Integration Manual

for S32K14X PWM Driver

Document Number: IM2PWMASR4.2 Rev0002 R1.0.1
Rev. 1.0

Contents

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

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	13/07/2018	NXP MCAL Team	Updated version for ASR 4.2.2S32K14X1.0.1 Release

Chapter 2
Introduction

This integration manual describes the integration requirements for PWM Driver for S32K14X microcontrollers.

2.1 Supported Derivatives

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

Table 2-1. S32K14X Derivatives

NXP Semiconductors	s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64

All of the above microcontroller devices are collectively named as S32K14X .

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
FTM	Flexible Timer
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	Micro Controller Unit
N/A	Not Applicable
OS	Operating System

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
PB Variant	Post Build Variant
PC Variant	Pre Compile Variant
PWM	Pulse Width Modulation
RAM	Random Access Memory
ROM	Read-only Memory
VLE	Variable Length Encoding

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	Specification of PWM Driver	AUTOSAR Release 4.2.2
2	S32K14X Reference Manual	Reference Manual, Rev. 7, 04/2018
3	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	26/02/2018
4	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	30/11/2017
5	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	30/11/2017
6	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	30/11/2017
7	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	30/11/2017

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 SemiconductorsS32K14X . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The PWM driver files are compiled using

- Green Hills Multi 7.1.4 / Compiler 2017.1.4
- (Linaro GCC 6.3-2017.06~dev) 6.3.1 20170509 (Thu Dec 7 13:28:42 CST 2017 build.sh rev=g7fea41d s=L631 Earmv7 -V release_g7fea41d_build_Fed_Earmv7)
- IAR: V8.11.2

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

Note

The TS_T40D2M10I1R0 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 = 40 identifies CORTEXM architecture and Derivative_Id = 2 identifies the S32K14X)

3.1.1 GHS Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-ansi	Specifies ANSI C with extensions. This mode extends the ANSI X3.159-1989 standard with certain useful and compatible constructs.
-Osize	Optimize for size.
-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
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
--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
-c	Produces an object file (called input-file.o) for each source file.
--no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup.
-keeptempfiles	Prevents the deletion of temporary files after they are used. If an assembly language file is created by the compiler, this option will place it in the current directory instead of the temporary directory. Produces an object file (called input-file.o) for each source file.
-list	Creates a listing by using the name of the object file with the .lst extension. Assembler option
-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
-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.

Table 3-2. Assembler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-c	Produces an object file (called input-file.o) for each source file.
-preprocess_assembly_files	Preprocesses assembly files
-asm=list	Creates a listing by using the name of the object file with the .lst extension. Assembler option

Table 3-3. Linker Options

Option	Description
-Mn	Map file numeric ordering
-delete	Removal from the executable of functions that are unused and unreferenced
-v	Display removed unused functions
-ignore_debug_references	Ignores relocations from DWARF debug sections when using -delete.
-map	Creates a detailed map file
-keepmap	Keep the map file in the event of a link error
-lstartup	Link libstartup library -Run-time environment startup routines
-lsys	Link libsys library -Run-time environment system routines
-larch	Link libarch library -Target-specific run-time support. Any file produced by the Green Hills Compiler may depend on symbols in this library.
-lansi	Link libansi library -the standard C library
-L(/lib/thumb2)	Link thumb2 library
-lutf8_s32	Include utf8_s32.a to use the Wide Character Functions

3.1.2 IAR Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu_mode=thumb	Selects generating code that executes in Thumb state.
--endian=little	Specifies the endianess of core: little endian.
-Ohz	Sets the optimization level to High, favoring size.
-c	Produces an object file (called input-file.o) for each source file.
--no_clustering	Disables static clustering optimizations.
--no_mem_idioms	Makes the compiler to not optimize code sequences that clear, set, or copy a memory region.
--no_explicit_zero_opt	Places the zero initialized variables in data section instead of bss.
--debug	Makes the compiler include information in the object modules.
--diag_suppress=Pa050	Suppresses diagnostic messages (warnings) about non-standard line endings.

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
-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
-DIAR	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the IAR preprocessor symbol.
--require_prototypes	Forces the compiler to verify that all functions have proper prototypes.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by compiler.
--no_system_include	Disables the automatic search for system include files.
-e	Enables language extensions. This option is needed by FLS driver which uses _packed structures.

