Integration Manual

for S32K14X GPT Driver

Document Number: IM2GPTASR4.2 Rev0002 R1.0.2
Rev. 1.0

Contents

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

External Assumptions for GPT driver

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	26/04/2019	NXP MCAL Team	Updated version for ASR 4.2.2S32K14XR1.0.2

Chapter 2
Introduction

This User Manual describes NXP Semiconductors AUTOSAR General Purpose Timer (GPT) for S32K14X .

AUTOSAR GPT driver configuration parameters and deviations from the specification are described in GPT Driver chapter of this document. AUTOSAR GPT driver requirements and APIs are described in the AUTOSAR GPT driver software specification document.

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, s32k142_lqfp48, s32k144_lqfp48, s32k148_lqfp100

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
API	Application Programming Interface
AUTOSAR	AUTomotive Open System ARchitecture
ASM	Assembler
BSMI	Basic Software Makefile Interface
C/CPP	C and C++ Source Code
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ETIMER	Enhanced Motor Control Timer
GPT	General Purpose Timer

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
ISR	Interrupt Service Routine
MCU	Micro Controller Unit
N/A	Not Applicable
LPIT	Low Power Interrupt Timer
LPTMR	Low Power Timer
RTC	Real Time Clock
FTM	FlexTimer Module

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	Specification of GPT Driver	AUTOSAR Release 4.2.2
2	S32K14X Reference Manual	Reference Manual, Rev. 9, 9/2018
3	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	30/11/2017
4	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	30/11/2017
5	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	30/11/2017
6	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	25/10/2018
7	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	07/01/2019

Chapter 3
Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar GPT driver for NXP SemiconductorsS32K14X . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The GPT 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 (Wed Jan 24 16:21:45 CST 2018 build.sh rev=g27a1317 s=L631 Earmv7 -V release_g27a1317_build_Fed_Earmv7)
- IAR: V8.11.2

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

Note

The TS_T40D2M10I2R0 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
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-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
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-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=Cortex-M0+	Selects target processor: Arm Cortex M0+
--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.

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
--diag_suppress=Pa050	Suppresses diagnostic messages (warnings) about non-standard line endings.
-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=Cortex-M0+	Selects target processor: Arm Cortex M0+
--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
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--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.
-Os	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.
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
-ansi	Specifies ANSI C with extensions.
-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.
-msoft-float	Use software 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.
-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..
-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
-DGCC	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GCC preprocessor symbol.

Table 3-8. Assembler Options

Option	Description
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-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 Autosar GPT driver for NXP Semiconductors's S32K14X microcontroller.

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.

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.

Plugin Files:

com.freescale.tools.tresos.xpath.jar - This plugin file is necessary to compile the GPT module. It will be installed in the EB Tresos plugins folder.

GPT Files:

..\Gpt_TS_T40D2M10I2R0\src\Gpt.c

..\Gpt_TS_T40D2M10I2R0\src\Gpt_Ipw.c

..\Gpt_TS_T40D2M10I2R0\src\Gpt_LPit.c

..\Gpt_TS_T40D2M10I2R0\src\Gpt_Lptmr.c

..\Gpt_TS_T40D2M10I2R0\src\Gpt_SRtc.c

..\Gpt_TS_T40D2M10I2R0\src\Gpt_Ftm.c

..\Gpt_TS_T40D2M10I2R0\include\Gpt.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_EnvCfg.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_Ftm.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_Ftm_Types.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_Ipw.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_Ipw_Irq.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_Ipw_Types.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_Irq.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_LPit.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_LPit_Types.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_Lptmr.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_Lptmr_Types.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_SRtc.h

..\Gpt_TS_T40D2M10I2R0\include\Gpt_SRtc_Types.h

..\Gpt_TS_T40D2M10I2R0\include\Reg_eSys_SRtc.h

GPT Generated Files:

- Gpt_Cfg.h
- Gpt_VS_<VariantNo>_PBcfg.c

For driver compilation, Gpt_VS_0_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:

- Gpt_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)
- Gpt_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 Gpt_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.

