Integration Manual

for S32K14X ADC Driver

Document Number: IM2ADCASR4.2 Rev0002R1.0.1
Rev. 1.0

Contents

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

Chapter 9
Integration Steps

Chapter 10
ISR Reference

Chapter 11
External Assumptions for ADC driver

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	13/07/2018	NXP MCAL Team	Updated version for ASR 4.2.2S32K14X1.0.1 Release

Chapter 2
Introduction

This integration manual describes the integration requirements for Adc Driver for S32K14X microcontrollers.

2.1 Supported Derivatives

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

Table 2-1. S32K14X Derivatives

NXP Semiconductors	s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64

All of the above microcontroller devices are collectively named as S32K14X .

2.2 Overview

AUTOSAR (AUTomotive Open System ARchitecture) is an industry partnership working to establish standards for software interfaces and software modules for automobile electronic control systems.

AUTOSAR

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.
- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".

- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

Boldface type: Bold is used for important terms, notes and warnings.

Italic font: Italic typeface is used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

Note

This is a note.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

Term	Definition
ADC	Analog to Digital Converter
API	Application Programming Interface
ASM	Assembler
AUTOSAR	Automotive Open System Architecture
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CS	Chip Select
CTU	Cross Trigger Unit
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
FIFO	First In First Out
LSB	Least Signifigant Bit

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
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

Table 2-3. Reference List

#	Title	Version
1	Specification of Adc Driver	AUTOSAR Release 4.2.2
2	S32K14X Reference Manual	Reference Manual, Rev. 7, 04/2018
3	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	30/11/2017
4	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	30/11/2017
5	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	30/11/2017
6	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	30/11/2017
7	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	26/02/2018

Chapter 3
Building the Driver

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

3.1 Build Options

The Adc driver files are compiled using

- Green Hills Multi 7.1.4 / Compiler 2017.1.4
- (Linaro GCC 6.3-2017.06~dev) 6.3.1 20170509 (Thu Dec 7 13:28:42 CST 2017 build.sh rev=g7fea41d s=L631 Earmv7 -V release_g7fea41d_build_Fed_Earmv7)
- IAR: V8.11.2

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

Note

The TS_T40D2M10I1R0 plugin name is composed as follow:

TS_T = Target_Id

D = Derivative_Id

M = SW_Version_Major

I = SW_Version_Minor

R = Revision

(i.e. Target_Id = 40 identifies CORTEXM architecture and Derivative_Id = 2 identifies the S32K14X)

3.1.1 GHS Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-ansi	Specifies ANSI C with extensions. This mode extends the ANSI X3.159-1989 standard with certain useful and compatible constructs.
-Osize	Optimize for size.
-dual_debug	Enables the generation of DWARF, COFF, or BSD debugging information in the object file
-G	Generates source level debugging information and allows procedure call from debugger's command line.
--no_exceptions	Disables support for exception handling
-Wundef	Generates warnings for undefined symbols in preprocessor expressions
-Wimplicit-int	Issues a warning if the return type of a function is not declared before it is called
-Wshadow	Issues a warning if the declaration of a local variable shadows the declaration of a variable of the same name declared at the global scope, or at an outer scope
-Wtrigraphs	Issues a warning for any use of trigraphs
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
--prototype_errors	Generates errors when functions referenced or called have no prototype
--incorrect_pragma_warnings	Valid #pragma directives with wrong syntax are treated as warnings
-noslashcomment	C++ like comments will generate a compilation error
-preprocess_assembly_files	Preprocesses assembly files
-nostartfile	Do not use Start files
--short_enum	Store enumerations in the smallest possible type
-c	Produces an object file (called input-file.o) for each source file.
--no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup.
-keeptempfiles	Prevents the deletion of temporary files after they are used. If an assembly language file is created by the compiler, this option will place it in the current directory instead of the temporary directory. Produces an object file (called input-file.o) for each source file.
-list	Creates a listing by using the name of the object file with the .lst extension. Assembler option
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DDISABLE_MCAL_INTERMODULE_ASR_CHECK	-D defines a preprocessor symbol to disable the inter-module version check for AR_RELEASE versions. DISABLE_MCAL_INTERMODULE_ASR_CHECK: By default in the package, drivers are compiled to perform the inter-module version check as per Autosar BSW004. When the inter-module version check needs to be disabled then the DISABLE_MCAL_INTERMODULE_ASR_CHECK global define must be added to the list of compiler options.
-DGHS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GHS preprocessor symbol.

Table 3-2. Assembler Options

Option	Description
-cpu=cortexm4	Selects target processor: Arm Cortex M4
-cpu=cortexm0plus	Selects target processor: Arm Cortex M0+
-c	Produces an object file (called input-file.o) for each source file.
-preprocess_assembly_files	Preprocesses assembly files
-asm=list	Creates a listing by using the name of the object file with the .lst extension. Assembler option

Table 3-3. Linker Options

Option	Description
-Mn	Map file numeric ordering
-delete	Removal from the executable of functions that are unused and unreferenced
-v	Display removed unused functions
-ignore_debug_references	Ignores relocations from DWARF debug sections when using -delete.
-map	Creates a detailed map file
-keepmap	Keep the map file in the event of a link error
-lstartup	Link libstartup library -Run-time environment startup routines
-lsys	Link libsys library -Run-time environment system routines
-larch	Link libarch library -Target-specific run-time support. Any file produced by the Green Hills Compiler may depend on symbols in this library.
-lansi	Link libansi library -the standard C library
-L(/lib/thumb2)	Link thumb2 library
-lutf8_s32	Include utf8_s32.a to use the Wide Character Functions

3.1.2 IAR Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--cpu_mode=thumb	Selects generating code that executes in Thumb state.
--endian=little	Specifies the endianess of core: little endian.
-Ohz	Sets the optimization level to High, favoring size.
-c	Produces an object file (called input-file.o) for each source file.
--no_clustering	Disables static clustering optimizations.
--no_mem_idioms	Makes the compiler to not optimize code sequences that clear, set, or copy a memory region.
--no_explicit_zero_opt	Places the zero initialized variables in data section instead of bss.
--debug	Makes the compiler include information in the object modules.

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
--diag_suppress=Pa050	Suppresses diagnostic messages (warnings) about non-standard line endings.
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DIAR	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the IAR preprocessor symbol.
--require_prototypes	Forces the compiler to verify that all functions have proper prototypes.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by compiler.
--no_system_include	Disables the automatic search for system include files.
-e	Enables language extensions. This option is needed by FLS driver which uses _packed structures.

