef`

Range

- `TRUE`: Scaling with the period change.
- `FALSE`: Not scaling with the period change.

Remarks

When `PwmOutputOffset` is enabled and `PwmChannelClass` is `PWM_VARIABLE_PERIOD`, `PwmOutputOffsetScaled` is enabled.

4.3.2.16 PwmStartAtInit

Description

PWM channel is started when `Pwm_Init()` is called. If disabled, PWM channel will be started when `Pwm_SetDutyCycle()`, `Pwm_SetPeriodAndDuty()`, or `Pwm_StartGroupTrigger()` is called.

Type

EcucBooleanParamDef

Range

- `TRUE`: PWM channel is started by `Pwm_Init()`
- `FALSE`: PWM channel will be started by `Pwm_SetDutyCycle()`, `Pwm_SetPeriodAndDuty()`, or `Pwm_StartGroupTrigger()`

Remarks

If `PwmStartAtInit` is set to `FALSE`, the output level depends on `PwmIdleState`.

In case of PWM start delay function:

- If `PwmStartAtInit` is set to `TRUE` and the channel is waiting for `Port_ActTrigger()`, the output level will be one of the following:
 - `PwmChannelStartDelay` is not 0 with enabled: If `PwmOutputOffset` is disabled and `PwmAlignment` is `PWM_LEFT_ALIGNED`, the output level is the active level (`PwmPolarity`), else it is the inactive level.
 - `PwmChannelStartDelay` is 0 or disabled and current `DutyCycle` is 100%: The output level is the active level (`PwmPolarity`).
 - `PwmChannelStartDelay` is 0 or disabled and current `DutyCycle` is 0%: The output level is the inactive level (`PwmPolarity`).
 - `PwmChannelStartDelay` is 0 or disabled and current `DutyCycle` is not 0% or 100%: If `PwmAlignment` is `PWM_LEFT_ALIGNED` and `PwmOutputOffset` is 0 and enabled, the output level is the active level (`PwmPolarity`), else it is the inactive level.
- In case of PWM group start delay function:
 - If `PwmStartAtInit` is set to `TRUE` and the channel is waiting for `Port_ActTrigger()`, the output level will be `PwmIdleState`.
 - If `PwmGroupStartDelay` is enabled, the `PwmStartAtInit` of all channels in a group should be of the same value.

4.3.2.17 PwmDebugMode

Description

Specifies the behavior of PWM channel when processor is in debug mode.

Type

EcucEnumerationParamDef

Range

- `PWM_DEBUG_MODE_CONTINUE`: The counter operation continues in debug mode.
- `PWM_DEBUG_MODE_HALT`: The counter operation freezes in debug mode.

4.3.2.18 PwmUpdateOutputAtInitEnable

Description

Enables to configuration settings for the line state and complementary line state for the configured channel when the `Pwm_Init` API is called.

`PwmUpdateOutputAtInitEnable` can be set if the channel resource has the stepping motor control feature and `PwmChannelClass` is disabled or `PwmChannelClass` is `PWM_FIXED_PERIOD`.

`PwmOutputAtInitSelect` and `PwmCompOutputAtInitSelect` can be set if `PwmUpdateOutputAtInitEnable` is `TRUE`.

Type

`EcucBooleanParamDef`

Range

- `TRUE`: The PWM line state is set to `PwmOutputAtInitSelect`. The PWM complementary line state is set to `PwmCompOutputAtInitSelect`.
- `FALSE`: The PWM line state is fixed to `PWM`. The PWM complementary line state is fixed to `PWM_INV` state.

4.3.2.19 PwmOutputAtInitSelect

Description

Specifies the line state of the channel when the `Pwm_Init()` is called.

`PwmOutputAtInitSelect` can be set if `PwmUpdateOutputAtInitEnable` is `TRUE`.

Type

`EcucEnumerationParamDef`

Range

- `LOW`: Line state is set to LOW.
- `HIGH`: Line state is set to HIGH.
- `PWM`: Line state is set to PWM.
- `PWM_INV`: Line state is set to PWM_INV.
- `HIGHZ`: Line state is set to HIGH-Z.

Remarks

The edge of `Pwm_EnableNotification` API is set as if `PwmOutputAtInitSelect` is always `PWM`. Therefore, it may be different from the edge of the output waveform.

4.3.2.20 PwmCompOutputAtInitSelect

Description

Specifies the complementary line state of the channel when the `Pwm_Init()` is called.

`PwmCompOutputAtInitSelect` can be set if `PwmUpdateOutputAtInitEnable` is `TRUE`.

Type

`EcucEnumerationParamDef`

Range

- LOW: Complementary line state is set to LOW.
- HIGH: Complementary line state is set to HIGH.
- PWM: Complementary line state is set to PWM.
- PWM_INV: Complementary line state is set to PWM_INV.
- HIGHZ: Complementary line state is set to HIGH-Z.

4.3.3 Container PwmChannelGroup

4.3.3.1 PwmChannelGroupId

Description

Group Id of the PWM channel group. This value will be assigned to the symbolic names.

Type

EcucIntegerParamDef

Range

0..[ChannelId]

Remarks

Value must be unique among all groups.

4.3.3.2 PwmChannelGroupStartTrigger

Description

Specifies the input trigger to start the PWM channel group synchronously. When this parameter is configured, a trigger signal is required to start all channels in the group. It is necessary to call `Port_ActTrigger()` to issue a trigger signal after calling `Pwm_StartGroupTrigger()`. In this case, the group trigger configuration is also required in the PORT module. When this parameter is not configured, the channels in the group are started sequentially by `Pwm_StartGroupTrigger()`.

Type

EcucEnumerationParamDef

Range

- TCPWM_0_TR_ALL_CNT_IN_0: Input trigger 0 of TCPWM instance 0
- TCPWM_0_TR_ALL_CNT_IN_1: Input trigger 1 of TCPWM instance 0
- ...
- TCPWM_1_TR_ALL_CNT_IN_0: Input trigger 0 of TCPWM instance 1
- TCPWM_1_TR_ALL_CNT_IN_1: Input trigger 1 of TCPWM instance 1
- ...
- TCPWM_m_TR_ALL_CNT_IN_n: Input trigger n of TCPWM instance m

Note: The number of available input triggers depends on the hardware resources.

Remarks

PWM, ICU, GPT, and OCU drivers use input triggers of TCPWM.

An error occurs, if the value of the `PwmChannelGroupStartTrigger` parameter is configured same, as the following parameters in the same configuration set:

- `PwmGroupStartDelayTrigger`
- `PwmStartDelayTrigger`
- `PwmChannelGroupStopTrigger`
- `PwmChannelGroupSwitchEventTrigger`

An error occurs, if the value of the `PwmChannelGroupStartTrigger` parameter is configured same, as the following parameters:

- `PwmStartTriggerSelect0`
- `PwmStartTriggerSelect1`
- `IcuInputTriggerSelection`
- `IcuChannelGroupStopTrigger`
- `OcuStartTriggerSelect0`
- `OcuStartTriggerSelect1`
- `GptPredefTimerStartTriggerSelect`
- `GptInputTriggerSelection`

A warning occurs, if the value of the `PwmChannelGroupStartTrigger` parameter is configured same, as the following parameter in the same configuration set:

- `PwmChannelGroupStartTrigger`

A warning occurs, if the value of the `PwmChannelGroupStartTrigger` parameter is configured same, as the following parameter:

- `IcuChannelGroupStartTrigger`

If `PwmChannelRef` has `PwmTimer` of different instance, an error occurs.

4.3.3.3 PwmChannelGroupStopTrigger

Description

Specifies the input trigger to stop the PWM channel group synchronously. When this parameter is configured, a trigger signal is required to stop all channels in the group. It is necessary to call `Port_ActTrigger()` to issue a trigger signal after calling `Pwm_StopGroupTrigger()`. In this case, the group trigger configuration is also required in PORT module. When this parameter is not configured, the channels in the group are stopped sequentially by `Pwm_StopGroupTrigger()`.

Type

`EcucEnumerationParamDef`

Range

- `TCPWM_0_TR_ALL_CNT_IN_0`: Input trigger 0 of TCPWM instance 0
- `TCPWM_0_TR_ALL_CNT_IN_1`: Input trigger 1 of TCPWM instance 0
- ...

EB tresos Studio configuration interface

- TCPWM_1_TR_ALL_CNT_IN_0: Input trigger 0 of TCPWM instance 1
- TCPWM_1_TR_ALL_CNT_IN_1: Input trigger 1 of TCPWM instance 1
- ...
- TCPWM_m_TR_ALL_CNT_IN_n: Input trigger n of TCPWM instance m

Note: The number of available input triggers depends on hardware resources.

Remarks

PWM, ICU, GPT, and OCU drivers use input triggers of TCPWM.

An error occurs, if the value of the `PwmChannelGroupStopTrigger` parameter is configured the same, as the following parameters in the same configuration set:

- `PwmGroupStartDelayTrigger`
- `PwmStartDelayTrigger`
- `PwmChannelGroupStartTrigger`
- `PwmChannelGroupSwitchEventTrigger`

An error occurs, if the value of the `PwmChannelGroupStopTrigger` parameter is configured the same, as the following parameters:

- `PwmStartTriggerSelect0`
- `PwmStartTriggerSelect1`
- `IcuInputTriggerSelection`
- `IcuChannelGroupStopTrigger`
- `OcuStartTriggerSelect0`
- `OcuStartTriggerSelect1`
- `GptPredefTimerStartTriggerSelect`
- `GptInputTriggerSelection`

A warning occurs, if the value of the `PwmChannelGroupStopTrigger` parameter is configured the same, as the following parameter in the same configuration set:

- `PwmChannelGroupStopTrigger`

A warning occurs, if the value of the `PwmChannelGroupStopTrigger` parameter is configured the same, as the following parameter:

- `IcuChannelGroupStopTrigger`

If `PwmChannelRef` has `PwmTimer` of different instance, an error occurs.

4.3.3.4 PwmChannelRef

Description

Assignment of channels to a channel group.

Type

EcucReferenceDef

Remarks

Referenced channel must be unique in a group. Referenced channel must be unique among all groups when PwmChannelGroupStopTrigger is effective.

When PwmChannelGroupSwitchEventTrigger is enabled and PwmChannelClass is not configured as disabled or PwmChannelClass is PWM_FIXED_PERIOD for channels in the group, an error occurs.

When PwmGroupStartDelay is configured, the following checks are conducted:

- If PwmChannelClass is disabled or PWM_FIXED_PERIOD, an error occurs.
- If PwmStartDelayTimer of the referenced channel is enabled, an error occurs.
- If PwmMcuClockReferencePoint of the referenced channel and PwmGroupStartDelayMcuClockReferencePoint are different from McuClockReferencePointFrequency, an error occurs.
- If PwmStartAtInit of all referenced channels is not same, an error occurs.
- If PwmChannelPrescale of all referenced channels is not same, an error occurs.
- If referenced channel is referred from other group, an error occurs.

4.3.3.5 PwmChannelGroupSwitchEventTrigger

Description

Specifies the input trigger for Pwm_SetDutyAndChannelOutputBuffer, Pwm_SetDutyCycleBuffer, Pwm_SetChannelOutputBuffer function with Port_ActTrigger. The input trigger is used to reserve switching of the output waveform after the end of the period for the channels in the group.

