User Manual

for MPC563XM PWM MCAL Driver

Document Number: UM14PWMASR3.0R2.0.0
Rev. 1.1

freescale™

Contents

Section Number **Title** **Page**

Chapter 1
Revision History

Chapter 2
Introduction

Chapter 3
Driver

Section Number	Title	Page

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	14.02.2011	Sinu Joji Isac	Monaco 1.9.0 release
1.1	20.12.2011	Subramanya M Naik	Updated for MPC5634M RTM 2.0.0 Release

Chapter 2
Introduction

This User Manual describes AUTOSAR Pulse Width Modulation (PWM) for MPC5634M.

AUTOSAR PWM driver configuration parameters and deviations from the specification are described in Pwm 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 Freescale Semiconductor:

Table 2-1. MPC5634M Derivatives

Freescale Semiconductor	mpc5634m_bga208, mpc5634m_qfp144, mpc5634m_qfp176

All of the above microcontroller devices are collectively named as MPC5634M.

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 type: Bold is used for important terms, notes and warnings.

Italic font: Italic typeface is 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.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
GPIO	General Purpose Input Output
GPT	General Purpose Timer
MCAL	Microntroller Abstraction Layer
MIDE	Multi Integrated Development Environment
PWM	Pulse Width Modulation

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
VLE	Variable Length Encoding
XML	Extensible Markup Language

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	AUTOSAR 3.0 PWM Driver Software Specification Document.	V2.2.0 R3.0 Rev 0001
2	MPC5634M Reference Manual	Rev. 6, 4 October 2011

Chapter 3
Driver

3.1 Requirements

Requirements for this driver are detailed in the AUTOSAR 3.0Pwm Driver Software Specification document (See Reference List).

3.2 Driver Design Summary

The AUTOSAR PWM Driver Specification defines the functionality, API and the configuration of the AUTOSAR Basic Software module PWM Driver. Each PWM channel is linked to hardware PWM which belongs to the microcontroller. EMIOS unified channels on EMIOS_A are used to implement the PWM functionality.

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

The modes supported:

- OPWFMB
- OPWMB

The channels supporting OPWFMB modes (with Variable Period and Variable Duty Cycle and having bus Internal Counter) are

- Ch0, Ch8, Ch9, Ch10, Ch12, Ch14, Ch15, Ch23.

The channels supporting OPWMB mode (with Fixed Period and Variable Duty Cycle and having bus Counter BusA or Bus Diverse) are

- Ch0, Ch2, Ch4, Ch8, Ch9, Ch10, Ch11, Ch12, Ch13, Ch14, Ch15, Ch23.

3.3 Additional Requirements and Deviations

The driver deviates from the AUTOSAR PWM Driver software specification in some places

The deviations from the AUTOSAR PWM Driver software specification are listed in table 2-3.

Table 3-1. Deviations Status Column Description

Term	Definition
N/A	Not Applicable
N/V	Not Verifiable
N/S	Out of Scope
N/R	Unclear Requirement
N/F	Not Fully Implemented
N/I	Not Implemented

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

Table 3-2. PWM Deviations Table

Requirement	Status	Description	Notes
PWM070	N/S	All time units used within the API services of the PWM driver shall be of the unit ticks	Upper layer responsibility to pass ticks to PWM driver API's

3.3.1 eMIOS specific implementation details

eMIOS counter:

Please note that the eMIOS internal hardware counter will always start from 0x1. This is true for all channel types in the current implementation: OPWFMB, OPWMB.

The configured default period value is incremented in the code by 1 at Pwm_Init(). Pwm_SetPeriodAndDuty() does not perform any other operation on the period value parameter and the value is written in the hw registers. Please also see the details for PwmPeriodDefault 2.6

Notifications:

Please note that not all combinations of signal polarity and notifications are supported by the EMIOS implementation:

Unsupported combination for each mode:

OPWFMB:

Polarity PWM_HIGH && Notification PWM_FALLING_EDGE

Pwm_Polarity PWM_LOW && Notification PWM_RISING_EDGE

OPWMB:

Polarity PWM_HIGH && Notification PWM_RISING_EDGE

Pwm_Polarity PWM_LOW && Notification PWM_FALLING_EDGE

In this case the following DET error will be generated:

Det_ReportError(PWM_MODULE_ID, 0, PWM_ENABLENOTIFICATION_ID, PWM_E_PARAM_NOTIFICATION);

Signal Period for the channels in OPWMB

Period value is calculated based on the unified channel registers, EMIOS_CADR and EMIOS_CBDR. When the counter bus is started the registers EMIOS_CADR and EMIOS_CBDR are initialized with non-zero values. So, care should be taken such that the counter bus should be started before initializing a channel in OPWMB mode. If counter bus is not started before initializing a channel in OPWMB mode, then care should be taken such that Period value should not be zero in the driver code to occur a Zero division on Duty cycle calculation.

3.4 Function Definitions

APIs of all functions supported by the driver are as per AUTOSAR PWM Driver software specification version 3.0

3.4.1 Function Pwm_Init

This function initializes the Pwm driver.

Prototype: `void Pwm_Init(const Pwm_ConfigType *ConfigPtr);`

Table 3-3. Pwm_Init Arguments

Type	Name	Direction	Description
const `Pwm_ConfigType` *	ConfigPtr	**input**	pointer to PWM top configuration structure

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.

3.4.2 Function Pwm_DeInit

This function deinitializes the Pwm driver.

Prototype: `void Pwm_DeInit(void);`

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.

3.4.3 Function Pwm_SetDutyCycle

This function sets the dutycycle for the specified Pwm channel.

Prototype: `void Pwm_SetDutyCycle(Pwm_ChannelType ChannelNumber, uint16 DutyCycle);`

Table 3-4. Pwm_SetDutyCycle Arguments

Type	Name	Direction	Description
`Pwm_ChannelType`	ChannelNumber	**input**	- pwm channel id
`uint16`	DutyCycle	**input**	- pwm dutycycle value 0x0000 for 0% ... 0x8000 for 100%

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%.

