Integration Manual

for S32K1_S32M24x ADC Driver

Document Number: IM2ADCASRR21-11 Rev0000R2.0.0 Rev. 1.0

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	04.08.2023	NXP RTD Team	S32K1_S32M24X Real-Time Drivers AUTOSAR 4.4 & R21-11 Version 2.0.0

Chapter 2

Introduction

This integration manual describes the integration requirements for ADC Driver for S32K1XX and S32M24X microcontrollers.

2.1 Supported Derivatives

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

- s32k116_qfn32
- s32k116_lqfp48
- s32k118_lqfp48
- s32k118_lqfp64
- s32k142_lqfp48
- s32k142_lqfp64
- s32k142_lqfp100
- s32k142w_lqfp48
- s32k142w_lqfp64
- s32k144_lqfp48
- s32k144_lqfp64 / MWCT1014S_lqfp64

- s32k144_lqfp100 / MWCT1014S_lqfp100
- s32k144_mapbga100
- s32k144w_lqfp48
- s32k144w_lqfp64
- s32k146_lqfp64
- s32k146_lqfp100 / MWCT1015S_lqfp100
- s32k146_mapbga100 / MWCT1015S_mapbga100
- s32k146_lqfp144
- s32k148_lqfp100
- s32k148_mapbga100 / MWCT1016S_mapbga100
- s32k148_lqfp144
- s32k148_lqfp176
- s32m241_lqfp64
- s32m242_lqfp64
- s32m243_lqfp64
- s32m244_lqfp64

All of the above microcontroller devices are collectively named as S32K1_S32M24X. Note: MWCT part numbers contain NXP confidential IP for Qi Wireless Power

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
ADC	Analog to Digital Converter
API	Application Programming Interface
ASM	Assembler
C/CPP	C and C++ Source Code
CS	Chip Select
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
FIFO	First In First Out
LSB	Least Signifigant Bit
MCU	Micro Controller Unit
MSB	Most Significant Bit
N/A	Not Applicable
PDB	Programmable Delay Block
RAM	Random Access Memory
SIM	System Integration Module
SIU	Systems Integration Unit
SWS	Software Specification
XML	Extensible Markup Language

2.5 Reference List

#	Title	Version
1	Specification of ADC Driver	AUTOSAR Release 4.4 & R21-11 Version 2.0.0
2	S32M24x Reference Manual	Rev. 2 Draft A, 05/2023
3	S32K1xx Reference Manual	Rev. 14, 09/2021
4	Datasheet	S32M2xx Data Sheet, Rev. 3 DraftA, 05/2023
5	Datasheet	S32K1xx Data Sheet, Rev. 14, 08/2021
6	S32M244_P64A+P73G Errata	Rev. 0
7	S32M242_N33V+P73G Errata	Rev. 0, 06/2023
8	S32K116_0N96V Errata	Rev. 22/OCT/2021
9	S32K118_0N97V Errata	Rev. 22/OCT/2021
10	S32K142_0N33V Errata	Rev. 22/OCT/2021
11	S32K144_0N57U Errata	Rev. 22/OCT/2021
12	S32K144W_0P64A Errata	Rev. 22/OCT/2021
13	S32K146_0N73V Errata	Rev. 22/OCT/2021
14	S32K148_0N20V Errata	Rev. 22/OCT/2021

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
- IAR Compiler/Assembler/Linker Options
- GHS Compiler/Assembler/Linker Options

The RTD driver files are compiled using:

- NXP GCC 10.2.0 20200723 (Build 1728 Revision g5963bc8)
- IAR ANSI C/C++ Compiler V8.40.3.228/W32 for ARM Functional Safety
- Green Hills Multi 7.1.6d / Compiler 2020.1.4

The compiler, assembler, and linker flags used for building the driver are explained below.
The TS_T40D2M20I0R0 part of the plugin name is composed as follows:

- T = Target_Id (e.g. T40 identifies Cortex-M architecture)
- D = Derivative_Id (e.g. D2 identifies S32K1 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-m4	Targeted ARM processor for which GCC should tune the performance of the code (for S32K14x or S32M24x devices)
-mcpu=cortex-m0plus	Targeted ARM processor for which GCC should tune the performance of the code (for S32K11x devices)
-mthumb	Generates code that executes in Thumb state
-mlittle-endian	Generate code for a processor running in little-endian mode
-mfpu=fpv4-sp-d16	Specifies the floating-point hardware available on the target (for S32K14x or S32M24x devices)
-mfloat-abi=hard	Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions (for S32K14x or S32M24x devices)
-mfpu=auto	Specifies the floating-point hardware available on the target (for S32K11x devices)
-mfloat-abi=soft	Specifies the floating-point ABI to use. Specifying "soft" causes GCC to generate output containing library calls for floating-point operations (for S32K11x devices)
-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 aforementioend -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.

Compiler Option	Description
-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.
-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
-DS32K1XX	Predefine S32K1XX as a macro, with definition 1
-DS32K148	Predefine S32K148 as a macro, with definition 1. S32↩ K148 can be replaced according to derivatives name S32K116,S32K118,S32K142,S32K142W,S32K144,S32↩ K144W,S32K146,S32K148,S32M244,S32M242.
-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
-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 (for S32K14x or S32↩ M24x devices)
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver (for S32K14x or S32M24x devices)
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPO↩ RT as a macro, with definition 1. Allows drivers to be configured in user mode.
–sysroot=	Specifies the path to the sysroot, for Cortex-M7 it is /arm-none-eabi/newlib
-specs=nano.specs	Use Newlib nano specs
-specs=nosys.specs	Do not use printf/scanf

3.1.1.2 GCC Assembler Options

Assembler Option	Description
-Xassembler-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=cortex-m4	Targeted ARM processor for which GCC should tune the performance of the code (for S32K14x or S32M24x devices)
-mcpu=cortex-m0plus	Targeted ARM processor for which GCC should tune the performance of the code (for S32K11x devices)
-mfpu=fpv4-sp-d16	Specifies the floating-point hardware available on the target (for S32K14x devices)
-mfloat-abi=hard	Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions (for S32K14x devices)
-mfpu=auto	Specifies the floating-point hardware available on the target (for S32K11x devices)
-mfloat-abi=soft	Specifies the floating-point ABI to use. Specifying "soft" causes GCC to generate output containing library calls for floating-point operations (for S32K11x devices)
-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=cortex-m4	Targeted ARM processor for which GCC should tune the performance of the code (for S32K14x or S32M24x devices)
-mcpu=cortex-m0plus	Targeted ARM processor for which GCC should tune the performance of the code (for S32K11x devices)
-mthumb	Generates code that executes in Thumb state
-mfpu=fpv4-sp-d16	Specifies the floating-point hardware available on the target (for S32K14x or S32M24x devices)
-mfloat-abi=hard	Specifies the floating-point ABI to use. "hard" allows generation of floating-point instructions and uses FPU-specific calling conventions (for S32K14x or S32M24x devices)
-mfpu=auto	Specifies the floating-point hardware available on the target (for S32K11x devices)
-mfloat-abi=soft	Specifies the floating-point ABI to use. Specifying "soft" causes GCC to generate output containing library calls for floating-point operations (for S32K11x devices)
-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
-n	Turn off page alignment of sections, and disable linking against shared libraries
–sysroot=	Specifies the path to the sysroot, for Cortex-M7 it is /arm-none-eabi/newlib

Linker Option	Description
-specs=nano.specs	Use Newlib nano specs
-specs=nosys.specs	Do not use printf/scanf

3.1.2 IAR Compiler/Assembler/Linker Options

3.1.2.1 IAR Compiler Options

Compiler Option	Description
–cpu=Cortex-M4	Targeted ARM processor for which IAR should tune the performance of the code (for S32K14x or S32M24x devices)
–cpu=Cortex-M0+	Targeted ARM processor for which IAR should tune the performance of the code (for S32K11x devices)
–cpu_mode=thumb	Generates code that executes in Thumb state
–endian=little	Generate code for a processor running in little-endian mode
–fpu=FPv4-SP	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant. (for S32K14x or S32M24x devices)
–fpu=none	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). No FPU. (for S32K11x devices)
-e	Enables all IAR C language extensions
-Ohz	Optimize for size. The compiler will emit AEABI attributes indicating the requested optimization goal. This information can be used by the linker to select smaller or faster variants of DLIB library functions
–debug	Makes the compiler include debugging information in the object modules. Including debug information will make the object files larger
–no_clustering	Disables static clustering optimizations. Static and global variables defined within the same module will not be arranged so that variables that are accessed in the same function are close to each other
–no_mem_idioms	Makes the compiler not optimize certain memory access patterns
–no_explicit_zero_opt	Do not treat explicit initializations to zero of static variables as zero initializations
–require_prototypes	Force the compiler to verify that all functions have proper prototypes. Generates an error otherwise
–no_wrap_diagnostics	Does not wrap long lines in diagnostic messages
–diag_suppress=Pa050	Suppresses diagnostic message Pa050
-DS32K1XX	Predefine S32K1XX as a macro, with definition 1
-DS32K148	Predefine S32K148 as a macro, with definition 1. S32↩ K148 can be replaced according to derivatives name S32K116,S32K118,S32K142,S32K142W,S32K144,S32↩ K144W,S32K146,S32K148,S32M244,S32M242.
-DIAR	Predefine IAR as a macro, with definition 1

Compiler Option	Description
-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.
-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 (for S32K14x or S32↩ M24x devices)
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver (for S32K14x or S32M24x devices)
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPO↩ RT as a macro, with definition 1. Allows drivers to be configured in user mode.

