Integration Manual

for MPC574XG PWM Driver

Document Number: IM35PWMASR4.2 Rev0002 R1.0.0
Rev. 5.0.0

Contents

Section number **Title** **Page**

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0.0	22/08/2014	Do The Viet	Pwm Integration Manual for MPC574XG - RTM0.9.0 Release
2.0.0	24/04/2015	Do The Viet	Pwm Integration Manual for MPC574XG - RTM1.0.0 Release
3.0.0	10/07/2015	Do The Viet	Pwm Integration Manual for MPC574XG - RTM1.0.1 Release
4.0.0	12/08/2016	Lam Nguyen	Pwm Integration Manual for MPC574XG - RTM1.0.2 Release
5.0.0	17/02/2017	An Do Xuan	Pwm Integration Manual for MPC574XG AUTOSAR4.2.2 - RTM1.0.0 Release

Chapter 2
Introduction

This integration manual describes the integration requirements for PWM Driver for MPC574XG microcontrollers.

2.1 Supported Derivatives

The software described in this document is intented to be used with the following microcontroller devices of NXP Semiconductor .

Table 2-1. MPC574XG Derivatives

NXP Semiconductor	MPC5748G_LQFP176, MPC5748G_MAPBGA256, MPC5748G_MAPBGA324, MPC5747G_LQFP176, MPC5747G_MAPBGA256, MPC5747G_MAPBGA324, MPC5746G_LQFP176, MPC5746G_MAPBGA256, MPC5746G_MAPBGA324, MPC5748C_LQFP176, MPC5748C_MAPBGA256, MPC5748C_MAPBGA324, MPC5747C_LQFP176, MPC5747C_MAPBGA256, MPC5747C_MAPBGA324, MPC5746C_LQFP176, MPC5746C_MAPBGA256, MPC5746C_MAPBGA324, MPC5746C_MAPBGA100, MPC5745C_LQFP176, MPC5745C_MAPBGA256, MPC5745C_MAPBGA100, MPC5744C_LQFP176, MPC5744C_MAPBGA256, MPC5744C_MAPBGA100, MPC5746B_LQFP176, MPC5746B_MAPBGA256, MPC5746B_MAPBGA100, MPC5744B_LQFP176, MPC5744B_MAPBGA256,

Table 2-1. MPC574XG Derivatives

	MPC5744B_MAPBGA100, MPC5745B_LQFP176, MPC5745B_MAPBGA256, MPC5745B_MAPBGA100

All of the above microcontroller devices are collectively named as MPC574XG .

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
ASM	Assembler
BSMI	Basic Software Make file Interface
BSW	Basic Software
CAN	Controller Area Network
DEM	Diagnostic Event Manager
DET	Development Error Tracer
C/CPP	C and C++ Source Code
ECU	Electronic Control Unit
ISR	Interrupt Service Routine
MCU	Microcontroller Unit
PWM	Pulse Width Modulation
RAM	Random Access Memory
ROM	Read-only Memory
OS	Operating System
PB Variant	Post Build Variant
PC Variant	Pre Compile Variant
VLE	Variable Length Encoding
N/A	Not Applicable
MCU	Micro Controller Unit
EMIOS	Configurable Enhanced Modular IO Subsystem

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	AUTOSAR 4.2 Rev0002PWM Driver Software Specification Document.	v2.5.0
2	MPC5748G Reference Manual	Rev. 5, 12/2016
3	MPC5746C Reference Manual	Rev. 4, 12/2016
4	MPC5748G_1N81M_Rev.2 (official document) (1N81M)	Jun-16
5	MPC5748G_1N81M_0N78S_Comparison_Summary_v2_0 (internal document) (1N81M, 0N78S)	31.10.2016

Table continues on the next page...

Table 2-3. Reference List (continued)

#	Title	Version
6	MPC5746C_1N06M_Rev.4 (official document) (1N06M)	Jul-16
7	MPC5746C_cut1.1_cut2.0_cut2.1_comparison_v0 (internal document) (1N06M, 0N84S, 1N84S)	14-Sep-16
8	C3M_cut2.1_new_errata_20170113 (internal document) (1N84S)	13-Jan-17

Chapter 3
Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar PWM driver for NXP SemiconductorMPC574XG . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The PWM driver files are compiled using

- Windriver DIAB DIAB_5_9_6_2
- Green Hills Multi 7.1.4 / Compiler 2015.1.6

The compiler, linker flags used for building the driver are explained below:

Note

The TS_T2D35M10I0R0 plugin name is composed as follow:

TS_T = Target_Id

D = Derivative_Id

M = SW_Version_Major

I = SW_Version_Minor

R = Revision

(i.e. Target_Id = 2 identifies PowerPC architecture and
Derivative_Id = 35 identifies the MPC574XG)

