

User Manual

for S32K1_S32M24x ADC Driver

Document Number: UM2ADCASRR21-11 Rev0000R2.0.0 Rev. 1.0

1 Revision History	2
2 Introduction	3
2.1 Supported Derivatives	3
2.2 Overview	4
2.3 About This Manual	5
2.4 Acronyms and Definitions	6
2.5 Reference List	6
3 Driver	7
3.1 Requirements	7
3.2 Driver Design Summary	7
3.3 Hardware Resources	8
3.4 Deviations from Requirements	8
3.5 Driver Limitations	14
3.6 Driver usage and configuration tips	14
3.6.1 Calibration function	14
3.6.2 DMA Transfer	15
3.6.3 External DMA channel feature	16
3.6.4 Conversion Time Once	17
3.6.5 Set Clock Mode	17
3.6.6 Set Channel Optimization	18
3.6.7 Optimize OneShot HwTrigger Conversions	20
3.6.8 Without Interrupts Group	21
3.6.9 Programmable Delay Block (PDB)	21
3.6.10 PDB in Back to Back Mode	23
3.6.11 ADC Group Uses PDB Channel Delays	24
3.6.12 Configuring the delay between PDB channel conversions	25
3.6.13 Some restrictions of Hardware	25
3.6.14 Optimize DMA Streaming Groups	25
3.7 Runtime errors	27
3.8 Symbolic Names Disclaimer	27
4 Tresos Configuration Plug-in	28
4.1 Module Adc	31
4.2 Container AdcConfigSet	32
4.3 Container AdcHwUnit	32
4.4 Parameter AdcHwUnitId	33
4.5 Parameter AdcLogicalUnitId	33
4.6 Parameter AdcTransferType	34
4.7 Parameter AdcClockSource	34

4.8 Parameter AdcPrescale	35
4.9 Parameter AdcAltPrescale	36
4.10 Parameter AdcCalibrationPrescale	36
4.11 Parameter AdcHwUnitUsrOffset	37
4.12 Parameter AdcHwUnitUsrGain	37
4.13 Parameter AdcHwUnitResolution	38
4.14 Reference AdcDmaChannelId	38
4.15 Reference AdcCountingDmaChannelId	39
4.16 Reference AdcHwUnitEcucPartitionRef	39
4.17 Container AdcNormalConvTimings	40
4.18 Parameter AdcHardwareAverageEnable	40
4.19 Parameter AdcHardwareAverageSelect	41
4.20 Parameter AdcSampleTimeDuration	41
4.21 Container AdcAlternateConvTimings	42
4.22 Parameter AdcHardwareAverageEnableAlternate	42
4.23 Parameter AdcHardwareAverageSelectAlternate	43
4.24 Parameter AdcSampleTimeDurationAlternate	44
4.25 Container PdbHwUnit	44
4.26 Parameter AdcPdbPrescalerDividerSelect	44
4.27 Parameter AdcPdbMultiplicationFactorSelect	45
4.28 Parameter AdcPdbCounterPeriod	46
4.29 Parameter AdcPdbErrorNotification	47
4.30 Parameter PdbInterChnBackToBackEnable	47
4.31 Container AdcChannel	48
4.32 Parameter AdcLogicalChannelId	48
4.33 Parameter AdcChannelName	49
4.34 Parameter AdcChannelId	49
4.35 Parameter AdcChannelConvTime	50
4.36 Parameter AdcChannelLimitCheck	50
4.37 Parameter AdcChannelHighLimit	51
4.38 Parameter AdcChannelLowLimit	51
4.39 Parameter AdcChannelRangeSelect	52
4.40 Parameter AdcChannelRefVoltsrcHigh	52
4.41 Parameter AdcChannelRefVoltsrcLow	53
4.42 Parameter AdcChannelResolution	53
4.43 Parameter AdcChannelSampTime	54
4.44 Container AdcGroup	54
4.45 Parameter AdcGroupAccessMode	55
4.46 Parameter AdcGroupConversionMode	55
4.47 Parameter AdcGroupConversionType	56

4.48 Parameter AdcGroupId	56
4.49 Parameter AdcGroupPriority	57
4.50 Parameter AdcGroupReplacement	57
4.51 Parameter AdcGroupTriggSrc	58
4.52 Parameter AdcHwTrigSignal	58
4.53 Parameter AdcHwTrigTimer	59
4.54 Parameter AdcNotification	59
4.55 Parameter AdcExtraNotification	60
4.56 Parameter AdcStreamingBufferMode	60
4.57 Parameter AdcEnableOptimizeDmaStreamingGroups	61
4.58 Parameter AdcEnableHalfInterrupt	61
4.59 Parameter AdcStreamingNumSamples	62
4.60 Parameter AdcStreamResultGroup	62
4.61 Parameter AdcEnableChDisableChGroup	63
4.62 Parameter AdcWithoutInterrupts	64
4.63 Parameter AdcWithoutDma	64
4.64 Parameter AdcExtDMACHanEnable	65
4.65 Parameter AdcGroupInBacktoBackMode	65
4.66 Parameter AdcGroupUsesChannelDelays	66
4.67 Parameter AdcDelayNextPdb	67
4.68 Parameter AdcPdbPeriodContinuousMode	68
4.69 Parameter AdcChannelDelay	68
4.70 Reference AdcGroupHwTriggerSource	69
4.71 Reference AdcGroupDefinition	69
4.72 Reference AdcGroupEcucPartitionRef	69
4.73 Container AdcGroupConversionConfiguration	70
4.74 Parameter AdcGroupHardwareAverageEnable	70
4.75 Parameter AdcGroupHardwareAverageSelect	71
4.76 Parameter AdcGroupSampleTimeDuration	71
4.77 Container AdcAlternateGroupConvTimings	72
4.78 Parameter AdcGroupAltHardwareAverageEnable	72
4.79 Parameter AdcGroupAltHardwareAverageSelect	73
4.80 Parameter AdcGroupAltSampleTimeDuration	73
4.81 Container AdcHwTrigger	74
4.82 Parameter AdcHwTrigSrc	74
4.83 Container AdcGeneral	75
4.84 Parameter AdcDeInitApi	75
4.85 Parameter AdcDevErrorDetect	76
4.86 Parameter AdcEnableLimitCheck	76
4.87 Parameter AdcEnableQueuing	77

4.88 Parameter AdcPriorityQueueMaxDepth	77
4.89 Parameter AdcEnableStartStopGroupApi	78
4.90 Parameter AdcGrpNotifCapability	78
4.91 Parameter AdcHwTriggerApi	79
4.92 Parameter AdcPriorityImplementation	79
4.93 Parameter AdcReadGroupApi	80
4.94 Parameter AdcResultAlignment	81
4.95 Parameter AdcVersionInfoApi	81
4.96 Parameter AdcLowPowerStatesSupport	82
4.97 Parameter AdcPowerStateAsynchTransitionMode	82
4.98 Reference AdcEcucPartitionRef	83
4.99 Reference AdcKernelEcucPartitionRef	83
4.100 Container AdcPowerStateConfig	83
4.101 Parameter AdcPowerState	84
4.102 Parameter AdcPowerStateReadyCbRef	84
4.103 Container AdcInterrupt	85
4.104 Parameter AdcInterruptSource	85
4.105 Parameter AdcInterruptEnable	86
4.106 Container AdcPublishedInformation	86
4.107 Parameter AdcChannelValueSigned	87
4.108 Parameter AdcGroupFirstChannelFixed	87
4.109 Parameter AdcMaxChannelResolution	88
4.110 Container CommonPublishedInformation	88
4.111 Parameter ArReleaseMajorVersion	88
4.112 Parameter ArReleaseMinorVersion	89
4.113 Parameter ArReleaseRevisionVersion	89
4.114 Parameter ModuleId	90
4.115 Parameter SwMajorVersion	90
4.116 Parameter SwMinorVersion	91
4.117 Parameter SwPatchVersion	91
4.118 Parameter VendorApiInfix	92
4.119 Parameter VendorId	92
4.120 Container AutosarExt	93
4.121 Parameter AdcTimeoutMethod	93
4.122 Parameter AdcTimeoutVal	94
4.123 Parameter AdcIpDevErrorDetect	94
4.124 Parameter PdbIpDevErrorDetect	95
4.125 Parameter AdcMulticoreSupport	95
4.126 Parameter AdcEnableGroupDependentChannelNames	96
4.127 Parameter AdcBypassAbortChainCheck	96

4.128 Parameter AdcConvTimeOnce	97
4.129 Parameter AdcOptimizeOneShotHwTriggerConversions	97
4.130 Parameter AdcOptimizeDmaStreamingGroups	98
4.131 Parameter AdcContinuousWithoutInterrupt	99
4.132 Parameter AdcEnableChDisableChApi	99
4.133 Parameter AdcEnableInitialNotification	99
4.134 Parameter AdcEnableDmaTransferMode	100
4.135 Parameter AdcEnableUserCodeSupport	100
4.136 Parameter AdcEnableSimSupplyMonitor	101
4.137 Parameter AdcEnablePdbInterChannelBackToBackSupport	101
4.138 Parameter AdcEnableSetChannel	102
4.139 Parameter AdcEnableDualClockMode	102
4.140 Parameter AdcEnableCalibration	103
4.141 Parameter AdcEnableReadRawDataApi	103
4.142 Parameter AdcEnableGroupStreamingResultReorder	104
5 Module Index	105
5.1 Software Specification	105
6 Module Documentation	106
6.1 Adc driver	106
6.1.1 Detailed Description	106
6.1.2 Data Structure Documentation	110
6.1.3 Macro Definition Documentation	112
6.1.4 Types Reference	119
6.1.5 Enum Reference	120
6.1.6 Function Reference	124
6.1.7 Variable Documentation	132
6.2 Adc IPL	133
6.2.1 Detailed Description	133
6.2.2 Data Structure Documentation	135
6.2.3 Types Reference	137
6.2.4 Enum Reference	137
6.2.5 Function Reference	142
6.3 Pdb Adc IPL	156
6.3.1 Detailed Description	156
6.3.2 Data Structure Documentation	158
6.3.3 Types Reference	160
6.3.4 Enum Reference	160
6.3.5 Function Reference	162

Chapter 1

Revision History

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

Chapter 2

Introduction

- [Supported Derivatives](#)
- [Overview](#)
- [About This Manual](#)
- [Acronyms and Definitions](#)
- [Reference List](#)

This User Manual describes NXP Semiconductor AUTOSAR ADC for S32K1XX and S32M24X microcontrollers. AUTOSAR ADC driver configuration parameters and deviations from the specification are described in Driver chapter of this document. AUTOSAR ADC driver requirements and APIs are described in the AUTOSAR ADC driver software specification document.

2.1 Supported Derivatives

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

- s32k116_qfn32
- s32k116_lqfp48
- s32k118_lqfp48
- s32k118_lqfp64
- s32k142_lqfp48
- s32k142_lqfp64
- s32k142_lqfp100
- s32k142w_lqfp48
- s32k142w_lqfp64
- s32k144_lqfp48

- s32k144_lqfp64 / MWCT1014S_lqfp64
- s32k144_lqfp100 / MWCT1014S_lqfp100
- s32k144_mapbga100
- s32k144w_lqfp48
- s32k144w_lqfp64
- s32k146_lqfp64
- s32k146_lqfp100 / MWCT1015S_lqfp100
- s32k146_mapbga100 / MWCT1015S_mapbga100
- s32k146_lqfp144
- s32k148_lqfp100
- s32k148_mapbga100 / MWCT1016S_mapbga100
- s32k148_lqfp144
- s32k148_lqfp176
- s32m241_lqfp64
- s32m242_lqfp64
- s32m243_lqfp64
- s32m244_lqfp64

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

2.2 Overview

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

AUTOSAR:

- paves the way for innovative electronic systems that further improve performance, safety and environmental friendliness.
- is a strong global partnership that creates one common standard: "Cooperate on standards, compete on implementation".
- is a key enabling technology to manage the growing electrics/electronics complexity. It aims to be prepared for the upcoming technologies and to improve cost-efficiency without making any compromise with respect to quality.
- facilitates the exchange and update of software and hardware over the service life of the vehicle.

2.3 About This Manual

This Technical Reference employs the following typographical conventions:

- **Boldface** style: Used for important terms, notes and warnings.
- *Italic* style: Used for code snippets in the text. Note that C language modifiers such "const" or "volatile" are sometimes omitted to improve readability of the presented code.

Notes and warnings are shown as below:

Note

This is a note.

Warning

This is a warning

2.4 Acronyms and Definitions

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

2.5 Reference List

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

Chapter 3

Driver

- [Requirements](#)
- [Driver Design Summary](#)
- [Hardware Resources](#)
- [Deviations from Requirements](#)
- [Driver Limitations](#)
- [Driver usage and configuration tips](#)
- [Runtime errors](#)
- [Symbolic Names Disclaimer](#)

3.1 Requirements

Requirements for this driver are detailed in the Autosar Driver Software Specification document (See [Table Reference List](#)).

It has vendor-specific requirements and implementation.

3.2 Driver Design Summary

The ADC Driver initializes and controls the internal Analog to Digital Converter Unit(s) of the microcontroller. It provides services to start and stop a conversion respectively to enable and disable the trigger source for a conversion. Furthermore it provides services to enable and disable a notification mechanism and routines to query the status and result of a conversion. The ADC Driver shall work on so called ADC channels. An ADC channel combines an analog input pin, the needed ADC circuitry itself and a conversion result register into an entity that can be individually controlled and accessed via the ADC Driver. The driver provides a service for Streaming management results and for De-Initialization of circuits.

3.3 Hardware Resources

This device provides two General-purpose ADC :ADC HW Unit 0 (ADC0), ADC HW Unit 1 (ADC1, only available on S32K14X devices). The number of ADC hardware units and channels are derivative specific, so please consult the reference manual.

3.4 Deviations from Requirements

The driver deviates from the AUTOSAR ADC Driver software specification in some places. The table identifies the AUTOSAR requirements that are not fully implemented, implemented differently, not available, not testable or out of scope for the ADC Driver.

Term	Definition
N/S	Out of scope
N/I	Not implemented
N/F	Not fully implemented

Below table identifies the AUTOSAR requirements that are not fully implemented, not implemented or out of scope for the driver.

Requirement	Status	Description	Notes
SWS_Adc_00082	N/F	Service name: - IoHwAb_Adc↔ Notification - Syntax: - void IoHwAb_Adc↔ Notification(void) - Sync/Async: - Synchronous - Reentrancy: - Non Reentrant - Parameters (in): - None - Parameters (inout): - None - Parameters (out): - None - Return value: - None - Description: - Will be called by the ADC Driver when a group conversion is completed for group . - Available via: - IoHwAb_Adc.h	"IoHwAb_AdcNotification<#group↔ ID>" is not fully implemented, because callback notification with this prototype can be configured per group, according with SWS_Adc_00084 and also in agreement with this statement from the 'Specification of ADC Driver AUTOSAR 4.4.0': "In this chapter all interfaces are listed where the target function could be configured. The target function is usually a call-back function. The names of this kind of interfaces are not fixed because they are configurable."

Requirement	Status	Description	Notes
ECUC_Adc_00461	N/F	<p>Name - AdcPowerState -</p> <p>Parent Container - AdcPowerState↵</p> <p>Config -</p> <p>Description - Each instance of this parameter describes a different power state supported by the ADC HW. It should be defined by the HW supplier and used by the ADCDriver to reference specific HW configurations which set the ADC HW module in the referenced power state. At least the power mode corresponding to full power state shall be always configured.</p> <p>-</p> <p>Multiplicity - 1 -</p> <p>Type - EcucIntegerParamDef (Symbolic Name generated for this parameter) -</p> <p>Range - 0 .. 18446744073709551615 -</p> <p>&#0160; -</p> <p>Default value - - -</p> <p>Post-Build Variant Value - false -</p> <p>Value Configuration Class - Pre-compile time - X - All Variants -</p> <p>Link time - - - -</p> <p>Post-build time - - - -</p> <p>Scope / Dependency - scope↵</p> <p>: localdependency: This parameter shall only be configured if the parameter AdcLowPowerStatesSupport is set to true.</p>	<p>This requirement shall be fulfilled to avoid VSMD errors. Asynchronous Power State transition mode doesn't support by hardware.</p> <p>Maximum value of this node is restricted from 18446744073709551615 to 9223372036854775807 due to EB Tresos limitation:</p> <p>"Deviating from the AUTOSAR traceable [TPS_GST_00008], the formula language interpreter of E↵B tresos Studio supports only the signed 64-bit range for integer values, i.e. from 0x8000000000000000 (-9223372036854775808) to 0x7fffffffffffffff (9223372036854775807)."</p> <p>"The formula language parser of EB tresos Studio is implemented in Java. It uses the long primitive type for integer values and therefore can only support the restricted range."</p>

Requirement	Status	Description	Notes
ECUC_Adc_00011	N/F	<p>Name - AdcChannelConvTime - Parent Container - AdcChannel - Description - Configuration of conversion time, i.e. the time during which the analogue value is converted into digital representation, (in clock cycles) for each channel, if supported by hardware.Implementation↵ Type: Adc_ConversionTimeType - Multiplicity - 0..1 - Type - EcucIntegerParamDef - Range - 0 .. 18446744073709551615 - &#0160; - Default value - - - Post-Build Variant Multiplicity - true - Post-Build Variant Value - true - Multiplicity Configuration Class - Pre-compile time - X - VARIANT-PRE-↵ COMPILE - Link time - - - - Post-build time - X - VARIANT-P↵ OST-BUILD - Value Configuration Class - Pre-compile time - X - VARIANT-PRE- COMPILE - Link time - - - - Post-build time - X - VARIANT-P↵ OST-BUILD - Scope / Dependency - scope: local</p>	<p>Maximum value of this node is restricted from 18446744073709551615 to 9223372036854775807 due to EB Tresos limitation: "Deviating from the AUTOSAR traceable [TPS_GST_00008], the formula language interpreter of E↵ B tresos Studio supports only the signed 64-bit range for integer values, i.e. from 0x8000000000000000 (-9223372036854775808) to 0x7fffffffffffffff (9223372036854775807)." "The formula language parser of EB tresos Studio is implemented in Java. It uses the long primitive type for integer values and therefore can only support the restricted range."</p>

Requirement	Status	Description	Notes
ECUC_Adc_00455	N/F	<p>Name - AdcChannelHighLimit -</p> <p>Parent Container - AdcChannel -</p> <p>Description - High limit - used for limit checking. -</p> <p>Multiplicity - 0..1 -</p> <p>Type - EcucIntegerParamDef -</p> <p>Range - 0 .. 18446744073709551615 -</p> <p>&#0160; -</p> <p>Default value - - -</p> <p>Post-Build Variant Multiplicity - false -</p> <p>Post-Build Variant Value - false -</p> <p>Multiplicity Configuration Class -</p> <p>Pre-compile time - X - All Variants -</p> <p>Link time - - - -</p> <p>Post-build time - - - -</p> <p>Value Configuration Class - Pre-compile time - X - All Variants -</p> <p>Link time - - - -</p> <p>Post-build time - - - -</p> <p>Scope / Dependency - scope↵ : localdependency: AdcEnable↵</p> <p>LimitCheck: not available if limit checking is not globally enabled.</p> <p>AdcChannelLimitCheck: not available if channel specific limit check is not enabled.</p> <p>AdcChannelLowLimit↵ : has to be greater or equal than AdcChannelLowLimit.</p>	<p>Maximum value of this node is restricted from 18446744073709551615 to 9223372036854775807 due to EB Tresos limitation:</p> <p>"Deviating from the AUTOSAR traceable [TPS_GST_00008], the formula language interpreter of E↵B tresos Studio supports only the signed 64-bit range for integer values, i.e. from 0x8000000000000000 (-9223372036854775808) to 0x7fffffffffffffff (9223372036854775807)."</p> <p>"The formula language parser of EB tresos Studio is implemented in Java. It uses the long primitive type for integer values and therefore can only support the restricted range."</p>

Requirement	Status	Description	Notes
ECUC_Adc_00454	N/F	<p>Name - AdcChannelLowLimit - Parent Container - AdcChannel - Description - Low limit - used for limit checking. - Multiplicity - 0..1 - Type - EcucIntegerParamDef - Range - 0 .. 18446744073709551615 - &#0160; - Default value - - - Post-Build Variant Multiplicity - false - Post-Build Variant Value - false - Multiplicity Configuration Class - Pre-compile time - X - All Variants - Link time - - - - Post-build time - - - - Value Configuration Class - Pre-compile time - X - All Variants - Link time - - - - Post-build time - - - - Scope / Dependency - scope↵ : localdependency: AdcEnableLimit↵ Check: not available if limit checking is not globally enabled. AdcChannel↵ LimitCheck: not available if channel specific limit check is not enabled. AdcChannelHighLimit: has to be less or equal than AdcChannelHighLimit.</p>	<p>Maximum value of this node is restricted from 18446744073709551615 to 9223372036854775807 due to EB Tresos limitation: "Deviating from the AUTOSAR traceable [TPS_GST_00008], the formula language interpreter of E↵B tresos Studio supports only the signed 64-bit range for integer values, i.e. from 0x8000000000000000 (-9223372036854775808) to 0x7fffffffffffffff (9223372036854775807)." "The formula language parser of EB tresos Studio is implemented in Java. It uses the long primitive type for integer values and therefore can only support the restricted range."</p>

Requirement	Status	Description	Notes
ECUC_Adc_00290	N/F	<p>Name - AdcChannelSampTime - Parent Container - AdcChannel - Description - Configuration of sampling time, i.e. the time during which the value is sampled, (in clock cycles) for each channel, if supported by hardware.Implementation↵ Type: Adc_SamplingTimeType - Multiplicity - 0..1 - Type - EcucIntegerParamDef - Range - 0 .. 18446744073709551615 - &#0160; - Default value - - - Post-Build Variant Multiplicity - true - Post-Build Variant Value - true - Multiplicity Configuration Class - Pre-compile time - X - VARIANT-PRE-↵ COMPILE - Link time - - - - Post-build time - X - VARIANT-P↵ OST-BUILD - Value Configuration Class - Pre-compile time - X - VARIANT-PRE- COMPILE - Link time - - - - Post-build time - X - VARIANT-P↵ OST-BUILD - Scope / Dependency - scope: local</p>	<p>Maximum value of this node is restricted from 18446744073709551615 to 9223372036854775807 due to EB Tresos limitation: "Deviating from the AUTOSAR traceable [TPS_GST_00008], the formula language interpreter of E↵ B tresos Studio supports only the signed 64-bit range for integer values, i.e. from 0x8000000000000000 (-9223372036854775808) to 0x7fffffffffffffff (9223372036854775807)." "The formula language parser of EB tresos Studio is implemented in Java. It uses the long primitive type for integer values and therefore can only support the restricted range."</p>

Requirement	Status	Description	Notes
ECUC_Adc_00401	N/F	Name - AdcHwTrigTimer - Parent Container - AdcGroup - Description - Reload value of the ADC module embedded timer (only if supported by ADC hardware).ImplementationType↵ : Adc_HwTriggerTimerType - Multiplicity - 0..1 - Type - EcucIntegerParamDef - Range - 0 .. 18446744073709551615 - - Default value - - - Post-Build Variant Multiplicity - true - Post-Build Variant Value - true - Multiplicity Configuration Class - Pre-compile time - X - VARIANT-↵ PRE-COMPILE - Link time - - - - Post-build time - X - VARIANT-P↵ OST-BUILD - Value Configuration Class - Pre-compile time - X - VARIANT-PRE-↵ COMPILE - Link time - - - - Post-build time - X - VARIANT-P↵ OST-BUILD - Scope / Dependency - scope↵ : localdependency: AdcTriggSrcHw: Valid only if the group is configured to be triggered by a hardware event.	Maximum value of this node is restricted from 18446744073709551615 to 9223372036854775807 due to EB Tresos limitation: "Deviating from the AUTOSAR traceable [TPS_GST_00008], the formula language interpreter of E↵B tresos Studio supports only the signed 64-bit range for integer values, i.e. from 0x8000000000000000 (-9223372036854775808) to 0x7fffffffffffffff (9223372036854775807)." "The formula language parser of EB tresos Studio is implemented in Java. It uses the long primitive type for integer values and therefore can only support the restricted range."

3.5 Driver Limitations

None.

3.6 Driver usage and configuration tips

3.6.1 Calibration function

ADC driver provides [Adc_Calibrate\(\)](#) API to eliminate the errors between the actual values and the expected converted values, which might be generated by variations in manufacturing and various runtime environmental effects. The API can be called explicitly by the user at any time after the driver was initialized and while there are no ongoing conversions. Calibration is not automatically started by initialization function.

