User Manual

for S32K1 PWM Driver

Document Number: UM2PWMASR4.4 Rev0000R1.0.1 Rev. 1.0

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	24.02.2022	NXP RTD Team	Prepared for release RTD S32K1 Version 1.0.1

Chapter 2

Introduction

This User Manual describes NXP Semiconductor AUTOSAR PWM for S32K1. AUTOSAR PWM driver configuration parameters and deviations from the specification are described in Driver chapter of this document. AUTOSAR PWM driver requirements and APIs are described in the AUTOSAR PWM driver software specification document.

2.1 Supported Derivatives

The software described in this document is intended to be used with the following microcontroller devices of NXP Semiconductors:

- s32k116_qfn32
- s32k116_lqfp48
- s32k118_lqfp48
- s32k118_lqfp64
- s32k142_lqfp48
- s32k142_lqfp64
- s32k142_lqfp100
- s32k142w_lqfp48
- s32k142w_lqfp64
- s32k144_lqfp48

- s32k144_lqfp64
- s32k144_lqfp100
- s32k144_mapbga100
- s32k144w_lqfp48
- s32k144w_lqfp64
- s32k146_lqfp64
- s32k146_lqfp100
- s32k146_mapbga100
- s32k146_lqfp144
- s32k148_lqfp100
- s32k148_mapbga100
- s32k148_lqfp144
- s32k148_lqfp176

All of the above microcontroller devices are collectively named as S32K1.

2.2 Overview

AUTOSAR (AUTomotive Open System ARchitecture) is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

AUTOSAR:

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.
- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".
- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.

2.3 About This Manual

This Technical Reference employs the following typographical conventions:

- **Boldface** style: Used for important terms, notes and warnings.
- *Italic* style: Used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

Note

> This is a note.

Warning

> This is a warning

2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CS	Chip Select
CTU	Cross Trigger Unit
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
FIFO	First In First Out
LSB	Least Signifigant Bit
MCU	Micro Controller Unit
MIDE	Multi Integrated Development Environment
MSB	Most Significant Bit
N/A	Not Applicable
RAM	Random Access Memory
SIU	Systems Integration Unit
SWS	Software Specification
VLE	Variable Length Encoding
XML	Extensible Markup Language

2.5 Reference List

#	Title	Version
1	Specification of PWM Driver	AUTOSAR Release 4.↩ 4.0
2	S32K1xx Series Reference Manual	Rev. 14, 09/2021
3	S32K1xx Data Sheet	Rev. 14, 08/2021
4	S32K116_0N96V	Rev. 22/OCT/2021
5	S32K118_0N97V	Rev. 22/OCT/2021
6	S32K142_0N33V	Rev. 22/OCT/2021
7	S32K144_0N57U	Rev. 22/OCT/2021
8	S32K144W_0P64A	Rev. 22/OCT/2021
9	S32K146_0N73V	Rev. 22/OCT/2021
10	S32K148_0N20V	Rev. 22/OCT/2021

Chapter 3

Driver

3.1 Requirements

Requirements for this driver are detailed in the Autosar Driver Software Specification document (See Table Reference List).

3.2 Driver Design Summary

The driver provides functions for initialization and control of the microcontroller internal PWM stage (pulse width modulation). The PWM module generates pulses with variable pulse width. It allows the selection of the duty cycle and the signal period time. Each PWM channel is linked to a hardware channel capable of implementing PWM functionality, which belongs to the microcontroller.

The S32K1 microcontrollers contains FlexTimer(FTM) and Flexio.

The FTM module has the following major features in this release:

- FTM source clock is selectable with prescaler option.
- 16-bit counter.

- Each channel can be configured for edge-aligned or center-aligned PWM mode.
- Each pair of channels can be combined to generate a PWM signal with independent control of both edges of PWM signal.
- The FTM channels can operate as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs.
- The deadtime insertion is available for each complementary pair.

The Flexio module has the following major features in this release:

- Up to 4 channels chosen among the Flexio timers.
- Up to 8 pins that can be chosen for any channel.
- Timer 8-bit High PWM Mode.

3.3 Hardware Resources

Ftm's resource

Derivatives	Pwm Module	Pwm Channels available	Fault channel
s32k116_qfn32	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6	FLT_2
	Ftm 1	Channel 0, Channel 1, Channel 4, Channel 5	N/A
s32k116_lqfp48	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_2
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2
s32k118_lqfp48	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_2
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2
s32k118_lqfp64	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
s32k142_lqfp48	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_2

Derivatives	Pwm Module	Pwm Channels available	Fault channel
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3	FLT_1
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5	FLT_1
s32k142_lqfp64	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT←_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT←_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5	FLT_0, FLT_1, FLT_2, FLT←_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5	FLT_0, FLT_1, FLT_2, FLT←_3
s32k142_lqfp100	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT←_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT←_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT←_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT←_3
s32k142w_lqfp48	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_2
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3	FLT_1
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5	FLT_1
s32k142w_lqfp64	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT←_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT←_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5	FLT_0, FLT_1, FLT_2, FLT←_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5	FLT_0, FLT_1, FLT_2, FLT←_3

Derivatives	Pwm Module	Pwm Channels available	Fault channel
s32k144_lqfp48	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_2
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3	FLT_1
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5	FLT_1
s32k144_lqfp64	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
s32k144_lqfp100	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
s32k144_mapbga100	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
s32k144w_lqfp48	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_2

Derivatives	Pwm Module	Pwm Channels available	Fault channel
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3	FLT_1
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5	FLT_1
s32k144w_lqfp64	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
s32k146_lqfp64	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 4	Channel 5, Channel 6	N/A
	Ftm 5	Channel 0, Channel 1, Channel 3, Channel 5	N/A
s32k146_lqfp100	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 4	Channel 2, Channel 5, Channel 6	FLT_0, FLT_1
	Ftm 5	Channel 0, Channel 1, Channel 3, Channel 5	FLT_0, FLT_1

Derivatives	Pwm Module	Pwm Channels available	Fault channel
s32k146_mapbga100	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 4	Channel 2, Channel 5, Channel 6	FLT_0, FLT_1
	Ftm 5	Channel 0, Channel 1, Channel 3, Channel 5	FLT_0, FLT_1
s32k146_lqfp144	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 4	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1
	Ftm 5	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1
s32k148_lfpq100	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 4	Channel 2, Channel 5, Channel 6	FLT_0, FLT_1
	Ftm 5	Channel 0, Channel 1, Channel 3, Channel 5	FLT_0, FLT_1

Derivatives	Pwm Module	Pwm Channels available	Fault channel
s32k148_mapbga100	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 4	Channel 2, Channel 5, Channel 6	FLT_0, FLT_1
	Ftm 5	Channel 0, Channel 1, Channel 3, Channel 5	FLT_0, FLT_1
s32k148_lqfp144	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 4	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1
	Ftm 5	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1
	Ftm 6	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 7	FLT_0, FLT_1
	Ftm 7	Channel 0, Channel 1, Channel 2, Channel 3, Channel 5, Channel 7	FLT_0, FLT_1
s32k148_lqfp176	Ftm 0	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 1	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 2	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3
	Ftm 3	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1, FLT_2, FLT↩_3

Derivatives	Pwm Module	Pwm Channels available	Fault channel
	Ftm 4	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1
	Ftm 5	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1
	Ftm 6	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1
	Ftm 7	Channel 0, Channel 1, Channel 2, Channel 3, Channel 4, Channel 5, Channel 6, Channel 7	FLT_0, FLT_1

Flexio's resource

Derivatives	Module	Flexio timer available	Flexio Channels available
All Derivatives	Flexio 0	Timer_0, Timer_1, Timer_2, Timer↩ _3	Channel_0, Channel_1, Channel_↩ 2, Channel_3, Channel_4, Channel_5, Channel_6, Channel_7

3.4 Deviations from Requirements

The driver deviates from the AUTOSAR PWM Driver software specification in some places. The table below identifies the AUTOSAR requirements that are not implemented or out of scope for the PWM Driver.

Term	Definition
N/S	Out of scope
N/I	Not implemented
N/F	Not fully implemented

Below table identifies the AUTOSAR requirements that are not fully implemented, implemented differently or out of scope for the PWM driver.

Requirement	Status	Description	Notes
SWS_Pwm_00065	N/S	The Pwm SWS shall not define the code file structure.	Code file structure does always not define in the driver.
SWS_Pwm_70075	N/S	Pwm_Irq.c shall include Pwm.h.	Pwm_Irq.c is not needed. Autosar specific interrupt behaviour is implemented using a normal function placed in the Pwm.c file.

Requirement	Status	Description	Notes
SWS_Pwm_00162	N/S	The Pwm Driver shall support synchronuous and asynchronous power state transitions, depending on the value of the configuration parameter PwmPowerStateAsynchTransitionMode.	Rejected: All PWM hardware do not support asynchronous power state transitions
SWS_Pwm_00164	N/S	In case the configuration parameter PwmPowerStateAsynchTransitionMode is set to TRUE, the preparation process shall continue in background after the relative API returns and its completion shall be notified by means of the configured callback.	Rejected: All PWM hardware do not support asynchronous power state transitions
SWS_Pwm_00189	N/S	Service name: - Pwm_Main_PowerTransitionManager - Syntax: - void Pwm_Main_PowerTransitionManager(void) - Service ID[hex]: - 0x0d - Description: - This API is cyclically called and supervises the power state transitions, checking for the readiness of the module and issuing the callbacks IoHwAb_Pwm_NotifyReadyForPowerState (see PwmPowerStateReadyCbkRef configuration parameter). - Available via: - SchM_Pwm.h -	Rejected: All PWM hardwares do not support asynchronous power state transitions
SWS_Pwm_00190	N/S	This API executes any non-immediate action needed to finalize a power state transition requested by Pwm_PreparePowerState().	Rejected: All PWM hardwares do not support asynchronous power state transitions
SWS_Pwm_00191	N/S	The rate of scheduling shall be defined by Pwm MainSchedulePeriod and shall be variable, as the function only needs to be called if a transition has been requested.	Rejected: All PWM hardwares do not support asynchronous power state transitions
SWS_Pwm_00192	N/S	This API shall also issue callback notifications to the eventually registered users (IoHwAbs) as configured, only in case the asynch mode is chosen.	Rejected: All PWM hardwares do not support asynchronous power state transitions

Requirement	Status	Description	Notes
SWS_Pwm_00193	N/S	In case the PWM module is not initialized, this function shall simply return without any further elaboration. This is needed to avoid to elaborate uninitialized variables. No DET error shall be entered, because this condition can easily be verified during the startup phase (tasks started before the initialization is complete).Rationale: during the startup phase it can happen that the OS already schedules tasks, which call main functions, while some modules are not initialised yet. This is no real error condition, although need handling, i.e. returning without execution.Although the transition state monitoring functionality is mandatory, the implementation of this API is optional, meaning that if the HW allows for other ways to deliver notification and watch the transition state the implementation of this function can be skipped.	Rejected: All PWM hardwares do not support asynchronous power state transitions
SWS_Pwm_00198	N/S	Service name: - IoHwAb↩_Pwm_NotifyReadyFor↩PowerState - Syntax: - void IoHwAb_Pwm_NotifyReady↩ForPowerState(void) - Service ID[hex]: - 0x60 - Sync/Async: - Synchronous - Reentrancy: - Non Reentrant - Parameters (in): - None - Parameters (inout): - None - Parameters (out): - None - Return value: - None - Description: - The API shall be invoked by the PWM Driver when the requested power state preparation for mode is completed. - Available via: - IoHwAb_Pwm.h -	Rejected: All PWM hardwares do not support asynchronous power state transitions

Requirement	Status	Description	Notes
SWS_Pwm_00199	N/S	In case the PWM Driver is configured to support power state management with asynchronous transitions, this API shall be called to signal completion of the power transition preparation phase to the IoHwAbs module.This is a callback, this API is to be implemented in the IoHwAbs component.	Rejected: All PWM hardwares do not support asynchronous power state transitions
ECUC_Pwm_00143	N/S	Name - PwmPowerState↩AsynchTransitionMode - Parent Container - PwmGeneral - Description - Enables / disables support of the PWM Driver to the asynchronous power state transition. - Multiplicity - 0..1 - Type - EcucBooleanParamDef - Default value - false - Post-Build Variant Multiplicity - false - Post-Build Variant Value - false - Multiplicity Configuration Class - Pre-compile time - X - All Variants - Link time - – - - Post-build time - – - - Value Configuration Class - Pre-compile time - X - All Variants - Link time - – - - Post-build time - – - - Scope / Dependency - scope: localdependency↩: This parameter shall only be configured if the parameter PwmLowPowerStatesSupport is set to true. -	Rejected: All PWM hardwares do not support asynchronous power state transitions
ECUC_Pwm_00144	N/S	Container Name - PwmPower↩StateConfig - Description - Each instance of this parameter defines a power state and the callback to be called when this power state is reached. - Configuration Parameters -	Rejected: All PWM hardwares do not support asynchronous power state transitions

Requirement	Status	Description	Notes
ECUC_Pwm_00146	N/S	Name - PwmPowerState - Parent Container - PwmPower↩ StateConfig - Description - Each instance of this parameter describes a different power state supported by the PWM HW. It should be defined by the HW supplier and used by the P↩ WMDriver to reference specific HW configurations which set the PWM HW module in the referenced power state.At least the power mode corresponding to full power state shall be always configured. - Multiplicity - 1 - Type - EcucIntegerParamDef (Symbolic Name generated for this parameter) - Range - 0 .. 18446744073709551615 - - Default value - – - Post-Build Variant Value - false - Value Configuration Class - Pre-compile time - X - All Variants - Link time - – - - Post-build time - – - - Scope / Dependency - scope↩ : localdependency: This parameter shall only be configured if the parameter PwmLowPower↩ StatesSupport is set to true. -	Rejected: All PWM hardwares do not support asynchronous power state transitions
ECUC_Pwm_00145	N/S	Name - PwmPowerState↩ ReadyCbkRef - Parent Container - PwmPowerStateConfig - Description - Each instance of this parameter contains a reference to a power mode callback defined in a CDD or IoHwAbs component. - Multiplicity - 1 - Type - EcucFunctionNameDef - Default value - – - maxLength - – - minLength - – - regular↩ Expression - – - Post-Build Variant Value - false - Value Configuration Class - Pre-compile time - X - All Variants - Link time - – - - Post-build time - – - - Scope / Dependency - scope: localdependency↩ : This parameter shall only be configured if the parameter PwmLowPowerStatesSupport is set to true. -	Rejected: All PWM hardwares do not support asynchronous power state transitions

Requirement	Status	Description	Notes
ECUC_Pwm_00150	N/S	Name - PwmKernelEcuc↩PartitionRef - Parent Container - PwmGeneral - Description - Maps the PWM kernel to zero or one ECUC partitions to assign the driver kernel to a certain core. The ECUC partition referenced is a subset of the ECUC partitions where the PWM driver is mapped to.Tags: atp.Status=draft - Multiplicity - 0..1 - Type - Reference to [EcucPartition] - Post-Build Variant Multiplicity - true - Post-Build Variant Value - true - Multiplicity Configuration Class - Pre-compile time - X - All Variants - Link time - – - - Post-build time - – - - Value Configuration Class - Pre-compile time - X - All Variants - Link time - – - - Post-build time - – - - Scope / Dependency - scope: ECU -	Type IV Autosar multicore not implemented for current module (AAI-445), therefore Pwm↩KernelEcucPartitionRef is not supported
SWS_Pwm_00153	N/S	These requirements are not applicable to this specification.	not a requirement
SWS_Pwm_CONSTR_00001	N/S	DRAFT: The ECUC partitions referenced by PwmKernel↩EcucPartitionRef shall be a subset of the ECUC partitions referenced by PwmEcuc↩PartitionRef.	Type IV Autosar multicore not implemented for current module (AAI-445), therefore Pwm↩KernelEcucPartitionRef is not supported
SWS_Pwm_91003	N/S	Service name: - Pwm↩_DisableNotification (draft) - Syntax: - void Pwm_DisableNotification(Pwm_ChannelType ChannelNumber) - Service ID[hex]: - 0x06 - Sync/Async: - Asynchronous - Reentrancy: - Reentrant for different channel numbers - Parameters (in): - Channel↩Number - Numeric identifier of the PWM - Parameters (inout): - None - Parameters (out): - None - Return value: - None - Description: - Service to disable the PWM signal edge notification.Tags: atp.↩Status=draft - Available via: - Pwm.h -	Description specified as draft is not clear. Should be re-assesed on next ASR version

Requirement	Status	Description	Notes
SWS_Pwm_91004	N/S	Service name: - Pwm↩_EnableNotification (draft) - Syntax: - void Pwm_EnableNotification(Pwm_ChannelType ChannelNumber, Pwm_EdgeNot - Service ID[hex]: - 0x07 - Sync/Async: - Asynchronous - Reentrancy: - Reentrant for different channel numbers - Parameters (in): - ChannelNumber - Numeric identifier of the PWM - Notification - Type of notification PWM_RISING_EDGE or PWM_FALLING_EDGE or PWM_BOTH_EDGES - Parameters (inout): - None - Parameters (out): - None - Return value: - None - Description: - Service to enable the PWM signal edge notification according to notification parameter.Tags: atp.Status=draft - Available via: - Pwm.h -	Description specified as draft is not clear. Should be re-assesed on next ASR version
SWS_Pwm_91000	N/S	Service name: - Pwm↩_SetDutyCycle (draft) - Syntax: - void Pwm_SetDutyCycle(Pwm_ChannelType ChannelNumber, uint16 DutyCycle) - Service ID[hex]: - 0x02 - Sync/Async: - Asynchronous - Reentrancy: - Reentrant for different channel numbers - Parameters (in): - Channel↩Number - Numeric identifier of the PWM - DutyCycle - Min=0x0000 Max=0x8000 - Parameters (inout): - None - Parameters (out): - None - Return value: - None - Description: - Service sets the duty cycle of the PWM channel.Tags: atp.Status=draft - Available via: - Pwm.h -	Description specified as draft is not clear. Should be re-assesed on next ASR version

Requirement	Status	Description	Notes
SWS_Pwm_91002	N/S	Service name: - Pwm_SetOutputToIdle (draft) - Syntax: - void Pwm_SetOutputToIdle(Pwm_ChannelType ChannelNumber) - Service ID[hex]: - 0x04 - Sync/Async: - Asynchronous - Reentrancy: - Reentrant for different channel numbers - Parameters (in): - ChannelNumber - Numeric identifier of the PWM - Parameters (inout): - None - Parameters (out): - None - Return value: - None - Description: - Service sets the PWM output to the configured Idle state.Tags: atp.Status=draft - Available via: - Pwm.h -	Description specified as draft is not clear. Should be re-assesed on next ASR version
SWS_Pwm_91001	N/S	Service name: - Pwm_SetPeriodAndDuty (draft) - Syntax: - void Pwm_SetPeriodAndDuty(Pwm_ChannelType ChannelNumber, Pwm_PeriodTy - Service ID[hex]: - 0x03 - Sync/Async: - Asynchronous - Reentrancy: - Reentrant for different channel numbers - Parameters (in): - ChannelNumber - Numeric identifier of the PWM - Period - Period of the PWM signal - DutyCycle - Min=0x0000 Max=0x8000 - Parameters (inout): - None - Parameters (out): - None - Return value: - None - Description: - Service sets the period and the duty cycle of a PWM channelTags: atp.Status=draft - Available via: - Pwm.h -	Description specified as draft is not clear. Should be re-assesed on next ASR version
SWS_Pwm_00175	N/S	The API shall report the DET error PWM_E_TRANSITION_NOT_POSSIBLE in case the requested power state cannot be directly reached from the current power state.	Currently, Pwm driver has only 2 powe state (High power and Low Power) so this DET error will not happen. Follow up ticket AAI-929
SWS_Pwm_00195	N/S	The API shall report the DET error PWM_E_TRANSITION_NOT_POSSIBLE in case the requested power state cannot be directly reached from the current power state.	Currently, Pwm driver has only 2 powe state (High power and Low Power) so this DET error will not happen. Follow up ticket AAI-929

Requirement	Status	Description	Notes
SWS_Pwm_00188	N/S	The API shall report the DET error PWM_E_TRANSITIO↩ N_NOT_POSSIBLE in case the requested power state cannot be directly reached from the current power state.All asynchronous operation needed to reach the target power state can be executed in background in the context of Pwm_Main_↩ PowerTransitionManager.	Currently, Pwm driver has only 2 powe state (High power and Low Power) so this DET error will not happen. Follow up ticket AAI-929

3.5 Driver Limitations

The Pwm driver software has the following limitations :

- The FTM reload point interrupt support is not implemented.
- The Pwm_SetDutyPhaseShift function when using Modfied_Combine_Mode is not support SyncUpdate = TRUE

There are some limitations for FlexIO IP due to the hardware limitations:

- FlexIO compare registers are not buffered so they are written directly, therefore the update of duty cycle and period will be immediate. Option PwmDutycycleUpdatedEndperiod and PwmPeriodUpdatedEndperiod is not supported for FlexIO.
- FlexIO notification only support for rising edge. Other cases are not supported, including falling edges and both edges notification.
- FlexIO does not support functionality of Pwm_SetOutputToIdle.
- FlexIO does not support for the duty cycle is (0)0% and (0x8000)100% and period is 0 as well When user call this function with parameter 0% , 100% duty cycle or 0 preriod ,it will generate a spike pulse.

3.6 Driver usage and configuration tips

Basically, PWM driver should be configured by EB tresos in 3 steps:

- Choose PWM module/channel to be used.
- Configure PWM module/channel in PwmModule tab.
- Configure logic channel in PwmChannel.

3.7 Runtime errors

None.

3.8 Symbolic Names Disclaimer

All containers having symbolicNameValue set to TRUE in the AUTOSAR schema will generate defines like:

#define <Mip>Conf_<Container_ShortName>_<Container_ID>

For this reason it is forbidden to duplicate the names of such containers across the RTD configurations or to use names that may trigger other compile issues (e.g. match existing `#ifdefs` arguments).

Chapter 4

Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the driver. All the parameters are described below.