Table 3-5. Assembler Options

Option	Description
--cpu=Cortex-M4	Selects target processor: Arm Cortex M4
--cpu=Cortex-M0+	Selects target processor: Arm Cortex M0+
--cpu_mode=thumb	Selects generating code that executes in Thumb state.
-g	Use this option to disable the automatic search for system include files.

Table 3-6. Linker Options

Option	Description
--map filename	Produces a map file.
--no_library_search	Disables automatic runtime library search.
--entry _start	Treats the symbol _start as a root symbol and as the start of the application.
--enable_stack_usage	Enables stack usage analysis.
--skip_dynamic_initialization	Suppress dynamic initialization during system startup.
--no_wrap_diagnostics	Disables line wrapping of diagnostic messages issued by linker.
--config	Specifies the configuration file to be used by the linker.

3.1.3 GCC Compiler/Linker/Assembler Options

Table 3-7. Compiler Options

Option	Description
-c	Produces an object file (called input-file.o) for each source file.
-Os	Use optimization for size.

Table continues on the next page...

Table 3-7. Compiler Options (continued)

Option	Description
-ggdb3	Produce debugging information for use by GDB. Level 3 includes extra information, such as all the macro definitions present in the program.
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-mthumb	Selects generating code that executes in Thumb state.
-ansi	Specifies ANSI C with extensions.
-mlittle-endian	Generate code for a processor running in little-endian mode.
-fomit-frame-pointer	Removes the frame pointer for all functions, which might make debugging harder.
-msoft-float	Use software floating-point instructions.
-fno-common	Specifies that the compiler should place uninitialized global variables in the data section of the object file, rather than generating them as common blocks.
-Wall	Enables all the warnings about constructions that some users consider questionable, and that are easy to avoid even in conjunction with macros.
-Wextra	Enables some extra warning flags that are not enabled by '-Wall'.
-Wstrict-prototypes	Warn if a function is declared or defined without specifying the argument types.
-Wno-sign-compare	Do not warn when a comparison between signed and unsigned values could produce an incorrect result when the signed value is converted to unsigned.
-fstack-usage	Geneates an extra file that specifies the maximum amount of stack used, on a per-function basis.
-fdump-ipa-all	Enables all inter-procedural analysis dumps.
-Werror=implicit-function-declaration	Generates an error when the prototype of the function is not defined..
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DGCC	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GCC preprocessor symbol.

Table 3-8. Assembler Options

Option	Description
-mcpu=cortex-m4	Selects target processor: Arm Cortex M4
-mcpu=cortex-m0plus	Selects target processor: Arm Cortex M0+
-c	Produces an object file (called input-file.o) for each source file.
-mthumb	This option specifies that the assembler should start assembling Thumb instructions.
-x assembler-with-cpp	Indicates that the assembly code contains C directives and the C preprocessor must be run.

Table 3-9. Linker Options

Option	Description
-Map=filename	Print a link map to the file mapfile.

Table continues on the next page...

Table 3-9. Linker Options (continued)

Option	Description
-T scriptfile	Use scriptfile as the linker script. This script replaces ld's default linker script(rather than adding to it), so commandfile must specify everything necessary to describe the output file.

3.2 Files required for Compilation

This section describes the include files required to compile, assemble (if assembler code) and link the Adc driver for S32K14X 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_T40D2M10I1R0\include\Adc.h
- ..\Adc_TS_T40D2M10I1R0\include\Adc_Adc12bsarv2.h
- ..\Adc_TS_T40D2M10I1R0\include\Adc_Adc12bsarv2_CfgEx.h
- ..\Adc_TS_T40D2M10I1R0\include\Adc_EnvCfg.h
- ..\Adc_TS_T40D2M10I1R0\include\Adc_Ipw.h
- ..\Adc_TS_T40D2M10I1R0\include\Adc_Pdb.h
- ..\Adc_TS_T40D2M10I1R0\include\Adc_Reg_eSys_Adc12bsarv2.h
- ..\Adc_TS_T40D2M10I1R0\include\Adc_Reg_eSys_Pdb.h
- ..\Adc_TS_T40D2M10I1R0\include\Adc_Types.h
- ..\Adc_TS_T40D2M10I1R0\src\Adc.c
- ..\Adc_TS_T40D2M10I1R0\src\Adc_Adc12bsarv2.c
- ..\Adc_TS_T40D2M10I1R0\src\Adc_Adc12bsarv2_Irq.c
- ..\Adc_TS_T40D2M10I1R0\src\Adc_Ipw.c
- ..\Adc_TS_T40D2M10I1R0\src\Adc_Pdb.c
- ..\Adc_TS_T40D2M10I1R0\src\Adc_Pdb_Irq.c

Adc Generated Files

- Adc_Cfg.h
- Adc_CfgDefines.h
- Adc_Cfg.c
- Adc_<VariantName>_PBcfg.c

For driver compilation, Adc_<VariantName>_PBcfg.c should be generated by the user using a configuration tool. The file contains the definition of the init pointer for the respective variant.

As a deviation from standard:

- Adc_<VariantName>_PBcfg.c files will contain the definition for all parameters that are variant aware, independent of the configuration class that will be selected (PC, LT, PB)

- Adc_Cfg.c file will contain the definition for all configuration structures containing only variables that are not variant aware, configured and generated only once. This file alone does not contain the whole structure needed by <Mdl>_Init function to configure the driver. Based on the number of variants configured in the EcuC, there can be more than one configuration structure for one module even for PreCompile variant.

