Integration Manual

for MPC563XM PWM MCAL Driver

Document Number: IM14PWMASR3.0R2.0.0
Rev. 1.1

freescale™

Contents

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	14/Feb/2011	Sinu Joji Isac	first version
1.1	20/Dec/2011	Subramanya M Naik	Updated for MPC5634M RTM 2.0.0 Release

Chapter 2 Introduction

This Integration Manual describes the integration requirements for Autosar PWM Driver for Freescale Semiconductor's MPC5634M microcontrollers.

The roadmap for the document is as follows:

Building the Driver : This section gives a brief overview of the build procedure (compiler,linker options and source files) and Plugins setup.

Function Calls to Module : This section lists the various function calls to modules during Start-up, Shutdown and Wake-up.

Module Requirements : This section specifies the various module requirements related to

- Exclusive areas to be defined in BSW scheduler
- Peripheral Hardware Requirements
- Specific interface to other modules
- ISR to configure within OS
- Dependencies with other AUTOSAR modules

Main API Requirements : This section specifies the requirements related to to the main Pwm API and gives a brief overview of the main functions calls within BSW scheduler, API_Name Requirements and calls to notification functions, callbacks, callouts.

Memory Allocation : This section describes the memory allocation requirements namely the sections to be defined in MemMap.h and the linker command file.

Configuration Parameters : This section covers the various Pre Compile, Link Time and Post Build time configuration parameters.

Integration Steps : This section describes in brief the steps for integrating Pwm module.

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
BSW	Basic Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
PWM	Pulse Width Modulation
ISR	Interrupt Service Routine
OS	Operating System
MCU	Microcontroller Unit
GUI	Graphical User Interface
API	Application Programming Interface
PB Variant	Post Build Variant
PC Variant	Pre Compile Variant

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
Building the Driver

This section describes the source files and various compiler, linker options used for building the Autosar Pwm driver for Freescale Semiconductor's MPC5634M microcontrollers. It also explains the Plugins setup procedure.

3.1 Build Options

The driver files are compiled using GHS 5.1.7, COSMIC Software PPC C Cross Compiler V4.3.4 - 16 Nov 2011 - Win32-F,DIAB 5_8_0_02 wind00198363 20100511 123238 and CW Version 4.3 build 182 . The compiler, linker flags used for building the driver are explained below:

Note:

The Pwm_TS_T2D14M20I0R0 is composed as follow: TS_T<Target_Id>D<Derivative_Id>M<SW_Version_Major><SW_Version_Minor>I<SW_Patch_Version>R0 (i.e. Target_Id = 2 identifies PowerPC architecture and Derivative_Id = 14 identifies the MPC5634M derivatives)

3.1.1 GHS Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-cpu= ppc5646	Selects target processor: ppc5646x
-ansi	Enforces strict ANSI mode (C89 standard)
-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.

Table continues on the next page...

Table 3-1. Compiler Options (continued)

Option	Description
--no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup. This may improve optimizations by giving the compiler optimizer more information about the location of the variable.
-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
-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.
-DGHS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GHS preprocessor symbol.

Table 3-2. Assembler Options

Option	Description
-cpu= ppc5646	Selects target processor: ppc5646

Table 3-3. Linker Options

Option	Description
-cpu= ppc5646	Selects target processor: ppc5646
-nostartfiles	Do not use Start files.
-vle	Enables VLE code generation

Table continues on the next page...

Table 3-3. Linker Options (continued)

Option	Description
-linker_warnings	Display linker warnings

3.1.2 DIAB Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
-tPPCE200Z4VEG:simple	Sets target processor to PPCE200Z4, generates ELF using EABI conventions, All Single Hardware Floating Point (Single precision uses hardware, double precision is mapped to single precision), 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
-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.
-Xsmall-const=0	Set Size Limit for “small const” Variables to zero.
-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
-Xalign-functions=4	Align each function on an address boundary divisible by 4
-g	Generate symbolic debugger information. Do most target-independent optimizations. Also, disable most target-dependent optimizations: option -g2 also disables basic reordering and all peephole optimizations.
-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
-Xforce-prototypes	Generate warnings if a function is used without a previous prototype declaration
-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
-Xlint	Generate warnings when suspicious and non-portable C code is encountered. Enables all warnings
-ei1604	Suppress the warning messages 1604.
-W:as:,-l	Pass the option “-l” (lower case letter L) to the assembler to get an assembler listing file