- Module Pwm
    - Container PwmChannelConfigSet
        - Container PwmChannel
            - Parameter PwmChannelId
            - Parameter PwmChannelClass
            - Parameter PwmPeriodInTicks
            - Parameter PwmPeriodDefault
            - Parameter PwmDutycycleDefault
            - Parameter PwmPolarity
            - Parameter PwmIdleState
            - Parameter PwmNotification
            - Reference PwmChannelEcucPartitionRef
            - Reference PwmHwChannel
            - Reference PwmMcuClockReferencePoint
        - Container PwmFtm
            - Parameter PwmHwInstance
            - Container PwmFtmClkCfg
            - Parameter PwmFtmClockSource
            - Parameter PwmFtmClockPrescaler
            - Parameter PwmFtmClockPrescalerAlternate
            - Container PwmFtmGlobalChCfg
            - Parameter PwmFtmCounterMode
            - Parameter PwmFtmPeriod
            - Parameter PwmFtmPeriodDither
            - Parameter PwmFtmDeadTime
            - Parameter PwmFtmDeadTimePrescaler
            - Container PwmFtmInstCfg
            - Parameter PwmFtmDebugMode
            - Parameter PwmFtmWriteProtection
            - Parameter PwmFtmInitTrigger

- Parameter PwmFtmInitTriggerMode
- Parameter PwmFtmOverflowIrq
- Parameter PwmFtmReloadIrq
- Container PwmFtmOverflowIrqCallback
- Parameter PwmFtmOverflowIrqFunctionCallback
- Parameter PwmFtmOverflowIrqParameterCallback
- Container PwmFtmReloadIrqCallback
- Parameter PwmFtmReloadIrqFunctionCallback
- Parameter PwmFtmReloadIrqParameterCallback
- Container PwmFtmSyncCfg
- Parameter PwmFtmSyncMode
- Parameter PwmFtmHwTrig0
- Parameter PwmFtmHwTrig1
- Parameter PwmFtmHwTrig2
- Parameter PwmFtmAutoHwTrig
- Parameter PwmFtmMaxLoadPoint
- Parameter PwmFtmMinLoadPoint
- Parameter PwmFtmHalfCycleLoadPoint
- Parameter PwmFtmReloadPointFrequency
- Parameter PwmFtmInvertControl
- Parameter PwmFtmSwOutControl
- Parameter PwmFtmOutputMask
- Parameter PwmFtmCounterInitial
- Parameter PwmFtmCounterSync
- Container FtmGlobalFaultCfg
- Parameter FtmFaultFunctionality
- Parameter FtmFaultFilterValue
- Parameter FtmFaultOutputState
- Parameter FtmFaultIrqUsed
- Container FtmFaultSettings
- Parameter FtmFaultId
- Parameter FtmFaultPolarity
- Parameter FtmEnableFaultFilter
- Parameter FtmFaultNotification
- Container PwmFtmCh
- Parameter PwmFtmChId
- Parameter PwmFtmChPulseType
- Parameter PwmFtmChInitOutput
- Parameter PwmFtmChPolarity
- Parameter PwmFtmChDutyCycle
- Parameter PwmFtmChDutyCycleDither
- Parameter PwmFtmChInterrupt
- Parameter PwmFtmChPwmOutputEnable
- Parameter PwmFtmChSwCtrlEnable
- Parameter PwmFtmChExternTriggerEnable
- Parameter PwmFtmChSwCtrlValue
- Parameter PwmFtmChMatchLoadPointEnable

        - Parameter PwmFtmPairChEnable
        - Container PwmFtmChInterruptCallback
        - Parameter PwmFtmChInterruptFunctionCallback
        - Parameter PwmFtmChInterruptParameterCallback
        - Container PwmFtmPairedCh
        - Parameter PwmFtmPairedChDeadtimeEnable
        - Parameter PwmFtmPairedChSync
        - Parameter PwmFtmPairedChExtTrig
        - Parameter PwmFtmPairedChComplementary
        - Parameter PwmFtmDutyDitheringPaired
        - Parameter PwmFtmPairedChCombineMode
        - Parameter PwmFtmPairedChComplementaryMode
        - Parameter PwmFtmPairedChPhaseShift
    - Container PwmFlexio
        - Parameter PwmHwInstance
        - Container PwmFlexioChannels
        - Parameter FlexioChId
        - Parameter FlexioPinId
        - Parameter FlexioChPrescaler
        - Parameter FlexioChPrescalerAlternate
        - Parameter FlexioChDutyCycle
        - Parameter FlexioChPeriod
        - Parameter FlexioChPolarity
        - Parameter FlexioChInterrupt
        - Reference FlexioMclChRef
        - Container FlexioChIrqCallback
        - Parameter FlexioChIrqFunctionCallback
        - Parameter FlexioChIrqParameterCallback
- Container PwmGeneral
    - Parameter PwmMulticoreEnabled
    - Parameter PwmDevErrorDetect
    - Parameter PwmDutycycleUpdatedEndperiod
    - Parameter PwmPeriodUpdatedEndperiod
    - Parameter PwmNotificationSupported
    - Parameter PwmEnableUserModeSupport
    - Parameter PwmLowPowerStatesSupport
    - Parameter PwmPowerStateAsynchTransitionMode
    - Parameter PwmEnableDualClockMode
    - Parameter PwmEnableExternalTrigger
    - Parameter PwmEnablePhaseShift
    - Parameter PwmEnableMaskOutputs
    - Parameter PwmFaultSupported
    - Parameter PwmMultiChannelSync
    - Parameter PwmIndex

    - Reference PwmEcucPartitionRef
    - Reference PwmKernelEcucPartitionRef
    - Container PwmPowerStateConfig
      - Parameter PwmPowerState
      - Parameter PwmPowerStateReadyCbkRef
  - Container PwmConfigurationOfOptApiServices
    - Parameter PwmDeInitApi
    - Parameter PwmGetOutputState
    - Parameter PwmSetDutyCycle
    - Parameter PwmSetOutputToIdle
    - Parameter PwmSetPeriodAndDuty
    - Parameter PwmVersionInfoApi
    - Parameter PwmGetChannelStateApi
    - Parameter PwmSetDutyCycle_NoUpdate
    - Parameter PwmSetPeriodAndDuty_NoUpdate
    - Parameter PwmSetPhaseShift
    - Parameter PwmSetPhaseShift_NoUpdate
    - Parameter PwmSetDutyPhaseShift
    - Parameter PwmSetChannelDeadTime
    - Parameter PwmForceOutputToZeroApi
    - Parameter PwmEnableTrigger
    - Parameter PwmDisableTrigger
    - Parameter PwmResetCounter
  - Container CommonPublishedInformation
    - Parameter ArReleaseMajorVersion
    - Parameter ArReleaseMinorVersion
    - Parameter ArReleaseRevisionVersion
    - Parameter ModuleId
    - Parameter SwMajorVersion
    - Parameter SwMinorVersion
    - Parameter SwPatchVersion
    - Parameter VendorApiInfix
    - Parameter VendorId

4.1 Module Pwm

Configuration of Pwm (Pulse Width Modulation) module.

Included containers:

- PwmChannelConfigSet
- PwmGeneral
- PwmConfigurationOfOptApiServices
- CommonPublishedInformation

Property	Value
type	ECUC-MODULE-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantSupport	true
supportedConfigVariants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

4.2 Container PwmChannelConfigSet

Container contains the channel configuration parameter of the Pwm driver.

Included subcontainers:

- PwmChannel
- PwmFtm
- PwmFlexio

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.3 Container PwmChannel

Configuration of an individual Pwm channel.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.4 Parameter PwmChannelId

Channel ID of the Pwm channel. This value will be assigned to the symbolic name derived of the PwmChannel container short name.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.5 Parameter PwmChannelClass

Class of Pwm Channel.

PWM_FIXED_PERIOD - Period of the channel will not be changed.

PWM_FIXED_PERIOD_SHIFTED - Period of the channel will not be changed, and support with phase shift feature.

PWM_VARIABLE_PERIOD - Period of the channel can be changed.

<note>

Due to Ftm hardware specific feature that the counter register and period register are shared common for all

channels in one Ftm instance, therefore if current channel is configured PWM_FIXED_PERIOD class, but another

channel in the same Ftm instance is configured PWM_VARIABLE_PERIOD, then when that channel call function to

change period, current channel will change period, too.

</note>

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	PWM_FIXED_PERIOD
literals	['PWM_FIXED_PERIOD', 'PWM_FIXED_PERIOD_SHIFTED', 'PWM_↩ VARIABLE_PERIOD']

4.6 Parameter PwmPeriodInTicks

Check this option to configure Default Period unit in ticks, or uncheck this to configure Default Period unit in seconds.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	true

4.7 Parameter PwmPeriodDefault

Default period value of Pwm channel at initialization.

The measure unit are in ticks (if Period In Ticks checked), or in seconds (if unchecked)

Valid range: [0, 65534]

<note>

The maximum period tick is 65534 (instead of maximum value of period register 65535) in order to achieve perfect 0 or 100% duty cycle.

All channels which are in the same Ftm instance with current channel must have the same Default Period value, due to Ftm hardware specific feature that the period register is shared common for all channels in one Ftm instance.

</note>

Property	Value
type	ECUC-FLOAT-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	1.0
max	65534.0
min	0.0

4.8 Parameter PwmDutycycleDefault

Default value for duty cycle of Pwm channel at initialization.

0 represents for 0% duty cycle

16384 (0x4000) represents for 50% duty cycle

32768 (0x8000) represents for 100% duty cycle

Valid value: [0,32768]

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE VARIANT-POST-BUILD: POST-BUILD
defaultValue	16384
max	32768
min	0

4.9 Parameter PwmPolarity

Define the polarity of Pwm channel at initialization.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE VARIANT-POST-BUILD: POST-BUILD
defaultValue	PWM_HIGH
literals	['PWM_HIGH', 'PWM_LOW']

4.10 Parameter PwmIdleState

Define Pwm channel state when the output is set to idle.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	PWM_LOW
literals	['PWM_HIGH', 'PWM_LOW']

4.11 Parameter PwmNotification

User callback notification function.

This option is only activated when PwmGeneral/PwmNotificationSupported is checked.

<note>

Use NULL_PTR without any quotes to determine no notification function is used.

If the string is different from above, it will be used as the notification function name.

Notification does not apply to channel that its alignment type is PWM_CENTER_ALIGNED.

</note>

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	NULL_PTR

4.12 Reference PwmChannelEcucPartitionRef

Maps a Pwm channel to zero or multiple ECUC partitions to limit the access to this channel group.

The ECUC partitions referenced are a subset of the ECUC partitions where the Pwm driver is mapped to.

When users choose ENABLE multicore feature by checking PwmMulticoreEnabled option, this will force to configure at least 1 ECUC partition in this list that is referenced from ECUC module; OR when DISABLE multicore feature, user have to remove all ECUC partitions in this list.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.13 Reference PwmHwChannel

Select the hw channel on which the functionality of the current PWM channel will be implemented.

Property	Value
type	ECUC-CHOICE-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destinations	['/TS_T40D2M10I1R0/Pwm/PwmChannelConfigSet/PwmFtm/PwmFtmCh', '/TS_T40D2M10I1R0/Pwm/PwmChannelConfigSet/PwmFlexio/PwmFlexio↩ Channels']

4.14 Reference PwmMcuClockReferencePoint

Reference to the clock source configuration, which is set in the MCU driver configuration.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Mcu/McuModuleConfiguration/McuClockSetting↩ Config/McuClockReferencePoint

4.15 Container PwmFtm

Configuration of a Ftm module available on the platform.

Included subcontainers:

- PwmFtmClkCfg
- PwmFtmGlobalChCfg
- PwmFtmInstCfg
- PwmFtmSyncCfg
- FtmGlobalFaultCfg
- PwmFtmCh

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.16 Parameter PwmHwInstance

Select the Ftm hardware module instance available on the current platform.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	Ftm_0
literals	['Ftm_0', 'Ftm_1', 'Ftm_2', 'Ftm_3', 'Ftm_4', 'Ftm_5', 'Ftm_6', 'Ftm_7']

4.17 Container PwmFtmClkCfg

Ftm Clock configuration settings.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.18 Parameter PwmFtmClockSource

Select Ftm module clock source.

This option will be written into Clock Source Selection bitfield (CLKS) of Status and Control register (SC).

FTM_PWM_IP_CLOCK_SOURCE_NONE : No input clock is given to Ftm. Ftm counter is not operational.

FTM_PWM_IP_CLOCK_SOURCE_SYSTEMCLK : System clock is given to Ftm, which is refered to PER_CLK.

FTM_PWM_IP_CLOCK_SOURCE_FIXEDCLK : An fixed frequnecy clock is given to Ftm. This selection is not supported in current platform.

FTM_PWM_IP_CLOCK_SOURCE_EXTERNALCLK : An external clock is given to Ftm, which is refered to Ftm_x_REF_CLK (x = 0 or 1).

<note>

Clock source and frequency are dependent on the value of the clock that is configured in MCU

and refered by MCU clock reference. Make sure that the correct reference is used for the configured clock.

</note>

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	System_clock
literals	['No_clock', 'System_clock', 'Fixed_clock', 'External_clock']

4.19 Parameter PwmFtmClockPrescaler

Select the prescaler value for clock source.

This option will be written into Prescale Factor Selection bitfield (PS) of Status and Control register (SC).

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE

Property	Value
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Divide__by__1
literals	['Divide__by__1', 'Divide__by__2', 'Divide__by__4', 'Divide__by__8', 'Divide__by__↩ 16', 'Divide__by__32', 'Divide__by__64', 'Divide__by__128']

4.20 Parameter PwmFtmClockPrescalerAlternate

Select the prescaler value for clock source.

This option will be written into Prescale Factor Selection bitfield (PS) of Status and Control register (SC).

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Divide__by__1
literals	['Divide__by__1', 'Divide__by__2', 'Divide__by__4', 'Divide__by__8', 'Divide__by__↩ 16', 'Divide__by__32', 'Divide__by__64', 'Divide__by__128']

4.21 Container PwmFtmGlobalChCfg

Ftm Clock configuration settings.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.22 Parameter PwmFtmCounterMode

Select edge alignment type which is applied to all channels in current Ftm module.

This option will be written into Center-Aligned Pwm Select (CPWMS) of Status and Control register (SC).

FTM_PWM_IP_EDGE_ALIGNED : The leading edges of all Pwm signals are aligned with the beginning of the period.

Ftm counter operates in Up Counting mode.

FTM_PWM_IP_CENTER_ALIGNED : The pulse width centers for all Pwm channels are aligned with the value of CNTIN.

Ftm counter operates in Up-Down Counting mode.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	Edge_Aligned_mode
literals	['Edge_Aligned_mode', 'Center_Aligned_mode']

4.23 Parameter PwmFtmPeriod

User need to check the FTM frequency before enter the period in frequency of PWM. The min pwm frequency must be greater than (FTM counter frequency / 65536).

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true

Property	Value
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	1
max	65535
min	0

4.24 Parameter PwmFtmPeriodDither

The value of the dithering applied to the PWM frequency.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	31
min	0

4.25 Parameter PwmFtmDeadTime

Deadtime counter used to create a offset between the output of two consecutive Ftm channels used complementary modes.

Complementary channels behave as a single channel with two outputs which may have a deadtime between them.

This option will be written into Extended Deadtime Value bitfield (DTVALEX) and Deadtime Value bitfield (DTVAL)

in Deadtime Configuration register (DEADTIME).

Valid value: [0,1023]

<note>For channels which are configured as independent this value has no effect.</note>

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	1023
min	0

4.26 Parameter PwmFtmDeadTimePrescaler

Deadtime prescaler.

This option will be written into Deadtime Prescaler Value (DTPS) in Deadtime Configuration register (DEADTIME).

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Divide_by_1
literals	['Divide_by_1', 'Divide_by_4', 'Divide_by_16']

4.27 Container PwmFtmInstCfg

Ftm Instance configuration settings.

Included subcontainers:

- PwmFtmOverflowIrqCallback
- PwmFtmReloadIrqCallback

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.28 Parameter PwmFtmDebugMode

Select the Ftm behavior in Debug mode. Refer to Debug mode sub-chapter for more details.

This option will be written into Debug Mode (BDMMODE) in Configuration register (CONF).

Mode 00

Counter - Stopped

Flag bit - Can be set

Output - Normal

Bufferd Registers - Writes bypass the registers buffers

Mode 01

Counter - Stopped

Flag bit - Is not set

Output - Forced to their safe value according to POLn bit

Bufferd Registers - Writes bypass the registers buffers

Mode 10

Counter - Stopped

Flag bit - Is not set

Output - Frozen when the chip enters Debug mode

Bufferd Registers - Writes bypass the registers buffers

Mode 11

Counter - Running

Flag bit - Can be set

Output - Normal

Bufferd Registers - Normal

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Debug_Mode_00
literals	['Debug_Mode_00', 'Debug_Mode_01', 'Debug_Mode_10', 'Debug_Mode_11']

4.29 Parameter PwmFtmWriteProtection

Enable/Disable the write protection of FTM register/bitfields.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.30 Parameter PwmFtmInitTrigger

Enable/Disable the generation of an initialization trigger.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.31 Parameter PwmFtmInitTriggerMode

Select the generation of initialization trigger mode.

Trigger on Reload Point - Initialization trigger is generated when FTM counter reaches a reload point.

Trigger on Counter Update - Initialization trigger is generated when FTM counter is updated.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Trigger_on_Reload_Point
literals	['Trigger_on_Reload_Point', 'Trigger_on_Counter_Update']

4.32 Parameter PwmFtmOverflowIrq

Enable/Disable the generation of an initialization trigger.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.33 Parameter PwmFtmReloadIrq

Enable/Disable the generation of an initialization trigger.

Do not support for this implementation.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.34 Container PwmFtmOverflowIrqCallback

Configure Notfication function and parameter for interrupt handler callback.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.35 Parameter PwmFtmOverflowIrqFunctionCallback

User callback function.

NOTE: Use NULL_PTR without any quotes. If the used string is different from NULL_PTR it will be used as the configured function name.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	NULL_PTR

4.36 Parameter PwmFtmOverflowIrqParameterCallback

User callback parameter.

NOTE: Use NULL_PTR without any quotes. If the used string is different from NULL_PTR it will be used as the configured parameter name.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	NULL_PTR

4.37 Container PwmFtmReloadIrqCallback

Configure Notfication function and parameter for interrupt handler callback.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.38 Parameter PwmFtmReloadIrqFunctionCallback

User callback function.

NOTE: Use NULL_PTR without any quotes. If the used string is different from NULL_PTR it will be used as the configured function name.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	NULL_PTR

4.39 Parameter PwmFtmReloadIrqParameterCallback

User callback parameter.

NOTE: Use NULL_PTR without any quotes. If the used string is different from NULL_PTR it will be used as the configured parameter name.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	NULL_PTR

4.40 Container PwmFtmSyncCfg

Ftm Synchronization configuration settings.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.41 Parameter PwmFtmSyncMode

Configure FTM Sync to use Software or Hardware trigger modes

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE VARIANT-POST-BUILD: POST-BUILD
defaultValue	Software_sync_trigger
literals	['Software_sync_trigger', 'Hardware_sync_trigger']

4.42 Parameter PwmFtmHwTrig0

Enables/disables hardware trigger for register sync

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.43 Parameter PwmFtmHwTrig1

Enables/disables hardware trigger for register sync

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.44 Parameter PwmFtmHwTrig2

Enables/disables hardware trigger for register sync

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.45 Parameter PwmFtmAutoHwTrig

Enable/disable auto clear of triggers in hardware trigger mode

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.46 Parameter PwmFtmMaxLoadPoint

Enable/disable max counter value loading point

(for Center-Aligned Pwm, the Reload point is in middle of period)

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.47 Parameter PwmFtmMinLoadPoint

Enable/disable min counter value loading point.

(for Center-Aligned Pwm, the Reload point is in end of period)

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.48 Parameter PwmFtmHalfCycleLoadPoint

Enable/disable half cycle loading point

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.49 Parameter PwmFtmReloadPointFrequency

Select the number of enabled reload opportunities should happen until an enabled reload opportunity becomes a reload point.

This option will be written into Frequency of the Reload Opportunities bitfield (LDFQ) in Configuration register (CONF).

<note>

For safety, this option is activated only when PwmGeneral/PwmDutycycleUpdatedEndperiod is checked.

</note>

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	31
min	0

4.50 Parameter PwmFtmInvertControl

Select when the Inverting Control register is synchronized.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Sync_disabled
literals	['Sync_disabled', 'Sync_on_FTM_Clock_rising_edge', 'Sync_on_FTM_↩ Sync_Trigger']

4.51 Parameter PwmFtmSwOutControl

Select when the Software Output Control register is synchronized.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Sync_disabled
literals	['Sync_disabled', 'Sync_on_FTM_Clock_rising_edge', 'Sync_on_FTM_↩ Sync_Trigger']

4.52 Parameter PwmFtmOutputMask

Select when the Output Mask register is synchronized.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Sync_disabled
literals	['Sync_disabled', 'Sync_on_FTM_Clock_rising_edge', 'Sync_on_FTM_↩ Sync_Trigger']

4.53 Parameter PwmFtmCounterInitial

Select when the Counter Initial, max counter, halfCycle and Counter Value register is synchronized.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Sync_on_FTM_Sync_Trigger
literals	['Sync_disabled', 'Sync_on_FTM_Clock_rising_edge', 'Sync_on_FTM_↩ Sync_Trigger']

4.54 Parameter PwmFtmCounterSync

Select when the Ftm Counter register is synchronized.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Sync_on_FTM_Sync_Trigger
literals	['Sync_disabled', 'Sync_on_FTM_Clock_rising_edge', 'Sync_on_FTM_↩ Sync_Trigger']

4.55 Container FtmGlobalFaultCfg

Global Fault configuration.

Autosar Requirements: Not specified.

Included subcontainers:

- FtmFaultSettings

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.56 Parameter FtmFaultFunctionality

Selects the fault control mode

> FAULT_DISABLED -> Fault control disabled for all channels.

> FAULT_EVEN_CHANNELS_MANUAL_CLEAR -> Fault control enabled for even channels (0, 2, 4..) with manual fault clearing.

> FAULT_ALL_CHANNELS_MANUAL_CLEAR -> Fault control enabled for all channels with manual fault clearing.

> FAULT_ALL_CHANNELS_AUTO_CLEAR -> Fault control enabled for all channels with automatic fault clearing.