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 width of the duty cycle parameter is 16 Bits.

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).

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.

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

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

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.

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.

Note

MISRA 2004 Required Rule 10.1, Implicit conversion changes signedness

3.4.4 Function Pwm_SetPeriodAndDuty

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

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

Table 3-5. Pwm_SetPeriodAndDuty Arguments

Type	Name	Direction	Description
`Pwm_ChannelType`	ChannelNumber	**input**	- pwm channel id
`Pwm_PeriodType`	Period	**input**	- pwm signal period value
`uint16`	DutyCycle	**input**	- pwm dutycycle value 0x0000 for 0% ... 0x8000 for 100%

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

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

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

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 width of the duty cycle parameter is 16 Bits.

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).

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.

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

The function Pwm_SetPeriodAndDuty shall update the period always at the end of the current period if supported by the implementation and configured with PwmPeriodUpdatedEndperiod.

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

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.

3.4.5 Function Pwm_SetOutputToIdle

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

Prototype: `void Pwm_SetOutputToIdle(Pwm_ChannelType ChannelNumber);`

Table 3-6. Pwm_SetOutputToIdle Arguments

Type	Name	Direction	Description
`Pwm_ChannelType`	ChannelNumber	**input**	- pwm channel id

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).
- 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_SetOutputToIdle shall be pre compile time configurable On/Off by the configuration parameter PwmSetOutputToIdle.

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.

3.4.6 Function Pwm_GetOutputState

This function returns the signal output state.

Prototype: `Pwm_OutputStateType Pwm_GetOutputState(Pwm_ChannelType ChannelNumber);`

Table 3-7. Pwm_GetOutputState Arguments

Type	Name	Direction	Description
`Pwm_ChannelType`	ChannelNumber	**input**	- pwm channel id

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).

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_GetOutputState shall be pre compile time configurable On/Off by the configuration parameter: PwmGetOutputState.

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

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

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.

Return: Pwm_OutputStateType pwm signal output logic value -PWM_LOW - The output state of PWM channel is low -PWM_HIGH - The output state of PWM channel is high

3.4.7 Function Pwm_DisableNotification

This function disables the user notifications.

Prototype: `void Pwm_DisableNotification(Pwm_ChannelType ChannelNumber);`

Table 3-8. Pwm_DisableNotification Arguments

Type	Name	Direction	Description
`Pwm_ChannelType`	ChannelNumber	**input**	- pwm channel id

The function Pwm_DisableNotification shall disable the PWM signal edge notification

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.

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

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.

3.4.8 Function Pwm_EnableNotification

This function enables the user notifications.

Prototype: `void Pwm_EnableNotification(Pwm_ChannelType ChannelNumber, Pwm_EdgeNotificationType Notification);`

Table 3-9. Pwm_EnableNotification Arguments

Type	Name	Direction	Description
`Pwm_ChannelType`	ChannelNumber	**input**	- pwm channel id
`Pwm_EdgeNotificationType`	Notification	**input**	- notification type to be enabled

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).

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.

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

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.

3.4.9 Function Pwm_GetVersionInfo

returns Pwm driver version details

Prototype: `void Pwm_GetVersionInfo(Std_VersionInfoType *versioninfo);`

Table 3-10. Pwm_GetVersionInfo Arguments

Type	Name	Direction	Description
`Std_VersionInfoType *`	versioninfo	**input, output**	- pointer to `Std_VersionInfoType` output variable

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

3.4.10 Function Pwm_SetCounterBus

Implementation specific function to change the frequency of pwm channels running on PWM_MODE_OPWMB or PWM_MODE_OPWMT mode.

Prototype: `void Pwm_SetCounterBus(Pwm_ChannelType ChannelNumber, uint32 Bus);`

Table 3-11. Pwm_SetCounterBus Arguments

Type	Name	Direction	Description
`Pwm_ChannelType`	ChannelNumber	**input**	- pwm channel id
`uint32`	Bus	**input**	- the eMIOS bus to change to

This function is useful to change the frequency of the output PWM signal between two counter buses frequency

If no DET error reported then the selected counter bus period will be read and the function `Pwm_LLD_SetDutyCycle()` will be called to change the output signal frequency and dutycycle

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.

If development error detection for the Pwm module is enabled:

- this function 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

3.4.11 Function Pwm_SetChannelOutput

Implementation specific function to set the state of the PWM pin as requested for the current cycle.

Prototype: `void Pwm_SetChannelOutput(Pwm_ChannelType ChannelNumber, Pwm_StateType State);`

Table 3-12. Pwm_SetChannelOutput Arguments

Type	Name	Direction	Description
`Pwm_ChannelType`	ChannelNumber	**input**	- pwm channel id
`Pwm_StateType`	State	**input**	- Active/Inactive state of the channel

This function is useful to set the state of the PWM pin as requested for the current cycle and continues with normal PWM operation from the next cycle

If no DET error reported then the state of the PWM pin will be Active (Same as the configured polarity) or Inactive (Opposite state of the configured polarity) based on the passed input parameter.

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.

If development error detection for the Pwm module is enabled:

- this function 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

3.4.12 Function Pwm_SetClockMode

Implementation specific function to change the peripheral clock frequency.

Prototype: `void Pwm_SetClockMode(Pwm_SelectPrescalerType Prescaler);`

Table 3-13. Pwm_SetClockMode Arguments

Type	Name	Direction	Description
`Pwm_SelectPrescalerType`	Prescaler	**input**	- prescaler type

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

If no DET error reported then the prescalers of the PWM channels will be set.

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.

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.

