Integration Manual

for S32K3 ADC Driver

Document Number: IM34ADCASRR21-11 Rev0000R3.0.0 Rev. 1.0

Chapter 1

Revision History

Revision	Date	Author	Description
1.0	31.03.2023	NXP RTD Team	S32K3 Real-Time Drivers AUTOSAR 4.4 & R21-11 Version 3.0.0

Chapter 2

Introduction

This integration manual describes the integration requirements for ADC Driver for S32K3XX microcontrollers.

2.1 Supported Derivatives

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

- s32k310_mqfp100
- s32k310_lqfp48
- s32k311_mqfp100 / MWCT2015S_mqfp100
- s32k311_lqfp48
- s32k312_mqfp100 / MWCT2016S_mqfp100
- s32k312_mqfp172 / MWCT2016S_mqfp172
- s32k314_mqfp172
- s32k314_mapbga257
- s32k322_mqfp100 / MWCT2D16S_mqfp100
- s32k322_mqfp172 / MWCT2D16S_mqfp172
- s32k324_mqfp172 / MWCT2D17S_mqfp172
- s32k324_mapbga257

- s32k341_mqfp100
- s32k341_mqfp172
- s32k342_mqfp100
- s32k342_mqfp172
- s32k344_mqfp172
- s32k344_mapbga257
- s32k394_mapbga289
- s32k396_mapbga289
- s32k358_mqfp172
- s32k358_mapbga289
- s32k328_mqfp172
- s32k328_mapbga289
- s32k338_mqfp172
- s32k338_mapbga289
- s32k348_mqfp172
- s32k348_mapbga289
- s32m274_lqfp64
- s32m276_lqfp64

All of the above microcontroller devices are collectively named as S32K3.

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
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CS	Chip Select
CTU	Cross Trigger Unit
BCTU	Body 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 Signifigant 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	Specification of ADC Driver	AUTOSAR Release R21-11
2	S32K3XX Reference Manual	Rev. 6, Draft B - 01/2023
3	S32M27x Reference Manual	Rev. 2, Draft A - 02/20223
4	S32K39 and S32K37 Reference Manual	Rev. 2, Draf A - 11/2022
5	S32K3XX Data Sheet	Rev. 6 — 11/2022
6	S32M27x Data Sheet	Rev. 2 RC - 12/2022
7	S32K396 Data Sheet	Rev. 1.1 - 08/2022
8	S32K358_0P14E Mask Set Errata	Rev. 28 - 9/2022
9	S32K396_0P40E Mask Set Errata	Rev. DEC2022 - 12/2022
10	S32K311_0P98C Mask Set Errata	Rev. 6/March/2023 - 3/2023
11	S32K312: Mask Set Errata for Mask 0P09C	Rev. 25/April/2022

#	Title	Version
12	S32K342: Mask Set Errata for Mask 0P97C	Rev. 10 - 11/2022
13	S32K3x4: Mask Set Errata for Mask 0P55A/1P55A	Rev. 14/Oct/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
- DIAB Compiler/Assembler/Linker Options
- GHS Compiler/Assembler/Linker Options
- IAR Compiler/Assembler/Linker Options

The RTD driver files are compiled using:

- NXP GCC 10.2.0 20200723 (Build 1728 Revision g5963bc8)
- Wind River Diab Compiler 7.0.4
- Compiler Versions: Green Hills Multi 7.1.6d / Compiler 2021.1.4
- Compiler Versions: IAR ANSI C/C++ Compiler V8.50.10 (safety version)

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

The TS_T40D34M30I0R0 part of the plugin name is composed as follows:

- T = Target_Id (e.g. T40 identifies Cortex-M architecture)
- D = Derivative_Id (e.g. D34 identifies S32K3 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
-mfpu=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 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.
-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

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
-DS32K3XX	Predefine S32K3XX as a macro, with definition 1
-D $ (DERIVATIVE)	Predefine S32K3's derivative as a macro, with definition 1. For example: Predefine for S32K344 will be -DS32K344.
-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 initalization 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 initalization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver
-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=cortexm7	Targeted ARM processor for which GCC should tune the performance of the code
-mfpu=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
-mthumb	Generates code that executes in Thumb state

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

3.1.2 DIAB Compiler/Assembler/Linker Options

3.1.2.1 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 #if 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
-DS32K3XX	Predefine S32K3XX as a macro, with definition 1
-D $ (DERIVATIVE)	Predefine S32K3's derivative 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 initalization 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 initalization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver
-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 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
-tARMCORTEXM7MG:simple	Selects target processor (hardware single-precision, software double-precision floating-point)

3.1.2.3 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
-tARMCORTEXM7MG:simple	Selects target processor (hardware single-precision, software double-precision floating-point)

3.1.3 GHS Compiler/Assembler/Linker Options

3.1.3.1 GHS Compiler Options

Compiler Option	Description
-cpu=cortexm7	Selects target processor: Arm Cortex M7
-thumb	Selects generating code that executes in Thumb state
-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
-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

Compiler Option	Description
–unsigned_chars	Let the type char be unsigned, like unsigned char
–unsigned_fields	Bitfelds 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 andgenerate 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
-DS32K3XX	Predefine S32K3XX as a macro, with definition 1
-D $ (DERIVATIVE)	Predefine S32K3's derivative as a macro, with definition 1. For example: Predefine for S32K344 will be -DS32K344.
-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 initalization 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 initalization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver
-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=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

3.1.3.3 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. DWA↩ RF 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
-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

3.1.4 IAR Compiler/Assembler/Linker Options

3.1.4.1 IAR Compiler Options

Compiler Option	Description
–cpu Cortex-M7	Targeted ARM processor for which IAR should tune the performance of the code
–cpu_mode thumb	Generates code that executes in Thumb state
–endian little	Generate code for a processor running in little-endian mode
–fpu VFPv5-SP	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant.
-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

Compiler Option	Description
–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
–do_explicit_zero_opt_in_named_sections	Disable the exception for variables in user-named sections, and thus treat explicit initializations to zero as zero initializations, not copy 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
-DS32K3XX	Predefine S32K3XX as a macro, with definition 1
-D $ (DERIVATIVE)	Predefine S32K3's derivative as a macro, with definition 1. For example: Predefine for S32K344 will be -DS32K344.
-DIAR	Predefine IAR 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 initalization 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 initalization in source file system.c under the Platform driver
-DENABLE_FPU	Predefine ENABLE_FPU as a macro, with definition 1. Enables FPU initalization in source file system.c under the Platform driver
-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.

3.1.4.2 IAR Assembler Options

Assembler Option	Description
–cpu Cortex-M7	Targeted ARM processor for which IAR should generate the instruction set
–fpu VFPv5-SP	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant.
–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

3.1.4.3 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-M7	Selects the ARM processor variant to link the application for
–fpu VFPv5-SP	Use this option to generate code that performs floating-point operations using a Floating Point Unit (FPU). Single-precision variant.
–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

3.2 Files required for compilation

This section describes the include files required to compile, assemble (if assembler code) and link the ADC driver for S32K3XX 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_T40D34M30I0R0\include\Adc.h
- ..\Adc_TS_T40D34M30I0R0\include\Adc_Ipw.h
- ..\Adc_TS_T40D34M30I0R0\include\Adc_Ipw_Irq.h
- ..\Adc_TS_T40D34M30I0R0\include\Adc_Ipw_Types.h
- ..\Adc_TS_T40D34M30I0R0\include\Adc_Sar_Ip.h
- ..\Adc_TS_T40D34M30I0R0\include\Adc_Sar_Ip_HeaderWrapper_S32K3.h
- ..\Adc_TS_T40D34M30I0R0\include\Adc_Sar_Ip_HwAccess.h
- ..\Adc_TS_T40D34M30I0R0\include\Adc_Sar_Ip_Irq.h
- ..\Adc_TS_T40D34M30I0R0\include\Adc_Sar_Ip_TrustedFunctions.h
- ..\Adc_TS_T40D34M30I0R0\include\Adc_Sar_Ip_Types.h
- ..\Adc_TS_T40D34M30I0R0\include\Adc_Types.h
- ..\Adc_TS_T40D34M30I0R0\include\Bctu_Ip.h
- ..\Adc_TS_T40D34M30I0R0\include\Bctu_Ip_Irq.h
- ..\Adc_TS_T40D34M30I0R0\include\Bctu_Ip_Types.h
- ..\Adc_TS_T40D34M30I0R0\src\Adc.c

- ..\Adc_TS_T40D34M30I0R0\src\Adc_Ipw.c
- ..\Adc_TS_T40D34M30I0R0\src\Adc_Ipw_Irq.c
- ..\Adc_TS_T40D34M30I0R0\src\Adc_Sar_Ip.c
- ..\Adc_TS_T40D34M30I0R0\src\Adc_Sar_Ip_Irq.c
- ..\Adc_TS_T40D34M30I0R0\src\Bctu_Ip.c
- ..\Adc_TS_T40D34M30I0R0\src\Bctu_Ip_Irq.c

In case K396 is used:

- ..\Adc_TS_T40D34M30I0R0\include\DSPSS_Api.h
- ..\Adc_TS_T40D34M30I0R0\include\DSPSS_CFSDADC_MemoryMap.h
- ..\Adc_TS_T40D34M30I0R0\include\DSPSS_CFSDADC_PMEM.h
- ..\Adc_TS_T40D34M30I0R0\include\DSPSS_CFSDADC_XMEM.h
- ..\Adc_TS_T40D34M30I0R0\include\DSPSS_Types.h
- ..\Adc_TS_T40D34M30I0R0\include\Sdadc_Ip_Irq.h
- ..\Adc_TS_T40D34M30I0R0\include\Sdadc_Ip_Types.h
- ..\Adc_TS_T40D34M30I0R0\include\Sdadc_Ip.h
- ..\Adc_TS_T40D34M30I0R0\src\DSPSS_Api.c
- ..\Adc_TS_T40D34M30I0R0\src\Sdadc_Ip_Irq.c
- ..\Adc_TS_T40D34M30I0R0\src\Sdadc_Ip.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_Sar_Ip_Cfg.h
- Adc_Sar_Ip_CfgDefines.h