3.1.2.2 IAR Assembler Options

Assembler Option	Description
–cpu=Cortex-M4	Targeted ARM processor for which IAR should tune the performance of the code (for S32K14x or S32M24x devices)
–cpu=Cortex-M0+	Targeted ARM processor for which IAR should tune the performance of the code (for S32K11x devices)
–fpu=FPv4-SP	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant. (for S32K14x devices)
–fpu=none	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). No FPU. (for S32K11x devices)
–cpu_mode thumb	Selects the thumb mode for the assembler directive CODE
-g	Disables the automatic search for system include files
-r	Generates debug information

3.1.2.3 IAR Linker Options

Linker Option	Description
–map filename	Produces a map file
–config linkerfile	Use linkerfile as the linker script. This script replaces the default linker script (rather than adding to it)
–cpu=Cortex-M4	Targeted ARM processor for which IAR should tune the performance of the code (for S32K14x or S32M24x devices)
–cpu=Cortex-M0+	Targeted ARM processor for which IAR should tune the performance of the code (for S32K11x devices)
–fpu=FPv4-SP	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant. (for S32K14x or S32M24x devices)
–fpu=none	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). No FPU. (for S32K11x devices)

Linker Option	Description
–entry _start	Treats _start as a root symbol and start label
–enable_stack_usage	Enables stack usage analysis. If a linker map file is produced, a stack usage chapter is included in the map file
–skip_dynamic_initialization	Dynamic initialization (typically initialization of C++ objects with static storage duration) will not be performed automatically during application startup
–no_wrap_diagnostics	Does not wrap long lines in diagnostic messages

3.1.3 GHS Compiler/Assembler/Linker Options

3.1.3.1 GHS Compiler Options

Compiler Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4 (for S32K14x or S32M24x devices)
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+ (for S32K11x devices)
-thumb	Selects generating code that executes in Thumb state
-fpu=vfpv4_d16	Specifies hardware floating-point using the v4 version of the VFP instruction set, with 16 double-precision floating-point registers (for S32K14x or S32M24x devices)
-fsingle	Use hardware single-precision, software double-precision FP instructions (for S32K14x or S32M24x devices)
-fsoft	Specifies software floating-point (SFP) mode. This setting causes your target to use integer registers to hold floating-point data and use library subroutine calls to emulate floating-point operations (for S32K11x devices)
-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
-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

Compiler Option	Description
–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
-DS32K1XX	Predefine S32K1XX as a macro, with definition 1
-DS32K148	Predefine S32K148 as a macro, with definition 1. S32↩ K148 can be replaced according to derivatives name S32K116,S32K118,S32K142,S32K142W,S32K144,S32↩ K144W,S32K146,S32K148,S32M244,S32M242.
-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
-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 (for S32K14x or S32↩ M24x devices)
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initialization in source file system.c under the Platform driver (for S32K14x or S32M24x devices)
-DMCAL_ENABLE_USER_MODE_SUPPORT	Predefine MCAL_ENABLE_USER_MODE_SUPPO↩ RT as a macro, with definition 1. Allows drivers to be configured in user mode

3.1.3.2 GHS Assembler Options

Assembler Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4 (for S32K14x or S32M24x devices)
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+ (for S32K11x devices)
-fpu=vfpv4_d16	Specifies hardware floating-point using the v4 version of the VFP instruction set, with 16 double-precision floating-point registers (for S32K14x devices)
-fsingle	Use hardware single-precision, software double-precision FP instructions (for S32↩ K14x devices)
-fsoft	Specifies software floating-point (SFP) mode. This setting causes your target to use integer registers to hold floating-point data and use library subroutine calls to emulate floating-point operations (for S32K11x devices)
-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

Assembler Option	Description
-c	Stop after assembly and produce an object file for each source file

3.1.3.3 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 the modules 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 (for S32K14x or S32M24x devices)
-lind_sf	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 (for S32K11x devices)
-v	Prints verbose information about the activities of the linker, including the libraries it searches to resolve undefined symbols
-keep=C40_Ip_AccessCode	Avoid linker remove function C40_Ip_AccessCode from Fls module because it is not referenced explicitly
-nostartfiles	Controls the start files to be linked into the executable

3.2 Files required for compilation

This section describes the include files required to compile, assemble (if assembler code) and link the ADC driver for S32K1XX and S32M24X 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.

Adc Files

- ..\Adc_TS_T40D2M20I0R0\include\Adc.h

- ..\Adc_TS_T40D2M20I0R0\include\Adc_Ip.h
- ..\Adc_TS_T40D2M20I0R0\include\Adc_Ip_HeaderWrapper_S32K14x_Extended.h
- ..\Adc_TS_T40D2M20I0R0\include\Adc_Ip_HeaderWrapper_S32K1xx.h
- ..\Adc_TS_T40D2M20I0R0\include\Adc_Ip_HwAccess.h
- ..\Adc_TS_T40D2M20I0R0\include\Adc_Ip_Irq.h
- ..\Adc_TS_T40D2M20I0R0\include\Adc_Ip_TrustedFunctions.h
- ..\Adc_TS_T40D2M20I0R0\include\Adc_Ip_Types.h
- ..\Adc_TS_T40D2M20I0R0\include\Adc_Ipw.h
- ..\Adc_TS_T40D2M20I0R0\include\Adc_Ipw_Irq.h
- ..\Adc_TS_T40D2M20I0R0\include\Adc_Ipw_Types.h
- ..\Adc_TS_T40D2M20I0R0\include\Adc_Types.h
- ..\Adc_TS_T40D2M20I0R0\include\Pdb_Adc_Ip.h
- ..\Adc_TS_T40D2M20I0R0\include\Pdb_Adc_Ip_HwAccess.h
- ..\Adc_TS_T40D2M20I0R0\include\Pdb_Adc_Ip_Irq.h
- ..\Adc_TS_T40D2M20I0R0\include\Pdb_Adc_Ip_TrustedFunctions.h
- ..\Adc_TS_T40D2M20I0R0\include\Pdb_Adc_Ip_Types.h
- ..\Adc_TS_T40D2M20I0R0\src\Adc.c
- ..\Adc_TS_T40D2M20I0R0\src\Adc_Ip.c
- ..\Adc_TS_T40D2M20I0R0\src\Adc_Ip_Irq.c
- ..\Adc_TS_T40D2M20I0R0\src\Adc_Ipw.c
- ..\Adc_TS_T40D2M20I0R0\src\Adc_Ipw_Irq.c
- ..\Adc_TS_T40D2M20I0R0\src\Pdb_Adc_Ip.c
- ..\Adc_TS_T40D2M20I0R0\src\Pdb_Adc_Ip_Irq.c

Adc Generated Files

- Adc_Cfg.c
- Adc_Cfg.h
- Adc_PBcfg.c
- Adc_PBcfg.h
- Adc_CfgDefines.h
- Adc_Ipw_Cfg.h
- Adc_Ipw_CfgDefines.h
- Adc_Ipw_PBcfg.c

- Adc_Ipw_PBcfg.h
- Adc_Ip_Cfg.h
- Adc_Ip_CfgDefines.h
- Adc_Ip_PBcfg.c
- Adc_Ip_PBcfg.h
- Pdb_Adc_Ip_Cfg.h
- Pdb_Adc_Ip_CfgDefines.h
- Pdb_Adc_Ip_PBcfg.c
- Pdb_Adc_Ip_PBcfg.h