Files from Base common folder

- ..\Base_ TS_T40D2M10I1R0 \include\Reg_eSys.h
- ..\Base_ TS_T40D2M10I1R0 \include\Compiler.h
- ..\Base_ TS_T40D2M10I1R0 \include\Compiler_Cfg.h
- ..\Base_ TS_T40D2M10I1R0 \include\ComStack_Cfg.h
- ..\Base_ TS_T40D2M10I1R0 \include\ComStack_Types.h
- ..\Base_ TS_T40D2M10I1R0 \include\RegLockMacros.h
- ..\Base_ TS_T40D2M10I1R0 \include\SilRegMacros.h
- ..\Base_ TS_T40D2M10I1R0 \include\Std_Types.h
- ..\Base_ TS_T40D2M10I1R0 \include\StdRegMacros.h
- ..\Base_ TS_T40D2M10I1R0 \include\Adc_MemMap.h
- ..\Base_ TS_T40D2M10I1R0 \include\Soc_Ips.h
- ..\Base_ TS_T40D2M10I1R0 \include\Mcal.h
- ..\Base_ TS_T40D2M10I1R0 \include\Platform_Types.h

Files from Dem folder:

- ..\Dem_ TS_T40D2M10I1R0 \include\Dem.h
- ..\Dem_ TS_T40D2M10I1R0 \include\Dem_Types.h
- ..\Dem_ TS_T40D2M10I1R0 \src\Dem.c

Files from Det folder:

- ..\Det_ TS_T40D2M10I1R0 \include\Det.h
- ..\Det_ TS_T40D2M10I1R0 \src\Det.c

Files from Mcl folder:

- ..\Mcl_TS_T40D2M10I1R0\include\CDD_Mcl.h
- ..\Mcl_TS_T40D2M10I1R0\include\Mcl.h
- ..\Mcl_TS_T40D2M10I1R0\include\Mcl_Dma.h
- ..\Mcl_TS_T40D2M10I1R0\include\Mcl_Dma_Types.h
- ..\Mcl_TS_T40D2M10I1R0\include\Mcl_DmaMux_Types.h
- ..\Mcl_TS_T40D2M10I1R0\include\Mcl_EnvCfg.h
- ..\Mcl_TS_T40D2M10I1R0\include\Mcl_IPW.h

- ..\Mcl_TS_T40D2M10I1R0\include\Mcl_IPW_Notif.h
- ..\Mcl_TS_T40D2M10I1R0\include\Mcl_IPW_Types.h
- ..\Mcl_TS_T40D2M10I1R0\include\Mcl_Notif.h
- ..\Mcl_TS_T40D2M10I1R0\include\Mcl_TrgMux.h
- ..\Mcl_TS_T40D2M10I1R0\include\Mcl_TrgMux_Types.h
- ..\Mcl_TS_T40D2M10I1R0\include\Mcl_Types.h
- ..\Mcl_TS_T40D2M10I1R0\include\Reg_eSys_Dma.h
- ..\Mcl_TS_T40D2M10I1R0\include\Reg_eSys_DmaMux.h
- ..\Mcl_TS_T40D2M10I1R0\include\Reg_eSys_TrgMux.h
- ..\Mcl_TS_T40D2M10I1R0\src\CDD_Mcl.c
- ..\Mcl_TS_T40D2M10I1R0\src\Mcl_Dma.c
- ..\Mcl_TS_T40D2M10I1R0\src\Mcl_Dma_Irq.c
- ..\Mcl_TS_T40D2M10I1R0\src\Mcl_DmaMux.c
- ..\Mcl_TS_T40D2M10I1R0\src\Mcl_IPW.c
- ..\Mcl_TS_T40D2M10I1R0\generate_PC\src\CDD_Mcl_Cfg.c
- ..\Mcl_TS_T40D2M10I1R0\generate_PB\src\CDD_Mcl_PBcfg_<VariantName>.c
- CDD_Mcl_Cfg.h
- Mcl_DmaMux.h
- CDD_Mcl_Cfg.c
- CDD_Mcl_PBcfg_<VariantName>.c

Files from Rte folder:

- ..\Rte_TS_T40D2M10I1R0\include\SchM_Adc.h
- ..\Rte_TS_T40D2M10I1R0\src\SchM_Adc.c

3.3 Setting up the Plug-ins

The Adc driver was designed to be configured by using the EB Tresos Studio (version EB tresos Studio 23.0.0 b170330-0431 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_T40D2M10I1R0\config\Adc.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k118_lqfp48.epd
  - ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k118_lqfp64.epd
  - ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k142_lqfp64.epd
  - ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k142_lqfp100.epd
  - ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k144_lqfp64.epd

- ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k144_lqfp100.epd
- ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k144_mapbga100.epd
- ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k146_lqfp64.epd
- ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k146_lqfp100.epd
- ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k146_lqfp144.epd
- ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k146_mapbga100.epd
- ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k148_lqfp144.epd
- ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k148_lqfp176.epd
- ..\Adc_TS_T40D2M10I1R0\autosar\Adc_s32k148_mapbga100.epd

- Code Generation Templates for Pre - Compile time configuration parameters:
  - ..\Adc_TS_T40D2M10I1R0 \generate_PC\include\Adc_Cfg.h
  - ..\Adc_TS_T40D2M10I1R0 \generate_PC\include\Adc_CfgDefines.h
  - ..\Adc_TS_T40D2M10I1R0 \generate_PC\src\Adc_Cfg.c

- Code Generation Templates for Post - Build time configuration parameters:
  - ..\Adc_TS_T40D2M10I1R0 \generate_PB\src\Adc_PBcfg_<VariantName>.c

Steps to generate the configuration:

1. Copy the module folders Adc_TS_T40D2M10I1R0, Base_TS_T40D2M10I1R0, Dem_TS_T40D2M10I1R0,Det_TS_T40D2M10I1R0,Rte_TS_T40D2M10I1R0, Resource_TS_T40D2M10I1R0, EcuM_TS_T40D2M10I1R0, Mcl_TS_T40D2M10I1R0, Mcu_TS_T40D2M10I1R0, Ecuc_TS_T40D2M10I1R0 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:

Before starting any ADC conversion, according to the AUTOSAR requirement ADC421, it is mandatory call the function Adc_SetupResultBuffer.

4.2 Function Calls during Shutdown

None.

4.3 Function Calls during Wake-up

None.

Chapter 5
Module requirements

5.1 Exclusive areas to be defined in BSW scheduler

In the current implementation, ADC is using the services of Run-Time Environment (RTE) for entering and exiting the critical regions. RTE implementation is done by the integrators of the MCAL using OS or non- OS services. For testing the ADC, stubs are used for RTE. All ADC notification functions are called outside any critical region. Global variables updates are performed by ISRs before calling the user notification functions. So the ADC internal state is consistent at the moment of the notification call. The ISR critical regions must not block the other critical regions to avoid deadlocks. This is ensured by exiting the ISR critical region before calling the user notification functions. The following critical regions are used in the ADC driver:

5.1.1 ADC_EXCLUSIVE_AREA_00

Used in function Adc_EnableGroupNotification, protects the write to Adc_aGroupStatus[Group].eNotification field. This field is used by other functions.

