

# Integration Manual

for S32 PMIC Driver

Document Number: IM11PMICASR4.4 Rev0000R4.0.0 Rev. 1.0

<b>1 Revision History</b>	<b>2</b>
<b>2 Introduction</b>	<b>3</b>
2.1 Supported Derivatives . . . . .	3
2.2 Overview . . . . .	4
2.3 About This Manual . . . . .	4
2.4 Acronyms and Definitions . . . . .	5
2.5 Reference List . . . . .	5
<b>3 Building the driver</b>	<b>6</b>
3.1 Build Options . . . . .	6
3.1.1 GCC Compiler/Assembler/Linker Options . . . . .	6
3.1.2 GHS Compiler/Assembler/Linker Options . . . . .	9
3.1.3 DIAB Compiler/Assembler/Linker Options . . . . .	11
3.2 Files required for compilation . . . . .	13
3.3 Setting up the plugins . . . . .	16
<b>4 Function calls to module</b>	<b>18</b>
4.1 Function Calls during Start-up . . . . .	18
4.2 Function Calls during Shutdown . . . . .	18
4.3 Function Calls during Wake-up . . . . .	18
<b>5 Module requirements</b>	<b>19</b>
5.1 Exclusive areas to be defined in BSW scheduler . . . . .	19
5.2 Exclusive areas not available on this platform . . . . .	22
5.3 Peripheral Hardware Requirements . . . . .	22
5.4 ISR to configure within AutosarOS - dependencies . . . . .	22
5.5 ISR Macro . . . . .	22
5.5.1 Without an Operating System . . . . .	22
5.5.2 With an Operating System . . . . .	23
5.6 Other AUTOSAR modules - dependencies . . . . .	23
5.7 Data Cache Restrictions . . . . .	23
5.8 User Mode support . . . . .	24
5.8.1 User Mode support . . . . .	24
5.8.2 User Mode configuration in AutosarOS . . . . .	24
5.9 Multicore support . . . . .	25
<b>6 Main API Requirements</b>	<b>26</b>
6.1 Main function calls within BSW scheduler . . . . .	26
6.2 API Requirements . . . . .	26
6.3 Calls to Notification Functions, Callbacks, Callouts . . . . .	26

<b>7 Memory allocation</b>	<b>27</b>
7.1 Sections to be defined in Pmic_MemMap.h . . . . .	27
7.2 Linker command file . . . . .	28
<b>8 Integration Steps</b>	<b>29</b>
<b>9 External assumptions for driver</b>	<b>30</b>

## Chapter 1

### Revision History

Revision	Date	Author	Description
1.0	31.10.2022	NXP AASW Team	Prepared for release S32 RTD AUTOSAR 4.4 Version 4.0.0 Release

## Chapter 2

### Introduction

- [Supported Derivatives](#)
- [Overview](#)
- [About This Manual](#)
- [Acronyms and Definitions](#)
- [Reference List](#)

This integration manual describes the integration requirements for PMIC Driver for S32 microcontrollers.

### 2.1 Supported Derivatives

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

- s32g274a\_bga525
- s32g254a\_bga525
- s32g233a\_bga525
- s32g234m\_bga525
- s32g378a\_bga525
- s32g379a\_bga525
- s32g398a\_bga525
- s32g399a\_bga525
- s32g338m\_bga525
- s32g339m\_bga525
- s32g358a\_bga525
- s32g359a\_bga525

All of the above microcontroller devices are collectively named as S32.

## 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** style: Used for important terms, notes and warnings.
- *Italic* style: 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.

Warning

This is a warning

## 2.4 Acronyms and Definitions

Term	Definition
API	Application Programming Interface
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CS	Chip Select
CTU	Cross Trigger Unit
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
FIFO	First In First Out
LSB	Least Significant Bit
MCU	Micro Controller Unit
MIDE	Multi Integrated Development Environment
MSB	Most Significant Bit
N/A	Not Applicable
RAM	Random Access Memory
SIU	Systems Integration Unit
SWS	Software Specification
VLE	Variable Length Encoding
XML	Extensible Markup Language

## 2.5 Reference List

#	Title	Version
1	S32G2 Reference Manual	Rev. 5, May 2022
2	S32G3 Reference Manual	Rev. 2 Draft C, June 2022
3	S32G2 Mask Set Errata for Mask 0P77B	Rev. 2.4
4	S32G3 Mask Set Errata for Mask 0P72B	Rev. 1.1
5	S32G2 Data Sheet	Rev. 5, May 2022
6	S32G3 Data Sheet	Rev. 2 Draft B, June 2022
7	VR5510 Data Sheet	Rev. 5, April 2022

## Chapter 3

### Building the driver

- [Build Options](#)
- [Files required for compilation](#)
- [Setting up the plugins](#)

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

#### 3.1 Build Options

- [GCC Compiler/Assembler/Linker Options](#)
- [GHS Compiler/Assembler/Linker Options](#)
- [DIAB Compiler/Assembler/Linker Options](#)

The RTD driver files are compiled using:

- NXP GCC 9.2.0 20190812 (Build 1649 Revision gaf57174)
- Green Hills Multi 7.1.6d / Compiler 2020.1.4
- Wind River Diab Compiler 7.0.3

The compiler, assembler, and linker flags used for building the driver are explained below.

