Integration Manual

for MPC574XG ADC Driver

Document Number: IM35ADCASR4.2 Rev002R1.0.0
Rev. 1.0

Contents

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

Section number	Title	Page

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	17-Feb-2017	Nguyen Huy Cuong	First version for Calypso 4.2 RTM 1.0.0.

Chapter 2 Introduction

This integration manual describes the integration requirements for Adc Driver for MPC574XG microcontrollers.

2.1 Supported Derivatives

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

Table 2-1. MPC574XG Derivatives

NXP Semiconductor	MPC5748G_LQFP176, MPC5748G_MAPBGA256, MPC5748G_MAPBGA324, MPC5747G_LQFP176, MPC5747G_MAPBGA256, MPC5747G_MAPBGA324, MPC5746G_LQFP176, MPC5746G_MAPBGA256, MPC5746G_MAPBGA324, MPC5748C_LQFP176, MPC5748C_MAPBGA256, MPC5748C_MAPBGA324, MPC5747C_LQFP176, MPC5747C_MAPBGA256, MPC5747C_MAPBGA324, MPC5746C_LQFP176, MPC5746C_MAPBGA256, MPC5746C_MAPBGA324, MPC5746C_MAPBGA100, MPC5745C_LQFP176, MPC5745C_MAPBGA256, MPC5745C_MAPBGA100, MPC5744C_LQFP176, MPC5744C_MAPBGA256, MPC5744C_MAPBGA100, MPC5746B_LQFP176, MPC5746B_MAPBGA256, MPC5746B_MAPBGA100, MPC5744B_LQFP176, MPC5744B_MAPBGA256,

Table 2-1. MPC574XG Derivatives

	MPC5744B_MAPBGA100, MPC5745B_LQFP176, MPC5745B_MAPBGA256, MPC5745B_MAPBGA100

All of the above microcontroller devices are collectively named as MPC574XG .

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
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	AUTOSAR 4.2 Rev0002Adc Driver Software Specification Document.	R4.2 Rev 002
2	MPC5748G Reference Manual	Rev. 5, 12/2016
3	MPC5746C Reference Manual	Rev. 4, 12/2016
4	MPC5748G_1N81M_Rev.2 (official document) (1N81M)	Jun-16

Table continues on the next page...

Table 2-3. Reference List (continued)

#	Title	Version
5	MPC5748G_1N81M_0N78S_Comparison_Summary_v2_0 (internal document) (1N81M, 0N78S)	31.10.2016
6	MPC5746C_1N06M_Rev.4 (official document) (1N06M)	Jul-16
7	MPC5746C_cut1.1_cut2.0_cut2.1_comparison_v0 (internal document) (1N06M, 0N84S, 1N84S)	14-Sep-16
8	C3M_cut2.1_new_errata_20170113 (internal document) (1N84S)	13-Jan-17

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 SemiconductorMPC574XG . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The Adc driver files are compiled using

- Windriver DIAB DIAB_5_9_6_2
- Green Hills Multi 7.1.4 / Compiler 2015.1.6

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

Note

The TS_T2D35M10I0R0 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 = 2 identifies PA architecture and Derivative_Id = 35 identifies the MPC574XG)

3.1.1 DIAB Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-tPPCE200Z4204N3VEN:simple	Sets target processor to PPCE200Z4204N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-tPPCE200Z210N3VEN:simple	Sets target processor to PPCE200Z210N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-Xdialect-ansi	Follow the ANSI C standard with some additions
-XO	Enables extra optimizations to produce highly optimized code
-g3	Generate symbolic debugger information and do all optimizations.
-Xsize-opt	Optimize for size rather than speed when there is a choice
-Xsmall-data=0	Set Size Limit for 'small data' Variables to zero.
-Xsmall-const=0	Set Size Limit for “small const” Variables to zero.
-Xaddr-sconst=0x11	Specify addressing for constant static and global variables with size less than or equal to -Xsmall-const to far-absolute.
-Xaddr-sdata=0x11	Specify addressing for non-constant static and global variables with size less than or equal to -Xsmall-data in size to far-absolute.
-Xno-common	Disable use of the 'COMMON' feature so that the compiler or assembler will allocate each uninitialized public variable in the .bss section for the module defining it, and the linker will require exactly one definition of each public variable
-Xnested-interrupts	Allow nested interrupts
-Xdebug-dwarf2	Generate symbolic debug information in dwarf2 format
-Xdebug-local-all	Force generation of type information for all local variables
-Xdebug-local-cie	Create common information entry per module
-Xdebug-struct-all	Force generation of type information for all typedefs, struct, union and class types
-Xforce-declarations	Generates warnings if a function is used without a previous declaration
-ee1481	Generate an error when the function was used before it has been declared
-Xmacro-undefined-warn	Generates a warning when an undefined macro name occurs in a #if preprocessor directive
-Xlink-time-lint	Enable the checking of object and function declarations across compilation units, as well as the consistency of compiler options used to compile source files
-W:as:,-l	Pass the option '-l' (lower case letter L) to the assembler to get an assembler listing file
-Wa,-Xisa-vle	Instruct the assembler to expect and assemble VLE (Variable Length Encoding) instructions rather than BookE instructions.
-DAUTOSAR_OS_NOT_USED	-D defines a preprocessor symbol and optionally can set it to a value. AUTOSAR_OS_NOT_USED: By default in the package, the drivers are compiled to be used without Autosar OS. If the drivers are used with Autosar OS, the compiler option '-DAUTOSAR_OS_NOT_USED' must be removed from project options
-DUSE_SW_VECTOR_MODE	-D defines a preprocessor symbol and optionally can set it to a value. USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode.

Table continues on the next page...

Table 3-1. Compiler Options (continued)

Option	Description
-DDIAB	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the DIAB preprocessor symbol.
-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.
-c	Stop after assembly, produce object file.

Table 3-2. Assembler Options

Option	Description
-tPPCE200Z4204N3VEN:simple	Sets target processor to PPCE200Z4204N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-tPPCE200Z210N3VEN:simple	Sets target processor to PPCE200Z210N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-g	Dump the symbols in the global symbol table in each archive file.
-Xisa-vle	Expect and assemble VLE (Variable Length Encoding) instructions rather than Book E instructions. The default code section is named .text_vle instead of .text, and the default code section fill "character" is set to 0x44444444 instead of 0. The .text_vle code section will have ELF section header flags marking it as VLE code, not Book E code.
-Xasm-debug-on	Generate debug line and file information
-Xdebug-dwarf2	Generate symbolic debug information in dwarf2 format
-Xsemi-is-newline	Treat the semicolon (;) as a statement separator instead of a comment character.