5.1.2 ADC_EXCLUSIVE_AREA_01

Used in function Adc_DisableGroupNotification, protects the write to Adc_aGroupStatus[Group].eNotification field. This field is used by other functions.

5.1.3 ADC_EXCLUSIVE_AREA_06

Used in function Adc_Adc12bsarv2_DeInit. This function is called from Adc_DeInit ASR API. Protects the following:

- Write to PDB Register
- Write to ADC Register
- Write to Adc_aGroupStatus[GroupIdx].eNotification

No interruptions must occur while these registers are reset.

5.1.4 ADC_EXCLUSIVE_AREA_07

Used in function Adc_Adc12bsarv2_StartGroupConversion. This function is called from Adc_StartGroupConversion ASR API. Protects the following:

- Write to Adc_aGroupStatus[Group].eConversion
- Write to Adc_aGroupStatus[Group].ResultIndex
- Read and write to Adc_aUnitStatus[Unit].SwNormalQueueIndex
- Read and write to Adc_aUnitStatus[Unit].SwNormalQueue
- Write to Adc_aGroupStatus[Group].CurrentChannel
- Write to ADC registers
- Write to PDB registers
- Write to DMA regs(disable DMA interrupt)

No interruptions must occur while region is running because other ADC functions are also working with the queue, PDB registers, DMA registers and ADC registers.

5.1.5 ADC_EXCLUSIVE_AREA_08

Used in function Adc_Adc12bsar_StopGroupConversion. This function is called from Adc_StopGroupConversion ASR API. Protects the following:

- Write to Adc_aGroupStatus[Group].bLimitCheckFailed
- Write to Adc_aGroupStatus[Group].eConversion
- Write to Adc_aGroupStatus[Group].ResultIndex
- Write to Adc_aGroupStatus[Group].eNotification
- Write to Adc_aRuntimeGroupChannel[Group].u32Mask
- Write to Adc_aUnitStatus[Unit].SwInjectedQueueIndex
- Write to ADC registers
- Write to PDB registers
- Write to DMA registers

No interruptions must occur while region is running because other ADC functions are also working with the queue, DMA registers, ADC registers, and PDB registers.

5.1.6 ADC_EXCLUSIVE_AREA_09

Used in function Adc_Adc12bsarv2_ReadGroup. This function is called from Adc_ReadGroup ASR API. Protects the following:

- Read of Adc_aGroupStatus[Group].eConversion
- Read of Adc_aGroupStatus[Group].ResultIndex
- Read of Adc_pCfgPtr->Groups[Group].pResultsBufferPtr[Group]

No interruptions must occur while region is running because other ADC functions are also setting the conversion and result index.

5.1.7 ADC_EXCLUSIVE_AREA_10

Used in function Adc_Adc12bsarv2_EnableHardwareTrigger. This function is called from Adc_EnableHardwareTrigger ASR API. Protects the following:

- Read of Adc_aRuntimeGroupChannel[Group].pChannel
- Read of Adc_aRuntimeGroupChannel[Group].u32Mask
- Read of Adc_aRuntimeGroupChannel[Group].ChannelCount
- Read of Adc_aUnitStatus[Unit].HwInjectedQueueIndex
- Write to Adc_aUnitStatus[Unit].eHwQueueGroupType
- Write to Adc_aUnitStatus[Unit].HwNormalQueue[Adc_aUnitStatus[Unit].HwNormalQueueIndex]
- Write to Adc_aUnitStatus[Unit].HwNormalQueueIndex
- Write to Adc_aUnitStatus[Unit].u8Sc1Used
- Write to Adc_aGroupStatus[Group].eConversion
- Write to Adc_aGroupStatus[Group].eHwTriggering
- Write to Adc_aGroupStatus[Group].ResultIndex
- Write to Adc_aGroupStatus[Group].CurrentChannel
- Write to Adc_Adc12bsarv2_aDmaTcdConfig[Unit]
- Write to Adc_Adc12bsarv2_aTcdAddress[Unit]
- Write to PDB register
- Write to ADC register

No interruptions must occur while region is running because other ADC functions are also setting the fields, PDB registers and ADC registers.

5.1.8 ADC_EXCLUSIVE_AREA_11

Used in function Adc_Adc12bsarv2_DisableHardwareTrigger. This function is called from Adc_DisableHardwareTrigger ASR API. Protects the following:

- Write to Adc_aGroupStatus[Group].eNotification
- Write to Adc_aGroupStatus[Group].eConversion
- Write to Adc_aGroupStatus[Group].eHwTriggering
- Write to Adc_aRuntimeGroupChannel[Group].u32Mask
- Write to Adc_aUnitStatus[Unit].HwNormalQueueIndex
- Write to ADC register
- Write to PDB register

No interruptions must occur while region is running because other ADC functions are also setting the fields, PDB registers and ADC registers.

5.1.9 ADC_EXCLUSIVE_AREA_12

Used in function Adc_GetStreamLastPointer ASR API. Protects the following:

- Read and write to Adc_aGroupStatus[Group].eConversion
- Read and write to Adc_aGroupStatus[Group].ResultIndex

No interruptions must occur while region is running because other ADC functions are also setting the fields.

5.1.10 ADC_EXCLUSIVE_AREA_13

Used in ISR processing function Adc_Adc12bsarv2_DmaEndGroupConversion and covers also the call to Adc_Adc12bsarv2_DmaEndNormalConv, Adc_Adc12bsarv2_DmaEndHardwareConv. It protects the following:

- Write to Adc_aGroupStatus[Group].eAlreadyConverted
- Write to Adc_aGroupStatus[Group].bLimitCheckFailed
- Read and write to Adc_GroupStatus[Group].CurrentChannel
- Write to Adc_aGroupStatus[Group].eConversion
- Read and write to Adc_aGroupStatus[Group].CurrentChannel
- Write to Adc_aRuntimeGroupChannel[Group].bRuntimeUpdated
- Read and write to Adc_aGroupStatus[Group].ResultIndex
- Read of Adc_aUnitStatus[Unit].SwNormalQueue
- Read of Adc_aRuntimeGroupChannel[Group].ChannelCount

- Read of Adc_aRuntimeGroupChannel[Group].u32Mask
- Read of Adc_aRuntimeGroupChannel[Group].pChannel
- Read of Adc_aRuntimeGroupChannel[Group].pu16Delays
- Write to Adc_aUnitStatus[Unit].HwNormalQueueIndex
- Write to Adc_Adc12bsarv2_aDmaTcdConfig[Unit]
- Write to Adc_aUnitStatus[Unit].u8Sc1Used
- Write to ADC register
- Write to PDB register

This region must protect against interruptions by other functions that can set this fields.