- Adc_Sar_Ip_PBcfg.c
- Adc_Sar_Ip_PBcfg.h
- Bctu_Ip_Cfg.h
- Bctu_Ip_CfgDefines.h
- Bctu_Ip_PBcfg.c
- Bctu_Ip_PBcfg.h

In case K396 is used:

- Sdadc_Ip_Cfg.h
- Sdadc_Ip_CfgDefines.h
- Sdadc_Ip_PBcfg.c
- Sdadc_Ip_PBcfg.h

Files from Base common folder

- ..\BaseNXP_TS_T40D34M30I0R0\include\Adc_MemMap.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\Compiler.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\Compiler_Cfg.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\ComStack_Cfg.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\ComStackTypes.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\Mcal.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\OsIf.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\OsIf_Cfg_TypesDef.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\OsIf_DeviceRegisters.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\OsIf_Timer_System_Internal_Systick.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\RegLockMacros.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\SilRegMacros.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\Soc_Ips.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\Platform_Types.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\StandardTypes.h
- ..\BaseNXP_TS_T40D34M30I0R0\include\Reg_eSys.h
- ..\BaseNXP_TS_T40D34M30I0R0\generate_PC\include\Mcal.h
- ..\BaseNXP_TS_T40D34M30I0R0\generate_PC\include\OsIf_Cfg.h
- ..\BaseNXP_TS_T40D34M30I0R0\header

Files from Det folder

- ..\Det_TS_T40D34M30I0R0\include\Det.h
- ..\Det_TS_T40D34M30I0R0\src\Det.c

Files from Mcl folder

- ..\Mcl_TS_T40D34M30I0R0\include\CDD_Mcl.h
- ..\Mcl_TS_T40D34M30I0R0\include\Mcl.h
- ..\Mcl_TS_T40D34M30I0R0\include\Dma_Ip.h
- ..\Mcl_TS_T40D34M30I0R0\include\Dma_Ip_Types.h
- ..\Mcl_TS_T40D34M30I0R0\include\Mcl_EnvCfg.h
- ..\Mcl_TS_T40D34M30I0R0\include\Mcl_Types.h
- ..\Mcl_TS_T40D34M30I0R0\src\CDD_Mcl.c
- ..\Mcl_TS_T40D34M30I0R0\src\Dma_Ip.c
- ..\Mcl_TS_T40D34M30I0R0\src\Dma_Ip_Irq.c
- ..\Mcl_TS_T40D34M30I0R0\generate_PC\src\CDD_Mcl_Cfg.c
- ..\Mcl_TS_T40D34M30I0R0\generate_PB\src\CDD_Mcl_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 b510999 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_T40D34M30I0R0\config\Adc.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k310_lqfp48.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k310_mqfp100.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k311_mqfp100.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k311_lqfp48.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k312_mqfp100.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k312_mqfp172.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k314_mqfp172.epd

  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k314_mapbga257.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k322_mqfp100.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k322_mqfp172.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k324_mqfp172.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k324_mapbga257.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k341_mqfp100.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k341_mqfp172.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k342_mqfp100.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k342_mqfp172.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k344_mqfp172.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k344_mapbga257.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k328_mqfp172.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k328_mapbga289.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k338_mqfp172.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k338_mapbga289.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k348_mqfp172.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k348_mapbga289.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k358_172mqfp.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k358_289bga.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32m276_lqfp64.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32m274_lqfp64.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k394_mapbga289.epd
  - ..\Adc_TS_T40D34M30I0R0\autosar\Adc_s32k396_mapbga289.epd.epd

- Code Generation Templates for parameters without variation points:
  - ..\Adc_TS_T40D34M30I0R0\output\include\Adc_Cfg.h
  - ..\Adc_TS_T40D34M30I0R0\output\include\Adc_CfgDefines.h
  - ..\Adc_TS_T40D34M30I0R0\output\include\Adc_Ipw_Cfg.h
  - ..\Adc_TS_T40D34M30I0R0\output\include\Adc_Ipw_CfgDefines.h
  - ..\Adc_TS_T40D34M30I0R0\output\include\Adc_Sar_Ip_Cfg.h
  - ..\Adc_TS_T40D34M30I0R0\output\include\Adc_Sar_Ip_CfgDefines.h
  - ..\Adc_TS_T40D34M30I0R0\output\include\Bctu_Ip_Cfg.h
  - ..\Adc_TS_T40D34M30I0R0\output\include\Bctu_Ip_CfgDefines.h
  - ..\Adc_TS_T40D34M30I0R0\output\src\Adc_Cfg.c
    In case K396 is used:
  - ..\Adc_TS_T40D34M30I0R0\output\include\Sdadc_Ip_Cfg.h
  - ..\Adc_TS_T40D34M30I0R0\output\include\Sdadc_Ip_CfgDefines.h

- Code Generation Templates for variant aware parameters:
  - ..\Adc_TS_T40D34M30I0R0\output\include\Adc_<VariantName>_PBcfg.h
  - ..\Adc_TS_T40D34M30I0R0\output\src\Adc_<VariantName>_PBcfg.c
  - ..\Adc_TS_T40D34M30I0R0\output\include\Adc_Ipw_<VariantName>_PBcfg.h

- ..\Adc_TS_T40D34M30I0R0\output\src\Adc_Ipw_<VariantName>_PBcfg.c
- ..\Adc_TS_T40D34M30I0R0\output\include\Adc_Sar_Ip_<VariantName>_PBcfg.h
- ..\Adc_TS_T40D34M30I0R0\output\src\Adc_Sar_Ip_<VariantName>_PBcfg.c
- ..\Adc_TS_T40D34M30I0R0\output\include\Bctu_Ip_<VariantName>_PBcfg.h
- ..\Adc_TS_T40D34M30I0R0\output\src\Bctu_Ip_<VariantName>_PBcfg.c
  In case K396 is used:
- ..\Adc_TS_T40D34M30I0R0\output\include\Sdadc_Ip_<VariantName>_PBcfg.h
- ..\Adc_TS_T40D34M30I0R0\output\src\Sdadc_Ip_<VariantName>_PBcfg.c

Steps to generate the configuration:

1. Copy the module folders Adc_TS_T40D34M30I0R0, BaseNXP_TS_T40D34M30I0R0, Det_TS_T40D34M30I0R0, Resource_TS_T40D34M30I0R0, Os_TS_T40D34M30I0R0, EcuM_TS_T40D34M30I0R0, Mcl_TS_T40D34M30I0R0, Rm_TS_T40D34M30I0R0, Mcu_TS_T40D34M30I0R0, Rte_TS_T40D34M30I0R0, Ecuc_TS_T40D34M30I0R0 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

- 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, 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_04 is used in function Adc_EnableCTUTrigger to protect the updates for Adc↩ _au8CtuGroupTriggersActive variable

ADC_EXCLUSIVE_AREA_05 is used in function Adc_DisableCTUTrigger to protect the updates for Adc↩ _au8CtuGroupTriggersActive variable

ADC_EXCLUSIVE_AREA_10 is used in function Adc_StartGroupConversion to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_StartConversion

ADC_EXCLUSIVE_AREA_10 is used in function Adc_StopGroupConversion to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_StartConversion

ADC_EXCLUSIVE_AREA_10 is used in function Adc_ReadGroup to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_StartConversion

ADC_EXCLUSIVE_AREA_11 is used in function Adc_SelfTest to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SelfTest

ADC_EXCLUSIVE_AREA_12 is used in function Adc_Calibrate to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_DoCalibration

ADC_EXCLUSIVE_AREA_13 is used in function Adc_SetHwUnitPowerMode to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerup

ADC_EXCLUSIVE_AREA_13 is used in function Adc_SetPowerState to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerup

ADC_EXCLUSIVE_AREA_13 is used in function Adc_Init to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerup

ADC_EXCLUSIVE_AREA_13 is used in function Adc_DeInit to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerup

ADC_EXCLUSIVE_AREA_13 is used in function Adc_SelfTest to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerup

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

ADC_EXCLUSIVE_AREA_13 is used in function Adc_SetClockMode to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerup

ADC_EXCLUSIVE_AREA_13 is used in function Adc_Sar_Ip_SetCtuMode to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerup

ADC_EXCLUSIVE_AREA_14 is used in function Adc_SetHwUnitPowerMode to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerdown

ADC_EXCLUSIVE_AREA_14 is used in function Adc_SetPowerState to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerdown

ADC_EXCLUSIVE_AREA_14 is used in function Adc_Init to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerdown

ADC_EXCLUSIVE_AREA_14 is used in function Adc_DeInit to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerdown

ADC_EXCLUSIVE_AREA_14 is used in function Adc_SelfTest to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerdown

ADC_EXCLUSIVE_AREA_14 is used in function Adc_Calibrate to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerdown

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

ADC_EXCLUSIVE_AREA_14 is used in function Adc_Sar_Ip_SetCtuMode to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_Powerdown

ADC_EXCLUSIVE_AREA_15 is used in function Adc_SetClockMode to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetClockMode

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

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

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

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

ADC_EXCLUSIVE_AREA_16 is used in function Adc_EnableCTUTrigger to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetAveraging

ADC_EXCLUSIVE_AREA_16 is used in function Adc_SetClockMode to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetAveraging

ADC_EXCLUSIVE_AREA_18 is used in function Adc_StartGroupConversion to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_AbortChain

ADC_EXCLUSIVE_AREA_18 is used in function Adc_StopGroupConversion to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_AbortChain

ADC_EXCLUSIVE_AREA_19 is used in function Adc_StartGroupConversion to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetConversionMode

ADC_EXCLUSIVE_AREA_19 is used in function Adc_StopGroupConversion to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetConversionMode

ADC_EXCLUSIVE_AREA_19 is used in function Adc_ReadGroup to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetConversionMode

ADC_EXCLUSIVE_AREA_19 is used in function Adc_EnableHardwareTrigger to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetConversionMode

ADC_EXCLUSIVE_AREA_20 is used in function Adc_EnableHardwareTrigger to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetCtuMode

ADC_EXCLUSIVE_AREA_20 is used in function Adc_EnableCtuControlMode to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetCtuMode

ADC_EXCLUSIVE_AREA_20 is used in function Adc_DisableCtuControlMode to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetCtuMode

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

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

ADC_EXCLUSIVE_AREA_22 is used in function Adc_TempSenseGetTemp to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_TempSenseGetTemp