Other include files:

Files from Base folder:

..\Base_TS_T40D2M10I2R0\include\Gpt_MemMap.h

..\Base_ TS_T40D2M10I2R0\include\Platform_Types.h

..\Base_TS_T40D2M10I2R0\include\Std_Types.h

..\Base_TS_T40D2M10I2R0\include\Soc_Ips.h

..\Base_TS_T40D2M10I2R0\include\Reg_eSys.h

..\Base_TS_T40D2M10I2R0\include\SilRegMacros.h

..\Base_TS_T40D2M10I2R0\include\Compiler.h

..\Base_ TS_T40D2M10I2R0\include\Mcal.h

Files from Det folder:

..\Det_TS_T40D2M10I2R0\include\Det.h

Files from Mcl folder:

..\Mcl_TS_T40D2M10I2R0\src\Ftm_Common.c

..\Mcl_TS_T40D2M10I2R0\src\Lpit_Common.c

..\Mcl_TS_T40D2M10I2R0\src\Lptmr_Common.c

..\Mcl_TS_T40D2M10I2R0\include\Ftm_Common.h

..\Mcl_TS_T40D2M10I2R0\include\Reg_eSys_Ftm.h

..\Mcl_TS_T40D2M10I2R0\include\Reg_eSys_Lpit.h

..\Mcl_TS_T40D2M10I2R0\include\Reg_eSys_Lptmr.h

Files from EcuM folder:

..\EcuM_TS_T40D2M10I2R0\include\EcuM.h

..\EcuM_TS_T40D2M10I2R0\include\EcuM_Cbk.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.

Module Parameter Definition File:

..\Gpt_TS_T40D2M10I2R0 \config\Gpt.xdm

..\EcuM_TS_T40D2M10I2R0\config\EcuM.xdm

Code Generation Templates for Pre-Compile time configuration parameters:

..\Gpt_TS_T40D2M10I2R0 \generate_PC\src\Gpt_Cfg.c

..\Gpt_TS_T40D2M10I2R0\generate_PC\include\Gpt_Cfg.h

..\EcuM_TS_T40D2M10I2R0\generate_PC\include\EcuM_Cfg.h

Code Generation Templates for Post-Build time configuration parameters:

..\Gpt_TS_T40D2M10I2R0\generate_PB\src\Gpt_PBcfg.c

Tresos Configuration tool files:

..\Gpt_TS_T40D2M10I2R0\META-INF\ MANIFEST.MF

..\Gpt_TS_T40D2M10I2R0\plugin.xml

Steps to generate configurations:

1.Copy the module folder (Gpt_TS_T40D2M10I2R0), (Base_TS_T40D2M10I2R0), (EcuM_TS_T40D2M10I2R0), (EcuC_TS_T40D2M10I2R0), (Mcu_TS_T40D2M10I2R0), (Resource_TS_T40D2M10I2R0) into the Tresos plugins folder.

2.Set the desired Tresos Output location folder for the generated sources and header files.

3.Use the Tresos GUI to modify configuration parameters values.

4.Generate the Pre-Compile and Post-Build files.

Chapter 4
Function calls to module

4.1 Function Calls during Start-up

GPT shall be initialized during STARTUP1 phase of EcuM initialization. The API to be called for this is Gpt_Init() MCU module shall be initialized before GPT is initialized.

4.2 Function Calls during Shutdown

If GptWakeupFunctionalityApi and GptWakeupSourceRef are enabled, Gpt_SetMode(GPT_MODE_SLEEP) API shall be called during GO SLEEP phase of EcuM to configure the hardware for Sleep mode.

4.3 Function Calls during Wake-up

For the platforms where the GPT driver controls wakeup hw sources, if the GptWakeupFunctionalityApi and GptWakeupSourceRef are enabled, the driver shall report the wakeup event to EcuM through EcuM_SetWakeupEvent(Source) upon the hw source event.