The TS\_T40D11M40I0R0 part of the plugin name is composed as follows:

- T = Target\_Id (e.g. T40 identifies Cortex-M architecture)
- D = Derivative\_Id (e.g. D11 identifies S32 platform)
- M = SW\_Version\_Major and SW\_Version\_Minor
- I = SW\_Version\_Patch
- R = Reserved

##### 3.1.1 GCC Compiler/Assembler/Linker Options

###### 3.1.1.1 GCC Compiler Options



Compiler Option	Description
-mcpu=cortex-m7	Targeted ARM processor for which GCC should tune the performance of the code
-mthumb	Generates code that executes in Thumb state
-mlittle-endian	Generate code for a processor running in little-endian mode
-mfpv=fpv5-sp-d16	Specifies the floating-point hardware available on the target
-mfloat-abi=hard	Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions
-std=c99	Specifies the ISO C99 base standard
-Os	Optimize for size. Enables all -O2 optimizations except those that often increase code size
-ggdb3	Produce debugging information for use by GDB using the most expressive format available, including GDB extensions if at all possible. Level 3 includes extra information, such as all the macro definitions present in the program
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid (or modify to prevent the warning), even in conjunction with macros
-Wextra	This enables some extra warning flags that are not enabled by -Wall
-pedantic	Issue all the warnings demanded by strict ISO C. Reject all programs that use forbidden extensions. Follows the version of the ISO C standard specified by the aforementioned -std option
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types
-Wundef	Warn if an undefined identifier is evaluated in an #if directive. Such identifiers are replaced with zero
-Wunused	Warn whenever a function, variable, label, value, macro is unused
-Werror=implicit-function-declaration	Make the specified warning into an error. This option throws an error when a function is used before being declared
-Wsign-compare	Warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned.
-Wdouble-promotion	Give a warning when a value of type float is implicitly promoted to double
-fno-short-enums	Specifies that the size of an enumeration type is at least 32 bits regardless of the size of the enumerator values.
-funsigned-char	Let the type char be unsigned by default, when the declaration does not use either signed or unsigned
-funsigned-bitfields	Let a bit-field be unsigned by default, when the declaration does not use either signed or unsigned
-fomit-frame-pointer	Omit the frame pointer in functions that don't need one. This avoids the instructions to save, set up and restore the frame pointer; on many targets it also makes an extra register available.

Compiler Option	Description
-fno-common	Makes the compiler place uninitialized global variables in the BSS section of the object file. This inhibits the merging of tentative definitions by the linker so you get a multiple-definition error if the same variable is accidentally defined in more than one compilation unit
-fstack-usage	This option is only used to build test for generation Ram/↔ Stack size report. Makes the compiler output stack usage information for the program, on a per-function basis
-fdump-ipa-all	This option is only used to build test for generation Ram/↔ Stack size report. Enables all inter-procedural analysis dumps
-c	Stop after assembly and produce an object file for each source file
-DS32XX	Predefine S32XX as a macro, with definition 1
-DS32G2XX	Predefine S32G2XX as a macro, with definition 1
-DGCC	Predefine GCC as a macro, with definition 1
-DUSE_SW_VECTOR_MODE	Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode
-DD_CACHE_ENABLE	Predefine D_CACHE_ENABLE as a macro, with definition 1. Enables data cache initialization in source file system.↔ c under the Platform driver
-DI_CACHE_ENABLE	Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initialization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPORT↔ RT as a macro, with definition 1. Allows drivers to be configured in user mode.

### 3.1.1.2 GCC Assembler Options

Assembler Option	Description
-X assembler-with-cpp	Specifies the language for the following input files (rather than letting the compiler choose a default based on the file name suffix)
-mcpu=cortexm7	Targeted ARM processor for which GCC should tune the performance of the code
-mfpv5-sp-d16	Specifies the floating-point hardware available on the target
-mfloat-abi=hard	Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions
-mthumb	Generates code that executes in Thumb state
-c	Stop after assembly and produce an object file for each source file

### 3.1.1.3 GCC Linker Options

Linker Option	Description
-Wl,-Map,filename	Produces a map file
-T linkerfile	Use linkerfile as the linker script. This script replaces the default linker script (rather than adding to it)
-entry=Reset_Handler	Specifies that the program entry point is Reset_Handler
-nostartfiles	Do not use the standard system startup files when linking
-mcpu=cortexm7	Targeted ARM processor for which GCC should tune the performance of the code
-mthumb	Generates code that executes in Thumb state
-mfpv=fpv5-sp-d16	Specifies the floating-point hardware available on the target
-mfloat-abi=hard	Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions
-mlittle-endian	Generate code for a processor running in little-endian mode
-ggdb3	Produce debugging information for use by GDB using the most expressive format available, including GDB extensions if at all possible. Level 3 includes extra information, such as all the macro definitions present in the program
-lc	Link with the C library
-lm	Link with the Math library
-lgcc	Link with the GCC library

### 3.1.2 GHS Compiler/Assembler/Linker Options

#### 3.1.2.1 GHS Compiler Options

Compiler Option	Description
-cpu=cortexm7	Selects target processor: Arm Cortex M7
-thumb	Selects generating code that executes in Thumb state
-fpv=vfpv5_d16	Specifies hardware floating-point using the v5 version of the VFP instruction set, with 16 double-precision floating-point registers
-fsingle	Use hardware single-precision, software double-precision FP instructions
-C99	Use (strict ISO) C99 standard (without extensions)
-ghstd=last	Use the most recent version of Green Hills Standard mode (which enables warnings and errors that enforce a stricter coding standard than regular C and C++)
-Osize	Optimize for size
-gnu_asm	Enables GNU extended asm syntax support
-dual_debug	Generate DWARF 2.0 debug information
-G	Generate debug information
-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
-Wimplicit-int	Produce warnings if functions are assumed to return int
-Wshadow	Produce warnings if variables are shadowed