ADC_EXCLUSIVE_AREA_25 is used in function Adc_SelfTest to protect the ADC_NCMRx register from read/modify/write operation in Adc_Sar_Ip_SelfTest

ADC_EXCLUSIVE_AREA_28 is used in function Adc_EnableWdgNotification to protect the ADC_CWE↩NRx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelNotifications

ADC_EXCLUSIVE_AREA_29 is used in function Adc_DisableWdgNotification to protect the ADC_CW↩ENRx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelNotifications

ADC_EXCLUSIVE_AREA_30 is used in function Adc_EnableWdgNotification to protect the ADC_CIMRx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelNotifications

ADC_EXCLUSIVE_AREA_31 is used in function Adc_DisableWdgNotification to protect the ADC_CIMRx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelNotifications

ADC_EXCLUSIVE_AREA_32 is used in function Adc_Init to protect the ADC_WTIMRx register from read/modify/write operation in Adc_Sar_Ip_SetWdgThreshold

ADC_EXCLUSIVE_AREA_32 is used in function Adc_DeInit to protect the ADC_WTIMRx register from read/modify/write operation in Adc_Sar_Ip_SetWdgThreshold

ADC_EXCLUSIVE_AREA_32 is used in function Adc_ConfigureThreshold to protect the ADC_WTIMRx register from read/modify/write operation in Adc_Sar_Ip_SetWdgThreshold

ADC_EXCLUSIVE_AREA_33 is used in function Adc_StartGroupConversion to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_33 is used in function Adc_StopGroupConversion to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_33 is used in function Adc_ReadGroup to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_33 is used in function Adc_EnableHardwareTrigger to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

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

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

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

ADC_EXCLUSIVE_AREA_34 is used in function Adc_DisableHardwareTrigger to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_34 is used in function Adc_EnableCTUTrigger to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_34 is used in function Adc_DisableCTUTrigger to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_35 is used in function Adc_SelfTest to protect the ADC_STCRx register from read/modify/write operation in Adc_Sar_Ip_SelfTest

ADC_EXCLUSIVE_AREA_36 is used in function Adc_Calibrate to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_DoCalibration

ADC_EXCLUSIVE_AREA_37 is used in function Adc_Init to protect the ADC_CALBISTREG register from read/modify/write operation in Adc_Sar_Ip_SetResolution

ADC_EXCLUSIVE_AREA_37 is used in function Adc_DeInit to protect the ADC_CALBISTREG register from read/modify/write operation in Adc_Sar_Ip_SetResolution

ADC_EXCLUSIVE_AREA_38 is used in function Adc_Init to protect the ADC_PSCR register from read/modify/write operation in Adc_Sar_Ip_SetPresamplingSource

ADC_EXCLUSIVE_AREA_38 is used in function Adc_DeInit to protect the ADC_PSCR register from read/modify/write operation in Adc_Sar_Ip_SetPresamplingSource

ADC_EXCLUSIVE_AREA_43 is used in function Adc_StartGroupConversion to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_43 is used in function Adc_StopGroupConversion to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_43 is used in function Adc_ReadGroup to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_43 is used in function Adc_EnableHardwareTrigger to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_44 is used in function Adc_StartGroupConversion to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_DisableDma

ADC_EXCLUSIVE_AREA_44 is used in function Adc_StopGroupConversion to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_DisableDma

ADC_EXCLUSIVE_AREA_44 is used in function Adc_EnableHardwareTrigger to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_DisableDma

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

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

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

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

ADC_EXCLUSIVE_AREA_46 is used in function Adc_StartGroupConversion to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

ADC_EXCLUSIVE_AREA_46 is used in function Adc_StopGroupConversion to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

ADC_EXCLUSIVE_AREA_46 is used in function Adc_ReadGroup to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

ADC_EXCLUSIVE_AREA_46 is used in function Adc_EnableHardwareTrigger to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

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

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

ADC_EXCLUSIVE_AREA_47 is used in function Adc_DisableHardwareTrigger to protect the ADC_DM↩ ARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

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

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

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

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

ADC_EXCLUSIVE_AREA_49 is used in function Adc_TempSenseSetPowerMode to protect the ETSCTL register from read/modify/write operation in Adc_Sar_Ip_TempSenseEnable

ADC_EXCLUSIVE_AREA_50 is used in function Adc_TempSenseSetPowerMode to protect the ETSCTL register from read/modify/write operation in Adc_Sar_Ip_TempSenseDisable

ADC_EXCLUSIVE_AREA_51 is used in function Adc_SetClockMode to protect the ADC_AMSIO register from read/modify/write operation in Adc_Sar_Ip_SetClockMode

ADC_EXCLUSIVE_AREA_54 is used in function Adc_EnableHardwareTrigger to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_54 is used in function Adc_EnableCTUTrigger to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_55 is used in function Adc_CtuSetPowerMode to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetLowPowerMode

ADC_EXCLUSIVE_AREA_55 is used in function Adc_SetPowerState to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetLowPowerMode

ADC_EXCLUSIVE_AREA_56 is used in function Adc_EnableHardwareTrigger to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_56 is used in function Adc_EnableCTUTrigger to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_56 is used in function Adc_CtuEnableNotification to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_57 is used in function Adc_EnableHardwareTrigger to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_57 is used in function Adc_EnableCTUTrigger to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_57 is used in function Adc_CtuDisableNotification to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_58 is used in function Adc_CtuEnableHwTrigger to protect the BCTU_TRG↩ CFGx register from read/modify/write operation in Bctu_Ip_EnableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_DisableHardwareTrigger to protect the BCTU_T↩ RGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_DisableCTUTrigger to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_CtuDisableHwTrigger to protect the BCTU_TRG↩ CFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_60 is used in function Adc_CtuStopLoopConversions to protect the BCTU_T↩ RGCFGx register from read/modify/write operation in Bctu_Ip_StopLoopConversions

ADC_EXCLUSIVE_AREA_61 is used in function Adc_EnableHardwareTrigger to protect the BCTU_TR↩ GCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_EnableCTUTrigger to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Init to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_EnableCtuControlMode to protect the BCTU_TR↩GCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_DisableCtuControlMode to protect the BCTU_T↩RGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_CtuSetListPointer to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_SetTriggerChnListAddr

ADC_EXCLUSIVE_AREA_62 is used in function Adc_DeInit to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_PrepareReconfigBlocking

ADC_EXCLUSIVE_AREA_62 is used in function Adc_EnableCtuControlMode to protect the BCTU_TR↩GCFGx register from read/modify/write operation in Bctu_PrepareReconfigBlocking

ADC_EXCLUSIVE_AREA_62 is used in function Adc_DisableCtuControlMode to protect the BCTU_T↩RGCFGx register from read/modify/write operation in Bctu_PrepareReconfigBlocking

ADC_EXCLUSIVE_AREA_63 is used in function Adc_EnableHardwareTrigger to protect the BCTU_TR↩GCFGx register from read/modify/write operation in Bctu_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_63 is used in function Adc_EnableCTUTrigger to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_63 is used in function Adc_CtuEnableNotification to protect the BCTU_TR↩GCFGx register from read/modify/write operation in Bctu_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_64 is used in function Adc_EnableHardwareTrigger to protect the BCTU_TR↩GCFGx register from read/modify/write operation in Bctu_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_64 is used in function Adc_EnableCTUTrigger to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_64 is used in function Adc_CtuDisableNotification to protect the BCTU_TR↩GCFGx register from read/modify/write operation in Bctu_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_65 is used in function Adc_DeInit to protect the BCTU_FIFOCR register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_65 is used in function Adc_EnableCtuControlMode to protect the BCTU_FI↩FOCR register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_65 is used in function Adc_DisableCtuControlMode to protect the BCTU_FI↩FOCR register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_65 is used in function Adc_Init to protect the BCTU_FIFOCR register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_66 is used in function Adc_DeInit to protect the BCTU_FIFOWM register from read/modify/write operation in Bctu_Ip_SetFifoWatermark

ADC_EXCLUSIVE_AREA_66 is used in function Adc_EnableCtuControlMode to protect the BCTU_FI↩ FOWM register from read/modify/write operation in Bctu_Ip_SetFifoWatermark

ADC_EXCLUSIVE_AREA_66 is used in function Adc_DisableCtuControlMode to protect the BCTU_FI↩ FOWM register from read/modify/write operation in Bctu_Ip_SetFifoWatermark

ADC_EXCLUSIVE_AREA_66 is used in function Adc_Init to protect the BCTU_FIFOWM register from read/modify/write operation in Bctu_Ip_SetFifoWatermark

ADC_EXCLUSIVE_AREA_66 is used in function Adc_CtuSetFifoWatermark to protect the BCTU_FIF↩ OWM register from read/modify/write operation in Bctu_Ip_SetFifoWatermark

ADC_EXCLUSIVE_AREA_67 is used in function Adc_DeInit to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_67 is used in function Adc_EnableCtuControlMode to protect the BCTU_LI↩ STCHRx register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_67 is used in function Adc_DisableCtuControlMode to protect the BCTU_LI↩ STCHRx register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Init to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_67 is used in function Adc_CtuSetList to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanList

ADC_EXCLUSIVE_AREA_67 is used in function Adc_EnableHardwareTrigger to protect the BCTU_LIS↩ TCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_67 is used in function Adc_EnableCTUTrigger to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_DeInit to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_68 is used in function Adc_EnableCtuControlMode to protect the BCTU_LI↩ STCHRx register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_68 is used in function Adc_DisableCtuControlMode to protect the BCTU_LI↩ STCHRx register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Init to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_68 is used in function Adc_CtuSetList to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanList

ADC_EXCLUSIVE_AREA_68 is used in function Adc_EnableHardwareTrigger to protect the BCTU_LIS↩ TCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_EnableCTUTrigger to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_70 is used in function Bctu_Ip_Deinit to protect the BCTU_MSR register from read/modify/write operation in Bctu_Ip_ClearStatusFlags

ADC_EXCLUSIVE_AREA_70 is used in function Adc_DeInit to protect the BCTU_MSR register from read/modify/write operation in Bctu_Ip_ClearStatusFlags

ADC_EXCLUSIVE_AREA_70 is used in function Adc_EnableCtuControlMode to protect the BCTU_MSR register from read/modify/write operation in Bctu_Ip_ClearStatusFlags