Chapter 5
Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

GPT_EXCLUSIVE_AREA_00 Used in Gpt_Lptmr_StartTimer function to protect the updates to:

- LPTMR_CSR_ADDR32 register

GPT_EXCLUSIVE_AREA_01 Used in Gpt_Lptmr_StopTimer function to protect the updates to:

- LPTMR_CSR_ADDR32 register

GPT_EXCLUSIVE_AREA_02 Used in Gpt_Lptmr_EnableInterrupt function to protect the updates to:

- LPTMR_CSR_ADDR32 register

GPT_EXCLUSIVE_AREA_03 Used in Gpt_Lptmr_SetClockMode function to protect the updates to:

- LPTMR_CSR_ADDR32 register

GPT_EXCLUSIVE_AREA_04 Used in Gpt_Lptmr_StartPredefTimer function to protect the updates to:

- LPTMR_CSR_ADDR32 register

GPT_EXCLUSIVE_AREA_05 Used in Gpt_Lptmr_StopPredefTimer function to protect the updates to:

- LPTMR_CSR_ADDR32 register

GPT_EXCLUSIVE_AREA_06 Used in Gpt_Lptmr_DisableInterrupt function to protect the updates to:

- LPTMR_CSR_ADDR32 register

GPT_EXCLUSIVE_AREA_09 Used in Gpt_Rtc_StartTimer function to protect the updates to:

- RTC_CTRL_ADDR32 register

GPT_EXCLUSIVE_AREA_10 Used in Gpt_Rtc_StopTimer function to protect the updates to:

- RTC_CTRL_ADDR32 register

GPT_EXCLUSIVE_AREA_11 Used in Gpt_Rtc_SetClockMode function to protect the updates to:

- RTC_CTRL_ADDR32 register

GPT_EXCLUSIVE_AREA_12 Used in Gpt_Rtc_EnableInterrupt function to protect the updates to:

- RTC_CTRL_ADDR32 register

GPT_EXCLUSIVE_AREA_13 Used in Gpt_Rtc_DisableInterrupt function to protect the updates to:

- RTC_CTRL_ADDR32 register

GPT_EXCLUSIVE_AREA_07 Used in Gpt_LPit_ProcessCommonInterrupt function to protect the updates to:

- LPIT_MSR_ADDR32 register

GPT_EXCLUSIVE_AREA_08 Used in Gpt_LPit_StartTimer function to protect the updates to:

- LPIT_TCTRL_ADDR32 register

GPT_EXCLUSIVE_AREA_14 Used in Gpt_LPit_StopTimer function to protect the updates to:

- LPIT_MSR_ADDR32 register
- LPIT_TCTRL_ADDR32 register

GPT_EXCLUSIVE_AREA_15 Used in Gpt_LPit_EnableInterrupt function to protect the updates to:

- LPIT_MSR_ADDR32 register
- LPIT_MIER_ADDR32 register

GPT_EXCLUSIVE_AREA_16 Used in Gpt_LPit_DisableInterrupt function to protect the updates to:

- LPIT_MSR_ADDR32 register
- LPIT_MIER_ADDR32 register

GPT_EXCLUSIVE_AREA_17 Used in Gpt_Ftm_ProcessCommonInterrupt function to protect the updates to:

- FTM_CSC_ADDR32 register

GPT_EXCLUSIVE_AREA_18 Used in Gpt_Ftm_StartTimer function to protect the updates to:
- FTM_CSC_ADDR32 register

GPT_EXCLUSIVE_AREA_19 Used in Gpt_Ftm_GetTimeElapsed function to protect the updates to:
- FTM_CSC_ADDR32 register

GPT_EXCLUSIVE_AREA_20 Used in Gpt_Ftm_SetClockMode function to protect the updates to:
- FTM_SC_ADDR32 register

GPT_EXCLUSIVE_AREA_21 Used in Gpt_Ftm_StartPredefTimer function to protect the updates to:
- FTM_CSC_ADDR32 register
- FTM_SC_ADDR32 register
- FTM_MODE_ADDR32 register
- FTM_CONF_ADDR32 register

GPT_EXCLUSIVE_AREA_22 Used in Gpt_Ftm_StopPredefTimer function to protect the updates to:
- FTM_CSC_ADDR32 register
- FTM_SC_ADDR32 register
- FTM_MODE_ADDR32 register
- FTM_CONF_ADDR32 register