Note: For any automatic fault clear modes, a Fault Notification function should be defined. In automatic modes, the hardware shall

be configured into manual fault clearing and automatic fault clearing shall be implemented by software.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	FAULT_DISABLED
literals	['FAULT_DISABLED', 'FAULT_EVEN_CHANNELS_MANUAL_CLEAR', 'FAULT_ALL_CHANNELS_MANUAL_CLEAR', 'FAULT_ALL_CHANN↩ ELS_AUTO_CLEAR']

4.57 Parameter FtmFaultFilterValue

Selects the filter value for the fault inputs.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	15
min	0

4.58 Parameter FtmFaultOutputState

Specifies the fault state for the PWM channel output during fault conditions

SAFE_VALUE: Outputs will be placed into safe values when fault events in ongoing (defined by polarity)

TRI_STATE: Outputs will be tri-stated when fault event is ongoing

Autosar Requirements: Not specified.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	SAFE_VALUE
literals	['SAFE_VALUE', 'TRI_STATE']

4.59 Parameter FtmFaultIrqUsed

Enable/Disable the generation of an the fault notification.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.60 Container FtmFaultSettings

Configuration of an individual Ftm fault input.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	4
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.61 Parameter FtmFaultId

Select the Ftm Fault Id available in module.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	FLT_0
literals	['FLT_0', 'FLT_1', 'FLT_2', 'FLT_3']

4.62 Parameter FtmFaultPolarity

Defines the polarity of the fault input.

Autosar Requirements: not specified.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	POLARITY_HIGH
literals	['POLARITY_HIGH', 'POLARITY_LOW']

4.63 Parameter FtmEnableFaultFilter

Enables/Disables the filter for the fault input.

Autosar Requirements: Not specified.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.64 Parameter FtmFaultNotification

User callback function.

Note: Please use NULL or NULL_PTR w/o any quotes. If the used string is different from NULL

or NULL_PTR it will be used as the configured function name.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	NULL_PTR

4.65 Container PwmFtmCh

Configuration of an individual Ftm hardware channel.

Included subcontainers:

- PwmFtmChInterruptCallback
- PwmFtmPairedCh

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.66 Parameter PwmFtmChId

Select the Ftm hardware channel available in module. This value will be assigned to the symbolic name derived of the PwmChannel container short name.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	CH_0
literals	['CH_0', 'CH_1', 'CH_2', 'CH_3', 'CH_4', 'CH_5', 'CH_6', 'CH_7']

4.67 Parameter PwmFtmChPulseType

The type of the generated PWM signal.

For Edge/Center aligned PWM:

High-true pulse - clear Output on match

Low-true pulse - set Output on match

For combined modes:

High-true pulse - set on channel (n) match, and clear on channel (n+1) match

Low-true pulse - clear on channel (n) match, and set on channel (n+1) match

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	High_true_pulse
literals	['High_true_pulse', 'Low_true_pulse']

4.68 Parameter PwmFtmChInitOutput

The initial state of the channel output is configured in PWM signal.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	LOW
literals	['LOW', 'HIGH']

4.69 Parameter PwmFtmChPolarity

Select operation mode of this Ftm channel.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Channel_active_HIGH
literals	['Channel_active_LOW', 'Channel_active_HIGH']

4.70 Parameter PwmFtmChDutyCycle

Define the duty cycle value (in tick) for this channel.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	65534
min	0

4.71 Parameter PwmFtmChDutyCycleDither

Define the duty cycle dither value (in tick) for this channel.

Valid value: [0, 1023]

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	31
min	0

4.72 Parameter PwmFtmChInterrupt

Enable/Disable the generation of an initialization trigger.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.73 Parameter PwmFtmChPwmOutputEnable

Enable/Disable the output of the PWM signal on the channel pin.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	true

4.74 Parameter PwmFtmChSwCtrlEnable

Enable/Disable the software control for this channels output.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.75 Parameter PwmFtmChExternTriggerEnable

Enable/Disable the generation of an external trigger.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.76 Parameter PwmFtmChSwCtrlValue

The software output control forces 0 or 1 to the channel output.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.77 Parameter PwmFtmChMatchLoadPointEnable

Enable/Disable the channel match as an reaload oportunity.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.78 Parameter PwmFtmPairChEnable

Enable the paired channel feature.

The channels n and n+1 will be used as a channel pair.

Note: Cannot be used for odd channels.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.79 Container PwmFtmChInterruptCallback

Configure Notfication function and parameter for interrupt handler callback.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.80 Parameter PwmFtmChInterruptFunctionCallback

User callback function.

NOTE: Use NULL_PTR without any quotes. If the used string is different from NULL_PTR it will be used as the configured function name.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	NULL_PTR

4.81 Parameter PwmFtmChInterruptParameterCallback

User callback parameter.

NOTE: Use NULL_PTR without any quotes. If the used string is different from NULL_PTR it will be used as the configured parameter name.

Property	Value
type	ECUC-FUNCTION-NAME-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	NULL_PTR

4.82 Container PwmFtmPairedCh

Configuration of a paired Ftm hardware channel.

This will configure the channel n+1.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.83 Parameter PwmFtmPairedChDeadtimeEnable

Enable the deadtime delay for Ftm channel.

The deadtime delay insertion ensures that no two complementary signals (channels n

and n+1) drive the active state at the same time.

This option is activated only when Ftm channel is using operation mode that has COMPLEMENTARY

(COMBINED_COMPLEMENTARY, PHASE_SHIFTED_COMPLEMENTARY).

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.84 Parameter PwmFtmPairedChSync

Enable the synchronization of the channel match value registers for this pair.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	true

4.85 Parameter PwmFtmPairedChExtTrig

Enable the generation of an external trigger for channel n+1 of this pair.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.86 Parameter PwmFtmPairedChComplementary

Enable the complementary mode for Ftm channel.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.87 Parameter PwmFtmDutyDitheringPaired

The PWM edge dithering in Combine/Modified Combine Mode can be done in the channel (n) match edge or in the channel (n+1) match edge.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	false

4.88 Parameter PwmFtmPairedChCombineMode

Select combine mode for the Ftm channel pair.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Combine__modes__disabled
literals	['Combine__modes__disabled', 'Combined__Mode', 'Modfied__Combine__Mode']

4.89 Parameter PwmFtmPairedChComplementaryMode

Select complementary mode of the Ftm channel n+1.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Invert__Output
literals	['Invert__Output', 'Duplicate__Output']

4.90 Parameter PwmFtmPairedChPhaseShift

Define the offset value (in tick) from 1 that the leading edge of pulse signal will start.

Valid value: [0, 65533]

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	65533
min	0

4.91 Container PwmFlexio

Configuration of an Flexio module available on the platform.

Included subcontainers:

- PwmFlexioChannels

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE VARIANT-POST-BUILD: POST-BUILD

4.92 Parameter PwmHwInstance

Select the hardware Flexio module.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	Flexio_0
literals	['Flexio_0']

4.93 Container PwmFlexioChannels

List of Flexio channels available on the platform.

Included subcontainers:

- FlexioChIrqCallback

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	4
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD

4.94 Parameter FlexioChId

Select one of the Flexio channels available on the platform.

NOTE: This also selects the used timer.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	CH_0
literals	['CH_0', 'CH_1', 'CH_2', 'CH_3']

4.95 Parameter FlexioPinId

Select one of the Flexio Pins available on the platform.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	PIN_0
literals	['PIN_0', 'PIN_1', 'PIN_2', 'PIN_3', 'PIN_4', 'PIN_5', 'PIN_6', 'PIN_7']

4.96 Parameter FlexioChPrescaler

Select clock prescaler used for this Flexio channel.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD

Property	Value
defaultValue	FLEXIO_PWM_IP_CLK_DIV_1
literals	['FLEXIO_PWM_IP_CLK_DIV_1', 'FLEXIO_PWM_IP_CLK_DIV_16', 'FLEXIO_PWM_IP_CLK_DIV_256']

4.97 Parameter FlexioChPrescalerAlternate

Select alternative clock prescaler used for this Flexio channel.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	FLEXIO_PWM_IP_CLK_DIV_1
literals	['FLEXIO_PWM_IP_CLK_DIV_1', 'FLEXIO_PWM_IP_CLK_DIV_16', 'FLEXIO_PWM_IP_CLK_DIV_256']

4.98 Parameter FlexioChDutyCycle

Value for duty cycle used for initialization.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	256
min	0

4.99 Parameter FlexioChPeriod

Period value used at initialization.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	0
max	512
min	0

4.100 Parameter FlexioChPolarity

Define the output polarity of the channel.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	FLEXIO_PWM_IP_ACTIVE_HIGH
literals	['FLEXIO_PWM_IP_ACTIVE_HIGH', 'FLEXIO_PWM_IP_ACTIVE_LOW']

4.101 Parameter FlexioChInterrupt

Define what happens when a flag event is generated.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	FLEXIO_PWM_IP_IRQ_DISABLED
literals	['FLEXIO_PWM_IP_IRQ_DISABLED', 'FLEXIO_PWM_IP_IRQ_ON_↩ PERIOD_END']

4.102 Reference FlexioMclChRef

Select the Flexio MCL hw channel on which the the current PWM channel will be implemented.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
requiresSymbolicNameValue	False
destination	/TS_T40D2M10I1R0/Mcl/MclConfig/FlexioCommon/FlexioMclLogicChannels

4.103 Container FlexioChIrqCallback

Configure Notification function and parameter for interrupt handler callback.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.104 Parameter FlexioChIrqFunctionCallback

User callback function.

NOTE: Use NULL_PTR without any quotes. If the used string is different from NULL_PTR it will be used as the configured function name.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	NULL_PTR

4.105 Parameter FlexioChIrqParameterCallback

User callback parameter.

NOTE: Use NULL_PTR without any quotes. If the used string is different from NULL_PTR it will be used as the configured parameter name.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: POST-BUILD
defaultValue	NULL_PTR

4.106 Container PwmGeneral

Container used for storing the general configuration of Pwm.

Included subcontainers:

- PwmPowerStateConfig

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.107 Parameter PwmMulticoreEnabled

Switch to enable/disable multicore feature.

User can choose ENABLE multicore feature by checking this option, this will force to configure at least 1 ECUC partition in

PwmChannelEcucPartitionRef, and each Pwm channel in PwmChannel to configure at least 1 ECUC partition reference

in PwmChannelEcucPartitionRef container to fulfill generating code condition; OR uncheck this option to DISABLE

multicore feature, performing this action will force user to remove all ECUC partition reference in every Pwm channels contained

in PwmChannel and in PwmChannelEcucPartitionRef.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.108 Parameter PwmDevErrorDetect

Switch to enable/disable the development error detection.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	true

4.109 Parameter PwmDutycycleUpdatedEndperiod

Switch to enable the update of the duty cycle parameter at the end of the current period.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.110 Parameter PwmPeriodUpdatedEndperiod

Switch to enable the update of the period parameter at the end of the current period.

<note>In current implementation, this option is locked and always enable by default.</note>

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	true

4.111 Parameter PwmNotificationSupported

Switch to indicate that the notifications are supported.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.112 Parameter PwmEnableUserModeSupport

If enabled, the Pwm module will adapt to run from User Mode.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.113 Parameter PwmLowPowerStatesSupport

If enabled, hardware offers low power mode and adds all power state management related APIs

(Pwm_SetPowerState, Pwm_GetCurrentPowerState, Pwm_GetTargetPowerState,

Pwm_PreparePowerState, Pwm_Main_PowerTransitionManager),

indicating if the hardware offers low power state management.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.114 Parameter PwmPowerStateAsynchTransitionMode

Enable the support of the Pwm driver to the asynchronous power state transition.

<note> This feature is not supported and is rejected, as all hardware modules do not support asynchronous power state transitions.</note>

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.115 Parameter PwmEnableDualClockMode

Switch to enable/disable dual clock mode feature, which will add/remove the service Pwm_SetClockMode() from the code.

This feature is used when the prescaler value needs to be changed to maintain same period at different frequency.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.116 Parameter PwmEnableExternalTrigger

Enable external trigger generation.

This option will activate the use of PwmEnableTrigger and PwmDisableTrigger.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.117 Parameter PwmEnablePhaseShift

Enable phase-shift feature.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.118 Parameter PwmEnableMaskOutputs

Enable mask output feature, which will add/remove the services Pwm_MaskOutputs and Pwm_UnMaskOutputs from the code.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.119 Parameter PwmFaultSupported

This enables the fault functionality.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.120 Parameter PwmMultiChannelSync

Enable the update synchronous feature for several channels.

This option will activate the use of PwmSetDutyCycle_NoUpdate, PwmSetPeriodAndDuty_NoUpdate and PwmSetPhaseShift_NoUpdate or PwmSetDutyPhaseShift.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.121 Parameter PwmIndex

Specify the InstanceId of this module instance. If only one instance is present it shall have the ID 0.

<note>In current implementation, this feature is not used.</note>

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	0
max	4294967295
min	0

4.122 Reference PwmEcucPartitionRef

Maps the Pwm driver to zero or multiple ECUC partitions to make the driver API available in the according partition.

Depending on the addressed timer resource the interfaces operate as follows.

When users choose ENABLE multicore feature by checking PwmMulticoreEnabled option, this will force to configure at least 1 ECUC partition in this list that is referenced from ECUC module; OR when DISABLE multicore feature, user have to remove all ECUC partitions in this list.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.123 Reference PwmKernelEcucPartitionRef

Maps the Pwm kernel to zero or one ECUC partitions to assign the driver kernel to a certain core.

The ECUC partition referenced is a subset of the ECUC partitions where the Pwm driver is mapped to.

<note>This feature is not supported and is rejected.</note>

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.124 Container PwmPowerStateConfig

Each instance of this parameter defines a power state and the callback to be called when this power state is reached.

<note>This feature is not supported and is rejected, as all hardware modules do not support asynchronous power state transitions.</note>

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.125 Parameter PwmPowerState

Each instance of this parameter describes a different power state supported by the Pwm hardware.

It should be defined by the hardware supplier and used by the Pwm Driver to reference specific HW configurations which set the Pwm HW module in the referenced power state.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	9223372036854775807
min	0

4.126 Parameter PwmPowerStateReadyCbkRef

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

Property	Value
type	ECUC-FUNCTION-NAME-DEF

Property	Value
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.127 Container PwmConfigurationOfOptApiServices

Container used for storing the configuration of all optional API's.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.128 Parameter PwmDeInitApi

Add/remove the service Pwm_DeInit() from the code.

This option will toggle PWM_DE_INIT_API define macro in Pwm_Cfg.h file.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	true

4.129 Parameter PwmGetOutputState

Add/remove the service Pwm_GetOutputState() from the code.

This option will toggle PWM_GET_OUTPUT_STATE_API define macro in Pwm_Cfg.h file.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.130 Parameter PwmSetDutyCycle

Add/remove the service Pwm_SetDutyCycle() from the code.

This option will toggle PWM_SET_DUTY_CYCLE_API define macro in Pwm_Cfg.h file.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false

Property	Value
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.131 Parameter PwmSetOutputToIdle

Add/remove the service Pwm_SetOutputToIdle() from the code.

This option will toggle PWM_SET_OUTPUT_TO_IDLE_API define macro in Pwm_Cfg.h file.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.132 Parameter PwmSetPeriodAndDuty

Add/remove the service Pwm_SetPeriodAndDuty() from the code.

This option will toggle PWM_SET_PERIOD_AND_DUTY_API define macro in Pwm_Cfg.h file.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.133 Parameter PwmVersionInfoApi

Add/remove the service Pwm_GetVersionInfo() from the code.

This option will toggle PWM_VERSION_INFO_API define macro in Pwm_Cfg.h file.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.134 Parameter PwmGetChannelStateApi

Add/remove the service Pwm_GetChannelState() from the code.

This option will toggle PWM_GET_CHANNEL_STATE_API define macro in Pwm_Cfg.h file.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.135 Parameter PwmSetDutyCycle_NoUpdate

Add/remove the service Pwm_SetDutyCycle_NoUpdate() from the code.

This option will toggle PWM_SET_DUTY_CYCLE_NO_UPDATE_API define macro in Pwm_Cfg.h file.

<note>This option is activated only when PwmGeneral/PwmMultiChannelSync is enabled.</note>

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.136 Parameter PwmSetPeriodAndDuty_NoUpdate

Add/remove the service Pwm_SetPeriodAndDuty_NoUpdate() from the code.

This option will toggle PWM_SET_PERIOD_AND_DUTY_NO_UPDATE_API define macro in Pwm_Cfg.h file.

<note>This option is activated only when PwmGeneral/PwmMultiChannelSync is enabled.</note>

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.137 Parameter PwmSetPhaseShift

Add/remove the service Pwm_SetPhaseShift() from the code.

This option will toggle PWM_SET_PHASE_SHIFT_API define macro in Pwm_Cfg.h file.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.138 Parameter PwmSetPhaseShift_NoUpdate

Add/remove the service Pwm_SetPhaseShift_NoUpdate() from the code.

This option will toggle PWM_SET_PHASE_SHIFT_NO_UPDATE_API define macro in Pwm_Cfg.h file.

<note>This option is activated only when PwmGeneral/PwmMultiChannelSync is enabled.</note>

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.139 Parameter PwmSetDutyPhaseShift

Add/remove the service Pwm_SetDutyPhaseShift() from the code.

This option will toggle PWM_SET_DUTY_PHASE_SHIFT_API define macro in Pwm_Cfg.h file.

<note>This option is activated only when PwmGeneral/PwmMultiChannelSync is enabled.</note>

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.140 Parameter PwmSetChannelDeadTime

Add/remove the service Pwm_SetChannelDeadTime() from the code.

This option will toggle PWM_SETCHANNELDEADTIME_API define macro.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.141 Parameter PwmForceOutputToZeroApi

Add/remove the service Pwm_ForceOutputToZero() from the code.

This option will toggle PWM_FORCE_OUTPUT_TO_ZERO_API define macro in Pwm_Cfg.h file.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.142 Parameter PwmEnableTrigger

Add/remove the service Pwm_EnableTrigger() from the code.

This option will toggle PWM_ENABLE_TRIGGER_API define macro in Pwm_Cfg.h file.

<note>This option is activated only when PwmGeneral/PwmEnableExternalTrigger is enabled.</note>

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.143 Parameter PwmDisableTrigger

Add/remove the service Pwm_DisableTrigger() from the code.

This option will toggle PWM_DISABLE_TRIGGER_API define macro in Pwm_Cfg.h file.

<note>This option is activated only when PwmGeneral/PwmEnableExternalTrigger is enabled.</note>

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.144 Parameter PwmResetCounter

Add/remove the service Pwm_ResetCounterEnable() and Pwm_ResetCounterDisable() from the code.

This option will toggle PWM_RESET_COUNTER_API define macro in Pwm_Cfg.h file.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.145 Container CommonPublishedInformation

Common container, aggregated by all modules. It contains published information about vendor and versions.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.146 Parameter ArReleaseMajorVersion

Major version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.147 Parameter ArReleaseMinorVersion

Minor version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION

Property	Value
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.148 Parameter ArReleaseRevisionVersion

Revision version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.149 Parameter ModuleId

Module ID of this module from Module List.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	121

Property	Value
max	121
min	121

4.150 Parameter SwMajorVersion

Major version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	1
max	1
min	1

4.151 Parameter SwMinorVersion

Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.152 Parameter SwPatchVersion

Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	1
max	1
min	1

4.153 Parameter VendorApiInfix

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name.

This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows:

<ModuleName>_>VendorId>_<VendorApiInfix>.

E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name Can_Write defined in the SWS will translate to Can_123_v11r456Write.

This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	

4.154 Parameter VendorId

Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	43
max	43
min	43

Chapter 5

Module Index

5.1 Software Specification

Here is a list of all modules:

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 7

Module Documentation

7.1 FlexIO Pwm IPL

7.1.1 Detailed Description

Function Reference

- Flexio_Pwm_Ip_StatusType Flexio_Pwm_Ip_InitChannel (uint8 InstanceId, const Flexio_Pwm_Ip_ChannelConfigTyp *const UserCfg)

  Initialize a flexio channel in pwm mode.
- Flexio_Pwm_Ip_StatusType Flexio_Pwm_Ip_DeInitChannel (uint8 InstanceId, uint8 Channel)

  Deinitialize a flexio channel.
- Flexio_Pwm_Ip_StatusType Flexio_Pwm_Ip_UpdatePeriodDuty (uint8 InstanceId, uint8 Channel, uint16 Period, uint16 DutyCycle)

  Set a new value for duty cycle and period of the channel.
- boolean Flexio_Pwm_Ip_GetOutputState (uint8 InstanceId, uint8 Channel)

  Get the logic level of the channel ouput.
- Flexio_Pwm_Ip_StatusType Flexio_Pwm_Ip_UpdateInterruptMode (uint8 InstanceId, uint8 Channel, Flexio_Pwm_Ip_InterruptType IrqMode)

  Update the interrupt mode for a channel.
- uint16 Flexio_Pwm_Ip_GetPeriod (uint8 InstanceId, uint8 Channel)

  Getting the period for a channel.

7.1.2 Function Reference

7.1.2.1 Flexio_Pwm_Ip_InitChannel()

```
Flexio_Pwm_Ip_StatusType Flexio_Pwm_Ip_InitChannel (
            uint8 InstanceId,
            const Flexio_Pwm_Ip_ChannelConfigType *const UserCfg )
```

Initialize a flexio channel in pwm mode.

The function will initialize one timer and pin of the selected flexio channel in pwm mode, with the configuration of the user. The interrupts will be disabled.

Parameters

in	*Instance↩ Id*	The Flexio instance id
in	*UserCfg*	The channel configuration for the selected Flexio instance

Returns

FLEXIO_PWM_IP_STATUS_SUCCESS - if the initialization was successfull

7.1.2.2 Flexio_Pwm_Ip_DeInitChannel()

```
Flexio_Pwm_Ip_StatusType Flexio_Pwm_Ip_DeInitChannel (
            uint8 InstanceId,
            uint8 Channel )
```

Deinitialize a flexio channel.

The function will reset the timer and pin of the selected flexio channel.

Parameters

in	*Instance↩ Id*	The Flexio instance id
in	*Channel*	The channel id for the selected Flexio instance

Returns

FLEXIO_PWM_IP_STATUS_SUCCESS - if the deinitialization was successfull

7.1.2.3 Flexio_Pwm_Ip_UpdatePeriodDuty()

```
Flexio_Pwm_Ip_StatusType Flexio_Pwm_Ip_UpdatePeriodDuty (
            uint8 InstanceId,
            uint8 Channel,
            uint16 Period,
            uint16 DutyCycle )
```

Set a new value for duty cycle and period of the channel.

The function will update the selected flexio channel with the new values for the duty cycle and period.

