Integration Manual

for MPC5634M GPT Driver

Document Number: IM14GPTASR3.0R2.0.0
Rev. 2.0

freescale™

Contents

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

Chapter 6
Main Api Requirements

Chapter 7
Memory Allocation

Chapter 8
Configuration Parameters

Chapter 9
Integration Steps

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	20.01.2011	Chandrakanth.P	Initial Version
2.0	28.11.2011	Hari SS (b10728)	Updated for RTM 2.0.0

Chapter 2
Introduction

This Integration Manual describes the integration requirements for Autosar GPT 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 GPT_main 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 PARAMETER CONSIDERATIONS

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 GPT 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 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
ASM	Assembler
BSMI	Basic Software Make file Interface
GPT	General Purpose Timer
DEM	Diagnostic Event Manager
DET	Development Error Tracer
C/CPP	C and C++ Source Code
VLE	Variable Length Encoding
N/A	Not Applicable
MCU	Micro Controller Unit
STM	System Timer Module
PIT	Periodic Interrupt Timer
RTC	Real Time Clock

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
API	Autonomous Periodic Interrupt

2.3 Reference List

Table 2-3. Reference List

#	Title	Version
1	AUTOSAR 3.0GPT 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 compilers, linker options used for building the Gpt driver. It also explains the EB tresos Studio plugin setup procedure.

3.1 Build Options

The driver files are compiled using

GHS 5.1.7

DIAB 5_8_0_02 wind00198363 20100511 123238

CW Version 4.3 build 182

COSMIC Software PPC C Cross Compiler V4.3.4 - 16 Nov 2011 - Win32-F

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

3.1.1 GHS Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-cpu=ppc563xm	Selects target processor: ppc563xm
-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.
--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

Table continues on the next page...

Table 3-1. Compiler Options (continued)

Option	Description
-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
-DGHS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GHS preprocessor symbol.
-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 3-2. Assembler Options

Option	Description
-cpu=ppc563xm	Selects target processor: ppc563xm

Table 3-3. Linker Options

Option	Description
-cpu=ppc563xm	Selects target processor: ppc563xm
-nostartfiles	Do not use Start files.
-vle	Enables VLE code generation
-linker_warnings	Display linker warnings

3.1.2 DIAB Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
-tPPCE200Z3VEG:simple	Sets target processor to PPCE200Z3, 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
-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
-DDIAB	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the DIAB preprocessor symbol.

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
-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

Table 3-5. Assembler Options

Option	Description
-tPPCE200Z3VEN:simple	Selects target processor: PPCE200Z3, 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
-tPPCE200Z3VEN:simple	Selects target processor: PPCE200Z3, 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

Table continues on the next page...

Table 3-7. Compiler Options (continued)

Option	Description
-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
-ir	Include the debug information
-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
-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
-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.
-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.

Table continues on the next page...

Table 3-10. Compiler Options (continued)

Option	Description
-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 Compilation

This section describes the include files required to compile, assemble (if assembler code) and link the Autosar Gpt driver for Freescale SemiconductorMPC5634M 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.

GPT Files:

..\Gpt_TS_T2D14M20I0R0\src\Gpt_NonASR.c

..\Gpt_TS_T2D14M20I0R0\src\eMIOS_Gpt_LLD.c

..\Gpt_TS_T2D14M20I0R0\src\Gpt.c

..\Gpt_TS_T2D14M20I0R0\src\Gpt_Irq.c

..\Gpt_TS_T2D14M20I0R0\src\Gpt_LLD.c

..\Gpt_TS_T2D14M20I0R0\src\PitRti_Gpt_LLD.c

..\Gpt_TS_T2D14M20I0R0\src\PitRti_Gpt_LLD_IRQ.c

..\Gpt_TS_T2D14M20I0R0\src\Reg_eSys_EMIOS_Gpt.c

..\Gpt_TS_T2D14M20I0R0\src\Reg_eSys_EMIOS_IRQ_Gpt.c

..\Gpt_TS_T2D14M20I0R0\src\Stm_Gpt_LLD.c

..\Gpt_TS_T2D14M20I0R0\src\Stm_Gpt_LLD_IRQ.c

..\Gpt_TS_T2D14M20I0R0\include\Gpt_NonASR.h

..\Gpt_TS_T2D14M20I0R0\include\eMIOS_Gpt_LLD.h

..\Gpt_TS_T2D14M20I0R0\include\eMIOS_Gpt_LLD_CfgEx.h

..\Gpt_TS_T2D14M20I0R0\include\Gpt.h

..\Gpt_TS_T2D14M20I0R0\include\Gpt_LLD.h

..\Gpt_TS_T2D14M20I0R0\include\PitRti_Gpt_LLD.h

..\Gpt_TS_T2D14M20I0R0\include\PitRti_Gpt_LLD_CfgEx.h

..\Gpt_TS_T2D14M20I0R0\include\Reg_eSys_EMIOS.h

..\Gpt_TS_T2D14M20I0R0\include\Reg_eSys_EMIOS_CfgEx.h

..\Gpt_TS_T2D14M20I0R0\include\Reg_eSys_PITRTI.h

..\Gpt_TS_T2D14M20I0R0\include\Reg_eSys_STM.h

..\Gpt_TS_T2D14M20I0R0\include\Stm_Gpt_LLD.h

..\Gpt_TS_T2D14M20I0R0\include\Stm_Gpt_LLD_CfgEx.h

Gpt_Cfg.c (For PC Variant) - This file should be generated by the user using a configuration tool for compilation

Gpt_PBCfg.c (For PB Variant) - This file should be generated by the user using a configuration tool for compilation

Gpt_Cfg.h - This file should be generated by the user using a configuration tool for compilation

Other include files:

Files from Base folder:

..\Base_TS_T2D14M20I0R0\include\MemMap.h

..\Base_ TS_T2D14M20I0R0\include\Platform_Types.h

..\Base_TS_T2D14M20I0R0\include\Std_Types.h

..\Base_TS_T2D14M20I0R0\include\Mcal.h

..\Base_TS_T2D14M20I0R0\include\Reg_eSys.h

..\Base_TS_T2D14M20I0R0\include\Reg_Macros.h

..\Base_TS_T2D14M20I0R0\include\Cer.h

..\Base_TS_T2D14M20I0R0\include\Compiler.h

..\Base_TS_T2D14M20I0R0\include\Compiler_Cfg.h

..\Base_TS_T2D14M20I0R0\include\ComStack_Types.h

..\Base_TS_T2D14M20I0R0\include\Soc_Ips.h

Files from Det folder:

..\Det_TS_T2D14M20I0R0\include\Det.h

Files from Dem folder:

..\Dem_TS_T2D14M20I0R0\include\Dem.h

Files from SchM folder:

..\SchM_TS_T2D14M20I0R0\include\SchM_Gpt.h

Files from EcuM folder:

..\EcuM_TS_T2D14M20I0R0\include\EcuM.h

..\EcuM_TS_T2D14M20I0R0\include\EcuM_Cbk.h

3.3 Setting up the Plug-ins

All the Autosar MCAL drivers for MPC5634M were designed to be configured using Tresos® Studio configuration and code generation tool from Tresos 2010a.sr4 20100415-release2010a-sr4.

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

Module Parameter Definition File:

..\Gpt_TS_T2D14M20I0R0 \config\Gpt.xdm

..\EcuM_TS_T2D14M20I0R0\config\EcuM.xdm

Code Generation Templates for Pre-Compile time configuration parameters:

..\Gpt_TS_T2D14M20I0R0 \generate_PC\src\Gpt_Cfg.c

..\Gpt_TS_T2D14M20I0R0\generate_PC\include\Gpt_Cfg.h

..\EcuM_TS_T2D14M20I0R0\generate_PC\include\EcuM_Cfg.h

Code Generation Templates for Post-Build time configuration parameters:

..\Gpt_TS_T2D14M20I0R0\generate_PB\src\Gpt_PBcfg.c

Tresos Configuration tool files:

..\Gpt_TS_T2D14M20I0R0\META-INF\ MANIFEST.MF

..\Gpt_TS_T2D14M20I0R0\plugin.xml

Steps to generate configurations:

1.Copy the module folder (Gpt_TS_T2D14M20I0R0), (EcuM_TS_T2D14M20I0R0) 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 STARTUP 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

If GptWakeupFunctionalityApi and GptWakeupSourceRef are enabled, the GPT shall report the wakeup event to EcuM through EcuM_SetWakeupEvent(Source) upon RTC event.