Table 3-5. Assembler Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu_mode=thumb	Selects generating code that executes in Thumb state.
-g	Use this option to disable the automatic search for system include files.

Table 3-6. Linker Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--map filename	Produces a map file.
--no_library_search	Disables automatic runtime library search.
--entry _start	Treats the symbol _start as a root symbol and as the start of the application.
--enable_stack_usage	Enables stack usage analysis.
--skip_dynamic_initialization	Suppress dynamic initialization during system startup.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by linker.
--config	Specifies the configuration file to be used by the linker.

3.1.3 GCC Compiler/Linker/Assembler Options

Table 3-7. Compiler Options

Option	Description
-c	Produces an object file (called input-file.o) for each source file.
-O1	Use optimization for size.
-ggdb3	Produce debugging information for use by GDB. Level 3 includes extra information, such as all the macro definitions present in the program.

Table continues on the next page...

Table 3-7. Compiler Options (continued)

Option	Description
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mthumb	Selects generating code that executes in Thumb state.
-mlittle-endian	Generate code for a processor running in little-endian mode.
-fomit-frame-pointer	Removes the frame pointer for all functions, which might make debugging harder.
-mhard-float	Use hardware floating-point instructions.
-fno-common	Specifies that the compiler should place uninitialized global variables in the data section of the object file, rather than generating them as common blocks.
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
-Wextra	Enables some extra warning flags that are not enabled by '-Wall'.
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types.
-Wno-sign-compare	Do not warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned.
-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
-DLINARO	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the LINARO preprocessor symbol.
-fstack-usage	Geneates an extra file that specifies the maximum amount of stack used, on a per-function basis.
-fdump-ipa-all	Enables all inter-procedural analysis dumps.
-Werror=implicit-function-declaration	Generates an error when the prototype of the function is not defined..

Table 3-8. Assembler Options

Option	Description
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-c	Produces an object file (called input-file.o) for each source file.
-mthumb	This option specifies that the assembler should start assembling Thumb instructions.
-x assembler-with-cpp	Indicates that the assembly code contains C directives and the C preprocessor must be run.

Table 3-9. Linker Options

Option	Description
-Map=filename	Print a link map to the file mapfile.
-T scriptfile	Use scriptfile as the linker script. This script replaces ld's default linker script(rather than adding to it), so commandfile must specify everything necessary to describe the output file.

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 S32K14X 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_T40D2M10I1R0 \src\Pwm.c
- ..\ PWM_TS_T40D2M10I1R0 \include\Pwm.h
- ..\ PWM_TS_T40D2M10I1R0 \include\Pwm_EnvCfg.h
- ..\ PWM_TS_T40D2M10I1R0 \include\Pwm_Types.h
- ..\ PWM_TS_T40D2M10I1R0 \include\Pwm_Notif.h
- ..\ PWM_TS_T40D2M10I1R0 \src\Pwm_Ipw.c
- ..\ PWM_TS_T40D2M10I1R0 \include\Pwm_Ipw.h
- ..\ PWM_TS_T40D2M10I1R0 \include\Pwm_Ipw_Notif.h
- ..\ PWM_TS_T40D2M10I1R0 \include\Pwm_Ipw_Types.h
- ..\ PWM_TS_T40D2M10I1R0 \src\Pwm_Ftm.c
- ..\ PWM_TS_T40D2M10I1R0 \include\Pwm_Ftm.h
- ..\ PWM_TS_T40D2M10I1R0 \include\Pwm_Ftm_Irq.h
- ..\ PWM_TS_T40D2M10I1R0 \include\Pwm_Ftm_Types.h

PWM Generated Files

- Pwm_Cfg.h
- Pwm_VS_<VariantNo>_PBcfg.c

For driver compilation, Pwm_VS_<VariantNo>_PBcfg.c should be generated by the user using a configuration tool. The file contains the definition of the init pointer for the respective variant.

As a deviation from standard: Pwm_VS_<VariantNo>_PBcfg.c files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB).

- Pwm_Cfg.c file will contain the definition for all configuration structures containing only variables that are not variant aware, configured and generated only once. This file alone does not contain the whole structure needed by Pwm_Init function to configure the driver. Based on the number of variants configured in the EcuC, there can be more than one configuration structure for one module even for PreCompile variant.