Compiler Option	Description
-Wtrigraphs	Produce warnings if trigraphs are detected
-Wundef	Produce a warning if undefined identifiers are used in #if preprocessor statements
-unsigned_chars	Let the type char be unsigned, like unsigned char
-unsigned_fields	Bitfields declared with an integer type are unsigned
-no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup
-no_exceptions	Disables C++ support for exception handling
-no_slash_comment	C++ style // comments are not accepted and generate errors
-prototype_errors	Controls the treatment of functions referenced or called when no prototype has been provided
-incorrect_pragma_warnings	Controls the treatment of valid #pragma directives that use the wrong syntax
-c	Stop after assembly and produce an object file for each source file
-DS32XX	Predefine S32XX as a macro, with definition 1
-DS32G2XX	Predefine S32G2XX as a macro, with definition 1
-DGHS	Predefine GHS as a macro, with definition 1
-DUSE_SW_VECTOR_MODE	Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode
-DD_CACHE_ENABLE	Predefine D_CACHE_ENABLE as a macro, with definition 1. Enables data cache initialization in source file system.c under the Platform driver
-DI_CACHE_ENABLE	Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initialization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPORT as a macro, with definition 1. Allows drivers to be configured in user mode

### 3.1.2.2 GHS Assembler Options

Assembler Option	Description
-cpu=cortexm7	Selects target processor: Arm Cortex M7
-fpu=vfpv5_d16	Specifies hardware floating-point using the v5 version of the VFP instruction set, with 16 double-precision floating-point registers
-fsingle	Use hardware single-precision, software double-precision FP instructions
-preprocess_assembly_files	Controls whether assembly files with standard extensions such as .s and .asm are preprocessed
-list	Creates a listing by using the name and directory of the object file with the .lst extension
-c	Stop after assembly and produce an object file for each source file

### 3.1.2.3 GHS Linker Options

Linker Option	Description
-e Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
-T linker_script_file.ld	Use linker_script_file.ld as the linker script. This script replaces the default linker script (rather than adding to it)
-map	Produce a map file
-keepmap	Controls the retention of the map file in the event of a link error
-Mn	Generates a listing of symbols sorted alphabetically/numerically by address
-delete	Instructs the linker to remove functions that are not referenced in the final executable. The linker iterates to find functions that do not have relocations pointing to them and eliminates them
-ignore_debug_references	Ignores relocations from DWARF debug sections when using -delete. DWARF debug information will contain references to deleted functions that may break some third-party debuggers
-Llibrary_path	Points to library_path (the libraries location) for thumb2 to be used for linking
-larch	Link architecture specific library
-lstartup	Link run-time environment startup routines. The source code for themodules in this library is provided in the src/libstartup directory
-lind_sd	Link language-independent library, containing support routines for features such as software floating point, run-time error checking, C99 complex numbers, and some general purpose routines of the ANSI C library
-v	Prints verbose information about the activities of the linker, including the libraries it searches to resolve undefined symbols
-nostartfiles	Controls the start files to be linked into the executable

### 3.1.3 DIAB Compiler/Assembler/Linker Options

#### 3.1.3.1 DIAB Compiler Options

Compiler Option	Description
-tARMCORTEXM7MG:simple	Selects target processor (hardware single-precision, software double-precision floating-point)
-mthumb	Selects generating code that executes in Thumb state
-std=c99	Follows the C99 standard for C
-Oz	Like -O2 with further optimizations to reduce code size
-g	Generates DWARF 4.0 debug information
-fstandalone-debug	Emits full debug info for all types used by the program
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types
-Wsign-compare	Produce warnings when comparing signed type with unsigned type
-Wdouble-promotion	Give a warning when a value of type float is implicitly promoted to double

Compiler Option	Description
-Wunknown-pragmas	Issues a warning for unknown pragmas
-Wundef	Warns if an undefined identifier is evaluated in an <code>#if</code> directive. Such identifiers are replaced with zero
-Wextra	Enables some extra warning flags that are not enabled by '-Wall'
-Wall	Enables all of the most useful warnings (for historical reasons this option does not literally enable all warnings)
-pedantic	Emits a warning whenever the standard specified by the -std option requires a diagnostic
-Werror=implicit-function-declaration	Generates an error whenever a function is used before being declared
-fno-common	Compile common globals like normal definitions
-fno-signed-char	Char is unsigned
-fno-trigraphs	Do not process trigraph sequences
-V	Displays the current version number of the tool suite
-c	Stop after assembly and produce an object file for each source file
-DS32XX	Predefine S32XX as a macro, with definition 1
-DS32G2XX	Predefine S32G2XX as a macro, with definition 1
-DDIAB	Predefine DIAB as a macro, with definition 1
-DUSE_SW_VECTOR_MODE	Predefine USE_SW_VECTOR_MODE as a macro, with definition 1. By default, the drivers are compiled to handle interrupts in Software Vector Mode
-DD_CACHE_ENABLE	Predefine D_CACHE_ENABLE as a macro, with definition 1. Enables data cache initialization in source file system.c under the Platform driver
-DI_CACHE_ENABLE	Predefine I_CACHE_ENABLE as a macro, with definition 1. Enables instruction cache initialization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPORT as a macro, with definition 1. Allows drivers to be configured in user mode

### 3.1.3.2 DIAB Assembler Options

Assembler Option	Description
-mthumb	Selects generating code that executes in Thumb state
-Xpreprocess-assembly	Invokes C preprocessor on assembly files before running the assembler
-Xassembly-listing	Produces an .lst assembly listing file
-c	Stop after assembly and produce an object file for each source file

### 3.1.3.3 DIAB Linker Options

Linker Option	Description
-e Reset_Handler	Make the symbol Reset_Handler be treated as a root symbol and the start label of the application
linker_script_file.dld	Use linker_script_file.dld as the linker script. This script replaces the default linker script (rather than adding to it)
-m30	m2 + m4 + m8 + m16
-Xstack-usage	Gathers and display stack usage at link time
-Xpreprocess-lecl	Perform pre-processing on linker scripts
-Llibrary_path	Points to the libraries location for ARMV7EMMG to be used for linking
-lc	Links with the standard C library
-lm	Links with the math library

## 3.2 Files required for compilation

This section describes the include files required to compile, assemble (if assembler code) and link the Pmic driver for S32G274 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.