ADC_EXCLUSIVE_AREA_70 is used in function Adc_DisableCtuControlMode to protect the BCTU_MSR register from read/modify/write operation in Bctu_Ip_ClearStatusFlags

ADC_EXCLUSIVE_AREA_70 is used in function Adc_DisableHardwareTrigger to protect the BCTU_MSR register from read/modify/write operation in Bctu_Ip_ClearStatusFlags

ADC_EXCLUSIVE_AREA_70 is used in function Adc_DisableCTUTrigger to protect the BCTU_MSR register from read/modify/write operation in Bctu_Ip_ClearStatusFlags

ADC_EXCLUSIVE_AREA_71 is used in function Adc_CtuStartConversion to protect the BCTU_MSR register from read/modify/write operation in Bctu_Ip_SwTriggerConversion

ADC_EXCLUSIVE_AREA_72 is used in function Adc_DisableHardwareTrigger to protect the SDADC_↩ MCR register read/modify/write operation in Sdadc_Ip_Powerup

ADC_EXCLUSIVE_AREA_73 is used in function Adc_DisableHardwareTrigger to protect the SDADC_↩ MCR register read/modify/write operation in Sdadc_Ip_Powerdown

ADC_EXCLUSIVE_AREA_74 is used in function Adc_EnableWdgNotification to protect the SDADC_MCR register read/modify/write operation in Sdadc_Ip_EnableWatchdog

ADC_EXCLUSIVE_AREA_75 is used in function Adc_DisableWdgNotification to protect the SDADC_MCR register read/modify/write operation in Sdadc_Ip_DisableWatchdog

ADC_EXCLUSIVE_AREA_78 is used in function Adc_EnableHardwareTrigger to protect the SDADC_MCR registers read/modify/write operation in Sdadc_Ip_SetInputChannel

ADC_EXCLUSIVE_AREA_79 is used in function Adc_EnableHardwareTrigger to protect the SDADC_FCR register read/modify/write operation in Sdadc_Ip_FlushFifo

ADC_EXCLUSIVE_AREA_80 is used in function Adc_EnableHardwareTrigger to protect the SDADC_R↩ SER register read/modify/write operation in Sdadc_Ip_EnableDmaEvents

ADC_EXCLUSIVE_AREA_81 is used in function Adc_ConfigureThreshold to protect the SDADC_RSER register read/modify/write operation in Sdadc_Ip_EnableInterruptEvents

ADC_EXCLUSIVE_AREA_82 is used in function Adc_DisableHardwareTrigger to protect the SDADC_R↩ SER register read/modify/write operation in Sdadc_Ip_DisableInterruptEvents

ADC_EXCLUSIVE_AREA_83 is used in function Adc_EnableHardwareTrigger to protect the SDADC_MCR register read/modify/write operation in Sdadc_Ip_EnableHwTrigger

ADC_EXCLUSIVE_AREA_84 is used in function Adc_DisableHardwareTrigger to protect the SDADC_↩ MCR register read/modify/write operation in Sdadc_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_85 is used in function Adc_EnableHardwareTrigger to protect the SDADC_MCR register read/modify/write operation in Sdadc_Ip_SetHwTrigger

ADC_EXCLUSIVE_AREA_86 is used in function Adc_Init to protect the DSPSS_DmaReadBufferWrap↩ Status read/modify/write operation in Sdadc_InitDspssThread

ADC_EXCLUSIVE_AREA_86 is used in function Adc_EnableHardwareTrigger to protect the DSPSS_↩ DmaReadBufferWrapStatus read/modify/write operation in Sdadc_Ip_FlushFifo

ADC_EXCLUSIVE_AREA_87 is used in function Adc_EnableHardwareTrigger to protect the DSPSS_SC↩ HEDULER_XMEM_THx register modify/write operation in Sdadc_Ip_ReloadConversion

ADC_EXCLUSIVE_AREA_88 is used in function Adc_Init to protect the DSP_CORE_BUFFER_CFG register modify/write operation in Sdadc_InitDspssThread

ADC_EXCLUSIVE_AREA_89 is used in function Adc_Init to protect the DSP_CORE_BUFFER_CFG register modify/write operation in Sdadc_InitDspssThread

ADC_EXCLUSIVE_AREA_90 is used in function Adc_EnableHardwareTrigger to protect the DSP_COR↩ E_BUFFER_CFG register modify/write operation in Sdadc_Ip_FlushFifo

ADC_EXCLUSIVE_AREA_91 is used in function Adc_Init to protect the DSPSS_INTERRUPT_ENAB↩ LE_REGISTER register modify/write operation in Sdadc_InitDspssThread

ADC_EXCLUSIVE_AREA_92 is used in function Adc_Init to protect the DSPSS_INTERRUPT_ENAB↩ LE_REGISTER2 register modify/write operation in Sdadc_InitDspssThread

ADC_EXCLUSIVE_AREA_93 is not available

ADC_EXCLUSIVE_AREA_94 is not available

ADC_EXCLUSIVE_AREA_95 is used in function Adc_Init to protect the DSPSS_DMA_BUFFER_CFG register modify/write operation in Sdadc_InitDspssThread

ADC_EXCLUSIVE_AREA_95 is used in function Adc_EnableHardwareTrigger to protect the DSPSS_D↩ MA_BUFFER_CFG register modify/write operation in Sdadc_Ip_FlushFifo

ADC_EXCLUSIVE_AREA_96 is used in function Adc_Calibrate to protect the SDADC_MCR register modify/write operation in Sdadc_GainCalibration

ADC_EXCLUSIVE_AREA_97 is used in function Adc_Calibrate to protect the SDADC_MCR register modify/write operation in Sdadc_GainCalibration

ADC_EXCLUSIVE_AREA_98 is used in function Adc_Calibrate to protect the SDADC_MCR register modify/write operation in Sdadc_OffsetCalibration

ADC_EXCLUSIVE_AREA_99 is used in function Adc_Calibrate to protect the SDADC_MCR register modify/write operation in Sdadc_OffsetCalibration

ADC_EXCLUSIVE_AREA_100 is not available

ADC_EXCLUSIVE_AREA_101 is used in function Adc_EnableHardwareTrigger to protect the SDADC_↩ CSR register modify/write operation in Sdadc_Ip_SetInputChannel

ADC_EXCLUSIVE_AREA_102 is used in function Adc_DisableHardwareTrigger to protect the SDADC_↩ RSER register modify/write operation in Sdadc_Ip_Powerdown

ADC_EXCLUSIVE_AREA_103 is used in function Adc_EnableHardwareTrigger to protect the IP_DSPS↩ S_0_DMA_BUFFER_CFG register modify/write operation in Sdadc_Ip_FlushFifo

ADC_EXCLUSIVE_AREA_103 is used in function Adc_UpdateStatusStartConversion to protect the IP_↩ DSPSS_0_DMA_BUFFER_CFG register modify/write operation in Sdadc_Ip_FlushFifo

ADC_EXCLUSIVE_AREA_103 is used in function Adc_UpdateStatusStopConversion to protect the IP_↩ DSPSS_0_DMA_BUFFER_CFG register modify/write operation in Sdadc_Ip_FlushFifo

ADC_EXCLUSIVE_AREA_103 is used in function Adc_UpdateSwQueueIndexNoInt to protect the IP_D↩ SPSS_0_DMA_BUFFER_CFG register modify/write operation in Sdadc_Ip_FlushFifo

Exclusive Areas are used in Interrupt service request (ISR)

ADC_EXCLUSIVE_AREA_00 is used in function Adc_Ipw_Adc0EndNormalChainNotification 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_Adc1EndNormalChainNotification 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_00 is used in function Adc_Ipw_Adc2EndNormalChainNotificationto protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_00 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotificationto protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_00 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_00 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the updates for Software Queue index part of Adc_axUnitStatus variable

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_StartConversion

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_StartConversion

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_StartConversion

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_StartConversion

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_StartConversion

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_StartConversion

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_StartConversion

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_StartConversion

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_StartConversion

ADC_EXCLUSIVE_AREA_18 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_AbortChain

ADC_EXCLUSIVE_AREA_18 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_AbortChain

ADC_EXCLUSIVE_AREA_18 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_AbortChain

ADC_EXCLUSIVE_AREA_18 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_AbortChain

ADC_EXCLUSIVE_AREA_18 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_AbortChain

ADC_EXCLUSIVE_AREA_18 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_AbortChain

ADC_EXCLUSIVE_AREA_18 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_AbortChain

ADC_EXCLUSIVE_AREA_18 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_AbortChain

ADC_EXCLUSIVE_AREA_18 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_AbortChain

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetExternalTrigger

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

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetExternalTrigger

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

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetExternalTrigger

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetExternalTrigger

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetExternalTrigger

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetExternalTrigger

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the ADC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetExternalTrigger

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_BctuEndListNotificationAdc0 to protect the A↩ DC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetExternalTrigger

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_BctuEndListNotificationAdc1 to protect the A↩ DC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetExternalTrigger

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Ipw_BctuEndListNotificationAdc2 to protect the A↩ DC_MCR register from read/modify/write operation in Adc_Sar_Ip_SetExternalTrigger

ADC_EXCLUSIVE_AREA_33 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_33 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_33 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_33 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_33 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_33 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_33 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_33 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_33 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_EnableNotifications

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

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

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

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

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the ADC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_BctuEndListNotificationAdc0 to protect the A↩DC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_BctuEndListNotificationAdc1 to protect the A↩DC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Ipw_BctuEndListNotificationAdc2 to protect the A↩DC_IMR register from read/modify/write operation in Adc_Sar_Ip_DisableNotifications

ADC_EXCLUSIVE_AREA_43 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_43 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_43 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_43 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_43 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_43 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_43 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_43 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_43 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_EnableDma

ADC_EXCLUSIVE_AREA_44 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_DisableDma

ADC_EXCLUSIVE_AREA_44 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_DisableDma

ADC_EXCLUSIVE_AREA_44 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_DisableDma

ADC_EXCLUSIVE_AREA_44 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_DisableDma

ADC_EXCLUSIVE_AREA_44 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_DisableDma

ADC_EXCLUSIVE_AREA_44 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_DisableDma

ADC_EXCLUSIVE_AREA_44 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_DisableDma