Type

EcucEnumerationParamDef

Range

- TCPWM_0_TR_ALL_CNT_IN_0: Input trigger 0 of TCPWM instance 0
- TCPWM_0_TR_ALL_CNT_IN_1: Input trigger 1 of TCPWM instance 0
- ...
- TCPWM_1_TR_ALL_CNT_IN_0: Input trigger 0 of TCPWM instance 1
- TCPWM_1_TR_ALL_CNT_IN_1: Input trigger 1 of TCPWM instance 1
- ...
- TCPWM_m_TR_ALL_CNT_IN_n: Input trigger n of TCPWM instance m

Note: The number of available input triggers depends on hardware resources.

Remarks

PWM, ICU, GPT, and OCU drivers use input triggers of TCPWM.

An error occurs, if the value of the `PwmChannelGroupSwitchEventTrigger` parameter is configured the same, as the following parameters in the same configuration set:

- `PwmGroupStartDelayTrigger`
- `PwmStartDelayTrigger`
- `PwmChannelGroupStartTrigger`
- `PwmChannelGroupStopTrigger`
- `PwmChannelGroupSwitchEventTrigger`

An error occurs, if the value of the `PwmChannelGroupSwitchEventTrigger` parameter is configured the same, as the following parameters:

- `PwmStartTriggerSelect0`
- `PwmStartTriggerSelect1`
- `IcuInputTriggerSelection`
- `IcuChannelGroupStartTrigger`
- `IcuChannelGroupStopTrigger`
- `OcuStartTriggerSelect0`
- `OcuStartTriggerSelect1`
- `GptPredefTimerStartTriggerSelect`
- `GptInputTriggerSelection`

4.3.4 Container PwmGroupStartDelay

4.3.4.1 PwmGroupStartDelayTimer

Description

Specifies the physical hardware timer for PWM group start delay function.

Type

`EcucEnumerationParamDef`

Range

- `TCPWM_0_0`: Timer counter pulse width modulation instance 0 Channel 0
- `TCPWM_0_1`: Timer counter pulse width modulation instance 0 Channel 1
- ...
- `TCPWM_1_0`: Timer counter pulse width modulation instance 1 Channel 0
- `TCPWM_1_1`: Timer counter pulse width modulation instance 1 Channel 1
- ...
- `TCPWM_m_n`: Timer counter pulse width modulation instance m Channel n

Note: *The number of available physical hardware timer channels depend on the configured target derivative.*

Remarks

`PwmGroupStartDelayTimer` must not use TCPWM channel that is used by `PwmTimer` and `PwmStartDelayTimer`.

GPT, ICU, OCU drivers, and OS use TCPWM channels. PWM driver must not use TCPWM channel that is used by other modules.

4.3.4.2 PwmGroupStartDelayMcuClockReferencePoint

Description

This parameter contains reference to `McuClockReferencePoint` for `PwmGroupStartDelayTimer`

Type

`EcucReferenceDef`

Remarks

The clock frequency of `PwmGroupStartDelayMcuClockReferencePoint` must be the same as `PwmMcuClockReferencePoint` of each channel in the group. The prescaler value `PwmChannelPrescale` of grouped channels must be same.

4.3.4.3 PwmGroupStartDelayTrigger

Description

Specifies the input trigger for PWM group start delay function. The input trigger is used to reserve switching of the output waveform after the delay time.

Type

`EcucEnumerationParamDef`

Range

- `TCPWM_0_TR_ALL_CNT_IN_0`: Input trigger 0 of TCPWM instance 0
- `TCPWM_0_TR_ALL_CNT_IN_1`: Input trigger 1 of TCPWM instance 0
- ...
- `TCPWM_1_TR_ALL_CNT_IN_0`: Input trigger 0 of TCPWM instance 1
- `TCPWM_1_TR_ALL_CNT_IN_1`: Input trigger 1 of TCPWM instance 1
- ...
- `TCPWM_m_TR_ALL_CNT_IN_n`: Input trigger n of TCPWM instance m

Note: The number of available input triggers depends on the hardware resources.

Remarks

PWM, ICU, GPT, and OCU drivers use input triggers of TCPWM.

An error occurs, if the value of the `PwmGroupStartDelayTrigger` parameter is configured the same, as the following parameters in the same configuration set:

- `PwmGroupStartDelayTrigger`
- `PwmStartDelayTrigger`

EB tresos Studio configuration interface

- `PwmChannelGroupStartTrigger`
- `PwmChannelGroupStopTrigger`
- `PwmChannelGroupSwitchEventTrigger`

An error occurs, if the value of the `PwmGroupStartDelayTrigger` parameter is configured the same, as the following parameters:

- `PwmStartTriggerSelect0`
- `PwmStartTriggerSelect1`
- `IcuInputTriggerSelection`
- `IcuChannelGroupStartTrigger`
- `IcuChannelGroupStopTrigger`
- `OcuStartTriggerSelect0`
- `OcuStartTriggerSelect1`
- `GptPredefTimerStartTriggerSelect`
- `GptInputTriggerSelection`

4.3.5 Container `PwmConfigurationOfOptApiServices`

4.3.5.1 `PwmSetOutputState`

Description

Adds or removes the service `Pwm_SetChannelOutput()` from the code.

Type

`EcucBooleanParamDef`

4.3.5.2 `PwmSetPrescaler`

Description

Adds or removes the service `Pwm_SetPrescaler()` from the code.

Type

`EcucBooleanParamDef`

4.3.5.3 `PwmSetTriggerDelay`

Description

Adds or removes the service `Pwm_SetTriggerDelay()` from the code.

Type

`EcucBooleanParamDef`

4.3.5.4 `PwmStopGroup`

Description

Adds or removes the service `Pwm_StopGroupTrigger()` from the code.

Type

EcucBooleanParamDef

4.3.5.5 PwmSetOutputOffset**Description**

Adds or removes the service `Pwm_SetOutputOffset()` from the code.

Type

EcucBooleanParamDef

4.3.5.6 PwmEnableDisableTriggerApi**Description**

Adds or removes `Pwm_EnableTrigger()` and `Pwm_DisableTrigger()` from the code.

A configurable option called `PwmEnableDisableTriggerApi` which specifies whether the output trigger is enabled by `Pwm_Init()` is available

If the option `PwmEnableDisableTriggerApi` is set to `TRUE`, `Pwm_Init()` will not enable the output trigger.

If the option `PwmEnableDisableTriggerApi` is set to `FALSE`, `Pwm_Init()` will enable the output trigger.

Type

EcucBooleanParamDef

4.3.5.7 PwmSetDutyCycleBufferApi**Description**

Adds or removes `Pwm_SetDutyCycleBuffer()` service from the code. Default value is `FALSE`.

Type

EcucBooleanParamDef

4.3.5.8 PwmSetDutyAndChannelOutputBufferApi**Description**

Adds or removes `Pwm_SetDutyAndChannelOutputBuffer()` service from the code. Default value is `FALSE`.

Type

EcucBooleanParamDef

4.3.5.9 PwmSetChannelOutputBufferApi**Description**

Adds or removes `Pwm_SetChannelOutputBuffer()` service from the code. Default value is `FALSE`.

Type

EcucBooleanParamDef

5 Functional description

5.1 Initialization

The PWM driver needs to be initialized once before use. Initialization of the PWM driver is made by calling `Pwm_Init()`.

Note: This PWM driver supports post-build-time configuration, thus different configuration set pointer can be passed to the function `Pwm_Init()`.

No other function must be called before `Pwm_Init()` has been executed except `Pwm_GetChannelStatus()` and `Pwm_GetVersionInfo()`.

Not all necessary initializations are performed by the PWM driver. The PORT module needs to be called before the PWM driver is used. See [Adapting your application](#).

After initialization, all configured channels run with disabled notification.

If the configured default value for `PwmDutycycleDefault` results in a HW tick limit, the channel is appropriately forced into 0% or 100% duty. The HW constraints restrictions are forced into a constant output level. See [Hardware resources](#).

5.2 Runtime reconfiguration

The PWM driver is not reconfigurable at runtime. The only way to change the driver's configuration is to stop all channels, de-initialize the driver, and reinitialize with a different configuration set.

5.3 PWM channel

The `PwmChannelId` defines the channel number to use. It specifies the PWM channel for which the service is done. A user-given symbolic name is available or generated for each configured channel. Use this symbolic name for API calls.

5.4 PWM channel group

PWM channels can be organized in channel groups. The parameter `PwmChannelGroupId` defines the group which references the selected channels. Only configured channels can be assigned to a group. The channel groups can be triggered by using `Pwm_StartGroupTrigger()` and `Pwm_StopGroupTrigger()`.

A user-given symbolic name is available or generated for each configured channel group. Use this symbolic name for API calls.

5.5 Notification

Notification can be enabled or disabled by calling `Pwm_EnableNotification()` and `Pwm_DisableNotification()`.

Notifications are disabled by default after calling of `Pwm_Init()`.

Notification is disabled when a channel duty is 0% or 100%.

If the configured notification function name (`PwmNotification`) is blank or NULL, no notification function prototype is declared. Enabling and disabling is still possible, but no notification call is done by the PWM driver.

Functional description

Note: *The notification can be disabled by calling `Pwm_DisableNotification` as mentioned in this chapter. However, it would not work with notifications that have already been handled, if `Pwm_DisableNotification` is called from a higher interruption during a few cycles before the user-defined notification function is called. The notification reason can be change by the second argument of `Pwm_EnableNotification`. However, it would not work with notifications that have already been handled, if `Pwm_EnableNotification` is called from a higher interruption during a few cycles before the user-defined notification function is called.*

5.6 Stopping a channel

A channel can be stopped by calling `Pwm_SetOutputToIdle()`.

5.7 Starting a channel

After a channel is stopped it can be restarted by calling `Pwm_SetDutyCycle()` or `Pwm_SetPeriodAndDuty()` with new duty/period values.

Notification depends on the new duty value.

5.8 Updating a channel duty and period

A channel duty can be updated by calling `Pwm_SetDutyCycle()` or `Pwm_SetPeriodAndDuty()`.

A channel period can be updated by calling `Pwm_SetPeriodAndDuty()`.

Note: *A channel duty is updated immediately by calling `Pwm_SetDutyCycle()` when `PwmDutyCycleUpdatedEndperiod` is `FALSE`. If not, the update is in sync with the end of period. A channel period and duty are updated immediate by calling `Pwm_SetPeriodAndDuty()` when `PwmPeriodUpdatedEndperiod` is `FALSE`. If not, the update is in sync with the end of period.*

5.9 Delaying PWM output

Start delay function is used to delay the output of a PWM with the configuration parameter `PwmChannelStartDelay`. There are two types of start delay functions:

- PWM channel start delay

The PWM channel start delay function is configured by `PwmChannelStartDelay`, `PwmStartDelayTimer`, `PwmStartDelayMcuClockReferencePoint`, and `PwmStartDelayTrigger` in the `PwmChannel` container. It requires an additional TCPWM resource per PWM channel for the PWM start delay function.

When the PWM start delay function is configured, the function `Pwm_Init()`, `Pwm_StartGroupTrigger()`, `Pwm_SetDutyCycle()`, or `Pwm_SetPeriodAndDuty()` starts the channel with a start delay value. When `PwmAlignment` is `PWM_LEFT_ALIGNED` and `PwmOutputOffset` is enabled, the output level during the start delay is at the `PwmPolarity` level. Other than that, the output level is inverse of the `PwmPolarity` level.

- PWM group start delay

The PWM group start delay function is configured by `PwmChannelStartDelay` in the `PwmChannel` container, `PwmGroupStartDelayTimer`, `PwmGroupStartDelayMcuClockReferencePoint`, and `PwmGroupStartDelayTrigger` in the `PwmChannelGroup` container. It requires a TCPWM resource per PWM channel group for the PWM start delay function. The delay time can be configured by `PwmChannelStartDelay` for each PWM channel.