Table 3-3. Linker Options

Option	Description
-tPPCE200Z4204N3VEN:simple	Sets target processor to tPPCE200Z4204N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-tPPCE200Z210N3VEN:simple	Sets target processor to tPPCE200Z210N3VEN, generates ELF using EABI conventions, No floating point support (minimizes the required runtime), selects simple environment settings for Startup Module and Libraries
-Xelf	Generates ELF object format for output file
-m6	Generates a detailed link map and cross reference table
-Xlink-time-lint	Enable the checking of object and function declarations across compilation units, as well as the consistency of compiler options used to compile source files

3.1.2 GHS Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
-cpu=ppc5748gz4204	Selects target processor: ppc5748gz4204
-cpu=ppc5748gz210	Selects target processor: ppc5748gz210
-ansi	Specifies ANSI C with extensions. This mode extends the ANSI X3.159-1989 standard with certain useful and compatible constructs.
-noSPE	Disables the use of SPE and vector floating point instructions by the compiler.
-Ospace	Optimize for size.
-sda=0	Enables the Small Data Area optimization with a threshold of 0.
-vle	Enables VLE code generation
-dual_debug	Enables the generation of DWARF, COFF, or BSD debugging information in the object file
-G	Generates source level debugging information and allows procedure call from debugger's command line.
--no_exceptions	Disables support for exception handling
-Wundef	Generates warnings for undefined symbols in preprocessor expressions
-Wimplicit-int	Issues a warning if the return type of a function is not declared before it is called
-Wshadow	Issues a warning if the declaration of a local variable shadows the declaration of a variable of the same name declared at the global scope, or at an outer scope
-Wtrigraphs	Issues a warning for any use of trigraphs
--prototype_errors	Generates errors when functions referenced or called have no prototype
--incorrect_pragma_warnings	Valid #pragma directives with wrong syntax are treated as warnings
-noslashcomment	C++ like comments will generate a compilation error
-preprocess_assembly_files	Preprocesses assembly files
-nostartfile	Do not use Start files
--short_enum	Store enumerations in the smallest possible type
--diag_error 223	Sets the specified compiler diagnostic messages to the level of error
-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
-DUSE_SW_VECTOR_MODE	-D defines a preprocessor symbol and optionally can set it to a value. USE_SW_VECTOR_MODE: By default in the package, drivers are compiled to be used with interrupt controller configured to be in hardware vector mode. In case of AUTOSAR_OS_NOT_USED, the compiler option "-DUSE_SW_VECTOR_MODE" must be added to the list of compiler options to be used with interrupt controller configured to be in software vector mode.
-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.
-c	Produces an object file (called input-file.o) for each source file.

Table 3-5. Assembler Options

Option	Description
-cpu=ppc5748gz4204	Selects target processor: ppc5748gz4204
-cpu=ppc5748gz210	Selects target processor: ppc5748gz210
-G	Generates source level debugging information and allows procedure call from debugger's command line.
-list	Creates a listing by using the name of the object file with the .lst extension

Table 3-6. Linker Options

Option	Description
-cpu=ppc5748gz4204	Selects target processor: ppc5748gz4204
-cpu=ppc5748gz210	Selects target processor: ppc5748gz210
-nostartfiles	Do not use Start files.
-vle	Enables VLE code generation
--nocpp	Do not Generate Constructors/Destructors
-Mn	sort numerically the MAP file
-delete	The -delete option instructs the linker to remove functions that are not referenced in the final executable.
-ignore_debug_references	Ignores relocations from DWARF debug sections when using -delete. DWARF debug information will contain references to deleted functions that may break some third-party debuggers.
-keepmap	keeps the MAP file in case of link error

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 MPC574XG 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_T2D35M10I0R0\include\Adc.h
- ..\Adc_TS_T2D35M10I0R0\include\Adc_Adcdig.h
- ..\Adc_TS_T2D35M10I0R0\include\Adc_Adcdig_CfgEx.h
- ..\Adc_TS_T2D35M10I0R0\include\Adc_Adcdig_Irq.h
- ..\Adc_TS_T2D35M10I0R0\include\Adc_CfgDefine.h
- ..\Adc_TS_T2D35M10I0R0\include\Adc_Bctu.h
- ..\Adc_TS_T2D35M10I0R0\include\Adc_EnvCfg.h

- ..\Adc_TS_T2D35M10I0R0\include\Adc_Ipw.h
- ..\Adc_TS_T2D35M10I0R0\include\Adc_Types.h
- ..\Adc_TS_T2D35M10I0R0\include\Adc_Reg_eSys_Adcdig.h
- ..\Adc_TS_T2D35M10I0R0\include\Adc_Reg_eSys_Bctu.h
- ..\Adc_TS_T2D35M10I0R0\src\Adc.c
- ..\Adc_TS_T2D35M10I0R0\src\Adc_Adcdig.c
- ..\Adc_TS_T2D35M10I0R0\src\Adc_Adcdig_Irq.c
- ..\Adc_TS_T2D35M10I0R0\src\Adc_Bctu.c
- ..\Adc_TS_T2D35M10I0R0\src\Adc_Bctu_Irq.c
- ..\Adc_TS_T2D35M10I0R0\src\Adc_Ipw.c
- ..\Adc_TS_T2D35M10I0R0\src\Adc_Reg_eSys_Adcdig.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 parameters that are not variant aware

Files from Base common folder

- ..\Base_TS_T2D35M10I0R0\include\Adc_MemMap.h
- ..\Base_TS_T2D35M10I0R0\include\Compiler.h
- ..\Base_TS_T2D35M10I0R0\include\Compiler_Cfg.h
- ..\Base_TS_T2D35M10I0R0\include\ComStack_Cfg.h
- ..\Base_TS_T2D35M10I0R0\include\CompilerDefinition.h
- ..\Base_TS_T2D35M10I0R0\include\ComStack_Types.h
- ..\Base_TS_T2D35M10I0R0\include\Mcal.h
- ..\Base_TS_T2D35M10I0R0\include\MemMap.h
- ..\Base_TS_T2D35M10I0R0\include\Platform_Types.h
- ..\Base_TS_T2D35M10I0R0\include\Reg_eSys.h
- ..\Base_TS_T2D35M10I0R0\include\RegLogMacros.h
- ..\Base_TS_T2D35M10I0R0\include\SilRegMacros.h
- ..\Base_TS_T2D35M10I0R0\include\Soc_Ips.h

- ..\Base_TS_T2D35M10I0R0\include\Std_Types.h
- ..\Base_TS_T2D35M10I0R0\include\StdRegMacros.h

Files from Dem folder:

- ..\Dem_TS_T2D35M10I0R0\include\Dem.h
- ..\Dem_TS_T2D35M10I0R0\include\Dem_IntErrId.h
- ..\Dem_TS_T2D35M10I0R0\include\Dem_Types.h
- ..\Dem_TS_T2D35M10I0R0\src\Dem.c