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
-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 3-5. Assembler Options

Option	Description
-tPPCE200Z4VEN:simple	Selects target processor: PPCE200Z4, generates ELF using EABI conventions, NO floating point support, 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

Table 3-6. Linker Options

Option	Description
-tPPCE200Z4VEN:simple	Selects target processor: PPCE200Z4, generates ELF using EABI conventions, NO floating point support, 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
-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.
-Xlibc-old	Enables usage of legacy (pre-release 5.6) libraries

3.1.3 CW Compiler/Linker/Assembler Options

Table 3-7. Compiler Options

Option	Description
-proc Zen	Generates and links object code for Zen processor. The compiler uses unsigned as the default parameter for the -char switch
-lang c	Expects source code to conform to the language specified by the ISO/IEC 9899-1990 ("C90") standard
-opt all	This option is selected all optimization (the same as -opt speed,level=4,intrinsics,noframe)
-common off	Disables moving uninitialized data into a common section
-sdatathreshold 0	Specifies the threshold size (in bytes) for an item considered by the linker to be small data. (The linker stores small data items in the Small Data address space. The compiler can generate faster code to access this data.)
-sdata2threshold 0	Specifies the threshold size (in bytes) for an item considered by the linker to be small constant data. (The linker stores small constant data items in the Small Constant Data address space.)
-vle	Tells the compiler and linker to generate and lay out Variable Length Encoded (VLE) instructions, available on Zen variants of Power Architecture processors
-use_lmw_stmw on	Enables the use of multiple load and store instructions for function prologues and epilogues
-ppc_asm_to_vle	Converts regular Power Architecture assembler mnemonics to equivalent VLE (Variable Length Encoded) assembler mnemonics in the inline assembler
-cpp_exceptions off	When on, generates executable code for C++ exceptions. When off, generates smaller, faster executable code
-func_align 4	Specifies alignment of functions in executable code
-sym dwarf-2,full	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format)
-gdwarf-2	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format). The linker ignores debugging information that is not in the Dwarf 1, Dwarf 2 format
-w on	Turns on most warning messages
-r	Compiler should expect function prototypes
-w undefmacro	Issues warning messages on the use of undefined macros in #if and #elif conditionals
-char unsigned	Controls the default sign of the char data type: char data items are unsigned
-nosyspath	Performs a search of both the user and system paths, treating #include statements of the form #include xyz the same as the form #include "xyz"
-fp none	No floating point code generation
-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.

Table continues on the next page...

Table 3-7. Compiler Options (continued)

Option	Description
-DMWERKS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the CWpreprocessor symbol.

Table 3-8. Assembler Options

Option	Description
-proc Zen	Generates and links object code for Zen processor. The compiler uses unsigned as the default parameter for the -char switch
-vle	Tells the compiler and linker to generate and lay out Variable Length Encoded (VLE) instructions, available on Zen variants of Power Architecture processors
-sym dwarf-2,full	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format)
-gdwarf-2	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format). The linker ignores debugging information that is not in the Dwarf 1, Dwarf 2 format.

Table 3-9. Linker Options

Option	Description
-proc Zen	Generates and links object code for Zen processor. The compiler uses unsigned as the default parameter for the -char switch
-code_merging all	Removes duplicated functions to reduce object code size
-far_near_addressing	Simplifies address computations to reduce object code size and improve performance
-vle_enhance_merging	Removes duplicated functions that are called by functions that use VLE instructions to reduce object code size
-listdwarf	DWARF debugging information in the linker's map file
-sym dwarf-2,full	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format)
-char unsigned	Controls the default sign of the char data type: char data items are unsigned.