3.1.1 DIAB Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-tPPCE200Z4VEN:simple	Sets target processor to PPCE200Z4, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-tPPCE200Z7VEN:simple	Sets target processor to PPCE200Z7, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-Xdialect-ansi	Follow the ANSI C standard with some additions
-XO	Enables extra optimizations to produce highly optimized code
-g3	Generate symbolic debugger information and do all optimizations.
-Xsize-opt	Optimize for size rather than speed when there is a choice
-Xsmall-data=0	Set Size Limit for 'small data' Variables to zero.
-Xaddr-sconst=0x11	Specify addressing for constant static and global variables with size less than or equal to -Xsmall-const to far-absolute.
-Xaddr-sdata=0x11	Specify addressing for non-constant static and global variables with size less than or equal to -Xsmall-data in size to far-absolute.
-Xno-common	Disable use of the 'COMMON' feature so that the compiler or assembler will allocate each uninitialized public variable in the .bss section for the module defining it, and the linker will require exactly one definition of each public variable
-Xnested-interrupts	Allow nested interrupts
-Xdebug-dwarf2	Generate symbolic debug information in dwarf2 format
-Xdebug-local-all	Force generation of type information for all local variables
-Xdebug-local-cie	Create common information entry per module
-Xdebug-struct-all	Force generation of type information for all typedefs, struct, union and class types
-Xforce-declarations	Generates warnings if a function is used without a previous declaration
-ee1481	Generate an error when the function was used before it has been declared
-Xmacro-undefined-warn	Generates a warning when an undefined macro name occurs in a #if preprocessor directive
-Xlink-time-lint	Enable the checking of object and function declarations across compilation units, as well as the consistency of compiler options used to compile source files
-W:as:,-l	Pass the option '-l' (lower case letter L) to the assembler to get an assembler listing file
-Wa,-Xisa-vle	Instruct the assembler to expect and assemble VLE (Variable Length Encoding) instructions rather than BookE instructions.
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DUSE_SW_VECTOR_MODE	-D defines a preprocessor symbol and optionally can set it to a value. USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode.
-DDIAB	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the DIAB preprocessor symbol.

Table continues on the next page...

Table 3-1. Compiler Options (continued)

Option	Description
-DDISABLE_MCAL_INTERMODULE_ASR_CHECK	-D defines a preprocessor symbol to disable the inter-module version check for AR_RELEASE versions. DISABLE_MCAL_INTERMODULE_ASR_CHECK: By default in the package, drivers are compiled to perform the inter-module version check as per Autosar BSW004. When the inter-module version check needs to be disabled then the DISABLE_MCAL_INTERMODULE_ASR_CHECK global define must be added to the list of compiler options.
-c	Stop after assembly, produce object file.

Table 3-2. Assembler Options

Option	Description
-tPPCE200Z4VEN:simple	Sets target processor to PPCE200Z4, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-tPPCE200Z7VEN:simple	Sets target processor to PPCE200Z7, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-g	Dump the symbols in the global symbol table in each archive file.
-Xisa-vle	Expect and assemble VLE (Variable Length Encoding) instructions rather than Book E instructions. The default code section is named .text_vle instead of .text, and the default code section fill "character" is set to 0x44444444 instead of 0. The .text_vle code section will have ELF section header flags marking it as VLE code, not Book E code.
-Xasm-debug-on	Generate debug line and file information
-Xdebug-dwarf2	Generate symbolic debug information in dwarf2 format

Table 3-3. Linker Options

Option	Description
-tPPCE200Z4VEN:simple	Sets target processor to PPCE200Z4, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-tPPCE200Z7VEN:simple	Sets target processor to PPCE200Z7, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-Xelf	Generates ELF object format for output file
-m6	Generates a detailed link map and cross reference table
-lc	Specifies to linker to search for libc.a
-Xlibc-old	Enables usage of legacy (pre-release 5.6) libraries
-Xlink-time-lint	Enable the checking of object and function declarations across compilation units, as well as the consistency of compiler options used to compile source files