Files from Det folder:

- ..\Det_TS_T2D35M10I0R0\include\Det.h
- ..\Det_TS_T2D35M10I0R0\src\Det.c

Files from Mcl folder:

- ..\Mcl_TS_T2D35M10I0R0\include\CDD_Mcl.h
- ..\Mcl_TS_T2D35M10I0R0\include\Mcl.h
- ..\Mcl_TS_T2D35M10I0R0\include\Mcl_Dma.h
- ..\Mcl_TS_T2D35M10I0R0\include\Mcl_Dma_Irq.h
- ..\Mcl_TS_T2D35M10I0R0\include\Mcl_Dma_Types.h
- ..\Mcl_TS_T2D35M10I0R0\include\Mcl_DmaMux.h
- ..\Mcl_TS_T2D35M10I0R0\include\Mcl_DmaMux_Types.h
- ..\Mcl_TS_T2D35M10I0R0\include\Mcl_EnvCfg.h
- ..\Mcl_TS_T2D35M10I0R0\include\Mcl_Ipw.h
- ..\Mcl_TS_T2D35M10I0R0\include\Mcl_Ipw_Notif.h
- ..\Mcl_TS_T2D35M10I0R0\include\Mcl_Ipw_Types.h
- ..\Mcl_TS_T2D35M10I0R0\include\Mcl_Notif.h
- ..\Mcl_TS_T2D35M10I0R0\include\Mcl_Types.h
- ..\Mcl_TS_T2D35M10I0R0\include\Reg_eSys_Dma.h
- ..\Mcl_TS_T2D35M10I0R0\include\Reg_eSys_DmaMux.h
- ..\Mcl_TS_T2D35M10I0R0\src\CDD_Mcl.c
- ..\Mcl_TS_T2D35M10I0R0\src\Mcl_Dma.c
- ..\Mcl_TS_T2D35M10I0R0\src\Mcl_Dma_Irq.c
- ..\Mcl_TS_T2D35M10I0R0\src\Mcl_DmaMux.c
- ..\Mcl_TS_T2D35M10I0R0\src\Mcl_Ipw.c
- CDD_Mcl_Cfg.h
- CDD_Mcl_Cfg.c
- CDD_Mcl_<VariantName>_PBcfg.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 21.0.0 b160607-0933 or later.)

Location of various files inside the FR module folder:

- VSMD (Vendor Specific Module Definition) file in EB tresos Studio XDM format:
  - ..\Adc_TS_T2D35M10I0R0\config\Adc.xdm
  - ..\Mcu_TS_T2D35M10I0R0\config\Mcu.xdm
  - ..\Mcl_TS_T2D35M10I0R0\config\Mcl.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5748g_lqfp176.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5748g_mapbga256.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5748g_mapbga324.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5747g_lqfp176.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5747g_mapbga256.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5747g_mapbga324.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5746g_lqfp176.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5746g_mapbga324.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5748c_lqfp176.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5748c_mapbga256.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5748c_mapbga324.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5747c_lqfp176.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5747c_mapbga256.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5747c_mapbga324.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5746c_lqfp176.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5746c_mapbga256.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5746c_mapbga324.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5746c_mapbga100.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5745c_lqfp176.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5745c_mapbga256.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5745c_mapbga100.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5746d_lqfp176.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5746d_mapbga256.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5746d_mapbga100.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5746b_lqfp176.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5746b_mapbga256.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5746b_mapbga100.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5745d_lqfp176.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5745d_mapbga256.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5745d_mapbga100.epd

  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5745b_lqfp176.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5745b_mapbga256.epd
  - ..\Adc_TS_T2D35M10I0R0 \autosar\Adc_mpc5745b_mapbga100.epd

- Code Generation Templates for parameters without variation points:
  - ..\Adc_TS_T2D35M10I0R0 \output\include\Adc_Cfg.h
  - ..\Adc_TS_T2D35M10I0R0 \output\include\Adc_CfgDefines.h
  - ..\Adc_TS_T2D35M10I0R0 \output\src\Adc_Cfg.c
  - ..\EcuM_TS_T2D35M10I0R0 \output\include\EcuM_Cfg.h
  - ..\Dem_TS_T2D35M10I0R0 \output\include\Dem_IntErrId.h
  - ..\Mcl_TS_T2D35M10I0R0 \output\include\CDD_Mcl_Cfg.h
  - ..\Mcl_TS_T2D35M10I0R0 \output\src\CDD_Mcl_Cfg.c
  - ..\Mcu_TS_T2D35M10I0R0 \output\include\Mcu_Cfg.h
  - ..\Mcu_TS_T2D35M10I0R0 \output\src\Mcu_Cfg.c

- Code Generation Templates for for variant aware parameters:
  - ..\Adc_TS_T2D35M10I0R0 \output\include\Adc_Cfg.h
  - ..\Adc_TS_T2D35M10I0R0 \output\include\Adc_CfgDefines.h
  - ..\Adc_TS_T2D35M10I0R0 \output\src\Adc_<VariantName>_PBcfg.c
  - ..\EcuM_TS_T2D35M10I0R0 \output\include\EcuM_Cfg.h
  - ..\Dem_TS_T2D35M10I0R0 \output\include\Dem_IntErrId.h
  - ..\Mcl_TS_T2D35M10I0R0 \output\include\CDD_Mcl_Cfg.h
  - ..\Mcl_TS_T2D35M10I0R0 \output\src\CDD_Mcl__<VariantName>_PBcfg.c
  - ..\Mcu_TS_T2D35M10I0R0 \output\include\Mcu_Cfg.h
  - ..\Mcu_TS_T2D35M10I0R0 \output\src\Mcu__<VariantName>_PBcfg.c

Steps to generate the configuration:

1. Copy the module folders Adc_TS_T2D35M10I0R0, Base_TS_T2D35M10I0R0, Dem_TS_T2D35M10I0R0, Resource_TS_T2D35M10I0R0, EcuM_TS_T2D35M10I0R0, Mcl_TS_T2D35M10I0R0, Mcu_TS_T2D35M10I0R0, Ecuc_TS_T2D35M10I0R0 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.

The internal connections of peripherals to ADC channels are as follows:

- Bandgap Reference PMC - ADC0 AN[17]
- Bandgap Reference PMC - ADC1 AN[49]

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_GroupStatus[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_GroupStatus[Group].eNotification field. This field is used by other functions.