Functional description

When the PWM group start delay function is configured, the function `Pwm_Init()` or `Pwm_StartGroupTrigger()` start the grouped channels with a start delay value. The output level during the start delay is an inverse of the `PwmPolarity` level.

Note: *All PWM channels in a group need the same time base for the PWM group start delay function. If the frequency of `PwmGroupStartDelayMcuClockReferencePoint` and `PwmMcuClockReferencePoint` of each channel in the group do not match, an error occurs on EB tresos. If `PwmChannelPrescale` of the grouped channels is not the same, an error occurs on EB tresos. If the grouped channels have different prescaler values, `Pwm_StartGroupTrigger` reports the error code `PWM_E_DIFFERENT_PRESCALER`. All PWM channels in a group must have the same value for `PwmStartAtInit`. `Pwm_SetDutyCycle` and `Pwm_SetPeriodAndDuty` start a stopped channel without the group start delay function.*

Note: *PWM channels with the PWM channel start delay can belong to a PWM channel group, but if `PwmGroupStartDelay` is enabled in the PWM channel group, `PwmStartDelayTimer`, `PwmStartDelayMcuClockReferencePoint`, and `PwmStartDelayTrigger` in the `PwmChannel` container cannot be configured in the grouped channels. The start delay function works as the PWM group start delay.*

5.10 Triggering input to a channel group

A channel group can be triggered by calling `Pwm_StartGroupTrigger()`. The trigger will start all channels in the group simultaneously. The settings of the channels are not modified.

A channel group can be triggered by calling `Pwm_StopGroupTrigger()`. The trigger will stop all channels in the group simultaneously. The settings of the channels are not modified.

If `PwmStartAtInit` is set to `FALSE` or the channels is set to idle (e.g. using `Pwm_SetOutputToIdle()`), the channels are started by `Pwm_StartGroupTrigger()`. If the channels were already started, then the channels in the group will be re-triggered.

The group's channels specify their delay time via `PwmChannelStartDelay`.

5.11 Output trigger to peripherals

The PWM driver generates the output triggers to peripherals such as ADC channel at the timing configured by `PwmHwTriggerOutputDefaultTime` or `PwmHwTriggerOutputDefaultTick`.

The parameter `PwmHwTriggerOutputLine` allows the PWM channel to output trigger. The trigger signal should be configured by the PORT driver and target peripherals. Multiplexer-based trigger or One-to-One trigger is available for this function.

The value of this parameter specifies the output trigger of TCPWM as follows:

- `TR_OUT0`: Output trigger 0
- `TR_OUT1`: Output trigger 1
- `BOTH`: Both output trigger 0 and 1

The available output trigger depends on the hardware.

5.12 Prescaler setting function

The PWM driver provides the prescaler setting function for PWM channel during runtime. This function is to maintain the same tick frequency of a channel by changing the prescaler setting without initialization when the input clock frequency for a channel is changed.

`Pwm_SetPrescaler()` changes the TCPWM prescaler setting of the selected channel according to the specified input clock frequency.

The PWM channel is stopped by calling `Pwm_SetPrescaler()`. If the PWM channel is in the running state before calling `Pwm_SetPrescaler()`, the PWM channel is restarted.

Input clock frequency of `PwmConf_PwmChannel_MY_CHANNEL` is changed by the MCU driver.

```
Pwm_SetPrescaler(PwmConf_PwmChannel_MY_CHANNEL, MY_CLOCK_FREQUENCY);
```

Note: *`Pwm_SetPrescaler()` calculates the prescaler value based on the value of the input clock frequency and tick frequency.
Calculating formula: `MY_CLOCK_FREQUENCY` divided by tick frequency (Round down decimals).
The input frequency of the argument of `Pwm_SetPrescaler()` may cause a poor accuracy of tick duration. In that case, the frequency of input clock should be adjusted by the MCU driver to meet the prescaler value (1, 2, 4, 8, 16, 32, 64, 128).*

1. The calculation result of input clock frequency divided by tick frequency does not meet the prescaler value (1, 2, 4, 8, 16, 32, 64, 128).
2. If the input clock frequency is close to the tick frequency, the appropriate prescaler value may not be set.

5.13 Retrieving status of a channel

The status of a channel can be assessed by calling `Pwm_GetChannelStatus()`. The internal register values are read out and correspondence between the software and hardware status is determined.

5.14 Sleep mode

The PWM module does not have a Sleep mode: therefore, the PWM channel should be stopped before MCU enters the Sleep mode.

Note: *All TCPWM counters must be stopped before entering DeepSleep mode with the following APIs.*

`Pwm_DeInit()`, `Pwm_SetOutputTogle()`, or `Pwm_StopGroupTrigger()` if the channel group is configured.

5.15 API parameter checking

The driver's services perform regular error checks.

When an error occurs, the error hook routine (configured via `PwmErrorCalloutFunction`) is called and the error code, service ID, module ID, and instance ID are passed as parameters.

If development error detection is enabled, all errors are also reported to DET, a central error hook function within the AUTOSAR environment. The checking cannot be deactivated for safety reasons.

The following development error checks are performed by the services of the PWM driver:

Functional description

- If one of the functions `Pwm_DeInit`, `Pwm_SetDutyCycle`, `Pwm_SetPeriodAndDuty`, `Pwm_SetOutputToIdle`, `Pwm_GetOutputState`, `Pwm_DisableNotification`, `Pwm_EnableNotification`, `Pwm_SetChannelOutput`, `Pwm_StartGroupTrigger`, `Pwm_StopGroupTrigger`, `Pwm_SetPrescaler`, `Pwm_SetOutputOffset`, `Pwm_SetTriggerDelay`, `Pwm_EnableTrigger`, `Pwm_DisableTrigger`, `Pwm_GetChannelStatus`, `Pwm_SetDutyCycleBuffer`, `Pwm_SetDutyAndChannelOutputBuffer`, or `Pwm_SetChannelOutputBuffer` is called before `Pwm_Init` is called, an error code `PWM_E_UNINIT` is reported.
- If the function `Pwm_GetChannelStatus` is called with abnormal configured data, an error code `PWM_E_UNINIT` is reported.
- If one of the functions `Pwm_SetDutyCycle`, `Pwm_SetPeriodAndDuty`, `Pwm_SetOutputToIdle`, `Pwm_GetOutputState`, `Pwm_DisableNotification`, `Pwm_EnableNotification`, `Pwm_DisableTrigger`, `Pwm_EnableTrigger`, `Pwm_SetChannelOutput`, `Pwm_SetOutputOffset`, `Pwm_SetTriggerDelay`, `Pwm_SetPrescaler`, `Pwm_GetChannelStatus`, `Pwm_SetDutyCycleBuffer`, `Pwm_SetDutyAndChannelOutputBuffer`, or `Pwm_SetChannelOutputBuffer` is called with an invalid parameter `ChannelNumber`, an error code `PWM_E_PARAM_CHANNEL` is reported.
- If the function `Pwm_SetPeriodAndDuty` is called and the period was configured as fixed, an error code `PWM_E_PERIOD_UNCHANGEABLE` is reported.
- If the function `Pwm_Init` is called when `Pwm_Init` was already called, an error code `PWM_E_ALREADY_INITIALIZED` is reported.
- If one of the functions `Pwm_GetVersionInfo` or `Pwm_GetChannelStatus` is called with NULL pointer, an error code `PWM_E_PARAM_POINTER` is reported.
- If one of the functions `Pwm_SetDutyCycle`, `Pwm_SetPeriodAndDuty`, `Pwm_SetDutyCycleBuffer`, or `Pwm_SetDutyAndChannelOutputBuffer` is called with an invalid duty parameter (`Duty > 0x8000`), an error code `PWM_E_DUTY_OUT_OF_RANGE` is reported.
- If the function `Pwm_SetPeriodAndDuty` is called with an invalid parameter `Period`, an error code `PWM_E_PERIOD_OUT_OF_RANGE` is reported.
- If the function `Pwm_EnableNotification` is called with an invalid parameter `Notification`, an error code `PWM_E_PARAM_NOTIFICATION` is reported.
- If the function `Pwm_Init` is called with an invalid parameter `ConfigPtr`, an error code `PWM_E_INIT_FAILED` is reported.
- If the function `Pwm_StartGroupTrigger` or `Pwm_StopGroupTrigger` is called with an invalid group identifier, an error code `PWM_E_PARAM_GROUP` is reported.
- If the function `Pwm_StartGroupTrigger` is called with a group identifier which includes channels that do not have the same prescaler value, an error code `PWM_E_DIFFERENT_PRESCALER` is reported.
- If the function `Pwm_SetChannelOutput` is called with an invalid parameter `State`, an error code `PWM_E_PARAM_STATE` is reported.
- If the function `Pwm_SetDutyAndChannelOutputBuffer` or `Pwm_SetChannelOutputBuffer` is called with an invalid parameter `LineState` or `LineCompState`, an error code `PWM_E_PARAM_STATE` is reported.
- If the function `Pwm_SetOutputOffset` is called with an invalid parameter `OffsetTick`, an error code `PWM_E_PARAM_TICK` is reported.
- If the function `Pwm_SetTriggerDelay` is called with an invalid parameter `TriggerTicks`, an error code `PWM_E_PARAM_TICK` is reported.
- If the function `Pwm_SetPrescaler` is called with an invalid parameter `ClockFrequency`, an error code `PWM_E_PARAM_CLOCK` is reported.
- If one of the functions `Pwm_SetDutyCycle`, `Pwm_SetPeriodAndDuty`, `Pwm_SetOutputToIdle`, `Pwm_SetChannelOutput`, `Pwm_StartGroupTrigger`, `Pwm_StopGroupTrigger`, `Pwm_DisableTrigger`, `Pwm_EnableTrigger`, `Pwm_SetOutputOffset`, `Pwm_SetTriggerDelay`, `Pwm_SetPrescaler`,

Functional description

`Pwm_SetDutyCycleBuffer`, `Pwm_SetDutyAndChannelOutputBuffer`, or `Pwm_SetChannelOutputBuffer` is called with waiting trigger state, an error code `PWM_E_WAITING_TRIGGER` is reported.

- If the function `Pwm_SetChannelOutput` is called while channel is in idle state, an error code `PWM_E_IDLE` is reported.

5.16 Configuration checking

The `PwmChannelId` defines the assignment of the channel to the physical PWM hardware channel. `PwmChannelId` and `PwmChannelGroupId` must be unique in the same configuration set.

5.17 Reentrancy

The following functions are reentrant if called on different channel numbers:

- `Pwm_SetDutyCycle()`
- `Pwm_SetPeriodAndDuty()`
- `Pwm_SetOutputToIdle()`
- `Pwm_GetOutputState()`
- `Pwm_DisableNotification()`
- `Pwm_EnableNotification()`
- `Pwm_GetChannelStatus()`
- `Pwm_SetChannelOutput()`
- `Pwm_SetPrescaler()`
- `Pwm_SetOutputOffset()`
- `Pwm_SetTriggerDelay()`
- `Pwm_EnableTrigger()`
- `Pwm_DisableTrigger()`
- `Pwm_SetDutyCycleBuffer()`
- `Pwm_SetDutyAndChannelOutputBuffer()`
- `Pwm_SetChannelOutputBuffer()`

Note: *The PWM module's user shall establish a mutual exclusion mechanism in case several function calls are made during run time in different tasks or ISRs targeting the same PWM channel.*

They are non-reentrant if called on the same channel number. The functions may be accessed by different tasks/interrupts simultaneously if the tasks or interrupts access disjunct sets of channels. All parameters of `Pwm_GetChannelStatus()` have to be called by different values. `Pwm_GetVersionInfo()` is reentrant. `Pwm_Init()` and `Pwm_DeInit()` are not reentrant.