### Pmic Files

- ..\Pmic\_TS\_T40D11M40I0R0\src\CDD\_Pmic.c
- ..\Pmic\_TS\_T40D11M40I0R0\src\Pmic\_Crc.c
- ..\Pmic\_TS\_T40D11M40I0R0\src\Pmic\_Dem.c
- ..\Pmic\_TS\_T40D11M40I0R0\src\Pmic\_Det.c
- ..\Pmic\_TS\_T40D11M40I0R0\src\Pmic\_Ipw.c
- ..\Pmic\_TS\_T40D11M40I0R0\src\Pmic\_VR55XX.c
- ..\Pmic\_TS\_T40D11M40I0R0\src\pmic\_i2c\_external\_access.c
- ..\Pmic\_TS\_T40D11M40I0R0\src\pmic\_ocotp\_external\_access.c
- ..\Pmic\_TS\_T40D11M40I0R0\src\pmic\_pin\_external\_access.c
- ..\Pmic\_TS\_T40D11M40I0R0\include\CDD\_Pmic.h
- ..\Pmic\_TS\_T40D11M40I0R0\include\Pmic\_Crc.h
- ..\Pmic\_TS\_T40D11M40I0R0\include\Pmic\_Dem.h
- ..\Pmic\_TS\_T40D11M40I0R0\include\Pmic\_Det.h
- ..\Pmic\_TS\_T40D11M40I0R0\include\Pmic\_Internals.h
- ..\Pmic\_TS\_T40D11M40I0R0\include\CDD\_Ipw.h
- ..\Pmic\_TS\_T40D11M40I0R0\include\Pmic\_Ipw\_Types.h
- ..\Pmic\_TS\_T40D11M40I0R0\include\Pmic\_Types.h

## Building the driver

- ..\Pmic\_TS\_T40D11M40I0R0\include\Pmic\_VR55XX.h
- ..\Pmic\_TS\_T40D11M40I0R0\include\Pmic\_VR55XX\_Types.h
- ..\Pmic\_TS\_T40D11M40I0R0\include\Reg\_eSys\_VR55XX.h
- ..\Pmic\_TS\_T40D11M40I0R0\include\pmic\_i2c\_external\_access.h
- ..\Pmic\_TS\_T40D11M40I0R0\include\pmic\_ocotp\_external\_access.h
- ..\Pmic\_TS\_T40D11M40I0R0\include\pmic\_pin\_external\_access.h

## Pmic Generated Files

- CDD\_Pmic\_Cfg.c - This file should be generated by the user using a configuration tool for compilation.
- CDD\_Pmic\_Cfg.h - This file should be generated by the user using a configuration tool for compilation.
- CDD\_Pmic\_[VariantName]\_PBcfg.c - This file should be generated by the user using a configuration tool for compilation.

## As a deviation from standard:

- CDD\_Pmic\_[VariantName]\_PBcfg.c - This file will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, PB).
- CDD\_Pmic\_Cfg.c - This 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 `<ph conref="../../variables.xml#variables/DRV_NAME">_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\_T40D11M40I0R0\include\
- ..\Base\_TS\_T40D11M40I0R0\header\
- ..\Base\_TS\_T40D11M40I0R0\src\

## Files from Dem folder:

- ..\Dem\_TS\_T40D11M40I0R0\include\
- ..\Dem\_TS\_T40D11M40I0R0\src\

## Files from Det folder:

- ..\Det\_TS\_T40D11M40I0R0\include\
- ..\Det\_TS\_T40D11M40I0R0\src\



**Files from I2c folder:**

- ..\I2c\_TS\_T40D11M40I0R0\include\
- ..\I2c\_TS\_T40D11M40I0R0\src\

**Files from Port folder:**

- ..\Port\_TS\_T40D11M40I0R0\include\
- ..\Port\_TS\_T40D11M40I0R0\src\

**Files from Dio folder:**

- ..\Dio\_TS\_T40D11M40I0R0\include\
- ..\Dio\_TS\_T40D11M40I0R0\src\

**Files from Ocotp folder:**

- ..\Ocotp\_TS\_T40D11M40I0R0\include\
- ..\Ocotp\_TS\_T40D11M40I0R0\src\

**Files from Mcu folder:**

- ..\Mcu\_TS\_T40D11M40I0R0\include\
- ..\Mcu\_TS\_T40D11M40I0R0\src\

**Files from RTE folder:**

- Rte\_TS\_T40D11M40I0R0\include\
- Rte\_TS\_T40D11M40I0R0\src\

**Note:** <plugin\_name>: TS\_T<40>D<11>M<SW\_Version\_Major>I<SW\_Version\_Minor>R0 (i.e. Target\_Id = 2 identifies PowerPC architecture and Derivative\_Id = 11 identifies the S32GXX)

### 3.3 Setting up the plugins

The Pmic driver was designed to be configured by using the EB Tresos Studio (version 27.1.0 b200625-0900 or later.)

#### Location of various files inside the PMIC module folder:

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - `..\Pmic_TS_T40D11M40I0R0\config\Pmic.xdm`
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - `..\Pmic_TS_T40D11M40I0R0\config\Pmic_<subderivative_name>.epd`
- Code Generation Templates for parameters without variation points:
  - `..\Pmic_TS_T40D11M40I0R0\generate_PC\include\CDD_Pmic_Cfg.h`
  - `..\Pmic_TS_T40D11M40I0R0\generate_PC\src\CDD_Pmic_Cfg.c`
- Code Generation Templates for variant aware parameters:
  - `..\Pmic_TS_T40D11M40I0R0\generate_PB\include\CDD_Pmic_Cfg.h`
  - `..\Pmic_TS_T40D11M40I0R0\generate_PB\include\CDD_Pmic_PBcfg.h`
  - `..\Pmic_TS_T40D11M40I0R0\generate_PB\src\CDD_Pmic_Cfg.c`
  - `..\Pmic_TS_T40D11M40I0R0\generate_PB\src\CDD_Pmic_PBcfg.c`

#### Steps to generate the configuration:

1. Copy the module folders `Pmic_TS_T40D11M40I0R0`, `Base_TS_T40D11M40I0R0`, `Resource_TS_T40D11M40I0R0`, `Det_TS_T40D11M40I0R0`, `EcuC_TS_T40D11M40I0R0`, `Os_TS_T40D11M40I0R0`, `I2c_TS_T40D11M40I0R0`, `Ocotp_TS_T40D11M40I0R0`, `Dio_TS_T40D11M40I0R0`, `Port_TS_T40D11M40I0R0`, `Mcu_TS_T40D11M40I0R0` into the Tresos plugins folder.
2. Set the desired Tresos Output location folder for the generated sources and header files.
3. Use the EB tresos Studio GUI to modify ECU configuration parameters values.
4. Generate the configuration files.

