

# User Manual

for S32K3 ADC Driver

Document Number: UM34ADCASRR21-11 Rev0000R3.0.0 Rev. 1.0

<b>1 Revision History</b>	<b>2</b>
<b>2 Introduction</b>	<b>3</b>
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
<b>3 Driver</b>	<b>8</b>
3.1 Requirements . . . . .	8
3.2 Driver Design Summary . . . . .	8
3.3 Hardware Resources . . . . .	9
3.4 Deviations from Requirements . . . . .	9
3.5 Driver Limitations . . . . .	19
3.6 Driver usage and configuration tips . . . . .	20
3.6.1 Calibration function . . . . .	20
3.6.2 Self-Test function . . . . .	20
3.6.3 BCTU control mode . . . . .	20
3.6.4 BCTU triggered mode . . . . .	21
3.6.5 DMA transfer . . . . .	22
3.6.6 Injected conversions . . . . .	24
3.6.7 Set Channel Optimization . . . . .	24
3.6.8 Enable/Disable channel . . . . .	25
3.6.9 Conversion Time Once . . . . .	26
3.6.10 Bypass Abort Chain Check . . . . .	26
3.6.11 Set Clock Mode . . . . .	27
3.6.12 Analog Watchdog feature . . . . .	28
3.6.13 Without Interrupts group . . . . .	29
3.6.14 Without Dma group . . . . .	31
3.6.15 External DMA channel feature . . . . .	31
3.6.16 Optimize DMA streaming groups . . . . .	32
3.6.17 TempSense functionality . . . . .	35
3.7 Runtime errors . . . . .	35
3.8 Symbolic Names Disclaimer . . . . .	36
<b>4 Tresos Configuration Plug-in</b>	<b>37</b>
4.1 Module Adc . . . . .	42
4.2 Container AdcConfigSet . . . . .	43
4.3 Container AdcHwUnit . . . . .	43
4.4 Parameter AdcHwUnitId . . . . .	44

4.5 Parameter AdcLogicalUnitId . . . . .	44
4.6 Parameter AdcTransferType . . . . .	45
4.7 Parameter AdcClockSource . . . . .	45
4.8 Parameter AdcPrescale . . . . .	46
4.9 Parameter AdcAltPrescale . . . . .	47
4.10 Parameter AdcCalibrationPrescale . . . . .	47
4.11 Parameter AdcHighSpeedEnable . . . . .	48
4.12 Parameter AdcAltHighSpeedEnable . . . . .	48
4.13 Parameter AdcPowerDownDelay . . . . .	49
4.14 Parameter AdcAltPowerDownDelay . . . . .	49
4.15 Parameter AdcMuxDelay . . . . .	50
4.16 Parameter AdcAutoClockOff . . . . .	51
4.17 Parameter AdcBypassSampling . . . . .	51
4.18 Parameter AdcHwUnitOverwriteEn . . . . .	51
4.19 Parameter AdcPresamplingInternalSignal0 . . . . .	52
4.20 Parameter AdcPresamplingInternalSignal1 . . . . .	52
4.21 Parameter AdcPresamplingInternalSignal2 . . . . .	53
4.22 Parameter AdcHwUnitUsrOffset . . . . .	54
4.23 Parameter AdcHwUnitUsrGain . . . . .	54
4.24 Parameter AdcHwUnitResolution . . . . .	55
4.25 Parameter AdcHwUnitBypassResolution . . . . .	55
4.26 Parameter AdcHwUnitDmaClearSource . . . . .	56
4.27 Parameter AdcSarHwUnitVref . . . . .	56
4.28 Reference AdcDmaChannelId . . . . .	57
4.29 Reference AdcCountingDmaChannelId . . . . .	57
4.30 Reference AdcHwUnitEcucPartitionRef . . . . .	59
4.31 Container SdAdcHwUnitSpecificConfiguration . . . . .	59
4.32 Parameter SdadcDecimaRate . . . . .	60
4.33 Parameter SdadcOutputSetDelay . . . . .	60
4.34 Parameter SdadcFifoThreshold . . . . .	61
4.35 Parameter SdadcCalibSkipped . . . . .	61
4.36 Parameter SdadcCalibAverage . . . . .	62
4.37 Container SdAdcDspssSpecificConfiguration . . . . .	62
4.38 Parameter SdAdcDspssInputThreshold . . . . .	63
4.39 Parameter SdAdcDspssOutputThreshold . . . . .	63
4.40 Parameter SdAdcDspssFIRUpsamplingFactor . . . . .	64
4.41 Parameter SdAdcDspssFIRDownsamplingFactor . . . . .	64
4.42 Parameter SdAdcDspssIIROrder . . . . .	65
4.43 Parameter SdAdcDspssIIRShift . . . . .	65
4.44 Parameter SdAdcDspssCalibrationUse . . . . .	66

4.45 Parameter SdAdcDspssCalibrationGain . . . . .	66
4.46 Parameter SdAdcDspssCalibrationOffset . . . . .	66
4.47 Parameter SdAdcDspssNumberSkippedSamples . . . . .	67
4.48 Container SdAdcDspssFIR Taps . . . . .	67
4.49 Parameter SdAdcDspssFIRCoefficients . . . . .	68
4.50 Container SdAdcDspssIIR Taps . . . . .	68
4.51 Parameter SdAdcDspssIIRCoefficients . . . . .	69
4.52 Container AdcSelfTestThresholdConfiguration . . . . .	69
4.53 Parameter AdcSTAW0RSelfTestHighThresholdValue . . . . .	70
4.54 Parameter AdcSTAW0RSelfTestLowThresholdValue . . . . .	70
4.55 Parameter AdcSTAW1RSelfTestLowThresholdValue . . . . .	70
4.56 Parameter AdcSTAW2RSelfTestLowThresholdValue . . . . .	71
4.57 Parameter AdcSTAW4RSelfTestHighThresholdValue . . . . .	71
4.58 Parameter AdcSTAW5RSelfTestHighThresholdValue . . . . .	72
4.59 Container AdcNormalConvTimings . . . . .	72
4.60 Parameter AdcHardwareAverageEnable . . . . .	73
4.61 Parameter AdcHardwareAverageSelect . . . . .	73
4.62 Parameter AdcSamplingDurationNormal0 . . . . .	74
4.63 Parameter AdcSamplingDurationNormal1 . . . . .	74
4.64 Parameter AdcSamplingDurationNormal2 . . . . .	75
4.65 Container AdcAlternateConvTimings . . . . .	75
4.66 Parameter AdcHardwareAverageEnableAlternate . . . . .	76
4.67 Parameter AdcHardwareAverageSelectAlternate . . . . .	76
4.68 Parameter AdcSamplingDurationAlt0 . . . . .	77
4.69 Parameter AdcSamplingDurationAlt1 . . . . .	77
4.70 Parameter AdcSamplingDurationAlt2 . . . . .	78
4.71 Container AdcChannel . . . . .	78
4.72 Parameter AdcLogicalChannelId . . . . .	79
4.73 Parameter AdcChannelName . . . . .	79
4.74 Parameter AdcChannelId . . . . .	80
4.75 Parameter AdcChannelConvTime . . . . .	81
4.76 Parameter AdcChannelLimitCheck . . . . .	81
4.77 Parameter AdcChannelHighLimit . . . . .	82
4.78 Parameter AdcChannelLowLimit . . . . .	82
4.79 Parameter AdcChannelRangeSelect . . . . .	83
4.80 Parameter AdcChannelRefVoltsrcHigh . . . . .	83
4.81 Parameter AdcChannelRefVoltsrcLow . . . . .	84
4.82 Parameter AdcChannelResolution . . . . .	84
4.83 Parameter AdcChannelSampTime . . . . .	85
4.84 Parameter AdcEnablePresampling . . . . .	85