GPT_EXCLUSIVE_AREA_23 Used in Gpt_SRtc_SetUserAccessAllowed function to protect the updates to:
- SRTC_CR_ADDR32 register

GPT_EXCLUSIVE_AREA_24 Used in Gpt_SRtc_StartTimer function to protect the updates to:
- SRTC_SR_ADDR32 register
- SRTC_IER_ADDR32 register

GPT_EXCLUSIVE_AREA_25 Used in Gpt_SRtc_StopTimer function to protect the updates to:
- SRTC_SR_ADDR32 register
- SRTC_IER_ADDR32 register

GPT_EXCLUSIVE_AREA_26 Used in Gpt_SRtc_EnableInterrupt function to protect the updates to:
- SRTC_IER_ADDR32 register

GPT_EXCLUSIVE_AREA_27 Used in Gpt_SRtc_DisableInterrupt function to protect the updates to:

- SRTC_IER_ADDR32 register

GPT_EXCLUSIVE_AREA_28 Used in ISR(Gpt_SRTC_0_Ch_0_ISR) function to protect the updates to:

- SRTC_CR_ADDR32 register
- SRTC_SR_ADDR32 register

GPT_EXCLUSIVE_AREA_29 Used in Gpt_Lptmr_ProcessCommonInterrupt function to protect the updates to:

- LPTMR_CSR_ADDR32 register

Critical Region Exclusive Matrix

Below is the table depicting the exclusivity between different critical region IDs from the Gpt 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.

Table 5-1. Exclusive Areas

	GPT_EA_00	GPT_EA_01	GPT_EA_02	GPT_EA_03	GPT_EA_04	GPT_EA_05	GPT_EA_06	GPT_EA_09	GPT_EA_10	GPT_EA_11	GPT_EA_12	GPT_EA_13	GPT_EA_07	GPT_EA_08	GPT_EA_14	GPT_EA_15	GPT_EA_16	GPT_EA_17	GPT_EA_18	GPT_EA_19	GPT_EA_20	GPT_EA_21	GPT_EA_22	GPT_EA_23	GPT_EA_24	GPT_EA_25	GPT_EA_26	GPT_EA_27	GPT_EA_28	GPT_EA_29
GPT_EA_00	x	x	x	x	x	x	x																							x
GPT_EA_01	x	x	x	x	x	x	x																							x
GPT_	x	x	x	x	x	x	x																							x

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	GPT_EA_00	GPT_EA_01	GPT_EA_02	GPT_EA_03	GPT_EA_04	GPT_EA_05	GPT_EA_06	GPT_EA_09	GPT_EA_10	GPT_EA_11	GPT_EA_12	GPT_EA_13	GPT_EA_07	GPT_EA_08	GPT_EA_14	GPT_EA_15	GPT_EA_16	GPT_EA_17	GPT_EA_18	GPT_EA_19	GPT_EA_20	GPT_EA_21	GPT_EA_22	GPT_EA_23	GPT_EA_24	GPT_EA_25	GPT_EA_26	GPT_EA_27	GPT_EA_28	GPT_EA_29
EA_02																														
GPT_EA_03	x	x	x	x	x	x	x																							x
GPT_EA_04	x	x	x	x	x	x	x																							x
GPT_EA_05	x	x	x	x	x	x	x																							x
GPT_EA_06	x	x	x	x	x	x	x																							x
GPT_EA_09								x	x	x	x	x																		
GPT_EA_10								x	x	x	x	x																		
GPT_EA_11								x	x	x	x	x																		

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	GPT_EA_00	GPT_EA_01	GPT_EA_02	GPT_EA_03	GPT_EA_04	GPT_EA_05	GPT_EA_06	GPT_EA_09	GPT_EA_10	GPT_EA_11	GPT_EA_12	GPT_EA_13	GPT_EA_07	GPT_EA_08	GPT_EA_14	GPT_EA_15	GPT_EA_16	GPT_EA_17	GPT_EA_18	GPT_EA_19	GPT_EA_20	GPT_EA_21	GPT_EA_22	GPT_EA_23	GPT_EA_24	GPT_EA_25	GPT_EA_26	GPT_EA_27	GPT_EA_28	GPT_EA_29
GPT_EA_12								x	x	x	x	x																		
GPT_EA_13								x	x	x	x	x																		
GPT_EA_07													x		x	x	x													
GPT_EA_08													x	x	x	x	x													
GPT_EA_14													x	x	x	x	x													
GPT_EA_15													x		x	x	x													
GPT_EA_16													x		x	x	x													