Files from Base common folder

- ..\Base_TS_T40D2M19I0R0\include\Adc_MemMap.h
- ..\Base_TS_T40D2M19I0R0\include\Compiler.h
- ..\Base_TS_T40D2M19I0R0\include\Compiler_Cfg.h
- ..\Base_TS_T40D2M19I0R0\include\ComStack_Cfg.h
- ..\Base_TS_T40D2M19I0R0\include\ComStackTypes.h
- ..\Base_TS_T40D2M19I0R0\include\Mcal.h
- ..\Base_TS_T40D2M19I0R0\include\OsIf.h
- ..\Base_TS_T40D2M19I0R0\include\OsIf_Cfg_TypesDef.h
- ..\Base_TS_T40D2M19I0R0\include\OsIf_DeviceRegisters.h
- ..\Base_TS_T40D2M19I0R0\include\OsIf_Timer_System_Internal_Systick.h
- ..\Base_TS_T40D2M19I0R0\include\RegLockMacros.h
- ..\Base_TS_T40D2M19I0R0\include\SilRegMacros.h
- ..\Base_TS_T40D2M19I0R0\include\Soc_Ips.h
- ..\Base_TS_T40D2M19I0R0\include\Platform_Types.h
- ..\Base_TS_T40D2M19I0R0\include\StandardTypes.h
- ..\Base_TS_T40D2M19I0R0\include\Reg_eSys.h
- ..\Base_TS_T40D2M19I0R0\generate_PC\include\Mcal.h
- ..\Base_TS_T40D2M19I0R0\generate_PC\include\OsIf_Cfg.h
- ..\Base_TS_T40D2M19I0R0\header

Files from Det folder

- ..\Det_TS_T40D2M19I0R0\include\Det.h
- ..\Det_TS_T40D2M19I0R0\src\Det.c

Files from Mcl folder

- ..\Mcl_TS_T40D2M19I0R0\include\CDD_Mcl.h
- ..\Mcl_TS_T40D2M19I0R0\include\Mcl.h
- ..\Mcl_TS_T40D2M19I0R0\include\Dma_Ip.h
- ..\Mcl_TS_T40D2M19I0R0\include\Dma_Ip_Types.h
- ..\Mcl_TS_T40D2M19I0R0\include\Mcl_Types.h
- ..\Mcl_TS_T40D2M19I0R0\src\CDD_Mcl.c
- ..\Mcl_TS_T40D2M19I0R0\src\Dma_Ip.c
- ..\Mcl_TS_T40D2M19I0R0\src\Dma_Ip_Irq.c
- ..\Mcl_TS_T40D2M19I0R0\generate_PC\src\CDD_Mcl_Cfg.c
- ..\Mcl_TS_T40D2M19I0R0\generate_PB\src\CDD_Mcl_PBcfg.c

Files from Rm folder

- ..\Rm_TS_T40D2M19I0R0\include\CDD_Rm.h
- ..\Rm_TS_T40D2M19I0R0\include\Dma_Mux_Ip.h
- ..\Rm_TS_T40D2M19I0R0\include\Dma_Mux_Ip_Types.h
- ..\Rm_TS_T40D2M19I0R0\src\CDD_Rm.c
- ..\Rm_TS_T40D2M19I0R0\src\Dma_Mux_Ip.c
- ..\Rm_TS_T40D2M19I0R0\generate_PC\src\CDD_Rm_Cfg.c
- ..\Rm_TS_T40D2M19I0R0\generate_PB\src\CDD_Rm_PBcfg.c

3.3 Setting up the plugins

The ADC driver was designed to be configured by using the EB Tresos Studio (version EB tresos Studio 29.0.0 or later.)

Location of various files inside the ADC module folder

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\Adc_TS_T40D2M20I0R0\config\Adc.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\Adc_TS_T40D2M20I0R0\autosar\Adc_s32m242_lqfp64.epd
  - ..\Adc_TS_T40D2M20I0R0\autosar\Adc_s32m244_lqfp64.epd
- Code Generation Templates for parameters without variation points:
  - ..\Adc_TS_T40D2M20I0R0\output\include\Adc_Cfg.h
  - ..\Adc_TS_T40D2M20I0R0\output\include\Adc_CfgDefines.h
  - ..\Adc_TS_T40D2M20I0R0\output\include\Adc_Ip_Cfg.h
  - ..\Adc_TS_T40D2M20I0R0\output\include\Adc_Ip_CfgDefines.h
  - ..\Adc_TS_T40D2M20I0R0\output\include\Adc_Ipw_Cfg.h
  - ..\Adc_TS_T40D2M20I0R0\output\include\Adc_Ipw_CfgDefines.h
  - ..\Adc_TS_T40D2M20I0R0\output\include\Pdb_Adc_Ip_Cfg.h
  - ..\Adc_TS_T40D2M20I0R0\output\include\Pdb_Adc_Ip_CfgDefines.h
  - ..\Adc_TS_T40D2M20I0R0\output\src\Adc_Cfg.c
- Code Generation Templates for for variant aware parameters:
  - ..\Adc_TS_T40D2M20I0R0\output\include\Adc_<VariantName>_PBcfg.h
  - ..\Adc_TS_T40D2M20I0R0\output\src\Adc_<VariantName>_PBcfg.c
  - ..\Adc_TS_T40D2M20I0R0\output\include\Adc_Ipw_<VariantName>_PBcfg.h
  - ..\Adc_TS_T40D2M20I0R0\output\src\Adc_Ipw_<VariantName>_PBcfg.c
  - ..\Adc_TS_T40D2M20I0R0\output\include\Adc_Ip_<VariantName>_PBcfg.h
  - ..\Adc_TS_T40D2M20I0R0\output\src\Adc_Ip_<VariantName>_PBcfg.c
  - ..\Adc_TS_T40D2M20I0R0\output\include\Pdb_Adc_Ip_<VariantName>_PBcfg.h
  - ..\Adc_TS_T40D2M20I0R0\output\src\Pdb_Adc_Ip_<VariantName>_PBcfg.c

Steps to generate the configuration:

1. Copy the module folders Adc_TS_T40D2M20I0R0, BaseNXP_TS_T40D2M20I0R0, Det_TS_T40D2M20↩ I0R0, Resource_TS_T40D2M20I0R0, Os_TS_T40D2M20I0R0, EcuM_TS_T40D2M20I0R0, Mcl_TS_↩ T40D2M20I0R0, Mcu_TS_T40D2M20I0R0, Rte_TS_T40D2M20I0R0, Ecuc_TS_T40D2M20I0R0, Rm_↩ TS_T40D2M20I0R0 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.

Chapter 4

Function calls to module

4.1 Function Calls during Start-up

Adc shall be initialized during STARTUP phase of EcuM initialization. The API to be called for this is Adc_Init(). The MCU module and the PORT module should be initialized before the Adc is initialized.

Note

Adc_Init() should be called before starting any ADC conversion or calling function Adc_SetupResultBuffer.

4.2 Function Calls during Shutdown

None.

4.3 Function Calls during Wake-up

None.

Chapter 5

Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

In the current implementation, ADC driver is using the services of Schedule Manager (SchM) for entering and exiting the critical regions, to preserve a resource. SchM implementation is done by the integrators of the RTD using OS or non-OS services. For testing the ADC, stubs are used for SchM. The following critical regions are used in the ADC driver:

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

ADC_EXCLUSIVE_AREA_00 is used in function Adc_StopGroupConversion to protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_00 is used in function Adc_ReadGroup to protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_01 is used in function Adc_StartGroupConversion to protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_02 is used in function Adc_StopGroupConversion to protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_03 is used in function Adc_ReadGroup to protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Init to protect the ADC_CFG1 register from read/modify/write operation in Adc_Ip_SetResolution

ADC_EXCLUSIVE_AREA_11 is used in function Adc_SetClockMode to protect the ADC_CFG1 register from read/modify/write operation in Adc_Ip_SetClockMode

ADC_EXCLUSIVE_AREA_13 is used in function Adc_Calibrate to protect the ADC_CFG1 register from read/modify/write operation in Adc_Ip_DoCalibration

ADC_EXCLUSIVE_AREA_14 is used in function Adc_SetClockMode to protect the ADC_CFG2 register from read/modify/write operation in Adc_Ip_SetClockMode

ADC_EXCLUSIVE_AREA_15 is used in function Adc_Calibrate to protect the ADC_CFG2 register from read/modify/write operation in Adc_Ip_DoCalibration

ADC_EXCLUSIVE_AREA_16 is used in function Adc_StartGroupConversion to protect the ADC_CFG2 register from read/modify/write operation in Adc_Ip_SetSampleTime

ADC_EXCLUSIVE_AREA_16 is used in function Adc_StopGroupConversion to protect the ADC_CFG2 register from read/modify/write operation in Adc_Ip_SetSampleTime

ADC_EXCLUSIVE_AREA_16 is used in function Adc_ReadGroup to protect the ADC_CFG2 register from read/modify/write operation in Adc_Ip_SetSampleTime

ADC_EXCLUSIVE_AREA_16 is used in function Adc_EnableHardwareTrigger to protect the ADC_CFG2 register from read/modify/write operation in Adc_Ip_SetSampleTime

ADC_EXCLUSIVE_AREA_17 is used in function Adc_StartGroupConversion to protect the ADC_SC1 register from read/modify/write operation in Adc_Ip_SetDisabledChannel