3.5 Symbolic Names DISCLAIMER

All containers having the symbolic name tag set as true in the Autosar schema will generate defines like:

#define <Container_Short_Name> <Container_ID>

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

3.6 Configuration Parameters

As per the AUTOSAR specification the driver has two types of configurations parameters - **Pre-Compile** parameters and **Post-Build** parameters.

Pre-Compile parameters are stored in the file "Pwm_Cfg.h" and "Pwm_Cfg.c". The Post-Build parameters are stored in the file "Pwm_PBcfg.c".

The files to be used for different configuration types are listed below:

1. **Variant PC:** Pwm_Cfg.h, Pwm_Cfg.c
2. **Variant PB**: Pwm_Cfg.h, Pwm_PBcfg.c
3. **Variant LT**: Not Applicable

A section for **Pwm_PBcfg.c** file is needed in linker file to place the post build configuration in desired location.

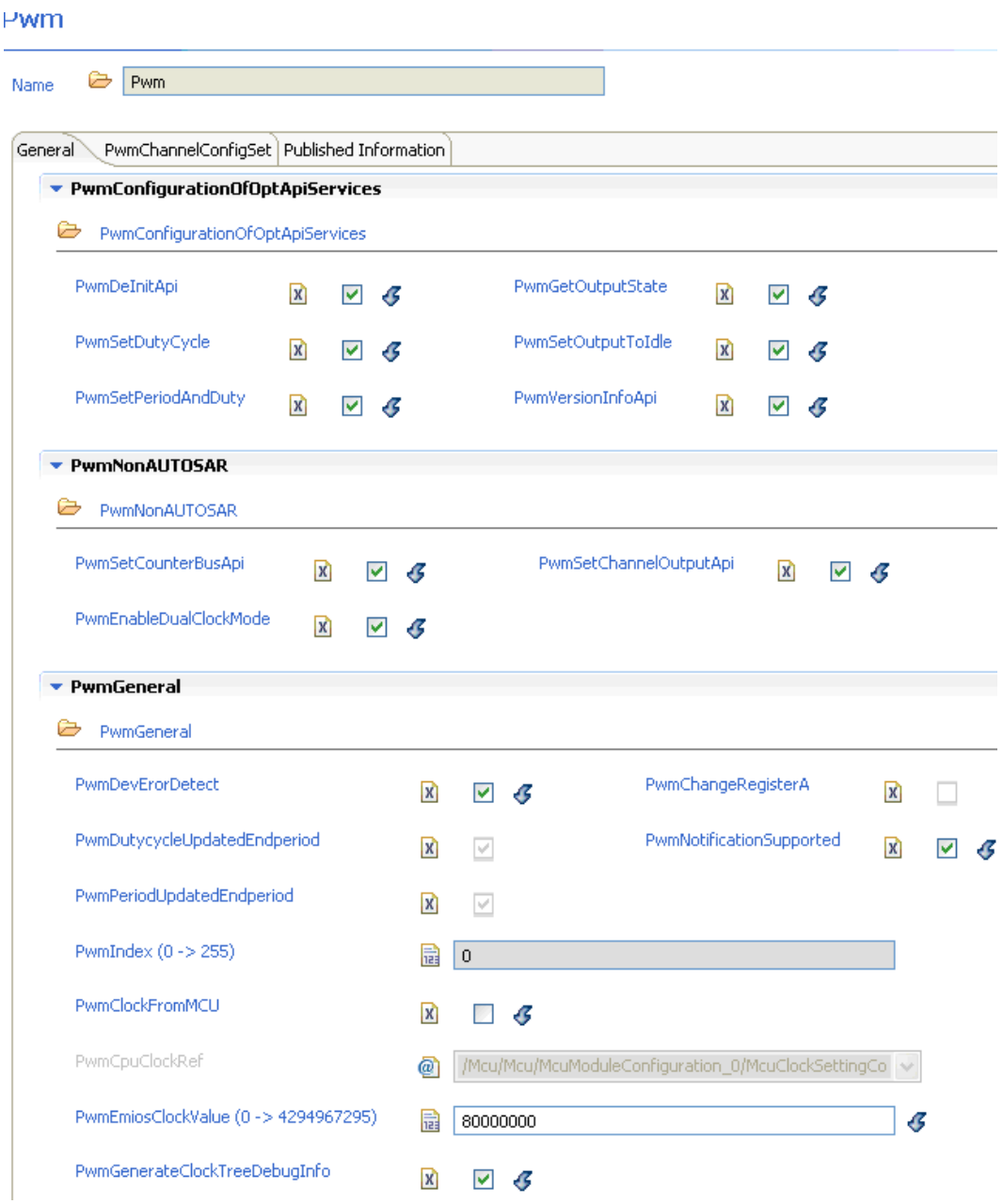
Please refer to section-8 of AUTOSAR_SWS_C_ImplementationRules file for complete details on configuration types.

3.6.1 Pre-Compile Parameters

Pre-Compile parameters, their possible values and meaning are described in the following text.

Pre-Compile parameters are implemented as preprocessor defines.

The configuration screen for PWM precompile parameters in Tresos® Studio configuration tool from tresos Tresos 2010a.sr4 20100415-release2010a-sr4 is given below:



NOTE

If pre-compile variant is preferred,

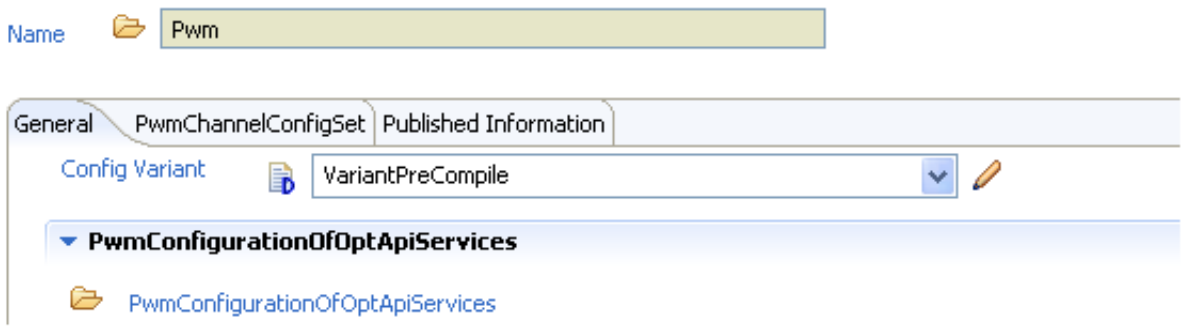
- IMPLEMENTATION_CONFIG_VARIANT in Tresos GUI should be selected as "VariantPreCompile" as shown below.
- 

Figure 3-1. Pwm Configuration Variant Selection.