Configuration-time-prerequisites:

- Enable AdcEnableCalibration in AutosarExt container.

Adc Enable Calibration API



3.6.2 DMA Transfer

ADC driver provides the DMA transfer mechanism for transferring the conversion results directly from data registers to system memory via DMA. This is opposed to Interrupt transfer mode, when ADC driver moves the data from registers to user buffer. This feature can be configured independently for each ADC Hardware Unit. If DMA is used, user cannot run Software and Hardware groups at the same time on a unit because the same DMA channel will be used for both and it cannot handle overwriting the results. The buffer which will record conversion results should be declared within NO_CACHEABLE section.

Configuration-time-prerequisites:

- **Adc Global Enable DMA support** needs to be enabled in AutosarExt container:

Adc Global Enable DMA support



- Configure **Adc Transfer Type** as ADC_DMA and select DMA Channel that will service the request from ADC in AdcHwUnit container:

Adc Transfer Type



Select Dma Channel



- Mcl configuration for the logical channel used for ADC must have Adc_Ipw_AdcXDmaTransferComplete← Notification set as interrupt callback (with X defining the ADC instance used, for example Adc_Ipw_Adc0← DmaTransferCompleteNotification()), global config must be enabled. The DMA source must be configured as the ADC instance used (Adc_Ip).

Name DMA_CHANNEL_ADC0

Logic Channel Configuration		Global	Transfer	ScatterGather
Logic Channel Name	DMA_LOGIC_CH_0			
Hardware Instance	DMA_IP_HW_INST_0			
Hardware Channel	DMA_IP_HW_CH_0			
Interrupt Callback	Adc_Ipw_Adc0DmaTransferCompleteNotification			
Error Interrupt Callback	NULL_PTR			
Enable Global Config	<input checked="" type="checkbox"/>		Enable Transfer Config	<input type="checkbox"/>
Enable Scatter/Gather	<input type="checkbox"/>			

Note

- When using **Enable Config Time Support** feature, it is not acceptable to have ADC configured as VariantPreCompile and Mcl configured as VariantPostBuild.
- **Adc Global Enable DMA support** shouldn't be enabled simultaneously with [Adc_SetChannel\(\)](#) because there is a risk that DMA TCD (Transfer Control Descriptor) size might be modified at runtime.
- When using DMA, the results transferred are 16-bit wide which includes D field of ADC Data Result Register.

3.6.3 External DMA channel feature

The ADC driver shall support configuring the DMA channel externally by using a boolean parameter named "↔ External Dma Chan Config" at group configuration level. In this mode, ADC driver will only enable/disable the DMA request for the unit. The user must configure the DMA to transfer data according to his needs.

Configuration-time-prerequisites:

- To use the External DMA channel feature, AdcExtDMACHanEnable needs to be enabled in each group container first:

Adc External Dma Chan Config



Note

- When External Dma Chan Config is enabled, the driver will not configure any TCDs for DMA
- The user still needs to register `Adc_Ipw_Adc0DmaTransferCompleteNotification()` as a notification for DMA, but if External Dma Chan Config is enabled for that group, the function only updates the group status and calls the `GroupNotification`.
- If a group is in Streaming access mode, the code for updating the destination in `Adc_Ipw_Adc0DmaTransferCompleteNotification()` (i.e. update destination address of TCD, enable DMA hardware request) will not be executed for the group so the update should be handled by the user in the notification of that group.
- If this feature is enabled simultaneously with `AdcWithoutInterrupts` ([Without Interrupts Group](#)), the user is not required to register `Adc_Ipw_Adc0DmaTransferCompleteNotification()` as a notification for DMA, because the group status is updated when calling [Adc_ReadGroup\(\)](#). It is required for the application to register a result buffer via [Adc_SetupResultBuffer\(\)](#), but the buffer and DMA transfers are completely managed by the application, so the actual buffer is used only by [Adc_ReadGroup\(\)](#). [Adc_ReadGroup\(\)](#) assumes that the registered buffer is one dimensional, with number of elements equal to the number of channels in the group and that it stores the latest conversion result (latest sample). This assumption is required to determine if latest conversion has occurred before updating the group status. Additionally, for groups configured with Ext DMA and Without interrupt, the [Adc_SetupResultBuffer\(\)](#) resets the buffer to an invalid value so [Adc_ReadGroup\(\)](#) can determine if the result has been transferred and status should be updated. Note that if updating the group status is not required, the user can read directly from the DMA destination buffer without calling [Adc_ReadGroup\(\)](#).

3.6.4 Conversion Time Once

ADC driver provides Conversion Time Once feature to enable/disable configuring conversion time and averaging settings only when initializing the ADC module. If this feature is enabled, these settings will be done only once in [Adc_Init\(\)](#) with values configured per unit, in AdcHwUnit container. If disabled, these settings will be done whenever a group is started, using values from AdcGroup container. When the feature is enabled, the latency of APIs that start group conversions will be reduced. If [Adc_SetClockMode\(\)](#) API is also used, the driver allows the user to configure alternate timing settings (for ALTERNATE mode), both at Unit and Group level.

Configuration-time-prerequisites:

- Enable AdcConvTimeOnce in AutosarExt container

Adc Conversion Time Once



- Configure the values to be used in AdcNormalConvTiming or AdcNormalAlternateTiming (if [Adc_SetClockMode\(\)](#) was invoked before with Alternate mode) in AdcHwUnit container

3.6.5 Set Clock Mode

ADC driver provides an optional API [Adc_SetClockMode\(\)](#) and configuration parameters to compensate the changes of the input clock of the controlled hardware. By enabling this feature, beside the Normal clock configuration set, user can configure an Alternate clock configuration set and can switch between Normal and Alternate at runtime using [Adc_SetClockMode\(\)](#). One clock configuration set includes the ADC parameters affected by clock configuration, such as: ADC Prescale and Sampling Duration. Configuration-time-prerequisites:

- To use the set clock mode feature, **Adc Set Clock Mode API** needs to be enabled in AutosarExt container first:

Adc Set Clock Mode API



- Enable and configure the alternate clock configuration set at HW unit level. If the Conversion Time Once feature is enabled, normal and alternate sampling duration can be configured as well.

Adc Alternate Prescale	UD	2	▼	✎
Adc User Offset (0 -> 255)	UD	0		✎
Adc User Gain (0 -> 1023)	UD	0		✎
Conversion resolution		RESOLUTION_12	▼	↻
▼ AdcNormalConvTimings				
Name		AdcNormalConvTimings		
Adc Hardware Average Enable	UD	<input checked="" type="checkbox"/>	↻	
Adc Hardware Average Select	UD	SAMPLES_8	▼	↻
Adc Sample Time Duration (1 -> 255)	UD	255		↻
▼ AdcAlternateConvTimings				
Name		AdcAlternateConvTimings		
Adc Alternate Hardware Average Enable	UD	<input checked="" type="checkbox"/>	↻	
Adc Alternate Hardware Average Select	UD	SAMPLES_4	▼	✎
Adc Sample Time Duration (1 -> 255)	UD	40		↻

- Enable and configure alternate clock configuration set at group level, if Conversion Time Once feature is disabled.

▼ AdcAlternateGroupConvTimings				
Name		AdcAlternateGroupConvTimings		
Adc Group Alternate Hardware Average Enable	UD	<input checked="" type="checkbox"/>	↻	
Adc Group Alternate Hardware Average Select	UD	SAMPLES_4	▼	✎
Adc Group Alternate Sample Time Duration (1 -> 255)	UD	2		✎

3.6.6 Set Channel Optimization

ADC driver provides an optional configuration parameter that allows the user to change the configuration of a group at runtime. According to Autosar specification, the group definition (list of channels) should be fixed at configuration time; if a different set of channels needs to be converted, another group must be started. However, in some applications stopping the group and starting another takes too long for the given time constraints. For this type of use case, `Adc_SetChannel()` can be used to quickly change the configuration of a group at runtime: the list of channels, and if applicable on the hardware platform - the set of associated conversion timing information for each channel.

Configuration-time-prerequisites:

- The `AdcEnableSetChannel` parameter needs to be enabled in AutosarExt container

Adc Set Channel API



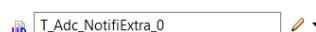
- If channels need to be updated in ISR notification, `AdcEnableInitialNotification` parameter can be enabled in AutosarExt container

Adc Initial Notification Capability



- The initial notification function needs to be enable for each Group where it needs to be used

Adc Group Extra Notification



`Adc_SetChannel()` API can be called by the user application whenever it is needed. In the interrupt routine that handled the conversion complete event, Adc driver will check if `Adc_SetChannel()` has been called and will update the required hardware registers to match the new configuration. The API can be called from the standard Autosar group conversion complete notification, but this notification has the disadvantage that it's called at the very end of ADC interrupt processing, leaving no time to do the updates in ADC hardware registers. So the very next conversion will not be affected by the changes, but the one after that will be.

Usage:

- With `Adc_SetChannel()`, configured group channels can be replaced with new list of channels depending on Channel and NumberOfChannel parameters. Element value of Channel paramter is derived from Logical channel ID.

The screenshot shows the configuration for `AdcHwUnit` with the name `AdcHwUnit_0`. The `AdcChannel` tab is selected, displaying a table of configured channels.

Index	Name	Adc Logical Channel ID	Adc Physical Channel Na...	Adc Physical Channel ID
0	AdcChannel_VREFH	0	VREFH_ADCH29	29
1	AdcChannel_BANDGAP	1	BANDGAP_ADCH27	27
2	AdcChannel_VREFL	2	VREFL_ADCH30	30

E.g: `AdcGroup_0` is configured with 2 Adc channels, when calling `Adc_SetChannel(AdcGroup_0, ChanArr, ChanDelay, ChanMask, ChanNum)`;

The screenshot shows the configuration for `AdcGroup` with the name `AdcGroup_0`. The `AdcGroupDefinition` tab is selected, displaying a table of configured group definitions.

Index	AdcGroupDefinition
0	/Adc/Adc/AdcConfigSet/AdcHwUnit_0/AdcChannel_VREFH
1	/Adc/Adc/AdcConfigSet/AdcHwUnit_0/AdcChannel_BANDGAP

- The group channels can be extended to 5 channels ($\text{ChanArr}[5] = \{2, 0, 1, 1, 0\}$ and $\text{ChanNum} = 5$)
- Or can be reduced to only 1 channel ($\text{ChanArr}[1] = \{2\}$ and $\text{ChanNum} = 1$)
- Or position of channels can be changed and number of channels remains ($\text{ChanArr}[2] = \{1, 0\}$ and $\text{ChanNum} = 2$)

For platforms supporting delays, subset of configured channels can be updated by using channel `ChanMask`. E.g: with `AdcGroup_0`, channel 1 can be changed to another channel without modifying channel 2 (`ChanMask = 0b10`; $\text{ChanArr}[2] = \{2, 1\}$ and $\text{ChanNum} = 2$).

Note

- If the `Adc_SetChannel()` is called before ADC receives the first hardware trigger signal and after calling `Adc_EnableHardwareTrigger()`, the results of this trigger will not be correct since the new list of channels will be updated in the interrupt of first trigger. To start a conversion with the new list, another signal needs to be provided.
- In case of calling `Adc_SetChannel()` after `Adc_StartGroupConversion()`, the list that will be converted depends on total conversion time. If the interrupt event happened before executing `Adc_SetChannel()`, the old list will be converted. Otherwise, if the total conversion time is longer than the time required to execute both functions (`Adc_SetChannel()` and `Adc_StartGroupConversion()`), then the results of the new list will be recorded.
- Sample index will be reset and result data will be overwritten starting from the first sample after runtime channels are updated in ISR.
- The feature `AdcEnableInitialNotification` can be used by the user application to call `Adc_SetChannel()` API before ADC driver updates the hardware configuration for the next conversion.

3.6.7 Optimize OneShot HwTrigger Conversions

ADC driver provides a configuration option to enable maximum speed optimizations for conversion processing of ADC groups configured with `ADC_CONV_MODE_ONESHOT` and `ADC_TRIGG_SRC_HW`. When this parameter is enabled, no other type of ADC group may be configured.

Configuration-time-prerequisites:

- Enable `AdcOptimizeOneShotHwTriggerConversions` in AutosarExt container:

`Adc Optimize OneShot HwTrigger Conversions`



Note

Some restrictions are applied when using this feature:

- Only hardware trigger, single access mode groups could be configured.

3.6.8 Without Interrupts Group

ADC driver provides this feature as a mechanism to reduce the CPU load. The basic functionality of ADC driver requires that an interrupt event occurs after each group conversion, during which the software will copy the data from data registers to the user buffer and update the Group status. If Without Interrupts feature is configured, the conversions can run without software intervention (without any interrupt events generated).

Configuration-time-prerequisites:

- Enable `AdcWithoutInterrupts` for the group using `without interrupt` feature in `AdcGroup` container:

[Adc Group Without Interrupts](#)



For the case of groups configured with `ADC_ACCESS_MODE_SINGLE` the application can retrieve the results by calling [Adc_ReadGroup\(\)](#). The result buffer configured by user via [Adc_SetupResultBuffer\(\)](#) is no longer used to store the results, but must still be registered to avoid raising DET required by AUTOSAR Standard. If HW unit transfer type is configured as `INTERRUPT` (not with DMA), [Adc_ReadGroup\(\)](#) reads the results directly from hardware registers (if the flags indicate that the group conversion was completed) and updates the group state afterwards. If HW unit transfer type is set to `DMA`, the results are transferred via DMA into an internal buffer. [Adc_ReadGroup\(\)](#) reads the results from this buffer and updates the group state afterwards.

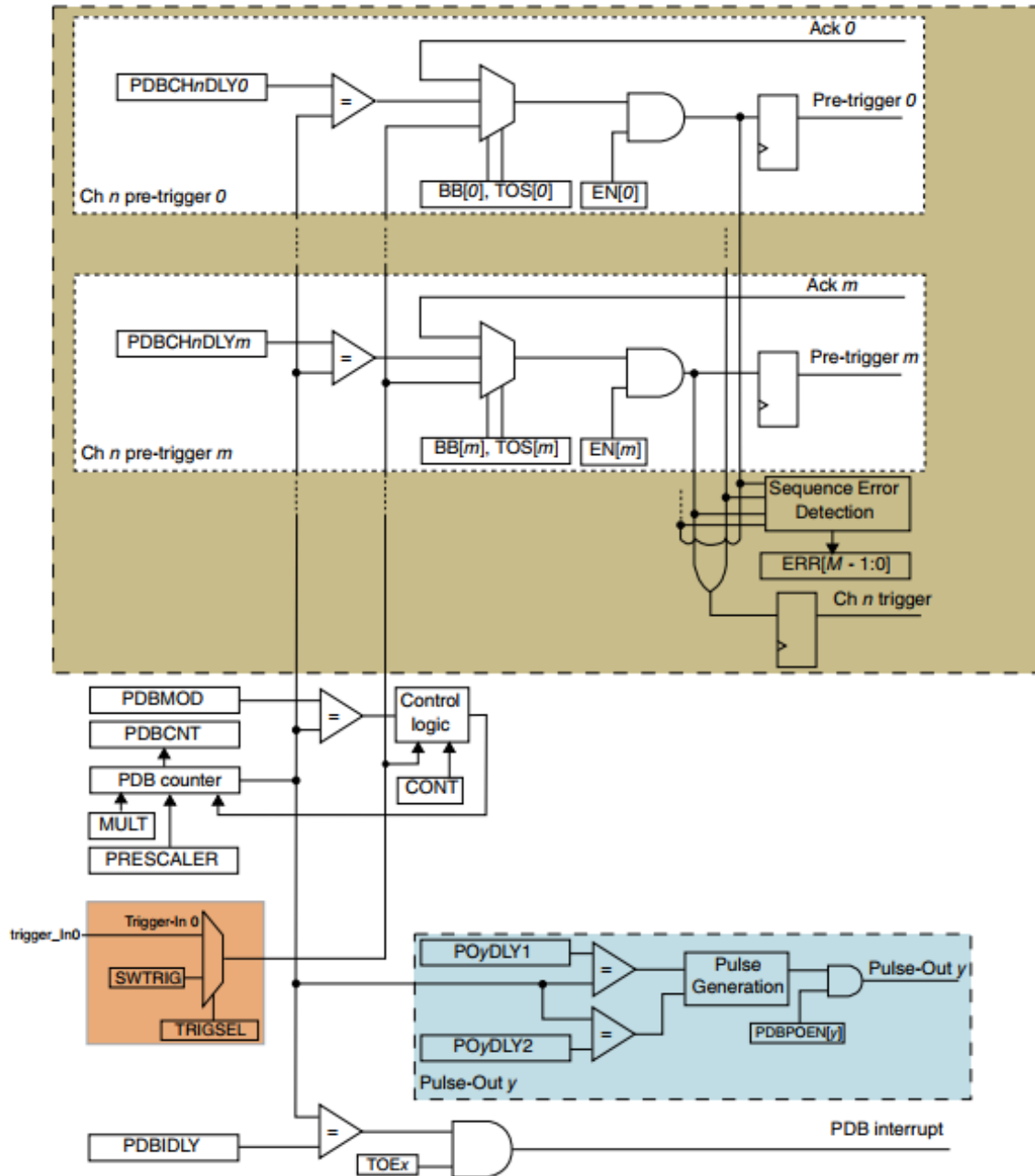
Note

Without Interrupt feature supports only the groups that have number of channels less than maximum SC registers.

3.6.9 Programmable Delay Block (PDB)

The PDB hardware unit is used to initiate individual channel conversions that are required for each group conversion. Each PDB unit consists of one or more PDB Channels, each with 8 PDB Pretriggers. Each PDB pretrigger will control the timing of one ADC channel conversion.


The major components of PDB unit are illustrated in below diagram, for more details please check the associated chapter in Reference Manual:







The PDB pretriggers can work in either Back to Back Mode or with individual Delays for channel conversions. This can be configured for each ADC group using the following fields: *Adc Group In Back to Back Mode*, *Adc Group Uses Channel Delays* and *Adc Delay Next PDB*.



Some general settings of PDB Unit can be configured by the user in ADC driver: clock prescaler settings, auto-clearing of PDB internal errors, user notification for PDB internal errors.



PdbHwUnit

Name  PdbHwUnit

Adc Pdb Prescaler Divider Select (0 -> 7)  0 

Adc Pdb Multiplication Factor Select for Prescaler (0 -> 3)  0 

Adc Pdb Counter Period (0 -> 65535)  65535 


Adc Pdb Sequence Error Notification  NULL_PTR 

Adc Pdb Prescaler Divider Select and **Adc Pdb Multiplication Factor Select for Prescaler** can be used to fine tune the frequency at which PDB counter works. These values will be written into PRESCALER and MULT bitfields of PDB_SC register during initialization. The PDB counter uses the peripheral clock divided by the product of a factor (selected by MULT field) and an integer factor (set by PRESCALER field) like in this formula (peripheral clock)/(MULT x PRESCALER). For more details about the implementation of these bit fields in the hardware please check the Reference Manual.

Setting **Adc Pdb Sequence Error Notification** is used to configure call back function for the PDB sequence error interrupt. The sequence error flags can be set as result of various invalid scenarios involving the PDB, most usual being that one pretrigger becomes asserted and requests a conversion to ADC unit while another conversion is still ongoing. When one of these flags is set, the PDB unit will be stuck and not request any conversions until the flags are cleared. The flags will be cleared by PDB sequence error interrupt (if enabled), and an optional notification function will be called to signal this event to the user.



The PDB is used together with ADC hardware unit for all group conversions (HW triggered or SW triggered). There is a dedicated sub-container in every group for configuring the delays associated with each channel:



General **AdcChannelDelay** AdcGroupDefinition AdcGroupEcucPartitionRef


 **AdcChannelDelay**


Index	Adc Channel Delay
0	0
1	50
2	100



3.6.10 PDB in Back to Back Mode

Adc External Dma Chan Config  ☐ 

Adc Group Uses Channel Delays  ☐ 

Adc Delay Next PDB (0 -> 65535)  0

Adc Pdb Period For Continuous Mode (0 -> 65535)  0

Adc Group In Back to Back Mode ☒  

When this parameter is enabled, the PDB pretriggers will be configured to work in back to back mode: PDB_CH1n_C1[BB] bits are set for all the group channels, except the first one. The first channel conversion complete will trigger the PDB channel's second pretrigger, and so on until the last channel of the group. This ensures that the channel conversions occur in sequence, as quickly as possible and without causing internal PDB errors.

The ADC driver can convert only up to N channels at once, where N is number of SC1 registers available. If the groups have more channels than that, ADC driver will split the group in sub-groups of N and convert them with sub-group conversions. Interrupt processing is needed between the sub-groups for register re-configuration. The N channels are converted using one or more PDB channels (each controlling 8 pretriggers), but the different channels do not work in back to back mode together, so a delay is needed between the PDB channels, otherwise PDB errors will occur. ADC driver will configure this delay value when Back to back mode is used, with the value defined by the user in **Adc Delay Next PDB** parameter. The sequences of 8 channels associated to a PDB channel will happen in back to back mode.

Note

- If **Adc Group In Back to Back Mode** is not enabled for a group, all pretriggers of the PDB channels will be triggered at the same time causing internal PDB errors, unless the coherence of channel conversions is ensured via the delay configured for each pretrigger. This is why at least one of **Adc Group In Back to Back Mode** and **Adc Group Uses Channel Delays** options needs to be configured for an ADC Group. If both parameters are configured for the ADC Group, the user can configure a delay only for the first channel conversion, and the rest will be converted in back to back mode.
- If **Adc Group Uses Channel Delays** is not enabled for the group, the first channel will have an associated delay of 0, and will convert immediately after the external trigger is received.
- If a hardware unit has only groups running in Back to Back Mode that have more than 8 channels, **Pdb Enable Inter-channel Back to Back Mode** from PdbHwUnit can be enabled to also have the PDB channels, in addition to pretriggers, running in Back to Back Mode. This ensures that each channel will start its pretrigger chain as quickly as possible and without causing internal PDB errors. This is available only on S32K14xW devices.

3.6.11 ADC Group Uses PDB Channel Delays

Adc Group Uses Channel Delays



When this parameter is enabled, the PDB pretriggers will have individual delays configured. PDB_CH1nDLYx will be set with the values configured in AdcChannelDelay list for each channel. The values must not be equal or too close, otherwise PDB internal errors will occur.

The ADC driver can convert only up to N channels at once, where N is number of SC1 registers available. If the groups have more channels than that, ADC driver will split the group in sub-groups of N and convert them with sub-group conversions. Interrupt processing is needed between the sub-groups for register re-configuration. The timing of the channel conversions will be accurate only for the first N channels; for the next ones, on top of the configured delay, the delay of previous conversions and interrupt processing will be implicitly added. For this reason, it's safe to configure a delay of 0 for the N+1 channel. But the delay values for each group of N need to be increasing, so that the channel conversions occur in the order they were defined in the group.

If **Adc Group In Back to Back Mode** is also configured for the ADC Group, the user can configure a delay only for the first channel conversion, and the rest will be converted in [Back to Back mode](#). The delay value defined by the user in **Adc Delay Next PDB** parameter will be used to avoid PDB internal errors.

3.6.12 Configuring the delay between PDB channel conversions

The PDB unit can have one or more PDB Channels of 8 pretriggers each that will be used together for setting up conversions of large groups. The PDB Channels will be started together when the PDB unit is triggered, thus risking to request conversions at the same time and causing internal errors. For this reason, the value of **Adc Delay Next PDB** is needed to be added as delay between PDB channels when Back to Back mode is used. The ADC driver supported to configure this value at group configuration level as below:

Adc Delay Next PDB (0 -> 65535)



Note

The **Adc Delay Next PDB** value should be greater than the timing to convert 8 channels.

3.6.13 Some restrictions of Hardware

- Hardware triggered groups and software triggered groups cannot function in parallel
- The Result Buffers will be overflowed if PDB trigger period between each channel is too short. Minimum required period depends on each derivative. It is recommended to use the default delay period or an [ADC group in back to back mode](#).

3.6.14 Optimize DMA Streaming Groups

The ADC driver provides an optional configuration parameter for reducing the number of DMA interrupts required for processing the conversions of Adc Groups which are configured as `ADC_ACCESS_MODE_STREAMING`. Instead of having an interrupt after every end of chain group conversion, only 2 interrupts will be raised for a stream of N conversions – one after N/2 conversions, and one after all N conversions are completed.