#### Dependencies

- RESOURCE is required to select processor derivative. Current driver has support for the following derivatives, each one having attached a Resource file: `s32g233a_bga525`, `s32g234m_bga525`, `s32g254a_bga525`, `s32g274a_bga525`, `s32g338m_bga525`, `s32g339m_bga525`, `s32g358a_bga525`, `s32g359a_bga525`, `s32g378a_bga525`, `s32g379a_bga525`, `s32g398a_bga525`, `s32g399a_bga525`.
- Base is required for platform-specific files.
- DET is required for signaling the development error detection (parameters out of range, null pointers, etc).
- DEM is required for signaling the production error detection (hardware failure, etc).
- GPT is required for handling the watchdog internal timer
- MCU is required for selecting the timebase for the watchdog internal timer, via Gpt
- MCL is required by Gpt to retrieve the timer specific files

- PMIC is required for sending the commands to the External Chip and for triggering correctly.
- I2C is required by PMIC for communicating with the external chip
- RTE is required for critical sections
- ECUM is required for selecting the reference to the wakeup source for every Gpt channel configured as a wakeup source.
- ECUC is required for selecting variants.
- OS is required for selecting core.
- OCOTP is required to check the DIE\_PROCESS OTP of Mcu before Pmic driver configures the static voltage scaler(SVS) of the VR5510 device. This module is only required on S32G2xx devices
- PORT is required for configuring port I2c connection. It needs by Pmic driver.
- DIO is required for configuring pin I2c connection. It needs by Pmic driver.

## Chapter 4

### Function calls to module

- [Function Calls during Start-up](#)
- [Function Calls during Shutdown](#)
- [Function Calls during Wake-up](#)

#### 4.1 Function Calls during Start-up

**Function Calls during Start-up** The MCU, PORT, I2C modules shall be initialized before the Pmic driver is initialized. If the Time Service (Tm) module is used for configuring the timeout mechanism in TICKS, then the GPT module shall also be initialized beforehand. If the External Watchdog (Wdg\_VR5510) module is also used, then this module should also be initialized beforehand in the following order: Pmic\_Init -> Wdg\_Vr5510\_Init -> Pmic\_InitDevice.

#### 4.2 Function Calls during Shutdown

##### Function Calls during Shutdown

- Pmic\_SetMode (STBY mode) API shall be called during GO SLEEP phase of the EcuM's Shutdown state to prepare the Pmic device for low-power state.
- Mcu\_SetMode (SoC\_STBY mode) API shall be called during GO SLEEP phase of the EcuM's Shutdown state to configure the hardware for Platform Standby mode. This shall be called after ICU & GPT are set to sleep.
- The timespan between the Pmic\_SetMode and Mcu\_SetMode function shall be lower than the configured TIMING\_WINDOW\_STBY period in case it was not disabled at initialization (TIMING\_WINDOW\_STBY can be configured by node "PmicSafetyStandbyWindowDuration" in the configuration-schema). Otherwise, the regulators will be disabled before the SoC has a chance to transition into Standby mode.

#### 4.3 Function Calls during Wake-up

None.

## Chapter 5

### Module requirements

- Exclusive areas to be defined in BSW scheduler
- Exclusive areas not available on this platform
- Peripheral Hardware Requirements
- ISR to configure within AutosarOS - dependencies
- ISR Macro
- Other AUTOSAR modules - dependencies
- Data Cache Restrictions
- User Mode support
- Multicore support

#### 5.1 Exclusive areas to be defined in BSW scheduler

In the current implementation, PMIC is using the services of Schedule Manager (SchM) for entering and exiting the exclusive areas. The following critical regions are used in the PMIC driver:

**Exclusive Areas are used in High level driver layer (HLD)**

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_WriteRegister` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers in `Pmic_VR55XX_I2C_WriteRegister`.

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_InitDevice` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers in `Pmic_VR55XX_I2C_WriteRegister`.

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_EmulateDeviceOTP` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers in `Pmic_VR55XX_I2C_WriteRegister`.

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_InitClock` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers in `Pmic_VR55XX_I2C_WriteRegister`.

## Module requirements

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_SetMode` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers in `Pmic_VR55XX_I2C_WriteRegister`.

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_SetAnalogMux` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers in `Pmic_VR55XX_I2C_WriteRegister`.

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_ConfigureWatchdog` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers in `Pmic_VR55XX_I2C_WriteRegister`.

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_TriggerWatchdog` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers in `Pmic_VR55XX_I2C_WriteRegister`.

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_SetReactions` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers in `Pmic_VR55XX_I2C_WriteRegister`.

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_DisableWatchdog` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers in `Pmic_VR55XX_I2C_WriteRegister`.

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_SwitchSVS` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers in `Pmic_VR55XX_I2C_WriteRegister`.

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_GotoInitFS` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers in `Pmic_VR55XX_I2C_WriteRegister`.

**PMIC\_EXCLUSIVE\_AREA\_01** is used in function `Pmic_ReadRegister` to protect the transferring data between SoC and PMIC device when the SoC is reading data from the PMIC registers in `Pmic_VR55XX_I2C_ReadRegister`.