5.1.3 ADC_EXCLUSIVE_AREA_03

Used in function Adc_Adcdig_Init. This function is called from Adc_Init ASR API. Protects the following:

- Write to Adc_GroupStatus[Group].eConversion
- Write to Adc_GroupStatus[Group].eAlreadyConverted
- Write to Adc_GroupStatus[Group].Adc_DmaContinuous
- Write to Adc_GroupStatus[Group].ResultIndex
- Write to Adc_GroupStatus[Group].bLimitCheckFailed
- Write to Adc_GroupStatus[Group].eHwTriggering
- Write to Adc_GroupStatus[Group].eNotification
- Write to Adc_Adcdig_aMhtChannelGroup
- Write to Adc_GroupStatus[Group].Adc_DmaContinuous
- Write to Adc_GroupStatus[Group].ResultIndex
- Write to Adc_GroupStatus[Group].bLimitCheckFailed
- Write to Adc_GroupStatus[Group].eHwTriggering
- Write to Adc_GroupStatus[Group].eNotification
- Write to Adc_GroupStatus[Group].eCtuTriggering
- Write to Adc_NCMRxMask[GroupIndex].Adc_u32NCMR0
- Write to Adc_NCMRxMask[GroupIndex].Adc_u32NCMR1
- Write to Adc_NCMRxMask[GroupIndex].Adc_u32NCMR2
- Write to Adc_UnitStatus[Unit].SwNormalQueue
- Write to Adc_UnitStatus[Unit].SwNormalQueueIndex
- Write to Adc_UnitStatus[Unit].SwInjectedQueue
- Write to Adc_UnitStatus[Unit].SwInjectedQueueIndex
- Write to Adc_UnitStatus[Unit].HwInjectedQueue
- Write to Adc_UnitStatus[Unit].HwInjectedQueueIndex
- Write to Adc_UnitStatus[Unit].HwNormalQueue
- Write to Adc_UnitStatus[Unit].HwNormalQueueIndex
- Write to Adc_UnitStatus[Unit].HwQueueGroupType
- Write to ADC registers
- Write to DMA registers
- Write to BCTU registers

No interruptions must occur while region is running because other ADC functions are also working with there fields and DMA registers.

5.1.4 ADC_EXCLUSIVE_AREA_04

Used in function Adc_ValidateSetMode, protects the write "SleepModeStatus" variable

5.1.5 ADC_EXCLUSIVE_AREA_05

Used in function Adc_Adcdig_SetMode, protects the read and write of MCR and MSR registers. This function is called from Adc_SetMode Non-ASR API. No interruptions must occur while these registers are wrote.

5.1.6 ADC_EXCLUSIVE_AREA_06

Used in function Adc_Adcdig_DeInit, TThis function is called from Adc_DeInit ASR API. Protects the following:

- Write to Adc_GroupStatus[GroupIdx].eNotification
- Write to Adc_NCMRxMask[GroupIndex].Adc_u32NCMR0
- Write to Adc_NCMRxMask[GroupIndex].Adc_u32NCMR1
- Write to Adc_NCMRxMask[GroupIndex].Adc_u32NCMR2
- Write to ADC registers
- Write to DMA registers
- Write to BCTU registers

5.1.7 ADC_EXCLUSIVE_AREA_07

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

- Write of Adc_UnitStatus[Unit].SwInjectedQueueIndex
- Write of Adc_UnitStatus[Unit].SwInjectedQueue
- Read and write of Adc_UnitStatus[Unit].SwNormalQueue
- Read and write of Adc_UnitStatus[Unit].SwNormalQueueIndex
- Read and write of ADC registers
- Read and write of DMA regs(disable DMA interrupt)
- Write of Adc_GroupStatus[Group].eConversion

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

5.1.8 ADC_EXCLUSIVE_AREA_08

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

- Write of Adc_GroupStatus[Group].eConversion
- Write of Adc_GroupStatus[Group].ResultIndex
- Write of Adc_GroupStatus[Group].eNotification
- Write of Adc_UnitStatus[Unit].SwInjectedQueueIndex
- Read of Adc_UnitStatus[Unit].SwInjectedQueue
- Read of Adc_UnitStatus[Unit].SwNormalQueueIndex
- Read of Adc_UnitStatus[Unit].SwNormalQueue
- Read and write of ADC registers
- Read and write of DMA registers(disable DMA interrupt)

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

5.1.9 ADC_EXCLUSIVE_AREA_09

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

- Read and write of Adc_GroupStatus[Group].eConversion
- Read and write of Adc_GroupStatus[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.

5.1.10 ADC_EXCLUSIVE_AREA_10

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

- Read and Write of Adc_UnitStatus[Unit].HwInjectedQueueIndex
- Write of Adc_UnitStatus[Unit].HwQueueGroupType
- Write of Adc_UnitStatus[Unit].HwNormalQueue
- Write of Adc_UnitStatus[Unit].HwNormalQueueIndex
- Write of Adc_UnitStatus[Unit].HwInjectedQueue
- Write to Adc_GroupStatus[Group].eConversion
- Write to Adc_GroupStatus[Group].eHwTriggering
- Write to Adc_GroupStatus[Group].ResultIndex
- Write to to ADC registers

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

5.1.11 ADC_EXCLUSIVE_AREA_11

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

- Write to Adc_GroupStatus[Group].eConversion
- Write to Adc_GroupStatus[Group].eHwTriggering
- Write to Adc_GroupStatus[Group].eNotification
- Write to Adc_Adcdig_abAlreadyCleared[Group]
- Write to Adc_UnitStatus[Unit].HwInjectedQueueIndex
- Write to Adc_UnitStatus[Unit].HwQueueGroupType
- Write to ADC registers
- Write to DMA registers

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

5.1.12 ADC_EXCLUSIVE_AREA_12

Used in function Adc_Adcdig_GetStreamLastPointer. This function is called from Adc_GetStreamLastPointer ASR API. Protects the following:

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

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

5.1.13 ADC_EXCLUSIVE_AREA_13

Used in ISR processing function Adc_DmaEndGroupConversion, and covers also the call to Adc_DmaEndNormalConv and Adc_DmaEndInjectedConv. It protects the following:

- Adc_GroupStatus[Group].ResultIndex
- Adc_GroupStatus[Group].eConversion
- Write to Adc_GroupStatus[Group].eAlreadyConverted
- Read of Adc_UnitStatus[Unit].SwNormalQueueIndex
- Read of Adc_UnitStatus[Unit].SwNormalQueue
- Read of Adc_UnitStatus[Unit].HwInjectedQueueIndex
- Read of Adc_UnitStatus[Unit].HwInjectedQueue

- Read of Adc_UnitStatus[Unit].HwNormalQueueIndex
- Read of Adc_UnitStatus[Unit].HwNormalQueue
- Write of Adc_GroupStatus[Group].bLimitCheckFailed
- Write to ADC registers
- Write of DMA registers

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

5.1.14 ADC_EXCLUSIVE_AREA_14

Used in ISR processing function Adc_Adcdig_EndNormalConv, protects the following:

- Read of Adc_UnitStatus[Unit].SwInjectedQueue
- Read of Adc_UnitStatus[Unit].SwNormalQueue
- Read of Adc_UnitStatus[Unit].SwNormalQueueIndex
- Write of Adc_UnitStatus[Unit].SwInjectedQueueIndex
- Write of Adc_CfgPtr->Groups[Group].pResultsBufferPtr
- Read and write of Adc_GroupStatus[Group].bLimitCheckFailed
- Read and write Adc_GroupStatus[Group].ResultIndex
- Read and write Adc_GroupStatus[Group].eConversion
- Write to ADC registers
- Write of Adc_UnitStatus[unit].HwQueueIndex
- Write Adc_UnitStatus[unit].SwInjQueueIndex

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

5.1.15 ADC_EXCLUSIVE_AREA_15

Used in ISR processing function Adc_Adcdig_EndInjectedConv, protects the following:

- Write to Adc_GroupStatus[Group].ResultIndex
- Write to Adc_GroupStatus[Group].eAlreadyConverted
- Write to Adc_GroupStatus[Group].eConversion
- Write to Adc_Adcdig_abAlreadyCleared[Group]
- Write to Adc_UnitStatus[Unit].HwInjectedQueueIndex
- Write to Adc_UnitStatus[Unit].HwNormalQueueIndex
- Write to Adc_GroupStatus[Group].ResultIndex
- Write to Adc_Bctu_au16ListChannelAddress[Unit]
- Write to ADC registers
- Write to BCTU registers

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

5.1.16 ADC_EXCLUSIVE_AREA_16

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

- Write to pStatus->Adc_UnitSelfTestStatus
- Write to pStatus->Adc_Calibrate_Failed_Steps[]
- Write to ADC registers

No interruptions must occur while these registers are wrote.

5.1.17 ADC_EXCLUSIVE_AREA_17

Used in function Adc_Adcdig_EndMultipleCtuConv, protects the following:

- Write to Adc_GroupStatus[Group].ResultIndex
- Write to Adc_GroupStatus[Group].eAlreadyConverted
- Write to Adc_GroupStatus[Group].eConversion
- Write to Adc_UnitStatus[Unit].HwInjectedQueueIndex

5.1.18 ADC_EXCLUSIVE_AREA_18

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

- Write to Adc_NCMRxMask[GroupId].Adc_u32NCMR0
- Write to Adc_NCMRxMask[GroupId].Adc_u32NCMR1
- Write to Adc_NCMRxMask[GroupId].Adc_u32NCMR2

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

5.1.19 ADC_EXCLUSIVE_AREA_19

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

- Write to eTempReturn variable
- Read of Adc_aUnitStatus[Unit].SwInjectedQueueIndex
- Read of Adc_aUnitStatus[Unit].HwInjectedQueueIndex

5.1.20 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

5.1.21 ADC_EXCLUSIVE_AREA_21

Used in ISR processing function Non ASR Adc_Adcdig_EnableCtuTrigger, protects the following:

- Write to Adc_UnitStatus[Unit].HwInjectedQueue[Adc_UnitStatus[Unit].HwInjectedQueueIndex]
- Write to Adc_UnitStatus[Unit].HwInjectedQueueIndex
- Write to Adc_GroupStatus[Group].eCtuTriggering
- Write to Adc_GroupStatus[Group].eConversion
- Write to Adc_GroupStatus[Group].ResultIndex
- Write to ADC registers

5.1.22 ADC_EXCLUSIVE_AREA_22

Used in ISR processing function Non ASR Adc_Adcdig_DisableCtuTrigger, protects the following:

- Write to Adc_GroupStatus[Group].eCtuTriggering
- Write to Adc_UnitStatus[Unit].HwInjectedQueueIndex
- Write to Adc_GroupStatus[Group].eConversion
- Write to ADC registers

5.1.23 ADC_EXCLUSIVE_AREA_23

Used in ISR processing function Non ASR Adc_Adcdig_HwResultReadGroup, protects the following:

- Read from ADC_CH_DATA register

5.1.24 ADC_EXCLUSIVE_AREA_25

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

- Write to Adc_NCMRxMask[GroupId].Adc_u32NCMR0
- Write to Adc_NCMRxMask[GroupId].Adc_u32NCMR1
- Write to Adc_NCMRxMask[GroupId].Adc_u32NCMR2

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

5.1.25 ADC_EXCLUSIVE_AREA_26

Used in the function Non ASR Adc_ConfigureThreshold to protect the call function Adc_Ipw_ConfigureThreshold The fields of this structure are updated by other ADC functions.

5.1.26 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 registers

5.1.27 ADC_EXCLUSIVE_AREA_29

Used in function Adc_ValidateStateCtuControlMode, protects the following:

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

5.1.28 ADC_EXCLUSIVE_AREA_30

Used in function Adc_SetChannel, protects the following:

- Write of Adc_aRuntimeGroupChannel[Group].pChannel
- Write of Adc_aRuntimeGroupChannel[Group].pu16Delays
- Write of Adc_aRuntimeGroupChannel[Group].u16Mask
- Write of Adc_aRuntimeGroupChannel[Group].ChannelCount
- Write of Adc_aRuntimeGroupChannel[Group].bRuntimeUpdated

5.1.29 ADC_EXCLUSIVE_AREA_31

Used in the function Adc_Init, Adc_EnableHardwareTrigger, Adc_DisableHardwareTrigger, Adc_EnableCTUTrigger, Adc_DisableCTUTrigger, Adc_SetMode, Adc_SetPowerState, Adc_EnableCtuControlMode, Adc_SetClockMode to protect the the following:

- Write to ADC registers

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.

Table 5-1. Exclusive Areas

	ADC_EA_00	ADC_EA_01	ADC_EA_02	ADC_EA_03	ADC_EA_04	ADC_EA_05	ADC_EA_06	ADC_EA_07	ADC_EA_08	ADC_EA_09	ADC_EA_10	ADC_EA_11	ADC_EA_12	ADC_EA_13	ADC_EA_14	ADC_EA_15	ADC_EA_16	ADC_EA_17	ADC_EA_18	ADC_EA_19	ADC_EA_20	ADC_EA_21	ADC_EA_22	ADC_EA_23	ADC_EA_24	ADC_EA_25	ADC_EA_26	ADC_EA_27	ADC_EA_28	ADC_EA_29
ADC_EA_00		x		x				x	x	x	x	x	x	x	x			x			x		x							x