- Pwm_Cfg.c should be compiled
- Pwm_PBcfg.c should not be compiled

3.6.1.1 PwmGeneral

This container contains various general parameters.

3.6.1.1.1 PwmChangeRegisterA

Table 3-14. PwmChangeRegisterA

Description	Preprocessor switch to indicate if register B or register A are updated when a change in dutycycle for a pwm channel configured in OPWMT mode is requested.Note: this switch is enabled only if OPWMT mode is supported on the current platform
Class	Implementation specific parameter
Range	True, False
Default	False
Source File	Pwm_Cfg.h
Source Representation	#define PWM_CHANGE_REGISTER_A_SWITCH <STD_OFF, STD_ON>

3.6.1.1.2 PwmClockFromMCU

Table 3-15. PwmClockFromMCU

Description	Switch to enable/disable the clock reference from MCU plugin. The default value is True and will enable the PWM plugin to compute the EMIOS clock using information from the MCU plugin. If set to False it will disable the MCU clock reference and enable the PwmEmiosClockValue field. PwmEmiosClockValue can be used to inform the Pwm plugin about the value of the EMIOS clock value. It will NOT set the EMIOS clock to the specified value. This feature should be used for testing/debug purpose only. For production the EMIOS clock must be referenced from the MCU plugin.
Class	Implementation specific parameter
Range	True, False
Default	True
Source File	N/A
Source Representation	N/A

3.6.1.1.3 PwmCpuClockRef

Table 3-16. PwmCpuClockRef

Description	Reference to MCU configuration used
Class	Implementation specific parameter
Range	Any valid reference to MCU configuration
Default	None
Source File	N/A
Source Representation	N/A

3.6.1.1.4 PwmDevErorDetect

Table 3-17. PwmDevErorDetect

Description	Preprocessor switch for enabling the development error detection
Class	Autosar Parameter
Range	True, False
Default	True
Source File	Pwm_Cfg.h
Source Representation	#define PWM_DEV_ERROR_DETECT <STD_OFF, STD_ON>

3.6.1.1.5 PwmDutycycleUpdatedEndperiod

Table 3-18. PwmDutycycleUpdatedEndperiod

Description	Switch for enabling the update of the duty cycle parameter at the end of the current period
Class	Autosar Parameter

Table continues on the next page...

Table 3-18. PwmDutycycleUpdatedEndperiod (continued)

Range	True, False
Default	True
Source File	Pwm_Cfg.h
Source Representation	#define PWM_DUTYCYCLE_UPDATED_ENDPERIOD <STD_OFF, STD_ON>
NOTE	*PwmDutycycleUpdatedEndPeriod parameter is not used in driver code. Retained in configuration for consistency with AutosarEcuParamdef.xml rev 17.*

3.6.1.1.6 PwmEmiosClockValue

Table 3-19. PwmEmiosClockValue

Description	Field used to inform the Pwm plugin about the value of the EMIOS clock value. It will NOT set the EMIOS clock to the specified value. It is enabled only when PwmClockFromMCU is set to false. Should be used only for testing/debug purpose only.
Class	Implementation specific parameter
Range	0 - 4294967295
Default	1000000
Source File	N/A
Source Representation	N/A

3.6.1.1.7 PwmGenerateClockTreeDebugInfo

Table 3-20. PwmGenerateClockTreeDebugInfo

Description	Switch to enable/disable the generation of debug information regarding the Pwm channel clock tree. Use this feature to debug problems regarding the period of pwm channels. Debug information will be available as comments in Pwm_Cfg.C and Pwm_PBCfg.c Example: /* --- Unified Channel clock debug information --- PwmPeriodDefaultUnits:Period_in_ticks PwmPeriodDefault:32767.0 this_unified_channel_period_in_ticks:32767 emios unified channel clock source: INTERNAL COUNTER this_unified_channel_prescaler_value:1 this_unified_channel_clock_frequency: 6.4E7 Hz this_unified_channel_clock_period:1.5625E-8 configured_channel_period: 5.11984375E-4 seconds configured_channel_freq:1953.1846064638205 Hz */
Class	Implementation specific parameter
Range	True, False
Default	False
Source File	N/A
Source Representation	N/A

3.6.1.1.8 PwmIndex

Table 3-21. PwmIndex

Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.
Class	Autosar Parameter
Range	0 - 255
Default	0
Source File	N/A
Source Representation	N/A
NOTE	*PwmIndex parameter is not implemented in driver code. Retained in configuration for consistency with AutosarEcuParamdef.xml rev 17.*

3.6.1.1.9 PwmNotificationSupported

Table 3-22. PwmNotificationSupported

Description	Preprocessor switch to indicate that the notifications are supported.
Class	Autosar Parameter
Range	True, False
Default	True
Source File	Pwm_Cfg.h
Source Representation	#define PWM_NOTIFICATION_SUPPORTED <STD_OFF, STD_ON>