Files from Base common folder

- ..\Base_TS_T40D2M10I1R0 \include\Compiler.h
- ..\Base_TS_T40D2M10I1R0 \include\Compiler_Cfg.h
- ..\Base_TS_T40D2M10I1R0 \include\ComStack_Types.h
- ..\Base_TS_T40D2M10I1R0 \include\MemMap.h
- ..\Base_TS_T40D2M10I1R0 \include\Mcal.h
- ..\Base_TS_T40D2M10I1R0 \include\Platform_Types.h
- ..\Base_TS_T40D2M10I1R0 \include\Std_Types.h
- ..\Base_TS_T40D2M10I1R0 \include\Reg_eSys.h
- ..\Base_TS_T40D2M10I1R0 \include\Soc_Ips.h
- ..\Base_TS_T40D2M10I1R0 \include\SilRegMacros.h

Files from Rte folder:

- ..\Rte_TS_T40D2M10I1R0 \include\SchM_Pwm.h

Files from Mcl folder:

- ..\Mcl_TS_T40D2M10I1R0 \include\Ftm_Common.c
- ..\Mcl_TS_T40D2M10I1R0 \include\Ftm_Common_Types.h
- ..\Mcl_TS_T40D2M10I1R0 \include\Reg_eSys_Ftm.h

Files from Det folder:

- ..\Det_TS_T40D2M10I1R0 \include\Det.h

3.3 Setting up the Plug-ins

All the Autosar MCAL drivers for S32K14X were designed to be configured using Tresos Studio configuration and code generation tool from EB tresos Studio 23.0.0 b170330-0431.

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_T40D2M10I1R0\config\Pwm.xdm
  - ..\Mcu_TS_T40D2M10I1R0\config\Mcu.xdm
  - ..\Mcl_TS_T40D2M10I1R0\config\Mcl.xdm
  - ..\Resource_TS_T40D2M10I1R0\config\Resource.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\PWM_TS_T40D2M10I1R0\autosar\Pwm_<subderivative_name>.epd
  - ..\Mcu_TS_T40D2M10I1R0\autosar\Mcu.epd
  - ..\Mcl_TS_T40D2M10I1R0\autosar\Mcl.epd
  - ..\Resource_TS_T40D2M10I1R0\autosar\Resource_<subderivative_name>.epd

- Code Generation Templates for parameters without variation points:
  - ..\PWM_TS_T40D2M10I1R0\output\src\Pwm_Cfg.c
  - ..\PWM_TS_T40D2M10I1R0\output\include\Pwm_Cfg.h
  - ..\Mcu_TS_T40D2M10I1R0\output\include\Mcu_Cfg.h
  - ..\Mcl_TS_T40D2M10I1R0\output\include\Mcl_Cfg.h
- Code Generation Templates for variant aware parameters:
  - ..\PWM_TS_T40D2M10I1R0\output\src\Pwm_PBCfg.c
  - ..\PWM_TS_T40D2M10I1R0\output\include\Pwm_Cfg.h
  - ..\Mcu_TS_T40D2M10I1R0\output\include\Mcu_Cfg.h
  - ..\Mcl_TS_T40D2M10I1R0\output\include\Mcl_Cfg.h

Steps to generate the configuration:

1. Copy the module folders PWM_TS_T40D2M10I1R0 , Dem_TS_T40D2M10I1R0 , Base_TS_T40D2M10I1R0 , Resource_TS_T40D2M10I1R0 , Mcu_TS_T40D2M10I1R0 , Mcl_TS_T40D2M10I1R0, EcuC_TS_T40D2M10I1R0into 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** : Resource module is used to select microcontroller's derivatives .
- **MCU** : The MCU driver provides services for basic microcontroller initialization, power down functionality, reset and microcontroller specific functions required by other MCAL software modules. The clocks need to be initialized prior to using the MDL driver.
- **MCL** : The MCL module is required for is required for support for PWM
- **RTE** : The RTE module is required for critical sections
- **DET** : The DET module is used for enabling Development error detection. The API function used is Det_ReportError(). The activation / deactivation of Development error detection is configurable using the 'MdlDevErrorDetect' configuration parameter

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).

NOTE

For proper driver usage, prior MCU and PORT modules initialization should be done.