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	ADC_EA_00	ADC_EA_01	ADC_EA_02	ADC_EA_03	ADC_EA_04	ADC_EA_05	ADC_EA_06	ADC_EA_07	ADC_EA_08	ADC_EA_09	ADC_EA_10	ADC_EA_11	ADC_EA_12	ADC_EA_13	ADC_EA_14	ADC_EA_15	ADC_EA_16	ADC_EA_17	ADC_EA_18	ADC_EA_19	ADC_EA_20	ADC_EA_21	ADC_EA_22	ADC_EA_23	ADC_EA_24	ADC_EA_25	ADC_EA_26	ADC_EA_27	ADC_EA_28	ADC_EA_29
ADC_EA_01	x			x				x	x	x	x	x	x	x	x			x			x		x							x
ADC_EA_02	x	x						x	x	x	x	x	x	x	x			x			x		x							x
ADC_EA_03	x	x			x			x	x	x	x	x	x	x	x			x			x		x							x
ADC_EA_04											x										x		x							x
ADC_EA_05											x										x		x							x
ADC_EA_06	x	x		x	x			x	x	x	x	x	x	x	x			x			x		x							x
ADC_EA_07	x	x		x			x		x	x	x	x	x	x	x			x			x		x							x

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	ADC_EA_00	ADC_EA_01	ADC_EA_02	ADC_EA_03	ADC_EA_04	ADC_EA_05	ADC_EA_06	ADC_EA_07	ADC_EA_08	ADC_EA_09	ADC_EA_10	ADC_EA_11	ADC_EA_12	ADC_EA_13	ADC_EA_14	ADC_EA_15	ADC_EA_16	ADC_EA_17	ADC_EA_18	ADC_EA_19	ADC_EA_20	ADC_EA_21	ADC_EA_22	ADC_EA_23	ADC_EA_24	ADC_EA_25	ADC_EA_26	ADC_EA_27	ADC_EA_28	ADC_EA_29
ADC_EA_08	x	x		x			x	x		x	x	x	x	x	x			x			x		x							x
ADC_EA_09	x	x		x			x	x	x		x	x	x	x	x			x			x		x							x
ADC_EA_10	x	x		x	x		x	x	x	x		x	x	x	x			x			x		x							x
ADC_EA_11	x	x		x			x	x	x	x	x		x	x	x			x			x		x							x
ADC_EA_12	x	x		x			x	x	x	x	x	x	x	x	x			x			x		x							x
ADC_EA_13	x	x		x			x	x	x	x	x	x	x		x			x			x		x							x
ADC_EA_14	x	x		x			x	x	x	x	x	x	x	x				x			x		x							x

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	ADC_EA_00	ADC_EA_01	ADC_EA_02	ADC_EA_03	ADC_EA_04	ADC_EA_05	ADC_EA_06	ADC_EA_07	ADC_EA_08	ADC_EA_09	ADC_EA_10	ADC_EA_11	ADC_EA_12	ADC_EA_13	ADC_EA_14	ADC_EA_15	ADC_EA_16	ADC_EA_17	ADC_EA_18	ADC_EA_19	ADC_EA_20	ADC_EA_21	ADC_EA_22	ADC_EA_23	ADC_EA_24	ADC_EA_25	ADC_EA_26	ADC_EA_27	ADC_EA_28	ADC_EA_29
ADC_EA_15	x	x		x			x	x	x	x	x	x	x	x				x			x		x							x
ADC_EA_16																														
ADC_EA_17	x	x					x	x	x	x	x	x	x	x							x		x							x
ADC_EA_18																														
ADC_EA_19																														
ADC_EA_20	x	x		x	x		x	x	x	x		x	x	x	x			x					x							x
ADC_EA_21	x	x		x	x		x	x	x	x	x	x	x	x	x			x					x							x

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	ADC_EA_00	ADC_EA_01	ADC_EA_02	ADC_EA_03	ADC_EA_04	ADC_EA_05	ADC_EA_06	ADC_EA_07	ADC_EA_08	ADC_EA_09	ADC_EA_10	ADC_EA_11	ADC_EA_12	ADC_EA_13	ADC_EA_14	ADC_EA_15	ADC_EA_16	ADC_EA_17	ADC_EA_18	ADC_EA_19	ADC_EA_20	ADC_EA_21	ADC_EA_22	ADC_EA_23	ADC_EA_24	ADC_EA_25	ADC_EA_26	ADC_EA_27	ADC_EA_28	ADC_EA_29
ADC_EA_22	x	x		x	x		x	x	x	x	x	x	x	x	x			x			x									x
ADC_EA_23																														
ADC_EA_24																														
ADC_EA_25																														
ADC_EA_26																														
ADC_EA_27																														
ADC_EA_28																														

Table continues on the next page...

Table 5-1. Exclusive Areas (continued)

	ADC_EA_00	ADC_EA_01	ADC_EA_02	ADC_EA_03	ADC_EA_04	ADC_EA_05	ADC_EA_06	ADC_EA_07	ADC_EA_08	ADC_EA_09	ADC_EA_10	ADC_EA_11	ADC_EA_12	ADC_EA_13	ADC_EA_14	ADC_EA_15	ADC_EA_16	ADC_EA_17	ADC_EA_18	ADC_EA_19	ADC_EA_20	ADC_EA_21	ADC_EA_22	ADC_EA_23	ADC_EA_24	ADC_EA_25	ADC_EA_26	ADC_EA_27	ADC_EA_28	ADC_EA_29
ADC_EA_29				x	x			x	x	x	x	x		x	x			x			x									

Note

- **ADC_EA_xx** means **ADC_EXCLUSIVE_AREA_xx**

5.2 Peripheral Hardware Requirements

The device provides two SAR Analog to Digital Converter HW units: ADC HW Unit 0 (10bit) and ADC HW Unit 1 (12bit).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 5-2. ADC ISR

ISR Name	HW INT Vector	Observations
ISR(Adc_Adcdig_EndGroupConvUnit0)	548	--
ISR(Adc_Adcdig_WatchDogThresholdUnit0)	550	--
ISR(Adc_Adcdig_EndGroupConvUnit1)	554	--
ISR(Adc_Adcdig_WatchDogThresholdUnit1)	556	--

If DMA transfer mode is used, MCL_DMA_CH_x_ISR routines should be used for each DMA channel. It depends on the MCL configuration. In this case, Adc_Adcdig_DmaTransferComplete**x** function should be configured for DMA notification parameter. This is required to update ADC driver internal statuses.

The following function should be configured for DMA notification parameter:

Table 5-3. DMA notification parameter

Function Name	Observations
Adc_Adcdig_DmaTransferComplete0	DMA notification parameter
Adc_Adcdig_DmaTransferComplete1	DMA notification parameter

Table 5-4. BCTU ISR

ISR Name	HW INT Vector	Observations
ISR(Bctu0_AdcEndConvInterrupt)	373	--

If DMA transfer mode is used, MCL_DMA_CH_x_ISR routines should be used for each DMA channel. It depends on the MCL configuration. In this case, Bctu0_Adc**x**DmaTransferComplete function should be configured for DMA notification parameter. This is required to update ADC driver internal statuses.