3.1.2 GHS Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
-cpu=ppc5775kz7260	Selects target processor: ppc5775kz7260
-cpu=ppc5775kz4201	Selects target processor: ppc5775kz4201
-ansi	Specifies ANSI C with extensions. This mode extends the ANSI X3.159-1989 standard with certain useful and compatible constructs.
-noSPE	Disables the use of SPE and vector floating point instructions by the compiler.
-Ospace	Optimize for size.
-sda=0	Enables the Small Data Area optimization with a threshold of 0.
-vle	Enables VLE code generation
-dual_debug	Enables the generation of DWARF, COFF, or BSD debugging information in the object file
-G	Generates source level debugging information and allows procedure call from debugger's command line.
--no_exceptions	Disables support for exception handling
-Wundef	Generates warnings for undefined symbols in preprocessor expressions
-Wimplicit-int	Issues a warning if the return type of a function is not declared before it is called
-Wshadow	Issues a warning if the declaration of a local variable shadows the declaration of a variable of the same name declared at the global scope, or at an outer scope
-Wtrigraphs	Issues a warning for any use of trigraphs
--prototype_errors	Generates errors when functions referenced or called have no prototype
--incorrect_pragma_warnings	Valid #pragma directives with wrong syntax are treated as warnings
-noslashcomment	C++ like comments will generate a compilation error
-preprocess_assembly_files	Preprocesses assembly files
-nostartfile	Do not use Start files
--short_enum	Store enumerations in the smallest possible type
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DUSE_SW_VECTOR_MODE	-D defines a preprocessor symbol and optionally can set it to a value. USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode.
-DDISABLE_MCAL_INTERMODULE_ASR_CHECK	-D defines a preprocessor symbol to disable the inter-module version check for AR_RELEASE versions. DISABLE_MCAL_INTERMODULE_ASR_CHECK: By default in the package, drivers are compiled to perform the inter-module version check as per Autosar BSW004. When the inter-module version check needs to be disabled then the DISABLE_MCAL_INTERMODULE_ASR_CHECK global define must be added to the list of compiler options.
-DGHS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GHS preprocessor symbol.
-c	Produces an object file (called input-file.o) for each source file.

Table 3-5. Assembler Options

Option	Description
-cpu=ppc5775kz7260	Selects target processor: ppc5775kz7260
-cpu=ppc5775kz4201	Selects target processor: ppc5775kz4201

Table 3-6. Linker Options

Option	Description
-cpu=ppc5775kz7260	Selects target processor: ppc5775kz7260
-cpu=ppc5775kz4201	Selects target processor: ppc5775kz4201
-nostartfiles	Do not use Start files.
-vle	Enables VLE code generation

3.2 Files required for Compilation

This section describes the include files required to compile, assemble (if assembler code) and link the PWM driver for MPC574XG microcontrollers.

To avoid integration of incompatible files, all the include files from other modules shall have the same AR_MAJOR_VERSION and AR_MINOR_VERSION, i.e. only files with the same AUTOSAR major and minor versions can be compiled.

PWM Files

- ..\ PWM_TS_T2D35M10I0R0 \include\Pwm.h
- ..\ PWM_TS_T2D35M10I0R0 \include\Pwm_EnvCfg.h
- ..\ PWM_TS_T2D35M10I0R0 \include\Pwm_Types.h
- ..\ PWM_TS_T2D35M10I0R0 \include\Pwm_eMios.h
- ..\ PWM_TS_T2D35M10I0R0 \include\Pwm_eMios_Types.h
- ..\ PWM_TS_T2D35M10I0R0 \include\Pwm_eMios_Common_Types.h
- ..\ PWM_TS_T2D35M10I0R0 \include\Pwm_Ipw.h
- ..\ PWM_TS_T2D35M10I0R0 \include\Pwm_Ipw_Notif.h
- ..\ PWM_TS_T2D35M10I0R0 \include\Pwm_Ipw_Types.h
- ..\ PWM_TS_T2D35M10I0R0 \include\Pwm_Notif.h
- ..\ PWM_TS_T2D35M10I0R0 \include\Reg_eSys_eMios.h
- ..\ PWM_TS_T2D35M10I0R0 \src\Pwm.c
- ..\ PWM_TS_T2D35M10I0R0 \src\Pwm_eMios.c
- ..\ PWM_TS_T2D35M10I0R0 \src\Pwm_Ipw.c

PWM Generated Files

- Pwm_PBcfg.c - For driver compilation, this file should be generated by the user using a configuration tool
- Pwm_Cfg.h - For driver compilation, this file should be generated by the user using a configuration tool

Files from Base common folder

- ..\Base_TS_T2D35M10I0R0 \include\Compiler.h
- ..\Base_TS_T2D35M10I0R0 \include\Compiler_Cfg.h
- ..\Base_TS_T2D35M10I0R0 \include\ComStack_Types.h
- ..\Base_TS_T2D35M10I0R0 \include\MemMap.h
- ..\Base_TS_T2D35M10I0R0 \include\Mcal.h
- ..\Base_TS_T2D35M10I0R0 \include\Platform_Types.h
- ..\Base_TS_T2D35M10I0R0 \include\Std_Types.h
- ..\Base_TS_T2D35M10I0R0 \include\Reg_eSys.h
- ..\Base_TS_T2D35M10I0R0 \include\Soc_Ips.h
- ..\Base_TS_T2D35M10I0R0 \include\SilRegMacros.h

Files from Rte folder:

- ..\Rte_TS_T2D35M10I0R0 \include\SchM_Pwm.h

3.3 Setting up the Plug-ins

All the Autosar MCAL drivers for MPC574XG were designed to be configured using Tresos Studio configuration and code generation tool from EB tresos Studio 21.0.0 b160607-0933.