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	GPT_EA_00	GPT_EA_01	GPT_EA_02	GPT_EA_03	GPT_EA_04	GPT_EA_05	GPT_EA_06	GPT_EA_09	GPT_EA_10	GPT_EA_11	GPT_EA_12	GPT_EA_13	GPT_EA_07	GPT_EA_08	GPT_EA_14	GPT_EA_15	GPT_EA_16	GPT_EA_17	GPT_EA_18	GPT_EA_19	GPT_EA_20	GPT_EA_21	GPT_EA_22	GPT_EA_23	GPT_EA_24	GPT_EA_25	GPT_EA_26	GPT_EA_27	GPT_EA_28	GPT_EA_29
GPT_EA_17																		x	x	x										
GPT_EA_18																		x	x	x										
GPT_EA_19																		x	x	x										
GPT_EA_20																					x	x	x							
GPT_EA_21																					x	x	x							
GPT_EA_22																					x	x	x							
GPT_EA_23																								x					x	

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	GPT_EA_00	GPT_EA_01	GPT_EA_02	GPT_EA_03	GPT_EA_04	GPT_EA_05	GPT_EA_06	GPT_EA_09	GPT_EA_10	GPT_EA_11	GPT_EA_12	GPT_EA_13	GPT_EA_07	GPT_EA_08	GPT_EA_14	GPT_EA_15	GPT_EA_16	GPT_EA_17	GPT_EA_18	GPT_EA_19	GPT_EA_20	GPT_EA_21	GPT_EA_22	GPT_EA_23	GPT_EA_24	GPT_EA_25	GPT_EA_26	GPT_EA_27	GPT_EA_28	GPT_EA_29
GPT_EA_24																									x	x	x	x	x	
GPT_EA_25																									x	x	x	x	x	
GPT_EA_26																									x	x	x	x		
GPT_EA_27																									x	x	x	x		
GPT_EA_28																								x					x	
GPT_EA_29	x	x	x	x	x	x																								x

Note

- **GPT_EA_xx** means **GPT_EXCLUSIVE_AREA_xx**

5.2 Peripheral Hardware Requirements

Driver implements the following channels on 4 S32K14X peripherals.

- 8 channels each are implemented on 4 or 8 FTM modules (derivative dependent).

- 4 channels are implemented on 1 LPIT module.

- 1 channel is implemented on 1 Lptmr module.

- 1 channel is implemented on 1 SRtc module.

Refer Table GPT Hardware Channel availability for S32K14X family in User Manual

5.3 ISR to configure within OS – dependencies

The following ISR's are used by the GPT driver:

Table 5-2. GPT ISR's S32K14X

ISR Name	Hardware interrupt vector
For FTM_0	
ISR(FTM_0_CH_0_CH_1_ISR)	99
ISR(FTM_0_CH_2_CH_3_ISR)	100
ISR(FTM_0_CH_4_CH_5_ISR)	101
ISR(FTM_0_CH_6_CH_7_ISR)	102
For FTM_1	
ISR(FTM_1_CH_0_CH_1_ISR)	105
ISR(FTM_1_CH_2_CH_3_ISR)	106
ISR(FTM_1_CH_4_CH_5_ISR)	107
ISR(FTM_1_CH_6_CH_7_ISR)	108
For FTM_2	
ISR(FTM_2_CH_0_CH_1_ISR)	111
ISR(FTM_2_CH_2_CH_3_ISR)	112
ISR(FTM_2_CH_4_CH_5_ISR)	113
ISR(FTM_2_CH_6_CH_7_ISR)	114
For FTM_3	
ISR(FTM_3_CH_0_CH_1_ISR)	117
ISR(FTM_3_CH_2_CH_3_ISR)	118
ISR(FTM_3_CH_4_CH_5_ISR)	119
ISR(FTM_3_CH_6_CH_7_ISR)	120
For FTM_4	
ISR(FTM_4_CH_0_CH_1_ISR)	123
ISR(FTM_4_CH_2_CH_3_ISR)	124

Table continues on the next page...