**PMIC\_EXCLUSIVE\_AREA\_01** is used in function `Pmic_InitDevice` to protect the transferring data between SoC and PMIC device when the SoC is reading data from the PMIC registers in `Pmic_VR55XX_I2C_ReadRegister`.

**PMIC\_EXCLUSIVE\_AREA\_01** is used in function `Pmic_EmulateDeviceOTP` to protect the transferring data between SoC and PMIC device when the SoC is reading data from the PMIC registers in `Pmic_VR55XX_I2C_ReadRegister`.

**PMIC\_EXCLUSIVE\_AREA\_01** is used in function `Pmic_InitClock` to protect the transferring data between SoC and PMIC device when the SoC is reading data from the PMIC registers in `Pmic_VR55XX_I2C_ReadRegister`.

**PMIC\_EXCLUSIVE\_AREA\_01** is used in function `Pmic_TriggerWatchdog` to protect the transferring data between SoC and PMIC device when the SoC is reading data from the PMIC registers in `Pmic_VR55XX_I2C_ReadRegister`.

**PMIC\_EXCLUSIVE\_AREA\_01** is used in function `Pmic_SetMode` to protect the transferring data between SoC and PMIC device when the SoC is reading data from the PMIC registers in `Pmic_VR55XX_I2C_ReadRegister`.

**PMIC\_EXCLUSIVE\_AREA\_01** is used in function `Pmic_GetRawFaultEvents` to protect the transferring data between SoC and PMIC device when the SoC is reading data from the PMIC registers in `Pmic_VR55XX_I2C_ReadRegister`.

**PMIC\_EXCLUSIVE\_AREA\_01** is used in function `Pmic_GetDeviceInfo` to protect the transferring data between SoC and PMIC device when the SoC is reading data from the PMIC registers in `Pmic_VR55XX_I2C_ReadRegister`.

**PMIC\_EXCLUSIVE\_AREA\_01** is used in function `Pmic_DisableWatchdog` to protect the transferring data between SoC and PMIC device when the SoC is reading data from the PMIC registers in `Pmic_VR55XX_I2C_ReadRegister`.

**PMIC\_EXCLUSIVE\_AREA\_01** is used in function `Pmic_SwitchSVS` to protect the transferring data between SoC and PMIC device when the SoC is reading data from the PMIC registers in `Pmic_VR55XX_I2C_ReadRegister`.

**PMIC\_EXCLUSIVE\_AREA\_01** is used in function `Pmic_GotoInitFS` to protect the transferring data between SoC and PMIC device when the SoC is reading data from the PMIC registers in `Pmic_VR55XX_I2C_ReadRegister`.

**PMIC\_EXCLUSIVE\_AREA\_02** is used in function `Pmic_InitDevice` to protect the release I2c bus before initialize device in `Pmic_VR55XX_InitDevice`.

**PMIC\_EXCLUSIVE\_AREA\_03** is used in function `Pmic_InitDevice` to protect the process from reading `watchdog_seed` value, calculating and writing `release_FS0B` value in `Pmic_VR55XX_InitDevice`.

**PMIC\_EXCLUSIVE\_AREA\_03** is used in function `Pmic_SwitchSVS` to protect the process from reading `watchdog_seed` value, calculating and writing `release_FS0B` value in `Pmic_VR55XX_SwitchSVS`.

**PMIC\_EXCLUSIVE\_AREA\_03** is used in function `Pmic_DisableWatchdog` to protect the process from reading `watchdog_seed` value, calculating and writing `release_FS0B` value in `Pmic_VR55XX_DisableWatchdog`.

**PMIC\_EXCLUSIVE\_AREA\_03** is used in function `Pmic_ReleaseFs0b` to protect the process from reading `watchdog_seed` value, calculating and writing `release_FS0B` value in `Pmic_VR55XX_ReleaseFs0b`.

#### Exclusive Areas are implemented in Low level driver layer (IPL)

**PMIC\_EXCLUSIVE\_AREA\_00** is used in function `Pmic_VR55XX_I2C_WriteRegister` to protect the transferring data between SoC and PMIC device when the SoC is writing data to the PMIC registers.

**PMIC\_EXCLUSIVE\_AREA\_01** is used in function `Pmic_VR55XX_I2C_ReadRegister` to protect the transferring data between SoC and PMIC device when the SoC is reading data from the PMIC registers.

**PMIC\_EXCLUSIVE\_AREA\_02** is used in function `Pmic_VR55XX_InitDevice` to protect the release I2c bus before initialize device.

**PMIC\_EXCLUSIVE\_AREA\_03** is used in function `Pmic_VR55XX_InitDevice` to protect the process from reading `watchdog_seed` value, calculating and writing `release_FS0B` value.

**PMIC\_EXCLUSIVE\_AREA\_03** is used in function `Pmic_VR55XX_SwitchSVS` to protect the process from reading `watchdog_seed` value, calculating and writing `release_FS0B` value.

**PMIC\_EXCLUSIVE\_AREA\_03** is used in function `Pmic_VR55XX_DisableWatchdog` to protect the process from reading `watchdog_seed` value, calculating and writing `release_FS0B` value.

**PMIC\_EXCLUSIVE\_AREA\_03** is used in function `Pmic_VR55XX_ReleaseFs0b` to protect the process from reading `watchdog_seed` value, calculating and writing `release_FS0B` value.

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

#	AREA_00	AREA_01	AREA_02	AREA_03
AREA_00	x			
AREA_01		x		
AREA_02			x	
AREA_03				x

## 5.2 Exclusive areas not available on this platform

None.

## 5.3 Peripheral Hardware Requirements

For S32 controllers, the Pmic driver functionality which enables communication with VR55XX revision B1 device. The VR55XX is a NXP device power control management.