- In this feature, if group has more than one channel, streaming DMA is required. In that case, driver will use both transferring DMA and streaming DMA (`AdcDmaChannelId` and `AdcStreamingDmaChannelId`). Streaming DMA acts as a hardware counter which has `CITER/BITER` value equals to group sample as configured by `AdcStreamingNumSamples` parameter. Each single ADC conversion completion will trigger a request to `AdcDmaChannelId`. Then this channel will copy 16-bit data from ADC Data Register (`Rn`) to the user buffers. When all ADC channels are completed, this channel will start DMA Channel Linking (`AdcStreamingDmaChannelId` - Major Elink Channel). The DMA's `CITER` of the Streaming DMA Channel will be decreased by 1 and also transfer a dummy data, that means one sample has been completed. When all ADC samples are completed, the streaming DMA channel will finish its major loop. Both interrupts occurring after half of conversions are done and after all of them are done will be raised by the streaming DMA channel.
- If group doesn't have this feature enabled or it has enabled but has only one channel, driver will only use transferring DMA.

Configuration-time-prerequisites:

- Enable both `AdcEnableDmaTransferMode` and `AdcOptimizeDmaStreamingGroups`:

Adc Optimize DMA Streaming Groups



Adc Enable/Disable Channels API



Adc Global Enable DMA support



Driver

- Configure `AdcTransferType` as `ADC_DMA` type and select transferring DMA (and also streaming DMA if there is at least one ADC group which enables optimize DMA streaming groups with more than one ADC channel).

Adc Transfer Type	<input type="text" value="ADC_DMA"/>
Select Dma Channel	<input type="text" value="/Mcl/Mcl/MclConfig/CHANNEL_FOR_ADC_0"/>
Select Adc Streaming Dma Channel	<input type="text" value="/Mcl/Mcl/MclConfig/COUNTING_CHANNEL_FOR_ADC_0"/>

- Note that transferring (and streaming DMA) need to have Interrupt Callback (e.g: `Adc_Ipw_Adc0DmaTransferCompleteNotification()`) and enable DMA MUX source (e.g: `DMA_IP_REQ_MUX0_ADC0`) equivalent to the ADC hardware unit is being used.

<p>Name <input type="text" value="DMA_CHANNEL_ADC0"/></p> <p>Logic Channel Configuration Global Transfer ScatterGather</p> <p>Logic Channel Name <input type="text" value="DMA_LOGIC_CH_0"/></p> <p>Hardware Instance <input type="text" value="DMA_IP_HW_INST_0"/></p> <p>Hardware Channel <input type="text" value="DMA_IP_HW_CH_0"/></p> <p>Interrupt Callback <input type="text" value="Adc_Ipw_Adc0DmaTransferCompleteNotification"/></p> <p>Error Interrupt Callback <input type="text" value="NULL_PTR"/></p> <p>Enable Global Config <input checked="" type="checkbox"/> </p> <p>Enable Transfer Config <input type="checkbox"/> </p> <p>Enable Scatter/Gather <input type="checkbox"/> </p>	<p>Name <input type="text" value="DMA_CHANNEL_ADC0"/></p> <p>Logic Channel Configuration Global Transfer ScatterGather</p> <p>dmaLogicChannel_GlobalConfigType</p> <p>Name <input type="text" value="dmaLogicChannel_GlobalConfigType"/></p> <p>Request</p> <p>Name <input type="text" value="dmaLogicChannelConfig_GlobalRequestType"/></p> <p>Enable DMAMUX Trigger <input type="checkbox"/> </p> <p>Enable DMAMUX Source <input checked="" type="checkbox"/> </p> <p>DMAMUX Source <input type="text" value="DMA_IP_REQ_MUX0_ADC0"/></p> <p>Enable DMA Request <input type="checkbox"/> </p>
<p>Name <input type="text" value="COUNTING_DMA_CHANNEL_ADC0"/></p> <p>Logic Channel Configuration Global Transfer ScatterGather</p> <p>Logic Channel Name <input type="text" value="DMA_LOGIC_CH_2"/></p> <p>Hardware Instance <input type="text" value="DMA_IP_HW_INST_0"/></p> <p>Hardware Channel <input type="text" value="DMA_IP_HW_CH_2"/></p> <p>Interrupt Callback <input type="text" value="Adc_Ipw_Adc0DmaTransferCompleteNotification"/></p> <p>Error Interrupt Callback <input type="text" value="NULL_PTR"/></p> <p>Enable Global Config <input checked="" type="checkbox"/> </p> <p>Enable Transfer Config <input type="checkbox"/> </p> <p>Enable Scatter/Gather <input type="checkbox"/> </p>	<p>Name <input type="text" value="COUNTING_DMA_CHANNEL_ADC0"/></p> <p>Logic Channel Configuration Global Transfer ScatterGather</p> <p>dmaLogicChannel_GlobalConfigType</p> <p>Name <input type="text" value="dmaLogicChannel_GlobalConfigType"/></p> <p>Request</p> <p>Name <input type="text" value="dmaLogicChannelConfig_GlobalRequestType"/></p> <p>Enable DMAMUX Trigger <input type="checkbox"/> </p> <p>Enable DMAMUX Source <input type="checkbox"/> </p> <p>DMAMUX Source <input type="text" value="DMA_IP_REQ_MUX0_DISABLED"/></p> <p>Enable DMA Request <input type="checkbox"/> </p>

- Global channel linking also need to be enabled for this feature

Name

Instance

Logic Instance Name

Hardware Instance

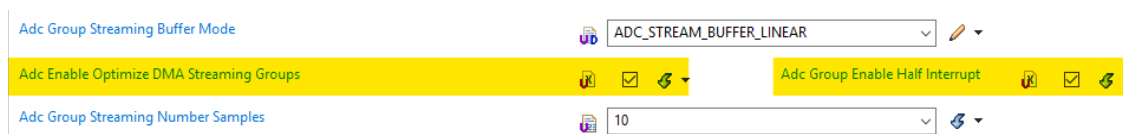
Debug ☐

Halt After Error ☐

Round Robin Channel Arbitration ☐

Global Channel linking ☒

- At group level, `AdcEnableOptimizeDmaStreamingGroups` (and `AdcEnableHalfInterrupt` if half interrupt is used) need to be enabled



Note

- `Adc_SetChannel()` and Optimize one-shot hardware trigger cannot be use concurrently with this feature.
- This feature is supported only for the groups which have number of channels less than total number of available SC registers.

3.7 Runtime errors

The driver generates the following DET runtime errors at runtime.

Function	Error Code	Condition triggering the error
Adc_StartGroupConversion()	ADC_E_TIMEOUT	Timed out. Ongoing conversion could not be aborted.
Adc_StopGroupConversion()	ADC_E_TIMEOUT	Timed out. Ongoing conversion could not be aborted.
Adc_Calibrate()	ADC_E_TIMEOUT	Timed out. Calibration operation timed out.

3.8 Symbolic Names Disclaimer

All containers having `symbolicNameValue` set to `TRUE` in the AUTOSAR schema will generate defines like:

```
#define <Mip>Conf_<Container_ShortName>_<Container_ID>
```

For this reason it is forbidden to duplicate the names of such containers across the RTD configurations or to use names that may trigger other compile issues (e.g. match existing `#ifdefs` arguments).

Chapter 4

Tresos Configuration Plug-in

This chapter describes the Tresos configuration plug-in for the driver. All the parameters are described below.

- Module [Adc](#)
 - Container [AdcConfigSet](#)
 - * Container [AdcHwUnit](#)
 - Parameter [AdcHwUnitId](#)
 - Parameter [AdcLogicalUnitId](#)
 - Parameter [AdcTransferType](#)
 - Parameter [AdcClockSource](#)
 - Parameter [AdcPrescale](#)
 - Parameter [AdcAltPrescale](#)
 - Parameter [AdcCalibrationPrescale](#)
 - Parameter [AdcHwUnitUsrOffset](#)
 - Parameter [AdcHwUnitUsrGain](#)
 - Parameter [AdcHwUnitResolution](#)
 - Reference [AdcDmaChannelId](#)
 - Reference [AdcCountingDmaChannelId](#)
 - Reference [AdcHwUnitEcucPartitionRef](#)
 - Container [AdcNormalConvTimings](#)
 - Parameter [AdcHardwareAverageEnable](#)
 - Parameter [AdcHardwareAverageSelect](#)
 - Parameter [AdcSampleTimeDuration](#)
 - Container [AdcAlternateConvTimings](#)
 - Parameter [AdcHardwareAverageEnableAlternate](#)
 - Parameter [AdcHardwareAverageSelectAlternate](#)
 - Parameter [AdcSampleTimeDurationAlternate](#)
 - Container [PdbHwUnit](#)
 - Parameter [AdcPdbPrescalerDividerSelect](#)
 - Parameter [AdcPdbMultiplicationFactorSelect](#)
 - Parameter [AdcPdbCounterPeriod](#)
 - Parameter [AdcPdbErrorNotification](#)
 - Parameter [PdbInterChnBackToBackEnable](#)

- Container [AdcChannel](#)
- Parameter [AdcLogicalChannelId](#)
- Parameter [AdcChannelName](#)
- Parameter [AdcChannelId](#)
- Parameter [AdcChannelConvTime](#)
- Parameter [AdcChannelLimitCheck](#)
- Parameter [AdcChannelHighLimit](#)
- Parameter [AdcChannelLowLimit](#)
- Parameter [AdcChannelRangeSelect](#)
- Parameter [AdcChannelRefVoltsrcHigh](#)
- Parameter [AdcChannelRefVoltsrcLow](#)
- Parameter [AdcChannelResolution](#)
- Parameter [AdcChannelSampTime](#)
- Container [AdcGroup](#)
- Parameter [AdcGroupAccessMode](#)
- Parameter [AdcGroupConversionMode](#)
- Parameter [AdcGroupConversionType](#)
- Parameter [AdcGroupId](#)
- Parameter [AdcGroupPriority](#)
- Parameter [AdcGroupReplacement](#)
- Parameter [AdcGroupTriggSrc](#)
- Parameter [AdcHwTrigSignal](#)
- Parameter [AdcHwTrigTimer](#)
- Parameter [AdcNotification](#)
- Parameter [AdcExtraNotification](#)
- Parameter [AdcStreamingBufferMode](#)
- Parameter [AdcEnableOptimizeDmaStreamingGroups](#)
- Parameter [AdcEnableHalfInterrupt](#)
- Parameter [AdcStreamingNumSamples](#)
- Parameter [AdcStreamResultGroup](#)
- Parameter [AdcEnableChDisableChGroup](#)
- Parameter [AdcWithoutInterrupts](#)
- Parameter [AdcWithoutDma](#)
- Parameter [AdcExtDMAChanEnable](#)
- Parameter [AdcGroupInBacktoBackMode](#)
- Parameter [AdcGroupUsesChannelDelays](#)
- Parameter [AdcDelayNextPdb](#)
- Parameter [AdcPdbPeriodContinuousMode](#)
- Parameter [AdcChannelDelay](#)
- Reference [AdcGroupHwTriggerSource](#)
- Reference [AdcGroupDefinition](#)
- Reference [AdcGroupEcucPartitionRef](#)
- Container [AdcGroupConversionConfiguration](#)
- Parameter [AdcGroupHardwareAverageEnable](#)
- Parameter [AdcGroupHardwareAverageSelect](#)
- Parameter [AdcGroupSampleTimeDuration](#)
- Container [AdcAlternateGroupConvTimings](#)

- Parameter [AdcGroupAltHardwareAverageEnable](#)
 - Parameter [AdcGroupAltHardwareAverageSelect](#)
 - Parameter [AdcGroupAltSampleTimeDuration](#)
- * Container [AdcHwTrigger](#)
 - Parameter [AdcHwTrigSrc](#)
- Container [AdcGeneral](#)
 - * Parameter [AdcDeInitApi](#)
 - * Parameter [AdcDevErrorDetect](#)
 - * Parameter [AdcEnableLimitCheck](#)
 - * Parameter [AdcEnableQueuing](#)
 - * Parameter [AdcPriorityQueueMaxDepth](#)
 - * Parameter [AdcEnableStartStopGroupApi](#)
 - * Parameter [AdcGrpNotifCapability](#)
 - * Parameter [AdcHwTriggerApi](#)
 - * Parameter [AdcPriorityImplementation](#)
 - * Parameter [AdcReadGroupApi](#)
 - * Parameter [AdcResultAlignment](#)
 - * Parameter [AdcVersionInfoApi](#)
 - * Parameter [AdcLowPowerStatesSupport](#)
 - * Parameter [AdcPowerStateAsynchTransitionMode](#)
 - * Reference [AdcEcucPartitionRef](#)
 - * Reference [AdcKernelEcucPartitionRef](#)
 - * Container [AdcPowerStateConfig](#)
 - Parameter [AdcPowerState](#)
 - Parameter [AdcPowerStateReadyCbkJRef](#)
- Container [AdcInterrupt](#)
 - * Parameter [AdcInterruptSource](#)
 - * Parameter [AdcInterruptEnable](#)
- Container [AdcPublishedInformation](#)
 - * Parameter [AdcChannelValueSigned](#)
 - * Parameter [AdcGroupFirstChannelFixed](#)
 - * Parameter [AdcMaxChannelResolution](#)
- Container [CommonPublishedInformation](#)
 - * Parameter [ArReleaseMajorVersion](#)
 - * Parameter [ArReleaseMinorVersion](#)
 - * Parameter [ArReleaseRevisionVersion](#)
 - * Parameter [ModuleId](#)
 - * Parameter [SwMajorVersion](#)
 - * Parameter [SwMinorVersion](#)
 - * Parameter [SwPatchVersion](#)
 - * Parameter [VendorApiInfix](#)

- * Parameter [VendorId](#)
- Container [AutosarExt](#)
 - * Parameter [AdcTimeoutMethod](#)
 - * Parameter [AdcTimeoutVal](#)
 - * Parameter [AdcIpDevErrorDetect](#)
 - * Parameter [PdbIpDevErrorDetect](#)
 - * Parameter [AdcMulticoreSupport](#)
 - * Parameter [AdcEnableGroupDependentChannelNames](#)
 - * Parameter [AdcBypassAbortChainCheck](#)
 - * Parameter [AdcConvTimeOnce](#)
 - * Parameter [AdcOptimizeOneShotHwTriggerConversions](#)
 - * Parameter [AdcOptimizeDmaStreamingGroups](#)
 - * Parameter [AdcContinuousWithoutInterrupt](#)
 - * Parameter [AdcEnableChDisableChApi](#)
 - * Parameter [AdcEnableInitialNotification](#)
 - * Parameter [AdcEnableDmaTransferMode](#)
 - * Parameter [AdcEnableUserModeSupport](#)
 - * Parameter [AdcEnableSimSupplyMonitor](#)
 - * Parameter [AdcEnablePdbInterChannelBackToBackSupport](#)
 - * Parameter [AdcEnableSetChannel](#)
 - * Parameter [AdcEnableDualClockMode](#)
 - * Parameter [AdcEnableCalibration](#)
 - * Parameter [AdcEnableReadRawDataApi](#)
 - * Parameter [AdcEnableGroupStreamingResultReorder](#)

4.1 Module Adc

Configuration of the Adc (Analog Digital Conversion) module.

Included containers:

- [AdcConfigSet](#)
- [AdcGeneral](#)
- [AdcInterrupt](#)
- [AdcPublishedInformation](#)
- [CommonPublishedInformation](#)
- [AutosarExt](#)

Property	Value
type	ECUC-MODULE-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantSupport	true
supportedConfigVariants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

4.2 Container AdcConfigSet

This container contains the configuration parameters and sub containers of the AUTOSAR Adc module.

Included subcontainers:

- [AdcHwUnit](#)
- [AdcHwTrigger](#)

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.3 Container AdcHwUnit

This container contains the Driver configuration (parameters) depending on grouping of channels. This container could contain HW specific parameters which are not defined in the Standardized Module Definition. They must be added in the Vendor Specific Module Definition.

Included subcontainers:

- [AdcNormalConvTimings](#)
- [AdcAlternateConvTimings](#)
- [PdbHwUnit](#)
- [AdcChannel](#)
- [AdcGroup](#)

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.4 Parameter AdcHwUnitId

Numeric ID of the HW Unit. This symbolic name allows accessing Hw Unit data. Enumeration literals are defined vendor specific.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC0
literals	['ADC0', 'ADC1']

4.5 Parameter AdcLogicalUnitId

Specifies the Logical id of the Hardware Unit.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false

Property	Value
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	1
min	0

4.6 Parameter AdcTransferType

Select the Interrupt or Dma transfer Type. If DMA is used, user must not run SW and HW groups at the same time on the same HW unit because the same DMA channel will be used for both.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_INTERRUPT
literals	['ADC_INTERRUPT', 'ADC_DMA']

4.7 Parameter AdcClockSource

The ADC module specific clock input for the conversion unit can statically be configured to select different clock sources if provided by hardware. Enumeration literals are defined vendor specific.

ADC_ALTCLK1 - Alternate clock 1.

ADC_ALTCLK2 - Alternate clock 2.

ADC_ALTCLK3 - Alternate clock 3.

ADC_ALTCLK4 - Alternate clock 4.

.

Note: Only ADC_ALTCLK1 is used by the current implementation.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_ALTCLK1
literals	['ADC_ALTCLK1', 'ADC_ALTCLK2', 'ADC_ALTCLK3', 'ADC_ALTCLK4']

4.8 Parameter AdcPrescale

Optional ADC module specific clock prescale factor, if supported by hardware.

ImplementationType: Adc_PrescaleType.

The Prescaler value for NORMAL mode. Only the following are allowed:

- 1: ADC clock frequency is equal to bus clock.
- 2: ADC clock frequency is half of bus clock.
- 4: ADC clock frequency is quarter of bus clock.
- 8: ADC clock frequency is eighth of bus clock.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	8
min	1

4.9 Parameter AdcAltPrescale

The Prescaler value for ALTERNATE mode. This feature can be configured if Adc Set Clock Mode API from General/AutosarExt is enabled. Only the following are allowed:

- 1: ADC clock frequency is equal to bus clock.
- 2: ADC clock frequency is half of bus clock.
- 4: ADC clock frequency is quarter of bus clock.
- 8: ADC clock frequency is eighth of bus clock.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2
max	8
min	1

4.10 Parameter AdcCalibrationPrescale

Specifies the used clock input for calibration. This feature can be configured if Adc Enable Calibration API from General/AutosarExt is enabled.

- 1: ADC clock frequency is equal to bus clock.
- 2: ADC clock frequency is half of bus clock.
- 4: ADC clock frequency is quarter of bus clock.
- 8: ADC clock frequency is eighth of bus clock.

IMPORTANT NOTE: The clock for calibration must be less than or equal to half of maximum specified frequency (50Mhz) and higher than minimum frequency (2Mhz).

Please refer to Datasheet for more details.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	8
max	8
min	1

4.11 Parameter AdcHwUnitUsrOffset

Set user defined offset.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	255
min	0

4.12 Parameter AdcHwUnitUsrGain

Set user defined gain.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	4
max	1023
min	0

4.13 Parameter AdcHwUnitResolution

Select the number of significant bits per conversion data.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	RESOLUTION_12
literals	['RESOLUTION_12', 'RESOLUTION_10', 'RESOLUTION_8']

4.14 Reference AdcDmaChannelId

ID of the DMA channel used to transfer the data.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcuDefs/Mcl/MclConfig/dmaLogicChannel_Type

4.15 Reference AdcCountingDmaChannelId

Configurable only when Transfer Type is DMA and at least one group of hw unit has:

AdcEnableOptimizeDmaStreamingGroups enabled with more than one ADC channel

OR has selected Without Interrupts, ACCESS_MODE_STREAMING and Group Streaming Results Reorder and number of channels > 1

OR has selected Without Interrupts, ACCESS_MODE_STREAMING and a single channel - for this case, AutosarExt Adc Enable Group Streaming Results Reorder feature must be enabled, but can be disabled at group level.

Linked to DMA channel by ADC driver, to count the number of samples converted in streaming mode

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcuDefs/Mcl/MclConfig/dmaLogicChannel_Type

4.16 Reference AdcHwUnitEcucPartitionRef

Maps a ADC hardware unit to zero or one ECUC partition to limit the access to this hardware unit. The ECUC

partitions referenced are a subset of the ECUC partitions where the ADC driver is mapped to. This parameter is disabled, there is no multicore support implemented for this platform.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcuDefs/EcuC/EcuPartitionCollection/EcuPartition

4.17 Container AdcNormalConvTimings

Selects Normal values used for programming clock frequency, conversion timing and hardware averaging used at initialization and also by Adc_SetClockMode API when it is called with parameter ADC_NORMAL.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.18 Parameter AdcHardwareAverageEnable

Enables the hardware average function of the ADC.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.19 Parameter AdcHardwareAverageSelect

Determines how many ADC conversions will be averaged to create the ADC average result. This functionality is activated when ADCx_MCR[AVGEN] = 1.

SAMPLES_4 - 4 samples averaged.

SAMPLES_8 - 8 samples averaged.

SAMPLES_16 - 16 samples averaged.

SAMPLES_32 - 32 samples averaged.

.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	SAMPLES_4
literals	['SAMPLES_4', 'SAMPLES_8', 'SAMPLES_16', 'SAMPLES_32']

4.20 Parameter AdcSampleTimeDuration

Input a value 1 to 255 means selecting a sample time of 2 to 256 ADCK clock cycles. The value written to this

register is the desired sample time minus 1.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	12
max	255
min	1

4.21 Container AdcAlternateConvTimings

Selects Alternate values used in Adc_SetClockMode API for programming CTR Conversion Timing Registers. This container is available only when AdcEnableDualClockMode has been enabled. Hardware averaging functionality is also available for configuration

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.22 Parameter AdcHardwareAverageEnableAlternate

Alternate configuration: Enables the hardware average function of the ADC.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.23 Parameter AdcHardwareAverageSelectAlternate

Alternate configuration: Determines how many ADC conversions will be averaged to create the ADC average result. This functionality is activated when `ADCx_MCR[AVGEN] = 1`.

SAMPLES_4 - 4 samples averaged.

SAMPLES_8 - 8 samples averaged.

SAMPLES_16 - 16 samples averaged.

SAMPLES_32 - 32 samples averaged.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	SAMPLES_4
literals	['SAMPLES_4', 'SAMPLES_8', 'SAMPLES_16', 'SAMPLES_32']

4.24 Parameter AdcSampleTimeDurationAlternate

Alternate configuration: Input a value 1 to 255 means selecting a sample time of 2 to 256 ADCK clock cycles. The value written to this register is the desired sample time minus 1.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	12
max	255
min	1

4.25 Container PdbHwUnit

This container contains the configuration of the PDB unit.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.26 Parameter AdcPdbPrescalerDividerSelect

Determines the prescaler used for the PDB counter.

0 - Counting uses the peripheral clock divided by multiplication factor selected by AdcPdbMultiplicationFactorSelect.

- 1 - Counting uses the peripheral clock divided by twice of the multiplication factor selected by `AdcPdbMultiplicationFactorSelect`.
- 2 - Counting uses the peripheral clock divided by four of the multiplication factor selected by `AdcPdbMultiplicationFactorSelect`.
- 3 - Counting uses the peripheral clock divided by eight of the multiplication factor selected by `AdcPdbMultiplicationFactorSelect`.
- 4 - Counting uses the peripheral clock divided by 16 times of the multiplication factor selected by `AdcPdbMultiplicationFactorSelect`.
- 5 - Counting uses the peripheral clock divided by 32 times of the multiplication factor selected by `AdcPdbMultiplicationFactorSelect`.
- 6 - Counting uses the peripheral clock divided by 64 times of the multiplication factor selected by `AdcPdbMultiplicationFactorSelect`.
- 7 - Counting uses the peripheral clock divided by 128 times of the multiplication factor selected by `AdcPdbMultiplicationFactorSelect`.

.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	7
min	0

4.27 Parameter `AdcPdbMultiplicationFactorSelect`

Selects the multiplication factor of the prescaler divider for the counter clock of the associated PDB unit.

- 0 - Multiplication factor is 1.
- 1 - Multiplication factor is 10.
- 2 - Multiplication factor is 20.
- 3 - Multiplication factor is 40.