5.1.11 ADC_EXCLUSIVE_AREA_14

Used in ISR processing function Adc_Adc12bsarv2_EndPartialConversion, protects the following:

- Write to Adc_aGroupStatus[Group].eAlreadyConverted
- Write to Adc_aGroupStatus[Group].bLimitCheckFailed
- Read and write to Adc_aGroupStatus[Group].CurrentChannel
- Write to Adc_aGroupStatus[Group].eConversion
- Read of Adc_aGroupStatus[Group].eNotification
- Read and write to Adc_aGroupStatus[Group].CurrentChannel
- Read and write to Adc_aRuntimeGroupChannel[Group].bRuntimeUpdated
- Read and write to Adc_aGroupStatus[Group].ResultIndex
- Write to Adc_aUnitStatus[Unit].SwNormalQueueIndex
- Write to Adc_Adc12bsarv2_aDmaTcdConfig[Unit]
- Read of Adc_aRuntimeGroupChannel[Group].u32Mask
- Read of Adc_aRuntimeGroupChannel[Group].pChannel
- Read of Adc_aRuntimeGroupChannel[Group].pu16Delays
- Read of Adc_aRuntimeGroupChannel[Group].ChannelCount
- Write to Adc_aUnitStatus[Unit].u8Sc1Used
- Write to ADC register
- Write to PDB register

This region must protect against interruptions by other functions that can set this fields.

5.1.12 ADC_EXCLUSIVE_AREA_16

Used in function Adc_Adc12bsarv2_Calibrate. This function is called from Adc_Calibrate Non-ASR API. Protects the following:

- Write to pStatus->Adc_UnitSelfTestStatus
- Write to ADC registers

No interruptions must occur while these registers are wrote.

5.1.13 ADC_EXCLUSIVE_AREA_20

Used in function Adc_ValidateNotBusyEnableCtuTrig, Adc_ValidateNotBusyNoQueue.This function is called from Adc_EnableHardwareTrigger ASR API and Adc_EnableCTUTrigger. Protects the following:

- Read of Adc_aUnitStatus[Unit].SwNormalQueueIndex
- Read of Adc_aUnitStatus[Unit].SwInjectedQueueIndex
- Read of Adc_aUnitStatus[Unit].HwInjectedQueueIndex
- Read of Adc_aUnitStatus[Unit].HwNormalQueueIndex

5.1.14 ADC_EXCLUSIVE_AREA_27

Used in the function Non ASR Adc_SetClockMode to protect the call function Adc_Ipw_SetClockMode. The fields of this structure are updated by other ADC functions. Protects the following:

- Write to Adc_eClockMode
- Write to ADC registers

5.1.15 ADC_EXCLUSIVE_AREA_30

Used in function Adc_SetChannel, protects the following:

- Write to Adc_aRuntimeGroupChannel[Group].pChannel
- Write to Adc_aRuntimeGroupChannel[Group].pu16Delays
- Write to Adc_aRuntimeGroupChannel[Group].u32Mask
- Write to Adc_aRuntimeGroupChannel[Group].ChannelCount
- Write to Adc_aRuntimeGroupChannel[Group].bRuntimeUpdated

No interruptions must occur while region is running because other ADC functions are also setting the fields.

Critical Region Exclusive Matrix

Below is the table depicting the exclusivity between different critical region IDs from the Adc driver. If there is an “X” in a table, it means that those 2 critical regions cannot interrupt each other.

The critical regions from interrupts are grouped in “Interrupt Service Routines Critical Regions (composed diagram)”. If an exclusive area is “exclusive” with the composed “Interrupt Service Routines Critical Regions (composed diagram)” group, it means that it is exclusive with each one of the ISR critical regions.

Exclusive Area Matrix

Exclusive Area ID	N/A	ADC_EXCLUSIVE_AREA_06 ADC_EXCLUSIVE_AREA_20	N/A	ADC_EXCLUSIVE_AREA_07	ADC_EXCLUSIVE_AREA_08	ADC_EXCLUSIVE_AREA_09	ADC_EXCLUSIVE_AREA_10 ADC_EXCLUSIVE_AREA_20	ADC_EXCLUSIVE_AREA_11	ADC_EXCLUSIVE_AREA_00	ADC_EXCLUSIVE_AREA_01	ADC_EXCLUSIVE_AREA_12	N/A	ADC_EXCLUSIVE_AREA_20 ADC_EXCLUSIVE_AREA_27	ADC_EXCLUSIVE_AREA_16 ADC_EXCLUSIVE_AREA_20	ADC_EXCLUSIVE_AREA_30	ADC_EXCLUSIVE_AREA_13	ADC_EXCLUSIVE_AREA_14	N/A	
ADC_EXCLUSIVE_AREA_00		x			x			x		x							x		
ADC_EXCLUSIVE_AREA_01		x			x			x	x								x		
ADC_EXCLUSIVE_AREA_06				x	x		x	x	x	x			x			x	x	x	
ADC_EXCLUSIVE_AREA_07		x			x	x	x	x			x		x	x		x	x	x	
ADC_EXCLUSIVE_AREA_08		x		x		x	x	x	x	x	x		x	x	x	x	x	x	
ADC_EXCLUSIVE_AREA_09		x		x	x		x	x			x					x	x		
ADC_EXCLUSIVE_AREA_10		x		x	x	x		x			x		x	x	x	x	x	x	
ADC_EXCLUSIVE_AREA_11		x		x	x	x	x		x	x	x		x	x	x	x	x	x	
ADC_EXCLUSIVE_AREA_12				x	x	x	x	x								x	x		
ADC_EXCLUSIVE_AREA_13		x		x	x	x	x	x			x		x	x	x		x	x	
ADC_EXCLUSIVE_AREA_14		x		x	x	x	x	x	x	x	x		x	x	x	x		x	
ADC_EXCLUSIVE_AREA_16		x		x	x		x	x					x			x	x		
ADC_EXCLUSIVE_AREA_20				x	x	x		x								x	x		
ADC_EXCLUSIVE_AREA_27		x		x	x		x	x						x		x	x		
ADC_EXCLUSIVE_AREA_30					x		x	x								x	x		

Figure 5-1. Exclusive areas.