3.1.4 CSMC Compiler/Linker/Assembler Options

Table 3-10. Compiler Options

Option	Description
-l	Create listing file; this option directs the compiler to produce an assembly language file with C source line interspersed in it. Please note that the C source lines are commented in the assembly language file: they start with ';'.
+modvc	Memory model with "medium size" application, in detail: "data" less than 64kb, "constants" less than 64kb, no code size limit
+rev	Tells the compiler to reverse the order of bits in the bitfields. You need this option in order to use most non-Cosmic header files.

Table continues on the next page...

Table 3-10. Compiler Options (continued)

Option	Description
-pc99	authorize the repetition of the const and volatile modifiers in the declaration either directly or indirectly in the typedef.
-odB5	disable the optimization B5.
-pxf	prefix filenames in the debug information with absolute full path name.
+debug	produce debug information to be used by the debug utilities provided with the compiler and by any external debugger.
-DCSMC	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the CSMC preprocessor symbol.
-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
-DEU_DISABLE_ANSILIB_CALLS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the EU_DISABLE_ANSILIB_CALLS preprocessor symbol.
-DMCAL_CER_VALIDATION	-D defines a preprocessor symbol for CER Report
-DMCAL_VERSION_CHECK	-D defines enable the cross check between the AutoSar component Version Numbers

Table 3-11. Assembler Options

Option	Description
-l	create a listing file. The name of the listing file is derived from the input file name by replacing the suffix by the ".ls" extension

Table 3-12. Linker Options

Option	Description
-p	display symbols with physical address instead of logical address in the map file.

3.2 Files Required for the Compilation

This section describes the include files required to compile, assemble (if assembler code) and link the PWM driver for MPC5634M 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 Pwm be compiled.

PWM Files

- ..\Pwm_TS_T2D14M20I0R0\include\Pwm.h
- ..\Pwm_TS_T2D14M20I0R0\include\Pwm_LLD.h

- ..\Pwm_TS_T2D14M20I0R0\include\Pwm_NonASR.h
- ..\Pwm_TS_T2D14M20I0R0\include\Reg_eSys_EMIOS.h
- ..\Pwm_TS_T2D14M20I0R0\include\Reg_eSys_EMIOS_CfgEx.h
- ..\Pwm_TS_T2D14M20I0R0\include\eMIOS_Pwm_LLD.h
- ..\Pwm_TS_T2D14M20I0R0\include\eMIOS_Pwm_LLD_CfgEx.h
- ..\Pwm_TS_T2D14M20I0R0\src\Pwm.c
- ..\Pwm_TS_T2D14M20I0R0\src\Pwm_Irq.c
- ..\Pwm_TS_T2D14M20I0R0\src\Pwm_LLD.c
- ..\Pwm_TS_T2D14M20I0R0\src\Pwm_NonASR.c
- ..\Pwm_TS_T2D14M20I0R0\src\Reg_eSys_EMIOS_IRQ_PWM.c
- ..\Pwm_TS_T2D14M20I0R0\src\Reg_eSys_EMIOS_PWM.c
- ..\Pwm_TS_T2D14M20I0R0\src\eMIOS_Pwm_LLD.c
- ..\Pwm_TS_T2D14M20I0R0\src\ eMIOS_Pwm_LLD_IRQ.c

PWM Generated Files

- Pwm_PBcfg.c (For PB Variant) - This file should be generated by the user using a configuration tool for compilation.
- Pwm_Cfg.c (For PC Variant) - This file should be generated by the user using a configuration tool for compilation.
- Pwm_Cfg.h (For PC and PB Variants) - This file should be generated by the user using a configuration tool for compilation.

