Integration Manual

for MPC5634M ADC Driver

Document Number: IM14ADCASR3.0R2.0.0
Rev. 1.2

freescale™

Contents

Section Number **Title** **Page**

Chapter 1
Revision History

Table 1-1. Revision History

Revision	Date	Author	Description
1.0	03-Feb-2011	Alfredo Di Martino	Update for Monaco automatic documentation
1.1	02-Aug-2011	Alfredo Di Martino	Update for Monaco HF automatic documentation
1.2	19-Dec-2011	Alfredo Di Martino	Updated for Monaco RTM 2.0.0

Chapter 2
Introduction

This integration manual describes the integration requirements for ShortName Driver for MPC5634M microcontrollers.

2.1 Supported Derivatives

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

Table 2-1. MPC5634M Derivatives

Freescale Semiconductor	mpc5634m_bga208, mpc5634m_qfp144, mpc5634m_qfp176

All of the above microcontroller devices are collectively named as MPC5634M .

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

Table continues on the next page...

Table 2-2. Acronyms and Definitions (continued)

Term	Definition
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 3.0ShortName Driver Software Specification Document.	V2.2.0 R3.0 Rev 0001
2	MPC5634M Reference Manual	Rev. 6, 4 October 2011

Chapter 3
Building the Driver

This section describes the source files and various compilers, linker options used for building the Autosar ShortName driver for Freescale SemiconductorMPC5634M . It also explains the EB Tresos Studio plugin setup procedure.

3.1 Build Options

The ShortName driver files are compiled using

- GHS 5.2.4
- DIAB 5_8_0_02 wind00198363 20100511 123238
- CW Version 4.3 build 182

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

Note

The TS_T2D14M20I0R0 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 PowerPC architecture and Derivative_Id = 14 identifies the MPC5634M)

3.1.1 CW Compiler/Linker/Assembler Options

Table 3-1. Compiler Options

Option	Description
-proc Zen	Generates and links object code for Zen processor. The compiler uses unsigned as the default parameter for the -char switch
-lang c	Expects source code to conform to the language specified by the ISO/IEC 9899-1990 ("C90") standard
-opt all	This option is selected all optimization (the same as -opt speed,level=4,intrinsics,noframe)
-common off	Disables moving uninitialized data into a common section
-sdatathreshold 0	Specifies the threshold size (in bytes) for an item considered by the linker to be small data. (The linker stores small data items in the Small Data address space. The compiler can generate faster code to access this data.)
-sdata2threshold 0	Specifies the threshold size (in bytes) for an item considered by the linker to be small constant data. (The linker stores small constant data items in the Small Constant Data address space.)
-vle	Tells the compiler and linker to generate and lay out Variable Length Encoded (VLE) instructions, available on Zen variants of Power Architecture processors
-use_lmw_stmw on	Enables the use of multiple load and store instructions for function prologues and epilogues
-ir	Include the debug information
-ppc_asm_to_vle	Converts regular Power Architecture assembler mnemonics to equivalent VLE (Variable Length Encoded) assembler mnemonics in the inline assembler
-cpp_exceptions off	When on, generates executable code for C++ exceptions. When off, generates smaller, faster executable code
-func_align 4	Specifies alignment of functions in executable code
-sym dwarf-2,full	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format)
-gdwarf-2	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format). The linker ignores debugging information that is not in the Dwarf 1, Dwarf 2 format
-w on	Turns on most warning messages
-r	Compiler should expect function prototypes
-w undefmacro	Issues warning messages on the use of undefined macros in #if and #elif conditionals
-char unsigned	Controls the default sign of the char data type: char data items are unsigned
-nosyspath	Performs a search of both the user and system paths, treating #include statements of the form #include xyz the same as the form #include "xyz"
-fp none	No floating point code generation
-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
-DEU_DISABLE_ANSILIB_CALLS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the EU_DISABLE_ANSILIB_CALLS preprocessor symbol.
-DMCAL_CER_VALIDATION	-D defines a preprocessor symbol for CER Report

Table continues on the next page...

Table 3-1. Compiler Options (continued)

Option	Description
-DMCAL_VERSION_CHECK	-D defines enable the cross check between the AutoSar component Version Numbers
-DMWERKS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the CWpreprocessor symbol.

Table 3-2. Assembler Options

Option	Description
-proc Zen	Generates and links object code for Zen processor. The compiler uses unsigned as the default parameter for the -char switch
-vle	Tells the compiler and linker to generate and lay out Variable Length Encoded (VLE) instructions, available on Zen variants of Power Architecture processors
-sym dwarf-2,full	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format)
-gdwarf-2	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format). The linker ignores debugging information that is not in the Dwarf 1, Dwarf 2 format.

Table 3-3. Linker Options

Option	Description
-proc Zen	Generates and links object code for Zen processor. The compiler uses unsigned as the default parameter for the -char switch
-code_merging all	Removes duplicated functions to reduce object code size
-far_near_addressing	Simplifies address computations to reduce object code size and improve performance
-vle_enhance_merging	Removes duplicated functions that are called by functions that use VLE instructions to reduce object code size
-listdwarf	DWARF debugging information in the linker's map file
-sym dwarf-2,full	Generate DWARF-2-conforming debugging information (Debug With Arbitrary Record Format)
-char unsigned	Controls the default sign of the char data type: char data items are unsigned.

3.1.2 DIAB Compiler/Linker/Assembler Options

Table 3-4. Compiler Options

Option	Description
-tPPCE200Z3VEG:simple	Sets target processor to PPCE200Z3, generates ELF using EABI conventions, All Single Hardware Floating Point (Single precision uses hardware, double precision is mapped to single precision), 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
-Xsize-opt	Optimize for size rather than speed when there is a choice

Table continues on the next page...