The following functions are reentrant if called on different channel numbers within different channel group numbers. If not, they are non-reentrant.

- `Pwm_StartGroupTrigger()`
- `Pwm_StopGroupTrigger()`

5.18 Debugging support

The PWM driver does not support debugging.

5.19 Execution time dependencies

The execution time of certain API functions is dependent on certain factors. See [Table 1](#) for details.

Table 1 Execution time dependencies

Affected function	Dependency
Pwm_Init() Pwm_DeInit() Pwm_StartGroupTrigger() Pwm_StopGroupTrigger()	Runtime depends on the number of configured channels and groups because all configured channels and groups are processed in these functions.

6 Hardware resources

6.1 Ports and pins

To use the PWM driver, you should configure the following pins within the PORT driver first.

The following physical hardware needs to be configured for PWM channel: (m is instance number. n is channel number):

- `PwmTimer = TCPWM_m_n`

The PORT driver must be configured with the appropriate port pin:

- `PortPinName = Pxxx_y` for PWM pin
- `PortPinDirection = PORT_PIN_OUT`
- `PortPinInitialMode = TCPWM_m_LINE_n`
- `PortPinOutputDrive = PORT_PIN_OUT_MODE_PULLDOWN` or `PORT_PIN_OUT_MODE_PULLUP`
- `PortPinMode = TCPWM_m_LINE_n` (same as `PortPinInitialMode`)

The associated port pin for each PWM channel is subderivative dependent and see the hardware manual.

6.2 Timer

For each configured PWM channel, one hardware timer of the TRAВЕО™ T2G family is reserved exclusively. This is done via configuration parameter `PwmTimer`.

6.3 Interrupts

The PWM driver uses the interrupts associated with the configured hardware resource. The ISR must be declared in the AUTOSAR OS as Category 1 Interrupt or Category 2 Interrupt.

Note: IRQ numbers depend on the subderivative. See the hardware manual.

You can define the ISR. The ISR is specified as:

The IRQ-Name of each PWM channel must be `Pwm_Isr_Vector_<IRQ No>_Cat1` for Category-1 ISR or `Pwm_Isr_Vector_<IRQ No>_Cat2` for Category-2 ISR.

Note: `Pwm_Isr_Vector_<IRQ No>_Cat2` must be called from the (OS) interrupt service routine. In the case of Category-1 usage, the address of `Pwm_Isr_Vector_<IRQ No>_Cat1` must be the entry in the (OS) interrupt vector table.

Example: Category-1 ISR for the configured PWM Channel 0 located in the generated file `src/Pwm_Irq.c`:

```
ISR_NATIVE(Pwm_Isr_Vector_273_Cat1)
{
...
}
```


Example: Category-2 ISR for the configured PWM Channel 0 located in the generated file `src/Pwm_Irq.c`:

```
ISR(Pwm_Isr_Vector_273_Cat2)
{
...
}
```

Note: *On the Arm® Cortex®-M4 CPU, priority inversion of interrupts may occur under specific timing conditions in the integrated system with TRAVERO™ T2G MCAL. For more details, see the following errata notice.*

Arm® Cortex®-M4 Software Developers Errata Notice - 838869:

“Store immediate overlapping exception return operation might vector to incorrect interrupt”

If the user application cannot tolerate the priority inversion, a DSB instruction should be added at the end of the interrupt function to avoid the priority inversion.

TRAVERO™ T2G MCAL interrupts are handled by an ISR wrapper (handler) in the integrated system. Thus, if necessary, the DSB instruction should be added just before the end of the handler by the integrator.

6.4 Triggers

In general, a trigger input signal indicates the completion of a peripheral action or a peripheral event. A trigger output signal initiates a peripheral action. There are two kinds of trigger groups: Trigger (Multiplexer-based trigger) group and trigger One-to-One group. Trigger group is a multiplexer-based connectivity group. This type connects a peripheral input trigger to multiple peripheral output triggers. Trigger One-to-One group is a One-to-One-based connectivity group. This type connects a peripheral input trigger to one specific peripheral output trigger.

For more detail about triggers, see hardware documentation [7].

PWM driver uses input or output trigger signal in the following cases:

- Triggering peripheral action

PWM channel generates output trigger signal to peripherals. See [Output trigger to peripherals](#).

- Triggering input for timer synchronous START or STOP or switching line state and complementary line state for `Pwm_SetDutyCycleBuffer`, `Pwm_SetDutyAndChannelOutputBuffer` and `Pwm_SetChannelOutputBuffer` APIs

A trigger multiplexer from CPUSS_ZERO to TCPWM channels should be configured in PORT driver.

- `PortOutputTrigger` configuration setting:
- `PortTrOutputName`: Select trigger signal to all TCPWM (e.g. `TCPWM_0_TR_ALL_CNT_IN_8`).
- `PortTrInputName`: Select trigger signal (e.g. `CPUSS_ZERO`)
- `PortTrInvertEnable`: Disable
- `PortTrSensitiveType`: `PORT_TR_SENSITIVE_EDGE`
- Debug capability to STOP a timer channel

Hardware resources

A trigger multiplexer from CTI (Cross triggering interface) to TCPWM for each TCPWM instance should be configured in PORT driver.

- PortOutputTrigger configuration setting 1:
- PortTrOutputName: Select debug input trigger (e.g. TR_GROUP_9_INPUT_1)
- PortTrInputName: Select CTI signal (e.g. CPUSS_CTI_TR_OUT_0)
- PortTrInvertEnable: Disable
- PortTrSensitiveType: PORT_TR_SENSITIVE_LEVEL

PortOutputTrigger configuration setting 2:

- PortTrOutputName: TCPWM_n_TR_DEBUG_FREEZE (n: TCPWM instance number)
 - PortTrInputName: Select debug output trigger (e.g. TR_GROUP_9_OUTPUT_1)
 - PortTrInvertEnable: Disable
 - PortTrSensitiveType: PORT_TR_SENSITIVE_LEVEL
- PwmStartDelayTrigger settings

A trigger multiplexer from TCPWM to TCPWM channel should be configured in PORT driver.

PortOutputTrigger configuration setting:

- PortTrOutputName: Select trigger signal (e.g. TCPWM_0_TR_ALL_CNT_IN_1).
 - PortTrInputName: Select same instance number and channel number of PwmStartDelayTimer (e.g. TCPWM_0_TR_OUT0_0). If there are OUT0 and OUT1, OUT0 should be selected.
 - PortTrInvertEnable: Disable
 - PortTrSensitiveType: PORT_TR_SENSITIVE_EDGE
- PwmGroupStartDelayTrigger settings

A trigger multiplexer from TCPWM to TCPWM channel should be configured in PORT driver.

PortOutputTrigger configuration setting:

- PortTrOutputName: Select trigger signal (e.g. TCPWM_0_TR_ALL_CNT_IN_1).
- PortTrInputName: Select same instance number and channel number of PwmGroupStartDelayTimer (e.g. TCPWM_0_TR_OUT0_0). If OUT0 and OUT1 are available, OUT0 should be selected.
- PortTrInvertEnable: Disable
- PortTrSensitiveType: PORT_TR_SENSITIVE_EDGE

7 Appendix A – API reference

7.1 Include files

The *Pwm.h* file includes the necessary external identifiers. Thus, the application only needs to include *Pwm.h* to make all API functions and data types available.

7.2 Data types

7.2.1 Pwm_ChannelType

Type

```
uint16
```

Description

Defines the channel Id of the PWM channel.

7.2.2 Pwm_PeriodType

Type

```
uint32
```

Description

Defines the period length in ticks. The period length can be up to 2^{16} ticks. To use this type for calculation, it is extended to 32 bits.

7.2.3 Pwm_OutputStateType

Type

```
typedef enum
{
    PWM_LOW = 0,
    PWM_HIGH
} Pwm_OutputStateType;
```

Description

Defines the output states of a PWM channel

7.2.4 Pwm_EdgeNotificationType

Type

```
typedef enum
{
    PWM_RISING_EDGE = 0,
    PWM_FALLING_EDGE,
    PWM_BOTH_EDGES
} Pwm_EdgeNotificationType;
```

Description

Defines the edge notification states of a PWM channel

7.2.5 Pwm_ClkFrequencyType

Type

```
uint32
```

Description

Defines the clock frequency

The unit of this type is Hz

7.2.6 Pwm_ConfigType

Type

```
typedef struct
```

Description

Defines the type of data structure containing the initialization data for the PWM driver

7.2.7 Pwm_ChannelGroupType

Type

```
uint16
```

Description

Defines the identifier of the PWM channel group

7.2.8 Pwm_DriverStatusType

Type

```
typedef enum
{
    PWM_S_UNINITIALIZED = 0,
    PWM_S_INITIALIZED
} Pwm_DriverStatusType;
```

Description

Defines the type of driver states

7.2.9 Pwm_ChannelStatusType

Type

```
typedef struct
{
    Pwm_PeriodType    PeriodTicks;
    Pwm_PeriodType    DelayTicks;
    Pwm_PeriodType    OutputOffsetTicks;
    Pwm_PeriodType    TriggerTicks;
    uint16            DutyCycle;
    Pwm_ChannelType    ChannelId;
    uint8              Prescaler;
    uint8              Status;
    boolean            NotificationEnabled;
    boolean            TriggerEnabled;
} Pwm_ChannelStatusType;
```

Description

Holds the status parameters of a channel.

Note: About the value of status member, see [Table 7](#). Status has multiple values.

7.2.10 Pwm_OutputLineStateType

Type

```
typedef enum
```

```
{
    PWM_LINESTATE_LOW = 0,
    PWM_LINESTATE_HIGH,
    PWM_LINESTATE_PWM,
    PWM_LINESTATE_PWM_INV,
    PWM_LINESTATE_HIGHZ
} Pwm_OutputLineStateType;
```

Description

Defines the output line states of the PWM channel.

7.3 Constants

7.3.1 Error codes

A service may return one of the error codes, listed in [Table 2](#), if development error detection is enabled.

Table 2 Error codes

Name	Value	Description
PWM_E_INIT_FAILED	0x10	API Pwm_Init service called with wrong parameter.
PWM_E_UNINIT	0x11	API service used without module initialization.
PWM_E_PARAM_CHANNEL	0x12	API service used with an invalid channel identifier.
PWM_E_PERIOD_UNCHANGEABLE	0x13	Usage of unauthorized PWM service on PWM channel configured a fixed period.
PWM_E_ALREADY_INITIALIZED	0x14	API Pwm_Init service called, but module is already initialized.
PWM_E_PARAM_POINTER	0x15	API service is called with a NULL parameter.
PWM_E_NOT_DISENGAGED	0x16	API Pwm_SetPowerState is called while the PWM module is still in use.
PWM_E_POWER_STATE_NOT_SUPPORTED	0x17	The requested power state is not supported by the PWM module.
PWM_E_TRANSITION_NOT_POSSIBLE	0x18	The requested power state is not reachable from the current one.
PWM_E_PERIPHERAL_NOT_PREPARED	0x19	API Pwm_SetPowerState has been called without having called the API Pwm_PreparePowerState before.
PWM_E_WAITING_TRIGGER	0x52	API service is called when waiting trigger.
PWM_E_IDLE	0x53	API service is called while channel is in idle state.
PWM_E_PARAM_STATE	0x54	Invalid channel state.
PWM_E_DUTY_OUT_OF_RANGE	0x55	Duty value argument greater than 0x8000.