.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	3
min	0

4.28 Parameter AdcPdbCounterPeriod

Specifies the period of the counter. When

the counter reaches this value, it will be reset back to zero. If

the PDB is in Continuous mode, the count begins anew. Reading this

field returns the value of the internal register that is effective

for the current cycle of PDB.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	65535
max	65535
min	0

4.29 Parameter AdcPdbErrorNotification

Callback function for PDB channel sequence error. This function pointer is called everytime when PDB channel sequence error occurred. Setting this field to a non-null value will also enable the interrupt.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.30 Parameter PdbInterChnBackToBackEnable

Enable the inter-channel back to back mode between PDBx CH0 and PDBx CH1 pre-triggers, forming a ring (PDBx_CH0_pretrigger7 -> PDBx_CH1_pretrigger0 and PDBx_CH1_pretrigger7 -> PDBx_CH0_pretrigger0).

NOTE: this configuration is dedicated to each PDB instance.

If this node is enabled, AdcDelayNextPdb is not required in all group configured with back to back mode and number of channels is greater than 8

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.31 Container AdcChannel

This container contains the channel configuration (parameters) depending on the hardware capability.

The organization of this data structure could contain dependencies to the microcontroller so this is left up to the implementer and its location is left up to the configuration.

Note: Since a AdcChannel can be part of several AdcGroups, this container is not realized as a subcontainer of AdcGroup but instead as a subcontainer of AdcHwUnit.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.32 Parameter AdcLogicalChannelId

This is the logical Id of the ADC channel.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	1024
min	0

4.33 Parameter AdcChannelName

This parameter defines the assignment of the channel to the physical ADC hardware channel. Note: - Range of the ADC Channels depends on the selected package.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	SE2_ADCH2
literals	['SE2_ADCH2', 'SE3_ADCH3', 'SE6_ADCH6', 'SE7_ADCH7', 'SE8_ADCH8', 'SE9_ADCH9', 'SE10_ADCH10', 'SE11_ADCH11', 'SE12_ADCH12', 'SE13_ADCH13', 'SE14_ADCH14', 'VDD_INT0_ADCH21', 'VDDA_INT0_ADCH21', 'VREFH_INT0_ADCH21', 'VDD_3V_INT0_ADCH21', 'VDD_FLAS←H_3V_INT0_ADCH21', 'VDD_LV_INT0_ADCH21', 'BANDGAP_ADCH27', 'VREFH_ADCH29', 'VREFL_ADCH30']

4.34 Parameter AdcChannelId

This parameter defines the assignment of the channel to the physical ADC hardware channel. Note: - Range of the ADC Channels depends on the selected package.

IMPORTANT NOTE: This node must be in sync with 'Adc Physical Channel Name': must be equal with number specified after 'ADCH' in the selected 'Adc Physical Channel Name'. E.g. INT_CH_0_ADCH21 => Channel ID must be 21. The node is required by Autosar standard. 'Adc Physical Channel Name' node is added because refers to names from ReferenceManual and is more user friendly.

In Tresos configurator, after selecting a new Channel Name, the new Channel Id value can be filled in automatically by using 'Calculate Value' button.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	10
max	1024
min	0

4.35 Parameter AdcChannelConvTime

Configuration of conversion time, i.e. the time during which the analogue value is converted into digital representation, (in clock cycles) for each channel, if supported by hardware.

ImplementationType: Adc_ConversionTimeType.

This parameter is not used by the current implementation.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	9223372036854775807
min	0

4.36 Parameter AdcChannelLimitCheck

Enables or disables limit checking for an ADC channel.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.37 Parameter AdcChannelHighLimit

High limit - used for limit checking.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	255
max	9223372036854775807
min	0

4.38 Parameter AdcChannelLowLimit

Low limit - used for limit checking.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false

Property	Value
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	9223372036854775807
min	0

4.39 Parameter AdcChannelRangeSelect

In case of active limit checking: defines which conversion values are taken into account related to the borders defined with AdcChannelLowLimit and AdcChannelHighLimit.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_RANGE_ALWAYS
literals	['ADC_RANGE_ALWAYS', 'ADC_RANGE_BETWEEN', 'ADC_RANGE↵_NOT_BETWEEN', 'ADC_RANGE_NOT_OVER_HIGH', 'ADC_RANG↵E_NOT_UNDER_LOW', 'ADC_RANGE_OVER_HIGH', 'ADC_RANGE↵_UNDER_LOW']

4.40 Parameter AdcChannelRefVoltsrcHigh

Upper reference voltage source for each channel. Enumeration literals are defined vendor specific.

This parameter is not used by the current implementation.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC

Property	Value
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	UPPER_REF_VOLT_0
literals	['UPPER_REF_VOLT_0']

4.41 Parameter AdcChannelRefVoltsrcLow

Lower reference voltage source for each channel. Enumeration literals are defined vendor specific.

This parameter is not used by the current implementation.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	LOWER_REF_VOLT_0
literals	['LOWER_REF_VOLT_0']

4.42 Parameter AdcChannelResolution

Channel Resolution in bits of converted value. This value is not used. The resolution can be modified at hardware unit level.

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	12
max	63
min	1

4.43 Parameter AdcChannelSampTime

Sampling time, i.e. the time during which the value is sampled, (in clock cycles) for each channel. Not used.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	22
max	9223372036854775807
min	0

4.44 Container AdcGroup

This container contains the Group configuration (parameters).

Included subcontainers:

- [AdcGroupConversionConfiguration](#)
- [AdcAlternateGroupConvTimings](#)

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.45 Parameter AdcGroupAccessMode

Type of access mode to group conversion results.

ImplementationType: Adc_GroupAccessModeType.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_ACCESS_MODE_SINGLE
literals	['ADC_ACCESS_MODE_SINGLE', 'ADC_ACCESS_MODE_STREAMING']

4.46 Parameter AdcGroupConversionMode

Type of conversion mode supported by the driver.

ImplementationType: Adc_GroupConvModeType.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_CONV_MODE_ONESHOT
literals	['ADC_CONV_MODE_ONESHOT', 'ADC_CONV_MODE_CONTINUOUS']

4.47 Parameter AdcGroupConversionType

Normal or Injected conversion type.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_CONV_TYPE_NORMAL
literals	['ADC_CONV_TYPE_NORMAL']

4.48 Parameter AdcGroupId

Numeric ID of the group. This parameter is the symbolic name to be used on the API. This symbolic name allows accessing Channel Group data. This value will be assigned to the symbolic name derived of the AdcGroup container shortName.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

Property	Value
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	1023
min	0

4.49 Parameter AdcGroupPriority

Priority level of the AdcGroup. This item is ignored if Adc/AdcGeneral/AdcPriorityImplementation is defined to ADC_PRIORITY_NONE.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	255
min	0

4.50 Parameter AdcGroupReplacement

Replacement mechanism, which is used on ADC group level, if a group conversion is interrupted by a group which has a higher priority.

ImplementationType: Adc_GroupReplacementType

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_GROUP_REPL_ABORT_RESTART
literals	['ADC_GROUP_REPL_ABORT_RESTART', 'ADC_GROUP_REPL_SU← SPEND_RESUME']

4.51 Parameter AdcGroupTriggSrc

Type of source event that starts a group conversion. It's possible select Hw or Sw trigger.

In case of Hw trigger the trigger source can be from the PDB. Only trigger sources from PDB is used by the current implementation. The trigger source connected to PDB can be configured from Mcl module.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_TRIGG_SRC_SW
literals	['ADC_TRIGG_SRC_HW', 'ADC_TRIGG_SRC_SW']

4.52 Parameter AdcHwTrigSignal

Configures the edge of the hardware trigger signal, i.e. to start the conversion. This parameter is not used by the current implementation.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false

Property	Value
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_HW_TRIG_RISING_EDGE
literals	['ADC_HW_TRIG_BOTH_EDGES', 'ADC_HW_TRIG_RISING_EDGE', 'ADC_HW_TRIG_FALLING_EDGE']

4.53 Parameter AdcHwTrigTimer

Reload value of the ADC module embedded timer. This parameter is not used by the current implementation.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	9223372036854775807
min	0

4.54 Parameter AdcNotification

Callback function for each group. This function pointer is called whenever the conversion of this group is completed.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false

Property	Value
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.55 Parameter AdcExtraNotification

Extra callback function for each group. This function pointer will be called at the beginning of the interrupt routine, before updating any HW registers or Group status.

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	NULL_PTR

4.56 Parameter AdcStreamingBufferMode

Select the streaming buffer as linear buffer (i.e. the ADC Driver stops the conversion as soon as the stream buffer is full) or as circular buffer (wraps around if the end of the stream buffer is reached).

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_STREAM_BUFFER_LINEAR
literals	['ADC_STREAM_BUFFER_LINEAR', 'ADC_STREAM_BUFFER_CIRCULAR']

4.57 Parameter AdcEnableOptimizeDmaStreamingGroups

Enable/Disable The Adc driver enable Optimize DMA streaming groups for reducing the number of interrupts required for processing the conversions of Adc Groups that consist of one or more channels (depending on HW capabilities) and which are configured as ADC_ACCESS_MODE_STREAMING.

When this feature is enabled, only one interrupt will be raised after the completion of all stream conversions (as configured by AdcStreamingNumSamples parameter). An additional interrupt to be raised after half of the stream is converted shall also be configurable.

This feature is enabled if Adc Global Enable DMA Transfer from General/AutosarExt is also enabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.58 Parameter AdcEnableHalfInterrupt

Enable/ Disable the interrupt when half sample complete for optimize DMA streaming groups feature. AutosarExt/AdcOptimizeDmaStreamingGroups must be enable for configure this feature.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.59 Parameter AdcStreamingNumSamples

Number of ADC values to be acquired per channel in streaming access mode.

Note: in single access mode this parameter assumes value 1, since only one sample per channel is processed.

ImplementationType: Adc_StreamNumSampleType.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	255
min	1

4.60 Parameter AdcStreamResultGroup

Arrange the ADC results as multiple sets of group result buffer.

E.g: for a group with channels {CH1 CH5 CH7} the resulting stream buffer shall be:

{ CH1, CH5, CH7, CH1, CH5, CH7, CH1, CH5, CH7 }

instead of

{ CH1, CH1, CH1, CH5, CH5, CH5, CH7, CH7, CH7} like supported by AUTOSAR standard.

This Parameter can be configured only for groups configured with ADC_ACCESS_MODE_STREAMING Access Mode

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.61 Parameter AdcEnableChDisableChGroup

If this parameter is enabled, it allows the feature of enabling or disabling a particular channel in the group.

Max.no of Groups with this feature enabled, should be configured are 254 if the configuration parameter AdcEnableChDisableChApi is enabled in Autosar Extension container

Note This is an Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.62 Parameter `AdcWithoutInterrupts`

Enable/ Disable the occurring of ADC Interrupts and Reading of the group conversion results periodically without interrupts (ADC interrupt or DMA notification).

1. When this parameter is enabled, interrupts are disabled. The conversion will run without software intervention (no interrupt generated anymore) and the application can read the results by calling `Adc_ReadGroup()`.
2. If Transfer Type is `ADC_INTERRUPT` (or `ADC_DMA`) and this one is enabled, the result buffer registered via `Adc_SetupResultBuffer` will no longer be used populated with results. `Adc_ReadGroup` will read the results directly from ADC HW registers (or Internal Dma Buffer).

When this parameter is Disabled, normal functionality shall be executed.

Note: This is an Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.63 Parameter `AdcWithoutDma`

When true, disables completely DMA configuration done by ADC driver for the group, to not affect other groups for which the ADC driver has already configured the DMA. It is intended to be used when DMA Transfer Mode is selected for the ADC unit, for groups required to work without interrupt and without DMA.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true

Property	Value
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.64 Parameter AdcExtDMAChanEnable

Enable/ Disable DMA functionality of individual group.

In this mode the adc will only enable the dma request for the last channel in the group and, so that the dma transfer will start after the group conversion has finished.

In this mode it is the users responsibility to configure the dma to transfer the data according to his needs

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.65 Parameter AdcGroupInBacktoBackMode

Enable/ Disable the channel conversions occuring in Back to Back mode.

The PDB hardware unit is used to initiate individual channel conversions that are required for each group conversion, with one PDB pretrigger controlling the timing of each ADC channel conversion.

When AdcGroupInBacktoBackMode is enabled, the PDB pretriggers will be configured to work in back to back mode: PDB_CH1n_C1[BB] bits are set for all the group channels, except the first one.

The first channel conversion complete will trigger the PDB channel's second pretrigger, and so on until the last channel of the group. This ensures that the channel conversions occur in sequence, as quickly as possible and without causing internal PDB errors.

The Adc driver can convert only up to N channels at once; for groups larger than N channels, Adc driver will split the group in sub-groups of N and convert them with sub-group conversions. Interrupt processing is needed between

the sub-groups for register re-configuration. The 16 channels are converted using 2 PDB channels (each controlling 8 pretriggers), but the 2 channels do not work in back to back mode together, so a delay is needed for the 2nd PDB channel, otherwise PDB errors will occur. Adc driver will configure this delay value when Back to back mode is used, with the value defined by the user in `AdcDelayNextPdb` parameter. The sequences of 8 channels associated to a PDB channel will happen in back to back mode.

If `AdcGroupInBacktoBackMode` is not enabled for a group, all pretriggers of the PDB channels will be triggered at the same time causing internal PDB errors, unless the coherence of channel conversions is ensured via the delay configured for each pretrigger. This is why at least one of `AdcGroupInBacktoBackMode` and `AdcGroupUsesChannelDelays` options needs to be configured for an Adc Group.

If both parameters are configured for the Adc Group, the user can configure a delay only for the first channel conversion, and the rest will be converted in back to back mode.

If `AdcGroupUsesChannelDelays` is not configured for the group, the first channel will have an associated delay of 0, and will convert immediately after the external trigger is received.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.66 Parameter `AdcGroupUsesChannelDelays`

Enable/ Disable the usage of channel delays in PDB pretriggers.

The PDB hardware unit is used to initiate individual channel conversions that are required for each group conversion, with one PDB pretrigger controlling the timing of each ADC channel conversion.

When `AdcGroupUsesChannelDelays` is enabled, the PDB pretriggers will have individual delays configured. `PDB_CH1nDLYx` will be set with the values configured in `AdcChannelDelay` list for each channel. The values must not be equal or too close, otherwise PDB internal errors will occur.

The Adc driver can convert only up to N channels at once, $N = 16/24/32$ channels - depending on derivative (refer to Reference Manual for details). For groups larger than N channels, Adc driver will split the group in sub-groups of N and convert them with sub-group conversions. Interrupt processing is needed between the sub-groups for register re-configuration. The timing of the channel conversions will be accurate only for the first 16 channels; for the next ones, the on top of the configured delay, the delay of previous conversions and interrupt processing will be implicitly added. For this reason, it's safe to configure a delay of 0 for the $Nth + 1$ channel. But the delay values for each group of 16 need to be increasing, so that the channel conversions occur in the order they were defined in the group.

If `AdcGroupInBacktoBackMode` is also configured for the Adc Group, the user can configure a delay only for the first channel conversion, and the rest will be converted in back to back mode (see description of `AdcGroupInBacktoBackMode`). The delay value defined by the user in `AdcDelayNextPdb` parameter will be used for 2nd PDB channel to avoid PDB internal errors.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.67 Parameter `AdcDelayNextPdb`

The PDB unit can have one or more PDB Channels of 8 pretriggers each that will be used together for setting up conversions of large groups. The PDB Channels will be started together when the PDB unit is triggered, thus risking to request conversions at the same time and causing internal errors. For this reason, the value of `AdcDelayNextPdb` is needed to be added as delay between PDB channels when Back to Back mode is used. The `AdcDelayNextPdb` value should be greater than the timing to convert 8 channels. The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by `CFG1[ADICLK]`, and the divide ratio is specified by `CFG1[ADIV]`. To calculate total conversion time for one channel the following formula is applied: ADC TOTAL CONVERSION TIME = Sample Phase Time (set by `SMPLTS+1`) + Hold Phase (1 ADC Cycle) + Compare Phase Time (8-bit Mode=20 ADC Cycles, 10-bit Mode=24 ADC Cycles, 12-bit Mode=28 ADC Cycles) + Single or First continuous time adder (5 ADC cycles + 5 bus clock cycles).

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	65535
min	0

4.68 Parameter AdcPdbPeriodContinuousMode

The period of PBD module in continuous mode. This is specific and only used for continuous group without interrupt. The PDB period should be big enough to ensure all of the channels complete the conversion before PDB restarts

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1000
max	65535
min	0

4.69 Parameter AdcChannelDelay

Delay of the associated PDB pretrigger for this channel. If Back to Back mode is not used for this group, the user must ensure the difference between the delays of different channels are big enough to accomodate the Adc conversions without PDB channel sequence errors. If Back to Back mode is used for this group, only the delay for the first channel will be considered - the rest of the channels will be converted in back to back mode. The delay value have to be a increasing numbers for each sequence of 16/24/32 channels - depending on derivative (refer to Reference Manual for details).

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	65535
min	0

4.70 Reference AdcGroupHwTriggerSource

Select the HW trigger signal for triggering the conversion group. Configurable only when HW trigger source is selected for the group.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D2M20I0R0/Adc/AdcConfigSet/AdcHwTrigger

4.71 Reference AdcGroupDefinition

Assignment of channels to a AdcGroups. For each AdcChannel that should belong to the group, a reference needs to be defined.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Adc/AdcConfigSet/AdcHwUnit/AdcChannel

4.72 Reference AdcGroupEcucPartitionRef

Maps an ADC channel group to zero or multiple ECUC partitions to limit the access to this channel group. The ECUC partitions referenced are a subset of the ECUC partitions where the ADC driver is mapped to.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcuDefs/EcuC/EcuPartitionCollection/EcuPartition

4.73 Container AdcGroupConversionConfiguration

Configure the Sampling and Conversion TimeGroup.

Hardware averaging functionality is also available for configuration

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.74 Parameter AdcGroupHardwareAverageEnable

Enables the hardware average function of the ADC.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.75 Parameter AdcGroupHardwareAverageSelect

Determines how many ADC conversions will be averaged to create the ADC average result. This functionality is activated when `ADCx_MCR[AVGEN] = 1`.

SAMPLES_4 - 4 samples averaged.

SAMPLES_8 - 8 samples averaged.

SAMPLES_16 - 16 samples averaged.

SAMPLES_32 - 32 samples averaged.

.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	SAMPLES_4
literals	['SAMPLES_4', 'SAMPLES_8', 'SAMPLES_16', 'SAMPLES_32']

4.76 Parameter AdcGroupSampleTimeDuration

Input a value 1 to 255 means selecting a sample time of 2 to 256 ADCK clock cycles. The value written to this register is the desired sample time minus 1.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2
max	255
min	1

4.77 Container AdcAlternateGroupConvTimings

Selects Alternate values used for programming CTR Conversion Timing Registers in Adc_SetClockMode API. This is available when AdcEnableDualClockMode has been enabled and AdcConvTimeOnce has been disable. Hardware averaging functionality is also available for configuration

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.78 Parameter AdcGroupAltHardwareAverageEnable

Enables the alternate hardware average function of the ADC.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.79 Parameter AdcGroupAltHardwareAverageSelect

Selects the Group Alternate Hardware Average to determine how many ADC conversions will be averaged to create the ADC average result. This functionality is activated when `ADCx_MCR[AVGEN] = 1`.

SAMPLES_4 - 4 samples averaged.

SAMPLES_8 - 8 samples averaged.

SAMPLES_16 - 16 samples averaged.

SAMPLES_32 - 32 samples averaged.

.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	SAMPLES_4
literals	['SAMPLES_4', 'SAMPLES_8', 'SAMPLES_16', 'SAMPLES_32']

4.80 Parameter AdcGroupAltSampleTimeDuration

Selects a alternate sample time of 2 to 256 ADCK clock cycles. The alternate value written to this register is the desired sample time minus 1.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2
max	255
min	1

4.81 Container AdcHwTrigger

This container contains the Hardware trigger source configured for the group. Editable only if at least one group has HW trigger source selected.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.82 Parameter AdcHwTrigSrc

On this implementation the HW triggers available are from PDB.

(Note: This is an Implementation Specific Parameter.)

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	TRIGGER_IN0
literals	['TRIGGER_IN0']

4.83 Container AdcGeneral

General configuration (parameters) of the ADC Driver software module.

Included subcontainers:

- [AdcPowerStateConfig](#)

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.84 Parameter AdcDeInitApi

Adds/removes the service `Adc_DeInit()` from the code.

true: `Adc_DeInit()` can be used.

false: `Adc_DeInit()` can not be used.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.85 Parameter AdcDevErrorDetect

Switches the development error detection and notification on or off.

true: detection and notification is enabled.

false: detection and notification is disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.86 Parameter AdcEnableLimitCheck

Enables or disables limit checking feature in the ADC driver.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.87 Parameter AdcEnableQueuing

Determines, if the queuing mechanism is active in case of priority mechanism disabled.

Note: If priority mechanism is enabled, queuing mechanism is always active and the parameter ADC_ENABLE_QUEUING is not evaluated.

true: Enabled.

false: Disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.88 Parameter AdcPriorityQueueMaxDepth

Maximum depth of queue used for queuing of incoming conversion requests when hardware unit is busy.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	1
max	1024
min	1

4.89 Parameter AdcEnableStartStopGroupApi

Adds / removes the services `Adc_StartGroupConversion()` and `Adc_StopGroupConversion()` from the code.

true: `Adc_StartGroupConversion()` and `Adc_StopGroupConversion()` can be used.

false: `Adc_StartGroupConversion()` and `Adc_StopGroupConversion()` can not be used.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.90 Parameter AdcGrpNotifCapability

Determines, if the group notification mechanism (the functions to enable and disable the notifications) is available at runtime.

true: Enabled.

false: Disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	true

4.91 Parameter AdcHwTriggerApi

Adds / removes the services `Adc_EnableHardwareTrigger()` and `Adc_DisableHardwareTrigger()` from the code.

true: `Adc_EnableHardwareTrigger()` and `Adc_DisableHardwareTrigger()` can be used.

false: `Adc_EnableHardwareTrigger()` and `Adc_DisableHardwareTrigger()` can not be used.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.92 Parameter AdcPriorityImplementation

Determines whether a priority mechanism is available for prioritization of the conversion requests and if available, the type of prioritization mechanism. The selection applies for groups with trigger source software and trigger source hardware. Two types of prioritization mechanism can be selected.

The hardware prioritization mechanism (`AdcPriorityHw`) uses the ADC hardware features for prioritization of the software conversion requests and hardware trigger signals for groups with trigger source hardware.

Tresos Configuration Plug-in

The mixed hardware and software prioritization mechanism (AdcPriorityHwSw) uses the ADC hardware features for prioritization of ADC hardware trigger for groups with trigger source hardware and a software implemented prioritization mechanism for groups with trigger source software.

The group priorities for software triggered groups are typically configured with lower priority levels than the group priorities for hardware triggered groups.

ImplementationType: Adc_PriorityImplementationType.

Note: In this version the ADC_PRIORITY_HW isn't used.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_PRIORITY_NONE
literals	['ADC_PRIORITY_HW', 'ADC_PRIORITY_HW_SW', 'ADC_PRIORITY_↵_NONE']

4.93 Parameter AdcReadGroupApi

Adds / removes the service Adc_ReadGroup() from the code.

true: Adc_ReadGroup() can be used.

false: Adc_ReadGroup() can not be used.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.94 Parameter AdcResultAlignment

Alignment of ADC raw results in ADC result buffer (left/right alignment).

Implementation Type: `Adc_ResultAlignmentType`.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_ALIGN_RIGHT
literals	['ADC_ALIGN_RIGHT', 'ADC_ALIGN_LEFT']

4.95 Parameter AdcVersionInfoApi

Adds / removes the service `Adc_GetVersionInfo()` from the code.

true: `Adc_GetVersionInfo()` can be used.

false: `Adc_GetVersionInfo()` can not be used.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

4.96 Parameter AdcLowPowerStatesSupport

Adds / removes all power state management related APIs (ADC_SetPowerState, ADC_GetCurrentPowerState, ADC_GetTargetPowerState, ADC_PreparePowerState, ADC_Main_PowerTransitionManager), indicating if the HW offers low power state management.

NOTE: K1 platform does not support low power modes for ADC so this feature is always disabled.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.97 Parameter AdcPowerStateAsynchTransitionMode

Enables / disables support of the ADC Driver to the asynchronous power state transition. This feature is not implemented on this platform.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.98 Reference AdcEcucPartitionRef

Maps the ADC driver to zero or multiple ECUC partitions to make the driver API available in the according partition.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.99 Reference AdcKernelEcucPartitionRef

Maps the ADC kernel to zero or one ECUC partition to assign the driver kernel to a certain core. The ECUC partition referenced is a subset of the ECUC partitions where the ADC driver is mapped to.