Parameters

in	*Instance↩ Id*	The Flexio instance id
in	*Channel*	The channel id for the selected Flexio instance
in	*Period*	The new value for the period
in	*DutyCycle*	The new value for the duty cycle

Returns

FLEXIO_PWM_IP_STATUS_SUCCESS - if the initialization was successfull

7.1.2.4 Flexio_Pwm_Ip_GetOutputState()

```
boolean Flexio_Pwm_Ip_GetOutputState (
            uint8 InstanceId,
            uint8 Channel )
```

Get the logic level of the channel ouput.

The function will return the logic level that the selected flexio channel is driving on on the output pin.

Parameters

in	*Instance↩ Id*	The Flexio instance id
in	*Channel*	The channel id for the selected Flexio instance

Returns

FLEXIO_PWM_IP_STATUS_SUCCESS - if the initialization was successfull

7.1.2.5 Flexio_Pwm_Ip_UpdateInterruptMode()

```
Flexio_Pwm_Ip_StatusType Flexio_Pwm_Ip_UpdateInterruptMode (
            uint8 InstanceId,
            uint8 Channel,
            Flexio_Pwm_Ip_InterruptType IrqMode )
```

Update the interrupt mode for a channel.

The function will set a new mode for the flag event response on the selected channel.

Parameters

in	*Instance↩ Id*	The Flexio instance id
in	*Channel*	The channel id for the selected Flexio instance
in	*IrqMode*	The new irq mode for the channel

Returns

FLEXIO_PWM_IP_STATUS_SUCCESS - if the initialization was successfull

7.1.2.6 Flexio_Pwm_Ip_GetPeriod()

```
uint16 Flexio_Pwm_Ip_GetPeriod (
            uint8 InstanceId,
            uint8 Channel )
```

Getting the period for a channel.

The function will get the period on the selected channel.

Parameters

in	*Instance↩ Id*	The Flexio instance id
in	*Channel*	The channel id for the selected Flexio instance

Returns

uint16

7.2 Ftm Pwm IPL

7.2.1 Detailed Description

Data Structures

- struct Ftm_Pwm_Ip_CallbackCfgType

  Structure for notification. More...
- struct Ftm_Pwm_Ip_SyncCfgType

  Ftm configuration parameters structure. More...
- struct Ftm_Pwm_Ip_InstanceCfgType

  Ftm configuration parameters structure. More...
- struct Ftm_Pwm_Ip_PairCfgType

  Ftm configuration parameters structure. More...
- struct Ftm_Pwm_Ip_ChannelConfigType

  Ftm configuration parameters structure. More...
- struct Ftm_Pwm_Ip_FaultChCfgType

  Ftm configuration parameters structure. More...
- struct Ftm_Pwm_Ip_UserCfgType

  Ftm configuration parameters structure. More...

Macros

- #define FTM_PWM_IP_MAX_DUTY_CYCLE

  Maximum value for PWM duty cycle.
- #define FTM_PWM_IP_DUTY_TO_TICKS_SHIFT

  Shift value which converts duty to ticks.
- #define FTM_PWM_IP_NO_EDGE

  FlexTimer edge interrupt.

Types Reference

- typedef void(∗ Ftm_Pwm_Ip_CallbackType) (uint8 CbParam)

  Notification function callback type.
- typedef void(∗ Ftm_Pwm_Ip_FaultCallbackType) (void)

  Fault channel notification typedef.

Enum Reference

- enum Ftm_Pwm_Ip_StatusType

  Status return codes.
- enum Ftm_Pwm_Ip_ChannelModeType

  FlexTimer operation mode.
- enum Ftm_Pwm_Ip_ChannelStateType

  FlexTimer channel state.
- enum Ftm_Pwm_Ip_PrescalerType

  Prescaler type.
- enum Ftm_Pwm_Ip_PowerStateType

  Power state type.
- enum Ftm_Ftm_Ip_EdgeNotifType

  Edge notification type.
- enum Ftm_Pwm_Ip_CounterModeType

  FlexTimer operation mode.
- enum Ftm_Pwm_Ip_ClkSourceType

  FlexTimer clock source selection.
- enum Ftm_Pwm_Ip_PolarityType

  The polarity of the channel output is configured in PWM signal.
- enum Ftm_Pwm_Ip_OutputStateType

  The initial state of the channel output is configured in PWM signal.
- enum Ftm_Pwm_Ip_ClkPrescalerType

  FlexTimer pre-scaler factor selection for the clock source.
- enum Ftm_Pwm_Ip_SyncType

  FTM sync source.
- enum Ftm_Pwm_Ip_SyncModeType

  FTM sync mode.
- enum Ftm_Pwm_Ip_DeadtimePrescalerType

  FlexTimer pre-scaler factor for the dead-time insertion.
- enum Ftm_Pwm_Ip_DebugModeType

  Options for the FlexTimer behavior in BDM Mode.
- enum Ftm_Pwm_Ip_InitTriggModeType

  FlexTimer PWM initialization trigger generation mode.
- enum Ftm_Pwm_Ip_PairChPolarityType

  FlexTimer PWM channel (n+1) polarity for combine mode.
- enum Ftm_Pwm_Ip_PairDitherEdgeType

  FlexTimer PWM Pair Edge Dithering for combine/mcombine modes.
- enum Ftm_Pwm_Ip_FaultControlModeType

  Defines the FTM fault control mode.
- enum Ftm_Pwm_Ip_FaultOutputStateType

  FlexTimer PWM Fault output state.

Function Reference

- void Ftm_Pwm_Ip_Init (uint8 Instance, const Ftm_Pwm_Ip_UserCfgType ∗UserCfg)

  Initializes the FTM driver. The clock sources depend on the specified hardware. So that needs to check the external clock or the fixed frequency clock is available on device or not.
- void Ftm_Pwm_Ip_DeInit (uint8 Instance)

  Shuts down the FTM driver.
- Ftm_Pwm_Ip_StatusType Ftm_Pwm_Ip_UpdatePwmChannel (uint8 Instance, uint8 Channel, uint16 FirstEdge, uint16 SecondEdge, boolean SoftwareTrigger)

  This function updates the waveform output in PWM mode (duty cycle and phase).
- void Ftm_Pwm_Ip_FastUpdatePwmDuty (uint8 Instance, uint8 NumberOfChannels, const uint8 ∗Channels, const uint16 ∗Duty, boolean SoftwareTrigger)

  This function will update the duty cycle of PWM output for multiple channels.
- Ftm_Pwm_Ip_StatusType Ftm_Pwm_Ip_UpdatePwmPeriod (uint8 Instance, uint32 NewValue, boolean SoftwareTrigger)

  This function will update the new period in the frequency or in the counter value into mode register which modify the period of PWM signal on the Channel output.
- void Ftm_Pwm_Ip_SwOutputControl (uint8 Instance, uint8 Channel, Ftm_Pwm_Ip_OutputStateType OutputState, boolean ActiveState)

  This function performs the update of the SWOCTRL register on request to set/clear output state of channel in idle state or output-forced.
- void Ftm_Pwm_Ip_SyncUpdate (uint8 Instance)

  Ftm_Pwm_Ip_SyncUpdate.
- void Ftm_Pwm_Ip_DisableNotification (uint8 Instance, uint8 Channel)

  This function disables the user notifications.
- void Ftm_Pwm_Ip_EnableNotification (uint8 Instance, uint8 Channel, Ftm_Ftm_Ip_EdgeNotifType EdgeNotif)

  This function enables the user notifications.
- void Ftm_Pwm_Ip_SetClockMode (uint8 Instance, Ftm_Pwm_Ip_PrescalerType Prescaler)

  This function sets the value of the prescaler on all FTM modules.
- void Ftm_Pwm_Ip_ResetCounter (uint8 Instance, boolean IsResetCnt)

  This function enabel/disabled counter reset by Pwm_SyncUpdate() call.
- void Ftm_Pwm_Ip_MaskOutputChannels (uint8 Instance, uint32 ChannelsMask, boolean SoftwareTrigger)

  This function will mask the output of the channels and at match events will be ignored by the masked channels.
- void Ftm_Pwm_Ip_UpdatePwmDutyCycleChannel (uint8 Instance, uint8 Channel, uint16 DutyCycle, boolean SoftwareTrigger)

  This function sets the duty cycle for the specified PWM channel.
- void Ftm_Pwm_Ip_UpdatePwmPeriodAndDuty (uint8 Instance, uint8 Channel, uint16 Period, uint16 Duty↩Cycle, boolean SoftwareTrigger)

  This function sets the duty cycle and period for the specified PWM channel.
- Ftm_Pwm_Ip_OutputStateType Ftm_Pwm_Ip_GetOutputState (uint8 Instance, uint8 Channel)

  This function used to get the current output state of the PWM channel.
- void Ftm_Pwm_Ip_SetPowerState (uint8 Instance, Ftm_Pwm_Ip_PowerStateType PowerState)

  Put PWM channel based on FTM to target power mode.
- uint16 Ftm_Pwm_Ip_GetChannelState (uint8 Instance, uint8 Channel)

  This function read the duty cycle value from the duty cycle array and returns this value.
- void Ftm_Pwm_Ip_SetPhaseShift (uint8 Instance, uint8 Channel, uint16 Period, uint16 PhaseShift, boolean SoftwareTrigger)

This function set phase shift value and also force duty cycle to 50%.

- void Ftm_Pwm_Ip_SetDutyPhaseShift (uint8 Instance, uint8 Channel, uint16 DutyCycle, uint16 PhaseShift, boolean SyncUpdate)

  This function set phase shift and duty cycle value (as immediate or synchronized base on API parameter SyncUpdate)
- void Ftm_Pwm_Ip_UnMaskOutputChannels (uint8 Instance, uint32 ChannelsMask, boolean Software↩Trigger)

  This function will unmask the output of the channels and at match events will be ignored by the masked channels.
- void Ftm_Pwm_Ip_EnableTrigger (uint8 Instance, uint32 TriggerMask)

  This function enables trigger generation for specific source.
- void Ftm_Pwm_Ip_DisableTrigger (uint8 Instance, uint32 TriggerMask)

  This function disables trigger generation for specific source.
- void Ftm_Pwm_Ip_SetChannelDeadTime (uint8 Instance, uint8 Channel, uint16 DeadTimeVal)

  This function is used to update the deadtime at runtime for Pwm channels.

7.2.2 Data Structure Documentation

7.2.2.1 struct Ftm_Pwm_Ip_CallbackCfgType

Structure for notification.

The structure used to notification

Definition at line 332 of file Ftm_Pwm_Ip_Types.h.

7.2.2.2 struct Ftm_Pwm_Ip_SyncCfgType

Ftm configuration parameters structure.

FlexTimer Sync and Load parameters

Definition at line 351 of file Ftm_Pwm_Ip_Types.h.

Data Fields

Type	Name	Description
Ftm_Pwm_Ip_SyncModeType	SyncMode	Configure FTM Sync to use Software or Hardware trigger modes
boolean	HwSync0	Enable/disable hardware 0 sync
boolean	HwSync1	Enable/disable hardware 1 sync
boolean	HwSync2	Enable/disable hardware 2 sync
boolean	HwTriggerAutoClear	Enable/disable auto clear of triggers in hardware trigger mode
boolean	MaxLoadPoint	Enable/disable maximum loading point
boolean	MinLoadPoint	Enable/disable minimum loading point
boolean	HalfCycleLoadPoint	Enable/disable half cycle loading point

Data Fields

Type	Name	Description
uint8	LoadPointFreq	Value of how many reload points must happen before reloading.
Ftm_Pwm_Ip_SyncType	InverterSync	Configures INVCTRL sync
Ftm_Pwm_Ip_SyncType	OutRegSync	Configures SWOCTRL sync
Ftm_Pwm_Ip_SyncType	OutMaskSync	Configures OUTMASK sync
Ftm_Pwm_Ip_SyncType	InitCounterSync	Configures CNTIN sync
Ftm_Pwm_Ip_SyncType	CounterSync	Configures FTM Counter sync

7.2.2.3 struct Ftm_Pwm_Ip_InstanceCfgType

Ftm configuration parameters structure.

Ftm IP specific instance configuration structure type

Definition at line 376 of file Ftm_Pwm_Ip_Types.h.

Data Fields

Type	Name	Description
Ftm_Pwm_Ip_ClkSourceType	ClkSrc	Clock source for Ftm instances
Ftm_Pwm_Ip_ClkPrescalerType	ClkPs	Pre-scaler for Ftm instances
Ftm_Pwm_Ip_ClkPrescalerType	AlternateClkPs	Alternative pre-scaler for Ftm instances
Ftm_Pwm_Ip_CounterModeType	CntMode	Mode of the counter
boolean	OverflowIrqEn	Selected the overflow interrupt
Ftm_Pwm_Ip_CallbackCfgType	OverflowCb	overflow callback
boolean	ReloadIrqEn	Selected the reload interrupt
Ftm_Pwm_Ip_CallbackCfgType	ReloadCb	Reload callback
Ftm_Pwm_Ip_DebugModeType	DebugMode	Debug Mode
boolean	WriteProtection	Selected write protection
boolean	InitTriggerEn	Selected trigger
Ftm_Pwm_Ip_InitTriggModeType	InitTrigMode	Chose generation of initialization trigger mode
uint16	PwmPeriod	Period of Ftm instances
uint8	PwmPeriodDither	Period Dither
uint16	DeadTimeValue	Deadtime Value
Ftm_Pwm_Ip_DeadtimePrescalerType	DeadTimePs	Deadtime pre-scaler Value
Ftm_Pwm_Ip_FaultControlModeType	FaultCtrMode	Defines the FTM fault control mode
Ftm_Pwm_Ip_FaultOutputStateType	FaultOutState	Fault output state
uint8	FaultInFilter	Fault Input Filter Value
boolean	FaultIrqEn	Fault interrupt enable
const Ftm_Pwm_Ip_SyncCfgType *	SyncCfg	Pointer to the configured channels for the FTM sync

7.2.2.4 struct Ftm__Pwm__Ip__PairCfgType

Ftm configuration parameters structure.

FlexTimer Channel Pair configuration parameters

Definition at line 412 of file Ftm__Pwm__Ip__Types.h.

Data Fields

Type	Name	Description
uint8	PairId	FTM pair id channel.
boolean	DeadtimeEn	Selected dead-time for pair channel.
boolean	ComplementaryModeEn	Selected dead-time for pair channel.
Ftm__Pwm__Ip__PairChPolarityType	PairChPolarity	Polarity of pair channel.
boolean	PairExtTrigEn	Selected external trigger of pair channel.
boolean	PairSyncEn	Selected sync of pair channel.
uint16	PhaseShiftValue	the phase Shift Value
Ftm__Pwm__Ip__PairDitherEdgeType	DitherEdge	

7.2.2.5 struct Ftm__Pwm__Ip__ChannelConfigType

Ftm configuration parameters structure.

FlexTimer Channel configuration parameters

Definition at line 438 of file Ftm__Pwm__Ip__Types.h.

Data Fields

Type	Name	Description
uint8	ChannelId	FTM channel ID.
Ftm__Pwm__Ip__ChannelModeType	ChannelMode	Mode of Ftm channel.
boolean	ChIrqEn	Selected for interrupt.
Ftm__Pwm__Ip__CallbackCfgType	ChannelCb	Callback for the ftm channels.
boolean	ChOutputEn	Enable/Disable the output of the PWM signal on the channel pin.
boolean	SwControlEn	Enable/Disable the software control for this channels output.
boolean	SwControlVal	Channel Software Output Control Value.
boolean	ExtTrigEn	Selected for external trigger.
boolean	ChMatchLoadEn	Selected for external trigger.
uint16	DutyCycle	Duty cycle value.
uint8	DutyCycleDither	Duty cycle value in Dither.
Ftm__Pwm__Ip__OutputStateType	InitOut	Output state of Ftm channel.
Ftm__Pwm__Ip__PolarityType	Polarity	Polarity of Ftm channel.
const Ftm__Pwm__Ip__PairCfgType ∗	PairCfg	The configuration of pair channel.

7.2.2.6 struct Ftm_Pwm_Ip_FaultChCfgType

Ftm configuration parameters structure.

FlexTimer fault configuration parameters

Definition at line 483 of file Ftm_Pwm_Ip_Types.h.

Data Fields

Type	Name	Description
uint8	FaultId	FTM fault ID is actived.
Ftm_Pwm_Ip_PolarityType	FaultPol	Defines the polarity of the fault input.
boolean	FaultFilEn	Fault Input Filter Enable
Ftm_Pwm_Ip_FaultCallbackType	FaultCb	Pointer to fault notification function.

7.2.2.7 struct Ftm_Pwm_Ip_UserCfgType

Ftm configuration parameters structure.

FlexTimer User Configuration structure

Definition at line 501 of file Ftm_Pwm_Ip_Types.h.

Data Fields

Type	Name	Description
const Ftm_Pwm_Ip_InstanceCfgType ∗	InstanceCfg	Pointer to the configured channels for the FTM instance.
const Ftm_Pwm_Ip_ChannelConfigType ∗const ∗	ConfiguredChArray	Pointer to the configured channels array for the FTM channel.
uint8	NoOfConfiguredCh	Number of the configured channels for the FTM channel.
const Ftm_Pwm_Ip_FaultChCfgType ∗const ∗	ConfiguredFaultArray	Pointer to the configured channels array for the FTM channel.
uint8	NoOfFaultCfg	Number of the configured channels for fault.

7.2.3 Macro Definition Documentation

7.2.3.1 FTM_PWM_IP_MAX_DUTY_CYCLE

```
#define FTM_PWM_IP_MAX_DUTY_CYCLE
```

Maximum value for PWM duty cycle.

Definition at line 122 of file Ftm_Pwm_Ip.h.

7.2.3.2 FTM_PWM_IP_DUTY_TO_TICKS_SHIFT

```
#define FTM_PWM_IP_DUTY_TO_TICKS_SHIFT
```

Shift value which converts duty to ticks.

Definition at line 124 of file Ftm_Pwm_Ip.h.

7.2.3.3 FTM_PWM_IP_NO_EDGE

```
#define FTM_PWM_IP_NO_EDGE
```

FlexTimer edge interrupt.

Definition at line 107 of file Ftm_Pwm_Ip_Types.h.

7.2.4 Types Reference

7.2.4.1 Ftm_Pwm_Ip_CallbackType

```
typedef void(* Ftm_Pwm_Ip_CallbackType) (uint8 CbParam)
```

Notification function callback type.

Definition at line 325 of file Ftm_Pwm_Ip_Types.h.

7.2.4.2 Ftm_Pwm_Ip_FaultCallbackType

```
typedef void(* Ftm_Pwm_Ip_FaultCallbackType) (void)
```

Fault channel notification typedef.

Definition at line 476 of file Ftm_Pwm_Ip_Types.h.

7.2.5 Enum Reference

7.2.5.1 Ftm_Pwm_Ip_StatusType

```
enum Ftm_Pwm_Ip_StatusType
```

Status return codes.

Enumerator

FTM_PWM_IP_STATUS_SUCCESS	Generic operation success status
FTM_PWM_IP_STATUS_ERROR	Generic operation failure status
FTM_PWM_IP_STATUS_BUSY	Generic operation busy status
FTM_PWM_IP_STATUS_TIMEOUT	Generic operation timeout status
FTM_PWM_IP_STATUS_UNSUPPORTED	Generic operation unsupported status

Definition at line 114 of file Ftm_Pwm_Ip_Types.h.

7.2.5.2 Ftm_Pwm_Ip_ChannelModeType

```
enum Ftm_Pwm_Ip_ChannelModeType
```

FlexTimer operation mode.

Enumerator

FTM_PWM_IP_MODE_EDGE_ALIGNED_HIGH	Edge aligned PWM High-true pulses (clear on match)
FTM_PWM_IP_MODE_EDGE_ALIGNED_LOW	Edge aligned PWM Low-true pulses (set on match)
FTM_PWM_IP_MODE_CENTER_ALIGNED_↩HIGH	Center aligned PWM High-true pulses (clear on match)
FTM_PWM_IP_MODE_CENTER_ALIGNED_↩LOW	Center aligned PWM Low-true pulses (set on match)
FTM_PWM_IP_MODE_COMBINE_HIGH	Combined channels PWM High-true pulses (set on channel (n) match, and clear on channel (n+1) match)
FTM_PWM_IP_MODE_COMBINE_LOW	Combined channels PWM Low-true pulses (clear on channel (n) match, and set on channel (n+1) match)
FTM_PWM_IP_MODE_MODIFIED_COMBIN↩E_HIGH	Modified combined channel PWM High-true pulses (set on channel (n) match, and clear on channel (n+1) match)
FTM_PWM_IP_MODE_MODIFIED_COMBIN↩E_LOW	Modified combined channel PWM Low-true pulses (clear on channel (n) match, and set on channel (n+1) match)
FTM_PWM_IP_MODE_NODEFINED	Non defined Mode

Definition at line 125 of file Ftm_Pwm_Ip_Types.h.

7.2.5.3 Ftm_Pwm_Ip_ChannelStateType

```
enum Ftm_Pwm_Ip_ChannelStateType
```

FlexTimer channel state.

Enumerator

FTM_PWM_IP_CHANNEL_UNINIT	The state's channel before initialization
FTM_PWM_IP_CHANNEL_RUNNING	The state's channel is running
FTM_PWM_IP_CHANNEL_IDLE	The state's channel is idle
FTM_PWM_IP_CHANNEL_OUTPUT_FORCED	The state's channel in force output channel mode

Definition at line 142 of file Ftm_Pwm_Ip_Types.h.

7.2.5.4 Ftm_Pwm_Ip_PrescalerType

```
enum Ftm_Pwm_Ip_PrescalerType
```

Prescaler type.

This enumeration specifies the possible types of prescalers used to configure base-clock timers

Enumerator

FTM_PWM_IP_PRIMARY_PRESCALER	Selected value is the default/primary prescaler.
FTM_PWM_IP_ALTERNATIVE_PRESCALER	Selected value is the alternative configured prescaler.

Definition at line 154 of file Ftm_Pwm_Ip_Types.h.

7.2.5.5 Ftm_Pwm_Ip_PowerStateType

```
enum Ftm_Pwm_Ip_PowerStateType
```

Power state type.

Power state currently active or set as target power state.

Enumerator

FTM_PWM_IP_FULL_POWER	PWM full power mode.
FTM_PWM_IP_LOW_POWER	PWM low power mode.
FTM_PWM_IP_NODEFINE_POWER	PWM no define power mode.

Definition at line 167 of file Ftm_Pwm_Ip_Types.h.

7.2.5.6 Ftm_Ftm_Ip_EdgeNotifType

```
enum Ftm_Ftm_Ip_EdgeNotifType
```

Edge notification type.