5.2 Peripheral Hardware Requirements

This device contains two 12-bit SAR ADC modules.The number of channels are derivative specific, so please consult the derivative manuals.

5.3 ISR to configure within OS – dependencies

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

Table 5-1. ADC ISR

ISR Name	HW INT Vector	Observations
ISR(Adc_Adc12bsarv2_EndGroupConv Unit0)	39	The function implements the ISR for the ADC Hardware Unit 0.
ISR(Adc_Adc12bsarv2_EndGroupConv Unit1)	40	The function implements the ISR for the ADC Hardware Unit 1.
ISR(Adc_Pdb_ChannelSequenceError0)	52	The function implements the ISR for the sequence error on PDB 0.
ISR(Adc_Pdb_ChannelSequenceError1)	68	The function implements the ISR for the sequence error on PDB 1.
Adc_Adc12bsarv2_DmaTransferComplete0()	--	The function is a notification called by MCL module after the transferis completed for the ADC Hardware Unit 0.
Adc_Adc12bsarv2_DmaTransferComplete1()	--	The function is a notification called by MCL module after the transferis completed for the ADC Hardware Unit 1.

5.4 ISR Macro

MCAL drivers use the ISR macro to define the functions that will process hardware interrupts. Depending on whether the OS is used or not, this macro can have different definitions:

a. OS is not used - AUTOSAR_OS_NOT_USED is defined:

i. If USE_SW_VECTOR_MODE is defined:

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

In this case, drivers' interrupt handlers are normal C functions and the prolog/epilog handle the context save and restore.

ii. If USE_SW_VECTOR_MODE is not defined:

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

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

Custom OS is used - AUTOSAR_OS_NOT_USED is not defined

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

In this case, OS is handling the execution context when an interrupt occurs. Drivers' interrupt handlers are normal C functions.

Other vendor's OS is used - AUTOSAR_OS_NOT_USED is not defined. Please refer to the OS documentation for description of the ISR macro.

5.5 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:** Mcl is used to configure the hardware trigger for ADC. Mcl module should be initialized to ensure that hardware trigger can work. In DMA mode, the Mcl is used to configure the DMA channel allocated for all ADC HW units. Mcl module should be initialized before starting any ADC conversion request.
- **Base:** The Base module contains the common files/definitions needed by all MCAL modules.
- **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.
- **Dem:** The ADC module shall report production errors to the Diagnostic Event Manager (DEM).
- **Port:** The PORT module shall configure the port pins used by the ADC module. Both analogue input pins and external trigger pins have to be considered.
- **Resource:** Sub-Derivative model is selected from Resource configuration. Support for the following derivatives, everyone having attached a Resource file: s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64 .
- **RTE:** Used to manage the exclusive area inside Adc module.
- **EcuC:** This module is required for configuring the variant handling in Tresos.

5.6 Data Cache Restriction

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 Memmap).

5.7 User Mode support

No special measures need to be taken to run **ADC** module from user mode. The Adc driver code can be executed at any time from both supervisor and user mode.

Chapter 6
Main API Requirements

6.1 Main functions calls within BSW scheduler

None.

6.2 API Requirements

None.

6.3 Calls to Notification Functions, Callbacks, Callouts

Call-back Notifications:

User Notification:

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

The syntax of this function is as follows:

```
void Adc_Notification_<group>()
```

AdcExtraNotification:

Extra callback function for each group. This function pointer will be called at the beginning of the interrupt routine, before updating any HW registers or Group status. The notifications can be configured as pointers to user defined functions. If notification is not desired, 'NULL_PTR' shall be configured.

The syntax of this function is as follows:

```
void Adc_ExtraNotification_<group>()
```

AdcPdbErrorNotification:

Callback function for PDB channel sequence error. This function pointer is called everytime when PDB channel sequence error occurred. The notifications can be configured as pointers to user defined functions. If notification is not desired, 'NULL_PTR' shall be configured.

The syntax of this function is as follows:

```
void Adc_PdbErrorNotification_<group>()
```

The extern declarations of those function are available in Adc_PBcfg.c/Adc_Cfg.c. The function has to be implemented by the user.

Chapter 7
Memory Allocation

7.1 Sections to be defined in MemMap.h

Section name	Type of section	Description
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_VAR_NO_INIT_8	Variables	Used for variables which have to be aligned to 8 bit. For instance used for variables of size 8 bit or used for composite data types: arrays, structs containing elements of maximum 8 bits. These variables are never cleared and never initialized by start-up code.
ADC_STOP_SEC_VAR_NO_INIT_8	Variables	End of above section.
ADC_START_SEC_VAR_NO_INIT_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 never cleared and never initialized by start-up code.
ADC_STOP_SEC_VAR_NO_INIT_UNSPECIFIED	Variables	End of above section.
ADC_START_SEC_VAR_INIT_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 initialized with values after every reset.
ADC_STOP_SEC_VAR_INIT_UNSPECIFIED	Variables	End of above section.
ADC_START_SEC_CONST_32	Constant Data	Used for constants that have to be aligned to 32 bit.
ADC_STOP_SEC_CONST_32	Constant Data	End of above section.
ADC_START_SEC_CONST_UNSPECIFIED	Constant Data	Used for constants, does not fit the criteria of 8,16 or 32 bit.

Table continues on the next page...

more:ADC_STOP_SEC_CONST_UNSPECIFIED	Constant Data	End of above section.
ADC_START_SEC_CONFIG_DATA_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

7.2 Linker command file

Memory shall be allocated for every section defined in Adc_MemMap.h

Chapter 8
Configuration parameters considerations

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

8.1 Configuration Parameters

Specifies whether the configuration parameter shall be of configuration class Post Build.