Chapter 5
Module Requirements

5.1 Peripheral Hardware Requirements

Driver implements 17 channels on four MPC5634M peripherals.

8 channels are implemented on the Enhanced Modular I/O Subsystem (eMIOS200) module.

4 channels are implemented on System Timer Module (STM).

5 channels are implemented on Periodic Interrupt Timer (PIT).

Refer Table GPT Hardware Channel availability for MPC5634M family in User Manual

5.2 ISR to configure within OS – dependencies

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

Table 5-1. GPT ISR's

ISR Name	Hardware interrupt vector
For EMIOS0	
ISR(EMIOS_0_CH_0_ISR)	51
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_12_ISR)	63
ISR(EMIOS_0_CH_14_ISR)	65
ISR(EMIOS_0_CH_15_ISR)	66
ISR(EMIOS_0_CH_23_ISR)	209
ForPIT Timers	

Table continues on the next page...

Table 5-1. GPT ISR's (continued)

ISR Name	Hardware interrupt vector
ISR(Gpt_PITRTI_RTC_Ch_0_ISR)	305
ISR(Gpt_PITRTI_TIMER_0_ISR)	301
ISR(Gpt_PITRTI_TIMER_1_ISR)	302
ISR(Gpt_PITRTI_TIMER_2_ISR)	303
ISR(Gpt_PITRTI_TIMER_3_ISR)	304
For STM Timers	
ISR(Gpt_STM_Ch_0_ISR)	200
ISR(Gpt_STM_Ch_1_ISR) (shared for channels 1,2,3)	201

5.3 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 SemiconductorOS 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.

5.4 Other AUTOSAR Modules - Dependencies

5.4.1 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 'GptDevErrorDetect' configuration parameter.

5.4.2 Diagnostic Event Manager:

This module is necessary for enabling reporting of production relevant error status. This is not used with current GPT implementation as the production relevant error codes are not present.

5.4.3 ECU Manager:

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.

5.4.4 MCU Module:

MCU module shall be initialized before using GPT .This module is required for setting the global prescaler value and to set the system clock frequency.

5.4.5 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_T2D14M20I0R0\autosar\EcuM.epd (Module Parameter Definition File – AUTOSAR Format)

..\EcuM_TS_T2D14M20I0R0\config\EcuM.xdm (Module Parameter Definition File – Tresos Format)

5.5 Exclusive Areas to Be Defined in BSW Scheduler

In the current implementation, GPT is using the services of Schedule Manager (SchM) for entering and exiting the critical regions. SchM implementation is done by the integrators of the MCAL using OS or non-OS services. For testing the ADC, stubs are used for SchM.

All GPT notification functions are called outside any critical region. Global variables updates are performed by ISRs before calling the user notification functions. So the GPT internal state is consistent at the moment of the notification call.

The following critical regions are used in the GPT driver:

5.5.1 EVEN_EXCLUSIVE_AREA

These areas are used in the functions Gpt_StartTimer/Gpt_StopTimer, channel is mapped to an even exclusive area according to formula (2*channel)for Gpt_StartTimer/ Gpt_StopTimer.

5.5.2 ODD_EXCLUSIVE_AREA

These areas are used in the functions Gpt_Enable/DisableNotification ,channel is mapped to an even exclusive area according to formula(2*channel+1)for Gpt_Enable/ DisableNotification.

5.5.3 GPT_EXCLUSIVE_AREA_64

Used in function Gpt_SetMode, protects the register write and the global variable Gpt_Current_Mode write - this operation has to be atomic

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.