Property	Value
type	ECUC-REFERENCE-DEF
origin	AUTOSAR_ECUC
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

4.100 Container AdcPowerStateConfig

Each instance of this parameter defines a power state and the callback to be called when this power state is reached.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

4.101 Parameter AdcPowerState

Each instance of this parameter describes a different power state

supported by the ADC HW. It should be defined by the HW supplier and

used by the ADCDriver to reference specific HW configurations which set

the ADC HW module in the referenced power state. At least the power mode

corresponding to full power state shall be always configured.

This parameter shall only be configured if the parameter AdcLowPowerStatesSupport is set to true.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	true
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	0
max	9223372036854775807
min	0

4.102 Parameter AdcPowerStateReadyCbkJRef

Each instance of this parameter contains a reference to a power mode callback defined in a CDD or IoHwAbs component.

This parameter shall only be configured if the parameter `AdcLowPowerStatesSupport` is set to true

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	NULL_PTR

4.103 Container `AdcInterrupt`

Selects whether the interrupt for each ADC Unit will be enabled. These settings are used for optimizing the code size by removing the interrupt handling code for interrupts that are not needed. .

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE

4.104 Parameter `AdcInterruptSource`

The name of the interrupt.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP

Property	Value
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	ADC0_COCO
literals	['ADC0_COCO', 'ADC1_COCO']

4.105 Parameter `AdcInterruptEnable`

Adds / removes the interrupt handling routine from the ADC driver code.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.106 Container `AdcPublishedInformation`

Additional published parameters not covered by `CommonPublishedInformation` container.

Note that these parameters do not have any configuration class setting, since they are published information.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.107 Parameter AdcChannelValueSigned

Information whether the result value of the ADC driver has sign information (true) or not (false). If the result shall be interpreted as signed value it shall apply to C-language rules.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	false

4.108 Parameter AdcGroupFirstChannelFixed

Information whether the first channel of an ADC Channel group can be configured (false) or is fixed (true) to a value determined by the ADC HW Unit.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	false

4.109 Parameter AdcMaxChannelResolution

Maximum Channel resolution in bits (does not specify accuracy).

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	AUTOSAR_ECUC
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	12
max	63
min	1

4.110 Container CommonPublishedInformation

Common container, aggregated by all modules. It contains published information about vendor and versions.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.111 Parameter ArReleaseMajorVersion

Major version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	4
max	4
min	4

4.112 Parameter ArReleaseMinorVersion

Minor version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	7
max	7
min	7

4.113 Parameter ArReleaseRevisionVersion

Revision version number of AUTOSAR specification on which the appropriate implementation is based on.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.114 Parameter ModuleId

Module ID of this module from Module List.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	123
max	123
min	123

4.115 Parameter SwMajorVersion

Major version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	2
max	2
min	2

4.116 Parameter SwMinorVersion

Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.117 Parameter SwPatchVersion

Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A

Property	Value
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

4.118 Parameter VendorApiInfix

In driver modules which can be instantiated several times on a single ECU, BSW00347 requires that the name of APIs is extended by the VendorId and a vendor specific name.

This parameter is used to specify the vendor specific name. In total, the implementation specific name is generated as follows:

`<ModuleName>__>VendorId>__<VendorApiInfix>.`

E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name `Can_Write` defined in the SWS will translate to `Can_123_v11r456Write`.

This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

Property	Value
type	ECUC-STRING-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	

4.119 Parameter VendorId

Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	43
max	43
min	43

4.120 Container AutosarExt

Autosar Extension API settings.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

4.121 Parameter AdcTimeoutMethod

Configures the timeout method for Adc.

Based on this selection a certain timeout method from OsIf will be used in the driver.

Note: If OSIF_COUNTER_SYSTEM or OSIF_COUNTER_CUSTOM are selected make sure the corresponding timer is enabled in OsIf General configuration.

Note: Implementation Specific Parameter.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	OSIF_COUNTER_DUMMY
literals	['OSIF_COUNTER_DUMMY', 'OSIF_COUNTER_SYSTEM', 'OSIF_COUNTER_CUSTOM']

4.122 Parameter AdcTimeoutVal

The timeout is used for preventing endless loops as resulted from driver FMEA analysis.

The hardware failure mode which is preventing, is potential cases with frozen peripheral status used for driver synchronization.

The timeout is used as escape for avoidance of endless loops. For more details please refer to driver FMEA.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	100000
max	4294967295
min	0

4.123 Parameter AdcIpDevErrorDetect

This parameter Enables / Disables development error detection for Adc IPL module.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
default Value	false

4.124 Parameter PdbIpDevErrorDetect

This parameter Enables / Disables development error detection for Pdb.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
default Value	false

4.125 Parameter AdcMulticoreSupport

This parameter globally enables the possibility to support multicore. If this parameter is enabled, at least one EcucPartition needs to be defined (in all variants).

Note This is not supported on this platform.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.126 Parameter AdcEnableGroupDependentChannelNames

This is used to generate ADC symbolic names, that depend also on the ADC group

to which each ADC channel is mapped. The generated symbolic name will be something

like:

```
#define "ADC_GroupName"_"ADC_ChannelName" "Channel index value",
```

where "Channel index value" is the channel index in the current group.

Channel indexes in each group are generated to allow result buffer access by symbolic names.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.127 Parameter AdcBypassAbortChainCheck

Bypass the delay introduced to check if an aborted conversion chain has stopped.

This increases ADC driver performance at the cost of HW-SW coherency no longer being guaranteed.

The user must make sure he does not call an ADC service before the hardware reaches the correct state.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.128 Parameter AdcConvTimeOnce

Implementation Specific Parameter.

Enable/Disable one time setting of the registers.

If Enabled, the setting of the conversion time registers will be done only once in `Adc_Init()` function for the configured hardware unit.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.129 Parameter AdcOptimizeOneShotHwTriggerConversions

Implementation Specific Parameter.

Enable/Disable The Adc driver optimization for HW Triggered groups, OneShot, Single access.

If Enabled, other types of groups cannot be configured in ADC driver and the code for interrupt routine / Dma notification will be optimized for speed.

Also, all groups must have configured channels no greater than number of available SC1 register.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE VARIANT-POST-BUILD: PRE-COMPILE
default Value	false

4.130 Parameter AdcOptimizeDmaStreamingGroups

Implementation Specific Parameter.

Enable/Disable The Adc driver enable Optimize DMA streaming groups for reducing the number of interrupts required for processing the conversions of Adc Groups that consist of one or more channels (depending on HW capabilities) and which are configured as ADC_ACCESS_MODE_STREAMING.

When this feature is enabled, only one interrupt will be raised after the completion of all stream conversions (as configured by AdcStreamingNumSamples parameter). An additional interrupt to be raised after half of the stream is converted shall also be configurable.

This feature is enabled if Adc Global Enable DMA Transfer from General/AutosarExt is also enabled.

Note:

- SetChannel(), Enable/DisableChannel() and Optimize one-shot hardware trigger cannot be use concurrently with this feature.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE VARIANT-POST-BUILD: PRE-COMPILE
default Value	false

4.131 Parameter AdcContinuousWithoutInterrupt

This parameter globally enables the possibility to configure Adc Continuous Group Without Interrupt.

This is an Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.132 Parameter AdcEnableChDisableChApi

Enable/disable the Autosar Extension implementation api(s) `Adc_EnableChannel()` and `Adc_DisableChannel()` in ADC driver.

Notethis feature is not implemented on this platform.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.133 Parameter AdcEnableInitialNotification

Enable/disable an extra notification to be called for each Adc Group conversion.

Note This feature is intended to be used together with `Adc_SetChannel` service. The initial notification can be used by the user application to call `Adc_SetChannel` API before ADC driver updates the hardware configuration for the next conversion.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.134 Parameter `AdcEnableDmaTransferMode`

This parameter globally enables the possibility to configure DMA transfer for ADC converted data. If this parameter is disabled then DMA handling code will be removed at pre-compile time and DMA transfer cannot be configure for any Adc unit in any variant. If this parameter is enabled then the DMA configuration code will not be removed.

Note This is an Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

4.135 Parameter `AdcEnableUserModeSupport`

When this parameter is enabled, the Adc module will adapt to run from User Mode, by configuring `REG_PROT` for ADC IPs

Note: The Adc driver code can be executed at any time from both supervisor and user mode.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.136 Parameter AdcEnableSimSupplyMonitor

When this parameter is enabled, the Adc module allows to use the internal channels on SIM.

Note: If Adc runs from User Mode, AdcEnableUserModeSupport must be enabled in order to configure SIM register.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.137 Parameter AdcEnablePdbInterChannelBackToBackSupport

When this parameter is enabled, PDB Inter-channel back to back feature can be configured in PdbHwUnit.

In this case, PDB CH0 and CH1 will be in back to back acknowledgement connections as a ring (E.g: PDB0 CH 0 pre-trigger 0 ->... -> PDB0 CH 0 pre-trigger 7 -> PDB0 CH 1 pre-trigger 0 -> -> PDB0 CH 1 pre-trigger 7 -> PDB0 CH 0 pre-trigger 0)

Note: If Pdb runs from User Mode, AdcEnableUserModeSupport must be enabled in order to configure SIM register.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
default Value	false

4.138 Parameter AdcEnableSetChannel

If this parameter has been configured to "TRUE", the Autosar Extension function "Adc_SetChannel()" shall be accessible, otherwise this function shall be removed from the code.

Note This is an Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
default Value	false

4.139 Parameter AdcEnableDualClockMode

Adds/removes the Dual Clock mode service Adc_SetClockMode from the code.

Also it enables the Programming of Conversion Timing registers in Adc_SetClockMode.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF

Property	Value
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.140 Parameter AdcEnableCalibration

If this parameter has been configured to "TRUE", the Autosar Extension function "Adc_Calibrate()" shall be accessible, otherwise this function shall be removed from the code.

Note This is an Implementation Specific Parameter.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.141 Parameter AdcEnableReadRawDataApi

When this parameter is enabled, the Api for reading the raw result data from an ADC unit is available to use at runtime.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false

Property	Value
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

4.142 Parameter AdcEnableGroupStreamingResultReorder

When this parameter is enabled, the adc results can be arranged as multiple sets of group result buffer if AdcStream-ResultGroup is enabled for that selected group.

E.g: for a group with channels {CH1 CH5 CH7} the resulting stream buffer shall be:

{ CH1, CH5, CH7, CH1, CH5, CH7, CH1, CH5, CH7}

instead of

{ CH1, CH1, CH1, CH5, CH5, CH5, CH7, CH7, CH7} like supported by AUTOSAR standard.

Apply only for ADC_ACCESS_MODE_STREAMING Access Mode.

Property	Value
type	ECUC-BOOLEAN-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	1
upperMultiplicity	1
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false



Chapter 5

Module Index

5.1 Software Specification

Here is a list of all modules:

Adc driver	106
Adc IPL	133
Pdb Adc IPL	156

Chapter 6

Module Documentation

6.1 Adc driver

6.1.1 Detailed Description Adc HLD.

Adc Autosar High Level Driver.

Data Structures

- struct [Adc_ValidationResultType](#)
Structure for validation results. [More...](#)
- struct [Adc_GroupStatusType](#)
Structure for group status. [More...](#)
- struct [Adc_UnitStatusType](#)
Structure for hardware unit status. [More...](#)
- struct [Adc_ConfigType](#)
Structure for ADC configuration. [More...](#)

Macros

- `#define ADC_E_UNINIT`
API service used without Adc module initialization.
- `#define ADC_E_BUSY`
Adc module is busy with a running operation.
- `#define ADC_E_IDLE`
Adc module is in idle state.
- `#define ADC_E_ALREADY_INITIALIZED`
The ADC module is already initialized.
- `#define ADC_E_PARAM_CONFIG`
The ADC module is not properly configured.
- `#define ADC_E_PARAM_POINTER`
API service is called using an invalid pointer (e.g. the pointer should not be NULL).

- #define `ADC_E_PARAM_GROUP`
API service used with an invalid ADC group.
- #define `ADC_E_WRONG_CONV_MODE`
API service used with an invalid ADC Conversion Mode.
- #define `ADC_E_WRONG_TRIGG_SRC`
API service used with an invalid ADC Trigger Source.
- #define `ADC_E_NOTIF_CAPABILITY`
Check the notification capability of a group.
- #define `ADC_E_BUFFER_UNINIT`
API service used without initializing the buffer.
- #define `ADC_E_QUEUE_FULL`
The `Adc_StartGroupConversion` and `Adc_EnableHardwareTrigger` services can not queue another conversion (queue is full)
- #define `ADC_E_PARAM_UNIT`
API service called using a wrong ADC unit.
- #define `ADC_E_INVALID_CLOCK_MODE`
`Adc_SetClockMode` service called using an invalid clock mode.
- #define `ADC_E_PARAM_CHANNEL`
`Adc_SetChannel` service called using an invalid channel list.
- #define `ADC_E_TIMEOUT`
An error occurred if the timeout counter variable has expired when checking status flags.
- #define `ADC_INIT_ID`
API service ID for `Adc_Init` function.
- #define `ADC_DEINIT_ID`
API service ID for `Adc_DeInit` function.
- #define `ADC_STARTGROUPCONVERSION_ID`
API service ID for `Adc_StartGroupConversion` function.
- #define `ADC_STOPGROUPCONVERSION_ID`
API service ID for `Adc_StopGroupConversion` function.
- #define `ADC_VALUEREADGROUP_ID`
API service ID for `Adc_ReadGroup` function.
- #define `ADC_ENABLEHARDWARETRIGGER_ID`
API service ID for `Adc_EnableHardwareTrigger` function.
- #define `ADC_DISABLEHARDWARETRIGGER_ID`
API service ID for `Adc_DisableHardwareTrigger` function.
- #define `ADC_ENABLEGROUPNOTIFICATION_ID`
API service ID for `Adc_EnableGroupNotification` function.
- #define `ADC_DISABLEGROUPNOTIFICATION_ID`
API service ID for `Adc_DisableGroupNotification` function.
- #define `ADC_GETGROUPSTATUS_ID`
API service ID for `Adc_GetGroupStatus` function.
- #define `ADC_GETVERSIONINFO_ID`
API service ID for `Adc_GetVersionInfo` function.
- #define `ADC_GETSTREAMLASTPOINTER_ID`
API service ID for `Adc_GetStreamLastPointer` function.
- #define `ADC_SETUPRESULTBUFFER_ID`
API service ID for `Adc_SetupResultBuffer` function.

- `#define` [ADC_SETCLOCKMODE_ID](#)
API service ID for `Adc_SetClockMode` function.
- `#define` [ADC_CALIBRATE_ID](#)
API service ID for `Adc_Calibrate` function.
- `#define` [ADC_SETCHANNEL_ID](#)
API service ID for `Adc_SetChannel` function.

Types Reference

- `typedef void(*` [Adc_NotifyType](#) `) (void)`
Notification function pointer definition.
- `typedef uint8` [Adc_ResolutionType](#)
channel resolution in number of bits
- `typedef Adc_ChannelType` [Adc_GroupDefType](#)
definition of channels in a group
- `typedef uint8` [Adc_PrescaleType](#)
clock prescaler factor
- `typedef uint8` [Adc_SamplingTimeType](#)
sampling time
- `typedef uint16` [Adc_StreamNumSampleType](#)
Number of samples of a streaming conversion buffer.

Enum Reference

- `enum` [Adc_GlobalStateType](#)
ADC driver status.
- `enum` [Adc_GroupConversionStateType](#)
ADC group already converted type.
- `enum` [Adc_GroupAccessModeType](#)
Adc group access Mode.
- `enum` [Adc_GroupReplacementType](#)
Adc group replacement.
- `enum` [Adc_StreamBufferModeType](#)
Adc group streaming buffer mode.
- `enum` [Adc_StatusType](#)
ADC group status.
- `enum` [Adc_NotificationType](#)
ADC group notification.
- `enum` [Adc_HwTriggerSignalType](#)
Adc hardware trigger edge.
- `enum` [Adc_TriggerSourceType](#)
Adc hardware trigger source.
- `enum` [Adc_HwTriggeringType](#)
Adc Hardware trigger.

Function Reference

- void [Adc_Init](#) (const [Adc_ConfigType](#) *ConfigPtr)
Initializes the ADC hardware unit and the driver.
- Std_ReturnType [Adc_SetupResultBuffer](#) ([Adc_GroupType](#) Group, [Adc_ValueGroupType](#) *const DataBufferPtr)
Initializes the group specific ADC result buffer pointer as configured to point to the pDataBufferPtr address which is passed as parameter.
- void [Adc_DeInit](#) (void)
Returns all ADC HW Units to a state comparable to their power on reset state.
- void [Adc_StartGroupConversion](#) ([Adc_GroupType](#) Group)
Starts the conversion of all channels of the requested ADC Channel group.
- void [Adc_StopGroupConversion](#) ([Adc_GroupType](#) Group)
Stops the conversion of all channels of the requested ADC Channel group.
- Std_ReturnType [Adc_ReadGroup](#) ([Adc_GroupType](#) Group, [Adc_ValueGroupType](#) *DataBufferPtr)
Reads the group conversion results.
- void [Adc_ReadRawData](#) ([Adc_HwUnitType](#) Unit, const [Adc_ChannelType](#) *const ChansArray, uint8 NumItems, [Adc_ValueGroupType](#) *const DataBufferPtr)
Read the raw result data from an ADC unit.
- void [Adc_EnableHardwareTrigger](#) ([Adc_GroupType](#) Group)
Enables the hardware trigger for the requested ADC Channel group.
- void [Adc_DisableHardwareTrigger](#) ([Adc_GroupType](#) Group)
Disables the hardware trigger for the requested ADC Channel group.
- void [Adc_EnableGroupNotification](#) ([Adc_GroupType](#) Group)
Enables the notification mechanism for the requested ADC channel group.
- void [Adc_DisableGroupNotification](#) ([Adc_GroupType](#) Group)
Disables the notification mechanism for the requested ADC channel group.
- [Adc_StatusType](#) [Adc_GetGroupStatus](#) ([Adc_GroupType](#) Group)
Returns the conversion status of the requested ADC Channel group.
- [Adc_StreamNumSampleType](#) [Adc_GetStreamLastPointer](#) ([Adc_GroupType](#) Group, [Adc_ValueGroupType](#) **PtrToSamplePtr)
Returns the number of valid samples per channel.
- void [Adc_GetVersionInfo](#) ([Std_VersionInfoType](#) *versioninfo)
Returns the version information of this module.
- void [Adc_Calibrate](#) ([Adc_HwUnitType](#) Unit, [Adc_CalibrationStatusType](#) *pStatus)
Executes high accuracy calibration of a ADC HW unit.
- Std_ReturnType [Adc_SetClockMode](#) ([Adc_SelectPrescalerType](#) Prescaler)
Set the ADC clock prescaler if available and modify the conversion timings.
- void [Adc_SetChannel](#) ([Adc_GroupType](#) Group, const [Adc_GroupDefType](#) *Channel, const uint16 *Delays, uint32 ChannelUpdateMask, [Adc_ChannelIndexType](#) NumberOfChannel)
Function to dynamic handling of ADC channels list for Adc channel group.

Variables

- const [Adc_ConfigType](#) * [Adc_apxCfgPtr](#) [(1U)]
Used to point the configuration structure.

6.1.2 Data Structure Documentation

6.1.2.1 struct Adc_ValidationResultType

Structure for validation results.

This structure contains the validation information

Definition at line 285 of file Adc_Types.h.

Data Fields

Type	Name	Description
boolean	EndValidations	Signal if validation ended.
Std_ReturnType	ValidParams	Return status.

6.1.2.2 struct Adc_GroupStatusType

Structure for group status.

This structure contains the group status information.

Definition at line 296 of file Adc_Types.h.

Data Fields

Type	Name	Description
volatile Adc_StatusType	Conversion	Group status.
volatile Adc_GroupConversionStateType	AlreadyConverted	Group was previously converted or not.
Adc_HwTriggeringType	HwTriggering	hw trigger enabled/disabled
Adc_NotificationType	Notification	notification enabled/disabled
volatile Adc_StreamNumSampleType	ResultIndex	index into streaming buffer that is currently being filled
Adc_ValueGroupType *	ResultsBufferPtr	Pointer to user result buffer array.
Adc_ChannelIndexType	CurrentChannel	Current channel in use.
volatile boolean	LimitCheckFailed	check limit check fail

6.1.2.3 struct Adc_UnitStatusType

Structure for hardware unit status.

This structure contains the HW unit status information.

Definition at line 322 of file Adc_Types.h.

Data Fields

Type	Name	Description
volatile Adc_QueueIndexType	SwNormalQueueIndex	Filled slots in the queue.
volatile Adc_GroupType	SwNormalQueue[(1U)]	Queued groups indexes, always executing Queue[0].
volatile Adc_GroupType	OngoingHwGroup	Ongoing hardware group ID.
uint8	Sc1Used	

6.1.2.4 struct Adc_ConfigType

Structure for ADC configuration.

Data structure containing the set of configuration parameters required for initializing the ADC Driver.

SWS_Adc_00505

Definition at line 441 of file Adc_Types.h.

Data Fields

- const Adc_GroupConfigurationType * [GroupsPtr](#)
Group configurations.
- Adc_GroupType [GroupCount](#)
Total number of groups.
- const uint16 * [GroupIdToIndexMapPtr](#)
Miscellaneous configuration parameters.
- uint8 [CoreId](#)
Configuration CoreID.
- const uint8 [AssignedPartitionCount](#)
Number of Partition.

6.1.2.4.1 Field Documentation**6.1.2.4.1.1 GroupsPtr** const Adc_GroupConfigurationType* GroupsPtr

Group configurations.

Definition at line 446 of file Adc_Types.h.

6.1.2.4.1.2 **GroupCount** `Adc_GroupType GroupCount`

Total number of groups.

Definition at line 448 of file `Adc_Types.h`.

6.1.2.4.1.3 **GroupIdToIndexMapPtr** `const uint16* GroupIdToIndexMapPtr`

Miscellaneous configuration parameters.

Definition at line 450 of file `Adc_Types.h`.

6.1.2.4.1.4 **CoreId** `uint8 CoreId`

Configuration CoreID.

Assigned Partition

Definition at line 452 of file `Adc_Types.h`.

6.1.2.4.1.5 **AssignedPartitionCount** `const uint8 AssignedPartitionCount`

Number of Partition.

<

Definition at line 456 of file `Adc_Types.h`.

6.1.3 Macro Definition Documentation

6.1.3.1 **ADC_E_UNINIT**

```
#define ADC_E_UNINIT
```

API service used without `Adc` module initialization.

Development errors. The following errors shall be detectable by the `ADC` module depending on its configuration (development / production mode).

All error codes

Definition at line 132 of file `Adc.h`.

6.1.3.2 ADC_E_BUSY

```
#define ADC_E_BUSY
```

Adc module is busy with a running operation.

Definition at line 137 of file Adc.h.

6.1.3.3 ADC_E_IDLE

```
#define ADC_E_IDLE
```

Adc module is in idle state.

Definition at line 142 of file Adc.h.

6.1.3.4 ADC_E_ALREADY_INITIALIZED

```
#define ADC_E_ALREADY_INITIALIZED
```

The ADC module is already initialized.

Definition at line 147 of file Adc.h.

6.1.3.5 ADC_E_PARAM_CONFIG

```
#define ADC_E_PARAM_CONFIG
```

The ADC module is not properly configured.

Definition at line 152 of file Adc.h.

6.1.3.6 ADC_E_PARAM_POINTER

```
#define ADC_E_PARAM_POINTER
```

API service is called using an invalid pointer (e.g. the pointer should not be NULL).

Definition at line 157 of file Adc.h.

6.1.3.7 ADC_E_PARAM_GROUP

```
#define ADC_E_PARAM_GROUP
```

API service used with an invalid ADC group.

Definition at line 162 of file Adc.h.

6.1.3.8 ADC_E_WRONG_CONV_MODE

```
#define ADC_E_WRONG_CONV_MODE
```

API service used with an invalid ADC Conversion Mode.

Definition at line 167 of file Adc.h.

6.1.3.9 ADC_E_WRONG_TRIGG_SRC

```
#define ADC_E_WRONG_TRIGG_SRC
```

API service used with an invalid ADC Trigger Source.

Definition at line 172 of file Adc.h.

6.1.3.10 ADC_E_NOTIF_CAPABILITY

```
#define ADC_E_NOTIF_CAPABILITY
```

Check the notification capability of a group.

Definition at line 177 of file Adc.h.

6.1.3.11 ADC_E_BUFFER_UNINIT

```
#define ADC_E_BUFFER_UNINIT
```

API service used without initializing the buffer.