3.6.1.1.10 PwmPeriodUpdatedEndperiod

Table 3-23. PwmPeriodUpdatedEndperiod

Description	Preprocessor switch for enabling the update of the period at the end of the current period
Class	Autosar Parameter
Range	True, False
Default	True
Source File	Pwm_Cfg.h
Source Representation	#define PWM_DUTY_PERIOD_UPDATED_ENDPERIOD <STD_OFF, STD_ON>
NOTE	*PwmPeriodUpdatedEndperiod parameter is not implemented in driver code. Retained in configuration for consistency with AutosarEcuParamdef.xml rev 17.*

3.6.1.2 IMPLEMENTATION_CONFIG_VARIANT

Table 3-24. IMPLEMENTATION_CONFIG_VARIANT

Description	Defines whether pre-compile version is used. Using this option with VariantPostBuild value, Tresos can generate many PwmChannelConfigSet variants.
Class	Implementation Specific Parameter
Range	VariantPreCompile, VariantPostBuild
Default	VariantPreCompile
Source File	Pwm_Cfg.h
Source Representation	#define PWM_PRECOMPILE_SUPPORT
Autosar 3.0 Requirement	NA
NOTE	*This parameter permit to generate many configurations if VariantPostBuild is selected.*

3.6.1.3 PwmConfigurationOfOptApiServices

The parameters in this container allow inclusion or exclusion of various AutoSAR APIs.

3.6.1.3.1 PwmDeInitApi

Table 3-25. PwmDeInitApi

Description	Adds / removes the service Pwm_DeInit() from the code.
Class	Autosar Parameter
Range	True, False
Default	True
Source File	Pwm_Cfg.h
Source Representation	#define PWM_DE_INIT_API <STD_OFF, STD_ON>

3.6.1.3.2 PwmGetOutputState

Table 3-26. PwmGetOutputState

Description	Adds / removes the service Pwm_GetOutputState() from the code
Class	Autosar Parameter
Range	True, False
Default	True
Source File	Pwm_Cfg.h
Source Representation	#define PWM_GET_OUTPUT_STATE_API <STD_OFF, STD_ON>

3.6.1.3.3 PwmSetDutyCycle

Table 3-27. PwmSetDutyCycle

Description	Adds / removes the service Pwm_SetDutyCycle() from the code.
Class	Autosar Parameter
Range	True, False
Default	True
Source File	Pwm_Cfg.h
Source Representation	#define PWM_SET_DUTY_CYCLE_API <STD_OFF, STD_ON>

3.6.1.3.4 PwmSetOutputToIdle

Table 3-28. PwmSetOutputToIdle

Description	Adds / removes the service Pwm_SetOutputToIdle() from the code.
Class	Autosar Parameter
Range	True, False
Default	True
Source File	Pwm_Cfg.h
Source Representation	#define PWM_SET_OUTPUT_TO_IDLE_API <STD_OFF, STD_ON>

3.6.1.3.5 PwmSetPeriodAndDuty

Table 3-29. PwmSetPeriodAndDuty

Description	Adds / removes the service Pwm_SetPeriodAndDuty() from the code.
Class	Autosar Parameter
Range	True, False
Default	True
Source File	Pwm_Cfg.h
Source Representation	#define PWM_SET_PERIOD_AND_DUTY_API <STD_OFF, STD_ON>

3.6.1.3.6 PwmVersionInfoApi

Table 3-30. PwmVersionInfoApi

Description	Preprocessor switch to indicate that the Pwm_ GetVersionInfo is supported
Class	Autosar Parameter
Range	True, False
Default	True
Source File	Pwm_Cfg.h
Source Representation	#define PWM_VERSION_INFO_API <STD_OFF, STD_ON>

3.6.1.4 PwmNonAUTOSAR

The parameters in this container allow inclusion or exclusion of various NonAutoSAR APIs.

3.6.1.4.1 PwmEnableDualClockMode

Table 3-31. PwmEnableDualClockMode

Description	Adds / removes the service Pwm_SetClockMode() from the code
Class	Implementation specific parameter
Range	True, False
Default	False
Source File	Pwm_Cfg.h
Source Representation	#define PWM_DUAL_CLOCK_MODE <STD_OFF, STD_ON>

3.6.1.4.2 PwmSetChannelOutputApi

Table 3-32. PwmSetChannelOutputApi

Description	Adds / removes the service Pwm_SetChannelOutput() from the code
Class	Implementation specific parameter
Range	True, False
Default	False
Source File	Pwm_Cfg.h
Source Representation	#define PWM_SETCHANNELOUTPUT_API <STD_OFF, STD_ON>

3.6.1.4.3 PwmSetCounterBusApi

Table 3-33. PwmSetCounterBusApi

Description	Preprocessor switch to indicate that the Pwm_SetCounterBus is supported
Class	Non-Autosar Parameter
Range	True, False
Default	True
Source File	Pwm_Cfg.h
Source Representation	#define PWM_SETCOUNTERBUS_API <STD_OFF, STD_ON>

3.6.2 Link-Time Parameters

Not Applicable.

3.6.3 Post-Build Parameters

Post-Build parameters, their possible values and their meaning are described in the following text. The Post-Build parameters are implemented as constant structures and arrays stored in flash memory of the MCU.