ADC_EXCLUSIVE_AREA_17 is used in function Adc_StopGroupConversion to protect the ADC_SC1 register from read/modify/write operation in Adc_Ip_SetDisabledChannel

ADC_EXCLUSIVE_AREA_17 is used in function Adc_ReadGroup to protect the ADC_SC1 register from read/modify/write operation in Adc_Ip_SetDisabledChannel

ADC_EXCLUSIVE_AREA_21 is used in function Adc_StartGroupConversion to protect the ADC_SC1 register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_21 is used in function Adc_StopGroupConversion to protect the ADC_SC1 register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_21 is used in function Adc_ReadGroup to protect the ADC_SC1 register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_21 is used in function Adc_EnableHardwareTrigger to protect the ADC_SC1 register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_21 is used in function Adc_DisableHardwareTrigger to protect the ADC_SC1 register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_22 is used in function Adc_StartGroupConversion to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_DisableDma

ADC_EXCLUSIVE_AREA_22 is used in function Adc_StopGroupConversion to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_DisableDma

ADC_EXCLUSIVE_AREA_22 is used in function Adc_ReadGroup to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_DisableDma

ADC_EXCLUSIVE_AREA_22 is used in function Adc_DisableHardwareTrigger to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_DisableDma

ADC_EXCLUSIVE_AREA_23 is used in function Adc_StartGroupConversion to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_SetTriggerMode

ADC_EXCLUSIVE_AREA_23 is used in function Adc_StopGroupConversion to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_SetTriggerMode

ADC_EXCLUSIVE_AREA_23 is used in function Adc_ReadGroup to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_SetTriggerMode

ADC_EXCLUSIVE_AREA_24 is used in function Adc_Calibrate to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_DoCalibration

ADC_EXCLUSIVE_AREA_26 is used in function Adc_StartGroupConversion to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_EnableDma

ADC_EXCLUSIVE_AREA_26 is used in function Adc_StopGroupConversion to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_EnableDma

ADC_EXCLUSIVE_AREA_26 is used in function Adc_ReadGroup to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_EnableDma

ADC_EXCLUSIVE_AREA_26 is used in function Adc_EnableHardwareTrigger to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_EnableDma

ADC_EXCLUSIVE_AREA_27 is used in function Adc_StartGroupConversion to protect the ADC_SC3 register from read/modify/write operation in Adc_Ip_SetAveraging

ADC_EXCLUSIVE_AREA_27 is used in function Adc_StopGroupConversion to protect the ADC_SC3 register from read/modify/write operation in Adc_Ip_SetAveraging

ADC_EXCLUSIVE_AREA_27 is used in function Adc_ReadGroup to protect the ADC_SC3 register from read/modify/write operation in Adc_Ip_SetAveraging

ADC_EXCLUSIVE_AREA_27 is used in function Adc_EnableHardwareTrigger to protect the ADC_SC3 register from read/modify/write operation in Adc_Ip_SetAveraging

ADC_EXCLUSIVE_AREA_28 is used in function Adc_SetClockMode to protect the ADC_SC3 register from read/modify/write operation in Adc_Ip_SetClockMode

ADC_EXCLUSIVE_AREA_29 is used in function Adc_Calibrate to protect the ADC_SC3 register from read/modify/write operation in Adc_Ip_DoCalibration

ADC_EXCLUSIVE_AREA_30 is used in function Adc_StartGroupConversion to protect the ADC_SC3 register from read/modify/write operation in Adc_Ip_SetContinuousMode

ADC_EXCLUSIVE_AREA_30 is used in function Adc_StopGroupConversion to protect the ADC_SC3 register from read/modify/write operation in Adc_Ip_SetContinuousMode

ADC_EXCLUSIVE_AREA_32 is used in function Adc_StartGroupConversion to protect the SIM_CHIPCTL register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_32 is used in function Adc_StopGroupConversion to protect the SIM_CHIPCTL register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_32 is used in function Adc_ReadGroup to protect the SIM_CHIPCTL register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_32 is used in function Adc_EnableHardwareTrigger to protect the SIM_CHIP↩ CTL register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_32 is used in function Adc_DisableHardwareTrigger to protect the SIM_CHI↩ PCTL register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_34 is used in function Adc_StartGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Enable

ADC_EXCLUSIVE_AREA_34 is used in function Adc_StopGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Enable

ADC_EXCLUSIVE_AREA_34 is used in function Adc_ReadGroup to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Enable

ADC_EXCLUSIVE_AREA_34 is used in function Adc_EnableHardwareTrigger to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Enable

ADC_EXCLUSIVE_AREA_35 is used in function Adc_StartGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Disable

ADC_EXCLUSIVE_AREA_35 is used in function Adc_StopGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Disable

ADC_EXCLUSIVE_AREA_35 is used in function Adc_ReadGroup to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Disable

ADC_EXCLUSIVE_AREA_35 is used in function Adc_EnableHardwareTrigger to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Disable

ADC_EXCLUSIVE_AREA_36 is used in function Adc_StartGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetTriggerInput

ADC_EXCLUSIVE_AREA_36 is used in function Adc_StopGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetTriggerInput

ADC_EXCLUSIVE_AREA_36 is used in function Adc_ReadGroup to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetTriggerInput

ADC_EXCLUSIVE_AREA_36 is used in function Adc_EnableHardwareTrigger to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetTriggerInput

ADC_EXCLUSIVE_AREA_37 is used in function Adc_StartGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetContinuousMode

ADC_EXCLUSIVE_AREA_37 is used in function Adc_StopGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetContinuousMode

ADC_EXCLUSIVE_AREA_37 is used in function Adc_ReadGroup to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetContinuousMode

ADC_EXCLUSIVE_AREA_38 is used in function Adc_StartGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SwTrigger

ADC_EXCLUSIVE_AREA_38 is used in function Adc_StopGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SwTrigger

ADC_EXCLUSIVE_AREA_38 is used in function Adc_ReadGroup to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SwTrigger

ADC_EXCLUSIVE_AREA_38 is used in function Adc_EnableHardwareTrigger to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SwTrigger

ADC_EXCLUSIVE_AREA_39 is used in function Adc_StartGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_LoadRegValues

ADC_EXCLUSIVE_AREA_39 is used in function Adc_StopGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_LoadRegValues

ADC_EXCLUSIVE_AREA_39 is used in function Adc_ReadGroup to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_LoadRegValues

ADC_EXCLUSIVE_AREA_39 is used in function Adc_EnableHardwareTrigger to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_LoadRegValues

ADC_EXCLUSIVE_AREA_40 is used in function Adc_StartGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_40 is used in function Adc_StopGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_40 is used in function Adc_ReadGroup to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_40 is used in function Adc_EnableHardwareTrigger to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_40 is used in function Adc_DisableHardwareTrigger to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_41 is used in function Adc_StartGroupConversion to protect the PDB_C1 register from read/modify/write operation in Pdb_Adc_Ip_ConfigAdcPretriggers

ADC_EXCLUSIVE_AREA_41 is used in function Adc_StopGroupConversion to protect the PDB_C1 register from read/modify/write operation in Pdb_Adc_Ip_ConfigAdcPretriggers

ADC_EXCLUSIVE_AREA_41 is used in function Adc_ReadGroup to protect the PDB_C1 register from read/modify/write operation in Pdb_Adc_Ip_ConfigAdcPretriggers

ADC_EXCLUSIVE_AREA_41 is used in function Adc_EnableHardwareTrigger to protect the PDB_C1 register from read/modify/write operation in Pdb_Adc_Ip_ConfigAdcPretriggers

ADC_EXCLUSIVE_AREA_45 is used in function Adc_StartGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_45 is used in function Adc_StopGroupConversion to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_45 is used in function Adc_ReadGroup to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_45 is used in function Adc_EnableHardwareTrigger to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_45 is used in function Adc_DisableHardwareTrigger to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_47 is used in function Adc_StartGroupConversion to protect the PDB_S register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_47 is used in function Adc_StopGroupConversion to protect the PDB_S register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_47 is used in function Adc_ReadGroup to protect the PDB_S register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_47 is used in function Adc_EnableHardwareTrigger to protect the PDB_S register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_47 is used in function Adc_DisableHardwareTrigger to protect the PDB_S register from read/modify/write operation in Pdb_Adc_Ip_DisableAndClearPdb

ADC_EXCLUSIVE_AREA_48 is used in function Adc_StartGroupConversion to protect the PDB_MOD register from read/modify/write operation in Pdb_Adc_Ip_SetModulus

ADC_EXCLUSIVE_AREA_48 is used in function Adc_StopGroupConversion to protect the PDB_MOD register from read/modify/write operation in Pdb_Adc_Ip_SetModulus

ADC_EXCLUSIVE_AREA_48 is used in function Adc_ReadGroup to protect the PDB_MOD register from read/modify/write operation in Pdb_Adc_Ip_SetModulus