This enumeration defines the type of edge transition that can generate a notification

Enumerator

FTM_PWM_IP_RISING_EDGE	A notification will be generated on the rising edge.
FTM_PWM_IP_FALLING_EDGE	A notification will be generated on the falling edge.
FTM_PWM_IP_BOTH_EDGES	A notification will be generated on any state transition.

Definition at line 183 of file Ftm_Pwm_Ip_Types.h.

7.2.5.7 Ftm_Pwm_Ip_CounterModeType

```
enum Ftm_Pwm_Ip_CounterModeType
```

FlexTimer operation mode.

Enumerator

FTM_PWM_IP_EDGE_ALIGNED	The FTM Counter is in UP Counting mode
FTM_PWM_IP_CENTER_ALIGNED	The FTM Counter is in UP-DOWN Counting mode

Definition at line 195 of file Ftm_Pwm_Ip_Types.h.

7.2.5.8 Ftm_Pwm_Ip_ClkSourceType

```
enum Ftm_Pwm_Ip_ClkSourceType
```

FlexTimer clock source selection.

Enumerator

FTM_PWM_IP_CLOCK_SOURCE_NONE	None use clock for FTM
FTM_PWM_IP_CLOCK_SOURCE_SYSTEMCLK	System clock
FTM_PWM_IP_CLOCK_SOURCE_FIXEDCLK	Fixed clock
FTM_PWM_IP_CLOCK_SOURCE_EXTERNALCLK	External clock

Definition at line 202 of file Ftm_Pwm_Ip_Types.h.

7.2.5.9 Ftm_Pwm_Ip_PolarityType

```
enum Ftm_Pwm_Ip_PolarityType
```

The polarity of the channel output is configured in PWM signal.

Enumerator

FTM_PWM_IP_POLARITY_LOW	The channel polarity is active LOW
FTM_PWM_IP_POLARITY_HIGH	The channel polarity is active HIGH

Definition at line 215 of file Ftm_Pwm_Ip_Types.h.

7.2.5.10 Ftm_Pwm_Ip_OutputStateType

```
enum Ftm_Pwm_Ip_OutputStateType
```

The initial state of the channel output is configured in PWM signal.

Enumerator

FTM_PWM_IP_OUTPUT_STATE_LOW	The channel output is LOW
FTM_PWM_IP_OUTPUT_STATE_HIGH	The channel output is HIGH

Definition at line 222 of file Ftm_Pwm_Ip_Types.h.

7.2.5.11 Ftm_Pwm_Ip_ClkPrescalerType

```
enum Ftm_Pwm_Ip_ClkPrescalerType
```

FlexTimer pre-scaler factor selection for the clock source.

Enumerator

FTM_PWM_IP_CLOCK_DIV_1	Divide by 1
FTM_PWM_IP_CLOCK_DIV_2	Divide by 2

Enumerator

FTM_PWM_IP_CLOCK_DIV_4	Divide by 4
FTM_PWM_IP_CLOCK_DIV_8	Divide by 8
FTM_PWM_IP_CLOCK_DIV_16	Divide by 16
FTM_PWM_IP_CLOCK_DIV_32	Divide by 32
FTM_PWM_IP_CLOCK_DIV_64	Divide by 64
FTM_PWM_IP_CLOCK_DIV_128	Divide by 128

Definition at line 229 of file Ftm_Pwm_Ip_Types.h.

7.2.5.12 Ftm_Pwm_Ip_SyncType

```
enum Ftm_Pwm_Ip_SyncType
```

FTM sync source.

Enumerator

FTM_PWM_IP_SYNC_DISABLED	synchronization for the FTM register is disabled
FTM_PWM_IP_SYNC_ON_CLK	synchronization for the FTM register is done on each FTM Clock rising edge
FTM_PWM_IP_SYNC_ON_TRIGGER	synchronization for the FTM register is done on each FTM Sync Trigger

Definition at line 242 of file Ftm_Pwm_Ip_Types.h.

7.2.5.13 Ftm_Pwm_Ip_SyncModeType

```
enum Ftm_Pwm_Ip_SyncModeType
```

FTM sync mode.

Enumerator

FTM_PWM_IP_SYNC_SWTRIGGER	synchronization trigger for FTM is generated by software
FTM_PWM_IP_SYNC_HWTRIGGER	synchronization trigger for FTM is generated by hardware

Definition at line 250 of file Ftm_Pwm_Ip_Types.h.

7.2.5.14 Ftm_Pwm_Ip_DeadtimePrescalerType

```
enum Ftm_Pwm_Ip_DeadtimePrescalerType
```

FlexTimer pre-scaler factor for the dead-time insertion.

Enumerator

FTM_PWM_IP_DEADTIME_DIV_1	Divide by 1
FTM_PWM_IP_DEADTIME_DIV_4	Divide by 4
FTM_PWM_IP_DEADTIME_DIV_16	Divide by 16

Definition at line 257 of file Ftm_Pwm_Ip_Types.h.

7.2.5.15 Ftm_Pwm_Ip_DebugModeType

```
enum Ftm_Pwm_Ip_DebugModeType
```

Options for the FlexTimer behavior in BDM Mode.

Enumerator

FTM_PWM_IP_BDM_MODE_00	FTM counter stopped, CH(n)F bit can be set, FTM channels in functional mode, writes to MOD,CNTIN and C(n)V registers bypass the register buffers
FTM_PWM_IP_BDM_MODE_01	FTM counter stopped, CH(n)F bit is not set, FTM channels outputs are forced to their safe value , writes to MOD,CNTIN and C(n)V registers bypass the register buffers
FTM_PWM_IP_BDM_MODE_10	FTM counter stopped, CH(n)F bit is not set, FTM channels outputs are frozen when chip enters in BDM mode, writes to MOD, CNTIN and C(n)V registers bypass the register buffers
FTM_PWM_IP_BDM_MODE_11	FTM counter in functional mode, CH(n)F bit can be set, FTM channels in functional mode, writes to MOD,CNTIN and C(n)V registers is in fully functional mode

Definition at line 265 of file Ftm_Pwm_Ip_Types.h.

7.2.5.16 Ftm_Pwm_Ip_InitTriggModeType

```
enum Ftm_Pwm_Ip_InitTriggModeType
```

FlexTimer PWM initialization trigger generation mode.

Enumerator

FTM_PWM_IP_INIT_TRIGG_RELOAD_POINT	Initialization trigger is generated when FTM counter reaches a reload point
FTM_PWM_IP_INIT_TRIGG_CNT_UPDATE	Initialization trigger is generated when FTM counter is updated

Definition at line 282 of file Ftm_Pwm_Ip_Types.h.

7.2.5.17 Ftm_Pwm_Ip_PairChPolarityType

```
enum Ftm_Pwm_Ip_PairChPolarityType
```

FlexTimer PWM channel (n+1) polarity for combine mode.

Enumerator

FTM_PWM_IP_MAIN_INVERTED	The channel (n+1) output is the inverse of the channel (n) output
FTM_PWM_IP_MAIN_DUPLICATED	The channel (n+1) output is the same as the channel (n) output

Definition at line 289 of file Ftm_Pwm_Ip_Types.h.

7.2.5.18 Ftm_Pwm_Ip_PairDitherEdgeType

```
enum Ftm_Pwm_Ip_PairDitherEdgeType
```

FlexTimer PWM Pair Edge Dithering for combine/mcombine modes.

Enumerator

FTM_PWM_IP_DITHER_LEADING_EDGE	The channel (n) dithering is used
FTM_PWM_IP_DITHER_TRAILING_EDGE	The channel (n+1) dithering is used

Definition at line 296 of file Ftm_Pwm_Ip_Types.h.

7.2.5.19 Ftm_Pwm_Ip_FaultControlModeType

```
enum Ftm_Pwm_Ip_FaultControlModeType
```

Defines the FTM fault control mode.

Enumerator

FTM_PWM_IP_FAULT_DISABLED	Fault control disabled for all channels.
FTM_PWM_IP_FAULT_EVEN_CHANNELS_↩ MANUAL_CLEAR	Fault control enabled for even channels (0, 2, 4..) with manual fault clearing.
FTM_PWM_IP_FAULT_ALL_CHANNELS_M↩ ANUAL_CLEAR	Fault control enabled for all channels with manual fault clearing.
FTM_PWM_IP_FAULT_ALL_CHANNELS_A↩ UTO_CLEAR	Fault control enabled for all channels with automatic fault clearing.

Definition at line 304 of file Ftm_Pwm_Ip_Types.h.

7.2.5.20 Ftm_Pwm_Ip_FaultOutputStateType

```
enum Ftm_Pwm_Ip_FaultOutputStateType
```

FlexTimer PWM Fault output state.

Enumerator

FTM_PWM_IP_OUT_SAFE_VALUE	FTM outputs will be placed into safe values
FTM_PWM_IP_OUT_TRI_STATE	FTM outputs will be tri-stated

Definition at line 313 of file Ftm_Pwm_Ip_Types.h.

7.2.6 Function Reference

7.2.6.1 Ftm_Pwm_Ip_Init()

```
void Ftm_Pwm_Ip_Init (
            uint8 Instance,
            const Ftm_Pwm_Ip_UserCfgType * UserCfg )
```

Initializes the FTM driver. The clock sources depend on the specified hardware. So that needs to check the external clock or the fixed frequency clock is available on device or not.

Parameters

in	*Instance*	The FTM peripheral instance number.
in	*info*	The FTM user configuration structure, see #Ftm_Pwm_Ip_UserConfigType.
out	*state*	The FTM state structure of the driver.

Returns

void

7.2.6.2 Ftm_Pwm_Ip_DeInit()

```
void Ftm_Pwm_Ip_DeInit (
            uint8 Instance )
```

Shuts down the FTM driver.

Parameters

in	*Instance*	The FTM peripheral instance number.

Returns

void

7.2.6.3 Ftm_Pwm_Ip_UpdatePwmChannel()

```
Ftm_Pwm_Ip_StatusType Ftm_Pwm_Ip_UpdatePwmChannel (
            uint8 Instance,
            uint8 Channel,
            uint16 FirstEdge,
            uint16 SecondEdge,
            boolean SoftwareTrigger )
```

This function updates the waveform output in PWM mode (duty cycle and phase).

@Note: Regarding the type of updating PWM in the duty cycle, if the expected duty is 100% then the value that is to be written to hardware will be exceed value of period. It means that the FTM counter will not match the value of the CV register in this case.

Parameters

in	*Instance*	The FTM peripheral instance number.
in	*Channel*	The channel number. In combined mode, the code finds the channel.
in	*FirstEdge*	Duty cycle or first edge time for PWM mode. Can take value between 0 - FTM_PWM_IP_MAX_DUTY_CYCLE(0 = 0% from period and FTM_PWM_IP_MAX_DUTY_CYCLE = 100% from period) Or value in ticks for the first of the PWM mode in which can have value between 0 and ftmPeriod is stored in the state structure.
in	*SecondEdge*	Second edge time - only for combined mode. Can take value between 0 - FTM_PWM_IP_MAX_DUTY_CYCLE(0 = 0% from period and FTM_PWM_IP_MAX_DUTY_CYCLE = 100% from period). Or value in ticks for the second of the PWM mode in which can have value between 0 and ftmPeriod is stored in the state structure.
in	*SoftwareTrigger*	If true a software trigger is generate to update PWM parameters.

Returns

success

- FTM_PWM_STATUS_SUCCESS : Completed successfully.
- FTM_PWM_STATUS_ERROR : Error occurred.

7.2.6.4 Ftm_Pwm_Ip_FastUpdatePwmDuty()

```
void Ftm_Pwm_Ip_FastUpdatePwmDuty (
            uint8 Instance,
            uint8 NumberOfChannels,
            const uint8 * Channels,
            const uint16 * Duty,
            boolean SoftwareTrigger )
```

This function will update the duty cycle of PWM output for multiple channels.

Parameters

in	*Instance*	The FTM peripheral instance number.
in	*NumberOfChannels*	The number of channels which should be updated.
in	*Channels*	The list of channels which should be updated.
in	*Duty*	The list of duty cycles for selected channels.
in	*SoftwareTrigger*	If true a software trigger is generate to update PWM parameters.

Returns

success

- FTM_PWM_STATUS_SUCCESS : Completed successfully.
- FTM_PWM_STATUS_ERROR : Error occurred.

7.2.6.5 Ftm_Pwm_Ip_UpdatePwmPeriod()

```
Ftm_Pwm_Ip_StatusType Ftm_Pwm_Ip_UpdatePwmPeriod (
            uint8 Instance,
            uint32 NewValue,
            boolean SoftwareTrigger )
```

This function will update the new period in the frequency or in the counter value into mode register which modify the period of PWM signal on the Channel output.

Parameters

in	*Instance*	The FTM peripheral instance number.
in	*NewValue*	The frequency or the counter value which will select with modified value for PWM signal. If the type of update in the duty cycle, the newValue parameter must be value between 1U and maximum is the frequency of the FTM counter. If the type of update in ticks, the newValue parameter must be value between 1U and 0xFFFFU.
in	*SoftwareTrigger*	If true a software trigger is generate to update PWM parameters.

Returns

operation status

- FTM_PWM_STATUS_SUCCESS : Completed successfully.
- FTM_PWM_STATUS_ERROR : Error occurred.

7.2.6.6 Ftm_Pwm_Ip_SwOutputControl()

```
void Ftm_Pwm_Ip_SwOutputControl (
            uint8 Instance,
            uint8 Channel,
            Ftm_Pwm_Ip_OutputStateType OutputState,
            boolean ActiveState )
```

This function performs the update of the SWOCTRL register on request to set/clear output state of channel in idle state or output-forced.

This function can be called from Pwm_Ftm_SetOutputToIdle, Pwm_Ftm_DeInit and Pwm_Ftm_ForceOutput↩ToZero functions

Parameters

in	*Instance*	FTM hardware module index
in	*Channel*	FTM hardware channel index
in	*OutputState*	PWM output level
in	*ActiveState*	Activation state of the SW Output Control TRUE - Enable SW Output Control FALSE - Disable SW Output Control

Returns

void

7.2.6.7 Ftm_Pwm_Ip_SyncUpdate()

```
void Ftm_Pwm_Ip_SyncUpdate (
            uint8 Instance )
```

Ftm_Pwm_Ip_SyncUpdate.

This function will allow synchronized loading of the duty registers for all the channels of a given FTM module

Parameters

in	*Instance*	The FTM peripheral instance number.

Returns

void

7.2.6.8 Ftm_Pwm_Ip_DisableNotification()

```
void Ftm_Pwm_Ip_DisableNotification (
            uint8 Instance,
            uint8 Channel )
```

This function disables the user notifications.

This function disables the user notifications of given channel

Parameters

in	*Instance*	FTM hardware module index
in	*Channel*	FTM hardware channel index

Returns

void

7.2.6.9 Ftm_Pwm_Ip_EnableNotification()

```
void Ftm_Pwm_Ip_EnableNotification (
            uint8 Instance,
            uint8 Channel,
            Ftm_Ftm_Ip_EdgeNotifType EdgeNotif )
```

This function enables the user notifications.

This function enables the user notifications of given channel

Parameters

in	*Instance*	FTM hardware module index
in	*Channel*	FTM hardware channel index
in	*EdgeNotif*	Notification type (RISING, FALLING, BOTH)

Returns

void

7.2.6.10 Ftm_Pwm_Ip_SetClockMode()

```
void Ftm_Pwm_Ip_SetClockMode (
            uint8 Instance,
            Ftm_Pwm_Ip_PrescalerType Prescaler )
```

This function sets the value of the prescaler on all FTM modules.

This function sets the prescaler depending on the value ePrescaler parameter.

Parameters

in	*Instance*	FTM hardware module index
in	*Prescaler*	Prescaler values

Returns

void

7.2.6.11 Ftm_Pwm_Ip_ResetCounter()

```
void Ftm_Pwm_Ip_ResetCounter (
             uint8 Instance,
             boolean IsResetCnt )
```

This function enabel/disabled counter reset by Pwm_SyncUpdate() call.

Parameters

in	*Instance*	FTM hardware module index
in	*IsResetCnt*	Reset counter or not

Returns

void

7.2.6.12 Ftm_Pwm_Ip_MaskOutputChannels()

```
void Ftm_Pwm_Ip_MaskOutputChannels (
             uint8 Instance,
             uint32 ChannelsMask,
             boolean SoftwareTrigger )
```

This function will mask the output of the channels and at match events will be ignored by the masked channels.

Parameters

in	*Instance*	The FTM peripheral instance number.
in	*ChannelsMask*	The mask which will select which channels will ignore match events.
in	*SoftwareTrigger*	If true a software trigger is generate to update PWM parameters.

Returns

success

- FTM_PWM_STATUS_SUCCESS : Completed successfully.

7.2.6.13 Ftm_Pwm_Ip_UpdatePwmDutyCycleChannel()

```
void Ftm_Pwm_Ip_UpdatePwmDutyCycleChannel (
            uint8 Instance,
            uint8 Channel,
            uint16 DutyCycle,
            boolean SoftwareTrigger )
```

This function sets the duty cycle for the specified PWM channel.

Parameters

in	*Instance*	The FTM peripheral instance number.
in	*Channel*	The FTM peripheral channel number.
in	*DutyCycle*	PWM duty cycle value [0-0x8000]
in	*SoftwareTrigger*	If true a software trigger is generate to update PWM parameters.

Returns

void

7.2.6.14 Ftm_Pwm_Ip_UpdatePwmPeriodAndDuty()

```
void Ftm_Pwm_Ip_UpdatePwmPeriodAndDuty (
            uint8 Instance,
            uint8 Channel,
            uint16 Period,
            uint16 DutyCycle,
            boolean SoftwareTrigger )
```

This function sets the duty cycle and period for the specified PWM channel.

Parameters

in	*Instance*	The FTM peripheral instance number.
in	*Channel*	The FTM peripheral channel number.
in	*Period*	The period will be updated
in	*DutyCycle*	PWM duty cycle value [0-0x8000]
in	*SoftwareTrigger*	If true a software trigger is generate to update PWM parameters.

Returns

void

7.2.6.15 Ftm_Pwm_Ip_GetOutputState()

```
Ftm_Pwm_Ip_OutputStateType Ftm_Pwm_Ip_GetOutputState (
            uint8 Instance,
            uint8 Channel )
```

This function used to get the current output state of the PWM channel.

Parameters

in	*Instance*	FTM hardware module index
in	*Channel*	FTM hardware channel index

Returns

Pwm_Ftm_OutputStateType FTM_PWM_OUTPUT_STATE_LOW The output pin is in LOW logic level FTM_PWM_OUTPUT_STATE_HIGH The output pin is in HIGH logic level

7.2.6.16 Ftm_Pwm_Ip_SetPowerState()

```
void Ftm_Pwm_Ip_SetPowerState (
            uint8 Instance,
            Ftm_Pwm_Ip_PowerStateType PowerState )
```

Put PWM channel based on FTM to target power mode.

This function will disable clock if target power mode is Low power, and enable clock if target power mode is full power

- nPowerState value is either PWM_LOW_POWER or PWM_FULL_POWER
- the values associated to either types are configured by tooling and the physical values associated are written to hardware registry.

Parameters

in	*Instance*	FTM hardware module index
in	*PowerState*	Target mode value

Returns

none

7.2.6.17 Ftm_Pwm_Ip_GetChannelState()

```
uint16 Ftm_Pwm_Ip_GetChannelState (
            uint8 Instance,
            uint8 Channel )
```

This function read the duty cycle value from the duty cycle array and returns this value.

Parameters

in	*Instance*	FTM hardware module index
in	*Channel*	FTM hardware channel index

Returns

uint16 Duty cycle value

7.2.6.18 Ftm_Pwm_Ip_SetPhaseShift()

```
void Ftm_Pwm_Ip_SetPhaseShift (
            uint8 Instance,
            uint8 Channel,
            uint16 Period,
            uint16 PhaseShift,
            boolean SoftwareTrigger )
```

This function set phase shift value and also force duty cycle to 50%.

In order to have Phase-Shifted Full-Bridge controller, Pwm_SetPhaseShift is introduced. This function bases on F↩ TM Combine mode with C(n) and C(n+1) combine to generate leading edge and trailing edge. Pwm_SetPhaseShift allows to set both phase shift value and period, the duty value is fixed to 50%.

Parameters

in	*Instance*	FTM hardware module index
in	*Channel*	FTM hardware channel index
in	*Period*	PWM signal period value
in	*PhaseShift*	Phase shift value
in	*SoftwareTrigger*	Buffer value be loaded or not

Returns

void

7.2.6.19 Ftm_Pwm_Ip_SetDutyPhaseShift()

```
void Ftm_Pwm_Ip_SetDutyPhaseShift (
             uint8 Instance,
             uint8 Channel,
             uint16 DutyCycle,
             uint16 PhaseShift,
             boolean SyncUpdate )
```

This function set phase shift and duty cycle value (as immediate or synchronized base on API parameter SyncUpdate)

Ftm_Pwm_Ip_SetDutyPhaseShift allows to set both phase shift and duty cycle value, The phase shift is the offset of the leading edge of the signal in respect to period starting point.

Parameters

in	*Instance*	FTM hardware module index
in	*Channel*	FTM hardware channel index
in	*DutyCycle*	Pwm duty cycle value 0x0000 for 0% ... 0x8000 for 100%
in	*PhaseShift*	Phase shift value (in ticks)
in	*SyncUpdate*	TRUE Set the phase shift and duty cycle value base on the synchronization when calling Pwm_SyncUpdate. FALSE Set phase shift and duty cycle value immediately

Returns

void

7.2.6.20 Ftm_Pwm_Ip_UnMaskOutputChannels()

```
void Ftm_Pwm_Ip_UnMaskOutputChannels (
             uint8 Instance,
             uint32 ChannelsMask,
             boolean SoftwareTrigger )
```

This function will unmask the output of the channels and at match events will be ignored by the masked channels.

Parameters

in	*Instance*	The FTM peripheral instance number.
in	*ChannelsMask*	The mask which will select which channels will ignore match events.
in	*SoftwareTrigger*	If true a software trigger is generate to update PWM parameters.

Returns

success

- FTM_PWM_STATUS_SUCCESS : Completed successfully.

7.2.6.21 Ftm_Pwm_Ip_EnableTrigger()

```
void Ftm_Pwm_Ip_EnableTrigger (
            uint8 Instance,
            uint32 TriggerMask )
```

This function enables trigger generation for specific source.