ADC_EXCLUSIVE_AREA_44 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_DisableDma

ADC_EXCLUSIVE_AREA_44 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_DisableDma

ADC_EXCLUSIVE_AREA_45 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_SetDmaClearSource

ADC_EXCLUSIVE_AREA_45 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_SetDmaClearSource

ADC_EXCLUSIVE_AREA_45 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_SetDmaClearSource

ADC_EXCLUSIVE_AREA_45 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_SetDmaClearSource

ADC_EXCLUSIVE_AREA_45 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_SetDmaClearSource

ADC_EXCLUSIVE_AREA_45 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_SetDmaClearSource

ADC_EXCLUSIVE_AREA_45 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_SetDmaClearSource

ADC_EXCLUSIVE_AREA_45 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_SetDmaClearSource

ADC_EXCLUSIVE_AREA_45 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the ADC_DMAE register from read/modify/write operation in Adc_Sar_Ip_SetDmaClearSource

ADC_EXCLUSIVE_AREA_46 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

ADC_EXCLUSIVE_AREA_46 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

ADC_EXCLUSIVE_AREA_46 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

ADC_EXCLUSIVE_AREA_46 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

ADC_EXCLUSIVE_AREA_46 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

ADC_EXCLUSIVE_AREA_46 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

ADC_EXCLUSIVE_AREA_46 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

ADC_EXCLUSIVE_AREA_46 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

ADC_EXCLUSIVE_AREA_46 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_EnableChannelDma

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Ipw_BctuEndListNotificationAdc0 to protect the A↩DC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Ipw_BctuEndListNotificationAdc1 to protect the A↩DC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Ipw_BctuEndListNotificationAdc2 to protect the A↩DC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDma

ADC_EXCLUSIVE_AREA_48 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDmaAll

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

ADC_EXCLUSIVE_AREA_48 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDmaAll

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

ADC_EXCLUSIVE_AREA_48 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDmaAll

ADC_EXCLUSIVE_AREA_48 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDmaAll

ADC_EXCLUSIVE_AREA_48 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDmaAll

ADC_EXCLUSIVE_AREA_48 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDmaAll

ADC_EXCLUSIVE_AREA_48 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the ADC_DMARx register from read/modify/write operation in Adc_Sar_Ip_DisableChannelDmaAll

ADC_EXCLUSIVE_AREA_54 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_54 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_54 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_54 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_54 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_54 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_54 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_54 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_54 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the BCTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_54 is used in function Adc_Ipw_BctuEndListNotificationAdc0 to protect the B↩ CTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_54 is used in function Adc_Ipw_BctuEndListNotificationAdc1 to protect the B↩ CTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_54 is used in function Adc_Ipw_BctuEndListNotificationAdc2 to protect the B↩ CTU_MCR register from read/modify/write operation in Bctu_Ip_SetGlobalTriggerEn

ADC_EXCLUSIVE_AREA_59 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_Ipw_BctuEndListNotificationAdc0 to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_Ipw_BctuEndListNotificationAdc1 to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_59 is used in function Adc_Ipw_BctuEndListNotificationAdc2 to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Ipw_BctuEndListNotificationAdc0 to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Ipw_BctuEndListNotificationAdc1 to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Adc_Ipw_BctuEndListNotificationAdc2 to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Ipw_BctuEndListNotificationAdc0 to protect the B↩CTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Ipw_BctuEndListNotificationAdc1 to protect the B↩CTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_67 is used in function Adc_Ipw_BctuEndListNotificationAdc2 to protect the B↩CTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Ipw_Adc0EndNormalChainNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Ipw_Adc0DmaTransferCompleteNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Ipw_Adc1EndNormalChainNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Ipw_Adc1DmaTransferCompleteNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Ipw_Adc2EndNormalChainNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Ipw_Adc2DmaTransferCompleteNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Ipw_Adc0EndInjectedChainNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Ipw_Adc1EndInjectedChainNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Ipw_Adc2EndInjectedChainNotification to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Ipw_BctuEndListNotificationAdc0 to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Ipw_BctuEndListNotificationAdc1 to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_68 is used in function Adc_Ipw_BctuEndListNotificationAdc2 to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_ConfigChanListSimple

ADC_EXCLUSIVE_AREA_69 is used in function ISR(Bctu_0_Isr) to protect the BCTU_FIFOERR register from read/modify/write operation in Bctu_Ip_IRQHandler

ADC_EXCLUSIVE_AREA_72 is used in function Adc_Ipw_UpdateGroupStateSdAdc to protect the SDADC_MCR register read/modify/write operation in Sdadc_Ip_Powerup

ADC_EXCLUSIVE_AREA_73 is used in function Adc_Ipw_UpdateGroupStateSdAdc to protect the SDADC_MCR register read/modify/write operation in Sdadc_Ip_Powerdown

ADC_EXCLUSIVE_AREA_82 is used in function Adc_Ipw_UpdateGroupStateSdAdc to protect the SDADC_RSER register read/modify/write operation in Sdadc_Ip_DisableInterruptEvents

ADC_EXCLUSIVE_AREA_82 is used in function Adc_Ipw_EndSoftwareConvSdAdc to protect the SDADC_RSER register read/modify/write operation in Sdadc_Ip_DisableInterruptEvents

ADC_EXCLUSIVE_AREA_84 is used in function Adc_Ipw_UpdateGroupStateSdAdc to protect the SDADC_MCR register read/modify/write operation in Sdadc_Ip_DisableHwTrigger

ADC_EXCLUSIVE_AREA_100 is used in function Adc_Ipw_UpdateGroupStateSdAdc to protect the DSPSS_DmaReadBufferWrapStatus register modify/write operation in DSPSS_DmaReadBufferWrapPtrCheck

ADC_EXCLUSIVE_AREA_102 is used in function Adc_Ipw_UpdateGroupStateSdAdc to protect the SDADC_RSER register odify/write operation in Sdadc_Ip_Powerdown

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

ADC_EXCLUSIVE_AREA_10 is used in function Adc_Sar_Ip_StartConversion to protect the updates for ADC_MCR register

ADC_EXCLUSIVE_AREA_11 is used in function Adc_Sar_Ip_SelfTest to protect the updates for ADC_MCR register

ADC_EXCLUSIVE_AREA_12 is used in function Adc_Sar_Ip_DoCalibration to protect the updates for ADC_MCR register

ADC_EXCLUSIVE_AREA_13 is used in function Adc_Sar_Ip_Powerup to protect the updates for ADC_MCR register

ADC_EXCLUSIVE_AREA_14 is used in function Adc_Sar_Ip_Powerdown to protect the updates for ADC_MCR register

ADC_EXCLUSIVE_AREA_15 is used in function Adc_Sar_Ip_SetClockMode to protect the updates for ADC_MCR register

ADC_EXCLUSIVE_AREA_16 is used in function Adc_Sar_Ip_SetAveraging to protect the updates for A↩ DC_MCR register

ADC_EXCLUSIVE_AREA_17 is used in function Adc_Sar_Ip_AbortConversion to protect the updates for ADC_MCR register

ADC_EXCLUSIVE_AREA_18 is used in function Adc_Sar_Ip_AbortChain to protect the updates for A↩ DC_MCR register

ADC_EXCLUSIVE_AREA_19 is used in function Adc_Sar_Ip_SetConversionMode to protect the updates for ADC_MCR register

ADC_EXCLUSIVE_AREA_20 is used in function Adc_Sar_Ip_SetCtuMode to protect the updates for A↩ DC_MCR register

ADC_EXCLUSIVE_AREA_21 is used in function Adc_Sar_Ip_SetExternalTrigger to protect the updates for ADC_MCR register

ADC_EXCLUSIVE_AREA_22 is used in function Adc_Sar_Ip_TempSenseGetTemp to protect the updates for ADC_MCR register

ADC_EXCLUSIVE_AREA_23 is used in function Adc_Sar_Ip_EnableChannel to protect the updates for ADC_NCMRx register

ADC_EXCLUSIVE_AREA_24 is used in function Adc_Sar_Ip_DisableChannel to protect the updates for ADC_NCMRx register

ADC_EXCLUSIVE_AREA_25 is used in function Adc_Sar_Ip_SelfTest to protect the updates for ADC_↩ NCMRx register

ADC_EXCLUSIVE_AREA_26 is used in function Adc_Sar_Ip_EnableChannel to protect the updates for ADC_JCMRx register

ADC_EXCLUSIVE_AREA_27 is used in function Adc_Sar_Ip_DisableChannel to protect the updates for ADC_JCMRx register

ADC_EXCLUSIVE_AREA_28 is used in function Adc_Sar_Ip_EnableChannelNotifications to protect the updates for ADC_CWENRx register

ADC_EXCLUSIVE_AREA_29 is used in function Adc_Sar_Ip_DisableChannelNotifications to protect the updates for ADC_CWENRx register

ADC_EXCLUSIVE_AREA_30 is used in function Adc_Sar_Ip_EnableChannelNotifications to protect the updates for ADC_CIMRx register

ADC_EXCLUSIVE_AREA_31 is used in function Adc_Sar_Ip_DisableChannelNotifications to protect the updates for ADC_CIMRx register

ADC_EXCLUSIVE_AREA_32 is used in function Adc_Sar_Ip_SetWdgThreshold to protect the updates for ADC_WTIMRx register

ADC_EXCLUSIVE_AREA_33 is used in function Adc_Sar_Ip_EnableNotifications to protect the updates for ADC_IMR register

ADC_EXCLUSIVE_AREA_34 is used in function Adc_Sar_Ip_DisableNotifications to protect the updates for ADC_IMR register

ADC_EXCLUSIVE_AREA_35 is used in function Adc_Sar_Ip_SelfTest to protect the updates for ADC_↩ STCRx register

ADC_EXCLUSIVE_AREA_36 is used in function Adc_Sar_Ip_DoCalibration to protect the updates for ADC_CALBISTREG register

ADC_EXCLUSIVE_AREA_37 is used in function Adc_Sar_Ip_SetResolution to protect the updates for ADC_CALBISTREG register

ADC_EXCLUSIVE_AREA_38 is used in function Adc_Sar_Ip_SetPresamplingSource to protect the updates for ADC_PSCR register

ADC_EXCLUSIVE_AREA_39 is used in function Adc_Sar_Ip_EnablePresampleConversion to protect the updates for ADC_PSCR register