Location of various files inside the plugin folder is explained below.

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\PWM_TS_T2D35M10I0R0\config\Pwm.xdm
  - ..\Mcu_TS_T2D35M10I0R0\config\Mcu.xdm
  - ..\EcuC_TS_T2D35M10I0R0\config\EcuC.xdm
  - ..\Resource_TS_T2D35M10I0R0\config\Resource.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\PWM_TS_T2D35M10I0R0\autosar\Pwm_<subderivative_name>.epd
  - ..\Mcu_TS_T2D35M10I0R0\autosar\Mcu.epd
  - ..\Mcl_TS_T2D35M10I0R0\autosar\Mcl.epd
  - ..\Resource_TS_T2D35M10I0R0\autosar\Resource_<subderivative_name>.epd
- Code Generation Templates for Pre-Compile time configuration parameters:

    - ..\PWM_TS_T2D35M10I0R0\output\src\Pwm_PBCfg.c
    - ..\PWM_TS_T2D35M10I0R0\output\include\Pwm_Cfg.h
    - ..\Mcu_TS_T2D35M10I0R0\output\src\Mcu_PBCfg.c
    - ..\Mcu_TS_T2D35M10I0R0\output\include\Mcu_Cfg.h

- Code Generation Templates for Post-Build time configuration parameters:
    - ..\PWM_TS_T2D35M10I0R0\output\src\Pwm_PBCfg.c
    - ..\PWM_TS_T2D35M10I0R0\output\include\Pwm_Cfg.h
    - ..\Mcu_TS_T2D35M10I0R0\output\src\Mcu_PBCfg.c
    - ..\Mcu_TS_T2D35M10I0R0\output\include\Mcu_Cfg.h

Steps to generate the configuration:

1. Copy the module folders PWM_TS_T2D35M10I0R0 , Resource_TS_T2D35M10I0R0 , Mcu_TS_T2D35M10I0R0 , Mcl_TS_T2D35M10I0R0into the Tresos plugins folder.
2. Set the desired Tresos Output location folder for the generated sources and header files.
3. Use the EB tresos Studio GUI to modify ECU configuration parameters values.
4. Generate the configuration files.

Dependencies

- **RESOURCE** is required to select processor derivative. Current Can driver has support for the following derivatives, everyone having attached a Resource file: MPC5748G_LQFP176, MPC5748G_MAPBGA256, MPC5748G_MAPBGA324, MPC5747G_LQFP176, MPC5747G_MAPBGA256, MPC5747G_MAPBGA324, MPC5746G_LQFP176, MPC5746G_MAPBGA256, MPC5746G_MAPBGA324, MPC5748C_LQFP176, MPC5748C_MAPBGA256, MPC5748C_MAPBGA324, MPC5747C_LQFP176, MPC5747C_MAPBGA256, MPC5747C_MAPBGA324, MPC5746C_LQFP176, MPC5746C_MAPBGA256, MPC5746C_MAPBGA324, MPC5746C_MAPBGA100, MPC5745C_LQFP176, MPC5745C_MAPBGA256, MPC5745C_MAPBGA100, MPC5744C_LQFP176, MPC5744C_MAPBGA256, MPC5744C_MAPBGA100, MPC5746B_LQFP176, MPC5746B_MAPBGA256, MPC5746B_MAPBGA100, MPC5744B_LQFP176, MPC5744B_MAPBGA256, MPC5744B_MAPBGA100, MPC5745B_LQFP176, MPC5745B_MAPBGA256, MPC5745B_MAPBGA100 .
- **MCU** is required for selecting the clock source and global prescaler used by the Pwm driver.
- **DET** is required for signaling the development error detection (parameters out of range, null pointers, etc).
- **RTE** is required for critical sections

Chapter 4
Function calls to module

4.1 Function Calls during Start-up

PWM shall be initialized during STARTUP phase of EcuM initialization. The API to be called for this is Pwm_Init(). The MCU module should be initialized before the PWM is initialized.

4.2 Function Calls during Shutdown

During shutdown phase, Pwm_DeInit() function can be called. Calling this function depends on the initialization-deinitialization strategy deployed by user.