Name	Value	Description
PWM_E_PERIOD_OUT_OF_RANGE	0x56	Period value argument out of range.
PWM_E_PARAM_GROUP	0x57	API service used with an invalid group identifier.
PWM_E_PARAM_NOTIFICATION	0x58	API service used with an invalid notification type.
PWM_E_PARAM_TICK	0x59	Timer tick value is out of range.
PWM_E_PARAM_CLOCK	0x60	Clock frequency is out of range.
PWM_E_DIFFERENT_PRESCALER	0x61	Specified grouped channels have different prescaler values when <code>PwmGroupStartDelay</code> is enabled.

7.3.2 Version information

The version information listed in [Table 3](#) is published in the driver's header file.

Table 3 Version information

Name	Value	Description
PWM_AR_RELEASE_MAJOR_VERSION	4	Major version number (AUTOSAR)
PWM_AR_RELEASE_MINOR_VERSION	2	Minor version number (AUTOSAR)
PWM_AR_RELEASE_REVISION_VERSION	2	Patch version number (AUTOSAR)
PWM_SW_MAJOR_VERSION	See release notes	Vendor specific major version number
PWM_SW_MINOR_VERSION	See release notes	Vendor specific minor version number
PWM_SW_PATCH_VERSION	See release notes	Vendor specific patch version number

7.3.3 Module information

Table 4 Module information

Name	Value	Description
PWM_MODULE_ID	121	Module ID (<code>Pwm</code>)
PWM_VENDOR_ID	66	Vendor ID

7.3.4 API service IDs

Table 5 API service IDs

Name	Value	API name
PWM_API_INIT	0x0	<code>Pwm_Init</code>
PWM_API_DEINIT	0x1	<code>Pwm_DeInit</code>
PWM_API_SETDUTYCYCLE	0x2	<code>Pwm_SetDutyCycle</code>
PWM_API_SETPERIODANDDDUTY	0x3	<code>Pwm_SetPeriodAndDuty</code>
PWM_API_SETOUTPUTTOIDLE	0x4	<code>Pwm_SetOutputToIdle</code>
PWM_API_GETOUTPUTSTATE	0x5	<code>Pwm_GetOutputState</code>
PWM_API_DISABLENOTIFICATION	0x6	<code>Pwm_DisableNotification</code>
PWM_API_ENABLENOTIFICATION	0x7	<code>Pwm_EnableNotification</code>

Appendix A – API reference

Name	Value	API name
PWM_API_GETVERSIONINFO	0x8	Pwm_GetVersionInfo
PWM_API_SETPOWERSTATE	0x9	Pwm_SetPowerState
PWM_API_GETCURRENTPOWERSTATE	0xA	Pwm_GetCurrentPowerState
PWM_API_GETTARGETPOWERSTATE	0xB	Pwm_GetTargetPowerState
PWM_API_PREPAREPOWERSTATE	0xC	Pwm_PrepPowerState
PWM_API_MAIN_POWERTRANSITIONMANAGER	0xD	Pwm_Main_PowerTransitionManager
PWM_API_GETCHANNELSTATUS	0x20	Pwm_GetChannelStatus
PWM_API_INTERRUPTHANDLER	0x40	Pwm_InterruptHandler
PWM_API_STARTGROUPTRIGGER	0x41	Pwm_StartGroupTrigger
PWM_API_STOPGROUPTRIGGER	0x42	Pwm_StopGroupTrigger
PWM_API_SETCHANNELOUTPUT	0x43	Pwm_SetChannelOutput
PWM_API_SETPRESCALER	0x44	Pwm_SetPrescaler
PWM_API_SETOUTPUTOFFSET	0x45	Pwm_SetOutputOffset
PWM_API_SETTRIGGERDELAY	0x46	Pwm_SetTriggerDelay
PWM_API_ENABLETRIGGER	0x47	Pwm_EnableTrigger
PWM_API_DISABLETRIGGER	0x48	Pwm_DisableTrigger
PWM_API_SETDUTYCYCLEBUFFER	0x49	Pwm_SetDutyCycleBuffer
PWM_API_SETCHANNELOUTPUTBUFFER	0x4A	Pwm_SetChannelOutputBuffer
PWM_API_SETDUTYCHANNELOUTPUTBUFFER	0x4B	Pwm_SetDutyAndChannelOutputBuffer

7.3.5 Symbolic names

Table 6 Symbolic names

Name	Description
PwmConf_PwmChannel_<n>	Symbolic name for PWM channel <i>n</i> (<i>n</i> is channel's short name).
PwmConf_PwmChannelGroup_<n>	Symbolic name for PWM group <i>n</i> (<i>n</i> is channel group's short name).
PwmConf [Config Set] PwmPeriodDefault_<n>	Symbolic name for the ticks value of the default period of PWM channel <i>n</i> (<i>n</i> is channel's short name) in <i>Config Set</i> (<i>Config Set</i> is configuration set's short name).
PwmConf_PwmChannelConfigSet [Config Set]	Symbolic names for PWM <i>Config Set</i> (<i>Config Set</i> is configuration set's short name).

7.3.6 Channel status values

Table 7 Channel status values

Name	Value	API name
PWM_CH_COND_TIMER_DISABLED	0x1	The PWM channel timer is OFF.
PWM_CH_COND_TIMER_ENABLED	0x2	The PWM channel timer is ON.
PWM_CH_COND_IDLE_LOW	0x4	The PWM channel timer is in idle state LOW.
PWM_CH_COND_IDLE_HIGH	0x8	The PWM channel timer is in idle state HIGH.
PWM_CH_COND_WAIT_HWTRIGGER	0x10	The PWM channel timer has been started and is waiting HW trigger.
PWM_CH_COND_WAIT_UPDATE	0x20	The PWM channel timer has been started and is waiting update.

7.4 Functions

7.4.1 Pwm_Init

Syntax

```
void Pwm_Init
(
    const Pwm_ConfigType* ConfigPtr
)
```

Service ID

- 0x0

Parameters (in)

- ConfigPtr - Pointer to the configuration set.

Parameters (out)

None

Return value

None

DET errors

- PWM_E_ALREADY_INITIALIZED - The routine Pwm_Init() is called while the PWM driver and hardware is already initialized.
- PWM_E_INIT_FAILED - The routine Pwm_Init() is called with an invalid parameter ConfigPtr.

DEM errors

None

Description

`Pwm_Init()` is a service for PWM initialization.

7.4.2 Pwm_DeInit**Syntax**

```
void Pwm_DeInit
(
    void
)
```

Service ID

0x1

Parameters (in)

None

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.

DEM errors

None

Description

`Pwm_DeInit()` is a service for PWM deinitialization. It disables all PWM interrupts and notifications and sets the PWM output signals to idle level as specified in configuration data.

7.4.3 Pwm_SetDutyCycle**Syntax**

```
void Pwm_SetDutyCycle
(
    Pwm_ChannelType ChannelNumber,
    uint16 DutyCycle
)
```

Service ID

0x2

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.
- `DutyCycle` - Duty cycle time.

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_DUTY_OUT_OF_RANGE` - The passed duty cycle is out of range.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid.
- `PWM_E_WAITING_TRIGGER` - The channel is waiting for the trigger.

DEM errors

None

Description

`Pwm_SetDutyCycle()` sets the duty cycle of the PWM channel.

If the parameter `DutyCycle` of `Pwm_SetDutyCycle()` is `0x0000` == 0%, the PWM output state is negated `PwmPolarity`. If the parameter `DutyCycle` of `Pwm_SetDutyCycle()` is `0x8000` == 100%, the PWM output state is active level (`PwmPolarity`).

If the parameter `DutyCycle` of `Pwm_SetDutyCycle()` is > 0% and < 100%, the PWM output is in active state according to duty cycle and period parameters.

Note: *If this function is called while the value of `Status` of `Pwm_ChannelStatusType` has `PWM_CH_COND_WAIT_UPDATE`, the channel behavior depends on the configuration parameter `PwmMultipleUpdateInPeriod`.*

7.4.4 Pwm_SetPeriodAndDuty**Syntax**

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

Service ID

- 0x3

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.
- `Period` - Period time in ticks. The minimum value depends on `PwmHwTriggerOutputDefaultTime` (if configured). The maximum value is 65534.
- `DutyCycle` - Duty cycle time. The value is interpreted in the same way as by `Pwm_SetDutyCycle()`.

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PERIOD_UNCHANGEABLE` - PWM channel is not declared as a variable period type.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid.
- `PWM_E_DUTY_OUT_OF_RANGE` - The passed duty cycle is out of range.
- `PWM_E_PERIOD_OUT_OF_RANGE` - The passed period is out of range.
- `PWM_E_WAITING_TRIGGER` - The channel is waiting for the trigger.

DEM errors

None

Description

`Pwm_SetPeriodAndDuty()` sets the period and the duty cycle of a PWM channel.

If the parameter `DutyCycle` of `Pwm_SetPeriodAndDuty()` is `0x0000 == 0%`, the PWM output state is negated `PwmPolarity`. If the parameter `DutyCycle` of `Pwm_SetDutyCycle()` is `0x8000 == 100%`, the PWM output state is active level (`PwmPolarity`).

If the parameter `DutyCycle` of `Pwm_SetPeriodAndDuty()` is $> 0\%$ and $< 100\%$, the PWM output is in active state according to duty cycle and period parameters.

The period can only be updated, if the channel is configured to variable period and if the parameter period of `Pwm_SetPeriodAndDuty()` is ≥ 0 and ≤ 65534 . If the parameter period is zero, the hardware is set to its minimum period of one tick. The setting of the duty-cycle is not relevant in this case; the output is set to zero percent duty-cycle.

Since the hardware's minimum period is one tick, the parameter period's value of '0' or '1' set the same frequency.

If the parameter `PwmHwTriggerOutputScaled` is enabled, the delay of output trigger is scaled with the period by `Pwm_SetPeriodAndDuty()`.

If the parameter `PwmOutputOffsetScaled` is enabled, the rising edge offset is scaled with the period by `Pwm_SetPeriodAndDuty()`.

Note: *If this function is called while the value of `Status` of `Pwm_ChannelStatusType` has `PWM_CH_COND_WAIT_UPDATE`, the channel behavior depends on the configuration parameter `PwmMultipleUpdateInPeriod`.*

7.4.5 Pwm_SetOutputToIdle

Syntax

```
void Pwm_SetOutputToIdle  
(  
    Pwm_ChannelType ChannelNumber  
)
```

Service ID

- 0x4

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid.
- `PWM_E_WAITING_TRIGGER` - The channel is waiting for the trigger.

DEM errors

None

Description

`Pwm_SetOutputToIdle()` sets the PWM output state to the `PwmIdleState` level.

7.4.6 Pwm_GetOutputState

Syntax

```
Pwm_OutputStateType Pwm_GetOutputState  
(  
    Pwm_ChannelType ChannelNumber  
)
```

Service ID

- 0x5

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.

Parameters (out)

None

Return value

- `PWM_LOW` - Either PWM output level is low or a development error has occurred.
- `PWM_HIGH` - PWM output level is high.

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid.

DEM errors

None

Description

`Pwm_GetOutputState()` shall read the PWM output signal level and return it.

Note: There is a possibility of returning to the previous state when `PwmDutycycleUpdatedEndperiod` is `TRUE` and this API is called immediately after executing `Pwm_SetDutyCycle()`.