Files from Base common folder

- ..\Base_TS_T2D14M20I0R0\include\Compiler.h
- ..\Base_TS_T2D14M20I0R0\include\Compiler_Cfg.h
- ..\Base_TS_T2D14M20I0R0\include\ComStack_Types.h
- ..\Base_TS_T2D14M20I0R0\include\MemMap.h
- ..\Base_TS_T2D14M20I0R0\include\Mcal.h
- ..\Base_TS_T2D14M20I0R0\include\Platform_Types.h
- ..\Base_TS_T2D14M20I0R0\include\Std_Types.h
- ..\Base_TS_T2D14M20I0R0\include\Reg_eSys.h
- ..\Base_TS_T2D14M20I0R0\include\Soc_Ips.h
- ..\Base_TS_T2D14M20I0R0\include\Reg_Macros.h
- ..\Base_TS_T2D14M20I0R0\include\Cer.h

Files from PwmIf folder:

- ..\PwmIf_TS_T2D14M20I0R0\include\PwmIf_Cbk.h
- ..\PwmIf_TS_T2D14M20I0R0\include\ComStack_Types.h

Files from Dem folder:

- ..\Dem_TS_T2D14M20I0R0\include\Dem.h
- ..\Dem_TS_T2D14M20I0R0\include\Dem_IntErrId.h

- ..\Dem_TS_T2D14M20I0R0\include\Dem_Types.h
- ..\Dem_TS_T2D14M20I0R0\include\Dem.c

Files from Det folder:

- ..\Det_TS_T2D14M20I0R0\include\Det.h
- ..\Det_TS_T2D14M20I0R0\include\Det.c

Files from Schm folder:

- ..\SchM_TS_T2D14M20I0R0\include\Schm_Pwm.h
- ..\SchM_TS_T2D14M20I0R0\src\Schm_Pwm.c

3.3 Setting up the Plugins

The PWM driver was designed to be configured by using the EB Tresos Studio (version TS_T2D14M20I0R0 or later.)

Location of various files inside the FR module folder:

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\Pwm_TS_T2D14M20I0R0\config\Pwm.xdm
  - ..\PwmIf_TS_T2D14M20I0R0\config\PwmIf.xdm
  - ..\EcuM_TS_T2D14M20I0R0\config\EcuM.xdm
  - ..\Base_TS_T2D14M20I0R0\config\Base.xdm
  - ..\Resource_TS_T2D14M20I0R0\config\Resource.xdm
  - ..\Mcu_TS_T2D14M20I0R0\config\Mcu.xdm

- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\Pwm_TS_T2D14M20I0R0\autosar\Pwm.epd
  - ..\PwmIf_TS_T2D14M20I0R0\autosar\PwmIf.epd
  - ..\EcuM_TS_T2D14M20I0R0\autosar\EcuM.epd
  - ..\Mcu_TS_T2D14M20I0R0\autosar\Mcu.epd

- Code Generation Templates for Pre-Compile time configuration parameters:
  - ..\Pwm_TS_T2D14M20I0R0\generate_PC\include\Pwm_Cfg.h
  - ..\Pwm_TS_T2D14M20I0R0\generate_PC\include\Pwm_Cfg.c
  - ..\Pwm_TS_T2D14M20I0R0\generate_PC\Pwm_Clock_Tree.m
  - ..\Pwm_TS_T2D14M20I0R0\generate_PC\Pwm_VersionCheck_Inc.m
  - ..\Pwm_TS_T2D14M20I0R0\generate_PC\Pwm_VersionCheck_Src.m
  - ..\Pwm_TS_T2D14M20I0R0\generate_PC\Pwm_NotifyCheck_Src.m

- Code Generation Templates for Post-Build time configuration parameters:
  - ..\Pwm_TS_T2D14M20I0R0\generate_PB\include\Pwm_Cfg.h

- ..\Pwm_TS_T2D14M20I0R0\generate_PB\src\ Pwm_PBcfg.c
- ..\Pwm_TS_T2D14M20I0R0\generate_PB\Pwm_Clock_Tree.m
- ..\Pwm_TS_T2D14M20I0R0\generate_PB\Pwm_VersionCheck_Src_PB.m
- ..\Pwm_TS_T2D14M20I0R0\generate_PB\Pwm_NotifyCheck_Src_PB.m