ADC_EXCLUSIVE_AREA_48 is used in function Adc_EnableHardwareTrigger to protect the PDB_MOD register from read/modify/write operation in Pdb_Adc_Ip_SetModulus

Exclusive Areas are used in Interrupt service request (ISR)

ADC_EXCLUSIVE_AREA_00 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_00 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_00 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_00 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_16 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the ADC_CFG2 register from read/modify/write operation in Adc_Ip_SetSampleTime

ADC_EXCLUSIVE_AREA_16 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_CFG2 register from read/modify/write operation in Adc_Ip_SetSampleTime

ADC_EXCLUSIVE_AREA_16 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the ADC_CFG2 register from read/modify/write operation in Adc_Ip_SetSampleTime

ADC_EXCLUSIVE_AREA_16 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_CFG2 register from read/modify/write operation in Adc_Ip_SetSampleTime

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the ADC_SC1 register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_SC1 register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the ADC_SC1 register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_SC1 register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_22 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_DisableDma

ADC_EXCLUSIVE_AREA_22 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_DisableDma

ADC_EXCLUSIVE_AREA_22 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_DisableDma

ADC_EXCLUSIVE_AREA_22 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_DisableDma

ADC_EXCLUSIVE_AREA_23 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_SetTriggerMode

ADC_EXCLUSIVE_AREA_23 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_SetTriggerMode

ADC_EXCLUSIVE_AREA_23 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_SetTriggerMode

ADC_EXCLUSIVE_AREA_23 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_SetTriggerMode

ADC_EXCLUSIVE_AREA_26 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_EnableDma

ADC_EXCLUSIVE_AREA_26 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_EnableDma

ADC_EXCLUSIVE_AREA_26 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_EnableDma

ADC_EXCLUSIVE_AREA_26 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_SC2 register from read/modify/write operation in Adc_Ip_EnableDma

ADC_EXCLUSIVE_AREA_27 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the ADC_SC3 register from read/modify/write operation in Adc_Ip_SetAveraging

ADC_EXCLUSIVE_AREA_27 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_SC3 register from read/modify/write operation in Adc_Ip_SetAveraging

ADC_EXCLUSIVE_AREA_27 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the ADC_SC3 register from read/modify/write operation in Adc_Ip_SetAveraging

ADC_EXCLUSIVE_AREA_27 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_SC3 register from read/modify/write operation in Adc_Ip_SetAveraging

ADC_EXCLUSIVE_AREA_32 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the SIM_CHIPCTL register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_32 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the SIM_CHIPCTL register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_32 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the SIM_CHIPCTL register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_32 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the SIM_CHIPCTL register from read/modify/write operation in Adc_Ip_ConfigChannel

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Enable

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Enable

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Enable

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Enable

ADC_EXCLUSIVE_AREA_35 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Disable

ADC_EXCLUSIVE_AREA_35 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Disable

ADC_EXCLUSIVE_AREA_35 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Disable

ADC_EXCLUSIVE_AREA_35 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_Disable

ADC_EXCLUSIVE_AREA_36 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetTriggerInput

ADC_EXCLUSIVE_AREA_36 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetTriggerInput

ADC_EXCLUSIVE_AREA_36 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetTriggerInput

ADC_EXCLUSIVE_AREA_36 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetTriggerInput

ADC_EXCLUSIVE_AREA_37 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetContinuousMode

ADC_EXCLUSIVE_AREA_37 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetContinuousMode

ADC_EXCLUSIVE_AREA_37 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetContinuousMode

ADC_EXCLUSIVE_AREA_37 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SetContinuousMode

ADC_EXCLUSIVE_AREA_38 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SwTrigger

ADC_EXCLUSIVE_AREA_38 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SwTrigger

ADC_EXCLUSIVE_AREA_38 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SwTrigger

ADC_EXCLUSIVE_AREA_38 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_SwTrigger

ADC_EXCLUSIVE_AREA_39 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_LoadRegValues

ADC_EXCLUSIVE_AREA_39 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_LoadRegValues

ADC_EXCLUSIVE_AREA_39 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_LoadRegValues

ADC_EXCLUSIVE_AREA_39 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the PDB_SC register from read/modify/write operation in Pdb_Adc_Ip_LoadRegValues

ADC_EXCLUSIVE_AREA_41 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the PDB_C1 register from read/modify/write operation in Pdb_Adc_Ip_ConfigAdcPretriggers

ADC_EXCLUSIVE_AREA_41 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the PDB_C1 register from read/modify/write operation in Pdb_Adc_Ip_ConfigAdcPretriggers

ADC_EXCLUSIVE_AREA_41 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the PDB_C1 register from read/modify/write operation in Pdb_Adc_Ip_ConfigAdcPretriggers

ADC_EXCLUSIVE_AREA_41 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the PDB_C1 register from read/modify/write operation in Pdb_Adc_Ip_ConfigAdcPretriggers

ADC_EXCLUSIVE_AREA_48 is used in function Adc_Ipw_Adc0EndConversionNotification to protect the PDB_MOD register from read/modify/write operation in Pdb_Adc_Ip_SetModulus

ADC_EXCLUSIVE_AREA_48 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the PDB_MOD register from read/modify/write operation in Pdb_Adc_Ip_SetModulus

ADC_EXCLUSIVE_AREA_48 is used in function Adc_Ipw_Adc1EndConversionNotification to protect the PDB_MOD register from read/modify/write operation in Pdb_Adc_Ip_SetModulus

ADC_EXCLUSIVE_AREA_48 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the PDB_MOD register from read/modify/write operation in Pdb_Adc_Ip_SetModulus

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

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Ip_SetResolution to protect the updates for ADC↩_CFG1 register

ADC_EXCLUSIVE_AREA_11 is used in function Adc_Ip_SetClockMode to protect the updates for ADC↩_CFG1 register

ADC_EXCLUSIVE_AREA_12 is used in function Adc_Ip_ClearLatchedTriggers to protect the updates for ADC_CFG1 register

ADC_EXCLUSIVE_AREA_13 is used in function Adc_Ip_DoCalibration to protect the updates for ADC↩_CFG1 register

ADC_EXCLUSIVE_AREA_14 is used in function Adc_Ip_SetClockMode to protect the updates for ADC↩_CFG2 register

ADC_EXCLUSIVE_AREA_15 is used in function Adc_Ip_DoCalibration to protect the updates for ADC↩_CFG2 register

ADC_EXCLUSIVE_AREA_16 is used in function Adc_Ip_SetSampleTime to protect the updates for AD↩C_CFG2 register

ADC_EXCLUSIVE_AREA_17 is used in function Adc_Ip_SetDisabledChannel to protect the updates for ADC_SC1 register

ADC_EXCLUSIVE_AREA_18 is used in function Adc_Ip_StartConversion to protect the updates for ADC_SC1 register

ADC_EXCLUSIVE_AREA_19 is used in function Adc_Ip_EnableChannelNotification to protect the updates for ADC_SC1 register

ADC_EXCLUSIVE_AREA_20 is used in function Adc_Ip_DisableChannelNotification to protect the updates for ADC_SC1 register

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ip_ConfigChannel to protect the updates for ADC_SC1 register

ADC_EXCLUSIVE_AREA_22 is used in function Adc_Ip_DisableDma to protect the updates for ADC_SC2 register

ADC_EXCLUSIVE_AREA_23 is used in function Adc_Ip_SetTriggerMode to protect the updates for ADC_SC2 register

ADC_EXCLUSIVE_AREA_24 is used in function Adc_Ip_DoCalibration to protect the updates for ADC_SC2 register

ADC_EXCLUSIVE_AREA_25 is used in function Adc_Ip_ClearTrigErrReg to protect the updates for ADC_SC2 register

ADC_EXCLUSIVE_AREA_26 is used in function Adc_Ip_EnableDma to protect the updates for ADC_SC2 register

ADC_EXCLUSIVE_AREA_27 is used in function Adc_Ip_SetAveraging to protect the updates for ADC_SC3 register

ADC_EXCLUSIVE_AREA_28 is used in function Adc_Ip_SetClockMode to protect the updates for ADC_SC3 register

ADC_EXCLUSIVE_AREA_29 is used in function Adc_Ip_DoCalibration to protect the updates for ADC_SC3 register

ADC_EXCLUSIVE_AREA_30 is used in function Adc_Ip_SetContinuousMode to protect the updates for ADC_SC3 register

ADC_EXCLUSIVE_AREA_31 is used in function Adc_Ip_SetSoftwarePretrigger to protect the updates for SIM_ADCOPT register

ADC_EXCLUSIVE_AREA_32 is used in function Adc_Ip_ConfigChannel to protect the updates for SIM_CHIPCTL register