ADC_EXCLUSIVE_AREA_40 is used in function Adc_Sar_Ip_DisablePresampleConversion to protect the updates for ADC_PSCR register

ADC_EXCLUSIVE_AREA_41 is used in function Adc_Sar_Ip_EnableChannelPresampling to protect the updates for ADC_PSRx register

ADC_EXCLUSIVE_AREA_42 is used in function Adc_Sar_Ip_DisableChannelPresampling to protect the updates for ADC_PSRx register

ADC_EXCLUSIVE_AREA_43 is used in function Adc_Sar_Ip_EnableDma to protect the updates for A↩ DC_DMAE register

ADC_EXCLUSIVE_AREA_44 is used in function Adc_Sar_Ip_DisableDma to protect the updates for A↩ DC_DMAE register

ADC_EXCLUSIVE_AREA_45 is used in function Adc_Sar_Ip_SetDmaClearSource to protect the updates for ADC_DMAE register

ADC_EXCLUSIVE_AREA_46 is used in function Adc_Sar_Ip_EnableChannelDma to protect the updates for ADC_DMARx register

ADC_EXCLUSIVE_AREA_47 is used in function Adc_Sar_Ip_DisableChannelDma to protect the updates for ADC_DMARx register

ADC_EXCLUSIVE_AREA_48 is used in function Adc_Sar_Ip_DisableChannelDmaAll to protect the updates for ADC_DMARx register

ADC_EXCLUSIVE_AREA_49 is used in function Adc_Sar_Ip_TempSenseEnable to protect the updates for ETSCTL register

ADC_EXCLUSIVE_AREA_50 is used in function Adc_Sar_Ip_TempSenseDisable to protect the updates for ETSCTL register

ADC_EXCLUSIVE_AREA_51 is used in function Adc_Sar_Ip_SetClockMode to protect the updates for ADC_AMSIO register

ADC_EXCLUSIVE_AREA_54 is used in function Bctu_Ip_SetGlobalTriggerEn to protect the updates for BCTU_MCR register

ADC_EXCLUSIVE_AREA_55 is used in function Bctu_Ip_SetLowPowerMode to protect the updates for BCTU_MCR register

ADC_EXCLUSIVE_AREA_56 is used in function Bctu_Ip_EnableNotifications to protect the updates for BCTU_MCR register

ADC_EXCLUSIVE_AREA_57 is used in function Bctu_Ip_DisableNotifications to protect the updates for BCTU_MCR register

ADC_EXCLUSIVE_AREA_58 is used in function Bctu_Ip_EnableHwTrigger to protect the updates for B↩ CTU_TRGCFGx register

ADC_EXCLUSIVE_AREA_59 is used in function Bctu_Ip_DisableHwTrigger to protect the updates for BCTU_TRGCFGx register

ADC_EXCLUSIVE_AREA_60 is used in function Bctu_Ip_StopLoopConversions to protect the updates for BCTU_TRGCFGx register

ADC_EXCLUSIVE_AREA_61 is used in function Bctu_Ip_Init to protect the BCTU_TRGCFGx register from read/modify/write operation in Bctu_Ip_ConfigTrigger

ADC_EXCLUSIVE_AREA_61 is used in function Bctu_Ip_SetTriggerChnListAddr to protect the updates for BCTU_TRGCFGx register

ADC_EXCLUSIVE_AREA_61 is used in function Bctu_Ip_ConfigTrigger to protect the updates for BCT↩ U_TRGCFGx register

ADC_EXCLUSIVE_AREA_62 is used in function Bctu_Ip_Deinit to protect the updates for BCTU_TR↩ GCFGx register

ADC_EXCLUSIVE_AREA_63 is used in function Bctu_Ip_EnableNotifications to protect the updates for BCTU_FIFOCR register

ADC_EXCLUSIVE_AREA_64 is used in function Bctu_Ip_DisableNotifications to protect the updates for BCTU_FIFOCR register

ADC_EXCLUSIVE_AREA_65 is used in function Bctu_Ip_Deinit to protect the BCTU_FIFOCR register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_65 is used in function Bctu_Ip_Init to protect the updates for BCTU_FIFOCR register

ADC_EXCLUSIVE_AREA_66 is used in function Bctu_Ip_Init to protect the BCTU_FIFOWM register from read/modify/write operation in Bctu_Ip_SetFifoWatermark

ADC_EXCLUSIVE_AREA_66 is used in function Bctu_Ip_Deinit to protect the BCTU_FIFOWM register from read/modify/write operation in Bctu_Ip_SetFifoWatermark

ADC_EXCLUSIVE_AREA_66 is used in function Bctu_Ip_SetFifoWatermark to protect the updates for BCTU_FIFOWM register

ADC_EXCLUSIVE_AREA_67 is used in function Bctu_Ip_Deinit to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_67 is used in function Bctu_Ip_Init to protect the updates for BCTU_LISTC↩ HRx register

ADC_EXCLUSIVE_AREA_67 is used in function Bctu_Ip_ConfigChanList to protect the updates for BC↩ TU_LISTCHRx register

ADC_EXCLUSIVE_AREA_67 is used in function Bctu_Ip_ConfigChanListSimple to protect the updates for BCTU_LISTCHRx register

ADC_EXCLUSIVE_AREA_68 is used in function Bctu_Ip_Deinit to protect the BCTU_LISTCHRx register from read/modify/write operation in Bctu_Ip_Init

ADC_EXCLUSIVE_AREA_68 is used in function Bctu_Ip_Init to protect the updates for BCTU_LISTC↩ HRx register

ADC_EXCLUSIVE_AREA_68 is used in function Bctu_Ip_ConfigChanList to protect the updates for BC↩ TU_LISTCHRx register

ADC_EXCLUSIVE_AREA_68 is used in function Bctu_Ip_ConfigChanListSimple to protect the updates for BCTU_LISTCHRx register

ADC_EXCLUSIVE_AREA_68 is used in function Bctu_SetConvListLastElement to protect the updates for BCTU_LISTCHRx register

ADC_EXCLUSIVE_AREA_69 is used in function Bctu_Ip_IRQHandler to protect the updates for BCTU↩ _FIFOERR register

ADC_EXCLUSIVE_AREA_70 is used in function Bctu_Ip_ClearStatusFlags to protect the updates for B↩ CTU_MSR register

ADC_EXCLUSIVE_AREA_71 is used in function Bctu_Ip_SwTriggerConversion to protect the updates for BCTU_SFTRGRx register

ADC_EXCLUSIVE_AREA_72 is used in function Sdadc_Ip_Powerup to protect the updates for SDADC↩ _MCR register

ADC_EXCLUSIVE_AREA_73 is used in function Sdadc_Ip_Powerdown to protect the updates for SDAD↩ C_MCR register

ADC_EXCLUSIVE_AREA_74 is used in function Sdadc_Ip_EnableWatchdog to protect the updates for SDADC_MCR register

ADC_EXCLUSIVE_AREA_75 is used in function Sdadc_Ip_DisableWatchdog to protect the updates for SDADC_MCR register

ADC_EXCLUSIVE_AREA_76 is used in function Sdadc_Ip_EnableWraparound to protect the updates for SDADC_MCR register

ADC_EXCLUSIVE_AREA_77 is used in function Sdadc_Ip_DisableWraparound to protect the updates for SDADC_MCR register

ADC_EXCLUSIVE_AREA_78 is used in function Sdadc_Ip_SetInputChannel to protect the updates for SDADC_MCR registers

ADC_EXCLUSIVE_AREA_79 is used in function Sdadc_Ip_FlushFifo to protect the updates for SDADC↩ _FCR register

ADC_EXCLUSIVE_AREA_80 is used in function Sdadc_Ip_EnableDmaEvents to protect the updates for SDADC_RSER register

ADC_EXCLUSIVE_AREA_81 is used in function Sdadc_Ip_EnableInterruptEvents to protect the updates for SDADC_RSER register

ADC_EXCLUSIVE_AREA_82 is used in function Sdadc_Ip_DisableInterruptEvents to protect the updates for SDADC_RSER register

ADC_EXCLUSIVE_AREA_83 is used in function Sdadc_Ip_EnableHwTrigger to protect the updates for SDADC_MCR register

ADC_EXCLUSIVE_AREA_84 is used in function Sdadc_Ip_DisableHwTrigger to protect the updates for SDADC_MCR register

ADC_EXCLUSIVE_AREA_85 is used in function Sdadc_Ip_SetHwTrigger to protect the updates for SD↩ ADC_MCR register

ADC_EXCLUSIVE_AREA_86 is used in function Sdadc_InitDspssThread to protect the updates for DSP↩ SS_DmaReadBufferWrapStatus

ADC_EXCLUSIVE_AREA_87 is used in function Sdadc_Ip_ReloadConversion to protect the updates for DSPSS_SCHEDULER_XMEM_THx register

ADC_EXCLUSIVE_AREA_88 is used in function Sdadc_InitDspssThread to protect the updates for DSP↩ _CORE_BUFFER_CFG register

ADC_EXCLUSIVE_AREA_89 is used in function Sdadc_InitDspssThread to protect the updates for DSP↩ _CORE_BUFFER_CFG register

ADC_EXCLUSIVE_AREA_90 is used in function Sdadc_Ip_FlushFifo to protect the updates for DSP_C↩ ORE_BUFFER_CFG register

ADC_EXCLUSIVE_AREA_91 is used in function Sdadc_InitDspssThread to protect the updates for DSP↩ SS_INTERRUPT_ENABLE_REGISTER register

ADC_EXCLUSIVE_AREA_92 is used in function Sdadc_InitDspssThread to protect the updates for DSP↩ SS_INTERRUPT_ENABLE_REGISTER2 register

ADC_EXCLUSIVE_AREA_93 is used in function DSPSS_InterruptDisable to protect the updates for DS↩ PSS_INTERRUPT_ENABLE_REGISTER register

ADC_EXCLUSIVE_AREA_94 is used in function DSPSS_InterruptDisable to protect the updates for DS↩ PSS_INTERRUPT_ENABLE_REGISTER2 register

ADC_EXCLUSIVE_AREA_95 is used in function Sdadc_InitDspssThread to protect the updates for DSP↩ SS_DMA_BUFFER_CFG register