4.85 Parameter AdcEnableThresholds . . . . .	86
4.86 Parameter AdcWdogNotification . . . . .	86
4.87 Reference AdcThresholdRegister . . . . .	87
4.88 Container AdcGroup . . . . .	87
4.89 Parameter AdcGroupAccessMode . . . . .	88
4.90 Parameter AdcGroupConversionMode . . . . .	88
4.91 Parameter AdcGroupConversionType . . . . .	89
4.92 Parameter AdcGroupId . . . . .	89
4.93 Parameter AdcGroupPriority . . . . .	90
4.94 Parameter AdcGroupReplacement . . . . .	90
4.95 Parameter AdcGroupTriggSrc . . . . .	91
4.96 Parameter AdcHwTrigSignal . . . . .	91
4.97 Parameter AdcHwTrigTimer . . . . .	92
4.98 Parameter AdcNotification . . . . .	92
4.99 Parameter AdcExtraNotification . . . . .	93
4.100 Parameter AdcStreamingBufferMode . . . . .	93
4.101 Parameter AdcEnableOptimizeDmaStreamingGroups . . . . .	94
4.102 Parameter AdcEnableHalfInterrupt . . . . .	94
4.103 Parameter AdcStreamingNumSamples . . . . .	95
4.104 Parameter AdcStreamResultGroup . . . . .	95
4.105 Parameter AdcEnableChDisableChGroup . . . . .	96
4.106 Parameter AdcWithoutInterrupts . . . . .	97
4.107 Parameter AdcWithoutDma . . . . .	97
4.108 Parameter AdcExtDMAChanEnable . . . . .	98
4.109 Reference AdcGroupHwTriggerSource . . . . .	98
4.110 Reference AdcGroupDefinition . . . . .	99
4.111 Reference AdcGroupEcucPartitionRef . . . . .	99
4.112 Container AdcGroupConversionConfiguration . . . . .	100
4.113 Parameter AdcGroupHardwareAverageEnable . . . . .	100
4.114 Parameter AdcGroupHardwareAverageSelect . . . . .	100
4.115 Parameter AdcSamplingDuration0 . . . . .	101
4.116 Parameter AdcSamplingDuration1 . . . . .	102
4.117 Parameter AdcSamplingDuration2 . . . . .	102
4.118 Container AdcAlternateGroupConvTimings . . . . .	102
4.119 Parameter AdcGroupAltHardwareAverageEnable . . . . .	103
4.120 Parameter AdcGroupAltHardwareAverageSelect . . . . .	103
4.121 Parameter AdcAltGroupSamplingDuration0 . . . . .	104
4.122 Parameter AdcAltGroupSamplingDuration1 . . . . .	104
4.123 Parameter AdcAltGroupSamplingDuration2 . . . . .	105
4.124 Container AdcThresholdControl . . . . .	105

4.125 Parameter AdcThresholdControlRegister . . . . .	106
4.126 Parameter AdcHighThreshold . . . . .	106
4.127 Parameter AdcLowThreshold . . . . .	107
4.128 Container AdcHwTrigger . . . . .	107
4.129 Parameter AdcHwTrigSrc . . . . .	108
4.130 Container BctuHwUnit . . . . .	109
4.131 Parameter BctuHwUnitId . . . . .	109
4.132 Parameter BctuLogicalUnitId . . . . .	110
4.133 Parameter BctuLowPowerMode . . . . .	110
4.134 Parameter BctuGlobalHwTriggers . . . . .	110
4.135 Parameter BctuNewDataDMAEnableMask . . . . .	111
4.136 Parameter BctuFifoDmaRawData . . . . .	111
4.137 Parameter BctuTriggerNotification . . . . .	112
4.138 Container BctuInternalTrigger . . . . .	112
4.139 Parameter BctuTriggerLoop . . . . .	113
4.140 Parameter BctuDataDestination . . . . .	113
4.141 Parameter BctuHwTriggerEnable . . . . .	113
4.142 Parameter BctuTriggerConversionMode . . . . .	114
4.143 Parameter BctuAdcTargetMask . . . . .	114
4.144 Parameter BctuConversionListStartIndex . . . . .	115
4.145 Reference BctuTriggerSource . . . . .	115
4.146 Reference BctuAdcChannelSingle . . . . .	116
4.147 Container BctuAdcNotifications . . . . .	116
4.148 Parameter BctuAdcNewDataNotification . . . . .	117
4.149 Parameter BctuDataOverrunNotification . . . . .	117
4.150 Parameter BctuListLastConversionNotification . . . . .	118
4.151 Reference BctuAdcNotificationsAdcIndex . . . . .	118
4.152 Container BctuListItems . . . . .	118
4.153 Parameter BctuAdcChannelList . . . . .	120
4.154 Parameter BctuNextChannelWaitOnTrig . . . . .	121
4.155 Parameter BctuLastChannel . . . . .	122
4.156 Container BctuResultFifos . . . . .	122
4.157 Parameter BctuResultFifoIndex . . . . .	122
4.158 Parameter BctuWatermarkValue . . . . .	123
4.159 Parameter BctuFifoNotificationsEnable . . . . .	123
4.160 Parameter BctuWatermarkNotification . . . . .	124
4.161 Parameter BctuUnderrunNotification . . . . .	124
4.162 Parameter BctuOverrunNotification . . . . .	125
4.163 Parameter BctuFifoDmaEnable . . . . .	125
4.164 Parameter BctuFifoDmaBuffer . . . . .	126