The configuration screen for PWM postbuild parameters in Tresos® Studio Configuration Tool from tresos Tresos 2010a.sr4 20100415-release2010a-sr4 is given below:

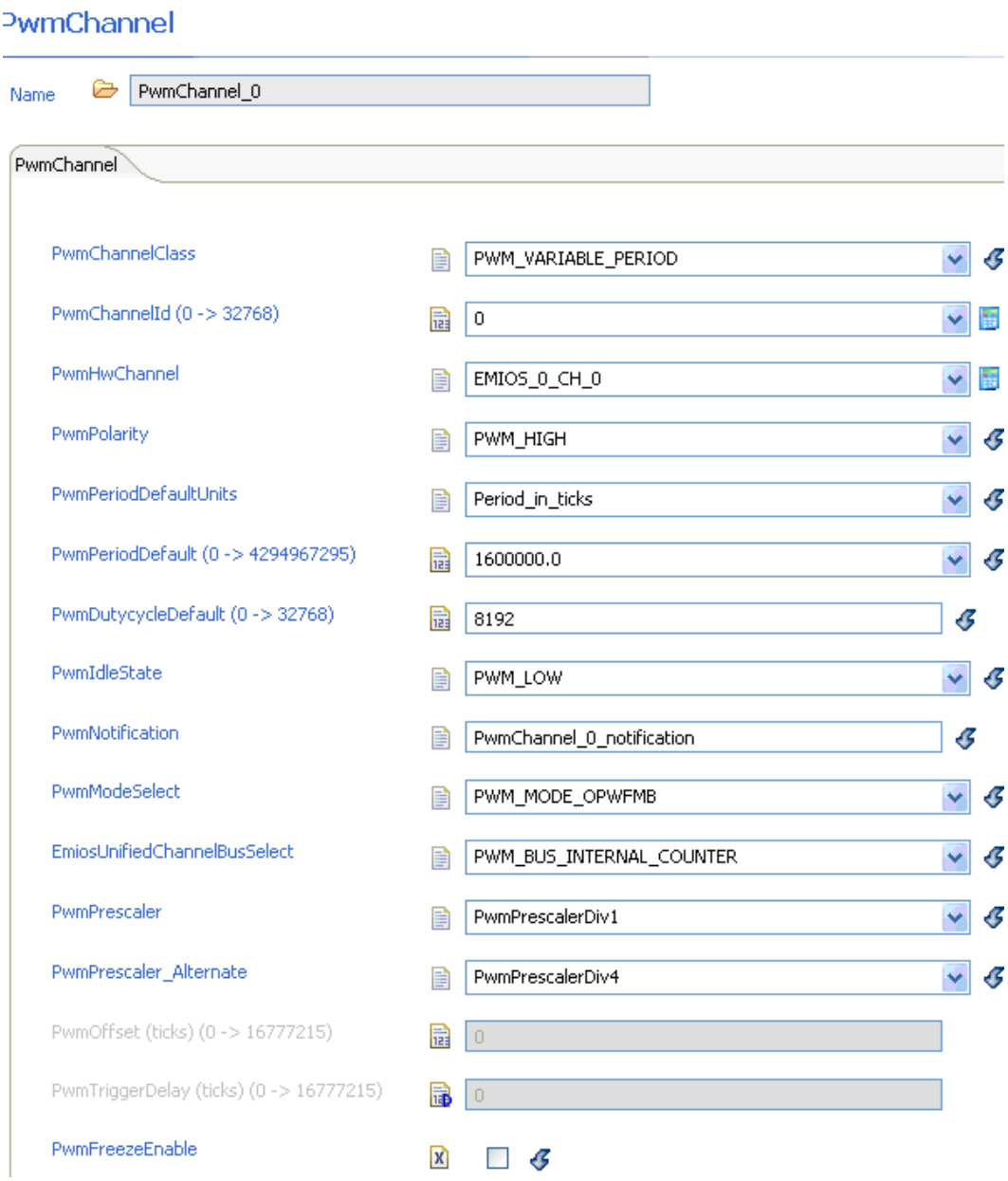


Figure 3-2. Pwm Channel Container.

3.6.3.1 Structures

3.6.3.1.1 Structure Pwm_ChannelConfigType

pwm channel high level configuration structure

Declaration

```
typedef struct
                    {
                      const uint8 Config_index,
```

```
                        const Pwm_IpType IpType,
                        const Pwm_NotifyType Pwm_Channel_Notification,
                        const Pwm_ChannelClassType Pwm_ChannelClassValue,
                        const uint16 Pwm_DefaultDutyCycleValue,
                        const Pwm_PeriodType Pwm_DefaultPeriodValue,
                        const Pwm_OutputStateType Pwm_IdleState,
                        const Pwm_OutputStateType Pwm_Polarity
                } Pwm_ChannelConfigType;
```

Table 3-34. Structure Pwm_ChannelConfigType member description

Member	Description
Config_index	index in the IP specific parameter table
IpType	the IP used to implement this specific Pwm channel
Pwm_CenterAligned	pwm generation mode: edge aligned or center aligned
Pwm_Channel_Notification	Pointer to notification function.
Pwm_ChannelClassValue	channel type: Variable/Fixed period
Pwm_DefaultDutyCycleValue	Default value for duty cycle: [0-0x8000] (0-100%).
Pwm_DefaultPeriodValue	Default value for period.
Pwm_IdleState	Pwm signal idle state: High or low.
Pwm_Polarity	Pwm signal polarity: High or low.

3.6.3.1.2 Structure Pwm_ConfigType

pwm high level configuration structure

Declaration

```
typedef struct
                {
                  const Pwm_ChannelType ChannelCount,
                        const Pwm_ChannelConfigType *const ChannelsPtr,
                        const Pwm_IpConfigType IpConfig
                } Pwm_ConfigType;
```

Table 3-35. Structure Pwm_ConfigType member description

Member	Description
ChannelCount	number of configured channels
ChannelsPtr	pointer to the configured channels
IpConfig	Combined IP specific configuration structure.

3.6.3.1.3 Structure Pwm_EMIOS_ChannelConfigType

EMIOS IP specific channel configuration structure for the PWM functionality.

Declaration

```
typedef struct
                {
```

```
  const Pwm_HwChannelType Pwm_HwChannelID,
          const Pwm_PeriodType Pwm_Offset,
          const Pwm_EmiosCtrlParamType Pwm_ParamValue,
          const uint32 Pwm_Prescaler_Alternate
} Pwm_EMIOS_ChannelConfigType;
```

Table 3-36. Structure Pwm_EMIOS_ChannelConfigType member description

Member	Description
Pwm_HwChannelID	eMIOS HW channel id
Pwm_Offset	leading edge of the output pulse
Pwm_ParamValue	EMIOSC[n] control.
Pwm_Prescaler_Alternate	prescaler value

3.6.3.1.4 Structure Pwm_LLD_ChannelConfigType

Low level channel configuration structure.