4.3 Function Calls during Wake-up

During Wake-up phase, Pwm_Init() function may be called but only if during a previous phase Pwm_DeInit() was called. Calling this function depends on the initializationdeinitialization strategy deployed by user.

Chapter 5
Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

There are some hardware and software resources are shared for some functions in Pwm_eMios.c. They are protected under Exclusive Areas.

In function Pwm_eMios_SetDutyCycle: **PWM_EXCLUSIVE_AREA_00**

In function Pwm_eMios_SetPeriodAndDuty: **PWM_EXCLUSIVE_AREA_01**

In function Pwm_eMios_SetOutputToIdle: **PWM_EXCLUSIVE_AREA_02**

In function Pwm_eMios_EnableNotification : **PWM_EXCLUSIVE_AREA_03**

In function Pwm_eMios_DisableNotification : **PWM_EXCLUSIVE_AREA_04**

In function Pwm_eMios_SetCounterBus : **PWM_EXCLUSIVE_AREA_05**

In function Pwm_eMios_SetChannelOutput : **PWM_EXCLUSIVE_AREA_06**

In function Pwm_eMios_SetClockMode: **PWM_EXCLUSIVE_AREA_07**

In function Pwm_eMios_BufferTransferEnableDisable:
PWM_EXCLUSIVE_AREA_08

In function Pwm_eMios_SetPowerState: **PWM_EXCLUSIVE_AREA_09**

Table 5-1. Exclusive Areas in Pwm_eMios.c

	PWM_EXCLUSIVE_AREA_00	PWM_EXCLUSIVE_AREA_01	PWM_EXCLUSIVE_AREA_02	PWM_EXCLUSIVE_AREA_03	PWM_EXCLUSIVE_AREA_04	PWM_EXCLUSIVE_AREA_05	PWM_EXCLUSIVE_AREA_06	PWM_EXCLUSIVE_AREA_07	PWM_EXCLUSIVE_AREA_08	PWM_EXCLUSIVE_AREA_09	Interrupt Service Routines Critical Regions(composed diagram)
PWM_EXCLUSIVE_AREA_00	X	X	X	X	X	X	X	X	X	X	
PWM_EXCLUSIVE_AREA_01	X	X	X	X	X	X	X	X	X	X	
PWM_EXCLUSIVE_AREA_02	X	X	X	X	X	X	X	X		X	
PWM_EXCLUSIVE_AREA_03	X	X	X	X	X	X	X	X		X	
PWM_EXCLUSIVE_AREA_04	X	X	X	X	X	X	X	X		X	
PWM_EXCLUSIVE_AREA_05	X	X	X	X	X	X	X	X	X	X	
PWM_EXCLUSIVE_AREA_06	X	X	X	X	X	X	X	X		X	
PWM_EXCLUSIVE_AREA_07	X	X	X	X	X	X	X	X		X	
PWM_EXCLUSIVE_AREA_08	X	X				X			X		

Table continues on the next page...

Table 5-1. Exclusive Areas in Pwm_eMios.c (continued)

	PWM_EXCLUSIVE_AREA_00	PWM_EXCLUSIVE_AREA_01	PWM_EXCLUSIVE_AREA_02	PWM_EXCLUSIVE_AREA_03	PWM_EXCLUSIVE_AREA_04	PWM_EXCLUSIVE_AREA_05	PWM_EXCLUSIVE_AREA_06	PWM_EXCLUSIVE_AREA_07	PWM_EXCLUSIVE_AREA_08	PWM_EXCLUSIVE_AREA_09	Interrupt Service Routines Critical Regions(composed diagram)
PWM_EXCLUSIVE_AREA_09	X	X	X	X	X	X	X	X		X	
Interrupt Service Routines Critical Regions (composed diagram)											

Critical Region Exclusive Matrix

Above is 2 tables depicting the exclusivity between different critical region IDs from the PWM driver. If there is an “X” in a table, it means that those 2 critical regions cannot interrupt each other.

The critical regions from interrupts are grouped in “Interrupt Service Routines Critical Regions (composed diagram)”. If an exclusive area is “exclusive” with the composed “Interrupt Service Routines Critical Regions (composed diagram)” group, it means that it is exclusive with each one of the ISR critical regions.

5.2 Peripheral Hardware Requirements

For MPC574XG controllers, Pwm functionality is provided by the EMIOS channels.