Note: There is a possibility of returning to the previous state when `PwmPeriodUpdatedEndperiod` is `TRUE` and this API is called immediately after executing `Pwm_SetPeriodAndDuty()`.

Note: If the value of those executed APIs is necessary, wait for the next period.

7.4.7 Pwm_DisableNotification

Syntax

```
void Pwm_DisableNotification
(
    Pwm_ChannelType ChannelNumber
)
```

Service ID

- 0x6

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid.

DEM errors

None

Description

This service disables the PWM signal edge notification.

7.4.8 Pwm_EnableNotification

Syntax

```
void Pwm_EnableNotification  
(  
    Pwm_ChannelType ChannelNumber,  
    Pwm_EdgeNotificationType Notification  
)
```

Service ID

- `0x7`

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.
- `Notification` - Type of notification edge can be `PWM_RISING_EDGE`, `PWM_FALLING_EDGE`, or `PWM_BOTH_EDGE`.

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid.
- `PWM_E_PARAM_NOTIFICATION` - `Notification` is invalid.

DEM errors

None

Description

This service enables the PWM signal edge notification according to parameter `Notification`.

7.4.9 Pwm_GetVersionInfo

Syntax

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

Service ID

- 0x8

Parameters (in)

None

Parameters (out)

- `versioninfo` - Pointer to the location where the version information of this module is stored.

Return value

None

DET errors

- `PWM_E_PARAM_POINTER` - `versioninfo` is NULL pointer.

DEM errors

None

Description

This service returns the version information of this module.

7.4.10 Pwm_GetChannelStatus

Syntax

```
Std_ReturnType Pwm_GetChannelStatus  
(  
    Pwm_DriverStatusType* DriverStatusPtr,  
    Pwm_ChannelStatusType* ChannelStatusPtr,  
    Pwm_ChannelType ChannelNumber  
)
```

Service ID

- 0x20

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.

Parameters (out)

- `DriverStatusPtr` - Pointer of driver status data.
- `ChannelStatusPtr` - Pointer to where to store the channel status parameters.

Return value

- `E_OK` - Software and hardware states are in sync.
- `E_NOT_OK` - Software and hardware states are inconsistent.

DET errors

- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid.
- `PWM_E_PARAM_POINTER` - API service is called with a NULL parameter.
- `PWM_E_UNINIT` - Configured data is abnormal.

DEM errors

None

Description

`Pwm_GetChannelStatus()` reads out internal registers and checks consistency of the software and hardware status. This is a vendor-specific function.

Note: `Pwm_GetChannelStatus()` may return `E_NOT_OK` if the hardware status has not yet changed to the running after `Pwm_Init()`, `Pwm_SetDutyCycle()`, `Pwm_SetPeriodAndDuty()`, and `Pwm_StartGroupTrigger()` for the stopped channel.

Note: `Pwm_GetChannelStatus()` also may return `E_NOT_OK` if the hardware status has not yet changed to the running after `Pwm_DisableTrigger()`, `Pwm_EnableTrigger()`, `Pwm_SetDutyCycle()`, `Pwm_SetPeriodAndDuty()`, `Pwm_SetOutputOffset()`, `Pwm_SetTriggerDelay()`, `Pwm_SetPrescaler()` for the running channel.

Note: It may occur when the tick frequency of the PWM channel is very slow.

Note: If this API returned `E_NOT_OK`, retry after waiting for one tick time of the PWM timer.

7.4.11 Pwm_StartGroupTrigger

Syntax

```
void Pwm_StartGroupTrigger
(
    Pwm_ChannelGroupType ChannelGroupNumber
)
```

Service ID

- 0x41

Parameters (in)

- `ChannelGroupNumber` - Numeric identifier of the PWM channel group.

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_GROUP` - `ChannelGroupNumber` is invalid.
- `PWM_E_WAITING_TRIGGER` - The channel is waiting for the trigger.
- `PWM_E_DIFFERENT_PRESCALER` - Specified grouped channels have different prescaler values when `PwmGroupStartDelay` is enabled.

DEM errors

None

Description

`Pwm_StartGroupTrigger()` starts the output by the channel group. This is a vendor-specific function!

7.4.12 Pwm_StopGroupTrigger

Syntax

```
void Pwm_StopGroupTrigger
(
    Pwm_ChannelGroupType ChannelGroupNumber
)
```

Service ID

- 0x42

Parameters (in)

- `ChannelGroupNumber` - Numeric identifier of the PWM channel group.

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_GROUP` - `ChannelGroupNumber` is invalid.
- `PWM_E_WAITING_TRIGGER` - The channel is waiting for the trigger.

DEM errors

None

Description

`Pwm_StopGroupTrigger()` stops the output by the channel group. `Pwm_StopGroupTrigger()` sets the PWM output state to the `PwmIdleState` level for each channel. This is a vendor-specific function.

7.4.13 Pwm_SetChannelOutput

Syntax

```
void Pwm_SetChannelOutput  
(  
    Pwm_ChannelType ChannelNumber,  
    Pwm_OutputStateType State  
)
```

Service ID

0x43

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.
- `State` - PWM output state.

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid.
- `PWM_E_PARAM_STATE` - `State` is invalid.
- `PWM_E_WAITING_TRIGGER` - The channel is waiting for the trigger.
- `PWM_E_IDLE` - The channel is idle state.

DEM errors

None

Description

`Pwm_SetChannelOutput()` sets the PWM output to the specified state.

This is a vendor-specific function.

Note: If the specified channel is called by the `Pwm_SetChannelOutput` API at the end of the period timing in the waiting update state, the output waveform might be disturbed.

In the following cases, the channel is set to the waiting update state (`PWM_CH_COND_WAIT_UPDATE`) until the next period.

Appendix A – API reference

- Calling `Pwm_SetTriggerDelay`, `Pwm_SetOutputOffset` or `Pwm_SetChannelOutput`
- Calling `Pwm_Init` or `Pwm_StartGroupTrigger` with any start delay configured channel
- Calling `Pwm_SetPeriodAndDuty` (`PwmPeriodUpdatedEndperiod` is `TRUE`)
- Calling `Pwm_SetDutyCycle` (`PwmDutycycleUpdatedEndperiod` is `TRUE`)

Workaround:

Call the `Pwm_SetChannelOutput` API after the next period.

7.4.14 Pwm_SetPrescaler

Syntax

```
void Pwm_SetPrescaler
(
    Pwm_ChannelType ChannelNumber,
    Pwm_ClkFrequencyType ClockFrequency
)
```

Service ID

- 0x44

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.
- `ClockFrequency` - Input clock frequency.

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid.
- `PWM_E_PARAM_CLOCK` - `ClockFrequency` is invalid.
- `PWM_E_WAITING_TRIGGER` - The channel is waiting for the trigger.

DEM errors

None

Description

`Pwm_SetPrescaler()` sets prescaler for the specified channel. This is a vendor-specific function.

Note: *If this function is called while the value of Status of `Pwm_ChannelStatusType` has `PWM_CH_COND_TIMER_ENABLED`, the channel is restarted and `PwmChannelStartDelay` is canceled.*

7.4.15 Pwm_SetOutputOffset

Syntax

```
void Pwm_SetOutputOffset  
(  
    Pwm_ChannelType ChannelNumber,  
    Pwm_PeriodType OffsetTick  
)
```

Service ID

- 0x45

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.
- `OffsetTick` - Offset time for PWM output. The maximum value is 65533 at the period maximum value (65534).

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid or `PwmOutputOffset` is disabled.
- `PWM_E_PARAM_TICK` - `OffsetTick` is invalid.
- `PWM_E_WAITING_TRIGGER` - The channel is waiting for the trigger.

DEM errors

None

Description

`Pwm_SetOutputOffset()` sets offset the PWM duty output.

This is a vendor-specific function.

Note: *If this function is called while the value of `Status` of `Pwm_ChannelStatusType` has `PWM_CH_COND_WAIT_UPDATE`, the channel behavior depends on the configuration parameter `PwmMultipleUpdateInPeriod`.*

Note: *`OffsetTick` can set the value from 0 to (period ticks - duty ticks - 1) when duty is not 0% nor 100%.*

Note: *`OffsetTick` can set the value from 0 to (period ticks - 1) when duty is 0% or 100%.*

7.4.16 Pwm_SetTriggerDelay

Syntax

```
void Pwm_SetTriggerDelay  
(  
    Pwm_ChannelType ChannelNumber,  
    Pwm_PeriodType TriggerTicks  
)
```

Service ID

- 0x46

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.
- `TriggerTicks` - Delay ticks for the output trigger. The maximum value is 65534 at the period maximum value (65534).

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid or `PwmOutputOffset` is enabled.
- `PWM_E_PARAM_TICK` - `TriggerTicks` is invalid.
- `PWM_E_WAITING_TRIGGER` - The channel is waiting for the trigger.

DEM errors

None

Description

`Pwm_SetTriggerDelay()` sets delay ticks for the output trigger.

This is a vendor-specific function.

Note: *If this function is called while the value of `Status` of `Pwm_ChannelStatusType` has `PWM_CH_COND_WAIT_UPDATE`, the channel behavior depends on the configuration parameter `PwmMultipleUpdateInPeriod`.*

7.4.17 Pwm_EnableTrigger

Syntax

```
void Pwm_EnableTrigger  
(  
    Pwm_ChannelType ChannelNumber  
)
```

Service ID

- 0x47

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid.
- `PWM_E_WAITING_TRIGGER` - The channel is waiting for the trigger.

DEM errors

None

Description

`Pwm_EnableTrigger()` enables the output trigger on the specified channel. This is a vendor-specific function.

Note: *If this function is called while the value of `Status` of `Pwm_ChannelStatusType` has `PWM_CH_COND_TIMER_ENABLED`, the channel is restarted and `PwmChannelStartDelay` is canceled.*

7.4.18 Pwm_DisableTrigger

Syntax

```
void Pwm_DisableTrigger  
(  
    Pwm_ChannelType ChannelNumber  
)
```

Service ID

0x48

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid.
- `PWM_E_WAITING_TRIGGER` - The channel is waiting for the trigger.

DEM errors

None

Description

`Pwm_DisableTrigger()` disables the output trigger on the specified channel. This is a vendor-specific function.

Note: *If this function is called while the value of `Status` of `Pwm_ChannelStatusType` has `PWM_CH_COND_TIMER_ENABLED`, the channel is restarted and `PwmChannelStartDelay` is canceled.*

7.4.19 Pwm_SetDutyCycleBuffer

Syntax

```
void Pwm_SetDutyCycleBuffer  
(  
    Pwm_ChannelType ChannelNumber,  
    uint16 DutyCycle  
)
```

Service ID

0x49

Parameters (in)

- `ChannelNumber` - Numeric identifier of the PWM channel.
- `DutyCycle` - Duty cycle time.

Parameters (out)

None

Return value

None

DET errors

- `PWM_E_UNINIT` - Driver is not yet initialized.
- `PWM_E_PARAM_CHANNEL` - `ChannelNumber` is invalid.
- `PWM_E_DUTY_OUT_OF_RANGE` - The passed duty cycle is out of range.
- `PWM_E_WAITING_TRIGGER` - The channel is waiting for the trigger.

DEM errors

None

Description

`Pwm_SetDutyCycleBuffer()` : Prepares to change the duty cycle of the PWM channel. The line state and complementary line state are retained. To use this setting, call the `Port_ActTrigger` API after this API.

This is a vendor-specific function.