Pwm channel eMIOS specific configuration structure

Declaration

```
typedef struct
                    {
                      const Pwm_EMIOS_ChannelConfigType EmiosCfg
                    } Pwm_LLD_ChannelConfigType;
```

Table 3-37. Structure Pwm_LLD_ChannelConfigType member description

Member	Description
EmiosCfg	Pwm channel eMIOS specific configuration structure.

3.6.3.2 Plug-in parameters

3.6.3.2.1 EmiosUnifiedChannelBusSelect

Table 3-38. EmiosUnifiedChannelBusSelect

Description	Selects the counter used with the unified channel InternalCounter - mandatory for OPWFMB BusA, BusDiverse - mandatory for OPWMB
Class	Implementation Specific Parameter
Range	PWM_BUS_A,PWM_BUS_DIVERSE,PWM_BUS_INTERNAL_COUNTER
Default	PWM_BUS_INTERNAL_COUNTER
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-38. EmiosUnifiedChannelBusSelect (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (**PWM_BUS_INTERNAL_COUNTER** | PWM_MODE_OPWFMB |`
	`  PWM_PRES_1), /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  0U, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.2 PwmChannelClass

Table 3-39. PwmChannelClass

Description	Class of PWM Channel.
Class	Autosar Parameter
Range	PWM_FIXED_PERIOD, PWM_VARIABLE_PERIOD, PWM_FIXED_PERIOD_SHIFTED
Default	PWM_VARIABLE_PERIOD
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-39. PwmChannelClass (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  **PWM_VARIABLE_PERIOD**, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`  PWM_PRES_1), /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  0U, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.3 PwmChannelId

Table 3-40. PwmChannelId

Description	Channel Id of the PWM channel. This value will be assigned to the symbolic name derived of the PwmChannel container short name.
Class	Autosar Parameter
Range	0 - 32768
Default	0
Source File	Pwm_Cfg.h
Source Representation	#define PwmChannel_0 0

3.6.3.2.4 PwmDutyCycleDefault

Table 3-41. PwmDutyCycleDefault

Description	Value of duty cycle used for initialization 0x0 represents 0% 0x8000(32768) represents 100%
Class	Implementation Specific Parameter
Range	0 - 0x8000 (32768)
Default	16384 (50%)
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-41. PwmDutyCycleDefault (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  **0x2000**, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`  PWM_PRES_1), /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  0U, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.5 PwmFreezeEnable

Table 3-42. PwmFreezeEnable

Description	Freeze Enable Bit. The FREN bit, if set and validated by FRZ bit in EMIOS_MCR register, freezes all unified channel registers when in debug mode to allow the MCU to perform debug functions.
Class	Implementation Specific Parameter
Range	True, False
Default	False
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-42. PwmFreezeEnable (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`  **PWM_FREEZE_ENABLE** | PWM_PRES_1),`
	`  /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  0U, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.6 PwmHwChannel

Table 3-43. PwmHwChannel

Description	Selects the physical PWM Channel.
Class	Implementation Specific Parameter
Range	EMIOS_A_0 to EMIOS_A_24 corresponds to EMIOS_A
Default	EMIOS_A_0
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-43. PwmHwChannel (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`    {`
	`        PWM_VARIABLE_PERIOD, /* channel type */`
	`        PWM_HIGH, /* polarity */`
	`        1600000, /* default period */`
	`        0x2000, /* default duty */`
	`        PWM_LOW, /* idle state */`
	`        PwmChannel_0_notification, /* notification f() */`
	`        PWM_EMIOS_CHANNEL, /* IP used */`
	`        {`
	`            EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`            (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`             PWM_FREEZE_ENABLE | PWM_PRES_1),`
	`             /* unified channel specific`
	`             parameters */`
	`            PWM_PRES_4`
	`            0U, /* offset */`
	`            0U, /* trigger delay */`
	`        }`
	
	`    }`
	`}`

3.6.3.2.7 PwmIdleState

Table 3-44. PwmIdleState

Description	Value of the PWM signal at idle state.
Class	Autosar Parameter
Range	PwmIdleStateLow, PwmIdleStateHigh
Default	PwmIdleStateLow
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-44. PwmIdleState (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  **PWM_LOW**, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`  PWM_FREEZE_ENABLE | PWM_PRES_1),`
	`  /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  0U, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.8 PwmModeSelect

Table 3-45. PwmModeSelect

Description	Select the PwmMode for the channel from drop downlist. OPWFMB, OPWMB Variable period channels are implemented using only the OPWFMB mode. The plug-in will check this and raise an error if this condition is not satisfied.
Class	Implementation Specific Parameter
Range	PWM_MODE_OPWFMB, PWM_MODE_OPWMB
Default	PWM_MODE_OPWFMB
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-45. PwmModeSelect (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | **PWM_MODE_OPWFMB** |`
	`  PWM_FREEZE_ENABLE | PWM_PRES_1),`
	`  /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  0U, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.9 PwmNotification

Table 3-46. PwmNotification

Description	User notification callback function.
Class	Autosar Parameter
Range	N/A
Default	NULL_PTR
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-46. PwmNotification (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  **PwmChannel_0_notification**, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`  PWM_FREEZE_ENABLE | PWM_PRES_1),`
	`  /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  0U, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.10 PwmOffset