ADC_EXCLUSIVE_AREA_96 is used in function Sdadc_GainCalibration to protect the updates for SDA↩ DC_MCR register

ADC_EXCLUSIVE_AREA_97 is used in function Sdadc_GainCalibration to protect the updates for SDA↩ DC_FCR register

ADC_EXCLUSIVE_AREA_98 is used in function Sdadc_OffsetCalibration to protect the updates for SD↩ ADC_MCR register

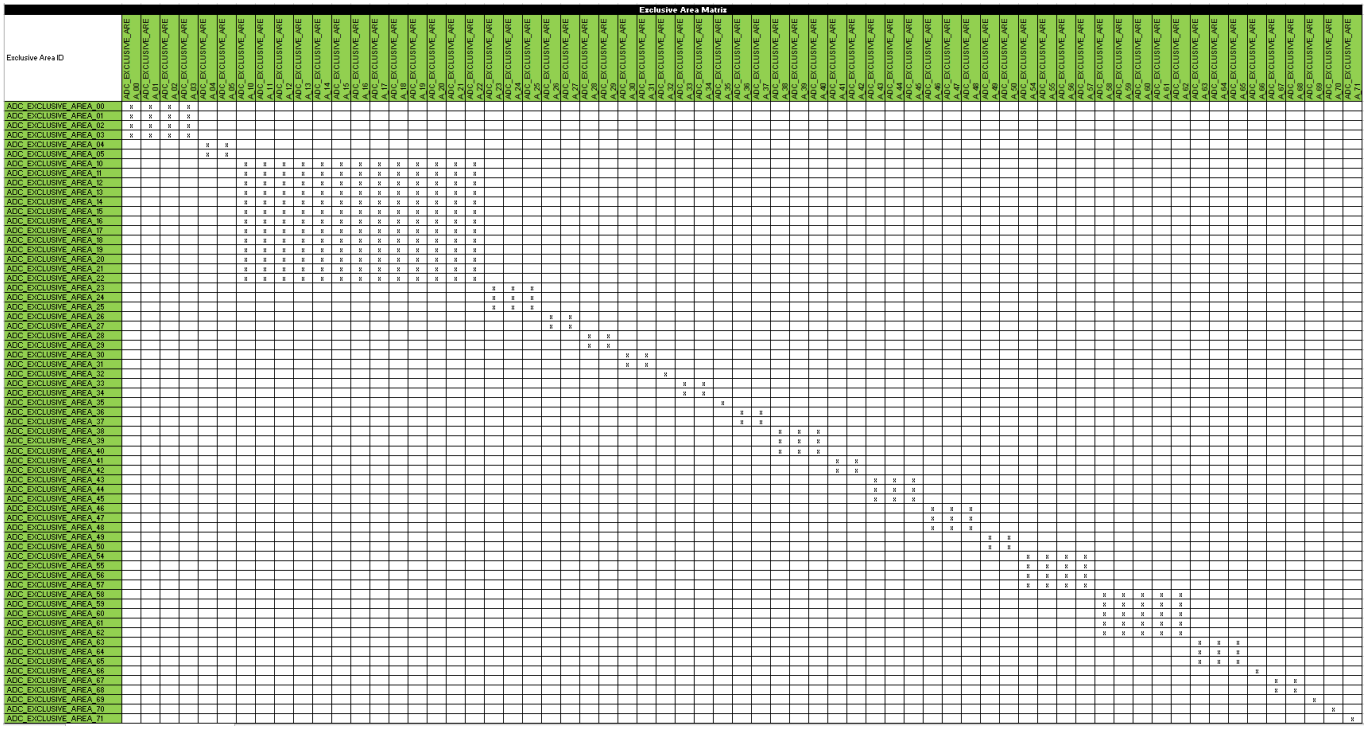
ADC_EXCLUSIVE_AREA_99 is used in function Sdadc_OffsetCalibration to protect the updates for SD↩ ADC_FCR register

ADC_EXCLUSIVE_AREA_100 is used in function DSPSS_DmaReadBufferWrapPtrCheck to protect the updates for DSPSS_DmaReadBufferWrapStatus

ADC_EXCLUSIVE_AREA_101 is used in function Sdadc_Ip_SetInputChannel to protect the updates for SDADC_CSR register

ADC_EXCLUSIVE_AREA_102 is used in function Sdadc_Ip_Powerdown to protect the updates for SDA↩ DC_RSER register

ADC_EXCLUSIVE_AREA_103 is used in function Sdadc_Ip_FlushFifo to protect the updates for DSPS↩ S_DMA_BUFFER_CFG register



Exclusive Area Matrix																																		
Exclusive Area ID	ADC_EXCLUSIVE_AREA_72	ADC_EXCLUSIVE_AREA_73	ADC_EXCLUSIVE_AREA_74	ADC_EXCLUSIVE_AREA_75	ADC_EXCLUSIVE_AREA_76	ADC_EXCLUSIVE_AREA_77	ADC_EXCLUSIVE_AREA_78	ADC_EXCLUSIVE_AREA_79	ADC_EXCLUSIVE_AREA_80	ADC_EXCLUSIVE_AREA_81	ADC_EXCLUSIVE_AREA_82	ADC_EXCLUSIVE_AREA_83	ADC_EXCLUSIVE_AREA_84	ADC_EXCLUSIVE_AREA_85	ADC_EXCLUSIVE_AREA_86	ADC_EXCLUSIVE_AREA_87	ADC_EXCLUSIVE_AREA_88	ADC_EXCLUSIVE_AREA_89	ADC_EXCLUSIVE_AREA_90	ADC_EXCLUSIVE_AREA_91	ADC_EXCLUSIVE_AREA_92	ADC_EXCLUSIVE_AREA_93	ADC_EXCLUSIVE_AREA_94	ADC_EXCLUSIVE_AREA_95	ADC_EXCLUSIVE_AREA_96	ADC_EXCLUSIVE_AREA_97	ADC_EXCLUSIVE_AREA_98	ADC_EXCLUSIVE_AREA_99	ADC_EXCLUSIVE_AREA_100	ADC_EXCLUSIVE_AREA_101	ADC_EXCLUSIVE_AREA_102	ADC_EXCLUSIVE_AREA_103
ADC_EXCLUSIVE_AREA_72	x	x	x	x	x	x	x					x	x	x											x		x					
ADC_EXCLUSIVE_AREA_73	x	x	x	x	x	x	x					x	x	x											x		x					
ADC_EXCLUSIVE_AREA_74	x	x	x	x	x	x	x					x	x	x											x		x					
ADC_EXCLUSIVE_AREA_75	x	x	x	x	x	x	x					x	x	x											x		x					
ADC_EXCLUSIVE_AREA_76	x	x	x	x	x	x	x					x	x	x											x		x					
ADC_EXCLUSIVE_AREA_77	x	x	x	x	x	x	x					x	x	x											x		x					
ADC_EXCLUSIVE_AREA_78	x	x	x	x	x	x	x					x	x	x											x		x					
ADC_EXCLUSIVE_AREA_79								x																		x		x				
ADC_EXCLUSIVE_AREA_80									x	x	x																			x	
ADC_EXCLUSIVE_AREA_81									x	x	x																			x	
ADC_EXCLUSIVE_AREA_82									x	x	x																			x	
ADC_EXCLUSIVE_AREA_83	x	x	x	x	x	x	x					x	x	x											x		x					
ADC_EXCLUSIVE_AREA_84	x	x	x	x	x	x	x					x	x	x											x		x					
ADC_EXCLUSIVE_AREA_85	x	x	x	x	x	x	x					x	x	x											x		x					
ADC_EXCLUSIVE_AREA_86															x														x			
ADC_EXCLUSIVE_AREA_87																x																
ADC_EXCLUSIVE_AREA_88																	x	x	x													
ADC_EXCLUSIVE_AREA_89																	x	x	x													
ADC_EXCLUSIVE_AREA_90																	x	x	x													
ADC_EXCLUSIVE_AREA_91																				x		x										
ADC_EXCLUSIVE_AREA_92																					x		x									
ADC_EXCLUSIVE_AREA_93																				x		x										
ADC_EXCLUSIVE_AREA_94																					x		x									
ADC_EXCLUSIVE_AREA_95																								x								x
ADC_EXCLUSIVE_AREA_96	x	x	x	x	x	x	x					x	x	x											x		x					
ADC_EXCLUSIVE_AREA_97								x																		x		x				
ADC_EXCLUSIVE_AREA_98	x	x	x	x	x	x	x					x	x	x											x		x					
ADC_EXCLUSIVE_AREA_99								x																		x		x				
ADC_EXCLUSIVE_AREA_100															x														x			
ADC_EXCLUSIVE_AREA_101																														x		
ADC_EXCLUSIVE_AREA_102									x	x	x																				x	
ADC_EXCLUSIVE_AREA_103																								x								x

(Extracted table from RTD_ADC_EXCLUSIVE_AREAS.xlsx)

5.2 Exclusive areas not available on this platform

None.

5.3 Peripheral Hardware Requirements

This device provides three General-purpose ADC :ADC HW Unit 0 (ADC0), ADC HW Unit 1 (ADC1), ADC HW Unit 2(ADC2). 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 for S32K39X derivative:

ISR Name	HW INT Vector	Observations
ISR(Bctu_0_Isr)	87	BCTU0 interrupt handler
ISR(Bctu_1_Isr)	87	BCTU1 interrupt handler
ISR(Sdadc_Isr)	179	SDADC interrupt handler
ISR(Adc_Sar_0_Isr)	180	ADC0 interrupt handler
ISR(Adc_Sar_1_Isr)	181	ADC1 interrupt handler
ISR(Adc_Sar_2_Isr)	182	ADC2 interrupt handler
ISR(Adc_Sar_3_Isr)	185	ADC3 interrupt handler
ISR(Adc_Sar_4_Isr)	186	ADC4 interrupt handler
ISR(Adc_Sar_5_Isr)	187	ADC5 interrupt handler
ISR(Adc_Sar_6_Isr)	188	ADC6 interrupt handler
ISR(Dspss_Thread_0_Isr)	225	DSPSS Thread 0 interrupt handler
ISR(Dspss_Thread_1_Isr)	226	DSPSS Thread 1 interrupt handler
ISR(Dspss_Thread_2_Isr)	227	DSPSS Thread 2 interrupt handler
ISR(Dspss_Thread_3_Isr)	228	DSPSS Thread 3 interrupt handler

Table with Interrupt Service Routines used for S32K3XX and M27X derivatives:

ISR Name	HW INT Vector	Observations
ISR(Bctu_0_Isr)	87	BCTU interrupt handler
ISR(Adc_Sar_0_Isr)	180	ADC0 interrupt handler
ISR(Adc_Sar_1_Isr)	181	ADC1 interrupt handler
ISR(Adc_Sar_2_Isr)	182	ADC2 interrupt handler

Note

Adc_Sar_2_Isr is not applicable for M27X, S32K342, S32K322, S32K341, S32K312 and S32K311 derivatives.