4.165 Reference BctuFifoDmaChannelId . . . . .	126
4.166 Container AdcGeneral . . . . .	126
4.167 Parameter AdcDeInitApi . . . . .	128
4.168 Parameter AdcDevErrorDetect . . . . .	128
4.169 Parameter AdcEnableLimitCheck . . . . .	129
4.170 Parameter AdcEnableQueuing . . . . .	129
4.171 Parameter AdcPriorityQueueMaxDepth . . . . .	130
4.172 Parameter AdcEnableStartStopGroupApi . . . . .	130
4.173 Parameter AdcGrpNotifCapability . . . . .	131
4.174 Parameter AdcHwTriggerApi . . . . .	131
4.175 Parameter AdcPriorityImplementation . . . . .	132
4.176 Parameter AdcReadGroupApi . . . . .	133
4.177 Parameter AdcResultAlignment . . . . .	133
4.178 Parameter AdcVersionInfoApi . . . . .	134
4.179 Parameter AdcLowPowerStatesSupport . . . . .	134
4.180 Parameter AdcPowerStateAsynchTransitionMode . . . . .	135
4.181 Reference AdcEcucPartitionRef . . . . .	135
4.182 Reference AdcKernelEcucPartitionRef . . . . .	135
4.183 Container AdcPowerStateConfig . . . . .	136
4.184 Parameter AdcPowerState . . . . .	136
4.185 Parameter AdcPowerStateReadyCbkJRef . . . . .	137
4.186 Container AdcInterrupt . . . . .	137
4.187 Parameter AdcInterruptSource . . . . .	138
4.188 Parameter AdcInterruptEnable . . . . .	138
4.189 Container AdcPublishedInformation . . . . .	139
4.190 Parameter AdcChannelValueSigned . . . . .	139
4.191 Parameter AdcGroupFirstChannelFixed . . . . .	140
4.192 Parameter AdcMaxChannelResolution . . . . .	140
4.193 Container CommonPublishedInformation . . . . .	141
4.194 Parameter ArReleaseMajorVersion . . . . .	141
4.195 Parameter ArReleaseMinorVersion . . . . .	141
4.196 Parameter ArReleaseRevisionVersion . . . . .	142
4.197 Parameter ModuleId . . . . .	142
4.198 Parameter SwMajorVersion . . . . .	143
4.199 Parameter SwMinorVersion . . . . .	143
4.200 Parameter SwPatchVersion . . . . .	144
4.201 Parameter VendorApiInfix . . . . .	144
4.202 Parameter VendorId . . . . .	145
4.203 Container AutosarExt . . . . .	145
4.204 Parameter AdcTimeoutMethod . . . . .	146

4.205 Parameter AdcTimeoutVal . . . . .	146
4.206 Parameter AdcSarIpDevErrorDetect . . . . .	147
4.207 Parameter SdadcIpDevErrorDetect . . . . .	147
4.208 Parameter BctuIpDevErrorDetect . . . . .	148
4.209 Parameter SdadcDspssEnable . . . . .	148
4.210 Parameter AdcMulticoreSupport . . . . .	149
4.211 Parameter AdcEnableGroupDependentChannelNames . . . . .	149
4.212 Parameter AdcBypassAbortChainCheck . . . . .	150
4.213 Parameter AdcConvTimeOnce . . . . .	150
4.214 Parameter AdcOptimizeOneShotHwTriggerConversions . . . . .	151
4.215 Parameter AdcOptimizeDmaStreamingGroups . . . . .	151
4.216 Parameter AdcPreSamplingOnce . . . . .	152
4.217 Parameter AdcEnableInitialNotification . . . . .	153
4.218 Parameter AdcEnableDmaTransferMode . . . . .	153
4.219 Parameter AdcUseSoftwareInjectedGroups . . . . .	154
4.220 Parameter AdcUseHardwareNormalGroups . . . . .	154
4.221 Parameter AdcEnableUserModeSupport . . . . .	155
4.222 Parameter AdcSetHwUnitPowerModeApi . . . . .	155
4.223 Parameter AdcEnableChDisableChApi . . . . .	156
4.224 Parameter AdcGetInjectedConvStatusApi . . . . .	156
4.225 Parameter AdcEnableThresholdConfigurationApi . . . . .	157
4.226 Parameter AdcEnableCtuTrigAutosarExtApi . . . . .	157
4.227 Parameter AdcCtuHardwareTriggerOptimization . . . . .	158
4.228 Parameter AdcEnableCtuControlModeApi . . . . .	158
4.229 Parameter CtuEnableDmaTransferMode . . . . .	159
4.230 Parameter AdcEnableWatchdogApi . . . . .	160
4.231 Parameter AdcEnableSetChannel . . . . .	160
4.232 Parameter AdcEnableDualClockMode . . . . .	161
4.233 Parameter AdcEnableCalibration . . . . .	161
4.234 Parameter AdcEnableApplyCalibration . . . . .	161
4.235 Parameter AdcEnableSelfTest . . . . .	162
4.236 Parameter AdcEnableReadRawDataApi . . . . .	163
4.237 Parameter AdcEnableGroupStreamingResultReorder . . . . .	163
4.238 Parameter AdcSarEnableTempsenseApi . . . . .	164
4.239 Parameter AdcSarPowerOnTempSense . . . . .	164
4.240 Parameter AdcSarTempSenseVsupply . . . . .	164
4.241 Reference SdadcDspssInitDmaChannelId . . . . .	165

<b>5 Module Index</b>	<b>166</b>
5.1 Software Specification . . . . .	166



<b>6 Module Documentation</b>	<b>167</b>
6.1 Adc driver	167
6.1.1 Detailed Description	167
6.1.2 Data Structure Documentation	174
6.1.3 Macro Definition Documentation	177
6.1.4 Types Reference	190
6.1.5 Enum Reference	191
6.1.6 Function Reference	195
6.1.7 Variable Documentation	222
6.2 Adc Sar IPL	223
6.2.1 Detailed Description	223
6.2.2 Data Structure Documentation	227
6.2.3 Macro Definition Documentation	234
6.2.4 Types Reference	235
6.2.5 Enum Reference	235
6.2.6 Function Reference	240
6.3 Bctu IPL	262
6.3.1 Detailed Description	262
6.3.2 Data Structure Documentation	264
6.3.3 Macro Definition Documentation	268
6.3.4 Enum Reference	268
6.3.5 Function Reference	269
6.4 IP_SDADC	282
6.4.1 Detailed Description	282
6.4.2 Data Structure Documentation	287
6.4.3 Macro Definition Documentation	292
6.4.4 Types Reference	301
6.4.5 Enum Reference	301
6.4.6 Function Reference	305
6.4.7 Variable Documentation	328