Steps to generate the configuration:

1. Copy the module folders Pwm_TS_T2D14M20I0R0, PwmIf_TS_T2D14M20I0R0, Base_TS_T2D14M20I0R0, Resource_TS_T2D14M20I0R0, EcuM_TS_T2D14M20I0R0, Mcu_TS_T2D14M20I0R0TS_T2D14M20I0R0TS_T2D14M20I0R0 into 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

- **MCU** is required for the reference clock.
- **RESOURCE** is required to select processor derivative. Current PWM driver has support for the following derivatives, everyone having attached a Resource file: mpc5634m_bga208, mpc5634m_qfp144, mpc5634m_qfp176.
- **ECUM** is required for selecting the reference to the wakeup source for every PWM controller.
- **DET** is required for signaling the development error detection (parameters out of range, null pointers, etc).
- **SchM** is required in order to support the critical sections.
- **DEM** is required for signaling the production error detection (hardware failure, etc).
- **Base**:is required to get the Base address of all the registers

Resource Parameters Configuration

1. Pwm.PwmChannelConfigSet.PwmChannel.PwmHwChannel : Selects the physical PWM channel
2. Pwm.EmiosModules: number of Emios modules
3. Pwm.Emios0Channels: number of channels on Emios0
4. Pwm.Emios16BitsVariant: 16bits or 24bits Emios implementation
5. Pwm.EmiosOpwmtAvailable: specifies if Opwmt mode is implemented

Chapter 4
Function Calls to Module

4.1 Function Calls during Start-up

This driver does not need OS Support except for ISRs. Hence can be initialized either in STARTUP1 or STARTUP2 phase of EcuM initialization. This depends on the implementation, desired duration for STARTUP1 & Target hardware design. The API to be called is Pwm_Init(ConfigPtr).

4.2 Function Calls during Shutdown

- Pwm_DeInit() can be called to deinitialize the PWM driver

4.3 Function Calls during Wake-up

None.

Chapter 5
Module Requirements

5.1 Exclusive areas to be defined in BSW scheduler

In the current implementation, PWM is using the services of Schedule Manager (SchM) for entering and exiting the critical regions.

Exclusive areas are used by the Pwm driver for each Pwm channel and are “channel wise” thus not fixed.

Each exclusive area protects the Pwm channels from concurrent accesses.

The protected resources are the hw registers and the global variables associated with each Pwm channel. The SchM implementation must support variables as effective parameters as the Pwm logical channel (for eMIOS) variable is used to determine the critical section.

The ISR critical regions must not block the other critical regions to avoid deadlocks. This is ensured by exiting the ISR critical region before calling the user notification functions.

In this case the SchM implementation must provide the required resources.

The following critical regions are used in the PWM driver:

5.2 LogicalChannelNumber

Used in functions EMIOS_Pwm_LLD_SetDutyCycle, EMIOS_Pwm_LLD_SetOutputToIdle, EMIOS_Pwm_LLD_GetOutputState, EMIOS_Pwm_LLD_EnableNotification, EMIOS_Pwm_LLD_DisableNotification, EMIOS_Pwm_LLD_SetCounterBus, EMIOS_Pwm_LLD_SetChannelOutput.

5.3 PWM_EXCLUSIVE_AREA_00

Used in functions EMIOS_Pwm_LLD_SetDutyCycle, EMIOS_Pwm_LLD_SetOutputToIdle, EMIOS_Pwm_LLD_GetOutputState, EMIOS_Pwm_LLD_EnableNotification, EMIOS_Pwm_LLD_DisableNotification, EMIOS_Pwm_LLD_SetCounterBus, EMIOS_Pwm_LLD_SetChannelOutput to protect the registers shared between the channels of the EMIOS..

5.4 Critical region exclusivity matrix

Below is the table 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.