Corresponding bits with trigger source as bellow: Bit 0 Channel 2 Trigger Enable Bit 1 Channel 3 Trigger Enable Bit 2 Channel 4 Trigger Enable Bit 3 Channel 5 Trigger Enable Bit 4 Channel 0 Trigger Enable Bit 5 Channel 1 Trigger Enable Bit 6 Initialization Trigger Enable

Parameters

Instance	FTM hardware module index TriggerMask Bit mask will be clear for FTM_EXTTRIG.

Returns

void

7.2.6.22 Ftm_Pwm_Ip_DisableTrigger()

```
void Ftm_Pwm_Ip_DisableTrigger (
            uint8 Instance,
            uint32 TriggerMask )
```

This function disables trigger generation for specific source.

Corresponding bits with trigger source as bellow: Bit 0 Channel 2 Trigger Enable Bit 1 Channel 3 Trigger Enable Bit 2 Channel 4 Trigger Enable Bit 3 Channel 5 Trigger Enable Bit 4 Channel 0 Trigger Enable Bit 5 Channel 1 Trigger Enable Bit 6 Initialization Trigger Enable

Parameters

Instance	FTM hardware module index TriggerMask Bit mask will be clear for FTM_EXTTRIG.

Returns

void

7.2.6.23 Ftm_Pwm_Ip_SetChannelDeadTime()

```
void Ftm_Pwm_Ip_SetChannelDeadTime (
            uint8 Instance,
            uint8 Channel,
            uint16 DeadTimeVal )
```

This function is used to update the deadtime at runtime for Pwm channels.

Parameters

in	*Instance*	The FTM peripheral instance number.
in	*Channel*	The FTM hardware channel index
in	*DeadTimeVal*	Deadtime value

Returns

void

7.3 Pwm Driver

7.3.1 Detailed Description

Data Structures

- struct Pwm_ChannelConfigType

  Pwm channel high level configuration structure. More...
- struct Pwm_ConfigType

  Pwm high level configuration structure. More...

Macros

- #define PWM_DUTY_CYCLE_100

  100% duty cycle
- #define PWM_E_INIT_FAILED

  API Pwm_Init service called with wrong parameter.
- #define PWM_E_UNINIT

  API service used without module initialization.
- #define PWM_E_PARAM_CHANNEL

  API service used with an invalid channel Identifier.
- #define PWM_E_PERIOD_UNCHANGEABLE

  Usage of unauthorized PWM service on PWM channel configured a fixed period.
- #define PWM_E_ALREADY_INITIALIZED

  API Pwm_Init service called while the PWM driver has already been initialized.
- #define PWM_E_PARAM_POINTER

  Generated when a NULL_PTR pointer is passed to Pwm_GetVersionInfo function.
- #define PWM_E_NOT_DISENGAGED

  Generated when Pwm_SetPowerState is called while the PWM module is still in use.
- #define PWM_E_POWER_STATE_NOT_SUPPORTED

  The requested power state is not supported by the PWM module.
- #define PWM_E_TRANSITION_NOT_POSSIBLE

  Generated The requested power state is not reachable from the current one.
- #define PWM_E_PERIPHERAL_NOT_PREPARED

  Generated when Pwm_SetPowerState has been called without having called the API Pwm_PreparePowerState before.
- #define PWM_E_PERIODVALUE

  Pwm_SetPeriodAndDuty called with invalid period range.
- #define PWM_E_PARAM_NOTIFICATION

  Invalid polarity selected for edge notification.
- #define PWM_E_PARAM_NOTIFICATION_NULL

  NULL_PTR function is configured as notification callback.
- #define PWM_E_DUTYCYCLE_RANGE

  Pwm_SetDutyCycle or Pwm_SetPeriodAndDuty called with invalid duty cycle range.
- #define PWM_E_COUNTERBUS

Generated when Pwm_SetCounterBus is called with an invalid Bus.
- #define PWM_E_CHANNEL_OFFSET_VALUE

  Generated when the configured offset for the OPWMB channel is more than the period of the associated MCB channel.
- #define PWM_E_OPWMB_CHANNEL_OFFSET_DUTYCYCLE_RANGE

  Generated when the requested offset value plus the current requested duty cycle leads to programming event B over the Period value leading to unexpected behavior of the PWM signal.
- #define PWM_E_PARAM_INSTANCE

  Generated when the module id is more than the number of module that supported by this platform.
- #define PWM_E_OPWMT_CHANNEL_TRIGGER_RANGE

  Generated when the configured trigger value for the OPWMT channel is equal or greater than the period of the channel.
- #define PWM_E_SET_CHANNEL_OUTPUT

  Generated when the output state value for the SetChannelOutput of the channel.
- #define PWM_E_UNEXPECTED_ISR

  Generated when an ISR has been triggered.
- #define PWM_E_PARAM_PHASESHIFT_RANGE

  Generated when requested phase shift value greater than 0x4000 (50%)
- #define PWM_E_CHANNEL_PHASE_SHIFT_NOT_SUPPORTED

  Generated when given channel does not support phase shift feature.
- #define PWM_E_DUTY_SYNCHRONOUS_NOT_SUPPORTED

  Generated when given channel does not support duty synchronous feature. (PHASE_SHIFTED_SYNCED and P↩ HASE_SHIFTED_COMPLEMENTARY)
- #define PWM_E_TRIGGER_MASK

  Generated when bit mask is not compatible with hardware register.
- #define PWM_E_FORCE_OUTPUT_NOT_SUPPORTED

  Generated when given channel does not support force output to zero feature.
- #define PWM_E_FORCE_OUT

  Generated when given channel does not support set force out feature.
- #define PWM_E_PARAM_CONFIG

  Generated when the requested resource is configured to be unavailable on the current core.
- #define PWM_E_DEADTIME_RANGE

  Generated when the configured dead time value is not valid.
- #define PWM_E_SETOUTPUTTOIDLE_NOT_SUPPORTED

  Generated when the configured dead time value is not valid.
- #define PWM_INIT_ID

  API service ID of Pwm_Init function.
- #define PWM_DEINIT_ID

  API service ID of Pwm_DeInit function.
- #define PWM_SETDUTYCYCLE_ID

  API service ID of Pwm_SetDutyCycle function.
- #define PWM_SETPERIODANDDUTY_ID

  API service ID of Pwm_SetPeriodAndDuty function.
- #define PWM_SETOUTPUTTOIDLE_ID

  API service ID of Pwm_SetOutputToIdle function.
- #define PWM_GETOUTPUTSTATE_ID

  API service ID of Pwm_GetOutputState function.
- #define PWM_DISABLENOTIFICATION_ID

API service ID of Pwm_DisableNotification function.
- #define PWM_ENABLENOTIFICATION_ID

  API service ID of Pwm_EnableNotification function.
- #define PWM_GETVERSIONINFO_ID

  API service ID of Pwm_GetVersionInfo function.
- #define PWM_SETPOWERSTATE_ID

  API service ID of Pwm_SetPowerState function.
- #define PWM_GETCURRENTPOWERSTATE_ID

  API service ID of Pwm_GetCurrentPowerState function.
- #define PWM_GETTARGETPOWERSTATE_ID

  API service ID of Pwm_GetTargetPowerState function.
- #define PWM_PREPAREPOWERSTATE_ID

  API service ID of Pwm_PreparePowerState function.
- #define PWM_MAIN_POWERTRANSITIONMANAGER_ID

  API service ID of Pwm_Main_PowerTransitionManager function.
- #define PWM_GETCHANNELSTATE_ID

  API service ID of Pwm_GetChannelState function.
- #define PWM_FORCEOUTPUTTOZERO_ID

  API service ID of Pwm_ForceOutputToZero function.
- #define PWM_SETCOUNTERBUS_ID

  API service ID of Pwm_SetCounterBus function.
- #define PWM_SETCHANNELOUTPUT_ID

  API service ID of Pwm_SetChannelOutput function.
- #define PWM_SETTRIGGERDELAY_ID

  API service ID of Pwm_SetTriggerDelay function.
- #define PWM_SETCLOCKMODE_ID

  API service ID of Pwm_SetClockMode function.
- #define PWM_SYNCUPDATE_ID

  API service ID of Pwm_SyncUpdate function.
- #define PWM_SETPERIODANDDUTY_NO_UPDATE_ID

  API service ID of Pwm_SetPeriodAndDuty_NoUpdate function.
- #define PWM_SETDUTYCYCLE_NO_UPDATE_ID

  API service ID of Pwm_SetDutyCycle_NoUpdate function.
- #define PWM_SETCHANNELDEADTIME_ID

  API service ID of Pwm_SetChannelDeadTime function.
- #define PWM_SETPHASESHIFT_ID

  API service ID of Pwm_SetPhaseShift function.
- #define PWM_SETPHASESHIFTNOUPDATE_ID

  API service ID of Pwm_SetPhaseShift function.
- #define PWM_ENABLETRIGGER_ID

  API service ID of Pwm_EnableTrigger function.
- #define PWM_DISABLETRIGGER_ID

  API service ID of Pwm_DisableTrigger function.
- #define PWM_RESETCOUNTERENABLE_ID

  API service ID of Pwm_ResetCounterEnable function.
- #define PWM_RESETCOUNTERDISABLE_ID

  API service ID of Pwm_ResetCounterDisable function.

- #define PWM_MASKOUTPUT_ID

  API service ID of Pwm_MaskOutputs function.
- #define PWM_UNMASKOUTPUT_ID

  API service ID of Pwm_UnMaskOutputs function.
- #define PWM_DISABLERELOADNOTIF_ID

  API service ID of Pwm_DisableReloadNotification function.
- #define PWM_ENABLERELOADNOTIF_ID

  API service ID of Pwm_EnableReloadNotification function.
- #define PWM_SETCHANNELFORCEOUT_ID

  API service ID of Pwm_SetChannelForceOut function.
- #define PWM_SETDUTYPHASESHIFT_ID

  API service ID of Pwm_SetDutyPhaseShift function.

Types Reference

- typedef uint8 Pwm_ChannelType

  PWM channel type.
- typedef uint8 Pwm_InstanceType

  PWM channel type.
- typedef Pwm_Ipw_PeriodType Pwm_PeriodType

  PWM period type.
- typedef void(∗ Pwm_NotifyType) (void)

  Channel notification typedef.

Enum Reference

- enum Pwm_OutputStateType

  Output signal level.
- enum Pwm_EdgeNotificationType

  Edge notification type.
- enum Pwm_ChannelClassType

  PWM channel class type.
- enum Pwm_PowerStateType

  Power state type.
- enum Pwm_PowerStateRequestResultType

  Result of power state type.
- enum Pwm_PrescalerType

  Prescaler type.

Function Reference

- void Pwm_Init (const Pwm_ConfigType *ConfigPtr)

  This function initializes the Pwm driver.
- void Pwm_DeInit (void)

  This function deinitializes the Pwm driver.
- void Pwm_SetDutyCycle (Pwm_ChannelType ChannelNumber, uint16 DutyCycle)

  This function sets the dutycycle for the specified Pwm channel.
- void Pwm_SetPeriodAndDuty (Pwm_ChannelType ChannelNumber, Pwm_PeriodType Period, uint16 DutyCycle)

  This function sets the period and the dutycycle for the specified Pwm channel.
- void Pwm_SetOutputToIdle (Pwm_ChannelType ChannelNumber)

  This function sets the generated pwm signal to the idle value configured.
- Pwm_OutputStateType Pwm_GetOutputState (Pwm_ChannelType ChannelNumber)

  This function returns the signal output state.
- void Pwm_EnableNotification (Pwm_ChannelType ChannelNumber, Pwm_EdgeNotificationType Notification)

  This function enables the user notifications.
- void Pwm_DisableNotification (Pwm_ChannelType ChannelNumber)

  This function disables the user notifications.
- void Pwm_GetVersionInfo (Std_VersionInfoType *versioninfo)

  This function returns Pwm driver version details.
- uint16 Pwm_GetChannelState (Pwm_ChannelType ChannelNumber)

  This function returns the duty cycle of the channel passed as parameter.
- void Pwm_ForceOutputToZero (Pwm_ChannelType ChannelNumber, boolean Force)

  This function enables-disables the forcing of the output of a given channel to logic 0.
- void Pwm_SetClockMode (Pwm_PrescalerType Prescaler)

  Implementation specific function to change the peripheral clock frequency.
- void Pwm_SetChannelDeadTime (Pwm_ChannelType ChannelNumber, uint16 DeadTimeTicks)
- void Pwm_SetDutyCycle_NoUpdate (Pwm_ChannelType ChannelNumber, uint16 DutyCycle)

  This function sets the values of dutycycle for the specified Pwm channel but without updating the PWM output.
- void Pwm_SetPeriodAndDuty_NoUpdate (Pwm_ChannelType ChannelNumber, Pwm_PeriodType Period, uint16 DutyCycle)

  This function sets the values of the period and the dutycycle for the specified Pwm channel into the hardware buffers but without updating the PWM output..
- void Pwm_SetPhaseShift_NoUpdate (Pwm_ChannelType ChannelNumber, Pwm_PeriodType Period, uint16 PhaseShift)

  This function set phase shift value and also force duty cycle to 50%. The output will take effect after Pwm_SyncUpdate be called.
- void Pwm_SyncUpdate (uint8 ModuleId)

  Implementation specific function to updates duty synchronization.
- void Pwm_SetPhaseShift (Pwm_ChannelType ChannelNumber, Pwm_PeriodType Period, uint16 PhaseShift)

  This function set phase shift value and also force duty cycle to 50%.
- void Pwm_SetDutyPhaseShift (Pwm_ChannelType ChannelNumber, uint16 DutyCycle, uint16 PhaseShift, boolean SyncUpdate)

  This function set phase shift and duty cycle value (as immediate or synchronized base on API parameter SyncUpdate)
- void Pwm_EnableTrigger (uint8 TriggerHostId, uint16 TriggerMask)

This function enable trigger generation for specific source.
- void Pwm_DisableTrigger (uint8 TriggerHostId, uint16 TriggerMask)

  This function disable trigger generation for specific source.
- void Pwm_ResetCounterEnable (uint8 ModuleId)

  This function shall enable the PWM timer HW counter reset by Pwm_SyncUpdate() function.
- void Pwm_ResetCounterDisable (uint8 ModuleId)

  This function shall disable the PWM timer HW counter reset by Pwm_SyncUpdate() function.
- void Pwm_MaskOutputs (uint8 ModuleId, uint8 ChannelMask)

  This function force channels output to their inactive state.
- void Pwm_UnMaskOutputs (uint8 ModuleId, uint8 ChannelMask)

  This function puts channels output to normal operation state.
- Std_ReturnType Pwm_SetPowerState (Pwm_PowerStateRequestResultType ∗Result)

  Enters the already prepared power state.
- Std_ReturnType Pwm_GetCurrentPowerState (Pwm_PowerStateType ∗CurrentPowerState, Pwm_PowerStateRequestRe ∗Result)

  Get the current power state of the Pwm HW unit.
- Std_ReturnType Pwm_GetTargetPowerState (Pwm_PowerStateType ∗TargetPowerState, Pwm_PowerStateRequestResu ∗Result)

  Get the target power state of the Pwm HW unit.
- Std_ReturnType Pwm_PreparePowerState (Pwm_PowerStateType PowerState, Pwm_PowerStateRequestResultType ∗Result)

  Starts the needed process to allow the Pwm HW module to enter the requested power state.
- void Pwm_Main_PowerTransitionManager (void)

  Cyclically called and supervises the power state transitions.

7.3.2 Data Structure Documentation

7.3.2.1 struct Pwm_ChannelConfigType

Pwm channel high level configuration structure.

Definition at line 720 of file Pwm.h.

Data Fields

- const Pwm_ChannelType ChannelId

  Id for the logical channel.
- const Pwm_ChannelClassType PwmChannelClass

  Channel class type: Variable/Fixed period.
- const Pwm_IpwChannelConfigType IpwChannelCfg

  The type of ip channel configured.
- const Pwm_OutputStateType ChannelIdleState

  The state of the channel output in idle mode.

7.3.2.1.1 Field Documentation

7.3.2.1.1.1 ChannelId `const Pwm_ChannelType ChannelId`

Id for the logical channel.

Definition at line 723 of file Pwm.h.

7.3.2.1.1.2 PwmChannelClass `const Pwm_ChannelClassType PwmChannelClass`

Channel class type: Variable/Fixed period.

Definition at line 725 of file Pwm.h.

7.3.2.1.1.3 IpwChannelCfg `const Pwm_IpwChannelConfigType IpwChannelCfg`

The type of ip channel configured.

Definition at line 727 of file Pwm.h.

7.3.2.1.1.4 ChannelIdleState `const Pwm_OutputStateType ChannelIdleState`

The state of the channel output in idle mode.

Definition at line 729 of file Pwm.h.

7.3.2.2 struct Pwm_ConfigType

Pwm high level configuration structure.

Definition at line 739 of file Pwm.h.

Data Fields

- const Pwm_ChannelType NumChannels

  Number of Pwm configured channels.
- const Pwm_ChannelConfigType(∗ PwmChannelsConfig)[ ]

  Pointer to the list of Pwm configured channels.
- const Pwm_InstanceType NumInstances

  Number of Pwm configured instances.
- const Pwm_IpwInstanceConfigType(∗ PwmInstancesConfig)[ ]

  Pointer to the list of Pwm configured channels.
- const Pwm_ChannelType HwToLogicChannelMap [(68U)]

  Index table to translate HW channels to logical used to process interrupts for notifications.

7.3.2.2.1 Field Documentation

7.3.2.2.1.1 NumChannels `const Pwm_ChannelType NumChannels`

Number of Pwm configured channels.

Definition at line 746 of file Pwm.h.

7.3.2.2.1.2 PwmChannelsConfig `const Pwm_ChannelConfigType(* PwmChannelsConfig)[]`

Pointer to the list of Pwm configured channels.

Definition at line 748 of file Pwm.h.

7.3.2.2.1.3 NumInstances `const Pwm_InstanceType NumInstances`

Number of Pwm configured instances.

Definition at line 751 of file Pwm.h.

7.3.2.2.1.4 PwmInstancesConfig `const Pwm_IpwInstanceConfigType(* PwmInstancesConfig)[]`

Pointer to the list of Pwm configured channels.

Definition at line 753 of file Pwm.h.

7.3.2.2.1.5 HwToLogicChannelMap `const Pwm_ChannelType HwToLogicChannelMap[(68U)]`

Index table to translate HW channels to logical used to process interrupts for notifications.

Definition at line 761 of file Pwm.h.

7.3.3 Macro Definition Documentation

7.3.3.1 PWM_DUTY_CYCLE_100

```
#define PWM_DUTY_CYCLE_100
```

100% duty cycle

Errors and exceptions that will be detected by the PWM driver generated when Pwm_SetDutyCycle or Pwm_↩ SetPeriodAndDuty are called with a value for duty cycle out of valid range [0x0000, 0x8000]

Definition at line 145 of file Pwm.h.

7.3.3.2 PWM_E_INIT_FAILED

```
#define PWM_E_INIT_FAILED
```

API Pwm_Init service called with wrong parameter.

Errors and exceptions that will be detected by the PWM driver

Definition at line 152 of file Pwm.h.

7.3.3.3 PWM_E_UNINIT

```
#define PWM_E_UNINIT
```

API service used without module initialization.

Errors and exceptions that will be detected by the PWM driver

Definition at line 158 of file Pwm.h.

7.3.3.4 PWM_E_PARAM_CHANNEL

```
#define PWM_E_PARAM_CHANNEL
```

API service used with an invalid channel Identifier.

Errors and exceptions that will be detected by the PWM driver

Definition at line 164 of file Pwm.h.

7.3.3.5 PWM_E_PERIOD_UNCHANGEABLE

```
#define PWM_E_PERIOD_UNCHANGEABLE
```

Usage of unauthorized PWM service on PWM channel configured a fixed period.

Errors and exceptions that will be detected by the PWM driver

Definition at line 170 of file Pwm.h.

7.3.3.6 PWM_E_ALREADY_INITIALIZED

```
#define PWM_E_ALREADY_INITIALIZED
```

API Pwm_Init service called while the PWM driver has already been initialized.

Errors and exceptions that will be detected by the PWM driver

Definition at line 176 of file Pwm.h.

7.3.3.7 PWM_E_PARAM_POINTER

```
#define PWM_E_PARAM_POINTER
```

Generated when a NULL_PTR pointer is passed to Pwm_GetVersionInfo function.

Errors and exceptions that will be detected by the PWM driver

Definition at line 182 of file Pwm.h.

7.3.3.8 PWM_E_NOT_DISENGAGED

```
#define PWM_E_NOT_DISENGAGED
```

Generated when Pwm_SetPowerState is called while the PWM module is still in use.

Errors and exceptions that will be detected by the PWM driver

Definition at line 188 of file Pwm.h.

7.3.3.9 PWM_E_POWER_STATE_NOT_SUPPORTED

```
#define PWM_E_POWER_STATE_NOT_SUPPORTED
```

The requested power state is not supported by the PWM module.

Errors and exceptions that will be detected by the PWM driver

Definition at line 194 of file Pwm.h.

7.3.3.10 PWM_E_TRANSITION_NOT_POSSIBLE

```
#define PWM_E_TRANSITION_NOT_POSSIBLE
```

Generated The requested power state is not reachable from the current one.

Errors and exceptions that will be detected by the PWM driver

Definition at line 200 of file Pwm.h.

7.3.3.11 PWM_E_PERIPHERAL_NOT_PREPARED

```
#define PWM_E_PERIPHERAL_NOT_PREPARED
```

Generated when Pwm_SetPowerState has been called without having called the API Pwm_PreparePowerState before.

Errors and exceptions that will be detected by the PWM driver

Definition at line 207 of file Pwm.h.

7.3.3.12 PWM_E_PERIODVALUE

```
#define PWM_E_PERIODVALUE
```

Pwm_SetPeriodAndDuty called with invalid period range.

Generated when Pwm_SetPeriodAndDuty is called with a value for period out of valid range [0x0000, PWM_M↩
AX_PERIOD]

Definition at line 214 of file Pwm.h.

7.3.3.13 PWM_E_PARAM_NOTIFICATION

```
#define PWM_E_PARAM_NOTIFICATION
```

Invalid polarity selected for edge notification.

Will be generated when an invalid polarity, edge notification is requested for one PWM channel. Due to the limitations that are present in the eMIOS implementation not all the polarity notifications combinations can be supported.

Definition at line 222 of file Pwm.h.

7.3.3.14 PWM_E_PARAM_NOTIFICATION_NULL

```
#define PWM_E_PARAM_NOTIFICATION_NULL
```

NULL_PTR function is configured as notification callback.