Note: *Do not call `Pwm_SetDutyCycleBuffer()` and following APIs in the same period:
`Pwm_SetPeriodAndDuty`, `Pwm_SetDutyCycle`, `Pwm_SetChannelOutput`,
`Pwm_SetOutputOffset`, `Pwm_SetTriggerDelay`.*

Note: *If this function is called again after the `Port_ActTrigger` API is called and before the end of the period, the result is not guaranteed.*

Note: *Call the `Port_ActTrigger` API when the channel is running.*

7.4.20 Pwm_SetChannelOutputBuffer

Syntax

```
void Pwm_SetChannelOutputBuffer  
(  
    Pwm_ChannelType ChannelNumber,  
    Pwm_OutputLineStateType LineState  
    Pwm_OutputLineStateType LineCompState  
)
```

Service ID

0x4a

Parameters (in)

- ChannelNumber - Numeric identifier of the PWM channel.
- LineState - PWM line output state.
- LineCompState - PWM line complementary output state.

Parameters (out)

None

Return value

None

DET errors

- PWM_E_UNINIT - Driver is not yet initialized.
- PWM_E_PARAM_CHANNEL - ChannelNumber is invalid.
- PWM_E_PARAM_STATE - LineState or LineCompState is invalid.
- PWM_E_WAITING_TRIGGER - The channel is waiting for the trigger.

DEM errors

None

Description

`Pwm_SetChannelOutputBuffer()` : Prepares to change the line state and complementary line state of the PWM channel. The duty cycle is retained. To use this setting, call the `Port_ActTrigger` API after this API. This is a vendor-specific function.

Note: Do not call `Pwm_SetChannelOutputBuffer()` and following APIs in the same period:
`Pwm_SetPeriodAndDuty`, `Pwm_SetDutyCycle`, `Pwm_SetChannelOutput`,
`Pwm_SetOutputOffset`, `Pwm_SetTriggerDelay`.

Note: If this function is called again after the `Port_ActTrigger` API is called and before the end of the period, the result is not guaranteed.

Note: Call the `Port_ActTrigger` API when the channel is running.

Note: The channel sets the configuration parameter *PwmSetOutputEnable* ON.

Note: The edge of the *Pwm_EnableNotification* API is set as if *LineState* is always PWM. Therefore, it may be different from the edge of the output waveform.

7.4.21 Pwm_SetDutyAndChannelOutputBuffer

Syntax

```
void Pwm_SetDutyAndChannelOutputBuffer
(
    Pwm_ChannelType ChannelNumber,
    uint16 DutyCycle,
    Pwm_OutputLineStateType LineState
    Pwm_OutputLineStateType LineCompState
)
```

Service ID

0x4b

Parameters (in)

- *ChannelNumber* - Numeric identifier of the PWM channel.
- *DutyCycle* - Duty cycle time.
- *LineState* - PWM line output state.
- *LineCompState* - PWM line complementary output state.

Parameters (out)

None

Return value

None

DET errors

- *PWM_E_UNINIT* - Driver is not yet initialized.
- *PWM_E_PARAM_CHANNEL* - *ChannelNumber* is invalid.
- *PWM_E_DUTY_OUT_OF_RANGE* - The passed duty cycle is out of range.
- *PWM_E_PARAM_STATE* - *LineState* or *LineCompState* is invalid.
- *PWM_E_WAITING_TRIGGER* - The channel is waiting for the trigger.

DEM errors

None

Description

Pwm_SetDutyAndChannelOutputBuffer() : Prepares to change the duty cycle, line state and complementary line state of the PWM channel. To use this setting, call the *Port_ActTrigger* API after this API.

This is a vendor-specific function.

Appendix A – API reference

- Note:** Do not call `Pwm_SetDutyAndChannelOutputBuffer()` API and following APIs in the same period: `Pwm_SetPeriodAndDuty`, `Pwm_SetDutyCycle`, `Pwm_SetChannelOutput`, `Pwm_SetOutputOffset`, `Pwm_SetTriggerDelay`.
- Note:** If this function is called again after the `Port_ActTrigger` API is called and before the end of the period, the result is not guaranteed.
- Note:** Call the `Port_ActTrigger` API with this API is effect when the channel is running.
- Note:** The channel sets the configuration parameter `PwmSetOutputEnable` ON.
- Note:** The edge of the `Pwm_EnableNotification` API is set as if `LineState` is always PWM. Therefore, it may be different from the edge of the output waveform.

7.5 Required callback functions

7.5.1 DET

If development error detection is enabled, the PWM driver uses the following callback function provided by DET. If you do not use DET, you must implement this function within your application.

Det_ReportError

Syntax

```
Std_ReturnType Det_ReportError
(
    uint16 ModuleId,
    uint8 InstanceId,
    uint8 ApiId,
    uint8 ErrorId
)
```

Reentrancy

- Reentrant

Parameters (in)

- `ModuleId` - Module ID of calling module.
- `InstanceId` - Instance ID of calling module.
- `ApiId` - ID of the API service that calls this function.
- `ErrorId` - ID of the detected development error.

Return value

- Returns always `E_OK`.

Description

Service for reporting development errors.

7.5.2 DEM

The PWM driver does not report to DEM.

7.5.3 PWM notifications

The following notification functions are used by the PWM driver to inform other software modules about the change of the PWM output signal.

The callback notification names are statically configurable by the PWM driver configuration via the `PwmNotification` parameter.

The notifications are enabled by calling `Pwm_EnableNotification()`.

The notifications are stopped by calling `Pwm_DisableNotification()`.

Pwm_Notification_<#Channel>

Syntax

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

Parameters (in)

None

Parameters (out)

None

Return value

None

Description

`Pwm_Notification_<#Channel>()` is called within the appropriate interrupt context after the requested edge type occurs on the corresponding PWM output signal.

7.5.4 Callout functions

Error callout API

The AUTOSAR PWM module requires an error callout handler. Each error is reported to this handler, error checking cannot be switched OFF. The name of the function to be called can be configured by the `PwmErrorCalloutFunction` parameter.

Syntax

```
void Error_Handler_Name
(
    uint16 ModuleId,
    uint8 InstanceId,
    uint8 ApiId,
    uint8 ErrorId
)
```

Reentrancy

- Reentrant

Parameters (in)

- `ModuleId` - Module ID of calling module.
- `InstanceId` - Instance ID of calling module.
- `ApiId` - ID of the API service that calls this function.
- `ErrorId` - ID of the detected error.

Return value

None

Description

Service for reporting errors.

Appendix B – Access register table

TCPWM

Table 8 TCPWM access register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
CTRL	31:0	Word (32 bits)	Depends on configuration value or API.	Counter control register	Pwm_DeInit Pwm_Init Pwm_EnableTrigger Pwm_DisableTrigger Pwm_StartGroupTrigger Pwm_StopGroupTrigger Pwm_SetPrescaler Pwm_SetOutputToIdle Pwm_SetPeriodAndDuty Pwm_SetDutyCycle Pwm_SetChannelOutput Pwm_SetOutputOffset Pwm_SetTriggerDelay Pwm_SetDutyCycleBuffer Pwm_SetChannelOutputBuffer Pwm_SetDutyAndChannelOutputBuffer	0xC40336FF (PwmTimer) 0x00000000 (Monitoring is not needed for PwmStartDelayTimer.)	0x*400*6** (PwmTimer: After Pwm_Init, digit * depends on configuration value and API.) 0x0000*0F0 (PwmTimer: After Pwm_DeInit, digit * depends on configuration value and API.) 0x00000000 (Monitoring is not needed for PwmStartDelayTimer.)
STATUS	31:0	Word (32 bits)	-	Counter status register	Read only.	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)

Appendix B – Access register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
COUNTER	31:0	Word (32 bits)	Counter value.	Counter count register	Pwm_DeInit Pwm_Init Pwm_EnableTrigger Pwm_DisableTrigger Pwm_SetPeriodAndDuty Pwm_SetDutyCycle Pwm_StartGroupTrigger Pwm_SetOutputOffset Pwm_SetTriggerDelay Pwm_SetPrescaler	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
CC0	31:0	Word (32 bits)	Duty tick value.	Counter compare/capture 0 register	Pwm_DeInit Pwm_Init Pwm_EnableTrigger Pwm_DisableTrigger Pwm_SetPeriodAndDuty Pwm_SetDutyCycle Pwm_StartGroupTrigger Pwm_SetOutputOffset Pwm_SetTriggerDelay Pwm_SetPrescaler	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
CC0_BUFF	31:0	Word (32 bits)	Duty tick value.	Counter buffered compare/capture 0 register	Pwm_DeInit Pwm_Init Pwm_EnableTrigger Pwm_DisableTrigger Pwm_SetPeriodAndDuty Pwm_SetDutyCycle Pwm_StartGroupTrigger Pwm_SetOutputOffset Pwm_SetTriggerDelay Pwm_SetPrescaler Pwm_SetChannelOutput Pwm_SetDutyCycleBuffer Pwm_SetDutyAndChannelOutputBuffer	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)

Appendix B – Access register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
CC1	31:0	Word (32 bits)	Trigger tick value.	Counter compare/capture 1 register	Pwm_DeInit Pwm_Init Pwm_EnableTrigger Pwm_DisableTrigger Pwm_SetPeriodAndDuty Pwm_SetDutyCycle Pwm_StartGroupTrigger Pwm_SetOutputOffset Pwm_SetTriggerDelay Pwm_SetPrescaler	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
CC1_BUFF	31:0	Word (32 bits)	Trigger tick value.	Counter buffered compare/capture 1 register	Pwm_DeInit Pwm_Init Pwm_EnableTrigger Pwm_DisableTrigger Pwm_SetPeriodAndDuty Pwm_SetDutyCycle Pwm_StartGroupTrigger Pwm_SetOutputOffset Pwm_SetTriggerDelay Pwm_SetPrescaler Pwm_SetChannelOutput Pwm_SetDutyCycleBuffer Pwm_SetDutyAndChannelOutputBuffer	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
PERIOD	31:0	Word (32 bits)	Period tick value.	Counter period register	Pwm_DeInit Pwm_Init Pwm_EnableTrigger Pwm_DisableTrigger Pwm_SetPeriodAndDuty Pwm_SetDutyCycle Pwm_StartGroupTrigger Pwm_SetOutputOffset Pwm_SetTriggerDelay Pwm_SetPrescaler	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)

Appendix B – Access register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
PERIOD_BUFF	31:0	Word (32 bits)	Period tick value.	Counter buffered period register	Pwm_DeInit Pwm_Init Pwm_EnableTrigger Pwm_DisableTrigger Pwm_SetPeriodAndDuty Pwm_SetDutyCycle Pwm_StartGroupTrigger Pwm_SetOutputOffset Pwm_SetTriggerDelay Pwm_SetPrescaler Pwm_SetChannelOutput	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
LINE_SEL	31:0	Word (32 bits)	0x00000032, 0x00000010, 0x00000001.	Counter line selection register	Pwm_DeInit Pwm_Init Pwm_EnableTrigger Pwm_DisableTrigger Pwm_SetPeriodAndDuty Pwm_SetDutyCycle Pwm_StartGroupTrigger Pwm_SetOutputOffset Pwm_SetTriggerDelay Pwm_SetPrescaler Pwm_SetChannelOutput	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
LINE_SEL_BUFF	31:0	Word (32 bits)	0x00000032, 0x00000001, 0x00000010.	Counter buffered line selection register	Pwm_DeInit Pwm_Init Pwm_EnableTrigger Pwm_DisableTrigger Pwm_SetPeriodAndDuty Pwm_SetDutyCycle Pwm_StartGroupTrigger Pwm_SetOutputOffset Pwm_SetTriggerDelay Pwm_SetPrescaler Pwm_SetChannelOutput Pwm_SetChannelOutputBuffer	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)