## 5.4 ISR to configure within AutosarOS - dependencies

isr to configure within AUTOSAROS dependencies template.

ISR Name	HW INT Vector	Observations
ISR(IsrName)	1	None

## 5.5 ISR Macro

RTD 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.

### 5.5.1 Without an Operating System

The macro *USING\_OS\_AUTOSAROS* must not be defined.

#### 5.5.1.1 Using Software Vector Mode

The macro *USE\_SW\_VECTOR\_MODE* must be defined and the ISR macro is defined as:

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

In this case, the drivers' interrupt handlers are normal C functions and their prologue/epilogue will handle the context save and restore.



### 5.5.1.2 Using Hardware Vector Mode

The macro `USE_SW_VECTOR_MODE` must not be defined and the ISR macro is defined as:

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

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

**5.5.2 With an Operating System** Please refer to your OS documentation for description of the ISR macro.

## 5.6 Other AUTOSAR modules - dependencies

Dependencies

- **BASE:** The BASE module contains the common files/definitions needed by all RTD (Real Time Drivers) modules.
- **DEM** The DEM module is used for enabling reporting of production relevant error status. The API function used is `Dem_SetEventStatus()`.
- **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 'WdgDevErrorDetect' configuration parameter.
- **ECUC:** The ECUC module is used for ECU configuration. RTD modules need ECUC to retrieve the variant information.
- **MCU:** The MCU driver provides services for basic micro-controller initialization, power down functionality, reset and micro-controller specific functions required by other RTD software modules. The clocks need to be initialized prior to using the `Wdg_VR5510` driver.
- **RESOURCE** Resource module is used to select micro-controller's derivatives. Current driver has support for the following derivatives, everyone having attached a Resource file: `s32g233a_bga525`, `s32g234m_bga525`, `s32g254a_bga525`, `s32g274a_bga525`, `s32g338m_bga525`, `s32g339m_bga525`, `s32g358a_bga525`, `s32g359a_bga525`, `s32g378a_bga525`, `s32g379a_bga525`, `s32g398a_bga525`, `s32g399a_bga525`.
- **RTE** The RTE module is needed for implementing data consistency of exclusive areas that are used by `Wdg` module.
- **I2C** The I2C module is used for data transmissions on the VR5510 chip and is already a dependency for the PMIC driver.
- **OCOTP** Module is needed by `Pmic` functions. OCOTP is required to check the `DIE_PROCESS_OTP` of `Mcu` before `Pmic` driver configures the static voltage scaler(SVS) of the VR5510 device. This module is only required on S32G2xx devices.
- **PORT** is required for configure port I2c connection. It needs by `Pmic` driver.
- **DIO** is required for configure pin I2c connection. It needs by `Pmic` driver.

## 5.7 Data Cache Restrictions

In the DMA transfer mode, DMA transfers may issue cache coherency problems. To avoid possible coherency issues when D-CACHE is enabled, the user shall ensure that the buffers used as TCD source and destination are allocated in the NON-CACHEABLE area (by means of `Pmic_Memmap`).

## 5.8 User Mode support

- [User Mode support](#)
- [User Mode configuration in AutosarOS](#)

**5.8.1 User Mode support** No special measures need to be taken to run Pmic module from user mode. The Pmic driver code can be executed at any time from both supervisor and user mode.

### 5.8.2 User Mode configuration in AutosarOS

When User mode is enabled, the driver may have the functions that need to be called as trusted functions in AutosarOS context. Those functions are already defined in driver and declared in the header `<IpName>_IpTrustedFunctions.h`. This header also included all headers files that contains all types definition used by parameters or return types of those functions. Refer the chapter [User Mode support](#) for more detail about those functions and the name of header files they are declared inside. Those functions will be called indirectly with the naming convention below in order to AutosarOS can call them as trusted functions.

```
Call_<Function_Name>_TRUSTED(parameter1,parameter2,...)
```

That is the result of macro expansion `OsIf_Trusted_Call` in driver code:

```
#define OsIf_Trusted_Call[1-6params](name,param1,...,param6) Call_##name##_TRUSTED(param1,...,param6)
```

So, the following steps need to be done in AutosarOS:

- Ensure `MCAL_ENABLE_USER_MODE_SUPPORT` macro is defined in the build system or somewhere global.
- Define and declare all functions that need to call as trusted functions follow the naming convention above in Integration/User code. They need to be visible in `Os.h` for the driver to call them. They will do the marshalling of the parameters and call `CallTrustedFunction()` in OS specific manner.
- `CallTrustedFunction()` will switch to privileged mode and call `TRUSTED_<Function_Name>()`.
- `TRUSTED_<Function_Name>()` function is also defined and declared in Integration/User code. It will un-marshalling of the parameters to call `<Function_Name>()` of driver. The `<Function_Name>()` functions are already defined in driver and declared in `<IpName>_IpTrustedFunctions.h`. This header should be included in OS for OS call and indexing these functions.

See the sequence chart below for an example calling `Linflexd_Uart_Ip_Init_Privileged()` as a trusted function.