Table 3-4. Compiler Options (continued)

Option	Description
-Xsmall-data=0	Set Size Limit for “small data” Variables to zero.
-Xsmall-const=0	Set Size Limit for “small const” Variables to zero.
-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
-Xalign-functions=4	Align each function on an address boundary divisible by 4
-g	Generate symbolic debugger information. Do most target-independent optimizations. Also, disable most target-dependent optimizations: option -g2 also disables basic reordering and all peephole optimizations.
-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
-Xforce-prototypes	Generate warnings if a function is used without a previous prototype declaration
-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
-Xlint	Generate warnings when suspicious and non-portable C code is encountered. Enables all warnings
-ei1604	Suppress the warning messages 1604.
-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
-DDIAB	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the DIAB preprocessor symbol.
-DEU_DISABLE_ANSILIB_CALLS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the EU_DISABLE_ANSILIB_CALLS preprocessor symbol.
-DMCAL_CER_VALIDATION	-D defines a preprocessor symbol for CER Report

Table 3-5. Assembler Options

Option	Description
-tPPCE200Z3VEN:simple	Selects target processor: PPCE200Z3, generates ELF using EABI conventions, NO floating point support, 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

Table 3-6. Linker Options

Option	Description
-tPPCE200Z3VEN:simple	Selects target processor: PPCE200Z3, generates ELF using EABI conventions, NO floating point support, 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
-lc	Specifies to linker to search for libc.a
-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.
-Xlibc-old	Enables usage of legacy (pre-release 5.6) libraries

3.1.3 GHS Compiler/Linker/Assembler Options

Table 3-7. Compiler Options

Option	Description
-cpu=ppc563xm	Selects target processor: ppc563xm
-ansi	Enforces strict ANSI mode (C89 standard)
-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.
--no_commons	Allocates uninitialized global variables to a section and initializes them to zero at program startup. This may improve optimizations by giving the compiler optimizer more information about the location of the variable.
-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

Table continues on the next page...

Table 3-7. Compiler Options (continued)

Option	Description
-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
-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
-DGHS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the GHS preprocessor symbol.
-DEU_DISABLE_ANSILIB_CALLS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the EU_DISABLE_ANSILIB_CALLS preprocessor symbol.
-DMCAL_CER_VALIDATION	-D defines a preprocessor symbol for CER Report
-DMCAL_VERSION_CHECK	-D defines enable the cross check between the AutoSar component Version Numbers

Table 3-8. Assembler Options

Option	Description
-cpu=ppc563xm	Selects target processor: ppc563xm

Table 3-9. Linker Options

Option	Description
-cpu=ppc563xm	Selects target processor: ppc563xm
-nostartfiles	Do not use Start files.
-vle	Enables VLE code generation
-linker_warnings	Display linker warnings

3.1.4 CSMC Compiler/Linker/Assembler Options

Table 3-10. Compiler Options

Option	Description
-l	Create listing file; this option directs the compiler to produce an assembly language file with C source line interspersed in it. Please note that the C source lines are commented in the assembly language file: they start with ';'.
+modvc	Memory model with "medium size" application, in detail: "data" less than 64kb, "constants" less than 64kb, no code size limit
+rev	Tells the compiler to reverse the order of bits in the bitfields. You need this option in order to use most non-Cosmic header files.
-pc99	authorize the repetition of the const and volatile modifiers in the declaration either directly or indirectly in the typedef.
-odB5	disable the optimization B5.
-pxf	prefix filenames in the debug information with absolute full path name.
+debug	produce debug information to be used by the debug utilities provided with the compiler and by any external debugger.
-DCSMC	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the CSMC preprocessor symbol.
-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
-DEU_DISABLE_ANSILIB_CALLS	-D defines a preprocessor symbol and optionally can set it to a value. This one defines the EU_DISABLE_ANSILIB_CALLS preprocessor symbol.
-DMCAL_CER_VALIDATION	-D defines a preprocessor symbol for CER Report
-DMCAL_VERSION_CHECK	-D defines enable the cross check between the AutoSar component Version Numbers

Table 3-11. Assembler Options

Option	Description
-l	create a listing file. The name of the listing file is derived from the input file name by replacing the suffix by the ".ls" extension

Table 3-12. Linker Options

Option	Description
-p	display symbols with physical address instead of logical address in the map file.

3.2 Files required for Compilation

This section describes the include files required to compile, assemble (if assembler code) and link the ShortName driver for MPC5634M 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.

ShortName Files

- ..\ Prefix _ TS_T2D14M20I0R0 \src\Adc.c
- ..\ Prefix _ TS_T2D14M20I0R0 \src\Adc_NonASR.c
- ..\ Prefix _ TS_T2D14M20I0R0 \src\EQADC_LLD.c
- ..\ Prefix _ TS_T2D14M20I0R0 \src\Adc_Irq.c
- ..\ Prefix _ TS_T2D14M20I0R0 \src\Adc_LLD.c
- ..\ Prefix _ TS_T2D14M20I0R0 \src\Dma_LLD.c
- ..\ Prefix _ TS_T2D14M20I0R0 \include\Adc.h
- ..\ Prefix _ TS_T2D14M20I0R0 \include\Adc_NonASR.h
- ..\ Prefix _ TS_T2D14M20I0R0 \include\Adc_LLD.h
- ..\ Prefix _ TS_T2D14M20I0R0 \include\EQADC_LLD.h
- ..\ Prefix _ TS_T2D14M20I0R0 \include\EQADC_LLD_CfgEx.h
- ..\ Prefix _ TS_T2D14M20I0R0 \include\Dma_LLD.h
- ..\ Prefix _ TS_T2D14M20I0R0 \include\Reg_eSys_EQADC.h
- ..\ Prefix _ TS_T2D14M20I0R0 \include\Reg_eSys_DMA.h

ShortName Generated Files

- ADC_Cfg.c (For PC Variant) - This file should be generated by the user using a configuration tool for compilation.
- ADC_PBcfg.c (For PB Variant) - This file should be generated by the user using a configuration tool for compilation.
- ADC_Cfg.h - This file should be generated by the user using a configuration tool for compilation.