The following function should be configured for DMA notification parameter:

Table 5-5. DMA notification parameter

Function Name	Observations
Bctu0_Adc0DmaTransferComplete	DMA notification parameter
Bctu0_Adc1DmaTransferComplete	DMA notification parameter

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** In DMA mode, the Mcl is used to configure the DMA channel allocated for all ADC HW units.
- **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.
- **RTE:** Used to manage the exclusive area inside Adc module.

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).

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:

None

User Notification:

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

The syntax of this function is as follows:

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

An extern declaration of this function is available in Adc_<VariantName>_PBcfg.c. 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 syntax of this function is as follows:

```
void Adc_Wdg_Notification_<channel>()
```

An extern declaration of this function is available in Adc_<VariantName>_PBcfg.c. The function has to be implemented by the user.

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

The syntax of this function is as follows:

```
void Adc_Bctu_Notification_<trigger>()
```

An extern declaration of this function is available in Adc_<VariantName>_PBcfg.c. The function has to be implemented by the user.

The ADC Driver provides an additional notification callback per group that is called after each group conversion, in the very beginning of interrupt processing (in contrast to the normal user notification, which is called at the end). This notification is meant to be used together with the Adc_SetCchannel API, to allow the user to call the API and update the configuration at an early time, so that the changes can take effect in the very next configuration. If this notification is not desired, it can be disabled from the configuration interface or 'NULL_PTR' can be configured.

The syntax of this function is as follows:

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

Chapter 7
Memory Allocation

7.1 Sections to be defined in Adc_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_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
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_8	Constant Data	Used for constants that have to be aligned to 8 bit.
ADC_STOP_SEC_CONST_8	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.
more:ADC_STOP_SEC_CONST_UNSPECIFIED	Constant Data	End of above section.
ADC_START_SEC_VAR_INIT_16	Variables	Used for variables which have to be aligned to 16 bit. For instance used for variables of size 16 bit or used for composite data types: arrays, structs containing elements of maximum 16 bits. .These variables are initialized with values after every reset.
ADC_STOP_SEC_VAR_INIT_16	Variables	End of above section.
ADC_START_SEC_VAR_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

Table continues on the next page...

		containing elements of maximum 8 bits. .These variables are initialized with values after every reset.
ADC_STOP_SEC_VAR_INIT_8	Variables	End of above section.
ADC_START_SEC_VAR_INIT_BOOLEAN	Variables	Used for variables which have to be aligned to 1 bit. For instance used for variables of size 1 bit or used for composite data types: arrays, structs containing elements of maximum 1 bits. .These variables are initialized with values after every reset.
ADC_STOP_SEC_VAR_INIT_BOOLEAN	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_VAR_INIT_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 never cleared and never initialized by start-up code.
ADC_STOP_SEC_VAR_INIT_UNSPECIFIED_NO_CACHEABLE	Non-Cacheable Variables	End of above section.
ADC_START_SEC_VAR_NO_INIT_32_NO_CACHEABLE	Non-Cacheable Variables	Used for variables, structures, arrays when the SIZE (alignment) does fit the criteria of 32 bit, and that have to be stored in a non-cacheable memory section. These variables are never cleared and never initialized by start-up code.
ADC_STOP_SEC_VAR_NO_INIT_32_NO_CACHEABLE	Non-Cacheable Variables	End of above section.
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.

Table continues on the next page...

ADC_START_SEC_VAR_NO_INIT_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 never cleared and never initialized by start-up code.
ADC_STOP_SEC_VAR_NO_INIT_UNSPECIFIED_NO_CACHEABLE	Non-Cacheable Variables	End of above section.

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
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

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	AdcVersionInfoApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	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
	AdcLowPowerStatesSupport	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcPowerStateAsynchTransitionMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcPriorityQueueMaxDepth	Pre Compile parameter for all Variants of Configuration	Pre Compile
AdcGeneral-AdcPowerStateConfig			
	AdcPowerState	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcPowerStateReadyCbkRef	Pre Compile parameter for all Variants of Configuration	Pre Compile
AdcConfigSet			
ADCHwUnit- General			
	AdcTransferType	VariantPC or VariantPB	Post Build
	AdcClockSource	VariantPC or VariantPB	Post Build
	AdcHwUnitId	VariantPC or VariantPB	Post Build
	AdcLogicalUnitId	VariantPC or VariantPB	Post Build
	AdcPrescale	VariantPC or VariantPB	Post Build
	AdcAltPrescale	VariantPC or VariantPB	Post Build
	AdcPowerDownDelay	VariantPC or VariantPB	Post Build
	AdcAltPowerDownDelay	VariantPC or VariantPB	Post Build
	AdcMuxDelay	VariantPC or VariantPB	Post Build
	AdcAutoClockOff	VariantPC or VariantPB	Post Build
	AdcBypassSampling	VariantPC or VariantPB	Post Build
	AdcPresamplingInternalSignal0	VariantPC or VariantPB	Post Build
	AdcPresamplingInternalSignal1	VariantPC or VariantPB	Post Build
	AdcPresamplingInternalSignal2	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
AdcHwUnit - AdcNormalConvTimings			
	AdcSamplingDurationNormal	VariantPC or VariantPB	Post Build
	AdcSamplingDurationNormal1	VariantPC or VariantPB	Post Build
	AdcSamplingDurationNormal2	VariantPC or VariantPB	Post Build
AdcHwUnit - AdcAlternateConvTimings			
	AdcSamplingDurationAlt	VariantPC or VariantPB	Post Build
	AdcSamplingDurationAlt1	VariantPC or VariantPB	Post Build
	AdcSamplingDurationAlt2	VariantPC or VariantPB	Post Build
ADCChannel- General			
	AdcChannelConvTime	VariantPC or VariantPB	Post Build
	AdcChannelHighLimit	VariantPC or VariantPB	Post Build
	AdcLogicalChannelId	VariantPC or VariantPB	Post Build
	AdcChannelId	VariantPC or VariantPB	Post Build
	AdcChannelLimitCheck	VariantPC or VariantPB	Post Build
	AdcChannelLowLimit	VariantPC or VariantPB	Post Build
	AdcChannelRangeSelect	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
	AdcEnablePresampling	VariantPC or VariantPB	Post Build
	AdcEnableThresholds	VariantPC or VariantPB	Post Build
	AdcThresholdRegister	VariantPC or VariantPB	Post Build
	AdcWdogNotification	VariantPC or VariantPB	Post Build
AdcGroup- General			
	AdcGroupAccessMode	VariantPC or VariantPB	Post Build
	AdcGroupConversionMode	VariantPC or VariantPB	Post Build
	AdcGroupConversionType	VariantPC or VariantPB	Post Build
	AdcGroupId	VariantPC or VariantPB	Post Build
	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

Table continues on the next page...