5.3 ISR to Configure Within OS – Dependencies

The table containing ISR's used by Pwm driver is presented below. Depending on the derivative used, some of the ISRs may not be available. For complete details please consult the Reference Manual:

Table 5-2. EMIOS 0 interrupts

EMIOS 0 Interrupts	Hardware interrupt vector
EMIOS_0_CH_0_CH_1_ISR	706
EMIOS_0_CH_2_CH_3_ISR	707
EMIOS_0_CH_4_CH_5_ISR	708
EMIOS_0_CH_6_CH_7_ISR	709
EMIOS_0_CH_8_CH_9_ISR	710
EMIOS_0_CH_10_CH_11_ISR	711
EMIOS_0_CH_12_CH_13_ISR	712
EMIOS_0_CH_14_CH_15_ISR	713
EMIOS_0_CH_16_CH_17_ISR	714
EMIOS_0_CH_18_CH_19_ISR	715
EMIOS_0_CH_20_CH_21_ISR	716
EMIOS_0_CH_22_CH_23_ISR	717
EMIOS_0_CH_24_CH_25_ISR	718
EMIOS_0_CH_26_CH_27_ISR	719
EMIOS_0_CH_28_CH_29_ISR	720
EMIOS_0_CH_30_CH_31_ISR	721

Table 5-3. EMIOS 1 interrupts

EMIOS 1 Interrupts	Hardware interrupt vector
EMIOS_1_CH_0_CH_1_ISR	722
EMIOS_1_CH_2_CH_3_ISR	723
EMIOS_1_CH_4_CH_5_ISR	724
EMIOS_1_CH_6_CH_7_ISR	725
EMIOS_1_CH_8_CH_9_ISR	726
EMIOS_1_CH_10_CH_11_ISR	727
EMIOS_1_CH_12_CH_13_ISR	728
EMIOS_1_CH_14_CH_15_ISR	729
EMIOS_1_CH_16_CH_17_ISR	730
EMIOS_1_CH_18_CH_19_ISR	731
EMIOS_1_CH_20_CH_21_ISR	732
EMIOS_1_CH_22_CH_23_ISR	733
EMIOS_1_CH_24_CH_25_ISR	734
EMIOS_1_CH_26_CH_27_ISR	735

Table continues on the next page...

Table 5-3. EMIOS 1 interrupts (continued)

EMIOS 1 Interrupts	Hardware interrupt vector
EMIOS_1_CH_28_CH_29_ISR	736
EMIOS_1_CH_30_CH_31_ISR	737

Table 5-4. EMIOS 2 interrupts

EMIOS 2 Interrupts	Hardware interrupt vector
EMIOS_2_CH_0_CH_1_ISR	738
EMIOS_2_CH_2_CH_3_ISR	739
EMIOS_2_CH_4_CH_5_ISR	740
EMIOS_2_CH_6_CH_7_ISR	741
EMIOS_2_CH_8_CH_9_ISR	742
EMIOS_2_CH_10_CH_11_ISR	743
EMIOS_2_CH_12_CH_13_ISR	744
EMIOS_2_CH_14_CH_15_ISR	745
EMIOS_2_CH_16_CH_17_ISR	746
EMIOS_2_CH_18_CH_19_ISR	747
EMIOS_2_CH_20_CH_21_ISR	748
EMIOS_2_CH_22_CH_23_ISR	749
EMIOS_2_CH_24_CH_25_ISR	750
EMIOS_2_CH_26_CH_27_ISR	751
EMIOS_2_CH_28_CH_29_ISR	752
EMIOS_2_CH_30_CH_31_ISR	753

NOTE

In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_HW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in hardware vector mode.

5.4 ISR Macro

MCAL drivers use the ISR macro to define the functions that will process hardware interrupts. Depending on whether the OS is used or not, this macro can have different definitions:

a. OS is not used - AUTOSAR_OS_NOT_USED is defined:

i. If USE_SW_VECTOR_MODE is defined:

```
#define ISR(IsrName) void IsrName(void)
```

In this case, drivers' interrupt handlers are normal C functions and the prolog/epilog handle the context save and restore.

ii. If USE_SW_VECTOR_MODE is not defined:

```
#define ISR(IsrName) INTERRUPT_FUNC void IsrName(void)
```

In this case, drivers' interrupt handlers must save and restore the execution context.

Custom OS is used - AUTOSAR_OS_NOT_USED is not defined

```
#define ISR(IsrName) void OS_isr_##IsrName()
```

In this case, OS is handling the execution context when an interrupt occurs. Drivers' interrupt handlers are normal C functions.

Other vendor's OS is used - AUTOSAR_OS_NOT_USED is not defined. Please refer to the OS documentation for description of the ISR macro.

5.5 Other AUTOSAR modules - dependencies

Development Error Tracer:

This module is necessary for enabling Development error detection. The API function used is Det_ReportError(). The activation / deactivation of Development error detection is configurable using the

'PwmDevErrorDetect' configuration parameter.

Mcu:

MCU module shall be initialized before using Pwm.

Port:

PORT module shall configure the EMIOS channels which are used by the Pwm driver.