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 initialization-deinitialization strategy deployed by user.

Chapter 5
Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

There are some resources are shared for some functions in Pwm_Ftm.c. They are protected under Exclusive Areas.

In function Pwm_Ftm_SetPeriodAndDuty: **PWM_EXCLUSIVE_AREA_00**

In function Pwm_Ftm_SetDutyCycle: **PWM_EXCLUSIVE_AREA_01**

In function Pwm_Ftm_SetOutputToIdle: **PWM_EXCLUSIVE_AREA_02**

In function Pwm_Ftm_DisableNotification: **PWM_EXCLUSIVE_AREA_03**

In function Pwm_Ftm_EnableNotification: **PWM_EXCLUSIVE_AREA_04**

In function Pwm_Ftm_ForceOutputToZero: **PWM_EXCLUSIVE_AREA_05**

In function Pwm_Ftm_SetClockMode: **PWM_EXCLUSIVE_AREA_06**

In function Pwm_Ftm_SelectCommonTimebase: **PWM_EXCLUSIVE_AREA_07**

In function Pwm_Ftm_SyncUpdate: **PWM_EXCLUSIVE_AREA_08**

In function Pwm_Ftm_SetPhaseShift: **PWM_EXCLUSIVE_AREA_09**

In function Pwm_Ftm_EnableTrigger: **PWM_EXCLUSIVE_AREA_10**

In function Pwm_Ftm_DisableTrigger: **PWM_EXCLUSIVE_AREA_11**

In function Pwm_Ftm_MaskOutputs: **PWM_EXCLUSIVE_AREA_12**

In function Pwm_Ftm_UnMaskOutputs: **PWM_EXCLUSIVE_AREA_13**

Table 5-1. Exclusive Areas in Pwm_Ftm.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	PWM_EXCLUSIVE_AREA_10	PWM_EXCLUSIVE_AREA_11	PWM_EXCLUSIVE_AREA_12	PWM_EXCLUSIVE_AREA_13	Interrupt Service Routines Critical Regions(composed diagram)
PWM_EXCLUSIVE_AREA_00	X	X	X	X	X	X	X	X	X	X			X	X	
PWM_EXCLUSIVE_AREA_01	X	X	X	X	X	X	X	X	X	X			X	X	
PWM_EXCLUSIVE_AREA_02	X	X	X	X	X	X	X	X	X	X			X	X	
PWM_EXCLUSIVE_AREA_03	X	X	X	X	X	X	X	X							
PWM_EXCLUSIVE_AREA_04	X	X	X	X	X	X	X	X							
PWM_EXCLUSIVE_AREA_05	X	X	X	X	X	X	X	X					X	X	

Table continues on the next page...

Table 5-1. Exclusive Areas in Pwm_Ftm.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	PWM_EXCLUSIVE_AREA_10	PWM_EXCLUSIVE_AREA_11	PWM_EXCLUSIVE_AREA_12	PWM_EXCLUSIVE_AREA_13	Interrupt Service Routines Critical Regions(composed diagram)
PWM_EXCLUSIVE_AREA_06	X	X	X	X	X	X	X	X							
PWM_EXCLUSIVE_AREA_07	X	X	X	X	X	X	X	X							
PWM_EXCLUSIVE_AREA_08	X	X	X						X	X			X	X	
PWM_EXCLUSIVE_AREA_09	X	X	X						X	X			X	X	
PWM_EXCLUSIVE_AREA_10											X	X			
PWM_EXCLUSIVE_AREA_11											X	X			
PWM_EXCLUSIV	X	X	X			X			X	X			X	X	

Table continues on the next page...

Table 5-1. Exclusive Areas in Pwm_Ftm.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	PWM_EXCLUSIVE_AREA_10	PWM_EXCLUSIVE_AREA_11	PWM_EXCLUSIVE_AREA_12	PWM_EXCLUSIVE_AREA_13	Interrupt Service Routines Critical Regions(composed diagram)
E_AREA_12															
PWM_EXCLUSIVE_AREA_13	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 S32K14X controllers, PWM functionality is provided by the FTM module.

Refer Table PWM Hardware Channel availability for S32K14X family in User Manual for more details.