Files from Base common folder

- ..\Base_ TS_T2D14M20I0R0 \include\Cer.h
- ..\Base_ TS_T2D14M20I0R0 \include\Compiler.h
- ..\Base_ TS_T2D14M20I0R0 \include\Compiler_Cfg.h
- ..\Base_ TS_T2D14M20I0R0 \include\ComStack_Types.h
- ..\Base_ TS_T2D14M20I0R0 \include\Mcal.h
- ..\Base_ TS_T2D14M20I0R0 \include\MemMap.h
- ..\Base_ TS_T2D14M20I0R0 \include\Platform_Types.h
- ..\Base_ TS_T2D14M20I0R0 \include\Reg_eSys.h
- ..\Base_ TS_T2D14M20I0R0 \include\Reg_Macros.h
- ..\Base_ TS_T2D14M20I0R0 \include\Std_Types.h

Files from Dem folder:

- ..\Dem_ TS_T2D14M20I0R0 \include\Dem.h

- ..\Dem_ TS_T2D14M20I0R0 \include\Dem_IntErrId.h
- ..\Dem_ TS_T2D14M20I0R0 \include\Dem_Types.h

Files from Det folder:
- ..\Det_ TS_T2D14M20I0R0 \include\Det.h

Files from SchM folder:
- ..\SchM_ TS_T2D14M20I0R0 \src\SchM_Adc.c
- ..\SchM_ TS_T2D14M20I0R0 \include\SchM_Adc.h

Note:

<plugin_name>: TS_T<2>D<14>M<SW_Version_Major>I<SW_Version_Minor>R0

(i.e. Target_Id = 2 identifies PowerPC architecture and Derivative_Id = 11 identifies the MPC5634M)

3.3 Setting up the Plug-ins

The ShortName driver was designed to be configured by using the EB Tresos Studio (version Tresos 2010a.sr4 20100415-release2010a-sr4 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_T2D14M20I0R0\config\Adc.xdm
  - ..\Base_TS_T2D14M20I0R0\config\Base.xdm
  - ..\Resource_TS_T2D14M20I0R0\config\Resource.xdm
- VSMD (Vendor Specific Module Definition) file(s) in AUTOSAR compliant EPD format:
  - ..\Adc_TS_T2D14M20I0R0\autosar\Adc.epd
- Code Generation Templates for Pre-Compile time configuration parameters:
  - ..\Adc_TS_T2D14M20I0R0\generate_PC\include\Adc_Cfg.h
  - ..\Adc_TS_T2D14M20I0R0\generate_PC\src\Adc_Cfg.c
  - ..\Adc_TS_T2D14M20I0R0\generate_PC\Adc_Clock_Tree.m
  - ..\Adc_TS_T2D14M20I0R0\generate_PC\Adc_VersionCheck_Inc.m
  - ..\Adc_TS_T2D14M20I0R0\generate_PC\Adc_VersionCheck_Src.m
  - ..\Adc_TS_T2D14M20I0R0\generate_PC\Adc_RegOperations.m
- Code Generation Templates for Post-Build time configuration parameters:
  - ..\Adc_TS_T2D14M20I0R0\generate_PB\src\Adc_PBcfg.c
  - ..\Adc_TS_T2D14M20I0R0\generate_PB\Adc_VersionCheck_Src_PB.m

- ..\Adc_TS_T2D14M20I0R0\generate_PB\Adc_RegOperations_PB.m
- ..\Adc_TS_T2D14M20I0R0\generate_PB\Adc_RegOperations.m

Steps to generate the configuration:

1. Copy the module folders Adc_TS_T2D14M20I0R0, Base_TS_T2D14M20I0R0, Resource_TS_T2D14M20I0R0, EcuM_TS_T2D14M20I0R0, Mcu_TS_T2D14M20I0R0 into the Tresos plugins folder.
2. Set the desired Tresos Output location folder for the generated sources and header files.
3. Use the EB tresos Studio GUI to modify ECU configuration parameters values.
4. Generate the configuration files.

Dependencies

- **MCU** is required for the reference clock.
- **RESOURCE** is required to select processor derivative. Current ShortName driver has support for the following derivatives, everyone having attached a Resource file: mpc5634m_bga208, mpc5634m_qfp144, mpc5634m_qfp176.
- **DET** is required for signaling the development error detection (parameters out of range, null pointers, etc).
- **SchM** is required in order to support the critical sections.
- **DEM** is required for signaling the production error detection (hardware failure, etc).

Chapter 4
Function calls to module

4.1 Function Calls during Start-up

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

Note:

Before to start any ADC conversion, according to the AUTOSAR requirement ADC421, it's 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 Schedule Manager (SchM) for entering and exiting the critical regions. SchM implementation is done by the integrators of the MCAL using OS or non- OS services. For testing the ADC, stubs are used for SchM. 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]. Notification 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].Notification field. This field is used by other functions.