## Chapter 1

### Revision History

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

## Chapter 2

### Introduction

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

This User Manual describes NXP Semiconductor AUTOSAR ADC for S32K3XX. 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:

- s32k310\_mqfp100
- s32k310\_lqfp48
- s32k311\_mqfp100 / MWCT2015S\_mqfp100
- s32k311\_lqfp48
- s32k312\_mqfp100 / MWCT2016S\_mqfp100
- s32k312\_mqfp172 / MWCT2016S\_mqfp172
- s32k314\_mqfp172
- s32k314\_mapbga257
- s32k322\_mqfp100 / MWCT2D16S\_mqfp100
- s32k322\_mqfp172 / MWCT2D16S\_mqfp172

- s32k324\_mqfp172 / MWCT2D17S\_mqfp172
- s32k324\_mapbga257
- s32k341\_mqfp100
- s32k341\_mqfp172
- s32k342\_mqfp100
- s32k342\_mqfp172
- s32k344\_mqfp172
- s32k344\_mapbga257
- s32k394\_mapbga289
- s32k396\_mapbga289
- s32k358\_mqfp172
- s32k358\_mapbga289
- s32k328\_mqfp172
- s32k328\_mapbga289
- s32k338\_mqfp172
- s32k338\_mapbga289
- s32k348\_mqfp172
- s32k348\_mapbga289
- s32m274\_lqfp64
- s32m276\_lqfp64

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

Note: MWCT part numbers contain NXP confidential IP for Qi Wireless Power.

## 2.2 Overview

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

AUTOSAR:

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

## 2.3 About This Manual

This Technical Reference employs the following typographical conventions:

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

Notes and warnings are shown as below:

Note

This is a note.

Warning

This is a warning

## 2.4 Acronyms and Definitions

Term	Definition
ADC	Analog to Digital Converter
API	Application Programming Interface
ASM	Assembler
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CS	Chip Select
CTU	Cross Trigger Unit
BCTU	Body Cross Trigger Unit
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMA	Direct Memory Access
ECU	Electronic Control Unit
FIFO	First In First Out
LSB	Least Significant Bit
MCU	Micro Controller Unit
MIDE	Multi Integrated Development Environment
MSB	Most Significant Bit
N/A	Not Applicable
RAM	Random Access Memory
SIU	Systems Integration Unit
SWS	Software Specification
VLE	Variable Length Encoding
XML	Extensible Markup Language

## 2.5 Reference List

#	Title	Version
1	Specification of ADC Driver	AUTOSAR Release R21-11
2	S32K3XX Reference Manual	Rev. 6, Draft B - 01/2023
3	S32M27x Reference Manual	Rev. 2, Draft A - 02/2022
4	S32K39 and S32K37 Reference Manual	Rev. 2, Draft A - 11/2022
5	S32K3XX Data Sheet	Rev. 6 — 11/2022
6	S32M27x Data Sheet	Rev. 2 RC - 12/2022
7	S32K396 Data Sheet	Rev. 1.1 - 08/2022
8	S32K358_0P14E Mask Set Errata	Rev. 28 - 9/2022
9	S32K396_0P40E Mask Set Errata	Rev. DEC2022 - 12/2022
10	S32K311_0P98C Mask Set Errata	Rev. 6/March/2023 - 3/2023
11	S32K312: Mask Set Errata for Mask 0P09C	Rev. 25/April/2022

#	Title	Version
12	S32K342: Mask Set Errata for Mask 0P97C	Rev. 10 - 11/2022
13	S32K3x4: Mask Set Errata for Mask 0P55A/1P55A	Rev. 14/Oct/2022

## Chapter 3

### 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 three General-purpose ADC :ADC HW Unit 0 (ADC0), ADC HW Unit 1 (ADC1), ADC HW Unit 2(ADC2). The number of ADC hardware units and channels are derivative specific, so please consult the reference manual.

### 3.4 Deviations from Requirements

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

Requirement	Status	Description	Notes
SWS_Adc_00082	N/F	Service name: - IoHwAb_Adc↔ Notification - Syntax: - void IoHw↔ Ab_AdcNotification( 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."
SWS_Adc_00345	N/S	The ADC module's priority mechanism shall allow suspending and resuming of channel group conversions.	Only abort/restart is supported
SWS_Adc_00341	N/S	If the priority mechanism is supported by the hardware: The ADC module shall support the static configuration option ADC_PRIORITY_HW to enable the priority mechanism using only the hardware priority mechanism.	HW prioritization mechanism is not available on platform
SWS_Adc_00339	N/S	If hardware priority mechanism is supported and selected: The ADC module shall allow the mapping of the configured priority levels (0-255) to the available hardware priority levels.	HW prioritization mechanism is not available on platform

Requirement	Status	Description	Notes
SWS_Adc_00337	N/S	The hardware prioritization mechanism shall be used in case of hardware triggered conversion requests.	HW prioritization mechanism is not available on platform
SWS_Adc_00488	N/S	In case development error reporting is activated: The API shall report the DET error ADC_E_POWER_STATE_E_NOT_SUPPORTED in case this API is called with an unsupported power state or the peripheral does not support low power states at all.	Adc_SetPowerState does not have the Adc_PowerStateType parameter. This parameter is passed and checked (for unsupported power state) through the AdcPreparePowerState function that is called before calling Adc_SetPowerState function.
SWS_Adc_00489	N/S	The API shall report a runtime error ADC_E_TRANSITION_NOT_POSSIBLE in case the requested power state cannot be directly reached from the current power state.	ADC hardware supports only two power states (full power and low power) and all transitions are valid; there is no need to report ADC_E_TRANSITION_NOT_POSSIBLE error.
SWS_Adc_00498	N/S	The API shall report a runtime error ADC_E_TRANSITION_NOT_POSSIBLE in case the requested power state cannot be directly reached from the current power state. All asynchronous operation, needed to reach the target power state, can be executed in background in the context of Adc_Main_PowerTransitionManager.	ADC hardware supports only two power states (full power and low power) and all transitions are valid; there is no need to report ADC_E_TRANSITION_NOT_POSSIBLE error.
SWS_Adc_00479	N/S	Service name: - Adc_Main_PowerTransitionManager - Syntax: - void Adc_Main_PowerTransitionManager( void ) - Service ID[hex]: - 0x14 - Description: - This API is cyclically called and supervises the power state transitions, checking for the readiness of the module and issuing the callbacks IoHwAbs_Adc_NotifyReadyForPowerState (see AdcPowerStateReadyCbRef configuration parameter). - Available via: - SchM_Adc.h -	Asynchronous Power State transition mode is not needed. It's not supported by hardware but still available in driver design and code (without any functionalities) since it's an Autosar API.
SWS_Adc_00499	N/S	This API executes any non-immediate action needed to finalize a power state transition requested by <a href="#">Adc_PreparePowerState()</a> .	Asynchronous Power State transition mode is not needed. It's not supported by hardware.
SWS_Adc_00500	N/S	The rate of scheduling shall be defined by Adc MainSchedulePeriod and shall be variable, as the function only needs to be called if a transition has been requested	Asynchronous Power State transition mode is not needed. It's not supported by hardware.
SWS_Adc_00501	N/S	This API shall also issue callback notifications to the eventually registered users (IoHwAbs) as configured, only in case the asynch mode is chosen.	Asynchronous Power State transition mode is not needed. It's not supported by hardware.