5.3 ISR to configure within OS – dependencies

The following ISR's are used for S32K11X derivatives by the PWM driver:

Table 5-2. PWM ISR's

ISR Name	CM0 Hardware interrupt vector
For FTM_0	
ISR(FTM_0_ISR)	12
ISR(FTM_0_FAULT_ISR)	13
ISR(FTM_0_OVF_ISR)	14
For FTM_1	
ISR(FTM_1_ISR)	15
ISR(FTM_1_FAULT_ISR)	16
ISR(FTM_1_OVF_ISR)	17

The following ISR's are used for S32K14X derivatives by the PWM driver:

Table 5-3. PWM ISR's

ISR Name	CM4 Hardware interrupt vector
For FTM_0	
ISR(FTM_0_CH_0_CH_1_ISR)	115
ISR(FTM_0_CH_2_CH_3_ISR)	116
ISR(FTM_0_CH_4_CH_5_ISR)	117
ISR(FTM_0_CH_6_CH_7_ISR)	118
ISR(FTM_0_FAULT_ISR)	119
ISR(FTM_0_OVF_ISR)	120
For FTM_1	
ISR(FTM_1_CH_0_CH_1_ISR)	121
ISR(FTM_1_CH_2_CH_3_ISR)	122
ISR(FTM_1_CH_4_CH_5_ISR)	123
ISR(FTM_1_CH_6_CH_7_ISR)	124
ISR(FTM_1_FAULT_ISR)	125
ISR(FTM_1_OVF_ISR)	126

Table continues on the next page...

Table 5-3. PWM ISR's (continued)

ISR Name	CM4 Hardware interrupt vector
For FTM_2	
ISR(FTM_2_CH_0_CH_1_ISR)	127
ISR(FTM_2_CH_2_CH_3_ISR)	128
ISR(FTM_2_CH_4_CH_5_ISR)	129
ISR(FTM_2_CH_6_CH_7_ISR)	130
ISR(FTM_2_FAULT_ISR)	131
ISR(FTM_2_OVF_ISR)	132
For FTM_3	
ISR(FTM_3_CH_0_CH_1_ISR)	133
ISR(FTM_3_CH_2_CH_3_ISR)	134
ISR(FTM_3_CH_4_CH_5_ISR)	135
ISR(FTM_3_CH_6_CH_7_ISR)	136
ISR(FTM_3_FAULT_ISR)	137
ISR(FTM_3_OVF_ISR)	138
For FTM_4	
ISR(FTM_4_CH_0_CH_1_ISR)	139
ISR(FTM_4_CH_2_CH_3_ISR)	140
ISR(FTM_4_CH_4_CH_5_ISR)	141
ISR(FTM_4_CH_6_CH_7_ISR)	142
ISR(FTM_4_FAULT_ISR)	143
ISR(FTM_4_OVF_ISR)	144
For FTM_5	
ISR(FTM_5_CH_0_CH_1_ISR)	145
ISR(FTM_5_CH_2_CH_3_ISR)	146
ISR(FTM_5_CH_4_CH_5_ISR)	147
ISR(FTM_5_CH_6_CH_7_ISR)	148
ISR(FTM_5_FAULT_ISR)	149
ISR(FTM_5_OVF_ISR)	150
For FTM_6	
ISR(FTM_6_CH_0_CH_1_ISR)	151
ISR(FTM_6_CH_2_CH_3_ISR)	152
ISR(FTM_6_CH_4_CH_5_ISR)	153
ISR(FTM_6_CH_6_CH_7_ISR)	154
ISR(FTM_6_FAULT_ISR)	155
ISR(FTM_6_OVF_ISR)	156
For FTM_7	
ISR(FTM_7_CH_0_CH_1_ISR)	157
ISR(FTM_7_CH_2_CH_3_ISR)	158
ISR(FTM_7_CH_4_CH_5_ISR)	159

Table continues on the next page...

Table 5-3. PWM ISR's (continued)

ISR Name	CM4 Hardware interrupt vector
ISR(FTM_7_CH_6_CH_7_ISR)	160
ISR(FTM_7_FAULT_ISR)	161
ISR(FTM_7_OVF_ISR)	162

NOTE

FTM_x_CH_n_CH_(n+1)_ISR or FTM_x_ISR is used to handle leading edge notification and FTM_x_OVF_ISR is used to handle trailing edge notification. Depend on polarity, the leading edge/trailing edge can be either PWM_RISING_EDGE or PWM_FALLING_EDGE. For example, if FTM_1_CH_3 is configured with polarity HIGH and notification is used with PWM_RISING_EDGE, then FTM_1_OVF_ISR need to be used.s

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 FTM channels which are used by the Pwm driver.