5.1.3 ADC_EXCLUSIVE_AREA_02 Used in function Adc_GetGroupStatus, protects the read of ADC_GroupStatus[Group].Conversion field. This field is used by other functions.

5.1.4 ADC_EXCLUSIVE_AREA_03 Used in function Adc_DeInit, protects the read of ADC_GroupStatus[Group].Conversion field. This field is used by other functions. No interruptions must occur during this DeInit loop for status check. This exclusive area is executed only if Development Error Detection is enabled.

5.1.5 ADC_EXCLUSIVE_AREA_04 Used in function Adc_EnableHardwareTrigger, protects the read and usage of the fields:

- ADC_UnitStatus[unit].QueueIndex
- ADC_UnitStatus[unit].HwQueueIndex

This fields are used by other functions. No interruptions must occur during the read of these variables. This exclusive area is executed only if Development Error Detection is enabled.

5.1.6 ADC_EXCLUSIVE_AREA_05 Used in function Adc_StartGroupConversion, protects the read and usage of the fields:

- ADC_UnitStatus[unit].QueueIndex
- ADC_UnitStatus[unit].HwQueueIndex
- ADC_UnitStatus[unit].Queue[x]
- ADC_GroupStatus[Group].Conversion

This fields are used by other functions.. No interruptions must occur during the read of these variables. This exclusive area is executed only if Development Error Detection is enabled.

5.1.7 ADC_EXCLUSIVE_AREA_06 Used in function ADCDig_LLD_DeInit, protects the read and write of all ADC and CTU registers. This LLD function is called from Adc_DeInit ASR API. No interruptions must occur while these registers are reset.

5.1.8 ADC_EXCLUSIVE_AREA_07 Used in function ADCDig_LLD_StartGroupConversion. This LLD function is called from Adc_StartGroupConversion ASR API. Protects the following:

- Read and write of ADC_UnitStatus[unit].Queue – insert into the queue
- Read and write of ADC_UnitStatus[unit].QueueIndex
- Read and write of ADC registers
- Read and write of DMA regs(disable DMA interrupt)
- Write of ADC_UnitStatus[unit].InjConvOngoin
- Write of ADC_GroupStatus[Group].Conversion

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_08 Used in function ADCDig_LLD_StopGroupConversion. This LLD function is called from Adc_StartGroupConversion ASR API. Protects the following:

- Write of ADC_GroupStatus[Group].Conversion
- Write of ADC_UnitStatus[unit].Status
- Write of ADC_GroupStatus[Group].ResultIndex
- Write of ADC_GroupStatus[Group].Notification
- Read and write of ADC_UnitStatus[unit].QueueIndex

- Read and write of ADC_UnitStatus[unit].Queue – remove from queue
- 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.

5.1.10 ADC_EXCLUSIVE_AREA_09 Used in function ADCDig_LLD_ReadGroup. This LLD function is called from Adc_ReadGroup ASR API. Protects the following:

- Read and write of ADC_GroupStatus[Group].Conversion
- Read and write of ADC_GroupStatus[Group].ResultIndex
- Read of Adc_Cfg_Ptr->Groups[Group].ResultsBufferPtr[Group]

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

5.1.11 ADC_EXCLUSIVE_AREA_10 Used in function ADCDig_LLD_EnableHardwareTrigger. This LLD function is called from Adc_EnableHardwareTrigger ASR API. Protects the following:

- Write to ADC_UnitStatus[unit].HwQueue
- Write to ADC_UnitStatus[unit].HwQueueIndex
- Write to ADC_GroupStatus[Group].Conversion
- Write to ADC_GroupStatus[Group].HwTriggering
- Write to ADC_GroupStatus[Group].ResultIndex
- Write to to ADC registers
- Write to to CTU registers

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

5.1.12 ADC_EXCLUSIVE_AREA_11 Used in function ADCDig_LLD_DisableHardwareTrigger. This LLD function is called from Adc_DisableHardwareTrigger ASR API. Protects the following:

- Write to ADC_GroupStatus[Group].Notification
- Write to ADC_GroupStatus[Group].HwTriggering
- Write to CTU registers
- Write to ADC registers
- Write to ADC_GroupStatus[Group].Conversion
- Write to ADC_UnitStatus[unit].HwQueue[0]
- Write to ADC_UnitStatus[unit].HwQueueIndex

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

5.1.13 ADC_EXCLUSIVE_AREA_12 Used in function ADCDig_LLD_GetStreamLastPointer. This LLD function is called from Adc_GetStreamLastPointer ASR API. Protects the following:

- Read and write to ADC_GroupStatus[Group].Conversion
- 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.

5.1.14 ADC_EXCLUSIVE_AREA_13 Used in ISR processing function Adc_Dma_Interrupt_Common_Func, and covers also the call to Adc_FIFO_Disable_Func. It protects the following:

- Write of DMA registers
- Read of ADC_UnitStatus[unit].Queue
- Calls Adc_DMAEndNormalConv:
  - Read and write to ADC_GroupStatus[Group].Conversion
  - Read and write to ADC_GroupStatus[Group].ResultIndex
  - Read and write to ADC registers
  - Read and write to ADC_UnitStatus[unit].QueueIndex
  - Write ADC_UnitStatus[unit].Status

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

5.1.15 ADC_EXCLUSIVE_AREA_14 Used in ISR processing function Adc_Irq_CFIFO_Empty_Common_Func, protects the following:

- Read and write to ADC_UnitStatus[unit].Queue
- Read and write to ADC_GroupStatus[Group].ResultIndex
- Write to Adc_Cfg_Ptr->Groups[Group].ResultsBufferPtr
- Read and write to ADC_GroupStatus[Group].Conversion
- Write to ADC registers
- Read and write to ADC_UnitStatus[unit].Status
- Read and write to ADC_UnitStatus[unit].QueueIndex