Will be generated when a NULL_PTR function is configured as notification callback for one PWM channel and Pwm_EnableNotification is called for that channel

Definition at line 229 of file Pwm.h.

7.3.3.15 PWM_E_DUTYCYCLE_RANGE

```
#define PWM_E_DUTYCYCLE_RANGE
```

Pwm_SetDutyCycle or Pwm_SetPeriodAndDuty called with invalid duty cycle range.

Generated when Pwm_SetDutyCycle or Pwm_SetPeriodAndDuty are called with a value for duty cycle out of valid range [0x0000, 0x8000]

Definition at line 236 of file Pwm.h.

7.3.3.16 PWM_E_COUNTERBUS

```
#define PWM_E_COUNTERBUS
```

Generated when Pwm_SetCounterBus is called with an invalid Bus.

Errors and exceptions that will be detected by the PWM driver

Definition at line 242 of file Pwm.h.

7.3.3.17 PWM_E_CHANNEL_OFFSET_VALUE

```
#define PWM_E_CHANNEL_OFFSET_VALUE
```

Generated when the configured offset for the OPWMB channel is more than the period of the associated MCB channel.

Errors and exceptions that will be detected by the PWM driver

Definition at line 249 of file Pwm.h.

7.3.3.18 PWM_E_OPWMB_CHANNEL_OFFSET_DUTYCYCLE_RANGE

```
#define PWM_E_OPWMB_CHANNEL_OFFSET_DUTYCYCLE_RANGE
```

Generated when the requested offset value plus the current requested duty cycle leads to programming event B over the Period value leading to unexpected behavior of the PWM signal.

Errors and exceptions that will be detected by the PWM driver

Definition at line 256 of file Pwm.h.

7.3.3.19 PWM_E_PARAM_INSTANCE

```
#define PWM_E_PARAM_INSTANCE
```

Generated when the module id is more than the number of module that supported by this platform.

Errors and exceptions that will be detected by the PWM driver

Definition at line 263 of file Pwm.h.

7.3.3.20 PWM_E_OPWMT_CHANNEL_TRIGGER_RANGE

```
#define PWM_E_OPWMT_CHANNEL_TRIGGER_RANGE
```

Generated when the configured trigger value for the OPWMT channel is equal or greater than the period of the channel.

Errors and exceptions that will be detected by the PWM driver

Definition at line 270 of file Pwm.h.

7.3.3.21 PWM_E_SET_CHANNEL_OUTPUT

```
#define PWM_E_SET_CHANNEL_OUTPUT
```

Generated when the output state value for the SetChannelOutput of the channel.

Errors and exceptions that will be detected by the PWM driver

Definition at line 277 of file Pwm.h.

7.3.3.22 PWM_E_UNEXPECTED_ISR

```
#define PWM_E_UNEXPECTED_ISR
```

Generated when an ISR has been triggered.

1. when the driver is not initialized
2. for a HW channel that is not used by any logic channel
3. for a logic channel that has no notification configured

Errors and exceptions that will be detected by the PWM driver

Definition at line 286 of file Pwm.h.

7.3.3.23 PWM_E_PARAM_PHASESHIFT_RANGE

```
#define PWM_E_PARAM_PHASESHIFT_RANGE
```

Generated when requested phase shift value greater than 0x4000 (50%)

Pwm_SetPhaseShift only works with Combine channel (COMBINED_SYNCED or COMBINED_COMPLEME↩ NTARY) Which do not support matching at next cycle. The duty cycle is always fixed value at 50%, so Phase Shift cannot greater than 50%

Definition at line 294 of file Pwm.h.

7.3.3.24 PWM_E_CHANNEL_PHASE_SHIFT_NOT_SUPPORTED

```
#define PWM_E_CHANNEL_PHASE_SHIFT_NOT_SUPPORTED
```

Generated when given channel does not support phase shift feature.

For FTM, only combine mode (COMBINED_SYNCED and COMBINED_COMPLEMENTARY) is supported. For eMIOS, only OPWMB, OPWM and OPWMT mode is supported.

Definition at line 301 of file Pwm.h.

7.3.3.25 PWM_E_DUTY_SYNCHRONOUS_NOT_SUPPORTED

```
#define PWM_E_DUTY_SYNCHRONOUS_NOT_SUPPORTED
```

Generated when given channel does not support duty synchronous feature. (PHASE_SHIFTED_SYNCED and PHASE_SHIFTED_COMPLEMENTARY)

For FTM, please note that Modified Combine mode does not support synchronous update Cn Therefore Pwm_↩ SetDutyCycle_NoUpdate and Pwm_SetPeriodAndDuty_NoUpdate should not be called in this case. For eMIOS, please note that channels using DAOC mode or channels in idle state does not support.

Definition at line 312 of file Pwm.h.

7.3.3.26 PWM_E_TRIGGER_MASK

```
#define PWM_E_TRIGGER_MASK
```

Generated when bit mask is not compatible with hardware register.

Definition at line 318 of file Pwm.h.

7.3.3.27 PWM_E_FORCE_OUTPUT_NOT_SUPPORTED

```
#define PWM_E_FORCE_OUTPUT_NOT_SUPPORTED
```

Generated when given channel does not support force output to zero feature.

Only channels in FlexPWM and FTM module are supported.

Definition at line 324 of file Pwm.h.

7.3.3.28 PWM_E_FORCE_OUT

```
#define PWM_E_FORCE_OUT
```

Generated when given channel does not support set force out feature.

Only channels in FlexPWM is supported.

Definition at line 330 of file Pwm.h.

7.3.3.29 PWM_E_PARAM_CONFIG

```
#define PWM_E_PARAM_CONFIG
```

Generated when the requested resource is configured to be unavailable on the current core.

Only multi-core configuration is available.

Definition at line 336 of file Pwm.h.

7.3.3.30 PWM_E_DEADTIME_RANGE

```
#define PWM_E_DEADTIME_RANGE
```

Generated when the configured dead time value is not valid.

Errors and exceptions that will be detected by the PWM driver

Definition at line 342 of file Pwm.h.

7.3.3.31 PWM_E_SETOUTPUTTOIDLE_NOT_SUPPORTED

```
#define PWM_E_SETOUTPUTTOIDLE_NOT_SUPPORTED
```

Generated when the configured dead time value is not valid.

Errors and exceptions that will be detected by the PWM driver

Definition at line 348 of file Pwm.h.

7.3.3.32 PWM_INIT_ID

```
#define PWM_INIT_ID
```

API service ID of Pwm_Init function.

Parameters used when raising an error/exception

Definition at line 354 of file Pwm.h.

7.3.3.33 PWM_DEINIT_ID

```
#define PWM_DEINIT_ID
```

API service ID of Pwm_DeInit function.

Parameters used when raising an error/exception

Definition at line 360 of file Pwm.h.

7.3.3.34 PWM_SETDUTYCYCLE_ID

```
#define PWM_SETDUTYCYCLE_ID
```

API service ID of Pwm_SetDutyCycle function.

Parameters used when raising an error/exception

Definition at line 366 of file Pwm.h.

7.3.3.35 PWM_SETPERIODANDDUTY_ID

```
#define PWM_SETPERIODANDDUTY_ID
```

API service ID of Pwm_SetPeriodAndDuty function.

Parameters used when raising an error/exception

Definition at line 372 of file Pwm.h.

7.3.3.36 PWM_SETOUTPUTTOIDLE_ID

```
#define PWM_SETOUTPUTTOIDLE_ID
```

API service ID of Pwm_SetOutputToIdle function.

Parameters used when raising an error/exception

Definition at line 378 of file Pwm.h.

7.3.3.37 PWM_GETOUTPUTSTATE_ID

```
#define PWM_GETOUTPUTSTATE_ID
```

API service ID of Pwm_GetOutputState function.

Parameters used when raising an error/exception

Definition at line 384 of file Pwm.h.

7.3.3.38 PWM_DISABLENOTIFICATION_ID

```
#define PWM_DISABLENOTIFICATION_ID
```

API service ID of Pwm_DisableNotification function.

Parameters used when raising an error/exception

Definition at line 390 of file Pwm.h.

7.3.3.39 PWM_ENABLENOTIFICATION_ID

```
#define PWM_ENABLENOTIFICATION_ID
```

API service ID of Pwm_EnableNotification function.

Parameters used when raising an error/exception

Definition at line 396 of file Pwm.h.

7.3.3.40 PWM_GETVERSIONINFO_ID

```
#define PWM_GETVERSIONINFO_ID
```

API service ID of Pwm_GetVersionInfo function.

Parameters used when raising an error/exception

Definition at line 402 of file Pwm.h.

7.3.3.41 PWM_SETPOWERSTATE_ID

```
#define PWM_SETPOWERSTATE_ID
```

API service ID of Pwm_SetPowerState function.

Parameters used when raising an error/exception

Definition at line 408 of file Pwm.h.

7.3.3.42 PWM_GETCURRENTPOWERSTATE_ID

```
#define PWM_GETCURRENTPOWERSTATE_ID
```

API service ID of Pwm_GetCurrentPowerState function.

Parameters used when raising an error/exception

Definition at line 414 of file Pwm.h.

7.3.3.43 PWM_GETTARGETPOWERSTATE_ID

```
#define PWM_GETTARGETPOWERSTATE_ID
```

API service ID of Pwm_GetTargetPowerState function.

Parameters used when raising an error/exception

Definition at line 420 of file Pwm.h.

7.3.3.44 PWM_PREPAREPOWERSTATE_ID

```
#define PWM_PREPAREPOWERSTATE_ID
```

API service ID of Pwm_PreparePowerState function.

Parameters used when raising an error/exception

Definition at line 426 of file Pwm.h.

7.3.3.45 PWM_MAIN_POWERTRANSITIONMANAGER_ID

```
#define PWM_MAIN_POWERTRANSITIONMANAGER_ID
```

API service ID of Pwm_Main_PowerTransitionManager function.

Parameters used when raising an error/exception

Definition at line 432 of file Pwm.h.

7.3.3.46 PWM_GETCHANNELSTATE_ID

```
#define PWM_GETCHANNELSTATE_ID
```

API service ID of Pwm_GetChannelState function.

Parameters used when raising an error/exception

Definition at line 438 of file Pwm.h.

7.3.3.47 PWM_FORCEOUTPUTTOZERO_ID

```
#define PWM_FORCEOUTPUTTOZERO_ID
```

API service ID of Pwm_ForceOutputToZero function.

Parameters used when raising an error/exception

Definition at line 444 of file Pwm.h.

7.3.3.48 PWM_SETCOUNTERBUS_ID

```
#define PWM_SETCOUNTERBUS_ID
```

API service ID of Pwm_SetCounterBus function.

Parameters used when raising an error/exception

Definition at line 450 of file Pwm.h.

7.3.3.49 PWM_SETCHANNELOUTPUT_ID

```
#define PWM_SETCHANNELOUTPUT_ID
```

API service ID of Pwm_SetChannelOutput function.

Parameters used when raising an error/exception

Definition at line 456 of file Pwm.h.

7.3.3.50 PWM_SETTRIGGERDELAY_ID

```
#define PWM_SETTRIGGERDELAY_ID
```

API service ID of Pwm_SetTriggerDelay function.

Parameters used when raising an error/exception

Definition at line 462 of file Pwm.h.

7.3.3.51 PWM_SETCLOCKMODE_ID

```
#define PWM_SETCLOCKMODE_ID
```

API service ID of Pwm_SetClockMode function.

Parameters used when raising an error/exception

Definition at line 468 of file Pwm.h.

7.3.3.52 PWM_SYNCUPDATE_ID

```
#define PWM_SYNCUPDATE_ID
```

API service ID of Pwm_SyncUpdate function.

Parameters used when raising an error/exception

Definition at line 474 of file Pwm.h.

7.3.3.53 PWM_SETPERIODANDDUTY_NO_UPDATE_ID

```
#define PWM_SETPERIODANDDUTY_NO_UPDATE_ID
```

API service ID of Pwm_SetPeriodAndDuty_NoUpdate function.

Parameters used when raising an error/exception

Definition at line 480 of file Pwm.h.

7.3.3.54 PWM_SETDUTYCYCLE_NO_UPDATE_ID

```
#define PWM_SETDUTYCYCLE_NO_UPDATE_ID
```

API service ID of Pwm_SetDutyCycle_NoUpdate function.

Parameters used when raising an error/exception

Definition at line 486 of file Pwm.h.

7.3.3.55 PWM_SETCHANNELDEADTIME_ID

```
#define PWM_SETCHANNELDEADTIME_ID
```

API service ID of Pwm_SetChannelDeadTime function.

Parameters used when raising an error/exception

Definition at line 492 of file Pwm.h.

7.3.3.56 PWM_SETPHASESHIFT_ID

```
#define PWM_SETPHASESHIFT_ID
```

API service ID of Pwm_SetPhaseShift function.

Parameters used when raising an error/exception

Definition at line 498 of file Pwm.h.

7.3.3.57 PWM_SETPHASESHIFTNOUPDATE_ID

```
#define PWM_SETPHASESHIFTNOUPDATE_ID
```

API service ID of Pwm_SetPhaseShift function.

Parameters used when raising an error/exception

Definition at line 504 of file Pwm.h.

7.3.3.58 PWM_ENABLETRIGGER_ID

```
#define PWM_ENABLETRIGGER_ID
```

API service ID of Pwm_EnableTrigger function.

Parameters used when raising an error/exception

Definition at line 510 of file Pwm.h.

7.3.3.59 PWM_DISABLETRIGGER_ID

```
#define PWM_DISABLETRIGGER_ID
```

API service ID of Pwm_DisableTrigger function.

Parameters used when raising an error/exception

Definition at line 516 of file Pwm.h.

7.3.3.60 PWM_RESETCOUNTERENABLE_ID

```
#define PWM_RESETCOUNTERENABLE_ID
```

API service ID of Pwm_ResetCounterEnable function.

Parameters used when raising an error/exception

Definition at line 522 of file Pwm.h.

7.3.3.61 PWM_RESETCOUNTERDISABLE_ID

```
#define PWM_RESETCOUNTERDISABLE_ID
```

API service ID of Pwm_ResetCounterDisable function.

Parameters used when raising an error/exception

Definition at line 528 of file Pwm.h.

7.3.3.62 PWM_MASKOUTPUT_ID

```
#define PWM_MASKOUTPUT_ID
```

API service ID of Pwm_MaskOutputs function.

Parameters used when raising an error/exception

Definition at line 534 of file Pwm.h.

7.3.3.63 PWM_UNMASKOUTPUT_ID

```
#define PWM_UNMASKOUTPUT_ID
```

API service ID of Pwm_UnMaskOutputs function.

Parameters used when raising an error/exception

Definition at line 540 of file Pwm.h.

7.3.3.64 PWM_DISABLERELOADNOTIF_ID

```
#define PWM_DISABLERELOADNOTIF_ID
```

API service ID of Pwm_DisableReloadNotification function.

Parameters used when raising an error/exception

Definition at line 546 of file Pwm.h.

7.3.3.65 PWM_ENABLERELOADNOTIF_ID

```
#define PWM_ENABLERELOADNOTIF_ID
```

API service ID of Pwm_EnableReloadNotification function.

Parameters used when raising an error/exception

Definition at line 552 of file Pwm.h.

7.3.3.66 PWM_SETCHANNELFORCEOUT_ID

```
#define PWM_SETCHANNELFORCEOUT_ID
```

API service ID of Pwm_SetChannelForceOut function.

Parameters used when raising an error/exception

Definition at line 558 of file Pwm.h.

7.3.3.67 PWM_SETDUTYPHASESHIFT_ID

```
#define PWM_SETDUTYPHASESHIFT_ID
```

API service ID of Pwm_SetDutyPhaseShift function.

Parameters used when raising an error/exception

Definition at line 564 of file Pwm.h.

7.3.4 Types Reference

7.3.4.1 Pwm_ChannelType

```
typedef uint8 Pwm_ChannelType
```

PWM channel type.

Definition at line 694 of file Pwm.h.

7.3.4.2 Pwm_InstanceType

```
typedef uint8 Pwm_InstanceType
```

PWM channel type.

Definition at line 700 of file Pwm.h.

7.3.4.3 Pwm_PeriodType

```
typedef Pwm_Ipw_PeriodType Pwm_PeriodType
```

PWM period type.

Definition at line 706 of file Pwm.h.

7.3.4.4 Pwm_NotifyType

```
typedef void(* Pwm_NotifyType) (void)
```

Channel notification typedef.

Definition at line 713 of file Pwm.h.

7.3.5 Enum Reference

7.3.5.1 Pwm_OutputStateType

```
enum Pwm_OutputStateType
```

Output signal level.

This enumeration specifies the return type of Pwm_GetOutputState

Enumerator

PWM_LOW	PWM level is logic low.
PWM_HIGH	PWM level is logic high.

Definition at line 575 of file Pwm.h.

7.3.5.2 Pwm_EdgeNotificationType

```
enum Pwm_EdgeNotificationType
```

Edge notification type.

This enumeration defines the type of edge transition that can generate a notification

Enumerator

PWM_FALLING_EDGE	A notification will be generated on the falling edge.
PWM_RISING_EDGE	A notification will be generated on the rising edge.
PWM_BOTH_EDGES	A notification will be generated on any state transition.

Definition at line 589 of file Pwm.h.

7.3.5.3 Pwm_ChannelClassType

```
enum Pwm_ChannelClassType
```

PWM channel class type.

This field will specify what parameters can be altered for the selected channel

Enumerator

PWM_VARIABLE_PERIOD	The period and duty cycle can be altered.
PWM_FIXED_PERIOD	Only the duty cycle can be altered.
PWM_FIXED_PERIOD_SHIFTED	Only the duty cycle can be altered.

Definition at line 605 of file Pwm.h.

7.3.5.4 Pwm_PowerStateType

```
enum Pwm_PowerStateType
```

Power state type.

Power state currently active or set as target power state.

Enumerator

PWM_FULL_POWER	PWM full power mode.
PWM_LOW_POWER	PWM low power mode.
PWM_NODEFINE_POWER	PWM no define power mode.

Definition at line 621 of file Pwm.h.

7.3.5.5 Pwm_PowerStateRequestResultType

```
enum Pwm_PowerStateRequestResultType
```

Result of power state type.

Result of the requests related to power state transitions.

Enumerator

PWM_SERVICE_ACCEPTED	Power state change executed.
PWM_NOT_INIT	Module not initialized.
PWM_SEQUENCE_ERROR	Wrong API call sequence.
PWM_HW_FAILURE	The HW module has a failure which prevents it to enter the required power state.
PWM_POWER_STATE_NOT_SUPP	Module does not support the requested power state.
PWM_TRANS_NOT_POSSIBLE	Module cannot transition directly from the current power state to the requested power state.

Definition at line 637 of file Pwm.h.

7.3.5.6 Pwm_PrescalerType

```
enum Pwm_PrescalerType
```

Prescaler type.

This enumeration specifies the possible types of prescalers used to configure base-clock timers

Enumerator

PWM_PRIMARY_PRESCALER	Selected value is the default/primary prescaler.
PWM_ALTERNATIVE_PRESCALER	Selected value is the alternative configured prescaler.

Definition at line 658 of file Pwm.h.

7.3.6 Function Reference

7.3.6.1 Pwm_Init()

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

This function initializes the Pwm driver.

The function Pwm_Init shall initialize all internals variables and the used PWM structure of the microcontroller according to the parameters specified in configPtr. If the duty cycle parameter equals:

```
- 0% or 100% : Then the PWM output signal shall be in the state according
    to the configured polarity parameter;

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

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

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

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

If development error detection for the Pwm module is enabled, the PWM functions shall check the parameter channelNumber and raise development error PWM_E_PARAM_CHANNEL if the parameter channelNumber is invalid.

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

```
- Report the error to the Development Error Tracer.
- Skip the desired functionality in order to avoid any corruptions of
    data or hardware registers (this means leave the function without any actions).
- Return pwm level low for the function Pwm_GetOutputState.
```

The function Pwm_Init shall disable all notifications. The reason is that the users of these notifications may not be ready. They can call Pwm_EnableNotification to start notifications.

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

If the PwmDevErorDetect switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter Error classification and chapter API specification (see PWM_SWS).

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

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

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

Parameters

in	*ConfigPtr*	Pointer to PWM top configuration structure

Returns

void

7.3.6.2 Pwm_DeInit()

```
void Pwm_DeInit (
             void  )
```

This function deinitializes the Pwm driver.

The function Pwm_DeInit shall deinitialize the PWM module.

```
The function Pwm_DeInit shall set the state of the PWM output signals
to the idle state.
The function Pwm_DeInit shall disable PWM interrupts and PWM signal edge notifications.
The function Pwm_DeInit shall be pre-compile time configurable On-Off by the
configuration parameter PwmDeInitApi function prototype.
If development error detection for the Pwm module is enabled,
when a development error occurs, the corresponding PWM function shall:
    - Report the error to the Development Error Tracer.
    - Skip the desired functionality in order to avoid any corruptions
        of data or hardware registers (this means leave the function without any actions).
    - Return pwm level low for the function Pwm_GetOutputState.

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

Returns

void

7.3.6.3 Pwm_SetDutyCycle()

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

This function sets the dutycycle for the specified Pwm channel.