ADC_EXCLUSIVE_AREA_34 is used in function Pdb_Adc_Ip_Enable to protect the updates for PDB_SC register

ADC_EXCLUSIVE_AREA_35 is used in function Pdb_Adc_Ip_Disable to protect the updates for PDB_SC register

ADC_EXCLUSIVE_AREA_36 is used in function Pdb_Adc_Ip_SetTriggerInput to protect the updates for PDB_SC register

ADC_EXCLUSIVE_AREA_37 is used in function Pdb_Adc_Ip_SetContinuousMode to protect the updates for PDB_SC register

ADC_EXCLUSIVE_AREA_38 is used in function Pdb_Adc_Ip_SwTrigger to protect the updates for PDB_SC register

ADC_EXCLUSIVE_AREA_39 is used in function Pdb_Adc_Ip_LoadRegValues to protect the updates for PDB_SC register

ADC_EXCLUSIVE_AREA_40 is used in function Pdb_Adc_Ip_DisableAndClearPdb to protect the updates for PDB_SC register

ADC_EXCLUSIVE_AREA_41 is used in function Pdb_Adc_Ip_ConfigAdcPretriggers to protect the updates for PDB_C1 register

ADC_EXCLUSIVE_AREA_42 is used in function Pdb_Adc_Ip_SetAdcPretriggerBackToBack to protect the updates for PDB_C1 register

ADC_EXCLUSIVE_AREA_43 is used in function Pdb_Adc_Ip_SetAdcPretriggerEnable to protect the updates for PDB_C1 register

ADC_EXCLUSIVE_AREA_44 is used in function Pdb_Adc_Ip_SetAdcPretriggerDelayEnable to protect the updates for PDB_C1 register

ADC_EXCLUSIVE_AREA_45 is used in function Pdb_Adc_Ip_DisableAndClearPdb to protect the updates for PDB_C1 register

ADC_EXCLUSIVE_AREA_46 is used in function Pdb_Adc_Ip_ClearAdcPretriggerFlags to protect the updates for PDB_S register

ADC_EXCLUSIVE_AREA_47 is used in function Pdb_Adc_Ip_DisableAndClearPdb to protect the updates for PDB_S register

ADC_EXCLUSIVE_AREA_48 is used in function Pdb_Adc_Ip_SetModulus to protect the updates for PDB_MOD register

Exclusive Area Matrix

Exclusive Area ID	ADC_EXCLUSIVE_AREA_00	ADC_EXCLUSIVE_AREA_01	ADC_EXCLUSIVE_AREA_02	ADC_EXCLUSIVE_AREA_03	ADC_EXCLUSIVE_AREA_10	ADC_EXCLUSIVE_AREA_11	ADC_EXCLUSIVE_AREA_12	ADC_EXCLUSIVE_AREA_13	ADC_EXCLUSIVE_AREA_14	ADC_EXCLUSIVE_AREA_15	ADC_EXCLUSIVE_AREA_16	ADC_EXCLUSIVE_AREA_17	ADC_EXCLUSIVE_AREA_18	ADC_EXCLUSIVE_AREA_19	ADC_EXCLUSIVE_AREA_20	ADC_EXCLUSIVE_AREA_21	ADC_EXCLUSIVE_AREA_22	ADC_EXCLUSIVE_AREA_23	ADC_EXCLUSIVE_AREA_24	ADC_EXCLUSIVE_AREA_25	ADC_EXCLUSIVE_AREA_26	ADC_EXCLUSIVE_AREA_27	ADC_EXCLUSIVE_AREA_28	ADC_EXCLUSIVE_AREA_29	ADC_EXCLUSIVE_AREA_30	ADC_EXCLUSIVE_AREA_31	ADC_EXCLUSIVE_AREA_32	ADC_EXCLUSIVE_AREA_34	ADC_EXCLUSIVE_AREA_35	ADC_EXCLUSIVE_AREA_36	ADC_EXCLUSIVE_AREA_37	ADC_EXCLUSIVE_AREA_38	ADC_EXCLUSIVE_AREA_39	ADC_EXCLUSIVE_AREA_40	ADC_EXCLUSIVE_AREA_41	ADC_EXCLUSIVE_AREA_42	ADC_EXCLUSIVE_AREA_43	ADC_EXCLUSIVE_AREA_44	ADC_EXCLUSIVE_AREA_45	ADC_EXCLUSIVE_AREA_46	ADC_EXCLUSIVE_AREA_47	ADC_EXCLUSIVE_AREA_48
ADC_EXCLUSIVE_AREA_00	x	x	x	x																																						
ADC_EXCLUSIVE_AREA_01	x	x	x	x																																						
ADC_EXCLUSIVE_AREA_02	x	x	x	x																																						
ADC_EXCLUSIVE_AREA_03	x	x	x	x																																						
ADC_EXCLUSIVE_AREA_10					x	x	x	x																																		
ADC_EXCLUSIVE_AREA_11					x	x	x	x																																		
ADC_EXCLUSIVE_AREA_12					x	x	x	x																																		
ADC_EXCLUSIVE_AREA_13					x	x	x	x																																		
ADC_EXCLUSIVE_AREA_14									x	x	x																															
ADC_EXCLUSIVE_AREA_15									x	x	x																															
ADC_EXCLUSIVE_AREA_16									x	x	x																															
ADC_EXCLUSIVE_AREA_17												x	x	x	x	x																										
ADC_EXCLUSIVE_AREA_18												x	x	x	x	x																										
ADC_EXCLUSIVE_AREA_19												x	x	x	x	x																										
ADC_EXCLUSIVE_AREA_20												x	x	x	x	x																										
ADC_EXCLUSIVE_AREA_21												x	x	x	x	x																										
ADC_EXCLUSIVE_AREA_22																	x	x	x	x	x																					
ADC_EXCLUSIVE_AREA_23																	x	x	x	x	x																					
ADC_EXCLUSIVE_AREA_24																	x	x	x	x	x																					
ADC_EXCLUSIVE_AREA_25																	x	x	x	x	x																					
ADC_EXCLUSIVE_AREA_26																	x	x	x	x	x																					
ADC_EXCLUSIVE_AREA_27																						x	x	x	x																	
ADC_EXCLUSIVE_AREA_28																						x	x	x	x																	
ADC_EXCLUSIVE_AREA_29																						x	x	x	x																	
ADC_EXCLUSIVE_AREA_30																						x	x	x	x																	
ADC_EXCLUSIVE_AREA_31																										x																
ADC_EXCLUSIVE_AREA_32																											x															
ADC_EXCLUSIVE_AREA_34																												x	x	x	x	x	x	x								
ADC_EXCLUSIVE_AREA_35																												x	x	x	x	x	x	x								
ADC_EXCLUSIVE_AREA_36																												x	x	x	x	x	x	x								
ADC_EXCLUSIVE_AREA_37																												x	x	x	x	x	x	x								
ADC_EXCLUSIVE_AREA_38																												x	x	x	x	x	x	x								
ADC_EXCLUSIVE_AREA_39																												x	x	x	x	x	x	x								
ADC_EXCLUSIVE_AREA_40																												x	x	x	x	x	x	x								
ADC_EXCLUSIVE_AREA_41																																			x	x	x	x	x			
ADC_EXCLUSIVE_AREA_42																																			x	x	x	x	x			
ADC_EXCLUSIVE_AREA_43																																			x	x	x	x	x			
ADC_EXCLUSIVE_AREA_44																																			x	x	x	x	x			
ADC_EXCLUSIVE_AREA_45																																			x	x	x	x	x			
ADC_EXCLUSIVE_AREA_46																																								x	x	
ADC_EXCLUSIVE_AREA_47																																								x	x	
ADC_EXCLUSIVE_AREA_48																																										x

(Extracted table from RTD_ADC_EXCLUSIVE_AREAS.xlsx)

5.2 Exclusive areas not available on this platform

ADC_EXCLUSIVE_AREA_04 is not available.

ADC_EXCLUSIVE_AREA_05 is not available.

5.3 Peripheral Hardware Requirements

This device provides two General-purpose ADC: ADC HW Unit 0 (ADC0), ADC HW Unit 1 (ADC1). The number of ADC hardware units and channels are derivative specific, so please consult the Reference Manual.