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

5.1.16 ADC_EXCLUSIVE_AREA_15 Used in ISR processing function Adc_Irq_RFIFO_Full_Common_Func, protects the following:

- Read and write to ADC_UnitStatus[unit].Queue
- Read and write to ADC_GroupStatus[Group].ResultIndex
- Write to Adc_Cfg_Ptr->Groups[Group].ResultsBufferPtr
- Read and write to ADC_GroupStatus[Group].Conversion
- Write to ADC registers

- Read and write to ADC_UnitStatus[unit].Status
- Read and write to ADC_UnitStatus[unit].QueueIndex

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

5.1.17 ADC_EXCLUSIVE_AREA_17 Used in function Adc_GetStreamLastPointer, protects the read of ADC_GroupStatus[Group].Conversion field. This field is updated by other ADC functions.

5.1.18 ADC_EXCLUSIVE_AREA_18 Used in function Adc_ReadGroup, protects the read of ADC_GroupStatus[Group].Conversion field. This field is updated by other ADC functions.

5.1.19 ADC_EXCLUSIVE_AREA_19 Used in function Adc_StopGroupConversion, protects the read of ADC_GroupStatus[Group].Conversion field. This field is updated by other ADC functions.

5.1.20 ADC_EXCLUSIVE_AREA_20 Used in function Adc_EnableHwTrigger, protects: ADC_GroupsInHwQueue(unit) and ADC_UnitStatus[unit].QueueIndex. This field is updated by other ADC functions.

5.1.21 ADC_EXCLUSIVE_AREA_25 Used in function Adc_EnableHwTrigger, protects: Adc_NCMRx_Mask[Adc_GroupId].Adc_NCMR0, Adc_NCMRx_Mask[Adc_GroupId].Adc_NCMR1 and Adc_NCMRx_Mask[Adc_GroupId].Adc_NCMR2. This field is updated by other ADC functions.

Critical Region Exclusive Matrix

Below is the table depicting the exclusivity between different critical region IDs from the ShortName 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.

Figure 5-1. Exclusive Areas

5.2 Peripheral Hardware Requirements

The device provides two precision Analog to Digital Converter HW units: ADC HW Unit 0 at 10-bit and ADC HW Unit 1 at 12-bit.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_Irq_eQADCA_PopResult0)	105	for Monaco1.5M
ISR(Adc_Irq_eQADCA_PopResult1)	110	for Monaco1.5M
ISR(Adc_Irq_eQADCA_PopResult2)	115	for Monaco1.5M
ISR(Adc_Irq_eQADCA_PopResult3)	120	for Monaco1.5M
ISR(Adc_Irq_eQADCA_PopResult4)	125	for Monaco1.5M
ISR(Adc_Irq_eQADCA_PopResult5)	130	for Monaco1.5M
ISR(Adc_Irq_eQADCA_CFIFO0_Empty)	104	for Monaco1.5M
ISR(Adc_Irq_eQADCA_CFIFO1_Empty)	109	for Monaco1.5M
ISR(Adc_Irq_eQADCA_CFIFO2_Empty)	114	for Monaco1.5M
ISR(Adc_Irq_eQADCA_CFIFO3_Empty)	119	for Monaco1.5M
ISR(Adc_Irq_eQADCA_CFIFO4_Empty)	124	for Monaco1.5M
ISR(Adc_Irq_eQADCA_CFIFO5_Empty)	129	for Monaco1.5M
ISR(Adc_Irq_eQADCA_RFIFO0_Full)	104	for Monaco1.5M
ISR(Adc_Irq_eQADCA_RFIFO1_Full)	109	for Monaco1.5M
ISR(Adc_Irq_eQADCA_RFIFO2_Full)	114	for Monaco1.5M
ISR(Adc_Irq_eQADCA_RFIFO3_Full)	119	for Monaco1.5M
ISR(Adc_Irq_eQADCA_RFIFO4_Full)	124	for Monaco1.5M
ISR(Adc_Irq_eQADCA_RFIFO5_Full)	129	for Monaco1.5M

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

- **Det** This module is necessary for enabling Development error detection. The API function used is Det_ReportError(). The activation/deactivation of Development error detection is configurable using 'CanDevErrorDetect' configuration parameter.
- **Dem:** This module is necessary for enabling reporting of production relevant error status. The API function used is Dem_ReportErrorStatus().
- **Mcu:** In DMA mode, the MCU is used to configure the DMA channel allocated for both. Also MCU is used to configure the ADC module clock source.
- **Resource:** Sub-Derivative model is selected from Resource configuration.