Requirement	Status	Description	Notes
SWS_Adc_00502	N/S	In case the ADC module is not initialized, this function shall simply return without any further elaboration. This is needed to avoid to elaborate uninitialized variables. No DET error shall be entered, because this condition can easily be verified during the startup phase (tasks started before the initialization is complete).Rationale↵: during the startup phase it can happen that the OS already schedules tasks, which call main functions, while some modules are not initialised yet. This is no real error condition, although need handling, i.e. returning without execution.Although the transition state monitoring functionality is mandatory, the implementation of this API is optional, meaning that if the HW allows for other ways to deliver notification and watch the transition state the implementation of this function can be skipped.	Asynchronous Power State transition mode is not needed. It's not supported by hardware.
SWS_Adc_00480	N/S	Service name: - IoHwAb_Adc_↵ NotifyReadyForPowerState - Syntax↵: - void IoHwAb_Adc_NotifyReady↵ ForPowerState( void ) - Sync/Async↵: - Synchronous - Reentrancy: - Non Reentrant - Parameters (in): - None - Parameters (inout): - None - Parameters (out): - None - Return value: - None - Description: - The API shall be invoked by the ADC Driver when the requested power state preparation for mode is completed. - Available via: - IoHwAb_Adc.h -	Asynchronous Power State transition mode is not needed. It's not supported by hardware but still available in driver design and code (without any functionalities) since it's an Autosar API.
SWS_Adc_00460	N/S	These requirements are not applicable to this specification.	Not a requirement.

Requirement	Status	Description	Notes
SWS_Adc_91004	N/S	Service name: - Adc_DisableGroupNotification (draft) - Syntax: - void Adc_DisableGroupNotification (Adc_GroupType Group) - Service ID[hex]: - 0x08 - Sync/Async: - Asynchronous - Reentrancy: - Reentrant - Parameters (in): - Group - Numeric ID of requested ADC Channel group. - Parameters (inout): - None - Parameters (out): - None - Return value: - None - Description: - Disables the notification mechanism for the requested ADC Channel group.Tags: atp.Status=draft - Available via: - <a href="#">Adc.h</a> -	[17/05/2019] The function is sync instead of async because the multicore implementation is done with type 2 instead of type 4.
SWS_Adc_91002	N/S	Service name: - Adc_DisableHardwareTrigger (draft) - Syntax: - void Adc_DisableHardwareTrigger (Adc_GroupType Group) - Service ID[hex]: - 0x06 - Sync/Async: - Asynchronous - Reentrancy: - Reentrant - Parameters (in): - Group - Numeric ID of requested ADC Channel group. - Parameters (inout): - None - Parameters (out): - None - Return value: - None - Description: - Disables the hardware trigger for the requested ADC Channel group.Tags: atp.Status=draft - Available via: - <a href="#">Adc.h</a> -	[17/05/2019] The function is sync instead of async because the multicore implementation is done with type 2 instead of type 4.
SWS_Adc_91003	N/S	Service name: - Adc_EnableGroupNotification (draft) - Syntax: - void Adc_EnableGroupNotification (Adc_GroupType Group) - Service ID[hex]: - 0x07 - Sync/Async: - Asynchronous - Reentrancy: - Reentrant - Parameters (in): - Group - Numeric ID of requested ADC Channel group. - Parameters (inout): - None - Parameters (out): - None - Return value: - None - Description: - Enables the notification mechanism for the requested ADC Channel group.Tags: atp.Status=draft - Available via: - <a href="#">Adc.h</a> -	[17/05/2019] The function is sync instead of async because the multicore implementation is done with type 2 instead of type 4.

Requirement	Status	Description	Notes
SWS_Adc_91001	N/S	Service name: - Adc_EnableHardwareTrigger (draft) - Syntax: - void Adc_EnableHardwareTrigger (Adc_GroupType Group) - Service ID[hex]: - 0x05 - Sync/Async: - Asynchronous - Reentrancy: - Reentrant - Parameters (in): - Group - Numeric ID of requested ADC Channel group. - Parameters (inout): - None - Parameters (out): - None - Return value: - None - Description: - Enables the hardware trigger for the requested ADC Channel group. Tags: atp.Status=draft - Available via: - <a href="#">Adc.h</a> -	[17/05/2019] The function is sync instead of async because the multicore implementation is done with type 2 instead of type 4.
SWS_Adc_91000	N/S	Service name: - Adc_SetupResultBuffer (draft) - Syntax: - Std_ReturnType Adc_SetupResultBuffer (Adc_GroupType Group, const Adc_ValueGroupType* DataBufferPtr) - Service ID[hex]: - 0x0c - Sync/Async: - Asynchronous - Reentrancy: - Reentrant - Parameters (in): - Group - Numeric ID of requested ADC channel group. - DataBufferPtr - pointer to result data buffer - Parameters (inout): - None - Parameters (out): - None - Return value: - Std_ReturnType - E_OK: result buffer pointer initialized correctly E_NOT_OK: operation failed or development error occurred - Description: - 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 DataBufferPtr 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. Tags: atp.Status=draft - Available via: - <a href="#">Adc.h</a> -	[17/05/2019] The function is sync instead of async because the multicore implementation is done with type 2 instead of type 4.