Table 5-1. Critical region exclusivity matrix

	PWM _EA_00	LogicalChannel Number i	LogicalChannel Number j
PWM _EA_00	X		
LogicalChannel Number i		X	
LogicalChannel Number j			X

Note

- PWM_EA_xx means PWM_EXCLUSIVE_AREA_xx

5.5 Peripheral Hardware Requirements

For MPC5634M controllers eMIOS provide PWM functionality.

5.6 ISR to Configure within OS - Dependencies

5.6.1 The following ISR’s are used by the PWM driver

Table 5-2. ISR List

ISR Name	Hardware Interrupt vector
ISR(EMIOS_0_CH_0_ISR)	51
ISR(EMIOS_0_CH_2_ISR)	53
ISR(EMIOS_0_CH_4_ISR)	55
ISR(EMIOS_0_CH_8_ISR)	59
ISR(EMIOS_0_CH_9_ISR)	60
ISR(EMIOS_0_CH_10_ISR)	61
ISR(EMIOS_0_CH_11_ISR)	62
ISR(EMIOS_0_CH_12_ISR)	63
ISR(EMIOS_0_CH_13_ISR)	64
ISR(EMIOS_0_CH_14_ISR)	65
ISR(EMIOS_0_CH_15_ISR)	66
ISR(EMIOS_0_CH_23_ISR)	209

5.6.2 Macros for Interrupts

For the ISR EMIOS_0_CH_X_ CH_Y_ISR must be used in the corresponding vector locations. ISR EMIOS_0_CH_X_ CH_Y_ISR is a common interrupt handler and is shared by GPT, ICU and PWM. Through configuration this common handler will route the interrupt to the driver that uses this Emios unified channel. 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.

5.7 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:

1. OS is not used - AUTOSAR_OS_NOT_USED is defined:
   - 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.

- 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.

2. Freescale Semiconductor 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.
3. 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.

   Please refer to the OS documentation for description of the ISR macro.

5.8 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.

Diagnostic Event Manager:

This module is necessary for enabling Production error detection. The API function used is Dem_ReportErrorStatus ().

Mcu:

MCU module shall be initialized before using Pwm.

This module is required for setting the eMIOS global Pre-scalar value and clock.

Port:

PORT module shall configure the eMIOS channels which are to be used for the Pwm

Configuration dependency to other module:

Care must be used not to allocate the same eMIOS channels to other MCAL drivers (ICU/GPT).

Chapter 6
Main API Requirements

6.1 Main functions calls within BSW scheduler

6.2 API Requirements

Not Applicable.

6.3 Calls to Notification Functions, Callbacks, Callouts

Call-back Notifications:

None

User Notification

- 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’ shall be configured.

  The syntax of this function is as follows:

  void PWM_Notification_channel(void)

  An extern declaration of this function is available in PWM_Cfg.c and PWM_PBcfg.c. The function has to be implemented by the user.

Chapter 7
Memory Allocation

7.1 Sections to be defined in MemMap.h

For Post Build data:

#ifdef PWM_START_CONFIG_DATA_UNSPECIFIED

#undef PWM_START_CONFIG_DATA_UNSPECIFIED

#undef MEMMAP_ERROR

*/*Memory Section for Post Build Data to be defined here. Example given in the next line*/*

#pragma ghs section const=".pb PWM_cfg"

#endif

#ifdef PWM_STOP_CONFIG_DATA_UNSPECIFIED

#undef PWM_STOP_CONFIG_DATA_UNSPECIFIED

#undef MEMMAP_ERROR

*/*End of section to be mentioned here. Example given in the next line.*/*

#pragma ghs section

#endif

For Code:

#ifdef PWM_START_SEC_CODE

```
#undef PWM_START_SEC_CODE
#undef MEMMAP_ERROR
/*Memory Section for Code to be defined here.*/
#endif


#ifdef PWM_STOP_SEC_CODE
#undef PWM_STOP_SEC_CODE
#undef MEMMAP_ERROR
/*End of section to be mentioned here*/
#endif
```