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

```
The function Pwm_SetDutyCycle shall set the PWM output state according
to the configured polarity parameter, when the duty cycle = 0% or 100%.
The function Pwm_SetDutyCycle shall modulate the PWM output signal according
to parameters period, duty cycle and configured polarity,
when the duty cycle > 0 % and < 100%.

If development error detection for the Pwm module is enabled,
the PWM functions shall check the parameter channelNumber and raise development error
PWM_E_PARAM_CHANNEL if the parameter channelNumber is invalid.

If development error detection for the Pwm module is enabled,
when a development error occurs, the corresponding PWM function shall:
    - Report the error to the Development Error Tracer.
    - Skip the desired functionality in order to avoid any corruptions
        of data or hardware registers (this means leave the function without any actions).
    - Return pwm level low for the function Pwm_GetOutputState.

The Pwm module shall comply with the following scaling scheme for the duty cycle:
    - 0x0000 means 0%.
    - 0x8000 means 100%.
    - 0x8000 gives the highest resolution while allowing 100% duty cycle to be
        represented with a 16 bit value.
        As an implementation guide, the following source code example is given:
        AbsoluteDutyCycle = ((uint32)AbsolutePeriodTime * RelativeDutyCycle) >> 15;

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

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration
in	*DutyCycle*	Pwm dutycycle value 0x0000 for 0% ... 0x8000 for 100%

Returns

void

7.3.6.4 Pwm_SetPeriodAndDuty()

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

This function sets the period and the dutycycle for the specified Pwm channel.

The function Pwm__SetPeriodAndDuty shall set the duty cycle of the PWM channel.

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

If development error detection for the Pwm module is enabled,
the PWM functions shall check the parameter channelNumber and raise development error
PWM_E_PARAM_CHANNEL if the parameter channelNumber is invalid.

If development error detection for the Pwm module is enabled,
when a development error occurs, the corresponding PWM function shall:
    - Report the error to the Development Error Tracer.
    - Skip the desired functionality in order to avoid any corruptions
        of data or hardware registers (this means leave the function without any actions).
    - Return pwm level low for the function Pwm_GetOutputState.

The Pwm module shall comply with the following scaling scheme for the duty cycle:
    - 0x0000 means 0%.
    - 0x8000 means 100%.
    - 0x8000 gives the highest resolution while allowing 100% duty cycle to be
        represented with a 16 bit value.
        As an implementation guide, the following source code example is given:
        AbsoluteDutyCycle = ((uint32)AbsolutePeriodTime * RelativeDutyCycle) >> 15;

If the PwmDevErorDetect switch is enabled, API parameter checking is enabled.
The detailed description of the detected errors can be found in chapter
Error classification and chapter API specification (see PWM_SWS).

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

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration
in	*Period*	Pwm signal period value
in	*DutyCycle*	Pwm dutycycle value 0x0000 for 0% ... 0x8000 for 100%

Returns

void

7.3.6.5 Pwm__SetOutputToIdle()

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

This function sets the generated pwm signal to the idle value configured.

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

If development error detection for the Pwm module is enabled, the PWM functions shall check the parameter channelNumber and raise development error PWM_E_PARAM_CHANNEL if the parameter channelNumber is invalid.

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

- Report the error to the Development Error Tracer.
- Skip the desired functionality in order to avoid any corruptions of data or hardware registers (this means leave the function without any actions).

If the PwmDevErorDetect switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter Error classification and chapter API specification (see PWM_SWS).

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

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

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

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration

Returns

void

7.3.6.6 Pwm_GetOutputState()

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

This function returns the signal output state.

The function Pwm_GetOutputState shall read the internal state of the PWM output signal and return it as defined in the diagram below (see PWM_SWS).

If development error detection for the Pwm module is enabled, the PWM functions shall check the parameter channelNumber and raise development error PWM_E_PARAM_CHANNEL if the parameter channelNumber is invalid.

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

- Report the error to the Development Error Tracer.
- Skip the desired functionality in order to avoid any corruptions of data or hardware registers (this means leave the function without any actions).
- Return pwm level low for the function Pwm_GetOutputState.

If the PwmDevErorDetect switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter Error classification and chapter API specification (see PWM_SWS).

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

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

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration

Returns

Pwm_OutputStateType Pwm signal output logic value

Return values

PWM_LOW	The output state of PWM channel is low
PWM_HIGH	The output state of PWM channel is high

7.3.6.7 Pwm_EnableNotification()

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

This function enables the user notifications.

The function Pwm_EnableNotification shall enable the PWM signal edge notification according to notification parameter. If development error detection for the Pwm module is enabled:

- The PWM functions shall check the parameter channelNumber and raise development error PWM_E_PARAM_CHANNEL if the parameter channelNumber is invalid.

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

```
- Report the error to the Development Error Tracer.
- Skip the desired functionality in order to avoid any corruptions of
    data or hardware registers (this means leave the function without any actions).
- Return pwm level low for the function Pwm_GetOutputState.
```

If the PwmDevErorDetect switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter Error classification and chapter API specification (see PWM_SWS).

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

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration
in	*Notification*	Notification type to be enabled

Returns

void

7.3.6.8 Pwm_DisableNotification()

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

This function disables the user notifications.

If development error detection for the Pwm module is enabled:

- The PWM functions shall check the parameter channelNumber and raise development error PWM_E_PARAM_CHANNEL if the parameter channelNumber is invalid.

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

- Report the error to the Development Error Tracer.
- Skip the desired functionality in order to avoid any corruptions of data or hardware registers (this means leave the function without any actions).

- Return pwm level low for the function Pwm_GetOutputState.

If the PwmDevErorDetect switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter Error classification and chapter API specification (see PWM_SWS).

All functions from the PWM module except Pwm_Init, Pwm_DeInit and Pwm_GetVersionInfo shall be re-entrant for different PWM channel numbers. In order to keep a simple module implementation, no check of PWM088 must be performed by the module. The function Pwm_DisableNotification shall be pre compile time configurable On-Off by the configuration parameter: PwmNotificationSupported.

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

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration

Returns

void

7.3.6.9 Pwm_GetVersionInfo()

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

This function returns Pwm driver version details.

The function Pwm_GetVersionInfo shall return the version information of this module. The version information includes: Module Id, Vendor Id, Vendor specific version number.

Parameters

out	*versioninfo*	Pointer to Std_VersionInfoType output variable

Returns

void

7.3.6.10 Pwm_GetChannelState()

```
uint16 Pwm_GetChannelState (
            Pwm_ChannelType ChannelNumber )
```

This function returns the duty cycle of the channel passed as parameter.

The function Pwm_GetChannelState shall return the DutyCycle of the channel. In case the channel is idle, the returned value will be zero.

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration

Returns

uint16 DutyCycle of the requested channel

7.3.6.11 Pwm_ForceOutputToZero()

```
void Pwm_ForceOutputToZero (
            Pwm_ChannelType ChannelNumber,
            boolean Force )
```

This function enables-disables the forcing of the output of a given channel to logic 0.

The function Pwm_GetChannelState shall return the DutyCycle of the channel. In case the channel is idle, the returned value will be zero.

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration
in	*Force*	Boolean value to state if the output of the channel will be forced to zero logic or not

Returns

void

7.3.6.12 Pwm_SetClockMode()

```
void Pwm_SetClockMode (
            Pwm_PrescalerType Prescaler )
```

Implementation specific function to change the peripheral clock frequency.

This function is useful to set the prescalers that divide the PWM channels clock frequency.

Parameters

in	*Prescaler*	Prescaler type

Returns

void

7.3.6.13 Pwm_SetChannelDeadTime()

```
void Pwm_SetChannelDeadTime (
            Pwm_ChannelType ChannelNumber,
            uint16 DeadTimeTicks )
```

This function is used to update the deadtime at runtime for Pwm channels.

Parameters

in	*ChannelNumber*	Pwm channel id
in	*DeadTimeTicks*	Dead Time value in ticks

Returns

void

7.3.6.14 Pwm_SetDutyCycle_NoUpdate()

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

This function sets the values of dutycycle for the specified Pwm channel but without updating the PWM output.

The function Pwm_SetDutyCycle_NoUpdate shall set the duty cycle of the PWM channel to the coresponding hardware buffers without updating the wave form on the output pin. This feature will allow a pre-buffering of new PWM duty cycle values for several channel, which can all be updated synchronos by calling Pwm_SyncUpdate.

The function Pwm_SetDutyCycle_NoUpdate shall set the PWM output state according to the configured polarity parameter, when the duty cycle = 0% or 100%. The function Pwm_SetDutyCycle_NoUpdate shall modulate the PWM output signal according to parameters period, duty cycle and configured polarity, when the duty cycle > 0 % and < 100%.

If development error detection for the Pwm module is enabled, the PWM functions shall check the parameter channelNumber and raise development error PWM_E_PARAM_CHANNEL if the parameter channelNumber is invalid.

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

- Report the error to the Development Error Tracer.
- Skip the desired functionality in order to avoid any corruptions of data or hardware registers (this means leave the function without any actions).
- Return pwm level low for the function Pwm_GetOutputState.

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

- 0x0000 means 0%.
- 0x8000 means 100%.
- 0x8000 gives the highest resolution while allowing 100% duty cycle to be represented with a 16 bit value. As an implementation guide, the following source code example is given: AbsoluteDutyCycle = ((uint32)Absolute↩ PeriodTime ∗ RelativeDutyCycle) >> 15;

If the PwmDevErorDetect switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter Error classification and chapter API specification (see PWM_SWS).

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration
in	*DutyCycle*	Pwm dutycycle value 0x0000 for 0% ... 0x8000 for 100%

Returns

void

7.3.6.15 Pwm_SetPeriodAndDuty_NoUpdate()

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

This function sets the values of the period and the dutycycle for the specified Pwm channel into the hardware buffers but without updating the PWM output..

The function Pwm_SetPeriodAndDuty_NoUpdate shall set the period and duty cycle of the PWM channel to the coresponding hardware buffers without updating the wave form on the output pin. This feature will allow a pre-buffering of new PWM duty cycle values for several channel, which can all be updated synchronos by calling Pwm_SyncUpdate.

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

If development error detection for the Pwm module is enabled, the PWM functions shall check the parameter channelNumber and raise development error PWM_E_PARAM_CHANNEL if the parameter channelNumber is invalid.

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

- Report the error to the Development Error Tracer.
- Skip the desired functionality in order to avoid any corruptions of data or hardware registers (this means leave the function without any actions).
- Return pwm level low for the function Pwm_GetOutputState.

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

- 0x0000 means 0%.
- 0x8000 means 100%.
- 0x8000 gives the highest resolution while allowing 100% duty cycle to be represented with a 16 bit value. As an implementation guide, the following source code example is given: AbsoluteDutyCycle = ((uint32)Absolute←PeriodTime * RelativeDutyCycle) >> 15;

If the PwmDevErorDetect switch is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter Error classification and chapter API specification (see PWM_SWS).

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

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration
in	*Period*	Pwm signal period value
in	*DutyCycle*	Pwm dutycycle value 0x0000 for 0% ... 0x8000 for 100%

Returns

void

7.3.6.16 Pwm_SetPhaseShift_NoUpdate()

```
void Pwm_SetPhaseShift_NoUpdate (
            Pwm_ChannelType ChannelNumber,
            Pwm_PeriodType Period,
            uint16 PhaseShift )
```

This function set phase shift value and also force duty cycle to 50%. The output will take effect after Pwm_Sync↩ Update be called.

In order to have Phase-Shifted Full-Bridge controller, Pwm_SetPhaseShift/Pwm_SetPhaseShift_NoUpdate is introduced. Pwm_SetPhaseShift allows to set both phase shift value and period, the duty value is fixed to 50%.

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration
in	*Period*	Pwm signal period value
in	*PhaseShift*	Phase shift value

Returns

void

7.3.6.17 Pwm_SyncUpdate()

```
void Pwm_SyncUpdate (
            uint8 ModuleId )
```

Implementation specific function to updates duty synchronization.

This function is used to update duty synchronization for channels in given module, this should be called after calling Pwm_SetPeriodAndDuty_NoUpdate() or Pwm_SetDutyCycle_NoUpdate() API.

Parameters

in	*Module↩ Id*	pwm module id(instance ID) Ex : PWM_EMIOS_INSTANCE_0 PWM_EMIOS_INSTANCE_1 ... PWM_FTM_INSTANCE_0

Returns

void

7.3.6.18 Pwm_SetPhaseShift()

```
void Pwm_SetPhaseShift (
            Pwm_ChannelType ChannelNumber,
            Pwm_PeriodType Period,
            uint16 PhaseShift )
```

This function set phase shift value and also force duty cycle to 50%.

In order to have Phase-Shifted Full-Bridge controller, Pwm_SetPhaseShift is introduced. Pwm_SetPhaseShift allows to set both phase shift value and period, the duty value is fixed to 50%.

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration
in	*Period*	Pwm signal period value
in	*PhaseShift*	Phase shift value

Returns

void

7.3.6.19 Pwm_SetDutyPhaseShift()

```
void Pwm_SetDutyPhaseShift (
            Pwm_ChannelType ChannelNumber,
            uint16 DutyCycle,
            uint16 PhaseShift,
            boolean SyncUpdate )
```

This function set phase shift and duty cycle value (as immediate or synchronized base on API parameter SyncUpdate)

Pwm_SetDutyPhaseShift allows to set both phase shift and duty cycle value, The phase shift is the offset of the leading edge of the signal in respect to period starting point.

Parameters

in	*ChannelNumber*	Pwm Channel Id in the configuration
in	*DutyCycle*	Pwm duty cycle value 0x0000 for 0% ... 0x8000 for 100%
in	*PhaseShift*	Phase shift value (in ticks)
in	*SyncUpdate*	Update duty and phases shift value synchronization for channels in given module or not TRUE Set the phase shift and duty cycle value base on the synchronization when calling Pwm_SyncUpdate. FALSE Set phase shift and duty cycle value immediately

Returns

void

7.3.6.20 Pwm_EnableTrigger()

```
void Pwm_EnableTrigger (
            uint8 TriggerHostId,
            uint16 TriggerMask )
```

This function enable trigger generation for specific source.

Corresponding bits with trigger source as bellow: Bit 0 Channel 2 Trigger Enable Bit 1 Channel 3 Trigger Enable Bit 2 Channel 4 Trigger Enable Bit 3 Channel 5 Trigger Enable Bit 4 Channel 0 Trigger Enable Bit 5 Channel 1 Trigger Enable Bit 6 Initialization Trigger Enable

Parameters

in	*Trigger↩ HostId*	Pwm hardware module id
in	*TriggerMask*	Bit mask will be set to enable trigger with corresponding sources.

Returns

void

7.3.6.21 Pwm_DisableTrigger()

```
void Pwm_DisableTrigger (
            uint8 TriggerHostId,
            uint16 TriggerMask )
```

This function disable trigger generation for specific source.

Corresponding bits with trigger source as bellow: Bit 0 Channel 2 Trigger Enable Bit 1 Channel 3 Trigger Enable Bit 2 Channel 4 Trigger Enable Bit 3 Channel 5 Trigger Enable Bit 4 Channel 0 Trigger Enable Bit 5 Channel 1 Trigger Enable Bit 6 Initialization Trigger Enable

Parameters

in	*Trigger↩ HostId*	Pwm hardware module id
in	*TriggerMask*	Bit mask will be cleared to disable trigger with corresponding sources.

Returns

void

7.3.6.22 Pwm_ResetCounterEnable()

```
void Pwm_ResetCounterEnable (
            uint8 ModuleId )
```

This function shall enable the PWM timer HW counter reset by Pwm_SyncUpdate() function.

Parameters

in	*Module↩ Id*	Pwm hardware module id

Returns

void

7.3.6.23 Pwm_ResetCounterDisable()

```
void Pwm_ResetCounterDisable (
            uint8 ModuleId )
```

This function shall disable the PWM timer HW counter reset by Pwm_SyncUpdate() function.

Parameters

in	*Module↩ Id*	Pwm hardware module id

Returns

void

7.3.6.24 Pwm_MaskOutputs()

```
void Pwm_MaskOutputs (
            uint8 ModuleId,
            uint8 ChannelMask )
```

This function force channels output to their inactive state.

Corresponding bits with channel will be masked: Bit 0 Channel 0 Output Mask Bit 1 Channel 1 Output Mask Bit 2 Channel 2 Output Mask Bit 3 Channel 3 Output Mask Bit 4 Channel 4 Output Mask Bit 5 Channel 5 Output Mask

Parameters

in	*ModuleId*	Pwm hardware module id
in	*ChannelMask*	Bit mask will be set to mask corresponding channel.

Returns

void

7.3.6.25 Pwm_UnMaskOutputs()

```
void Pwm_UnMaskOutputs (
            uint8 ModuleId,
            uint8 ChannelMask )
```

This function puts channels output to normal operation state.

Corresponding bits with channel will be masked: Bit 0 Channel 0 Output Mask Bit 1 Channel 1 Output Mask Bit 2 Channel 2 Output Mask Bit 3 Channel 3 Output Mask Bit 4 Channel 4 Output Mask Bit 5 Channel 5 Output Mask

Parameters

in	*ModuleId*	Pwm hardware module id
in	*ChannelMask*	Bit mask will be cleared to unmask corresponding channel.

Returns

void

7.3.6.26 Pwm_SetPowerState()

```
Std_ReturnType Pwm_SetPowerState (
            Pwm_PowerStateRequestResultType * Result )
```

Enters the already prepared power state.

This API configures the Pwm module so that it enters the already prepared power state, chosen between a predefined set of configured ones.

Parameters

out	*Result*	Pointer to a variable to store the result of this function

Returns

Std_ReturnType Standard return type.

Return values

E_OK	Power Mode changed.
E_NOT_OK	Request rejected.

7.3.6.27 Pwm_GetCurrentPowerState()

```
Std_ReturnType Pwm_GetCurrentPowerState (
             Pwm_PowerStateType * CurrentPowerState,
             Pwm_PowerStateRequestResultType * Result )
```

Get the current power state of the Pwm HW unit.

This API returns the current power state of the Pwm HW unit.

Parameters

out	*CurrentPowerState*	The current power mode of the Pwm HW Unit is returned in this parameter
out	*Result*	Pointer to a variable to store the result of this function

Returns

Std_ReturnType Standard return type.

Return values

E_OK	Mode could be read.
E_NOT_OK	Service is rejected.

7.3.6.28 Pwm_GetTargetPowerState()

```
Std_ReturnType Pwm_GetTargetPowerState (
            Pwm_PowerStateType * TargetPowerState,
            Pwm_PowerStateRequestResultType * Result )
```

Get the target power state of the Pwm HW unit.

This API returns the target power state of the Pwm HW unit.

Parameters

out	*TargetPowerState*	The Target power mode of the Pwm HW Unit is returned in this parameter.
out	*Result*	Pointer to a variable to store the result of this function.

Returns

Std_ReturnType Standard return type.

Return values

E_OK	Mode could be read.
E_NOT_OK	Service is rejected.

7.3.6.29 Pwm_PreparePowerState()

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

Starts the needed process to allow the Pwm HW module to enter the requested power state.

This API starts the needed process to allow the Pwm HW module to enter the requested power state.

Parameters

in	*PowerState*	The target power state intended to be attained.
out	*Result*	Pointer to a variable to store the result of this function.

Returns

Std_ReturnType Standard return type.

Return values

E_OK	Mode could be read.
E_NOT_OK	Service is rejected.

7.3.6.30 Pwm_Main_PowerTransitionManager()

```
void Pwm_Main_PowerTransitionManager (
            void  )
```

Cyclically called and supervises the power state transitions.

This API is cyclically called and supervises the power state transitions, checking for the readiness of the module and issuing the callbacks.

Parameters

in	*none*	
out	*none*	

Returns

none

Chapter 8

Data Structure Documentation

8.1 Flexio_Pwm_Ip_ChannelConfigType Struct Reference

PWM configuration parameters structure.

```
#include <Flexio_Pwm_Ip_Types.h>
```

Data Fields

- uint8 TimerId

  Flexio used timer index.
- uint8 PinId

  Flexio used pin index.
- uint16 Period

  Pwm period in ticks.
- uint16 DutyCycle

  Pwm duty cycle in ticks.
- Flexio_Pwm_Ip_InterruptType IrqMode

  Interrupt mode.
- Flexio_Pwm_Ip_IplNotificationType IplCallback

  User notification callback for IPL.
- Flexio_Pwm_Ip_HldNotificationType HldCallback

  User notification callback for HLD.

8.1.1 Detailed Description

PWM configuration parameters structure.

Flexio IP specific channel configuration structure type

Definition at line 250 of file Flexio_Pwm_Ip_Types.h.

8.1.2 Field Documentation

8.1.2.1 TimerId

```
uint8 TimerId
```

Flexio used timer index.

Definition at line 253 of file Flexio_Pwm_Ip_Types.h.

8.1.2.2 PinId

```
uint8 PinId
```

Flexio used pin index.

Definition at line 255 of file Flexio_Pwm_Ip_Types.h.

8.1.2.3 Period

```
uint16 Period
```

Pwm period in ticks.

Definition at line 261 of file Flexio_Pwm_Ip_Types.h.

8.1.2.4 DutyCycle

```
uint16 DutyCycle
```

Pwm duty cycle in ticks.

Definition at line 263 of file Flexio_Pwm_Ip_Types.h.

8.1.2.5 IrqMode

```
Flexio_Pwm_Ip_InterruptType IrqMode
```

Interrupt mode.

Definition at line 269 of file Flexio_Pwm_Ip_Types.h.

8.1.2.6 IplCallback

```
Flexio_Pwm_Ip_IplNotificationType IplCallback
```

User notification callback for IPL.

Definition at line 271 of file Flexio_Pwm_Ip_Types.h.

8.1.2.7 HldCallback

```
Flexio_Pwm_Ip_HldNotificationType HldCallback
```

User notification callback for HLD.

Definition at line 273 of file Flexio_Pwm_Ip_Types.h.

8.2 Flexio_Pwm_Ip_HldNotificationType Struct Reference

Structure for notification.

```
#include <Flexio_Pwm_Ip_Types.h>
```

Data Fields

- Flexio_Pwm_Ip_HldCallbackType CbFunction

  Callback function pointer.
- uint8 CbParameter

  Callback function parameter pointer.

8.2.1 Detailed Description

Structure for notification.

The structure used to notification

Definition at line 236 of file Flexio_Pwm_Ip_Types.h.

8.2.2 Field Documentation

8.2.2.1 CbFunction

```
Flexio_Pwm_Ip_HldCallbackType CbFunction
```

Callback function pointer.

Definition at line 239 of file Flexio_Pwm_Ip_Types.h.

8.2.2.2 CbParameter

```
uint8 CbParameter
```

Callback function parameter pointer.

Definition at line 241 of file Flexio_Pwm_Ip_Types.h.

8.3 Flexio_Pwm_Ip_IplNotificationType Struct Reference

Structure for notification.

```
#include <Flexio_Pwm_Ip_Types.h>
```

Data Fields

- Flexio_Pwm_Ip_IplCallbackType CbFunction
  Callback function pointer.
- void * CbParameter
  Callback function parameter pointer.

8.3.1 Detailed Description

Structure for notification.

The structure used to notification

Definition at line 222 of file Flexio_Pwm_Ip_Types.h.

8.3.2 Field Documentation

8.3.2.1 CbFunction

```
Flexio_Pwm_Ip_IplCallbackType CbFunction
```

Callback function pointer.

Definition at line 225 of file Flexio_Pwm_Ip_Types.h.

8.3.2.2 CbParameter

```
void* CbParameter
```

Callback function parameter pointer.

Definition at line 227 of file Flexio_Pwm_Ip_Types.h.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.