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>&amp;#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 -  &amp;#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 -  Parent Container - AdcChannel -  Description - High limit - used for limit checking. -  Multiplicity - 0..1 -  Type - EcucIntegerParamDef -  Range - 0 .. 18446744073709551615 -  &amp;#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: AdcEnable↵  LimitCheck: not available if limit checking is not globally enabled.  AdcChannelLimitCheck: not available if channel specific limit check is not enabled. 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:  "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_00454	N/F	<p>Name - AdcChannelLowLimit -  Parent Container - AdcChannel -  Description - Low limit - used for limit checking. -  Multiplicity - 0..1 -  Type - EcucIntegerParamDef -  Range - 0 .. 18446744073709551615 -  &amp;#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 -  &amp;#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 - &#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↵ : 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

- AdcPreSamplingOnce is always enabled and cannot be modified
- CTR1[TSENSOR\_SEL] field is being overwritten by the last bit of the value provided by user for sampling duration for standard channels(32-63), on platforms that have this field present
- Limited documentation for SDADC and DSPSS
- On S32K39:
  - For ADCSAR:
    - \* Calibration always return E\_NOT\_OK
  - For SDADC, the following features are not supported or implemented:
    - \* Limit checks
    - \* The ADC driver can only read out half of the number indicated by fifo threshold, even if the number of available conversion results in fifo is the threshold value plus 1. For example: if threshold is 7, after 4 reads the fifo empty flag was set

- For DSPSS, the following features are not supported or implemented:
  - \* Optimize DMA streaming groups
  - \* DSPSS thread calibration (the function **DSPSS\_ThreadsCalibrationSet** is not currently used by ADC driver)

## 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 Self-Test function

For safety applications, it is important to verify correct operation at regular intervals. ADC driver provides [Adc\\_SelfTest\(\)](#) API for this purpose to check its components and flags errors, if any are found. The API can be called explicitly by the user at any time after the driver was initialized and while there are no ongoing conversions.

Configuration-time-prerequisites:

- Enable AdcEnableSelfTest in AutosarExt container.

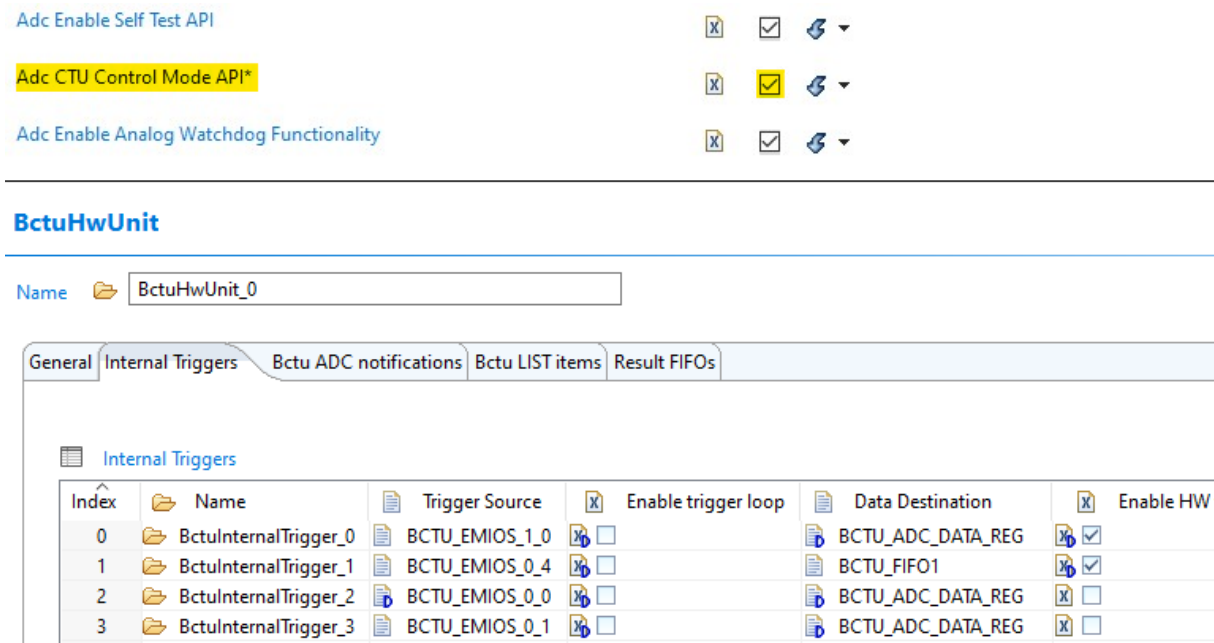
Adc Enable Self Test API



### 3.6.3 BCTU control mode

- Control mode is used by enabling the control mode API from the configuration and by calling the [Adc\\_EnableCtuControlMode\(\)](#) function at runtime.
- In control mode the BCTU assumes complete control over the ADC module and groups can no longer be used. If enabled, triggered mode cannot be used in parallel.
- Software triggered groups can be used on different ADC units which are not controlled by BCTU
- Also SW triggered conversions can be used on the ADC unit controlled by BCTU using [Adc\\_CtuStartConversion\(\)](#)
- In control mode the BCTU hardware is configured as specified directly by the user in the BCTU configuration container and with dedicated APIs at runtime for certain parameters.

- The data can be obtained by calling functions dedicated for reading the FIFOs or data registers from the notifications, or by DMA transfers
- A notification can be configured for each BCTU event.
- The BCTU is configured only once when calling [Adc\\_EnableCtuControlMode\(\)](#) and not every time a hardware trigger is enabled.
- This mode enables the usage of functions prefixed with `Adc_Ctu`, e.g. [Adc\\_CtuReadConvData\(\)](#), [Adc\\_CtuEnableHwTrigger\(\)](#), etc.
- In Tresos, CTU control mode is enabled by ticking its box and this enables the configuration of the BCTU HW unit(see image)



### 3.6.4 BCTU triggered mode

- In this mode the BCTU is not directly configurable by the users. The BCTU configuration is generated based on the group configured information
- Trigger mode is the default mode of triggering (if [Adc\\_EnableCtuControlMode\(\)](#) is not called)
- In Trigger mode the BCTU is used strictly as a trigger source for ADC groups and the result data is read from the adc module.
- This is done by configuring the BCTU list to look exactly like the list of channels in the desired group and activating only the trigger source associated to that group.
- In the end of list interrupt coming from the BCTU, the ADC driver copies and process the adc data from the result registers of the adc module and optionally calls a user notification mapped for the active group.
- The BCTU is reconfigured each time [Adc\\_EnableHardwareTrigger\(\)](#) is called, so it triggers the channels in the same sequence as they are defined in the groups.
- In Tresos, the triggers must be configured from a separate tab.