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_DmaTransferCompleteX function should be configured for DMA notification parameter. This is required to update ADC driver internal statuses.

The following functions should be configured for DMA notification parameter for K39X derivative.

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
Adc_Ipw_Adc2DmaTransferCompleteNotification	DMA notification parameter for transfer complete, ADC unit 2
Adc_Ipw_Adc3DmaTransferCompleteNotification	DMA notification parameter for transfer complete, ADC unit 3

Function Name	Observations
Adc_Ipw_Adc4DmaTransferCompleteNotification	DMA notification parameter for transfer complete, ADC unit 4
Adc_Ipw_Adc5DmaTransferCompleteNotification	DMA notification parameter for transfer complete, ADC unit 5
Adc_Ipw_Adc6DmaTransferCompleteNotification	DMA notification parameter for transfer complete, ADC unit 6
Adc_Ipw_Sdadc0DmaTransferComplete↩ Notification	DMA notification parameter for transfer complete, SDADC unit 0
Adc_Ipw_Sdadc1DmaTransferComplete↩ Notification	DMA notification parameter for transfer complete, SDADC unit 1
Adc_Ipw_Sdadc2DmaTransferComplete↩ Notification	DMA notification parameter for transfer complete, SDADC unit 2
Adc_Ipw_Sdadc3DmaTransferComplete↩ Notification	DMA notification parameter for transfer complete, SDADC unit 3
Bctu_Ip_Bctu0Fifo1DmaComplete	DMA notification parameter for transfer complete from Bctu0 FIFO 1 when it exceeded Watermark value
Bctu_Ip_Bctu0Fifo2DmaComplete	DMA notification parameter for transfer complete from Bctu0 FIFO 2 when it exceeded Watermark value
Bctu_Ip_Bctu0Fifo3DmaComplete	DMA notification parameter for transfer complete from Bctu0 FIFO 3 when it exceeded Watermark value
Bctu_Ip_Bctu1Fifo1DmaComplete	DMA notification parameter for transfer complete from Bctu1 FIFO 1 when it exceeded Watermark value
Bctu_Ip_Bctu1Fifo2DmaComplete	DMA notification parameter for transfer complete from Bctu1 FIFO 2 when it exceeded Watermark value
Bctu_Ip_Bctu1Fifo3DmaComplete	DMA notification parameter for transfer complete from Bctu1 FIFO 3 when it exceeded Watermark value

The following functions should be configured for DMA notification parameter for K3XX and M27X derivatives:

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
Adc_Ipw_Adc2DmaTransferCompleteNotification	DMA notification parameter for transfer complete, ADC unit 2
Bctu_Ip_Bctu0Fifo1DmaComplete	DMA notification parameter for transfer complete from Bctu FIFO 1 when it exceeded Watermark value
Bctu_Ip_Bctu0Fifo2DmaComplete	DMA notification parameter for transfer complete from Bctu FIFO 2 when it exceeded Watermark value

ADC unit 2 is not available on M27X, S32K342, S32K322, S32K341, S32K312 and S32K311 platforms thus the corresponding notification function does not apply here.

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 init before ADC.
- Rm: In DMA mode, the Rm is used to configure the DMAMUX allocated for all ADC HW units. Rm should be init before ADC.
- 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, everyone having attached a Resource file: s32k358_172mqfp, s32k358_289bga.
- RTE: Used to manage the exclusive area inside 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.

5.8.2 User Mode configuration in AutosarOS

When User mode is enabled, the driver may has 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.

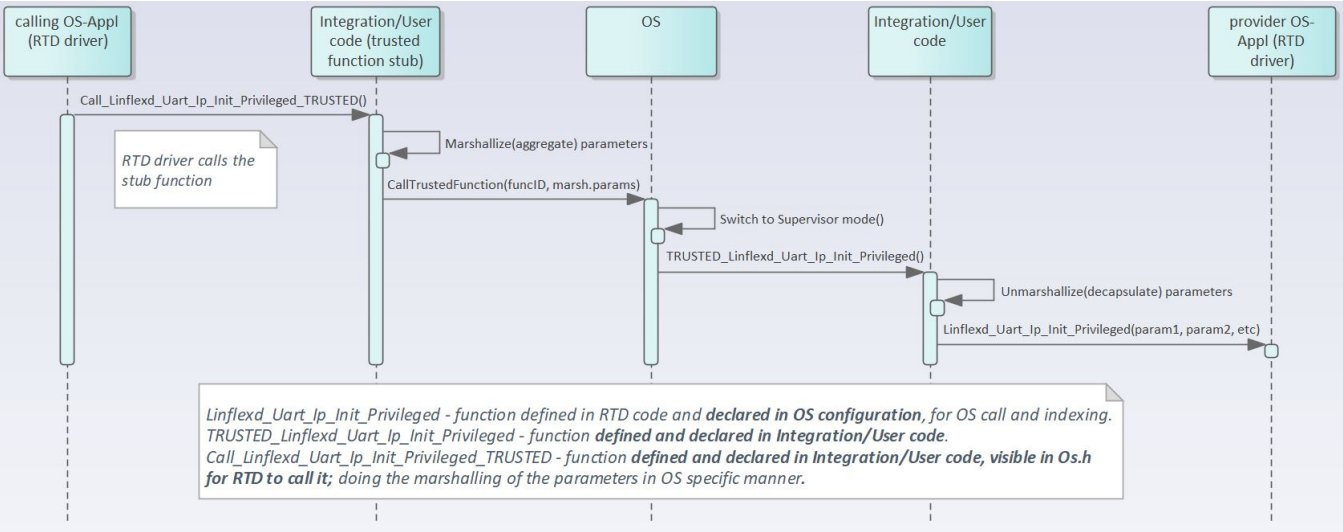


Figure 5.1 Example sequence chart for calling `Linflexd_Uart_Ip_Init_Privileged` as trusted function

5.9 Multicore support

1. The Adc implements the "Autosar 4.4 MCAL Multicore Distribution" according to type II, in which the mappable element is set to HW Unit. For additional details, please refer to AUTOSAR_EXP_BSWDistribution↩ Guide.

2. The Adc and the mappable elements can only be allocated to one ECUC partition, by means of "AdcEcuc↩ PartionRef". If there is only one Adc instance used, the Adc behavior reverts to single-core implementation, similar to previous Autosar versions. If the number of Adc instances is more than one, the Adc will enforce the following multicore assumptions:

   - The Adc assumes there is a single EcucPartition allocated per core. Internally, the module will use the Core ID returned by GetCoreID API to reference the appropriate global data and configuration elements.
   - The Adc assumes the EcucCoreIDs are defined in a compact/consecutive order, starting from zero. The rationale is that the number of EcucPartitions is used for dimensioning the Adc internal variables and the EcucCoreIDs are used for indexing those variables. (AR-86601 Zero based and dense IDs for OS-Cores and OSApplications).
   - The Adc assumes that initialization is performed on each core, Adc_Init() is called separately for each core, using a different configuration structure. (Type II)
   - The Adc initialization expects the upper layer will pass the correct initialization pointer, specific to the partition in which the driver is to be used. For example: EcucPartition_1 is assigned to CoreID 1; Adc_Init function will be called with Adc_Config_EcucPartition_1 configuration structure, on Core 1.
   - The Adc will check upon each API call if the requested resource is configured to be available on the current core, if DET error reporting is enabled.
   - The Adc requires that all variables in NonCacheable MemMap sections be allocated accordingly, to avoid data corruption in multicore context.

- The Adc assumes that each interrupt is routed by the system only to the core on which is supposed to be serviced.
- The Adc assumes that CTU HW triggered groups must be configured and used only on a single core.

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.

If the Analog Watchdog feature is used, the ADC Driver provides a notification callback per channel that is called whenever the measurement value is out of the defined range. The notifications can be configured as pointers to user defined functions. If WDG 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

- Sections to be defined in Adc_MemMap.h
- Linker command file

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_INIT_U↩ NSPECIFIED	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 and initialized by start-up code.
ADC_STOP_SEC_VAR_INIT_UN↩ SPECIFIED	Variables	End of above section.

Section name	Type of section	Description
ADC_START_SEC_VAR_CLEARED_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_CLEARED_8	Variables	End of above section.
ADC_START_SEC_VAR_CLEARED_UNSPECIFIED_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_CLEARED_UNSPECIFIED_NO_CACHEABLE	Non-Cacheable Variables	End of above section.
ADC_START_SEC_VAR_CLEARED_16_NO_CACHEABLE	Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 16 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_CLEARED_16_NO_CACHEABLE	Variables	End of above section.
ADC_START_SEC_VAR_CLEARED_32_NO_CACHEABLE	Variables	Used for variables, structures, arrays when the SIZE (alignment) does not fit the criteria of 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_CLEARED_32_NO_CACHEABLE	Variables	End of above section.
ADC_START_SEC_CONFIG_DATA_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_DATA_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 availablity 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 assumptions for driver

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 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_00092	The integrator shall allocate a single EcucPartition per core or the partition in which the Adc is allocated shall be exclusively mapped to a core. Note: Internally, the Adc will use the Core ID returned by GetCoreID API to reference the appropriate global data and configuration elements, that is why a core should reference only one configured partition.
EA_RTD_00093	The application shall define EcucCoreIDs in a compact/consecutive order, starting from zero.
EA_RTD_00094	When multicore support is enabled, the application shall call Adc_Init() for each core, using the dedicated configuration pointer for that core.
EA_RTD_00096	The application shall pass the correct initialization pointer, specific to the partition in which the driver is to be used.
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.