Chapter 6
Main API Requirements

6.1 Main functions calls within BSW scheduler, at which cycle (10ms, 20ms,...?)

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 channel 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_<channel>()
```

An extern declaration of this function is available in Adc_PBcfg.c. The function has to be implemented by the user.

Chapter 7
Memory Allocation

7.1 Sections to be defined in MemMap.h

For Post Build data:

```
#ifdef ADC_START_CONFIG_DATA_UNSPECIFIED
#undef ADC_START_CONFIG_DATA_UNSPECIFIED
#undef MEMMAP_ERROR
/*Memory Section for Post Build Data to be defined here.
    Example given in the next line*/
#pragma ghs section const=".pbgpt_cfg"
#endif
#ifdef ADC_STOP_CONFIG_DATA_UNSPECIFIED
#undef ADC_STOP_CONFIG_DATA_UNSPECIFIED
#undef MEMMAP_ERROR
/*End of section to be mentioned here. Example given in the
    next line.*/
#pragma ghs section
#endif
```

For Code:

```
#ifdef ADC_START_SEC_CODE
#undef ADC_START_SEC_CODE
#undef MEMMAP_ERROR
/*Memory Section for Code to be defined here.*/
#endif
#ifdef ADC_STOP_SEC_CODE
#undef ADC_STOP_SEC_CODE
#undef MEMMAP_ERROR
/*End of section to be mentioned here*/
#endif
```

For Variables:

```
#ifdef ADC_START_SEC_VAR_UNSPECIFIED
#undef ADC_START_SEC_VAR_UNSPECIFIED
#undef MEMMAP_ERROR
/*Memory Section for Variables to be defined here.*/
#endif
#ifdef ADC_STOP_SEC_VAR_UNSPECIFIED
```

```
#undef ADC_STOP_SEC_VAR_UNSPECIFIED
#undef MEMMAP_ERROR
/*End of section to be mentioned here*/
#endif
```

For Constant data:

```
#ifdef ADC_START_SEC_CONST_UNSPECIFIED
#undef ADC_START_SEC_CONST_UNSPECIFIED
#undef MEMMAP_ERROR
/*Memory Section for Constants to be defined here.*/
#endif
#ifdef ADC_STOP_SEC_CONST_UNSPECIFIED
#undef ADC_STOP_SEC_CONST_UNSPECIFIED
#undef MEMMAP_ERROR
/*End of section to be mentioned here*/
#endif
```

7.2 Linker command file

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

Chapter 8
Configuration parameters considerations

Configuration parameter class for Autosar ShortName driver fall into the following variants as defined in table.

8.1 Configuration parameters considerations

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

Table 8-1. Configuration Parameters

Configuration Container	Configuration Parameters	Configuration Variant	Current Implementation
ADC General			
	AdcDeInitApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcDevErrorDetect	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
	AdcVersionInfoApi	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
	AdcPriorityImplementation	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcEnableQueuing	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcCalibrationApi	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcTimeout	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcConfigSet - AdcDigitalFilterLength	Pre Compile parameter for all Variants of Configuration	Pre Compile
AdcConfigSet - AdcGeneric			
	AdcPriorityQueueMaxDepth	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcMaxGroupChannels	Pre Compile parameter for all Variants of Configuration	Pre Compile
	AdcTransferType	Pre Compile parameter for all Variants of Configuration	Pre Compile
ADCHwUnit- General			
	AdcClockSource	VariantPC or VariantPB	Post Build
	AdcHwUnitId	VariantPC or VariantPB	Post Build
	AdcPrescale	VariantPC or VariantPB	Post Build
	AdcCalibration	VariantPC or VariantPB	Post Build
	AdcExternalMultiplexing	VariantPC or VariantPB	Post Build
ADCChannel- General			
	AdcChannelConvTime	VariantPC or VariantPB	Post Build
	AdcChannelId	VariantPC or VariantPB	Pre Compile
	AdcHwChannel	VariantPC or VariantPB	Post Build
	AdcChannelRefVoltsrcHigh	VariantPC or VariantPB	Post Build
	AdcChannelRefVoltsrcLow	VariantPC or VariantPB	Post Build
	AdcChannelResolution	VariantPC or VariantPB	Post Build
	AdcChannelCalibration	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
	AdcChannelSampTime	VariantPC or VariantPB	Post Build
ADCGroup- General			
	AdcGroupAccessMode	VariantPC or VariantPB	Post Build
	AdcGroupConversionMode	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
	AdcStreamingBufferMode	VariantPC or VariantPB	Post Build
	AdcResultBufferPointer	VariantPC or VariantPB	Post Build
	AdcStreamingNumSamples	VariantPC or VariantPB	Post Build
	AdcHwTrigSrc	VariantPC or VariantPB	Post Build
	AdcGroupISRAlignSamples	VariantPC or VariantPB	Post Build
	AdcGroupFifo	VariantPC or VariantPB	Post Build
	AdcGroupISRAlignSamples	VariantPC or VariantPB	Post Build
NonAutosar			
	AdcChIndexSymNames	VariantPC or VariantPB	Post Build
	AdcReadGroupOptimization	VariantPC or VariantPB	Post Build
AdcPublishedInformation			
	AdcChannelValueSigned	VariantPC or VariantPB	Post Build
	AdcGroupFirstChannelFixed	VariantPC or VariantPB	Post Build
	AdcMaxChannelResolution	VariantPC or VariantPB	Post Build
CommonPublishedInformation			
	VendorId	AdcGroupFirstChannelFixed	Post Build
	ModuleId	VariantPC or VariantPB	Post Build
	ArMajorVersion	VariantPC or VariantPB	Post Build
	ArMinorVersion	VariantPC or VariantPB	Post Build
	ArPatchVersion	VariantPC or VariantPB	Post Build
	SwMajorVersion	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
	SwMinorVersion	VariantPC or VariantPB	Post Build
	SwPatchVersion	VariantPC or VariantPB	Post Build
	VendorApiInFix	VariantPC or VariantPB	Post Build

Chapter 9
Integration Steps

This section gives a brief overview of the steps needed for integrating Driver Full Name Driver :

- Generate the required ShortName configurations. For more details refer to section Files required for Compilation
- Allocate proper memory sections in MemMap.h and linker command file. For more details refer to section Sections to be defined in MemMap.h
- Make sure all include files for compilation are as per section ISR Reference
- Map the ISRs to their vector locations. For more details refer to section ISR to configure within OS – dependencies
- Compile & build the ShortName with all the dependent modules. For more details refer to section Building the Driver & Other AUTOSAR modules - dependencies

Chapter 10
ISR Reference

ISR functions exported by the ShortName 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 ShortName Driver software specification Version 3.0 .

10.1.2 Enum Reference

Enumeration of all constants supported by the driver are as per AUTOSAR ShortName Driver software specification Version 3.0 .

10.1.3 Function Reference

Functions of all functions supported by the driver are as per AUTOSAR ShortName Driver software specification Version 3.0 .

10.1.3.1 Function Adc_Irq_eQADCA_CFIFO0_Empty

This function implements the ISR is called when CFIFO0 is not full on the HW unit0 of eQADC_A.

Figure 10-1. Function Adc_Irq_eQADCA_CFIFO0_Empty References.

Details:

The function implements the ISR for the HW unit0 or unit1 of eQADC_A.

Note

Violates MISRA 2004 Advisory Rule 19.1, only preprocessor statements and comments before 'include' - See `Adc_Irq_c_REF_1`Violates MISRA 2004 Advisory Rule 19.15, repeated include file MemMap.h

- See `Adc_Irq_c_REF_2`

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_CFIFO0_Empty(void);`

Table 10-1. Adc_Irq_eQADCA_CFIFO0_Empty Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.2 Function Adc_Irq_eQADCA_CFIFO1_Empty

This function implements the ISR is called when CFIFO1 is not full on the HW unit0 of eQADC_A.

Figure 10-2. Function Adc_Irq_eQADCA_CFIFO1_Empty References.

Details:

The function implements the ISR for the HW unit0 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_CFIFO1_Empty(void);`

Table 10-2. Adc_Irq_eQADCA_CFIFO1_Empty Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.3 Function Adc_Irq_eQADCA_CFIFO2_Empty

This function implements the ISR is called when CFIFO2 is not full on the HW unit0 of eQADC_A.

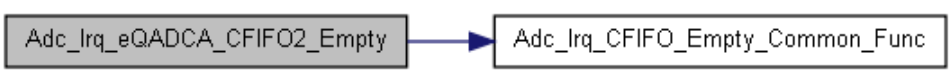

Figure 10-3. Function Adc_Irq_eQADCA_CFIFO2_Empty References.

Details:

The function implements the ISR for the HW unit0 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_CFIFO2_Empty(void);`

Table 10-3. Adc_Irq_eQADCA_CFIFO2_Empty Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.4 Function Adc_Irq_eQADCA_CFIFO3_Empty

This function implements the ISR is called when CFIFO3 is not full on the HW unit1 of eQADC_A.

Figure 10-4. Function Adc_Irq_eQADCA_CFIFO3_Empty References.

Details:

The function implements the ISR for the HW unit1 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_CFIFO3_Empty(void);`

Table 10-4. Adc_Irq_eQADCA_CFIFO3_Empty Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.5 Function Adc_Irq_eQADCA_CFIFO4_Empty