Adc Group Replacement

ADC\_GROUP\_REPL\_ABORT\_RESTART

Adc Group Trigger Source

ADC\_TRIGG\_SRC\_HW

Adc Group Hw Trigger Source

/Adc/Adc/AdcConfigSet/AdcHwTrigger\_3

---

General

AdcHwUnit

AdcHwTrigger

BctuHwUnit

AdcEcucPartitionRef

AdcPowerStateConfig

AdcInterrupt

Published Information

AdcHwTrigger

Index	Name	Adc Group Hardware Trigger Source
0	AdcHwTrigger_0	EXT_TRIG
1	AdcHwTrigger_1	BCTU_EMIOS_1_0
2	AdcHwTrigger_2	BCTU_EMIOS_0_4
3	AdcHwTrigger_3	BCTU_EMIOS_0_0
4	AdcHwTrigger_4	BCTU_EMIOS_0_1

### 3.6.5 DMA transfer

ADC driver provides the DMA transfer mechanism for transferring the conversion results directly from data registers to system memory via Direct Memory Access (DMA). This is opposed to Interrupt transfer mode, when ADC driver code moves the data from registers to user buffer. This feature can be configured independently for each ADC HW unit. The buffer containing conversion results should be declared within a NO\_CACHEABLE data section.

Configuration-time-prerequisites:

- `AdcEnableDmaTransferMode` needs to be enabled in `AutosarExt` container
- Configured `AdcTransferType` as `ADC_DMA` in `AdcHwUnit` container
- 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 and scatter/gather enabled. The DMA source must be configured as the ADC instance used (`SarAdc`). Also, the number of `ScatterGather` elements configured for the Mcl logical channel must be equal to the maximum number of `AdcChannels` configured in `Adc` groups. The elements in the `ScatterGather` configuration must be linked consecutively (element n linked to element n+1), except the last element in the list that should be marked as such. The other parameters for the `ScatterGather` elements will be overwritten at runtime by the ADC driver.

Logic Channel Configuration

Global Transfer ScatterGather Crc

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\_Adc0DmaTransferComplete

Error Interrupt Callback NULL\_PTR


Enable Global Config ☒ Enable Transfer Config ☐

Enable Scatter/Gather ☒ Enable CRC Config ☐

## Note



- When using multi-core feature, the DMA channel should have the same partition as the ADC module.
- When using Enable Config Time Support feature, if ADC is configured as VariantPreCompile, Mcl cannot be configured as VariantPostBuild.
- `AdcEnableDmaTransferMode` shouldn't be configured simultaneously with `Adc_SetChannel()` or `Adc_EnableChannel()/Adc_DisableChannel()` because there is a risk that DMA TCD size will be modified at runtime.
- When using DMA, the results transferred to buffers are 15-bit wide, same as `Adc CDATA` register, for any resolution selected. If results are accessed directly from buffers or via pointer retrieved by `Adc_GetStreamLastPointer()` results will have the width mentioned previously. If results are retrieved via `Adc_ReadGroup()`, result values are returned having width according to the selected resolution and alignment.
- For groups having only one channel, the driver does not use the Scatter-Gather mode. So Scatter-Gather must be disabled in DMA Channel Configuration.




## Logic Channel




Name  CHANNEL\_FOR\_ADC\_0



---



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\_Adc0DmaTransferCompleteNotificator 

Error Interrupt Callback  NULL\_PTR 

 Ecuc Partition Ref 

Enable Global Config  ☒ 

Enable Transfer Config  ☐ 

Enable Scatter/Gather  ☐ 

### 3.6.6 Injected conversions

Adc driver supports to configure groups with conversion type Injected for high priority conversions. An injected conversion is a conversion chain that can be injected at any time, including into an ongoing normal conversion chain. Injected conversion can only be run in One-Shot mode, and can only interrupt a normal conversion. When an injected chain is inserted into an ongoing normal chain, the ongoing normal channel is suspended and the injected channel request is processed. After completing the last channel in the injected chain, normal conversion resumes from the channel at which it was suspended. Depending on the hardware capabilities, a Software Normal Group can be active in the Queue at the same time as a Hardware Injected group, or a Software Injected Group can be active at the same time with a Hardware Normal Group, so that 2 groups can be converted almost in parallel. For more details, please refer to the Reference Manual of the microcontroller, chapter Injected conversion.

Configuration-time-prerequisites:

- In AdcGroup container, configure AdcGroupConversionType as ADC\_CONV\_TYPE\_INJECTED, AdcGroupConversionMode as ADC\_CONV\_MODE\_ONESHOT and the AdcGroupPriority must be 255 (highest)
- If it is a Software group, AdcUseSoftwareInjectedGroups needs to be enabled in AutosarExt container

### 3.6.7 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



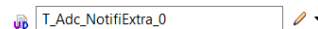
- If channels need to be updated in ISR notification, `AdcEnableInitialNotification` parameter can be enabled in AutosarExt container

#### 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.

#### 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.

### 3.6.8 Enable/Disable channel

The ADC driver provides optional functions `Adc_EnableChannel()` and `Adc_DisableChannel()` to enable/disable one ADC channel of a given software triggered (non-injected) ADC group.

### Configuration-time-prerequisites:

- The Adc Enable/Disable Channels API parameter needs to be enabled in AutosarExt container

#### Adc Enable/Disable Channels API



### Note

- The user should use generated symbolic channel name defines (e.g. `AdcChannel_0_0`), because the API assumes `ChannelId` to have the following format: logical channel ID on bits until position defined by `ADC_CHANNEL_SYMBOLIC_NAME_SHIFT_HW_UNIT_ID_U16`, and logical unit ID from that position
- The driver will not update the values in result buffers corresponding to disabled channels, keeping in the buffer the last results from when the channel was enabled.
- `Adc Enable/Disable Channels API` and `Adc Set Channel API` cannot be enabled simultaneously

### 3.6.9 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
- Configure the values will be used in `AdcNormalConvTiming` or `AdcNormalAlternateTiming` (if `Adc_SetClockMode()` was invoked before with Alternate mode) in `AdcHwUnit` container

### Note

- The `AdcPreSamplingOnce` feature is always enabled from configurator but readonly. `Adc_Init()` will initialize once the presample for all channels in `AdcChannel` container per unit.

### 3.6.10 Bypass Abort Chain Check

When a conversion chain is aborted, ADC driver will block program execution to wait for the conversion chain to be stopped. This wait can be skipped by enabling `AdcBypassAbortChainCheck`. Doing this will improve performance at the cost of HW-SW coherency no longer being guaranteed. In this case, the user must make sure he does not call an ADC service before the hardware reaches the correct state.