For Variables:

```
#ifdef PWM_START_SEC_VAR_UNSPECIFIED
#undef PWM_START_SEC_VAR_UNSPECIFIED
#undef MEMMAP_ERROR
/*Memory Section for Variables to be defined here.*/
#endif


#ifdef PWM_STOP_SEC_VAR_UNSPECIFIED
#undef PWM_STOP_SEC_VAR_UNSPECIFIED
#undef MEMMAP_ERROR
/*End of section to be mentioned here*/
#endif
```

For Constant data:

```
#ifdef PWM_START_SEC_CONST_UNSPECIFIED
```

```
#undef PWM_START_SEC_CONST_UNSPECIFIED
#undef MEMMAP_ERROR
/*Memory Section for Constants to be defined here.*/
#endif


#ifdef PWM_STOP_SEC_CONST_UNSPECIFIED
#undef PWM_STOP_SEC_CONST_UNSPECIFIED
#undef MEMMAP_ERROR
/*End of section to be mentioned here*/
#endif
```

7.2 Linker Command File

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

Chapter 8
Configuration Parameters

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

8.1 CONFIGURATION PARAMETER CONSIDERATIONS

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

Table 8-1. Configuration Parameters

Configuration Container	**Configuration Parameters**	**Configuration Variant**	**Current Implementation**
PwmGeneral	PwmDevErrorDetect	Pre Compile parameter for all Variants of Configuration	PreCompile
	PwmChangeRegisterA	Pre Compile parameter for all Variants of Configuration	PreCompile
	PwmDutycycleUpdatedEndperiod	Pre Compile parameter for all Variants of Configuration	This parameter is not used in the current implementation.
	PwmNotificationSupported	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmPeriodUpdatedEndperiod	Pre Compile parameter for all Variants of Configuration	This parameter is not used in the current implementation.
	PwmIndex	Pre Compile parameter for all Variants of Configuration	This parameter is not used in the current implementation.
	PwmClockFromMCU	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmCpuClockRef	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmEmiosClockValue	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmGenerateClockTreeDebugInfo	N/A Debug purpose only	N/A Debug purpose only

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

PwmConfigurationOfOptApiServices	PwmGetOutputState	Pre Compile parameter for all Variants of Configuration25	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
	PwmDeInitApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmVersionInfoApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
PwmNonAUTOSAR	PwmSetCounterBusApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmSetChannelOutputApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmEnableDualClockMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
PwmChannel	PwmChannelClass	VariantPC or VariantPB	Post Build
	PwmChannelId	VariantPC or VariantPB	Pre Compile
	PwmHwChannel	VariantPC or VariantPB	Pre Compile
	PwmPolarity	VariantPC or VariantPB	Post Build
	PwmPeriodDefaultUnits	VariantPC or VariantPB	Post Build
	PwmPeriodDefault	VariantPC or VariantPB	Post Build
	PwmDutycycleDefault	VariantPC or VariantPB	Post Build
	PwmIdleState	VariantPC or VariantPB	Post Build
	PwmNotification	VariantPC or VariantPB	Post Build
	PwmModeSelect	VariantPC or VariantPB	Post Build
	EmiosUnifiedChannelBusSelect	VariantPC or VariantPB	Post Build
	PwmPrescaler	VariantPC or VariantPB	Post Build
	PwmPrescaler_Alternate	VariantPC or VariantPB	Post Build
	PwmOffset	VariantPC or VariantPB	Post Build
	PwmTriggerDelay	VariantPC or VariantPB	Post Build
	PwmFreezeEnable	VariantPC or VariantPB	Post Build

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating Pwm:

- Generate the required Pwm configurations. For more details refer to section **Calls to Notification Functions, Callbacks, Callouts**.
- Allocate proper memory sections in MemMap.h and linker command file. For more details refer to section **Sections to be defined in MemMap.h**.
- Make sure all include files for compilation are as per section**ISR macro**.
- Map the ISRs to their vector locations. For more details refer to section **Peripheral Hardware Requirements**.
- Compile & build with all the dependent modules. For more details refer to section **Building the Driver** and **ISR to Configure within OS - Dependencies** .