Table 8-1. Configuration Parameters

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
Adc	IMPLEMENTATION_CONFIG_VARIANT	Pre Compile parameter for all Variants of Configuration	Pre Compile
AdcConfigSet/AdcHwUnit	AdcTransferType	VariantPC or VariantPB	Post Build
	AdcClockSource	VariantPC or VariantPB	Post Build
	AdcHwUnitId	VariantPC or VariantPB	Post Build
	AdcLogicalUnitId	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcVoltageReferenceSelection	VariantPC or VariantPB	Post Build
	AdcPrescale	VariantPC or VariantPB	Post Build
	AdcResolution	VariantPC or VariantPB	Post Build
	AdcOffsetCorrectionValue	VariantPC or VariantPB	Post Build
AdcConfigSet/AdcHwUnit/AdcPdbSettings	AdcPdbPrescalerDividerSelect	VariantPC or VariantPB	Post Build
	AdcPdbMultiplicationFactorSelect	VariantPC or VariantPB	Post Build
	AdcPdbChannelSequenceErrorEnable	VariantPC or VariantPB	Post Build
	AdcPdbErrorNotification	VariantPC or VariantPB	Post Build
AdcConfigSet/AdcHwUnit/AdcNormalConversionTimings	AdcHardwareAverageEnable	VariantPC or VariantPB	Post Build
	AdcHardwareAverageSelect	VariantPC or VariantPB	Post Build
	AdcSampleTimeDuration	VariantPC or VariantPB	Post Build

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	AdcClockDivideSelect	VariantPC or VariantPB	Post Build
AdcConfigSet/AdcHwUnit/ AdcAlternateConversionTimings	AdcHardwareAverageEnableAlternate	VariantPC or VariantPB	Post Build
	AdcHardwareAverageSelectAlternate	VariantPC or VariantPB	Post Build
	AdcSampleTimeDurationAlternate	VariantPC or VariantPB	Post Build
	AdcClockDivideSelectAlternate	VariantPC or VariantPB	Post Build
AdcConfigSet/AdcHwUnit/ AdcChannel	AdcLogicalChannelId	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcChannelId	VariantPC or VariantPB	Post Build
	AdcChannelLimitCheck	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcChannelHighLimit	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcChannelLowLimit	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcChannelRangeSelect	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcChannelConvTime	VariantPC or VariantPB	Post Build
	AdcChannelRefVoltsrcHigh	VariantPC or VariantPB	Post Build
	AdcChannelRefVoltsrcLow	VariantPC or VariantPB	Post Build
	AdcChannelResolution	VariantPC or VariantPB	Post Build
	AdcChannelSampTime	VariantPC or VariantPB	Post Build
AdcConfigSet/AdcHwUnit/ AdcGroup	AdcGroupAccessMode	VariantPC or VariantPB	Post Build
	AdcGroupConversionMode	VariantPC or VariantPB	Post Build
	AdcGroupConversionType	VariantPC or VariantPB	Post Build
	AdcGroupId	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcGroupPriority	VariantPC or VariantPB	Post Build
	AdcGroupReplacement	VariantPC or VariantPB	Post Build
	AdcGroupTriggSrc	VariantPC or VariantPB	Post Build
	AdcHwTrigSignal	VariantPC or VariantPB	Post Build
	AdcHwTrigTimer	VariantPC or VariantPB	Post Build
	AdcNotification	VariantPC or VariantPB	Post Build
	AdcExtraNotification	VariantPC or VariantPB	Post Build
	AdcStreamingBufferMode	VariantPC or VariantPB	Post Build
	AdcEnableDoubleBuffering	VariantPC or VariantPB	Post Build
	AdcStreamingNumSamples	VariantPC or VariantPB	Post Build
	AdcWithoutInterrupts	VariantPC or VariantPB	Post Build

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	AdcGroupInBacktoBackMode	VariantPC or VariantPB	Post Build
	AdcGroupUsesChannelDelays	VariantPC or VariantPB	Post Build
	AdcDelayNextPdb	VariantPC or VariantPB	Post Build
	AdcPdbPeriodContinuousMode	VariantPC or VariantPB	Post Build
AdcConfigSet/AdcHwUnit/AdcGroupAdcGroupDefinition	AdcGroupDefinition	VariantPC or VariantPB	Post Build
AdcConfigSet/AdcHwUnit/AdcGroupAdcChannelDelay	AdcChannelDelay	VariantPC or VariantPB	Post Build
AdcConfigSet/AdcHwUnit/AdcGroup/AdcGroupNormalConversionTimings	AdcGroupHardwareAverageEnable	VariantPC or VariantPB	Post Build
	AdcGroupHardwareAverageSelect	VariantPC or VariantPB	Post Build
	AdcGroupSampleTimeDuration	VariantPC or VariantPB	Post Build
	AdcGroupClockDivideSelect	VariantPC or VariantPB	Post Build
AdcConfigSet/AdcHwUnit/AdcGroup/AdcGroupAlternateConversionTimings	AdcGroupAltHardwareAverageEnable	VariantPC or VariantPB	Post Build
	AdcGroupAltHardwareAverageSelect	VariantPC or VariantPB	Post Build
	AdcGroupAltSampleTimeDuration	VariantPC or VariantPB	Post Build
	AdcGroupAltClockDivideSelect	VariantPC or VariantPB	Post Build
AdcGeneral	AdcDeInitApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcDevErrorDetect	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableLimitCheck	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableQueuing	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableStartStopGroupApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcGrpNotifCapability	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcHwTriggerApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcReadGroupApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcVersionInfoApi	Pre Compile parameter for all Variants of Configuration	Pre Compile

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	AdcPriorityImplementation	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcResultAlignment	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcTimeout	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcDmaTimeout	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcPriorityQueueMaxDepth	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcLowPowerStatesSupport	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcPowerStateAsynchTransitionMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
AdcPowerStateConfig	AdcPowerState	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcPowerStateReadyCbkRef	Pre Compile parameter for all Variants of Configuration	Pre Compile
AdcInterrupt	AdcInterruptSource	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcInterruptEnable	Pre Compile parameter for all Variants of Configuration	Pre Compile
AdcPublishedInformation	AdcChannelValueSigned	VariantPC or VariantPB	Post Build
	AdcGroupFirstChannelFixed	VariantPC or VariantPB	Post Build
	AdcMaxChannelResolution	VariantPC or VariantPB	Post Build
CommonPublishedInformation	ArReleaseMajorVersion	VariantPC or VariantPB	Post Build
	ArReleaseMinorVersion	VariantPC or VariantPB	Post Build
	ArReleaseRevisionVersion	VariantPC or VariantPB	Post Build
	ModuleId	VariantPC or VariantPB	Post Build
	SwMajorVersion	VariantPC or VariantPB	Post Build
	SwMinorVersion	VariantPC or VariantPB	Post Build
	SwPatchVersion	VariantPC or VariantPB	Post Build
	VendorApiInfix	VariantPC or VariantPB	Post Build
	VendorId	VariantPC or VariantPB	Post Build
NonAutosar	AdcEnableGroupDependentChannelNames	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableDualClockMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableCalibration	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableSetChannel	Pre Compile parameter for all Variants of Configuration	Pre Compile