Table 5-2. GPT ISR's S32K14X (continued)

ISR Name	Hardware interrupt vector
ISR(FTM_4_CH_4_CH_5_ISR)	125
ISR(FTM_4_CH_6_CH_7_ISR)	126
For FTM_5	
ISR(FTM_5_CH_0_CH_1_ISR)	129
ISR(FTM_5_CH_2_CH_3_ISR)	130
ISR(FTM_5_CH_4_CH_5_ISR)	131
ISR(FTM_5_CH_6_CH_7_ISR)	132
For FTM_6	
ISR(FTM_6_CH_0_CH_1_ISR)	135
ISR(FTM_6_CH_2_CH_3_ISR)	136
ISR(FTM_6_CH_4_CH_5_ISR)	137
ISR(FTM_6_CH_6_CH_7_ISR)	138
For FTM_7	
ISR(FTM_7_CH_0_CH_1_ISR)	141
ISR(FTM_7_CH_2_CH_3_ISR)	142
ISR(FTM_7_CH_4_CH_5_ISR)	143
ISR(FTM_7_CH_6_CH_7_ISR)	144
For LPIT_0	
ISR(Gpt_LPIT_0_TIMER_0_ISR)	49
ISR(Gpt_LPIT_0_TIMER_1_ISR)	50
ISR(Gpt_LPIT_0_TIMER_2_ISR)	51
ISR(Gpt_LPIT_0_TIMER_3_ISR)	52
For LPTMR_0	
ISR(Gpt_LPTMR_0_CH_0_ISR)	58
For SRtc_0	
ISR(Gpt_SRtc_0_CH_0_ISR)	46

Table 5-3. GPT ISR's S32K11X

ISR Name	Hardware interrupt vector
For FTM_0	
ISR(FTM_0_ISR)	12
For FTM_1	
ISR(FTM_1_ISR)	15
For LPIT_0	
ISR(LPIT_0_ISR)	20
For LPTMR_0	
ISR(LPTMR_0_CH_0_ISR)	8
For SRtc_0	
ISR(SRTC_0_CH_0_ISR)	7

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:

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. NXP SemiconductorsOS 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.

5.5 Other AUTOSAR modules - Dependencies

5.5.1 Base:

The BASE module contains the common files/definitions needed by all MCAL modules.

5.5.2 Det:

The DET module is used for enabling Default error detection. The API function used is Det_ReportError(). The activation/deactivation of Default error detection is configurable using the ‘GptDevErrorDetect’ configuration parameter.

5.5.3 Dem:

The DEM module is used for enabling reporting of production relevant error status. The API function used is Dem_ReportErrorStatus().

5.5.4 EcuM:

This module is used for processing the Wakeup notifications of GPT. Whenever the module is in ‘Sleep’ mode and a wakeup event occurs on a wakeup capable channel, it is reported to EcuM by calling EcuM_SetWakeupEvent() API through the EcuM_CheckWakeup() API. This is configurable using the ‘GptReportWakeupSource’ configuration parameter.

5.5.5 Mcl:

MCL module shall be initialized before using GPT .This module is used to obtain the common interrupts sources.

5.5.6 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 GPT driver. This module is required for setting the global prescaler value and to set the system clock frequency.

5.5.7 EcuC:

The ECUC module is used for ECU configuration. MCAL modules need ECUC to retrieve the variant information.

5.5.8 Resource:

Resource module is used to select microcontroller's derivatives.

5.5.9 Configuration dependency to other module:

For generating configuration files of GPT, EcuM also is required as GPT refers to EcuM parameter. EcuM need to be configure first before generating configuration files of GPT.

Hence template files for EcuM are provided at

..\EcuM_TS_T40D2M10I2R0\autosar\EcuM.epd (Module Parameter Definition File – AUTOSAR Format)

..\EcuM_TS_T40D2M10I2R0\config\EcuM.xdm (Module Parameter Definition File – Tresos Format)

5.6 Data cache restriction

None

5.7 User Mode support

The Gpt module can be run from user mode if 'GptEnableUserModeSupport' is enabled in the configuration.