	GPT_EXCLUSIVE_AREA_00	GPT_EXCLUSIVE_AREA_01	GPT_EXCLUSIVE_AREA_02	GPT_EXCLUSIVE_AREA_64
GPT_EXCLUSIVE_AREA_00	X			
GPT_EXCLUSIVE_AREA_01		X		
GPT_EXCLUSIVE_AREA_02			X	
GPT_EXCLUSIVE_AREA_64				X

Chapter 6
Main Api Requirements

6.1 Main Functions Calls within BSW Scheduler

None

6.2 Calls to Notification Functions, Callbacks, Callouts

6.2.1 Call-back Notifications

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

6.2.2 User Notifications

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 MemMap.h

Sections to be defined in MemMap.h

7.1.1 For Post Build Data

#ifdef GPT_START_CONFIG_DATA_UNSPECIFIED

#undef GPT_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=".pbgpt_cfg"

#endif

#ifdef GPT_STOP_CONFIG_DATA_UNSPECIFIED

#ifdef GPT_STOP_CONFIG_DATA_UNSPECIFIED

#undef MEMMAP_ERROR

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

#pragma ghs section

#endif

7.1.2 For Code

#ifdef GPT_START_SEC_CODE

```
#undef GPT_START_SEC_CODE
#undef MEMMAP_ERROR
/*Memory Section for Code to be defined here.*/
#endif
#ifdef GPT_STOP_SEC_CODE
#undef GPT_STOP_SEC_CODE
#undef MEMMAP_ERROR
/*End of section to be mentioned here*/
#endif
```

7.1.3 For Variables

```
#ifdef GPT_START_SEC_VAR_UNSPECIFIED
#undef GPT_START_SEC_VAR_UNSPECIFIED
#undef MEMMAP_ERROR
/*Memory Section for Variables to be defined here.*/
#endif
#ifdef GPT_STOP_SEC_VAR_UNSPECIFIED
#undef GPT_STOP_SEC_VAR_UNSPECIFIED
#undef MEMMAP_ERROR
/*End of section to be mentioned here*/
#endif
```

7.1.4 For Constant Data

```
#ifdef GPT_START_SEC_CONST_UNSPECIFIED
#undef GPT_START_SEC_CONST_UNSPECIFIED
#undef MEMMAP_ERROR
/*Memory Section for Constants to be defined here.*/
```

```
#endif
#ifdef GPT_STOP_SEC_CONST_UNSPECIFIED
#undef GPT_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 GPT driver fall into the following variants as defined below:

8.1 CONFIGURATION PARAMETER CONSIDERATIONS

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

Table 8-1. Configuration Parameters

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
GptDriverConfiguration			
	GptDevErrorDetect	Pre Compile parameter for all Variants of Configuration	Pre Compile
	GptReportWakeupSource	Pre Compile parameter for all Variants of Configuration	Pre Compile
GptConfigurationOfOptApiServices			
	GptDeinitApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	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
GptChannelConfigSet			
	GptSTMChannelPrescale	VariantPC or VariantPB	VariantPC or VariantPB

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	GptSTMChannelPrescale_Alternate	VariantPC or VariantPB	VariantPC or VariantPB
GptChannelConfiguration			
	GptChannelId	VariantPC or VariantPB	VariantPC or VariantPB
	GptHwchannel	VariantPC or VariantPB	VariantPC or VariantPB
	GptChannelMode	VariantPC or VariantPB	VariantPC or VariantPB
	GptChannelClkSrc	VariantPC or VariantPB	VariantPC or VariantPB
	GptChannelPrescale	VariantPC or VariantPB	VariantPC or VariantPB
	GptFreezeEnable	VariantPC or VariantPB	VariantPC or VariantPB
	GptEnableWakeUp	VariantPC or VariantPB	VariantPC or VariantPB
	GptNotification	VariantPC or VariantPB	VariantPC or VariantPB
	GptChannelPrescale_Alternate	VariantPC or VariantPB	VariantPC or VariantPB
GptWakeupConfiguration			
	GptWakeupSourceRef	VariantPC or VariantPB	VariantPC or VariantPB
GptNonAUTOSAR			
	GptEnableDualClockMode	VariantPC or VariantPB	Precompile

Chapter 9
Integration Steps

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

- Generate the required GPT configurations. For more details refer to section **Files required for Compilation**
- Allocate proper memory sections in MemMap.h and linker command file. For more details refer to section **Memory Allocation**
- 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 the GPT with all the dependent modules. For more details refer to section **Building the Driver** and **ISR to configure within OS – dependencies**

How to Reach Us:

Home Page:
www.freescale.com

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

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

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

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

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

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

Document Number: IM14GPTASR3.0R2.0.0
Rev. 2.0

freescale™