Definition at line 182 of file Adc.h.

6.1.3.12 ADC_E_QUEUE_FULL

```
#define ADC_E_QUEUE_FULL
```

The `Adc_StartGroupConversion` and `Adc_EnableHardwareTrigger` services can not queue another conversion (queue is full)

Definition at line 210 of file `Adc.h`.

6.1.3.13 ADC_E_PARAM_UNIT

```
#define ADC_E_PARAM_UNIT
```

API service called using a wrong ADC unit.

Definition at line 247 of file `Adc.h`.

6.1.3.14 ADC_E_INVALID_CLOCK_MODE

```
#define ADC_E_INVALID_CLOCK_MODE
```

`Adc_SetClockMode` service called using an invalid clock mode.

Definition at line 264 of file `Adc.h`.

6.1.3.15 ADC_E_PARAM_CHANNEL

```
#define ADC_E_PARAM_CHANNEL
```

`Adc_SetChannel` service called using an invalid channel list.

Definition at line 271 of file `Adc.h`.

6.1.3.16 ADC_E_TIMEOUT

```
#define ADC_E_TIMEOUT
```

An error occurred if the timeout counter variable has expired when checking status flags.

Definition at line 277 of file `Adc.h`.

6.1.3.17 ADC_INIT_ID

```
#define ADC_INIT_ID
```

API service ID for Adc_Init function.

All AUTOSAR API's service IDs

Definition at line 302 of file Adc.h.

6.1.3.18 ADC_DEINIT_ID

```
#define ADC_DEINIT_ID
```

API service ID for Adc_DeInit function.

Definition at line 307 of file Adc.h.

6.1.3.19 ADC_STARTGROUPCONVERSION_ID

```
#define ADC_STARTGROUPCONVERSION_ID
```

API service ID for Adc_StartGroupConversion function.

Definition at line 312 of file Adc.h.

6.1.3.20 ADC_STOPGROUPCONVERSION_ID

```
#define ADC_STOPGROUPCONVERSION_ID
```

API service ID for Adc_StopGroupConversion function.

Definition at line 317 of file Adc.h.

6.1.3.21 ADC_VALUEREADGROUP_ID

```
#define ADC_VALUEREADGROUP_ID
```

API service ID for Adc_ReadGroup function.

Definition at line 322 of file Adc.h.

6.1.3.22 ADC_ENABLEHARDWARETRIGGER_ID

```
#define ADC_ENABLEHARDWARETRIGGER_ID
```

API service ID for Adc_EnableHardwareTrigger function.

Definition at line 327 of file Adc.h.

6.1.3.23 ADC_DISABLEHARDWARETRIGGER_ID

```
#define ADC_DISABLEHARDWARETRIGGER_ID
```

API service ID for Adc_DisableHardwareTrigger function.

Definition at line 332 of file Adc.h.

6.1.3.24 ADC_ENABLEGROUPNOTIFICATION_ID

```
#define ADC_ENABLEGROUPNOTIFICATION_ID
```

API service ID for Adc_EnableGroupNotification function.

Definition at line 337 of file Adc.h.

6.1.3.25 ADC_DISABLEGROUPNOTIFICATION_ID

```
#define ADC_DISABLEGROUPNOTIFICATION_ID
```

API service ID for Adc_DisableGroupNotification function.

Definition at line 342 of file Adc.h.

6.1.3.26 ADC_GETGROUPSTATUS_ID

```
#define ADC_GETGROUPSTATUS_ID
```

API service ID for Adc_GetGroupStatus function.

Definition at line 347 of file Adc.h.

6.1.3.27 ADC_GETVERSIONINFO_ID

```
#define ADC_GETVERSIONINFO_ID
```

API service ID for `Adc_GetVersionInfo` function.

Definition at line 352 of file `Adc.h`.

6.1.3.28 ADC_GETSTREAMLASTPOINTER_ID

```
#define ADC_GETSTREAMLASTPOINTER_ID
```

API service ID for `Adc_GetStreamLastPointer` function.

Definition at line 357 of file `Adc.h`.

6.1.3.29 ADC_SETUPRESULTBUFFER_ID

```
#define ADC_SETUPRESULTBUFFER_ID
```

API service ID for `Adc_SetupResultBuffer` function.

Definition at line 362 of file `Adc.h`.

6.1.3.30 ADC_SETCLOCKMODE_ID

```
#define ADC_SETCLOCKMODE_ID
```

API service ID for `Adc_SetClockMode` function.

All Autosar Extension API's service IDs NOTE: Parameters used when raising an error/exception

Definition at line 422 of file `Adc.h`.

6.1.3.31 ADC_CALIBRATE_ID

```
#define ADC_CALIBRATE_ID
```

API service ID for `Adc_Calibrate` function.

Definition at line 447 of file `Adc.h`.

6.1.3.32 ADC_SETCHANNEL_ID

```
#define ADC_SETCHANNEL_ID
```

API service ID for `Adc_SetChannel` function.

Definition at line 484 of file `Adc.h`.

6.1.4 Types Reference

6.1.4.1 Adc_NotifyType

```
typedef void(* Adc_NotifyType) (void)
```

Notification function pointer definition.

Definition at line 252 of file `Adc_Types.h`.

6.1.4.2 Adc_ResolutionType

```
typedef uint8 Adc_ResolutionType
```

channel resolution in number of bits

Definition at line 256 of file `Adc_Types.h`.

6.1.4.3 Adc_GroupDefType

```
typedef Adc_ChannelType Adc_GroupDefType
```

definition of channels in a group

Definition at line 266 of file `Adc_Types.h`.

6.1.4.4 Adc_PrescaleType

```
typedef uint8 Adc_PrescaleType
```

clock prescaler factor

Definition at line 270 of file `Adc_Types.h`.

6.1.4.5 Adc_SamplingTimeType

```
typedef uint8 Adc_SamplingTimeType
```

sampling time

Definition at line 274 of file Adc_Types.h.

6.1.4.6 Adc_StreamNumSampleType

```
typedef uint16 Adc_StreamNumSampleType
```

Number of samples of a streaming conversion buffer.

Definition at line 278 of file Adc_Types.h.

6.1.5 Enum Reference

6.1.5.1 Adc_GlobalStateType

```
enum Adc_GlobalStateType
```

ADC driver status.

Used to differentiate if ADC driver is already uninit, during init or already initialized or not.

Enumerator

ADC_STATE_UNINIT	Adc driver uninitialized.
ADC_STATE_BUSY	Adc driver busy.
ADC_STATE_IDLE	Adc driver idle.

Definition at line 119 of file Adc_Types.h.

6.1.5.2 Adc_GroupConversionStateType

```
enum Adc_GroupConversionStateType
```

ADC group already converted type.

Used to differentiate if group is already converted or not.

Enumerator

ADC_NOT_YET_CONVERTED	Group not yet converted.
ADC_ALREADY_CONVERTED	Group is already converted.

Definition at line 132 of file Adc_Types.h.

6.1.5.3 Adc_GroupAccessModeType

enum [Adc_GroupAccessModeType](#)

Adc group access Mode.

Used for value received by Tressos interface configuration.

SWS_Adc_00528

Enumerator

ADC_ACCESS_MODE_SINGLE	Single access mode.
ADC_ACCESS_MODE_STREAMING	Streaming access mode.

Definition at line 145 of file Adc_Types.h.

6.1.5.4 Adc_GroupReplacementType

enum [Adc_GroupReplacementType](#)

Adc group replacement.

Used for value received by Tressos interface configuration.

SWS_Adc_00523

Enumerator

ADC_GROUP_REPL_ABORT_RESTART	Abort and restart of group.
ADC_GROUP_REPL_SUSPEND_RESUME	Suspend and resuming of group.

Definition at line 158 of file Adc_Types.h.

6.1.5.5 Adc_StreamBufferModeType

enum [Adc_StreamBufferModeType](#)

Adc group streaming buffer mode.

Used for value received by Tressos interface configuration.

SWS__Adc__00519

Enumerator

ADC_STREAM_BUFFER_LINEAR	Linear streaming.
ADC_STREAM_BUFFER_CIRCULAR	Circular streaming.

Definition at line 171 of file Adc_Types.h.

6.1.5.6 Adc_StatusType

enum [Adc_StatusType](#)

ADC group status.

ADC group enumeration type.

SWS__Adc__00513

Enumerator

ADC_IDLE	Group is in IDLE state.
ADC_BUSY	Group is in BUSY state.
ADC_COMPLETED	Group is in COMPLETED state.
ADC_STREAM_COMPLETED	Group is in STREAM_COMPLETED state.

Definition at line 184 of file Adc_Types.h.

6.1.5.7 Adc_NotificationType

enum [Adc_NotificationType](#)

ADC group notification.

Indicates if notification is enabled for the group.

Enumerator

ADC_NOTIFICATION_DISABLED	Notification is disabled.
ADC_NOTIFICATION_ENABLED	Notification is enabled.

Definition at line 197 of file Adc_Types.h.

6.1.5.8 Adc_HwTriggerSignalType

enum `Adc_HwTriggerSignalType`

Adc hardware trigger edge.

Used for value received by Tressos interface configuration.

SWS_Adc_00520

Enumerator

ADC_HW_TRIG_RISING_EDGE	Rising edge.
ADC_HW_TRIG_FALLING_EDGE	Falling edge.
ADC_HW_TRIG_BOTH_EDGES	falling and rising edge

Definition at line 211 of file Adc_Types.h.

6.1.5.9 Adc_TriggerSourceType

enum `Adc_TriggerSourceType`

Adc hardware trigger source.

Used for value received by Tressos interface configuration.

SWS_Adc_00514

Enumerator

ADC_TRIGG_SRC_SW	Software triggered.
ADC_TRIGG_SRC_HW	Hardware triggered.

Definition at line 225 of file Adc_Types.h.

6.1.5.10 Adc_HwTriggeringType

enum `Adc_HwTriggeringType`

Adc Hardware trigger.

Indicates if hardware trigger is enabled for group.

Enumerator

ADC_HWTRIGGER_DISABLED	Hardware trigger is disabled.
ADC_HWTRIGGER_ENABLED	Hardware trigger is enabled.

Definition at line 240 of file `Adc_Types.h`.

6.1.6 Function Reference

6.1.6.1 Adc_Init()

```
void Adc_Init (
    const Adc_ConfigType * ConfigPtr )
```

Initializes the ADC hardware unit and the driver.

This function will initialize both the ADC HW unit and the driver structures.

Parameters

in	<i>ConfigPtr</i>	Pointer to configuration set in Variant PB (Variant PC requires a NULL_PTR).
----	------------------	------------------------------------------------------------------------------

Returns

void

6.1.6.2 Adc_SetupResultBuffer()

```
Std_ReturnType Adc_SetupResultBuffer (
    Adc_GroupType Group,
    Adc_ValueGroupType *const DataBufferPtr )
```

Initializes the group specific ADC result buffer pointer as configured to point to the `pDataBufferPtr` address which is passed as parameter.

Initializes ADC driver with the group specific result buffer start address where the conversion results will be stored. The application has to ensure that the application buffer, where `pDataBufferPtr` points to, can hold all the conversion results of the specified group. The initialization with `Adc_SetupResultBuffer` is required after reset, before a group conversion can be started.

Parameters

in	<i>Group</i>	Numeric ID of requested ADC channel group.
in	<i>pDataBufferPtr</i>	Pointer to result data buffer

Returns

`Std_ReturnType` Standard return type. `E_OK`: Result buffer pointer initialized correctly. `E_NOT_OK`: Operation failed or development error occurred.

6.1.6.3 `Adc_DeInit()`

```
void Adc_DeInit (
    void )
```

Returns all ADC HW Units to a state comparable to their power on reset state.

Returns all ADC HW Units to a state comparable to their power on reset state, and de-initialize the ADC driver.

Returns

`void`

6.1.6.4 `Adc_StartGroupConversion()`

```
void Adc_StartGroupConversion (
    Adc_GroupType Group )
```

Starts the conversion of all channels of the requested ADC Channel group.

This function will start the SW conversion of all channels of the requested ADC channel group.

Parameters

in	<i>Group</i>	Numeric ID of requested ADC channel group.
----	--------------	--------------------------------------------

Returns

void

6.1.6.5 **Adc_StopGroupConversion()**

```
void Adc_StopGroupConversion (
    Adc_GroupType Group )
```

Stops the conversion of all channels of the requested ADC Channel group.

This function will stop the SW conversion of all channels of the requested ADC channel group.

Parameters

in	<i>Group</i>	Numeric ID of requested ADC channel group.
----	--------------	--------------------------------------------

Returns

void

6.1.6.6 **Adc_ReadGroup()**

```
Std_ReturnType Adc_ReadGroup (
    Adc_GroupType Group,
    Adc_ValueGroupType * DataBufferPtr )
```

Reads the group conversion results.

Reads the group conversion results of the last completed conversion round of the requested group and stores the channel values starting at the pDataBufferPtr address. The group channel values are stored in ascending channel number order (in contrast to the storage layout of the result buffer if streaming access is configured).

Parameters

in	<i>Group</i>	Numeric ID of requested ADC Channel group.
in	<i>pDataBufferPtr</i>	ADC results of all channels of the selected group are stored in the data buffer addressed with the pointer.

Returns

Std_ReturnType Standard return type. E_OK: results are available and written to the data buffer. E_NO←
T_OK: no results are available or development error occurred.

6.1.6.7 Adc_ReadRawData()

```
void Adc_ReadRawData (
    Adc_HwUnitType Unit,
    const Adc_ChannelType *const ChansArray,
    uint8 NumItems,
    Adc_ValueGroupType *const DataBufferPtr )
```

Read the raw result data from an ADC unit.

Read the raw result data from an ADC unit. Intended for reading ADC results directly from ADC registers and can eliminate surplus interrupts if there are more triggered measurements than FIFO length. Measured values remain in ADC result registers(user must ensure that they are not overwritten).

Parameters

in	<i>Unit</i>	Adc unit used. Recommended to use generated define for Adc Logical Unit Id.
in	<i>ChansArray</i>	List of channels for which results to be read
in	<i>NumItems</i>	Number of results to read
out	<i>DataBufferPtr</i>	Destination pointer in which the results will be written

Returns

void

6.1.6.8 Adc_EnableHardwareTrigger()

```
void Adc_EnableHardwareTrigger (
    Adc_GroupType Group )
```

Enables the hardware trigger for the requested ADC Channel group.

This function will enable the HW trigger source for the requested ADC channel group. This function does set the CTU register for all platform that have the CTU Hw Unit.

Parameters

in	<i>Group</i>	Numeric ID of requested ADC channel group.
----	--------------	--------------------------------------------

Returns

void

6.1.6.9 `Adc_DisableHardwareTrigger()`

```
void Adc_DisableHardwareTrigger (
    Adc_GroupType Group )
```

Disables the hardware trigger for the requested ADC Channel group.

This function will disable the HW trigger source for the requested ADC channel group.

Parameters

in	<i>Group</i>	Numeric ID of requested ADC channel group.
----	--------------	--------------------------------------------

Returns

void

6.1.6.10 `Adc_EnableGroupNotification()`

```
void Adc_EnableGroupNotification (
    Adc_GroupType Group )
```

Enables the notification mechanism for the requested ADC channel group.

This function will enable the notification mechanism only for the requested ADC channel group.

Parameters

in	<i>Group</i>	Numeric ID of requested ADC channel group.
----	--------------	--------------------------------------------

Returns

void

6.1.6.11 `Adc_DisableGroupNotification()`

```
void Adc_DisableGroupNotification (
    Adc_GroupType Group )
```

Disables the notification mechanism for the requested ADC channel group.

This function will disable the notification mechanism only for the requested ADC channel group.

Parameters

in	<i>Group</i>	Numeric ID of requested ADC channel group.
----	--------------	--------------------------------------------

Returns

void

6.1.6.12 **Adc_GetGroupStatus()**

```
Adc_StatusType Adc_GetGroupStatus (
    Adc_GroupType Group )
```

Returns the conversion status of the requested ADC Channel group.

This function will return the conversion status of the requested ADC channel group.

Parameters

in	<i>Group</i>	Numeric ID of requested ADC channel group.
----	--------------	--------------------------------------------

Returns

Adc_StatusType Conversion status for the requested group. ADC_IDLE in case of errors. conversion status in case of no errors.

6.1.6.13 **Adc_GetStreamLastPointer()**

```
Adc_StreamNumSampleType Adc_GetStreamLastPointer (
    Adc_GroupType Group,
    Adc_ValueGroupType ** PtrToSamplePtr )
```

Returns the number of valid samples per channel.

Returns the number of valid samples per channel, stored in the result buffer. Reads a pointer, pointing to a position in the group result buffer. With the pointer position, the results of all group channels of the last completed conversion round can be accessed. With the pointer and the return value, all valid group conversion results can be accessed (the user has to take the layout of the result buffer into account).

Parameters

in	<i>Group</i>	Numeric ID of requested ADC channel group.
out	<i>PtrToSamplePtr</i>	Pointer to result buffer pointer.

Returns

Adc_StreamNumSampleType Number of valid samples per channel. 0 in case of errors. >0 Number of valid samples per channel.

6.1.6.14 Adc_GetVersionInfo()

```
void Adc_GetVersionInfo (
    Std_VersionInfoType * versioninfo )
```

Returns the version information of this module.

Returns the version information of this module.

Parameters

out	<i>pVersionInfo</i>	Pointer to where to store the version information of this module. structure in case of no errors.
-----	---------------------	---------------------------------------------------------------------------------------------------

6.1.6.15 Adc_Calibrate()

```
void Adc_Calibrate (
    Adc_HwUnitType Unit,
    Adc_CalibrationStatusType * pStatus )
```

Executes high accuracy calibration of a ADC HW unit.

This function calibrates the ADC HW unit and updates calibration related registers

Parameters

in	<i>Unit</i>	Adc unit used. Recommended to use generated define for Adc Logical Unit Id
in	<i>pStatus</i>	Status of the ADC HW unit calibration and list of failed and passed tests.

Returns

void

6.1.6.16 `Adc_SetClockMode()`

```
Std_ReturnType Adc_SetClockMode (
    Adc_SelectPrescalerType Prescaler )
```

Set the ADC clock prescaler if available and modify the conversion timings.

This function sets the ADC clock prescaler (Analog clock frequency selector)

Parameters

in	<i>Prescaler</i>	Normal or Alternate mode.
----	------------------	---------------------------

Returns

Std_ReturnType Standard return type. E_OK: In case of successful settings. E_NOT_OK: In case of unsuccessful settings.

6.1.6.17 `Adc_SetChannel()`

```
void Adc_SetChannel (
    Adc_GroupType Group,
    const Adc_GroupDefType * Channel,
    const uint16 * Delays,
    uint32 ChannelUpdateMask,
    Adc_ChannelIndexType NumberOfChannel )
```

Function to dynamic handling of ADC channels list for Adc channel group.

Dynamic handling of ADC channels list. This function to dynamic handling of ADC channels list for Adc channel group.

Parameters

in	<i>Group</i>	Group Id.
in	<i>Channel</i>	Pointer to array of channels to be reconfigured for the group. Channel value is logical channel ID.
in	<i>Delays</i>	Pointer to array of delay value associated with array of channels to be reconfigured.
in	<i>ChannelUpdateMask</i>	Bitmask selecting which channels to be reconfigured.
in	<i>NumberOfChannel</i>	Number of channels in channels array.

Note

For platforms supporting delays (Only if ADC_IPW_DELAY_AVAILABLE == STD_ON): Delays:

- If `NULL_PTR`: channel delay values are not reconfigured.
- If group has configured only 1 delay: pointer to new delay value.
- If group has configured delay for each channel: array with new delay values - number of elements must be `NumberOfChannel`.

`ChannelUpdateMask`:

- Bitmask example: `0b0110` only reconfigures channels from positions 1 and 2.
- This bit mask can be used only if number of group channels are not greater than number of SC1 registers
- Last bit of this mask must be set for having interrupt if `NumberOfChannel` is different than number of configured channels.

6.1.7 Variable Documentation

6.1.7.1 `Adc_apxCfgPtr`

```
const Adc_ConfigType* Adc_apxCfgPtr[(1U)] [extern]
```

Used to point the configuration structure.

6.2 Adc IPL

6.2.1 Detailed Description Adc HW module.

Adc IP layer hardware module.

Data Structures

- struct [Adc_Ip_ChanConfigType](#)
Defines the channel configuration. [More...](#)
- struct [Adc_Ip_ClockConfigType](#)
Defines the ADC clock configuration. [More...](#)
- struct [Adc_Ip_ConfigType](#)
Defines the module configuration. [More...](#)
- struct [Adc_Ip_StateStructType](#)
Structure used to store runtime info. [More...](#)

Types Reference

- typedef void [Adc_Ip_ChanNotificationType](#)(const uint8 ControlChanIdx)
Defines the channel notification header.

Enum Reference

- enum [Adc_Ip_StatusType](#)
ADC IP status return type.
- enum [Adc_Ip_ClockSelType](#)
Clock divider selection.
- enum [Adc_Ip_ResolutionType](#)
Conversion resolution selection.
- enum [Adc_Ip_ClkSourceType](#)
Input clock source selection.
- enum [Adc_Ip_AvgSelectType](#)
Hardware average selection.
- enum [Adc_Ip_TrigType](#)
Trigger type selection.
- enum [Adc_Ip_VoltageReferenceType](#)
Voltage reference selection.
- enum [Adc_Ip_InputChannelType](#)
Enumeration of input channels assignable to a control channel.
Note 0: entries in this enum are affected by `::FEATURE_ADC_NUM_EXT_CHANS`, which is device dependent and controlled from "device_name"_features.h file.

Function Reference

- void [Adc_Ip_Init](#) (const uint32 Instance, const [Adc_Ip_ConfigType](#) *const Config)
Initialize ADC module.
- void [Adc_Ip_DeInit](#) (const uint32 Instance)
Deinitialize ADC module.
- void [Adc_Ip_SetAveraging](#) (const uint32 Instance, const boolean AvgEn, const [Adc_Ip_AvgSelectType](#) AvgSel)
Configure averaging.
- void [Adc_Ip_SetSampleTime](#) (const uint32 Instance, const uint8 SampleTime)
Configure sample time.
- void [Adc_Ip_EnableDma](#) (const uint32 Instance)
Enable DMA.
- void [Adc_Ip_DisableDma](#) (const uint32 Instance)
Disable DMA.
- void [Adc_Ip_SetTriggerMode](#) (const uint32 Instance, const [Adc_Ip_TrigType](#) TriggerMode)
Switch between hardware and software trigger.
- void [Adc_Ip_SetContinuousMode](#) (const uint32 Instance, const boolean ContinuousModeEnable)
Switch between continuous and one shot conversion mode.
- void [Adc_Ip_SetResolution](#) (const uint32 Instance, const [Adc_Ip_ResolutionType](#) Resolution)
Set ADC module resolution.
- void [Adc_Ip_SetClockMode](#) (const uint32 Instance, const [Adc_Ip_ClockConfigType](#) *const Config)
Set the ADC clock values.
- void [Adc_Ip_ConfigChannel](#) (const uint32 Instance, const [Adc_Ip_ChانConfigType](#) *const ChanConfig)
Configure the selected control channel with the given configuration structure.
- [Adc_Ip_StatusType](#) [Adc_Ip_SetDisabledChannel](#) (const uint32 Instance, const uint8 ControlChanIdx, const boolean WithTimeout)
Disable selected channel.
- void [Adc_Ip_StartConversion](#) (const uint32 Instance, [Adc_Ip_InputChannelType](#) InputChannel, const boolean InterruptEnable)
Start a software triggered conversion.
- boolean [Adc_Ip_GetConvActiveFlag](#) (const uint32 Instance)
Read and return conversion active flag status.
- boolean [Adc_Ip_GetChanInterrupt](#) (const uint32 Instance, const uint8 ControlChanIdx)
Check if selected channel has interrupt set.
- boolean [Adc_Ip_GetConvCompleteFlag](#) (const uint32 Instance, const uint8 ControlChanIdx)
Get the value of conversion complete flag of a channel.
- uint16 [Adc_Ip_GetConvData](#) (const uint32 Instance, const uint8 ControlChanIdx)
Get the last result for the selected control channel.
- [Adc_Ip_StatusType](#) [Adc_Ip_DoCalibration](#) (const uint32 Instance)
Perform calibration of the ADC module.
- [Adc_Ip_StatusType](#) [Adc_Ip_ClearLatchedTriggers](#) (const uint32 Instance)
Clear latched triggers under processing.
- void [Adc_Ip_EnableChannelNotification](#) (const uint32 Instance, const uint8 ControlChanIdx)
Enable channel notification.
- void [Adc_Ip_DisableChannelNotification](#) (const uint32 Instance, const uint8 ControlChanIdx)
Disable channel notification.