In this case, the Gpt module will set the SUP bit in SRTC_CR register of SRTC hw IP.

In addition to setting 'GptEnableUserModeSupport' in the configuration, the application shall call the following function as trusted function:

- **Gpt_SRtc_SetUserAccessAllowed()**

Chapter 6
Main API Requirements

6.1 Main functions calls within BSW scheduler

None

6.2 Main API Requirements

None.

6.3 Calls to notification functions, callbacks, callouts

6.3.1 Call-back Notifications:

There are no call-back notifications defined inside the GPT driver.

6.3.2 User Notification:

The GPT Driver provides a notification per channel that is called whenever the defined time period is over.

The notifications can be configured as pointers to user defined functions. If notification is not desired,

'NULL_PTR' shall be configured.

An example of the syntax of this function is as follows:

void Gpt_Notification_<channel>

(

void

)

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

Chapter 7
Memory Allocation

7.1 Sections to be defined in Gpt_MemMap.h

Tables descibe Sections to be defined in Gpt_MemMap.h:

Table 7-1. Sectionto be define

<Section name>	Tyep of section	Description
GPT_START_SEC_CONFIG_DATA_<ALIGNMENT>	Configuration Data	Start of Memory Section for Config Data.
GPT_STOP_SEC_CONFIG_DATA_<ALIGNMENT>	Configuration Data	End of Memory Section for Config Data.
GPT_START_SEC_CODE	Code	Start of memory Section for Code in Flash.
GPT_STOP_SEC_CODE	Code	Stop of memory Section for Code in Flash.
GPT_START_SEC_RAMCODE	Code	Start of memory Section for Code in Ram.
GPT_STOP_SEC_RAMCODE	Code	Stop of memory Section for Code in Ram.
GPT_START_SEC_VAR_< INIT_POLICY>_<ALIGNMENT>	Variables	Start of memory Section for Variables.
GPT_STOP_SEC_VAR_ <INIT_POLICY>_<ALIGNMENT>	Variables	Stop of memory Section for Variables.
GPT_START_SEC_CONST_<ALIGNMENT>	Constant data	Start of memory Section for Constant.
GPT_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 GPT_MemMap.h

Chapter 8
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
Gpt	IMPLEMENTATION_CONFIG_VARIANT	Pre Compile parameter for all Variants of Configuration	Pre Compile
GptChannelConfigSet/ GptChannelConfiguration	GptChannelId	VariantPC or VariantPB	VariantPB
	GptHwChannel	VariantPC or VariantPB	VariantPB
	GptChannelMode	VariantPC or VariantPB	VariantPB
	GptChannelTickFrequency	VariantPC or VariantPB	VariantPB
	GptFtmPrescaler	VariantPC or VariantPB	VariantPB
	GptFtmPrescaler_Alternate	VariantPC or VariantPB	VariantPB
	GptLptmrPrescaler	VariantPC or VariantPB	VariantPB
	GptLptmrPrescaler_Alternate	VariantPC or VariantPB	VariantPB
	GptChannelClkSrcRef	VariantPC or VariantPB	VariantPB
	GptFtmChannelClkSrc	VariantPC or VariantPB	VariantPB
	GptLptmrChannelClkSrc	VariantPC or VariantPB	VariantPB
	GptSRtcChannelClkSrc	VariantPC or VariantPB	VariantPB
	GptLPitEnReloadOnTrigger	VariantPC or VariantPB	VariantPB
	GptLPitEnStopOnInterrupt	VariantPC or VariantPB	VariantPB
	GptLPitEnStartOnTrigger	VariantPC or VariantPB	VariantPB
	GptLPitTriggerChannels	VariantPC or VariantPB	VariantPB
	GptLPitIsExternalTrigger	VariantPC or VariantPB	VariantPB
	GptChannelTickValueMax	VariantPC or VariantPB	VariantPB
	GptFreezeEnable	VariantPC or VariantPB	VariantPB
	GptEnableWakeup	VariantPC or VariantPB	VariantPB
	GptNotification	VariantPC or VariantPB	VariantPB
GptChannelConfigSet/ GptChannelConfiguration/ GptWakeupConfiguration	GptWakeupSourceRef	VariantPC or VariantPB	VariantPB
GptConfigurationOfOptApiServices	GptDeinitApi	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
	GptEnableDisableNotificationApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	GptTimeElapsedApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	GptTimeRemainingApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	GptVersionInfoApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	GptWakeupFunctionalityApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
GptDriverConfiguration	GptDevErrorDetect	Pre Compile parameter for all Variants of Configuration	Pre Compile
	GptReportWakeupSource	Pre Compile parameter for all Variants of Configuration	Pre Compile
	GptRegisterLocking	Pre Compile parameter for all Variants of Configuration	Pre Compile
GptDriverConfiguration/ GptClockReferencePoint	GptClockReference	Pre Compile parameter for all Variants of Configuration	Pre Compile
GptNonAUTOSAR	GptEnableDualClockMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
	GptChangeNextTimeoutValueApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	GptEnableUserModeSupport	Pre Compile parameter for all Variants of Configuration	Pre Compile
	GptEnableTriggers	Pre Compile parameter for all Variants of Configuration	Pre Compile
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 General Purpose Timer :