Appendix B – Access register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
					Pwm_SetDutyAndChannelOutputBuffer		
DT	31:0	Word (32 bits)	0x00000000 Divider value.	Counter PWM dead time register	Pwm_DeInit Pwm_Init Pwm_SetPrescaler	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
TR_CMD	31:0	Word (32 bits)	Depends on configuration value or API.	Counter trigger command register	Pwm_Init Pwm_SetPeriodAndDuty Pwm_SetDutyCycle Pwm_StartGroupTrigger Pwm_DisableTrigger Pwm_EnableTrigger Pwm_SetPrescaler Pwm_SetOutputOffset Pwm_SetChannelOutput Pwm_SetTriggerDelay	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
TR_IN_SEL0	31:0	Word (32 bits)	0x00000100 Stop trigger select value << 24 Reload trigger select value << 16 Capture0 trigger select value.	Counter input trigger selection register 0	Pwm_DeInit Pwm_Init Pwm_StartGroupTrigger	0xFFFFFFFF	0x00000100
TR_IN_SEL1	31:0	Word (32 bits)	0x00000000.	Counter input trigger selection register 1	Pwm_DeInit Pwm_Init	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)

Appendix B – Access register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
TR_IN_EDGE_SEL	31:0	Word (32 bits)	Depends on configuration value.	Counter input trigger edge selection register	Pwm_DeInit Pwm_Init Pwm_StartGroupTrigger	0x00000FFF	0x00000F0C
TR_PWM_CTRL	31:0	Word (32 bits)	Depends on configuration value.	Counter trigger PWM control register	Pwm_DeInit Pwm_Init	0x000000FF	0x00000000 (Monitoring is not needed.)
TR_OUT_SEL	31:0	Word (32 bits)	Depends on configuration value.	Counter output trigger selection register	Pwm_DeInit Pwm_Init Pwm_DisableTrigger Pwm_EnableTrigger	0x00000077	0x00000000 (Monitoring is not needed.)
INTR	31:0	Word (32 bits)	Depends on configuration value or API.	Interrupt request register	Pwm_DeInit Pwm_Init Pwm_Isr_Vector_[VectorNumber]_Cat1 Pwm_Isr_Vector_[VectorNumber]_Cat2	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
INTR_SET	31:0	Word (32 bits)	-	Interrupt set request register	Do not use.	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)
INTR_MASK	31:0	Word (32 bits)	0x00000000, 0x00000001, 0x00000002, 0x00000003.	Interrupt mask register	Pwm_DeInit Pwm_Init Pwm_EnableNotification Pwm_DisableNotification Pwm_SetPeriodAndDuty Pwm_SetDutyCycle	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)

Appendix B – Access register table

Register	Bit No.	Access size	Value	Description	Timing	Mask value	Monitoring value
INTR_MASKED	31:0	Word (32 bits)	-	Interrupt masked request register	Read only.	0x00000000 (Monitoring is not needed.)	0x00000000 (Monitoring is not needed.)

Revision history

Revision	Issue date	Description of change
**	06/14/2018	Initial release
*A	12/20/2018	<p>Changed from PwmStartTriggerSelect to PwmStartTriggerSelect0 and PwmStartTriggerSelect1 in chapter 2.3, 3.4.1 and 4.3.1.</p> <p>Added the description of PwmStartDelayTrigger in chapter 3.4.1.</p> <p>Added the description in chapter 3.4.6.</p> <p>Changed Remarks of PwmChannelClass in chapter 4.2.2.</p> <p>Changed Remarks of PwmPeriodDeault in chapter 4.2.2.</p> <p>Changed PwmAdcTriggerDefaultType to PwmHwTriggerOutputUnitType in chapter 4.3.1.</p> <p>Changed Remarks of PwmStartTriggerSelect0 and PwmStartTriggerSelect1 in chapter 4.3.1.</p> <p>Changed Range and Remarks of PwmTimer in chapter 4.3.2.</p> <p>Changed PwmHardwareTriggeredAdc to PwmHwTriggerOutputLine in chapter 4.3.2.</p> <p>Changed PwmAdcTriggerFactor to PwmHwTriggerOutputFactor and added Remarks in chapter 4.3.2.</p> <p>Changed PwmAdcTriggerDefault to PwmHwTriggerOutputDefaultTime in chapter 4.3.2.</p> <p>Changed PwmAdcTriggerDefaultTick to PwmHwTriggerOutputDefaultTick in chapter 4.3.2.</p> <p>Changed Remarks of PwmChannelStartDelay and PwmStartAtInit in chapter 4.3.2.</p> <p>Changed Remarks and Range of PwmOutputOffset in chapter 4.3.2.</p> <p>Added PwmStartDelayTimer, PwmStartDelayMcuClockReferencePoint, PwmStartDelayTrigger and PwmOutputOffsetScaled in chapter 4.3.2.</p> <p>Changed Range and Remarks of PwmChannelGroupStartTrigger and PwmChannelGroupStopTrigger in chapter 4.3.3.</p> <p>Changed Description of PwmEnableDisableTriggerApi in chapter 4.3.4.</p> <p>Removed PwmOutputOffset and Pwm_SetOutputOffset() in chapter 5.9.</p> <p>Changed the title in chapter 5.9.</p> <p>Changed the title and description in chapter 5.10.</p> <p>Added PWM_E_WAITING_TRIGGER and PWM_E_IDLE in chapter 5.14.</p> <p>Added Note in chapter 5.16.</p> <p>Changed resource name in chapter 6.1.</p> <p>Changed description in chapter 6.4.</p> <p>Added PwmStartDelayTrigger settings in chapter 6.4.</p> <p>Added PWM_E_WAITING_TRIGGER and PWM_E_IDLE in chapter A.1.3.</p> <p>Added Table A-6 in chapter A.1.3.</p> <p>Changed in chapter A.1.4.</p> <p>Changed in chapter B.1.1.</p> <p>Updated Hardware Documentation Information</p>

Revision history

Revision	Issue date	Description of change
		Changed the reference number of Layered Software Architecture in chapter 1.3
*B	06/14/2019	Updated hardware documentation information.
*C	07/08/2019	Deleted wrong description of Pwm_SetPeriodAndDuty in chapter A.1.4.
*D	10/30/2019	<p>Updated Remarks of PwmChannelClass in chapter 4.2.2.</p> <p>Added PwmMultipleUpdateInPeriod in chapter 4.3.1.</p> <p>Updated Range and Remarks of PwmTimer in chapter 4.3.2.</p> <p>Updated Remarks of PwmChannelStartDelay in chapter 4.3.2.</p> <p>Added PwmChannelPrescale in chapter 4.3.2.</p> <p>Updated Remarks of PwmStartDelayTimer in chapter 4.3.2.</p> <p>Updated Remarks of PwmStartDelayMcuClockReferencePoint in chapter 4.3.2.</p> <p>Updated Remarks of PwmStartDelayTrigger in chapter 4.3.2.</p> <p>Added PwmHwTriggerOutputScaled in chapter 4.3.2.</p> <p>Updated Remarks of PwmStartAtInit in chapter 4.3.2.</p> <p>Updated Remarks of PwmChannelGroupStartTrigger in chapter 4.3.3.</p> <p>Updated Remarks of PwmChannelGroupStopTrigger in chapter 4.3.3.</p> <p>Updated Remarks of PwmChannelRef in chapter 4.3.3.</p> <p>Added PwmGroupStartDelayTimer in chapter 4.3.4.</p> <p>Added PwmGroupStartDelayMcuClockReferencePoint in chapter 4.3.4.</p> <p>Added PwmGroupStartDelayTrigger in chapter 4.3.4.</p> <p>Added chapter 5.9 PWM Start Delay.</p> <p>Added Pwm_StartGroupTrigger parameter checking in chapter 5.15.</p> <p>Added PwmGroupStartDelayTrigger settings in chapter 6.4.</p> <p>Added PWM_E_DIFFERENT_PRESCALER in chapter A.1.3.</p> <p>Updated Note of Pwm_SetDutyCycle in chapter A.1.4.</p> <p>Updated Description and Note of Pwm_SetPeriodAndDuty in chapter A.1.4.</p> <p>Updated DET Errors of Pwm_StartGroupTrigger in chapter A.1.4.</p> <p>Updated Note of Pwm_SetOutputOffset in chapter A.1.4.</p> <p>Updated Note of Pwm_SetTriggerDelay in chapter A.1.4.</p> <p>Updated Monitoring Value of CTRL and Timing of TR_IN_SEL1 in chapter B.1.1.</p>
*E	08/19/2020	<p>Changed a memmap file include folder in chapter 2.6.</p> <p>Updated description in chapter 3.4.6.</p> <p>Updated Remarks of PwmTimer in chapter 4.3.2.</p>
*F	2020-11-19	MOVED TO INFINEON TEMPLATE.
*G	2021-02-18	<p>Added note in chapter 5.5.</p> <p>Corrected numbering in chapter 7.</p> <p>Added note in chapter 7.4.13.</p>
*H	2021-06-18	<p>Added the following chapters for supporting stepping motor control:</p> <ul style="list-style-type: none"> Chapter 4.3.2.18 PwmUpdateOutputAtInitEnable Chapter 4.3.2.19 PwmOutputAtInitSelect Chapter 4.3.2.20 PwmCompOutputAtInitSelect

Revision history

Revision	Issue date	Description of change
		<ul style="list-style-type: none"> Chapter 4.3.3.5 PwmChannelGroupSwitchEventTrigger Chapter 4.3.5.7 PwmSetDutyCycleBufferApi Chapter 4.3.5.8 PwmSetDutyAndChannelOutputBufferApi Chapter 4.3.5.9 PwmSetChannelOutputBufferApi Chapter 7.2.10 Pwm_OutputLineStateType Chapter 7.4.19 Pwm_SetDutyCycleBuffer Chapter 7.4.20 Pwm_SetChannelOutputBuffer Chapter 7.4.21 Pwm_SetDutyAndChannelOutputBuffer <p>Updated the following chapters for supporting stepping motor control: Added PwmChannelGroupSwitchEventTrigger condition and improved readability in error conditions and warning conditions:</p> <ul style="list-style-type: none"> Chapter 4.3.1.5 PwmStartTriggerSelect0 Chapter 4.3.1.6 PwmStartTriggerSelect1 Chapter 4.3.2.9 PwmStartDelayTrigger Chapter 4.3.3.2 PwmChannelGroupStartTrigger Chapter 4.3.3.3 PwmChannelGroupStopTrigger Chapter 4.3.3.4 PwmChannelRef Chapter 4.3.4.3 PwmGroupStartDelayTrigger <p>Added Pwm_SetDutyCycleBuffer, Pwm_SetChannelOutputBuffer and Pwm_SetDutyAndChannelOutputBuffer API description:</p> <ul style="list-style-type: none"> Chapter 5.15 API Parameter Checking Chapter 5.17 Reentrancy Chapter 6.4 Triggers <p>Chapter 8.1. TCPWM</p>
*I	2021-08-18	<p>Added a note on DeepSleep mode in 5.14 Sleep mode</p> <p>Added a note on Arm® errata in 6.3 Interrupts</p>
*J	2021-12-22	Updated to the latest branding guidelines.

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-12-22

Published by

Infineon Technologies AG

81726 Munich, Germany

© 2021 Infineon Technologies AG.

All Rights Reserved.

Do you have a question about this document?

Go to www.cypress.com/support

Document reference

002-23351 Rev. *J

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.