- void [Adc_Ip_ClearTrigErrReg](#) (const uint32 Instance)
Clear all trigger error flags.
- uint32 [Adc_Ip_GetTrigErrReg](#) (const uint32 Instance)
Get all trigger error flags.
- uint32 [Adc_Ip_GetDataAddress](#) (const uint32 Instance, const uint8 Index)
Return the address of the specified data register.
- [Adc_Ip_StatusType](#) [Adc_Ip_GetChanData](#) (const uint32 Instance, const [Adc_Ip_InputChannelType](#) Channel, uint16 *const Result)
Get the last result for the selected control channel.
- void [Adc_Ip_SetSupplyMonitoringEnable_TrustedCall](#) (const boolean SupplyEnable)
This function enable supply monitoring for the internal channels on SIM registers.
- void [Adc_Ip_ConfigSupplyMonitoringChannel_TrustedCall](#) (const uint32 SupplyChannel)
This function configures the internal channels on on SIM registers.
- void [Adc_Ip_ResetSupplyMonitoringChannel_TrustedCall](#) (void)
This function resets the muxing for ADC channel on SIM register to reset value.

6.2.2 Data Structure Documentation

6.2.2.1 struct [Adc_Ip_ChanConfigType](#)

Defines the channel configuration.

This structure is used to configure channels

Implements : [Adc_Ip_ChanConfigType_Class](#)

Definition at line 343 of file [Adc_Ip_Types.h](#).

Data Fields

Type	Name	Description
uint8	ChnIdx	Control channel 0
Adc_Ip_InputChannelType	Channel	Selection of input channel for measurement
boolean	InterruptEnable	Enable interrupts for this channel

6.2.2.2 struct [Adc_Ip_ClockConfigType](#)

Defines the ADC clock configuration.

This structure is used to configure ADC clock

Implements : [Adc_Ip_ClockConfigType_Class](#)

Definition at line 359 of file [Adc_Ip_Types.h](#).

Data Fields

Type	Name	Description
Adc_Ip_ClockSelType	ClockDivide	Selected clock
Adc_Ip_ClkSourceType	InputClock	Input clock source
uint8	SampleTime	Sample time in AD Clocks
boolean	AvgEn	Enable averaging functionality
Adc_Ip_AvgSelectType	AvgSel	Selection for number of samples used for averaging

6.2.2.3 struct Adc_Ip_ConfigType

Defines the module configuration.

This structure is used to configure the ADC module

Implements : `Adc_Ip_ConfigType_Class`

Definition at line 375 of file `Adc_Ip_Types.h`.

Data Fields

Type	Name	Description
Adc_Ip_ClockSelType	ClockDivide	Divider of the input clock for the ADC
Adc_Ip_ClockSelType	CalibrationClockDivide	Divider of the input clock for Calibration
Adc_Ip_ClkSourceType	InputClock	Input clock source
uint8	SampleTime	Sample time in AD Clocks
boolean	AvgEn	Enable averaging functionality
Adc_Ip_AvgSelectType	AvgSel	Selection for number of samples used for averaging
Adc_Ip_ResolutionType	Resolution	ADC resolution (8,10,12 bit)
Adc_Ip_TrigType	TriggerMode	ADC trigger mode (software, hardware) - affects only the first control channel
boolean	DmaEnable	Enable DMA for the ADC
Adc_Ip_VoltageReferenceType	VoltageRef	Voltage reference used
boolean	ContinuousConvEnable	Enable Continuous conversions
boolean	SupplyMonitoringEnable	Only available for ADC 0. Enable internal supply monitoring - enables measurement of ADC_IP_INPUTC↔HAN_SUPPLY_ sources.
boolean	CompareEnable	Enable the compare feature
boolean	CompareGreaterThanEnable	Enable Greater-Than functionality
boolean	CompareRangeFuncEnable	Enable Range functionality
uint16	CompVal1	First Compare Value
uint16	CompVal2	Second Compare Value
uint16	UsrGain	User-configurable gain

Data Fields

Type	Name	Description
uint16	UsrOffset	User-configurable Offset (2's complement, subtracted from result)
uint8	NumChannels	User-configurable Number of Channels
const Adc_Ip_ChanConfigType *	ChannelConfigs	User-configurable channel configuration
Adc_Ip_ChanNotificationType *	ConversionCompleteNotification	Individual channel notification

6.2.2.4 struct Adc_Ip_StateStructType

Structure used to store runtime info.

This structure is used to store ADC runtime info

Implements : [Adc_Ip_StateStructType_Class](#)

Definition at line 420 of file [Adc_Ip_Types.h](#).

Data Fields

Type	Name	Description
boolean	Init	Check if the driver was initialized.
Adc_Ip_ClockSelType	CalibrationClockDivide	Divider of the input clock for Calibration
Adc_Ip_ChanNotificationType *	ConversionCompleteNotification	Individual channel notification
Adc_Ip_InputChannelType	ChannelConfig[ADC_MAX_CHAN_COUNT]	hardware channel config

6.2.3 Types Reference**6.2.3.1 Adc_Ip_ChanNotificationType**

```
typedef void Adc_Ip_ChanNotificationType(const uint8 ControlChanIdx)
```

Defines the channel notification header.

This header is used for channel notification callbacks. Note: the return paramter is the index of the control channel(numeric index of SC1x e.g. SC1A has index 0), not the input channel.

Implements : [Adc_Ip_ChanNotificationType_Class](#)

Definition at line 333 of file [Adc_Ip_Types.h](#).

6.2.4 Enum Reference

6.2.4.1 Adc_Ip_StatusType

enum `Adc_Ip_StatusType`

ADC IP status return type.

This structure is used as return type

Implements : `Adc_Ip_StatusType_Class`

Enumerator

<code>ADC_IP_STATUS_SUCCESS</code>	Function completed successfully
<code>ADC_IP_STATUS_ERROR</code>	Function didn't complete successfully
<code>ADC_IP_STATUS_TIMEOUT</code>	Function operation timed out

Definition at line 102 of file `Adc_Ip_Types.h`.

6.2.4.2 Adc_Ip_ClockSelType

enum `Adc_Ip_ClockSelType`

Clock divider selection.

This structure is used to configure the converter input clock

Implements : `Adc_Ip_ClockSelType_Class`

Enumerator

<code>ADC_IP_CLK_FULL_BUS</code>	Input clock divided by 1.
<code>ADC_IP_CLK_HALF_BUS</code>	Input clock divided by 2.
<code>ADC_IP_CLK_QUARTER_BUS</code>	Input clock divided by 4.
<code>ADC_IP_CLK_EIGHTH_BUS</code>	Input clock divided by 8.

Definition at line 116 of file `Adc_Ip_Types.h`.

6.2.4.3 Adc_Ip_ResolutionType

enum `Adc_Ip_ResolutionType`

Conversion resolution selection.

Implements : `Adc_Ip_ResolutionType_Class`

Enumerator

ADC_IP_RESOLUTION_8BIT	8-bit resolution mode
ADC_IP_RESOLUTION_12BIT	12-bit resolution mode
ADC_IP_RESOLUTION_10BIT	10-bit resolution mode

Definition at line 129 of file Adc_Ip_Types.h.

6.2.4.4 Adc_Ip_ClkSourceType

enum `Adc_Ip_ClkSourceType`

Input clock source selection.

This structure is used to select the input clock source

Implements : `Adc_Ip_ClkSourceType_Class`

Enumerator

ADC_IP_CLK_ALT↔ _1	Input clock alternative 1.
ADC_IP_CLK_ALT↔ _2	Input clock alternative 2.
ADC_IP_CLK_ALT↔ _3	Input clock alternative 3.
ADC_IP_CLK_ALT↔ _4	Input clock alternative 4.

Definition at line 143 of file Adc_Ip_Types.h.

6.2.4.5 Adc_Ip_AvgSelectType

enum `Adc_Ip_AvgSelectType`

Hardware average selection.

This structure is used to select the number of conversions to average in order to get the conversion data.

Implements : `Adc_Ip_AvgSelectType_Class`

Enumerator

ADC_IP_AVG_4_CONV	4 conversions per conversion data
ADC_IP_AVG_8_CONV	8 conversions per conversion data
ADC_IP_AVG_16_CONV	16 conversions per conversion data
ADC_IP_AVG_32_CONV	32 conversions per conversion data

Definition at line 159 of file `Adc_Ip_Types.h`.

6.2.4.6 `Adc_Ip_TrigType`

```
enum Adc_Ip_TrigType
```

Trigger type selection.

Implements : `Adc_Ip_TrigType_Class`

Enumerator

ADC_IP_TRIGGER_SOFTWARE	Software trigger.
ADC_IP_TRIGGER_HARDWARE	Hardware trigger.

Definition at line 171 of file `Adc_Ip_Types.h`.

6.2.4.7 `Adc_Ip_VoltageReferenceType`

```
enum Adc_Ip_VoltageReferenceType
```

Voltage reference selection.

Implements : `Adc_Ip_VoltageReferenceType_Class`

Enumerator

ADC_IP_VOLTAGEREF_VREF	VrefH and VrefL as Voltage reference.
ADC_IP_VOLTAGEREF_VALT	ValtH and ValtL as Voltage reference.

Definition at line 230 of file `Adc_Ip_Types.h`.

6.2.4.8 Adc_Ip_InputChannelType

enum `Adc_Ip_InputChannelType`

Enumeration of input channels assignable to a control channel.

Note 0: entries in this enum are affected by `::FEATURE_ADC_NUM_EXT_CHANS`, which is device dependent and controlled from `"device_name"_features.h` file.

Note 1: the actual number of external channels may differ between device packages and ADC instances. Reading a channel that is not connected externally, will return a random value within the range. Please refer to the Reference Manual for the maximum number of external channels for each device variant and ADC instance.

Note 2: `ADC_IP_INPUTCHAN_SUPPLY_` select which internal supply channel to be measured. They are only available for ADC0 and measured internally via internal input channel 0. Please note that supply monitoring needs to be enabled separately via dedicated flag in `adc_converter_config_t`.

Implements : `Adc_Ip_InputChannelType_Class`

Enumerator

<code>ADC_IP_INPUTCHAN_EXT0</code>	External input channel 0
<code>ADC_IP_INPUTCHAN_EXT1</code>	External input channel 1
<code>ADC_IP_INPUTCHAN_EXT2</code>	External input channel 2
<code>ADC_IP_INPUTCHAN_EXT3</code>	External input channel 3
<code>ADC_IP_INPUTCHAN_EXT4</code>	External input channel 4
<code>ADC_IP_INPUTCHAN_EXT5</code>	External input channel 5
<code>ADC_IP_INPUTCHAN_EXT6</code>	External input channel 6
<code>ADC_IP_INPUTCHAN_EXT7</code>	External input channel 7
<code>ADC_IP_INPUTCHAN_EXT8</code>	External input channel 8
<code>ADC_IP_INPUTCHAN_EXT9</code>	External input channel 9
<code>ADC_IP_INPUTCHAN_EXT10</code>	External input channel 10
<code>ADC_IP_INPUTCHAN_EXT11</code>	External input channel 11
<code>ADC_IP_INPUTCHAN_EXT12</code>	External input channel 12
<code>ADC_IP_INPUTCHAN_EXT13</code>	External input channel 13
<code>ADC_IP_INPUTCHAN_EXT14</code>	External input channel 14
<code>ADC_IP_INPUTCHAN_DISABLED</code>	Channel disabled
<code>ADC_IP_INPUTCHAN_INT0</code>	Internal input channel 0
<code>ADC_IP_INPUTCHAN_INT1</code>	Internal input channel 1
<code>ADC_IP_INPUTCHAN_INT2</code>	Internal input channel 2
<code>ADC_IP_INPUTCHAN_INT3</code>	Internal input channel 3
<code>ADC_IP_INPUTCHAN_TEMP</code>	Temperature Sensor

Enumerator

ADC_IP_INPUTCHAN_BANDGAP	Band Gap
ADC_IP_INPUTCHAN_VREFH	Voltage Reference Select High
ADC_IP_INPUTCHAN_VREFL	Voltage Reference Select Low The following channels are measured via internal input channel 0
ADC_IP_INPUTCHAN_SUPPLY_VDD	Monitor internal supply 5 V input VDD supply.
ADC_IP_INPUTCHAN_SUPPLY_VDDA	Monitor internal supply 5 V input analog supply.
ADC_IP_INPUTCHAN_SUPPLY_VREFH	Monitor internal supply ADC reference supply.
ADC_IP_INPUTCHAN_SUPPLY_VDD_3V	Monitor internal supply 3.3 V oscillator regulator output.
ADC_IP_INPUTCHAN_SUPPLY_VDD_FLASH_3V	Monitor internal supply 3.3 V flash regulator output.
ADC_IP_INPUTCHAN_SUPPLY_VDD_LV	Monitor internal supply 1.2 V core regulator output.

Definition at line 251 of file `Adc_Ip_Types.h`.

6.2.5 Function Reference

6.2.5.1 Adc_Ip_Init()

```
void Adc_Ip_Init (
    const uint32 Instance,
    const Adc_Ip_ConfigType *const Config )
```

Initialize ADC module.

This function initializes the ADC module by configuring all available features.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>config</i>	- configuration struct pointer

Returns

void

6.2.5.2 Adc_Ip_DeInit()

```
void Adc_Ip_DeInit (
    const uint32 Instance )
```

Deinitialize ADC module.

This function resets the ADC internal registers to default values.

Parameters

in	<i>Instance</i>	- ADC instance number
----	-----------------	-----------------------

Returns

void

6.2.5.3 Adc_Ip_SetAveraging()

```
void Adc_Ip_SetAveraging (
    const uint32 Instance,
    const boolean AvgEn,
    const Adc_Ip_AvgSelectType AvgSel )
```

Configure averaging.

This function enables averaging and selects the number of conversions to average. The mask parameter should be set using the `Adc_Ip_AvgSelectType` enum elements that have the pattern `ADC_IP_AVG_...` e.g. `ADC_IP_AVERAGE_4_CONV`.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>AvgEn</i>	- enable or disable averaging
in	<i>AvgSel</i>	- selects number of conversions to average

Returns

void

6.2.5.4 Adc_Ip_SetSampleTime()

```
void Adc_Ip_SetSampleTime (  
    const uint32 Instance,  
    const uint8 SampleTime )
```

Configure sample time.

This function sets the sample time for selected ADC instance.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>SampleTime</i>	- sample time

Returns

void

6.2.5.5 Adc_Ip_EnableDma()

```
void Adc_Ip_EnableDma (  
    const uint32 Instance )
```

Enable DMA.

This function enables DMA.

Parameters

in	<i>Instance</i>	- ADC instance number
----	-----------------	-----------------------

Returns

void

6.2.5.6 Adc_Ip_DisableDma()

```
void Adc_Ip_DisableDma (  
    const uint32 Instance )
```

Disable DMA.

This function disables DMA.

Parameters

in	<i>Instance</i>	- ADC instance number
----	-----------------	-----------------------

Returns

void

6.2.5.7 `Adc_Ip_SetTriggerMode()`

```
void Adc_Ip_SetTriggerMode (
    const uint32 Instance,
    const Adc_Ip_TrigType TriggerMode )
```

Switch between hardware and software trigger.

This function enables either hardware or software trigger.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>TriggerMode</i>	- selected trigger mode(sw/hw)

Returns

void

6.2.5.8 `Adc_Ip_SetContinuousMode()`

```
void Adc_Ip_SetContinuousMode (
    const uint32 Instance,
    const boolean ContinuousModeEnable )
```

Switch between continuous and one shot conversion mode.

This function switches between ADC continuous conversion mode and one shot mode.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>ContinuousModeEnable</i>	- mode to set: continuous(TRUE) or one shot(FALSE)

Returns

void

6.2.5.9 `Adc_Ip_SetResolution()`

```
void Adc_Ip_SetResolution (
    const uint32 Instance,
    const Adc_Ip_ResolutionType Resolution )
```

Set ADC module resolution.

This function sets ADC module resolution.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>Resolution</i>	- resolution value

Returns

void

6.2.5.10 `Adc_Ip_SetClockMode()`

```
void Adc_Ip_SetClockMode (
    const uint32 Instance,
    const Adc_Ip_ClockConfigType *const Config )
```

Set the ADC clock values.

This function initializes the ADC clock configuration.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>Config</i>	- the clock configuration

Returns

void

6.2.5.11 Adc_Ip_ConfigChannel()

```
void Adc_Ip_ConfigChannel (
    const uint32 Instance,
    const Adc_Ip_ChانConfigType *const ChanConfig )
```

Configure the selected control channel with the given configuration structure.

When Software Trigger mode is enabled, configuring control channel index 0, implicitly triggers a new conversion on the selected ADC input channel. Therefore, Adc_Ip_ConfigChannel can be used for sw-triggering conversions.

Configuring any control channel while it is actively controlling a conversion (sw or hw triggered) will implicitly abort the on-going conversion.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>ChanConfig</i>	- configuration structure

Returns

void

6.2.5.12 Adc_Ip_SetDisabledChannel()

```
Adc_Ip_StatusType Adc_Ip_SetDisabledChannel (
    const uint32 Instance,
    const uint8 ControlChanIdx,
    const boolean WithTimeout )
```

Disable selected channel.

This function sets the input channel of the selected control channel to ADC_IP_INPUTCHAN_DISABLED. If WithTimeout is TRUE then the function will also wait for the register to be updated. Note: the control channel index is the numeric index of SC1x (e.g. SC1A has index 0), not the input channel.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>ControlChanIdx</i>	- control channel index
in	<i>WithTimeout</i>	- enables loop that checks if the register was updated

Returns

Adc_Ip_StatusType

6.2.5.13 **Adc_Ip_StartConversion()**

```
void Adc_Ip_StartConversion (
    const uint32 Instance,
    Adc_Ip_InputChannelType InputChannel,
    const boolean InterruptEnable )
```

Start a software triggered conversion.

This function starts a software conversion on the selected input channel by writing the values given to the SC1A register. Note: hardware configuration on the control channel with index 0 will be overwritten. Note: This will not work if hardware triggered mode is selected.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>InputChannel</i>	- channel index
in	<i>InterruptEnable</i>	- enables end of conversion interrupt. No effect if ADC_IP_AIEN_INTERRUPT_ENABLE = STD_OFF

Returns

void

6.2.5.14 **Adc_Ip_GetConvActiveFlag()**

```
boolean Adc_Ip_GetConvActiveFlag (
    const uint32 Instance )
```

Read and return conversion active flag status.

Parameters

in	<i>Instance</i>	- ADC instance number
----	-----------------	-----------------------

Returns

conversion active flag status

6.2.5.15 `Adc_Ip_GetChanInterrupt()`

```
boolean Adc_Ip_GetChanInterrupt (
    const uint32 Instance,
    const uint8 ControlChanIdx )
```

Check if selected channel has interrupt set.

This function checks and returns if the selected control channel has the interrupt flag set. Note: the control channel index is the numeric index of SC1x (e.g. SC1A has index 0), not the input channel.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>ControlChanIdx</i>	- control channel index

Returns

TRUE if channel has interrupt set or FALSE otherwise

6.2.5.16 `Adc_Ip_GetConvCompleteFlag()`

```
boolean Adc_Ip_GetConvCompleteFlag (
    const uint32 Instance,
    const uint8 ControlChanIdx )
```

Get the value of conversion complete flag of a channel.

This function returns the value of the conversion complete(COCO) flag of a given channel. Note: the control channel index is the numeric index of SC1x (e.g. SC1A has index 0), not the input channel.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>ControlChanIdx</i>	- control channel index

Returns

value of conversion complete flag

6.2.5.17 Adc_Ip_GetConvData()

```
uint16 Adc_Ip_GetConvData (
    const uint32 Instance,
    const uint8 ControlChanIdx )
```

Get the last result for the selected control channel.

This function retrieves the last conversion result for the selected control channel. This function does no validity check on the result. In order to check if the result is valid, the user must call `Adc_Ip_GetConvCompleteFlag` function before this one. Note: the control channel index is the numeric index of SC1x (e.g. SC1A has index 0), not the input channel.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>ControlChanIdx</i>	- control channel index

Returns

conversion result

6.2.5.18 Adc_Ip_DoCalibration()

```
Adc_Ip_StatusType Adc_Ip_DoCalibration (
    const uint32 Instance )
```

Perform calibration of the ADC module.

This function performs a calibration of the ADC module. The input clock frequency for calibration must be less than or equal to half of the maximum specified frequency (50Mhz) and greater than minimum specified frequency (20Mhz). Please refer to Datasheet for more details

Parameters

in	<i>Instance</i>	- ADC instance number
----	-----------------	-----------------------

Returns

the calibration result

- `ADC_IP_STATUS_SUCCESS`: calibration successful
- `ADC_IP_STATUS_TIMEOUT`: calibration step timed out

6.2.5.19 `Adc_Ip_ClearLatchedTriggers()`

```
Adc_Ip_StatusType Adc_Ip_ClearLatchedTriggers (
    const uint32 Instance )
```

Clear latched triggers under processing.

This function clears all trigger latched flags of the ADC instance. This function must be called after the hardware trigger source for the ADC has been deactivated.

Parameters

in	<i>Instance</i>	- ADC instance number
----	-----------------	-----------------------

Returns

clearing status

- `ADC_IP_STATUS_SUCCESS`: operation successful
- `ADC_IP_STATUS_TIMEOUT`: operation timed out

6.2.5.20 `Adc_Ip_EnableChannelNotification()`

```
void Adc_Ip_EnableChannelNotification (
    const uint32 Instance,
    const uint8 ControlChanIdx )
```

Enable channel notification.

This function enables the notification for the selected channel. Note: the control channel index is the numeric index of SC1x (e.g. SC1A has index 0), not the input channel. Note: It's required to read result data in user notification in order to clear the COCO flags and avoid ISR getting invoked repeatedly

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>ControlChanIdx</i>	- control channel index

Returns

void

6.2.5.21 Adc_Ip_DisableChannelNotification()

```
void Adc_Ip_DisableChannelNotification (
    const uint32 Instance,
    const uint8 ControlChanIdx )
```

Disable channel notification.

This function disables the notification for the selected channel. Note: the control channel index is the numeric index of SC1x (e.g. SC1A has index 0), not the input channel.

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>ControlChanIdx</i>	- control channel index

Returns

void

6.2.5.22 Adc_Ip_ClearTrigErrReg()

```
void Adc_Ip_ClearTrigErrReg (
    const uint32 Instance )
```

Clear all trigger error flags.

This function clears all trigger error flags of the ADC instance.

Parameters

in	<i>Instance</i>	- ADC instance number
----	-----------------	-----------------------

Returns

void

6.2.5.23 Adc_Ip_GetTrigErrReg()

```
uint32 Adc_Ip_GetTrigErrReg (
    const uint32 Instance )
```

Get all trigger error flags.

This function returns all trigger error flags of the ADC instance.

Parameters

in	<i>Instance</i>	- ADC instance number
----	-----------------	-----------------------

Returns

trigger error flags bit-mask

6.2.5.24 `Adc_Ip_GetDataAddress()`

```
uint32 Adc_Ip_GetDataAddress (
    const uint32 Instance,
    const uint8 Index )
```

Return the address of the specified data register.

This function returns the address of the specified data register

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>Index</i>	- ADC channel of the Hw unit

Returns

status:

- value of the address of the data for the specified channel

6.2.5.25 `Adc_Ip_GetChanData()`

```
Adc_Ip_StatusType Adc_Ip_GetChanData (
    const uint32 Instance,
    const Adc_Ip_InputChannelType Channel,
    uint16 *const Result )
```

Get the last result for the selected control channel.

This function retrieves the last conversion result for the selected input channel, by looking which control channel was configured with it. If multiple control channels are configured simultaneously with the same requested input channel, the result of the first control channel found will be returned. If no control channel is configured with the given input channel then `ADC_IP_STATUS_ERROR` will be returned. This function does no validity check on the result. In order to check if the result is valid, the user must call `Adc_Ip_GetConvCompleteFlag` function before this one.

Module Documentation

Parameters

in	<i>Instance</i>	- ADC instance number
in	<i>Channel</i>	- input channel
out	<i>Result</i>	- pointer to the buffer where the result will be written

Returns

status

6.2.5.26 **Adc_Ip_SetSupplyMonitoringEnable_TrustedCall()**

```
void Adc_Ip_SetSupplyMonitoringEnable_TrustedCall (  
    const boolean SupplyEnable )
```

This function enable supply monitoring for the internal channels on SIM registers.