Table 3-47. PwmOffset

Description	PwmOffset (in ticks) of the PWM output. This parameter is used only in OPWMB modes. Counter will always start from 1, So the offset value is incremented by 1 in the code, then the value is updated into register. Please note the PwmOffset parameter must be less than the period of the MCB channel used as reference. When Offset !=0 100% dutycycle can't be reached due to the fact that the PWM signal is shifted offset ticks. The max dutycycle achievable when PwmOffset != 0 can be calculated using the following formula. Dutycycle less then ((Period - PwmOffset) / Period) * 0x8000. Period is the period in ticks of the MCB channel used as reference. If this condition is not satisfied then Det_ReportError(PWM_MODULE_ID, 0, PWM_SETDUTYCYCLE_ID, PWM_E_OPWMB_CHANNEL_OFFSET_DUTYCYCLE_RANGE) will be generated.
Class	Implementation Specific Parameter
Range	0 -> 16777215
Default	0
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-47. PwmOffset (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`  PWM_FREEZE_ENABLE | PWM_PRES_1),`
	`  /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  **0U**, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.11 PwmPeriodDefault

Table 3-48. PwmPeriodDefault

Description	Value of period used for Initialization. The Unit of Period is defined by PwmPeriodDefaultUnits. Note: Counter will always start from 0x1, so the period value is incremented by 1 in the code, then the value is updated into period register. Valid values [1, 0xFFFFFE = 16777214]. In case of PWM_MODE_OPWMB or this value is ignored and the period for this channel is equal with the period of the selected unified channel running in MCB mode. Please note that the period of the MCB channel must be less than 0xFFFF or else reaching 0% will not be possible.
Class	Autosar Parameter
Range	1 - 4294967295
Default	1600000
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-48. PwmPeriodDefault (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`  PWM_FREEZE_ENABLE | PWM_PRES_1),`
	`  /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  0U, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.12 PwmPeriodDefaultUnits

Table 3-49. PwmPeriodDefaultUnits

Description	Defines the measurement units for PwmPeriodDefault.
Class	Implementation specific parameter
Range	Period_in_seconds / Period_in_ticks / Frequency_in_Hz
Default	Period_in_seconds
Source File	This parameter is used only by the generation code in the transformation from seconds / frequency (if selected) to unified channel ticks

Table continues on the next page...

Table 3-49. PwmPeriodDefaultUnits (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`  PWM_FREEZE_ENABLE | PWM_PRES_1),`
	`  /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  0U, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.13 PwmPolarity

Table 3-50. PwmPolarity

Description	Defines the PWM channel signal polarity.
Class	Autosar Parameter
Range	PwmPolarityLow, PwmPolarityHigh
Default	PwmPolarityHigh
Source File	This parameter is used only by the generation code in the transformation from seconds / frequency (if selected) to unified channel ticks

Table continues on the next page...

Table 3-50. PwmPolarity (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  **PWM_HIGH**, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`  PWM_FREEZE_ENABLE | PWM_PRES_1),`
	`  /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  0U, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.14 PwmPrescaler

Table 3-51. PwmPrescaler

Description	Pwm Channel Prescaler. Clock prescaler to decrease Pwm period. Affects the PWM period only in OPWFMB mode when the internal counter must be selected.
Class	Implementation Specific Parameter
Range	PwmPrescalerDiv1,PwmPrescalerDiv2,PwmPrescalerDiv3,PwmPrescalerDiv4
Default	PwmPrescalerDiv1
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-51. PwmPrescaler (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`  PWM_FREEZE_ENABLE | **PWM_PRES_1**),`
	`  /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  0U, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.15 PwmPrescaler_Alternate

Table 3-52. PwmPrescaler_Alternate

Description	Pwm Channel Prescaler. This Parameter sets the prescaler value to maintain the same period for a different frequency. Affects the PWM period only in OPWFMB mode when the internal counter must be selected. Note the EMIOSMCR register has to configured as well: see McuModuleConfiguration/.../McuClockSettingConfig/.../EMIOSSettings
Class	Implementation Specific Parameter
Range	PwmPrescalerDiv1 To PwmPrescalerDiv4
Default	PwmPrescalerDiv1
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-52. PwmPrescaler_Alternate (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`  PWM_FREEZE_ENABLE | PWM_PRES_1),`
	`  /* unified channel specific`
	`  parameters */`
	`  **PWM_PRES_4**,`
	`  0U, /* offset */`
	`  0U, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

3.6.3.2.16 PwmTriggerDelay

Table 3-53. PwmTriggerDelay

Description	PwmTriggerDelay specifies the position of the trigger. Please note that if the configured value is outside of the configured period (reference channel period), the trigger will never be generated.
Class	Implementation Specific Parameter
Range	[1-max period of reference channel]
Default	0
Source File	Pwm_Cfg.c, Pwm_PBcfg.c

Table continues on the next page...

Table 3-53. PwmTriggerDelay (continued)

Source Representation	The highlighted portion of the following structure:
	
	`CONST(Pwm_ChannelConfigType, PWM_CONST)`
	`Pwm_InitChannel_0[PWM_CONF_CHANNELS_PB_0] =`
	`{`
	
	`  {`
	`  PWM_VARIABLE_PERIOD, /* channel type */`
	`  PWM_HIGH, /* polarity */`
	`  1600000, /* default period */`
	`  0x2000, /* default duty */`
	`  PWM_LOW, /* idle state */`
	`  PwmChannel_0_notification, /* notification f() */`
	`  PWM_EMIOS_CHANNEL, /* IP used */`
	`  {`
	`  EMIOS_0_CH_0, /* EMIOS HW unified channel ID */`
	`  (PWM_BUS_INTERNAL_COUNTER | PWM_MODE_OPWFMB |`
	`  PWM_FREEZE_ENABLE | PWM_PRES_1),`
	`  /* unified channel specific`
	`  parameters */`
	`  PWM_PRES_4`
	`  0U, /* offset */`
	`  **123U**, /* trigger delay */`
	`  }`
	
	`  }`
	`}`

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductors products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor 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. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claims alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-complaint and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale's Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© 2011 Freescale Semiconductor, Inc.

Document Number: UM14PWMASR3.0R2.0.0
Rev. 1.1