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	AdcEnableInitialNotification	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcDisableDemReportErrorStatus	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcConvTimeOnce	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcOptimizeOneShotHwTriggerConversions	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableDoubleBufferingOptimization	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableDmaTrasferMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcUseHardwareNormalGroups	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableUserModeSupport	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcBypassConsistencyLoop	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcContinuousWithoutInterrupt	Pre Compile parameter for all Variants of Configuration	Pre Compile
AdcDemEventParameterRefs	ADC_E_TIMEOUT	Pre Compile parameter for all Variants of Configuration	Pre Compile

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating Analog to Digital Converter :

- Generate the required Adc configurations. For more details refer to section Files required for Compilation
- Allocate proper memory sections in Adc_MemMap.h and linker command file. For more details refer to section Sections to be defined in MemMap.h
- Compile & build the Adc with all the dependent modules. For more details refer to section Building the Driver

Chapter 10
ISR Reference

ISR functions exported by the Adc driver.

10.1 Software specification

The following sections contains driver software specifications.

10.1.1 Define Reference

Constants supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.2 Rev0002 .

10.1.2 Enum Reference

Enumeration of all constants supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.2 Rev0002 .

10.1.3 Function Reference

Functions of all functions supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.2 Rev0002 .

10.1.3.1 Function Adc_Adc12bsarv2_EndGroupConvUnit0

This function implements the ISR for the conversion done of the ADC Hardware unit 0.

Details:

The function implements the ISR for the ADC Hardware unit 0.

Return: void.

Prototype: `void Adc_Adc12bsarv2_EndGroupConvUnit0(void);`

10.1.3.2 Function Adc_Adc12bsarv2_EndGroupConvUnit1

This function implements the ISR for the conversion done of the ADC Hardware unit 1.

Details:

The function implements the ISR for the ADC Hardware unit 1.

Return: void.

Prototype: `void Adc_Adc12bsarv2_EndGroupConvUnit1(void);`

10.1.3.3 Function Adc_Pdb_ChannelSequenceError0

The function implements the ISR for the sequence error on PDB 0.

Details:

The function implements the ISR for the sequence error on PDB 0.

Return: void.

Prototype: `void Adc_Pdb_ChannelSequenceError0(void);`

10.1.3.4 Function Adc_Pdb_ChannelSequenceError1

The function implements the ISR for the sequence error on PDB 1.

Details:

The function implements the ISR for the sequence error on PDB 1.

Return: void.

Prototype: `void Adc_Pdb_ChannelSequenceError1(void);`

10.1.3.5 Function Adc_Adc12bsarv2_DmaTransferComplete0

This function implements the ISR for the conversion done of the ADC Hardware unit 1.

Details:

The function is a notification called by MCL module after the transfer is completed for the ADC Hardware Unit 0.

Return: void.

Prototype: `void Adc_Adc12bsarv2_DmaTransferComplete0(void);`

10.1.3.6 Function Adc_Adc12bsarv2_DmaTransferComplete1

This function implements the ISR for the conversion done of the ADC Hardware unit 1.

Details:

The function is a notification called by MCL module after the transfer is completed for the ADC Hardware Unit 1.

Return: void.

Prototype: `void Adc_Adc12bsarv2_DmaTransferComplete1(void);`

10.1.4 Structs Reference

Data structures supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.2 Rev0002 .

10.1.5 Types Reference

Types supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.2 Rev0002 .

10.1.6 Variables Reference

Variables supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.2 Rev0002 .

Chapter 11
External Assumptions for ADC driver

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

[SMCAL_CPR_EXT162]

<< If DMA transfer mode is used, the user must not run SW and HW groups at the same time on the same HW unit >>

[SMCAL_CPR_EXT163]

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

[SMCAL_CPR_EXT176]

<< The integrator shall assure that <MSN>_Init() and <MSN>_DeInit() functions do not interrupt each other. >>

[SMCAL_CPR_EXT177]

<< When caches are enabled and data buffers are allocated in cachable 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 to compete for the same cache lines.

[SMCAL_CPR_EXT178]

<< Before calling the Adc_SetMode() API, the user shall ensure that no conversion is ongoing >>

[SMCAL_CPR_EXT179]

<< Before calling the Adc_SetClockMode() API, the user shall ensure that no conversion is ongoing >>

[SMCAL_CPR_EXT180]

<< Before calling the Adc_Calibrate() API, the user shall ensure that no conversion is ongoing >>

[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_ReadGroup will return E_NOT_OK

[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 ADC414 is a safety integrity assumption for external environment, which shall be implemented for FTE; For GTE and NTE ADC414 has a role to increase availablity because the check will be supported by ADC driver;

[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. >>

[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. >>

[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. >>

[SWS_Adc_00249]

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

[SWS_Adc_00250]

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

[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). >>

[SWS_Adc_00358]

<< The ADC module's environment shall not call the function Adc_DeInit while any group is not in state ADC_IDLE. >>

[SWS_Adc_00146]

<< The ADC module's environment shall only call Adc_StartGroupConversion for groups configured with software trigger source. >>

[SWS_Adc_00283]

<< The ADC module's environment shall only call the function Adc_StopGroupConversion for groups configured with trigger source software. >>

[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). >>

[SWS_Adc_00120]

<< The ADC module's environment shall only call the function Adc_EnableHardwareTrigger for groups configured in hardware trigger mode (see AdcGroupTriggSrc). >>

[SWS_Adc_00121]

<< The ADC module's environment shall only call the function Adc_DisableHardwareTrigger for groups configured in hardware trigger mode (see AdcGroupTriggSrc). >>

[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. >>

[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. >>

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.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Document Number IM2ADCASR4.2 Rev0002R1.0.1
Revision 1.0

AUTOSAR