5.4 ISR to configure within AutosarOS - dependencies

Table with Interrupt Service Routines used (S32K11X):

ISR Name	HW INT Vector	Observations
ISR(Adc_0_Isr)	28	Function handles end conversion interrupt of ADC Hardware Unit 0
ISR(Pdb_0_Isr)	19	Function handles sequence error interrupt of PDB Hardware Unit 0

Table with Interrupt Service Routines used (S32K14X):

ISR Name	HW INT Vector	Observations
ISR(Adc_0_Isr)	39	Function handles end conversion interrupt of ADC Hardware Unit 0
ISR(Adc_1_Isr)	40	Function handles end conversion interrupt of ADC Hardware Unit 1
ISR(Pdb_0_Isr)	52	Function handles sequence error interrupt of PDB Hardware Unit 0
ISR(Pdb_1_Isr)	68	Function handles sequence error interrupt of PDB Hardware Unit 1

If DMA transfer mode is used, MCL DMA channel ISR routines should be used for each DMA channel. It depends on the MCL configuration. In this case, Adc_Ipw_AdcXDmaTransferCompleteNotification function should be configured as DMA notification parameter. This is required to update ADC driver internal status.
The following functions should be configured as DMA notification parameter:

Function Name	Observations
Adc_Ipw_Adc0DmaTransferCompleteNotification	DMA notification parameter for transfer complete, ADC unit 0
Adc_Ipw_Adc1DmaTransferCompleteNotification	DMA notification parameter for transfer complete, ADC unit 1

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 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

- Mcu: The Microcontroller Unit Driver (MCU Driver) is primarily responsible for initializing and controlling the chips internal clock sources and clock prescalers. The clock frequency may affect the Trigger frequency, Conversion time and Sampling time.
- Mcl: In DMA mode, the Mcl is used to configure the DMA channel allocated for all ADC HW units. MCL should be initiated before ADC.
- Rm: In DMA mode, Rm is used to configure DMAMUX component that controls DMA channel routing.
- Det: If development error detection for the ADC module is enabled: The ADC module shall raise errors to the Development Error Tracer (DET) whenever a development error is encountered by this module.
- Port: The PORT module shall configure the port pins used by the ADC module. Both analog input pins and external trigger pins have to be considered.
- Base: The Base module contains the common files/definitions needed by all RTD modules.
- Resource: is required to select processor derivative. Current Adc driver has support for the following derivatives, each with a Resource file: s32m242_lqfp64, s32m244_lqfp64.
- RTE: Used to manage exclusive areas used by Adc module.
- EcuC: This module is required for configuring the variant handling in Tresos.
- Os: This module is required for configuring the Partition mapping with core ID in Tresos.

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 Adc_MemMap.h).

5.8 User Mode support

- User Mode configuration in the module
- User Mode configuration in AutosarOS

5.8.1 User Mode configuration in the module

The Adc can be run in user mode if the following steps are performed:

- Enable **AdcEnableUserModeSupport** from the configuration

- Call the following functions as trusted functions:

Function syntax	Description	Available via
void Adc_Ip_SetSupply↩ MonitoringEnable_Trusted↩ Call(const boolean SupplyEnable)	This function enable supply monitoring for the internal channels on SIM registers	Adc_Ip_TrustedFunctions.h
void Adc_Ip_ConfigSupply↩ MonitoringChannel_Trusted↩ Call(const uint32 SupplyChannel)	This function configures the internal channels on on SIM registers	Adc_Ip_TrustedFunctions.h
void Adc_Ip_ResetSupply↩ MonitoringChannel_Trusted↩ Call(void)	This function resets the muxing for ADC channel on SIM register to reset value	Adc_Ip_TrustedFunctions.h
void Adc_Ip_SetTriggerSource↩ Select_TrustedCall(const uint32 Instance, const uint8 TriggerSource)	This function selects trigger source for an ADC instance	Adc_Ip_TrustedFunctions.h
void Adc_Ip_SetPretrigger↩ SourceSelect_TrustedCall(const uint32 Instance, const uint8 PretriggerSource)	This function selects pretrigger source for an ADC instance	Adc_Ip_TrustedFunctions.h
void Adc_Ip_SetSoftware↩ Pretrigger_TrustedCall(const uint32 Instance, const uint8 SoftwarePretrigger)	This function writes the software pretrigger source to be configured for an ADC instance	Adc_Ip_TrustedFunctions.h
void Pdb_Adc_Ip_Config↩ InstanceBackToBack_Trusted↩ Call(const boolean InstanceBack↩ ToBackEnable)	This function enables the instance back to back mode	Pdb_Adc_Ip_Trusted↩ Functions.h
void Pdb_Adc_Ip_ConfigInter↩ ChannelBackToBack_Trusted↩ Call(const uint32 Instance, const boolean InterChannelBackTo↩ BackEnable)	This function enables the inter-channel back to back mode	Pdb_Adc_Ip_Trusted↩ Functions.h

Note

For derivative S32K118 and S32K116, when User Mode Support is enabled, because of the cortex M0+ architecture, global interrupts can not be disabled/enabled using PRIMASK register. Due to this constraint interrupts need to be disabled/enabled one by one.

In order to use features that rely on System Integration Module(SIM) like Internal Supply Monitoring, it is necessary to access SIM registers from supervisor mode. If the application is not set to run from supervisor mode, it is required to enable ADC user mode support, otherwise a bus error will occur.

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>_Ip↩ _TrustedFunctions.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 configuration in the module 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 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 unmarshalling of the parameters to call `<Function_Name>`() of driver. The `<Function_Name>`() functions are already defined in driver and declared in `<IpName>_Ip_TrustedFunctions.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.

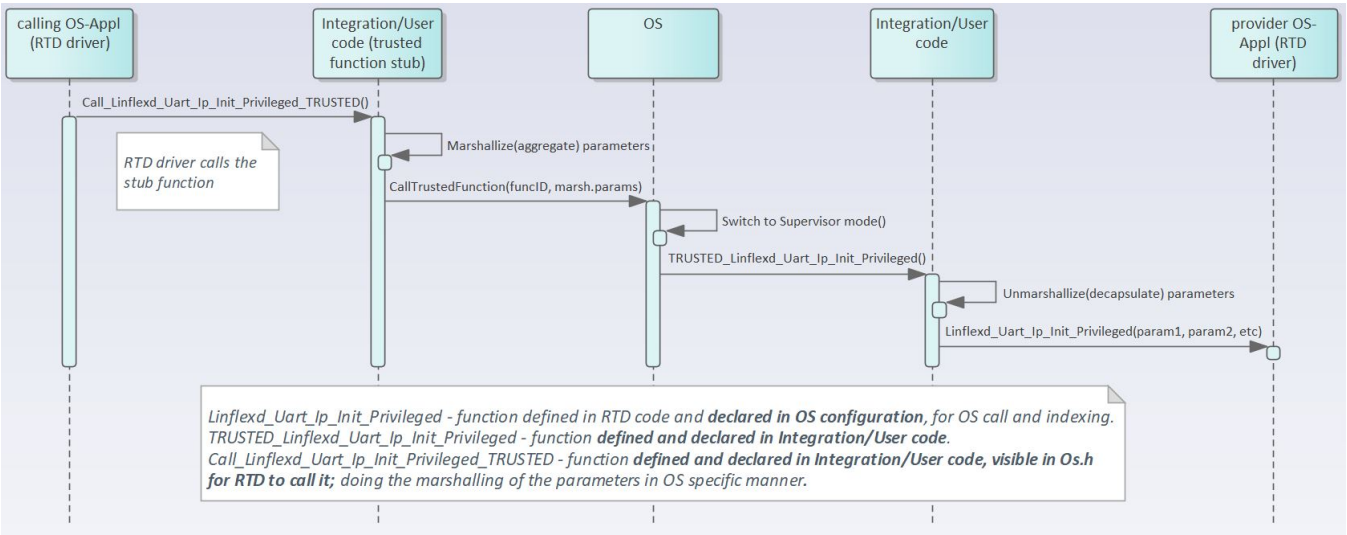

Figure 5.1 Example sequence chart for calling `Linflexd_Uart_Ip_Init_Privileged` as trusted function

5.9 Multicore support

Since the hardware is single core, Multicore support was not implemented.

Chapter 6

Main API Requirements

6.1 Main function calls within BSW scheduler

None.

6.2 API Requirements

None.

6.3 Calls to Notification Functions, Callbacks, Callouts

Call-back Notifications:

None

User Notifications:

The ADC Driver provides a notification callback per group that is called when completing the group conversion. The notifications can be configured as pointers to user defined functions. If notification is not desired, 'NULL_PTR' shall be configured. The function has to be implemented by the user.

The ADC Driver provides also a notification callback that is called whenever PDB channel sequence error occurred. The notifications can be configured as pointers to user defined functions. If PDB Sequence Error notification is not desired, 'NULL_PTR' shall be configured. The function has to be implemented by the user. The function prototype is generated by the ADC configuration.