This function enable supply monitoring for the internal channels on SIM registers

Parameters

in	<i>SupplyEnable</i>	Enable (TRUE) or Disable (FALSE) supply monitoring
----	---------------------	----------------------------------------------------

6.2.5.27 **Adc_Ip_ConfigSupplyMonitoringChannel_TrustedCall()**

```
void Adc_Ip_ConfigSupplyMonitoringChannel_TrustedCall (  
    const uint32 SupplyChannel )
```

This function configures the internal channels on on SIM registers.

This function configures the internal channels on on SIM registers

Parameters

in	<i>SupplyChannel</i>	Selected supply channel
----	----------------------	-------------------------

6.2.5.28 **Adc_Ip_ResetSupplyMonitoringChannel_TrustedCall()**

```
void Adc_Ip_ResetSupplyMonitoringChannel_TrustedCall (  
    const uint32 SupplyChannel )
```

```
void )
```

This function resets the muxing for ADC channel on SIM register to reset value.

This function resets the muxing for ADC channel on SIM register to reset value

Parameters

in	<i>none</i>	
----	-------------	--

6.3 Pdb Adc IPL

6.3.1 Detailed Description Pdb HW module.

Pdb IP layer hardware module.

Data Structures

- struct [Pdb_Adc_Ip_PretriggersConfigType](#)
Defines the type of structure for configuring an ADC pretrigger. Implements : Pdb_Adc_Ip_PretriggerConfigType↔_Class. [More...](#)
- struct [Pdb_Adc_Ip_ChanConfigType](#)
Defines the type of structure for configuring a single PDB channel. Implements : Pdb_Adc_Ip_ChanConfigType↔_Class. [More...](#)
- struct [Pdb_Adc_Ip_ConfigType](#)
Defines the type of structure for basic timer in PDB. [More...](#)
- struct [Pdb_Adc_Ip_StateStructType](#)
Structure used to store runtime info. [More...](#)

Types Reference

- typedef void [Pdb_Adc_Ip_SeqErrNotificationType](#)(uint8 ChanIdx, uint16 SeqErrMask)
Defines the sequence error notification header.

Enum Reference

- enum [Pdb_Adc_Ip_LoadValueModeType](#)
Defines the type of value load mode for the PDB module.
- enum [Pdb_Adc_Ip_ClkPrescalerDivType](#)
Defines the type of prescaler divider for the PDB counter clock. Implements : Pdb_Adc_Ip_ClkPrescalerDivType↔_Class.
- enum [Pdb_Adc_Ip_ClkPrescalerMultFactType](#)
Defines the type of the multiplication source mode for PDB.
- enum [Pdb_Adc_Ip_TriggerSrcType](#)
Defines the type of trigger source mode for the PDB.

Function Reference

- void [Pdb_Adc_Ip_Init](#) (const uint32 Instance, const [Pdb_Adc_Ip_ConfigType](#) *const Config)
Initialize PDB module.
- void [Pdb_Adc_Ip_DeInit](#) (const uint32 Instance)
Deinitialize PDB module.
- void [Pdb_Adc_Ip_Enable](#) (const uint32 Instance)
Enable the PDB module.
- void [Pdb_Adc_Ip_Disable](#) (const uint32 Instance)
Disable the PDB module.
- void [Pdb_Adc_Ip_SetTriggerInput](#) (const uint32 Instance, const [Pdb_Adc_Ip_TriggerSrcType](#) Trigger←Source)
Set PDB trigger source.
- void [Pdb_Adc_Ip_SetContinuousMode](#) (const uint32 Instance, const boolean State)
Set PDB mode.
- void [Pdb_Adc_Ip_SwTrigger](#) (const uint32 Instance)
Trigger the PDB with a software trigger.
- uint32 [Pdb_Adc_Ip_GetTimerValue](#) (const uint32 Instance)
Get the current value of the PDB counter.
- void [Pdb_Adc_Ip_LoadRegValues](#) (const uint32 Instance)
Load buffered register values.
- void [Pdb_Adc_Ip_SetModulus](#) (const uint32 Instance, const uint16 ModVal)
Set the period of the counter.
- void [Pdb_Adc_Ip_ConfigAdcPretriggers](#) (const uint32 Instance, const uint8 ChanIdx, const [Pdb_Adc_Ip_PretriggersConf](#) *const Config)
Configure all pretriggers on the selected channel.
- uint32 [Pdb_Adc_Ip_GetAdcPretriggerFlags](#) (const uint32 Instance, const uint8 ChanIdx)
Get ADC pretrigger channel flags from the PDB module.
- void [Pdb_Adc_Ip_ClearAdcPretriggerFlags](#) (const uint32 Instance, const uint8 ChanIdx, const uint16 PretriggMask)
Clear ADC pretrigger channel flags from the PDB module.
- void [Pdb_Adc_Ip_SetAdcPretriggerBackToBack](#) (const uint32 Instance, const uint8 ChanIdx, const uint8 PretriggIdx, const boolean Value)
Set back to back mode for the selected pretrigger.
- void [Pdb_Adc_Ip_SetAdcPretriggerEnable](#) (const uint32 Instance, const uint8 ChanIdx, const uint8 PretriggIdx, const boolean Value)
Enable or disable the selected pretrigger.
- void [Pdb_Adc_Ip_SetAdcPretriggerDelayEnable](#) (const uint32 Instance, const uint8 ChanIdx, const uint8 PretriggIdx, const boolean Value)
Set the delay enable for the selected pretrigger.
- void [Pdb_Adc_Ip_SetAdcPretriggerDelayValue](#) (const uint32 Instance, const uint8 ChanIdx, const uint8 PretriggIdx, const uint16 DelayValue)
Set the pretrigger delay value.
- void [Pdb_Adc_Ip_DisableAndClearPdb](#) (const uint32 Instance)
Disable and clear PDB module.
- void [Pdb_Adc_Ip_ConfigInterChannelBackToBack_TrustedCall](#) (const uint32 Instance, const boolean Inter←ChannelBackToBackEnable)
This function enables the inter-channel back to back mode.

6.3.2 Data Structure Documentation

6.3.2.1 struct Pdb_Adc_Ip_PretriggersConfigType

Defines the type of structure for configuring an ADC pretrigger. Implements : Pdb_Adc_Ip_PretriggerConfigType←_Class.

Definition at line 167 of file Pdb_Adc_Ip_Types.h.

Data Fields

Type	Name	Description
uint8	EnableMask	Mask of enabled pretriggers.
uint8	EnableDelayMask	Mask of pretriggers with delays enabled.
uint8	BackToBackEnableMask	Mask of pretriggers with back to back mode enabled. If enabled, the pretrigger will be activated automatically when the ADC COCO flag corresponding to the previous pretrigger in the chain, is set. The previous pretrigger for pretriggers with index 0, depend on features InstanceBackToBackEnable and InterChannelBackToBackEnable.

6.3.2.2 struct Pdb_Adc_Ip_ChانConfigType

Defines the type of structure for configuring a single PDB channel. Implements : Pdb_Adc_Ip_ChانConfigType←_Class.

Definition at line 180 of file Pdb_Adc_Ip_Types.h.

Data Fields

Type	Name	Description
uint8	ChnIdx	PDB channel index.
Pdb_Adc_Ip_PretriggersConfigType	PretriggersConfig	Pretriggers configuration.
uint16	PretriggerDelays[PDB_DLY_COUNT]	Pretriggers delay array, positional dependent(delay for pretrigger 0 must be at index 0)

6.3.2.3 struct Pdb_Adc_Ip_ConfigType

Defines the type of structure for basic timer in PDB.

Implements : Pdb_Adc_Ip_ConfigType←_Class

Definition at line 192 of file Pdb_Adc_Ip_Types.h.

Data Fields

Type	Name	Description
Pdb_Adc_Ip_LoadValueModeType	LoadValueMode	Select the load mode.
Pdb_Adc_Ip_ClkPrescalerDivType	PrescalerDiv	Select the prescaler divider.
Pdb_Adc_Ip_ClkPrescalerMultFactType	ClkPreMultFactor	Select multiplication factor for prescaler.
Pdb_Adc_Ip_TriggerSrcType	TriggerSource	Select the trigger input source.
boolean	ContinuousModeEnable	Enable the continuous mode.
boolean	DmaEnable	Enable the dma for timer.
uint16	ModValue	Modulus register value.
boolean	InterChannelBackToBackEnable	Enable the inter-channel back to back mode between PDBx CH0 and PDBx CH1 pretriggers, forming a ring (PDBx_CH0_pretrigger7 -> PDBx_CH1_pretrigger0 and PDBx_CH1_pretrigger7 -> PDBx_CH0_pretrigger0). Note: this configuration is dedicated to each PDB instance. Note: must not be enabled simultaneously with instance back to back on any PDB instance.
uint8	NumChans	Number of PDB Channels
const Pdb_Adc_Ip_ChanConfigType *	ChanConfigs	PDB Channel configuration
Pdb_Adc_Ip_SeqErrNotificationType *	SeqErrNotification	Sequence error notification.

6.3.2.4 struct Pdb_Adc_Ip_StateStructType

Structure used to store runtime info.

This structure is used to store PDB runtime info

Implements : [Pdb_Adc_Ip_StateStructType_Class](#)

Definition at line 226 of file [Pdb_Adc_Ip_Types.h](#).

Data Fields

Type	Name	Description
boolean	Init	Check if the driver was initialized.
Pdb_Adc_Ip_SeqErrNotificationType *	SeqErrNotification	Sequence error notification

6.3.3 Types Reference

6.3.3.1 Pdb_Adc_Ip_SeqErrNotificationType

```
typedef void Pdb_Adc_Ip_SeqErrNotificationType(uint8 ChanIdx, uint16 SeqErrMask)
```

Defines the sequence error notification header.

This header is used for sequence error notification callbacks

Implements : Pdb_Adc_Ip_SeqErrNotificationType_Class

Definition at line 161 of file Pdb_Adc_Ip_Types.h.

6.3.4 Enum Reference

6.3.4.1 Pdb_Adc_Ip_LoadValueModeType

```
enum Pdb_Adc_Ip_LoadValueModeType
```

Defines the type of value load mode for the PDB module.

Some timing related registers, such as the MOD, IDLY, CHnDLYm, INTx and POyDLY, buffer the setting values. Only the load operation is triggered. The setting value is loaded from a buffer and takes effect. There are four loading modes to fit different applications. Implements : Pdb_Adc_Ip_LoadValueModeType_Class

Enumerator

PDB_ADC_IP_LOAD_VAL_IMMEDIATELY	Loaded immediately after load operation.
PDB_ADC_IP_LOAD_VAL_AT_MODULO_COUNTER	Loaded when counter hits the modulo after load operation.
PDB_ADC_IP_LOAD_VAL_AT_NEXT_TRIGGER	Loaded when detecting an input trigger after load operation.
PDB_ADC_IP_LOAD_VAL_AT_MODULO_COUNTER_OR_NEXT_TRIGGER	Loaded when counter hits the modulo or detecting an input trigger after load operation.

Definition at line 103 of file Pdb_Adc_Ip_Types.h.

6.3.4.2 Pdb_Adc_Ip_ClkPrescalerDivType

```
enum Pdb_Adc_Ip_ClkPrescalerDivType
```

Defines the type of prescaler divider for the PDB counter clock. Implements : Pdb_Adc_Ip_ClkPrescalerDivType_Class.

Enumerator

PDB_ADC_IP_CLK_PREDIV_BY_1	Counting divided by 1 x prescaler multiplication factor (selected by MULT).
PDB_ADC_IP_CLK_PREDIV_BY_2	Counting divided by 2 x prescaler multiplication factor (selected by MULT).
PDB_ADC_IP_CLK_PREDIV_BY_4	Counting divided by 4 x prescaler multiplication factor (selected by MULT).
PDB_ADC_IP_CLK_PREDIV_BY_8	Counting divided by 8 x prescaler multiplication factor (selected by MULT).
PDB_ADC_IP_CLK_PREDIV_BY_16	Counting divided by 16 x prescaler multiplication factor (selected by MULT).
PDB_ADC_IP_CLK_PREDIV_BY_32	Counting divided by 32 x prescaler multiplication factor (selected by MULT).
PDB_ADC_IP_CLK_PREDIV_BY_64	Counting divided by 64 x prescaler multiplication factor (selected by MULT).
PDB_ADC_IP_CLK_PREDIV_BY_128	Counting divided by 128 x prescaler multiplication factor (selected by MULT).

Definition at line 115 of file Pdb_Adc_Ip_Types.h.

6.3.4.3 Pdb_Adc_Ip_ClkPrescalerMultFactType

```
enum Pdb_Adc_Ip_ClkPrescalerMultFactType
```

Defines the type of the multiplication source mode for PDB.

Selects the multiplication factor of the prescaler divider for the PDB counter clock. Implements : Pdb_Adc_Ip_↵
ClkPrescalerMultFactType_Class

Enumerator

PDB_ADC_IP_CLK_PREMULT_FACT_AS_1	Multiplication factor is 1.
PDB_ADC_IP_CLK_PREMULT_FACT_AS_10	Multiplication factor is 10.
PDB_ADC_IP_CLK_PREMULT_FACT_AS_20	Multiplication factor is 20.
PDB_ADC_IP_CLK_PREMULT_FACT_AS_40	Multiplication factor is 40.

Definition at line 133 of file Pdb_Adc_Ip_Types.h.

6.3.4.4 Pdb_Adc_Ip_TriggerSrcType

```
enum Pdb_Adc_Ip_TriggerSrcType
```

Module Documentation

Defines the type of trigger source mode for the PDB.

Selects the trigger input source for the PDB. The trigger input source can be internal or the software trigger.
Implements : Pdb_Adc_Ip_TriggerSrcType_Class

Enumerator

PDB_ADC_IP_TRIGGER_IN0	Source trigger comes from TRGMUX.
PDB_ADC_IP_SOFTWARE_TRIGGER	Select software trigger.

Definition at line 148 of file Pdb_Adc_Ip_Types.h.

6.3.5 Function Reference

6.3.5.1 Pdb_Adc_Ip_Init()

```
void Pdb_Adc_Ip_Init (
    const uint32 Instance,
    const Pdb_Adc_Ip_ConfigType *const Config )
```

Initialize PDB module.

This function initializes the PDB counter, input triggers and general pretrigger settings. It resets PDB registers and enables the PDB clock. Therefore, it should be called before any other operation. After it is initialized, the PDB can act as a triggered timer, which enables other features in PDB module.

Parameters

in	<i>Instance</i>	- PDB instance number
in	<i>Config</i>	- configuration struct pointer

Returns

void

6.3.5.2 Pdb_Adc_Ip_DeInit()

```
void Pdb_Adc_Ip_DeInit (
    const uint32 Instance )
```

Deinitialize PDB module.

This function resets the PDB internal registers to default values.

When the PDB module is not used. Calling this function would shut down the PDB module and reduce the power consumption.

Note: instance back to back configuration is common between PDB instances 0 and 1 (configures the same register even if configured for either PDB instance) This function disables it, so affects all other instances.

Parameters

in	<i>Instance</i>	- PDB instance number
----	-----------------	-----------------------

Returns

void

6.3.5.3 Pdb_Adc_Ip_Enable()

```
void Pdb_Adc_Ip_Enable (  
    const uint32 Instance )
```

Enable the PDB module.

This function enables the PDB module, counter is on.

Parameters

in	<i>Instance</i>	- PDB instance number
----	-----------------	-----------------------

Returns

void

6.3.5.4 Pdb_Adc_Ip_Disable()

```
void Pdb_Adc_Ip_Disable (  
    const uint32 Instance )
```

Disable the PDB module.

This function disables the PDB module, counter is off.

Parameters

in	<i>Instance</i>	- PDB instance number
----	-----------------	-----------------------

6.3.5.5 Pdb_Adc_Ip_SetTriggerInput()

```
void Pdb_Adc_Ip_SetTriggerInput (
    const uint32 Instance,
    const Pdb_Adc_Ip_TriggerSrcType TriggerSource )
```

Set PDB trigger source.

This function sets the PDB trigger source.

Parameters

in	<i>Instance</i>	- PDB instance number
in	<i>TriggerSource</i>	- trigger source

Returns

void

6.3.5.6 Pdb_Adc_Ip_SetContinuousMode()

```
void Pdb_Adc_Ip_SetContinuousMode (
    const uint32 Instance,
    const boolean State )
```

Set PDB mode.

This function sets the PDB mode to continuous or one shot.

Parameters

in	<i>Instance</i>	- PDB instance number
in	<i>State</i>	- mode to set: continuous(true) or one shot(FALSE)

Returns

void

6.3.5.7 Pdb_Adc_Ip_SwTrigger()

```
void Pdb_Adc_Ip_SwTrigger (
    const uint32 Instance )
```

Trigger the PDB with a software trigger.

This function triggers the PDB with a software trigger. When the PDB is set to use the software trigger as input, calling this function triggers the PDB.

Parameters

in	<i>Instance</i>	- PDB instance number
----	-----------------	-----------------------

Returns

void

6.3.5.8 Pdb_Adc_Ip_GetTimerValue()

```
uint32 Pdb_Adc_Ip_GetTimerValue (  
    const uint32 Instance )
```

Get the current value of the PDB counter.

This function gets the current counter value.

Parameters

in	<i>Instance</i>	- PDB instance number
----	-----------------	-----------------------

Returns

Current PDB counter value.

6.3.5.9 Pdb_Adc_Ip_LoadRegValues()

```
void Pdb_Adc_Ip_LoadRegValues (  
    const uint32 Instance )
```

Load buffered register values.

This function sets the LDOK bit. Writing one to this bit updates the internal registers MOD, IDLY, CHnDLYm and POyDLY with the values written to their buffers. The MOD, IDLY, CHnDLYm and POyDLY take effect according to the load mode settings.

After one is written to the LDOK bit, the values in the buffers of above mentioned registers are not effective and cannot be written until the values in the buffers are loaded into their internal registers. The moment when this happens depends on the value of the LDMOD register. Only when this register is in it's default state(0), the load operation will happen immediately. Please check the reference manual for more information. The LDOK can be written only when the the PDB is enabled or as alone with it. It is automatically cleared either when the values in the buffers are loaded into the internal registers or when the PDB is disabled.

Parameters

in	<i>Instance</i>	- PDB instance number
----	-----------------	-----------------------

Returns

void

6.3.5.10 Pdb_Adc_Ip_SetModulus()

```
void Pdb_Adc_Ip_SetModulus (
    const uint32 Instance,
    const uint16 ModVal )
```

Set the period of the counter.

This function sets the PDB Modulus value. Note: This function writes in an internal buffer. Depending on the value of the LDMOD register, it might be necessary to call `Pdb_Adc_Ip_LoadRegValues` in order to update the value of the register. The value of the register can be changed only when the PDB module is enabled.

Parameters

in	<i>Instance</i>	- PDB instance number
in	<i>ModVal</i>	- modulus value

Returns

void

6.3.5.11 Pdb_Adc_Ip_ConfigAdcPretriggers()

```
void Pdb_Adc_Ip_ConfigAdcPretriggers (
    const uint32 Instance,
    const uint8 ChanIdx,
    const Pdb_Adc_Ip_PretriggersConfigType *const Config )
```

Configure all pretriggers on the selected channel.

This function configures the back to back modes, delay enable and output enable settings for all pretriggers on the selected channel.

Parameters

in	<i>Instance</i>	- PDB instance number
in	<i>ChanIdx</i>	- PDB channel
in	<i>Config</i>	- pretriggers config

Returns

void

6.3.5.12 Pdb_Adc_Ip_GetAdcPretriggerFlags()

```
uint32 Pdb_Adc_Ip_GetAdcPretriggerFlags (
    const uint32 Instance,
    const uint8 ChanIdx )
```

Get ADC pretrigger channel flags from the PDB module.

This function gets all ADC pretrigger flags from the selected channel.

Parameters

in	<i>Instance</i>	- PDB instance number
in	<i>ChanIdx</i>	- PDB channel

Returns

bitmask of channel flags selected by pretriggMask

6.3.5.13 Pdb_Adc_Ip_ClearAdcPretriggerFlags()

```
void Pdb_Adc_Ip_ClearAdcPretriggerFlags (
    const uint32 Instance,
    const uint8 ChanIdx,
    const uint16 PretriggMask )
```

Clear ADC pretrigger channel flags from the PDB module.

This function clears the ADC pretrigger channel flags selected by pretriggMask from channel channel.

Parameters

in	<i>Instance</i>	- PDB instance number
in	<i>ChanIdx</i>	- PDB channel
in	<i>PretriggMask</i>	- ADC pretriggers mask

6.3.5.14 Pdb_Adc_Ip_SetAdcPretriggerBackToBack()

```
void Pdb_Adc_Ip_SetAdcPretriggerBackToBack (
    const uint32 Instance,
    const uint8 ChanIdx,
    const uint8 PretriggIdx,
    const boolean Value )
```

Set back to back mode for the selected pretrigger.

This function sets back to back mode for the selected pretrigger on the given channel.

Parameters

in	<i>Instance</i>	- PDB instance number
in	<i>ChanIdx</i>	- PDB channel
in	<i>PretriggIdx</i>	- ADC pretrigger index
in	<i>Value</i>	- enable(true) or disable(FALSE)

6.3.5.15 Pdb_Adc_Ip_SetAdcPretriggerEnable()

```
void Pdb_Adc_Ip_SetAdcPretriggerEnable (
    const uint32 Instance,
    const uint8 ChanIdx,
    const uint8 PretriggIdx,
    const boolean Value )
```

Enable or disable the selected pretrigger.

This function enables or disables the selected pretrigger on the given channel.

Parameters

in	<i>Instance</i>	- PDB instance number
in	<i>ChanIdx</i>	- PDB channel
in	<i>PretriggIdx</i>	- ADC pretrigger mask
in	<i>Value</i>	- enable(true) or disable(FALSE)

6.3.5.16 Pdb_Adc_Ip_SetAdcPretriggerDelayEnable()

```
void Pdb_Adc_Ip_SetAdcPretriggerDelayEnable (
    const uint32 Instance,
    const uint8 ChanIdx,
    const uint8 PretriggIdx,
    const boolean Value )
```

Set the delay enable for the selected pretrigger.

This function sets the delay enable value for the selected pretrigger on the given channel.

Parameters

in	<i>Instance</i>	- PDB instance number
in	<i>ChanIdx</i>	- PDB channel
in	<i>PretriggIdx</i>	- ADC pretrigger index
in	<i>Value</i>	- enable(true) or disable(FALSE)

6.3.5.17 Pdb_Adc_Ip_SetAdcPretriggerDelayValue()

```
void Pdb_Adc_Ip_SetAdcPretriggerDelayValue (
    const uint32 Instance,
    const uint8 ChanIdx,
    const uint8 PretriggIdx,
    const uint16 DelayValue )
```

Set the pretrigger delay value.

This function sets the pretrigger delay value Note: This function writes in an internal buffer. Depending on the value of the LDMOD register, it might be necessary to call Pdb_Adc_Ip_LoadRegValues in order to update the value of the register. The value of the register can be changed only when the PDB module is enabled.

Parameters

in	<i>Instance</i>	- PDB instance number
in	<i>ChanIdx</i>	- PDB channel
in	<i>PretriggIdx</i>	- ADC pretrigger index
in	<i>DelayValue</i>	- pretrigger delay value

6.3.5.18 Pdb_Adc_Ip_DisableAndClearPdb()

```
void Pdb_Adc_Ip_DisableAndClearPdb (
    const uint32 Instance )
```

Disable and clear PDB module.

This function disables PDB module and clears all channels configuration and status registers. Note: This function does not deinitialize the module.

Parameters

in	<i>Instance</i>	- PDB instance number
----	-----------------	-----------------------

6.3.5.19 Pdb_Adc_Ip_ConfigInterChannelBackToBack_TrustedCall()

```
void Pdb_Adc_Ip_ConfigInterChannelBackToBack_TrustedCall (
    const uint32 Instance,
    const boolean InterChannelBackToBackEnable )
```

This function enables the inter-channel back to back mode.

This function enables the inter-channel back to back mode between PDBx CH0 and PDBx CH1 pre-triggers, forming a ring (PDBx_CH0_pretrigger7 -> PDBx_CH1_pretrigger0 and PDBx_CH1_pretrigger7 -> PDBx_CH0_pretrigger0).

Parameters

in	<i>Instance</i>	Adc instance
in	<i>SoftwarePretrigger</i>	Pre-trigger source selection

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and Vision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2023 NXP B.V.