Table 8-1. Configuration Parameters (continued)

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
	AdcStreamingBufferMode	VariantPC or VariantPB	Post Build
	AdcEnableDoubleBuffering	VariantPC or VariantPB	Post Build
	AdcStreamingNumSamples	VariantPC or VariantPB	Post Build
	AdcEnableChDisableChGroup	VariantPC or VariantPB	Post Build
	AdcWithoutInterrupts	VariantPC or VariantPB	Post Build
	AdcMultipleHardwareTriggerGroup	VariantPC or VariantPB	Pre Compile
AdcGroupDefinition			
	AdcGroupDefinition	VariantPC or VariantPB	Post Build
AdcHwTrig			
	AdcHwTrigSrc	VariantPC or VariantPB	Post Build
AdcGroupConversionConfiguration			
	AdcSamplingDuration	VariantPC or VariantPB	Post Build
	AdcSamplingDuration1	VariantPC or VariantPB	Post Build
	AdcSamplingDuration2	VariantPC or VariantPB	Post Build
AdcAlternateGroupConvTimings			
	AdcAltGroupSamplingDuration	VariantPC or VariantPB	Post Build
	AdcAltGroupSamplingDuration1	VariantPC or VariantPB	Post Build
	AdcAltGroupSamplingDuration2	VariantPC or VariantPB	Post Build
AdcHwUnit - AdcThresholdControl			
	AdcThresholdControlRegister	VariantPC or VariantPB	Post Build
	AdcHighThreshold	VariantPC or VariantPB	Post Build
	AdcLowThreshold	VariantPC or VariantPB	Post Build
AdcConfigSet- BCTUHwUnit			
	BCTUDMAChannelEnable	VariantPC or VariantPB	Post Build
BCTUHwUnit - BCTU_InputTrigger			
	BCTUInputTriggerID	VariantPC or VariantPB	Post Build
	BCTUTriggerLoop	VariantPC or VariantPB	Post Build
	BCTUMode	VariantPC or VariantPB	Post Build
	BCTUUserBuffer	VariantPC or VariantPB	Post Build
	BCTUUserCallback	VariantPC or VariantPB	Post Build
BCTU_InputTrigger - AdcChannelTriggered			
	AdcChannel	VariantPC or VariantPB	Post Build
	ADCHWUNIT	VariantPC or VariantPB	Post Build
	COMMAND	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
NonAutosar			
	AdcSetModeApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableGroupDependentChannelNames	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcBypassConsistencyLoop	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableChDisableChApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcGetInjectedConvStatusApi	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
	AdcPreSamplingOnce	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableSetChannel	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableInitialNotification	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableDualClockMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableThresholdConfiguration	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableCalibration	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableCtuTrigNonAutosarApi	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
	AdcEnableCtuControlModeApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableDmaTrasferMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
	BctuEnableDmaTrasferMode	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableWatchdogFunctionality	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcUseSoftwareInjectedGroups	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcUseHardwareNormalGroups	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcDisableDemReportErrorStatus	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableMultiHardwareTrigger	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcHardwareQueueMaxDepth	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableUserModeSupport	Pre Compile parameter for all Variants of Configuration	Pre Compile
AdcDemEventParameterRefs			
	ADC_E_TIMEOUT	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
AdcPublishedInformation			
	AdcChannelValueSigned	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
	AdcGroupFirstChannelFixed	VariantPC or VariantPB	Post Build
	AdcMaxChannelResolution	VariantPC or VariantPB	Post Build

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 Adc_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_Adcdig_DmaTransferComplete0

This function handles internal ADC updates after a DMA transfer is completed.

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_Adcdig_DmaTransferComplete0(void);`

10.1.3.2 Function Adc_Adcdig_DmaTransferComplete1

This function handles internal ADC updates after a DMA transfer is completed.

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_Adcdig_DmaTransferComplete1(void);`

10.1.3.3 Function Adc_Adcdig_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: `ISR(Adc_Adcdig_EndGroupConvUnit0);`

10.1.3.4 Function Adc_Adcdig_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: `ISR(Adc_Adcdig_EndGroupConvUnit1);`

10.1.3.5 Function Adc_Adcdig_WatchDogThresholdUnit0

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.

Note

This ISR is only defined if the ADC_WDG_SUPPORTED is enabled.

Prototype: `ISR(Adc_Adcdig_WatchDogThresholdUnit0);`

10.1.3.6 Function Adc_Adcdig_WatchDogThresholdUnit1

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.

Note

This ISR is only defined if the ADC_WDG_SUPPORTED is enabled.

Prototype: `ISR(Adc_Adcdig_WatchDogThresholdUnit1);`

10.1.3.7 Function Bctu0_AdcEndConvInterrupt

This function handles the BCTU_0_ADC_EndConvInterrupt.

Details:

The function handles the BCTU_0_ADC_EndConvInterrupt.

Prototype: `ISR(Bctu0_AdcEndConvInterrupt);`

10.1.3.8 Function Bctu0_Adc0DmaTransferComplete

This function handles internal BCTU updates after a DMA transfer is completed.

Details:

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

Return: void.

Prototype: `void Bctu0_Adc0DmaTransferComplete(void);`

10.1.3.9 Function Bctu0_Adc1DmaTransferComplete

This function handles internal BCTU updates after a DMA transfer is completed.

Details:

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

Return: void.

Prototype: `void Bctu0_Adc1DmaTransferComplete(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. >>

[ADC384]

<< 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

[ADC414]

<< The ADC module's environment shall check the integrity (see Note ADC413) 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;

[ADC415]

<< The ADC module shall not check the integrity (see Note ADC413) if several calls for the same ADC group are used during runtime in different tasks or ISRs. >>

[ADC230]

<< Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file Dem_IntErrId.h and included via Dem.h. >>

[ADC247]

<< If the register can affect several hardware modules and if it is an I/O register, it shall be initialized by the PORT driver. >>

[ADC248]

<< 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. >>

[ADC249]

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

[ADC250]

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

[ADC421]

<< 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. >>

[ADC422]

<< 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). >>

[ADC358]

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

[ADC146]

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

[ADC283]

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

[ADC273]

<< 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). >>

[ADC120]

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

[ADC121]

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

[ADC305]

<< 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. >>

[ADC219]

<< The ADC module's environment shall guarantee the consistency of the data that has been read by checking the return value of Adc_GetGroupStatus. >>

Document Number IM35ADCASR4.2 Rev002R1.0.0
Revision 1.0