Chapter 7

Memory allocation

7.1 Sections to be defined in Adc_MemMap.h

Section name	Type of section	Description
ADC_START_SEC_CONFIG_DAT↩ A_UNSPECIFIED	Configuration Data	Start of Memory Section for Config Data
ADC_STOP_SEC_CONFIG_DATA↩ _UNSPECIFIED	Configuration Data	End of Memory Section for Config Data
ADC_START_SEC_CODE	Code	Start of memory Section for Code
ADC_STOP_SEC_CODE	Code	End of memory Section for Code
ADC_START_SEC_CONST_UNSP↩ ECIFIED	Constant Data	The parameters that are not variant aware shall be stored in memory section for constants.
ADC_STOP_SEC_CONST_UNSPE↩ CIFIED	Constant Data	End of above section.
ADC_START_SEC_VAR_CLEAR↩ ED_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.
ADC_STOP_SEC_VAR_CLEARE↩ D_UNSPECIFIED	Variables	End of above section.
ADC_START_SEC_VAR_CLEAR↩ ED_8	Variables	Used for variables which have to be aligned to 8 bits. For instance used for variables of size 8 bits or used for composite data types: arrays, structs containing elements of maximum 8 bits. These variables are cleared to zero by start-up code.
ADC_STOP_SEC_VAR_CLEARE↩ D_8	Variables	End of above section.

Section name	Type of section	Description
ADC_START_SEC_VAR_CLEAR↩ ED_UNSPECIFIED_NO_CACHEA↩ BLE	Non-Cacheable Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 8, 16 or 32 bit, and that have to be stored in a non-cacheable memory section. These variables are cleared to zero by start-up code.
ADC_STOP_SEC_VAR_CLEARE↩ D_UNSPECIFIED_NO_CACHEABLE	Non-Cacheable Variables	End of above section.
ADC_START_SEC_VAR_CLEAR↩ ED_16_NO_CACHEABLE	Non-Cacheable Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 8, 16 or 32 bit, and that have to be stored in a non-cacheable memory section. These variables are cleared to zero by start-up code.
ADC_STOP_SEC_VAR_CLEARE↩ D_16_NO_CACHEABLE	Non-Cacheable Variables	End of above section.
ADC_START_SEC_VAR_CLEAR↩ ED_32_NO_CACHEABLE	Non-Cacheable Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 8, 16 or 32 bit, and that have to be stored in a non-cacheable memory section. These variables are cleared to zero by start-up code.
ADC_STOP_SEC_VAR_CLEARE↩ D_32_NO_CACHEABLE	Non-Cacheable Variables	End of above section.
ADC_START_SEC_CONFIG_DAT↩ A_8	Configuration Data	Used for configs data that have to be aligned to 8 bits.
ADC_STOP_SEC_CONFIG_DATA↩ _8	Configuration Data	End of above section.
ADC_START_SEC_CONFIG_DAT↩ A_16	Configuration Data	Used for configs data that have to be aligned to 16 bits.
ADC_STOP_SEC_CONFIG_DATA↩ _16	Configuration Data	End of above section.
ADC_START_SEC_CONST_32	Constant Data	Used for constants that have to be aligned to 32 bits.
ADC_STOP_SEC_CONST_32	Constant Data	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

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

External Assumption Req ID	External Assumption Text
SWS_Adc_00384	The ADC module's environment shall ensure that a conversion has been completed for the requested group before requesting the conversion result. Note: If no conversion has been completed for the requested channel group (e.g. because the conversion of the ADC Channel group has been stopped by the user) the value returned by the ADC module will be arbitrary (Adc↩ _GetStreamLastPointer will return 0 and read NULL_PTR; Adc_Read↩ Group will return E_NOT_OK). ADC module couldn't handle external environment usage.
SWS_Adc_00414	The ADC module's environment shall check the integrity (see Note SWS_↩ Adc_00413) if several calls for the same ADC group are used during runtime in different tasks or ISR's. Note: The SWS_Adc_00414 is a safety integrity assumption for external environment, which shall be implemented for F↩ TE; For GTE and NTE SWS_Adc_00414 has a role to increase availability because the check will be supported by ADC driver. ADC module couldn't handle external environment usage
SWS_Adc_00415	The ADC module shall not check the integrity (see Note SWS_Adc_00413) if several calls for the same ADC group are used during runtime in different tasks or ISRs. Note: ADC module couldn't handle external environment usage
SWS_Adc_00247	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: ADC registers shall not affect other hardware modules
SWS_Adc_00248	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: ADC registers shall not affect other hardware modules
SWS_Adc_00249	One-time writable registers that require initialization directly after reset shall be initialized by the startup code. Note: ADC registers shouldn't be initialized in startup code
SWS_Adc_00250	All other registers shall be initialized by the startup code. Note: ADC registers shouldn't be initialized in startup code
SWS_Adc_00421	The ADC module's environment shall ensure that no group conversions are started without prior initialization of the according result buffer pointer to point to a valid result buffer. Note: ADC module couldn't handle external environment usage

External Assumption Req ID	External Assumption Text
SWS_Adc_00422	The ADC module's environment shall ensure that the application buffer, which address is passed as parameter in Adc_SetupResultBuffer, has the according size to hold all group channel conversion results and if streaming access is selected, hold these results multiple times as specified with streaming sample parameter (see ADC292). Note: ADC module couldn't handle external environment usage
SWS_Adc_00358	The ADC module's environment shall not call the function Adc_DeInit while any group is not in state ADC_IDLE. Note: ADC module couldn't handle external environment usage
SWS_Adc_00146	The ADC module's environment shall only call Adc_StartGroupConversion for groups configured with software trigger source. Note: ADC module couldn't handle external environment usage
SWS_Adc_00283	The ADC module's environment shall only call the function Adc_Stop↩ GroupConversion for groups configured with trigger source software. Note: ADC module couldn't handle external environment usage
SWS_Adc_00273	The ADC module's environment shall guarantee that no concurrent conversions take place on the same HW Unit (happening of different hardware triggers at the same time). Note: ADC module couldn't handle external environment usage
SWS_Adc_00120	The ADC module's environment shall only call the function Adc_Enable↩ HardwareTrigger for groups configured in hardware trigger mode (see Adc↩ GroupTriggSrc). Note: ADC module couldn't handle external environment usage
SWS_Adc_00121	The ADC module's environment shall only call the function Adc_Disable↩ HardwareTrigger for groups configured in hardware trigger mode (see Adc↩ GroupTriggSrc). Note: ADC module couldn't handle external environment usage
SWS_Adc_00305	To guarantee consistent returned values, it is assumed that ADC group conversion is always started (or enabled in case of HW group) successfully by SW before status polling begins. Note: ADC module couldn't handle external environment usage
SWS_Adc_00219	The ADC module's environment shall guarantee the consistency of the data that has been read by checking the return value of Adc_GetGroupStatus. Note: ADC module couldn't handle external environment usage
EA_RTD_00070	If DMA transfer mode is used, the user must not run SW and HW groups at the same time on the same HW unit.
EA_RTD_00071	If interrupts are locked, a centralized function pair to lock and unlock interrupts shall be used.
EA_RTD_00081	The integrator shall assure that <MSN>_Init() and <MSN>_DeInit() functions do not interrupt each other.
EA_RTD_00082	When caches are enabled and data buffers are allocated in cacheable memory regions the buffers involved in DMA transfer shall be aligned with both start and end to cache line size. Note: **Rationale**: This ensures that no other buffers/variables compete for the same cache lines.
EA_RTD_00083	Before calling the Adc_SetMode() API, the user shall ensure that no conversion is ongoing.
EA_RTD_00084	Before calling the Adc_SetClockMode() API, the user shall ensure that no conversion is ongoing.

External assumptions for driver

External Assumption Req ID	External Assumption Text
EA_RTD_00085	Before calling the Adc_Calibrate() API, the user shall ensure that no conversion is ongoing.
EA_RTD_00089	If DMA transfer is used, data masking (clearing all bit values that do not belong in data bitfield) and data alignment considerations are the responsibility of the user, Adc driver will transfer the data as is.
EA_RTD_00099	Before calling the Adc_SetHwUnitPowerMode() API, the user shall ensure that no conversion is ongoing
EA_RTD_00100	Before calling the Adc_SetPowerState() API, the user shall ensure that no conversion is ongoing.
EA_RTD_00106	Standalone IP configuration and HL configuration of the same driver shall be done in the same project
EA_RTD_00107	The integrator shall use the IP interface only for hardware resources that were configured for standalone IP usage. Note:• The integrator shall not directly use the IP interface for hardware resources that were allocated to be used in HL context.
EA_RTD_00108	The integrator shall use the IP interface to a build a CDD, therefore the BSWMD will not contain reference to the IP interface
EA_RTD_00113	When RTD drivers are integrated with AutosarOS and User mode support is enabled, the integrator shall assure that the definition and declaration of all RTD functions needed to be called as trusted functions follow the naming convention Call<Function_Name>TRUSTED(parameter1,parameter2,...) in Integration/User code. They need to visible in Os.h for the driver to call them. They will call RTD <Function_Name>() as trusted functions in OS specific manner.

How to Reach Us:

Home Page:
nxp.com

Web Support:
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.