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

Chapter 10
ISR Reference

None.

Chapter 11
External Assumptions for GPT driver

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

[SMCAL_CPR_EXT40]

<< The application shall not preempt a channel related function (like starting/stoping a timer) by calling Gpt_SetMode() or Gpt_DeInit(). >>

[SMCAL_CPR_EXT41]

<< The application shall not preempt a GPT function working on a GPT channel by calling another GPT function targeting the same channel. >>

[SMCAL_CPR_EXT42]

<< The application must not concurrently call Gpt functions with one exception: GetVersionInfo only can get interrupted or may interrupt >>

NOTE

A transversal GPT functions are those functions addressing the entire set of channels, like Gpt_Init(), Gpt_DeInit(), Gpt_SetMode(), Gpt_PeriodicCheck(), ...

[SMCAL_CPR_EXT43]

<< The application shall not call any function of the GPT module before having called Gpt_Init. >>

[SMCAL_CPR_EXT44]

<< Wakeup enabled timers shall be started or stopped only when GPT driver is in GPT_MODE_NORMAL mode.The external application shall invoke Gpt_EnableWakeup() and Gpt_DisableWakeup() only when GPT driver is in GPT_MODE_NORMAL mode. >>

NOTE

If Gpt_EnableWakeup(), Gpt_DisableWakeup(), Gpt_StartTimer() and Gpt_StopTimer() are called while GPT is already in SLEEP mode, the GPT driver behavior is not guaranteed. Therefore any wakeup channel configuration shall be done before entering in sleep mode.

[SMCAL_CPR_EXT163]

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

[SMCAL_CPR_EXT176]

<< The integrator shall assure that Gpt_Init() and Gpt_DeInit() functions do not interrupt each other. >>

[SWS_Gpt_00353]

<< If the register can affect several hardware modules and if it is an I/O register it shall be initialized by the PORT driver. >>

NOTE

The GPT driver manages hardware which does not include input/output configurable pins.

[SWS_Gpt_00354]

<< If the register can affect several hardware modules and if it is not an I/O register it shall be initialized by the MCU driver. >>

NOTE

The requirement is implicitly fulfilled at MCU level, as the MCU shall initialize the clock tree used also by the GPT driver.

[SWS_Gpt_00355]

<< One-time writable registers that require initialization directly after reset shall be initialized by the startup code. >>

NOTE

to be traced at the sMCAL generic level; The Interrupt Controller shall be initialized by the integrating application before to start using the GPTdriver.

[SWS_Gpt_00356]

<< All other registers shall be initialized by the startup code. >>

NOTE

to be traced at the sMCAL generic level; The Interrupt Controller shall be initialized by the integrating application before to start using the GPTdriver.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number IM2GPTASR4.2 Rev0002 R1.0.2
Revision 1.0