This function implements the ISR is called when CFIFO4 is not full on the HW unit1 of eQADC_A.

Figure 10-5. Function Adc_Irq_eQADCA_CFIFO4_Empty References.

Details:

The function implements the ISR for the HW unit1 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_CFIFO4_Empty(void);`

Table 10-5. Adc_Irq_eQADCA_CFIFO4_Empty Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.6 Function Adc_Irq_eQADCA_CFIFO5_Empty

This function implements the ISR is called when CFIFO5 is not full on the HW unit1 of eQADC_A.

Figure 10-6. Function Adc_Irq_eQADCA_CFIFO5_Empty References.

Details:

The function implements the ISR for the HW unit1 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_CFIFO5_Empty(void);`

Table 10-6. Adc_Irq_eQADCA_CFIFO5_Empty Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.7 Function Adc_Irq_eQADCA_PopResult0

This function implements the ISR for the conversion done from RFIFO0 on the HW unit0 or unit1 of eQADC_A.

Figure 10-7. Function Adc_Irq_eQADCA_PopResult0 References.

Details:

The function implements the ISR for the HW unit0 or unit1 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_PopResult0(void);`

Table 10-7. Adc_Irq_eQADCA_PopResult0 Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.8 Function Adc_Irq_eQADCA_PopResult1

This function implements the ISR for the conversion done from RFIFO1 on the HW unit0 of eQADC_A.

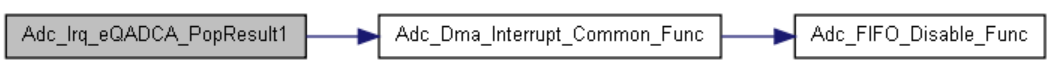

Figure 10-8. Function Adc_Irq_eQADCA_PopResult1 References.

Details:

The function implements the ISR for the HW unit0 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_PopResult1(void);`

Table 10-8. Adc_Irq_eQADCA_PopResult1 Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.9 Function Adc_Irq_eQADCA_PopResult2

This function implements the ISR for the conversion done from RFIFO2 on the HW unit0 of eQADC_A.

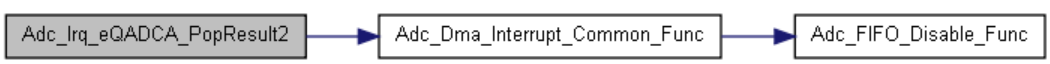

Figure 10-9. Function Adc_Irq_eQADCA_PopResult2 References.

Details:

The function implements the ISR for the HW unit0 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_PopResult2(void);`

Table 10-9. Adc_Irq_eQADCA_PopResult2 Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.10 Function Adc_Irq_eQADCA_PopResult3

This function implements the ISR for the conversion done from RFIFO3 on the HW unit1 eQADC_A.

Figure 10-10. Function Adc_Irq_eQADCA_PopResult3 References.

Details:

The function implements the ISR for the HW unit1 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_PopResult3(void);`

Table 10-10. Adc_Irq_eQADCA_PopResult3 Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.11 Function Adc_Irq_eQADCA_PopResult4

This function implements the ISR for the conversion done from RFIFO4 on the HW unit1 of eQADC_A.

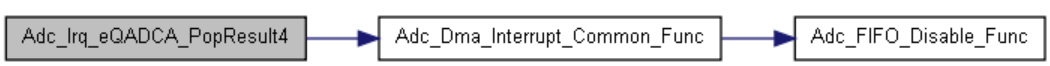

Figure 10-11. Function Adc_Irq_eQADCA_PopResult4 References.

Details:

The function implements the ISR for the HW unit1 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_PopResult4(void);`

Table 10-11. Adc_Irq_eQADCA_PopResult4 Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.12 Function Adc_Irq_eQADCA_PopResult5

This function implements the ISR for the conversion done from RFIFO5 on the HW unit1 of eQADC_A.

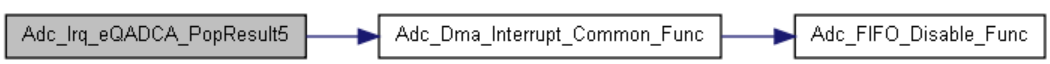

Figure 10-12. Function Adc_Irq_eQADCA_PopResult5 References.

Details:

The function implements the ISR for the HW unit1 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_PopResult5(void);`

Table 10-12. Adc_Irq_eQADCA_PopResult5 Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.13 Function Adc_Irq_eQADCA_RFIFO0_Full

This function implements the ISR is called when RFIFO0 have data for the HW unit0 of eQADC_A.

Figure 10-13. Function Adc_Irq_eQADCA_RFIFO0_Full References.

Details:

The function implements the ISR is called when RFIFO0 have data for the HW unit0 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_RFIFO0_Full(void);`

Table 10-13. Adc_Irq_eQADCA_RFIFO0_Full Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.14 Function Adc_Irq_eQADCA_RFIFO1_Full

This function implements the ISR is called when RFIFO1 have data for the HW unit0 of eQADC_A.

Figure 10-14. Function Adc_Irq_eQADCA_RFIFO1_Full References.

Details:

The function implements the ISR is called when RFIFO1 have data for the HW unit0 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_RFIFO1_Full(void);`

Table 10-14. Adc_Irq_eQADCA_RFIFO1_Full Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.15 Function Adc_Irq_eQADCA_RFIFO2_Full

This function implements the ISR is called when RFIFO2 have data for the HW unit0 of eQADC_A.

Figure 10-15. Function Adc_Irq_eQADCA_RFIFO2_Full References.

Details:

The function implements the ISR is called when RFIFO2 have data for the HW unit0 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_RFIFO2_Full(void);`

Table 10-15. Adc_Irq_eQADCA_RFIFO2_Full Arguments

Type	Name	Direction	Description
	None.	**input**	.

Table continues on the next page...

Table 10-15. Adc_Irq_eQADCA_RFIFO2_Full Arguments (continued)

Type	Name	Direction	Description
	None.	**output**	.
	None.	**input, output**	.

10.1.3.16 Function Adc_Irq_eQADCA_RFIFO3_Full

This function implements the ISR is called when RFIFO3 have data for the HW unit1 of eQADC_A.

Figure 10-16. Function Adc_Irq_eQADCA_RFIFO3_Full References.

Details:

The function implements the ISR is called when RFIFO3 have data for the HW unit1 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_RFIFO3_Full(void);`

Table 10-16. Adc_Irq_eQADCA_RFIFO3_Full Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.17 Function Adc_Irq_eQADCA_RFIFO4_Full

This function implements the ISR is called when RFIFO4 have data for the HW unit1 of eQADC_A.

Figure 10-17. Function Adc_Irq_eQADCA_RFIFO4_Full References.

Details:

The function implements the ISR is called when RFIFO4 have data for the HW unit1 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_RFIFO4_Full(void);`

Table 10-17. Adc_Irq_eQADCA_RFIFO4_Full Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.3.18 Function Adc_Irq_eQADCA_RFIFO5_Full

This function implements the ISR is called when RFIFO5 have data for the HW unit1 of eQADC_A.

Figure 10-18. Function Adc_Irq_eQADCA_RFIFO5_Full References.

Details:

The function implements the ISR is called when RFIFO5 have data for the HW unit1 of eQADC_A.

Return: None.

Pre: None.

Post: None.

Prototype: `void Adc_Irq_eQADCA_RFIFO5_Full(void);`

Table 10-18. Adc_Irq_eQADCA_RFIFO5_Full Arguments

Type	Name	Direction	Description
	None.	**input**	.
	None.	**output**	.
	None.	**input, output**	.

10.1.4 Structs Reference

Data structures supported by the driver are as per AUTOSAR ShortName Driver software specification Version 3.0 .

10.1.5 Types Reference

Types supported by the driver are as per AUTOSAR ShortName Driver software specification Version 3.0 .

10.1.6 Variables Reference

Variables supported by the driver are as per AUTOSAR ShortName Driver software specification Version 3.0 .

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: IM14ADCASR3.0R2.0.0
Rev. 1.2