ECUC:

ECUC is required for configuring the variant handling in Tresos.

Configuration dependency to other module:

Chapter 6
Main API Requirements

6.1 Main functions calls within BSW scheduler

None

6.2 API Requirements

None.

6.3 Calls to Notification Functions, Callbacks, Callouts

Call-back Notifications:

None.

User Notification:

PWM driver does not provide standard notification that are raised at edge of PWM pulse. There are Reload Notifications that might take considering for an alternative solution.

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.

An extern declaration of the notification functions is available in Pwm_Cfg.h. 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
	PwmForceOutputToZeroApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmSetDutyCycle_NoUpdate	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmSetPeriodAndDuty_NoUpdate	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmSetPhaseShift	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
	PwmSetPhaseShiftNoUpdate	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmEnableTrigger	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmDiableTrigger	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmSwResetCounter	Pre Compile parameter for all Variants of Configuration	Pre Compile
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
	PwmEnableDualClockMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmCommonTimebaseSupported	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmPeriodUpdatedEndperiod	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmEnablePhaseShift	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmIndex	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmFaultSupport	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmFtmEnableExtTrigger	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmEnablePhaseShift	Pre Compile parameter for all Variants of Configuration	Pre Compile
	PwmMultiChannelSynch	Pre Compile parameter for all Variants of Configuration	Pre Compile
	EnableMaskingOperations	Pre Compile parameter for all Variants of Configuration	Pre Compile
PwmChannelConfigSet/ PwmChannel	PwmChannelId	VariantPC or VariantPB	Pre Compile or Post Build
	PwmFtmChannel	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