GPT/ICU:

If an ICU/GPT channel is configured on the same eMIOS module the user shall insure that the ICU/GPT module configures a different MASTER BUS channel and is not overwriting the MASTER BUS used by PWM channels. For more information see ICU, GPT and PWM user manuals.

Chapter 6
Main API Requirements

6.1 Main functions calls within BSW scheduler

None.

6.2 Calls to Notification Functions, Callbacks, Callouts

The Pwm Driver provides a notification per channel that is called whenever the selected edges are generated. The notifications can be configured as pointers to user defined functions. If notification is not desired for a specific channel then 'NULL_PTR' or 'NULL' shall be configured. The syntax of this function is as follows: void Pwm_Notification_#channel(void) An extern declaration of the notification functions is available in Pwm_PBCfg.c. The notification functions have to be implemented by the user.

Chapter 7
Memory Allocation

7.1 Sections to be defined in MemMap.h

Tables descibe Sections to be defined in MemMap.h:

Table 7-1. Sectionto be define

<Section name>	Tyep of section	Description
PWM_START_SEC_CONFIG_DATA_<ALIGNMENT>	Configuration Data	Start of Memory Section for Config Data.
PWM_STOP_SEC_CONFIG_DATA_<ALIGNMENT>	Configuration Data	End of Memory Section for Config Data.
PWM_START_SEC_CODE	Code	Start of memory Section for Code in Flash.
PWM_STOP_SEC_CODE	Code	Stop of memory Section for Code in Flash.
PWM_START_SEC_RAMCODE	Code	Start of memory Section for Code in Ram.
PWM_STOP_SEC_RAMCODE	Code	Stop of memory Section for Code in Ram.
PWM_START_SEC_VAR_< INIT_POLICY>_<ALIGNMENT>	Variables	Start of memory Section for Variables.
PWM_STOP_SEC_VAR_ <INIT_POLICY>_<ALIGNMENT>	Variables	Stop of memory Section for Variables.
PWM_START_SEC_CONST_<ALIGNMENT>	Constant data	Start of memory Section for Constant.
PWM_STOP_SEC_CONST_<ALIGNMENT>	Constant data	Stop of memory Section for Constant.

Which the shortcut '<ALIGNMENT >' means the variable alignment. In order to avoid memory gaps in the allocation variables are allocated according their size. Possible ALIGNMENT postfixes are described in the table at the end of this section.

The shortcut '<INIT_POLICY>' means the initialization policy of variables. Possible '<INIT_POLICY>' postfixes are described in the table at the end of this section.

Tables descibe value range of shortcut ALIGMENT, INIT_POLICY:

Table 7-2. Range of <ALIGNMENT>

<ALIGMENT>	Description
BOOLEAN	Used for variables and constants of size 1 bit
8	Used for variables and constants which have to be aligned to 8 bit. For instance used for variables of size 8 bit or used for composite data types: arrays, structs and unions containing elements of maximum 8 bits
16	Used for variables and constants which have to be aligned to 16 bit. For instance used for variables of size 16 bit or used for composite data types: arrays, structs and unions containing elements of maximum 16 bits
32	Used for variables and constants which have to be aligned to 32 bit. For instance used for variables of size 32 bit or used for composite data types: arrays, structs and unions containing elements of maximum 32 bits
UNSPECIFIED	Used for variables, constants, structure, array and unions when SIZE (alignment) does not fit the criteria of 8,16 or 32 bit. For instance used for variables of unknown size

Table 7-3. Range of <INIT_POLICY>

<INIT_POLICY>	Description
NO-INIT	Used for variables that are never cleared and never initialized by start up code (BSS)
INIT	Used for variables that are initialized with values after every reset

7.2 Linker command file

Memory shall be allocated for every section defined in PWM_MemMap.h

Chapter 8
Configuration parameters considerations

Configuration parameter class for Autosar PWM driver fall into the following variants as defined below:

8.1 Configuration Parameters

Specifies whether the configuration parameter shall be of configuration class Post Build.