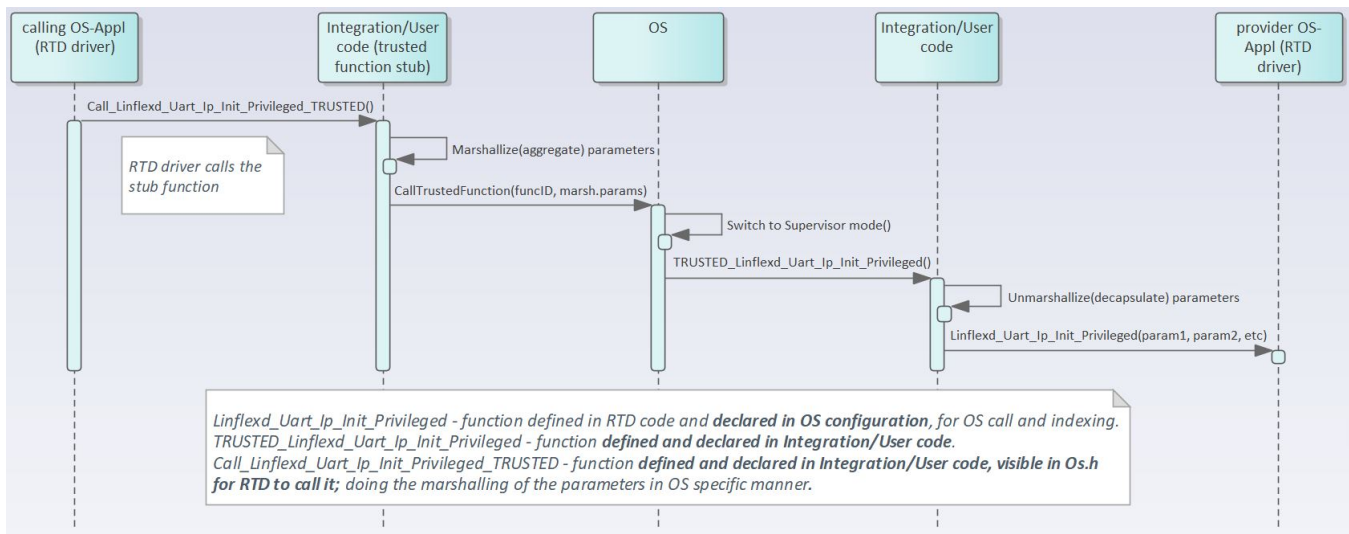


Figure 5.1 Example sequence chart for calling `Linflexd_Uart_Ip_Init_Privileged` as trusted function

## 5.9 Multicore support

Pmic driver doesn't support multicore.

## Chapter 6

### Main API Requirements

- [Main function calls within BSW scheduler](#)
- [API Requirements](#)
- [Calls to Notification Functions, Callbacks, Callouts](#)

#### 6.1 Main function calls within BSW scheduler

None.

#### 6.2 API Requirements

None.

#### 6.3 Calls to Notification Functions, Callbacks, Callouts

**Configurable User Notifications** The Pmic driver provides the following configurable user notifications:

- **PmicWatchdogTaskNotification:** This callback should be enabled only when not using an external watchdog driver. It is used to notify the upper layer when to start the Watchdog triggering task and each time the Watchdog configuration is changed. It has three parameters: the first represents the device id, the second represents the Watchdog Window duration, and the third represents the Watchdog Window duty cycle.

## Chapter 7

### Memory allocation

- [Sections to be defined in Pmic\\_MemMap.h](#)
- [Linker command file](#)

#### 7.1 Sections to be defined in Pmic\_MemMap.h

Section name	Type of section	Description
PMIC_START_SEC_CONFIG_DATA↔ _UNSPECIFIED	Configuration Data	Start of Memory Section for Config Data.
PMIC_STOP_SEC_CONFIG_DATA↔ _UNSPECIFIED	Configuration Data	End of Memory Section for Config Data.
PMIC_START_SEC_CODE	Code	Start of memory Section for Code.
PMIC_STOP_SEC_CODE	Code	End of memory Section for Code.
PMIC_START_SEC_VAR_CLEARED↔ _UNSPECIFIED	Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 8, 16 or 32 bit. These variables are cleared to zero by start-up code.
PMIC_STOP_SEC_VAR_CLEARED↔ _UNSPECIFIED	Variables	End of above section.
PMIC_START_SEC_VAR_CLEARED↔ _BOOLEAN	Variables	Used for variables, structures, arrays that have to be aligned to 8 bit. These variables are cleared to zero by start-up code.
PMIC_STOP_SEC_VAR_CLEARED↔ _BOOLEAN	Variables	End of above section.
PMIC_START_SEC_VAR_CLEARED↔ _16	Variables	Used for variables that have to be aligned to 16 bit. These variables are cleared to zero by start-up code.
PMIC_STOP_SEC_VAR_CLEARED↔ _16	Variables	End of above section.
PMIC_START_SEC_CONST_8	Variables	Used for constant variables that have to be aligned to 8 bit. These variables are initialized with values after every reset
PMIC_STOP_SEC_CONST_8	Variables	End of above section.

## 7.2 Linker command file

Memory shall be allocated for every section defined in the driver's "<Module>\_\_MemMap.h.

## Chapter 8

### Integration Steps

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

1. Generate the required module configuration(s). For more details refer to section [Files Required for Compilation](#)
2. Allocate the proper memory sections in the driver's memory map header file ("`<Module>_MemMap.h`") and linker command file. For more details refer to section [Sections to be defined in `<Module>\_MemMap.h`](#)
3. Compile & build the module with all the dependent modules. For more details refer to section [Building the Driver](#)

## Chapter 9

### External assumptions for driver

**Table 9.1 External Assumption**

External Assumption Req_Id	External Assumption Text
SWS_CDD_Pmic_00024	It is the system integrator's responsibility to make sure the software system integration requirements (SIRs) stated in the safety manual of the targeted Pmic device are fulfilled.
SWS_CDD_Pmic_00048	Pmic_InitDevice shall be called every time the device is reset (e.g. by transitioning out of standby mode or deep-sleep mode or transitioning into shutdown mode or reset mode).
SWS_CDD_Pmic_00048	Pmic_InitDevice function shall not configure any Failsafe register if device already was Initialized and moved to Normal_FS state. Eg: If device was Initialized and switched to Normal_FS in a boot phase, Pmic_InitDevice function will return successfully without any failsafe configuration in application phase."



**How to Reach Us:**

**Home Page:**

[nxp.com](http://nxp.com)

**Web Support:**

[nxp.com/support](http://nxp.com/support)

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: [nxp.com/SalesTermsandConditions](http://nxp.com/SalesTermsandConditions).

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2022 NXP B.V.