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
PwmChannelConfigSet/ PwmFtmModule	FtmModule	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
	PwmMcuClockReferencePoint	VariantPC or VariantPB	Pre Compile or Post Build
	PwmClockSelection	VariantPC or VariantPB	Pre Compile or Post Build
	PwmChanEdgeAlignment	VariantPC or VariantPB	Pre Compile or Post Build
	PwmUpdatedEndPeriod	VariantPC or VariantPB	Pre Compile or Post Build
	PwmUpdatedMiddlePeriod	VariantPC or VariantPB	Pre Compile or Post Build
	PwmReloadFrequency	VariantPC or VariantPB	Pre Compile or Post Build
	PwmDeadTimeCount	VariantPC or VariantPB	Pre Compile or Post Build
	DeadTimePrescaler	VariantPC or VariantPB	Pre Compile or Post Build
	PwmBDMEnable	VariantPC or VariantPB	Pre Compile or Post Build
	PwmFtmFaultFunctionality	VariantPC or VariantPB	Pre Compile or Post Build
PwmChannelConfigSet/ PwmFtmModule/ PwmFtmChannelFaultSettings	PwmFilterValue	VariantPC or VariantPB	Pre Compile or Post Build
	PwmDisableOutputOnFault0	VariantPC or VariantPB	Pre Compile or Post Build
	PwmDisableOutputOnFault1	VariantPC or VariantPB	Pre Compile or Post Build
	PwmDisableOutputOnFault2	VariantPC or VariantPB	Pre Compile or Post Build
	PwmDisableOutputOnFault3	VariantPC or VariantPB	Pre Compile or Post Build
	PwmFtmFaultOutputState	VariantPC or VariantPB	Pre Compile or Post Build
PwmChannelConfigSet/ PwmFtmModule/ PwmFtmChannelFaultSettings/ PwmFtmChannelFault0Settings	PwmFault0Polarity	VariantPC or VariantPB	Pre Compile or Post Build
	PwmEnableFault0Filter	VariantPC or VariantPB	Pre Compile or Post Build
	PwmFault0Notification	VariantPC or VariantPB	Pre Compile or Post Build
PwmChannelConfigSet/ PwmFtmModule/ PwmFtmChannelFaultSettings/ PwmFtmChannelFault1Settings	PwmFault1Polarity	VariantPC or VariantPB	Pre Compile or Post Build
	PwmEnableFault1Filter	VariantPC or VariantPB	Pre Compile or Post Build
	PwmFault1Notification	VariantPC or VariantPB	Pre Compile or Post Build
PwmChannelConfigSet/ PwmFtmModule/ PwmFtmChannelFaultSettings/ PwmFtmChannelFault2Settings	PwmFault2Polarity	VariantPC or VariantPB	Pre Compile or Post Build
	PwmEnableFault2Filter	VariantPC or VariantPB	Pre Compile or Post Build
	PwmFault2Notification	VariantPC or VariantPB	Pre Compile or Post Build
PwmChannelConfigSet/ PwmFtmModule/ PwmFtmChannelFaultSettings/ PwmFtmChannelFault3Settings	PwmFault3Polarity	VariantPC or VariantPB	Pre Compile or Post Build
	PwmEnableFault3Filter	VariantPC or VariantPB	Pre Compile or Post Build
	PwmFault3Notification	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/ PwmFtmModule/ PwmExternalTriggerSettings	PwmEnableExternalTriggerChannel0	VariantPC or VariantPB	Pre Compile or Post Build
	PwmEnableExternalTriggerChannel1	VariantPC or VariantPB	Pre Compile or Post Build
	PwmEnableExternalTriggerChannel2	VariantPC or VariantPB	Pre Compile or Post Build
	PwmEnableExternalTriggerChannel3	VariantPC or VariantPB	Pre Compile or Post Build
	PwmEnableExternalTriggerChannel4	VariantPC or VariantPB	Pre Compile or Post Build
	PwmEnableExternalTriggerChannel5	VariantPC or VariantPB	Pre Compile or Post Build
	PwmEnableExternalTriggerChannel6	VariantPC or VariantPB	Pre Compile or Post Build
	PwmEnableExternalTriggerChannel7	VariantPC or VariantPB	Pre Compile or Post Build
	PwmEnableInitializationTrigger	VariantPC or VariantPB	Pre Compile or Post Build
PwmChannelConfigSet/ PwmFtmModule/ PwmFtmChannels	PwmFtmChannelId	VariantPC or VariantPB	Pre Compile or Post Build
	ChannelEdgeSetup	VariantPC or VariantPB	Pre Compile or Post Build
	ChannelCombDeadTimeEn	VariantPC or VariantPB	Pre Compile or Post Build
	PwmPhaseShift	VariantPC or VariantPB	Pre Compile or Post Build
CommonPublishedInformation	ArReleaseMajorVersion	VariantPC or VariantPB	Pre Compile or Post Build
	ArReleaseMinorVersion	VariantPC or VariantPB	Pre Compile or Post Build
	ArReleaseRevisionVersion	VariantPC or VariantPB	Pre Compile or Post Build
	ModuleId	VariantPC or VariantPB	Pre Compile or Post Build
	SwMajorVersion	VariantPC or VariantPB	Pre Compile or Post Build
	SwMinorVersion	VariantPC or VariantPB	Pre Compile or Post Build
	SwPatchVersion	VariantPC or VariantPB	Pre Compile or Post Build
	VendorApiInfix	VariantPC or VariantPB	Pre Compile or Post Build
	VendorId	VariantPC or VariantPB	Pre Compile or Post Build

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 Sections to be defined in MemMap.h
- 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.

[SMCAL_CPR_EXT163]

<< If interrupts are locked a centralized function pair to lock and unlock interrupts shall be used. >>

[SMCAL_CPR_EXT166]

<< When PWM read-back is deployed using ICU, in order to perform the PWM readback, the integrator shall use the following configuration: a given PWM channel on one PWM Hw module shall be only readback on an ICU channel implemented using another Hw module >>

NOTE

Implementation hints: the configuration can be realized using the safety PORT module

[SWS_Pwm_00001]

<< The Pwm SWS does not cover PWM emulation on general purpose I/O. >>

[SWS_Pwm_00089]

<< 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.?() >>

[SWS_Pwm_00093]

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

[SWS_Pwm_00116]

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

[SWS_Pwm_10086]

<< 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. >>

[SWS_Pwm_20086]

<< 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. >>

[SWS_Pwm_00119]

<< 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 IM2PWMASR4.2 Rev0002 R1.0.1
Revision 1.0

AUTOSAR