Table 8-1. Configuration Parameters

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
Pwm	IMPLEMENTATION_CONFIG_VARIANT	Pre Compile parameter for all Variants of Configuration	Pre Compile
PwmConfigurationOfOptApiServices	PwmDeInitApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmGetOutputState	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmSetDutyCycle	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmSetOutputToIdle	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmSetPeriodAndDuty	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmVersionInfoApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmGetChannelStateApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	MultiPwmChannelSynch	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmSetCounterBusApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmSetChannelOutputApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmSetTriggerDelayApi	Pre Compile parameter for all Variants of Configuration	Pre Compile

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
PwmGeneral	PwmDevErorDetect	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmDutycycleUpdatedEndperiod	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmNotificationSupported	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmPeriodUpdatedEndperiod	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmIndex	Pre Compile parameter for all Variants of Configuration	Pre Compile
	Pwm Enable User Mode Support	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmEnableDualClockMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmChangeRegisterA	Pre Compile parameter for all Variants of Configuratiliteralon	Pre Compile
PwmChannelConfigSet/ PwmChannel	PwmChannelId	VariantPC or VariantPB	Pre Compile or Post Build
	PwmHwIP	VariantPC or VariantPB	Pre Compile or Post Build
	PwmeMiosChannel	VariantPC or VariantPB	Pre Compile or Post Build
	PwmPeriodInTicks	VariantPC or VariantPB	Pre Compile or Post Build
	PwmPeriodDefault	VariantPC or VariantPB	Pre Compile or Post Build
	PwmChannelClass	VariantPC or VariantPB	Pre Compile or Post Build
	PwmPolarity	VariantPC or VariantPB	Pre Compile or Post Build
	PwmDutycycleDefault	VariantPC or VariantPB	Pre Compile or Post Build
	PwmIdleState	VariantPC or VariantPB	Pre Compile or Post Build
	PwmNotification	VariantPC or VariantPB	Pre Compile or Post Build
	PwmMcuClockReferencePoint	VariantPC or VariantPB	Pre Compile or Post Build
PwmChannelConfigSet/ PwmeMios	PwmeMiosModule	VariantPC or VariantPB	Pre Compile or Post Build
PwmChannelConfigSet/ PwmeMios/ PwmeMiosChannels	PwmeMiosChannel	VariantPC or VariantPB	Pre Compile or Post Build
	PwmPrescaler	VariantPC or VariantPB	Pre Compile or Post Build
	PwmPrescaler_Alternate	VariantPC or VariantPB	Pre Compile or Post Build
	PwmModeSelect	VariantPC or VariantPB	Pre Compile or Post Build
	EmiosUnifiedChannelBusSelect	VariantPC or VariantPB	Pre Compile or Post Build
	PwmOffset	VariantPC or VariantPB	Pre Compile or Post Build
	PwmTriggerDelay	VariantPC or VariantPB	Pre Compile or Post Build
	Pwm_Deadtime	VariantPC or VariantPB	Pre Compile or Post Build
	OffsetDelayAdjust	VariantPC or VariantPB	Pre Compile or Post Build
	PwmFreezeEnable	VariantPC or VariantPB	Pre Compile or Post Build

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
PwmChannelConfigSet/ PwmeMios/ PwmeMiosMasterBus	PwmeMiosMasterBus	VariantPC or VariantPB	Pre Compile or Post Build
	MasterModeSelect	VariantPC or VariantPB	Pre Compile or Post Build
	MasterBusPeriodInTicks	VariantPC or VariantPB	Pre Compile or Post Build
	MasterBusPrescaler	VariantPC or VariantPB	Pre Compile or Post Build
	MasterBusPeriodDefault	VariantPC or VariantPB	Pre Compile or Post Build
	MasterBusPrescaler_Alternate	VariantPC or VariantPB	Pre Compile or Post Build
CommonPublishedInformation	ArReleaseMajorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	ArReleaseMinorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	ArReleaseRevisionVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	ModuleId	Pre Compile parameter for all Variants of Configuration	Pre Compile
	SwMajorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	SwMinorVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	SwPatchVersion	Pre Compile parameter for all Variants of Configuration	Pre Compile
	VendorApiInfix	Pre Compile parameter for all Variants of Configuration	Pre Compile
	VendorId	Pre Compile parameter for all Variants of Configuration	Pre Compile

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating Pulse Width Modulation :

- Generate the required PWM configurations. For more details refer to section Files required for Compilation
- Allocate proper memory sections in PWM_MemMap.h and linker command file. For more details refer to section
- Compile & build the PWM with all the dependent modules. For more details refer to section Building the Driver

Chapter 10
External Assumptions for PWM driver

The section presents requirements that must be complied with when integrating PWM driver into the application.

[PWM075c]

Std_Types.h shall include Compiler.h and Platform_Types.h.

[PWM066c]

The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

NOTE

[PWM089]

The Pwm module's user shall ensure the integrity if several function calls are made during run time in different tasks or ISRs for the same PWM channel.

[PWM093]

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

[PWM116]

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

[PWM120a]

For pre-compile and link time configuration variants, a NULL pointer shall be passed to the initialization routine.

[PWM086a]

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

[PWM086b]

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

[PWM119]

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

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number IM35PWMASR4.2 Rev0002 R1.0.0
Revision 5.0.0

AUTOSAR