Configuration-time-prerequisites:

- Enable `AdcBypassAbortChainCheck` in `AutosarExt` container

`Adc Bypass Abort Chain Check`



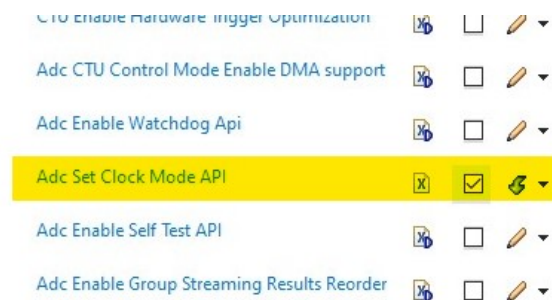
### Note

This can be enabled if no queues are being used.

### 3.6.11 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 power down delay, 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:



- 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.

#### AdcHwUnit

Name AdcHwUnit\_0

General | AdcChannel | AdcGroup | AdcThresholdControl | AdcHwUnitEcucPartitionRef

Adc Hardware Unit	ADC0
Adc Logical Unit ID	0
Adc Transfer Type	ADC_INTERRUPT
Select Dma Channel	
Select Adc Streaming Dma Channel	
Adc Source Clock	CLK_SRC_0
Adc Prescaler Value	1
Adc Alternate Prescale	2
Adc Calibration Prescale	2
Adc Power Down Delay (0 -> 255)	15
Adc Alternate Power Down Delay (0 -> 255)	15
Adc Mux Delay (0 -> 65535)	15
Adc Auto Clock Off	<input type="checkbox"/>
Adc Bypass Sampling	<input type="checkbox"/>
Adc Presampling channel (0 - 31)	VREFL

▼

AdcAlternateConvTimings

Name

AdcAlternateConvTimings

---

Adc Alternate Hardware Average Enable	<div> <div></div> <div></div> <div></div> </div>
Adc Alternate Hardware Average Select	<div> <div></div> <div>SAMPLES_4</div> <div>▼</div> </div>
Adc Unit Alternate Sampling Duration 0 (8 -> 255)	<div> <div></div> <div>8</div> <div></div> </div>
Adc Unit Alternate Sampling Duration 1 (8 -> 255)	<div> <div></div> <div>12</div> <div></div> </div>
Adc Unit Alternate Sampling Duration 2 (8 -> 255)	<div> <div></div> <div>13</div> <div></div> </div>

- 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	<div> <div></div> <div></div> <div></div> </div>
Adc Group Alternate Hardware Average Select	<div> <div></div> <div>SAMPLES_4</div> <div>▼</div> </div>
Adc Group Alternate Sampling Duration 0 (8 -> 255)	<div> <div></div> <div>11</div> <div></div> </div>
Adc Group Alternate Sampling Duration 1 (8 -> 255)	<div> <div></div> <div>14</div> <div></div> </div>
Adc Group Alternate Sampling Duration 2 (8 -> 255)	<div> <div></div> <div>13</div> <div></div> </div>

### 3.6.12 Analog Watchdog feature

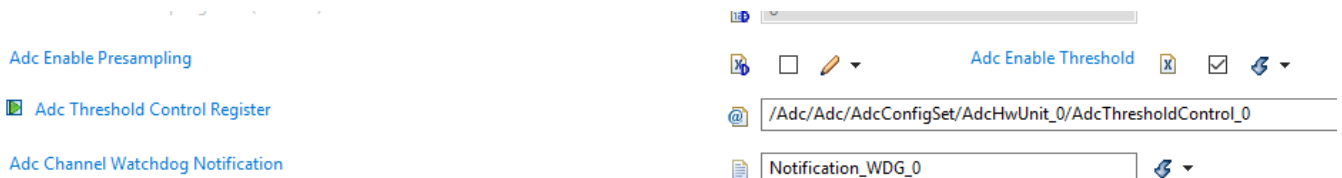
ADC driver provides the Analog Watchdog feature for determining whether the result of a conversion lies within a given guard area specified by an upper and a lower threshold value named THR<sub>H</sub> and THR<sub>L</sub>, respectively. The upper/lower values are set at configuration time and can be changed at runtime via the API [Adc\\_ConfigureThreshold\(\)](#). Each channel can be configured to use this feature or not. If the result of a conversion does not lie within the configured range, an interrupt will occur and a user configured notification will be called. For more details about Analog Watchdog feature see the reference manual of the microcontroller. Adc driver also provide optional functions to enable/disable the analog watchdog event notification of each configured ADC channel: [Adc\\_EnableWdgNotification\(\)](#) and [Adc\\_DisableWdgNotification\(\)](#)

Configuration-time-prerequisites:

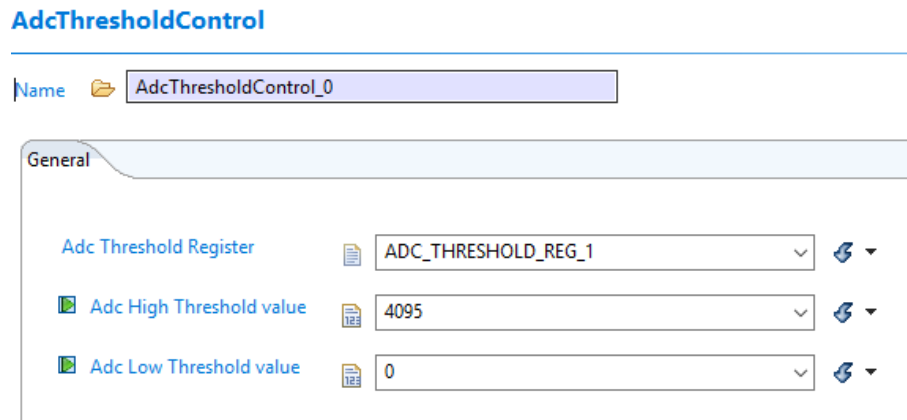
- To use the Analog Watchdog feature, AdcEnableWatchdogFunctionality needs to be enabled in AutosarExt container first:

Adc CTO Control Mode API	<div> <div></div> <div></div> <div></div> </div>
Adc Enable Analog Watchdog Functionality	<div> <div></div> <div></div> <div></div> </div>
Adc Use Hardware Normal Groups	<div> <div></div> <div></div> <div></div> </div>

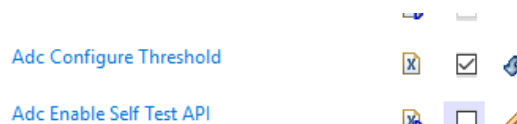
- Configure the channel to use Analog Watchdog feature in AdcChannel container. Enable AdcEnableThresholds and configure the AdcThresholdRegister and AdcWdogNotification:



- The upper/lower threshold values need to be configured in AdcThresholdControl container:



- To use `Adc_ConfigureThreshold()`, `AdcEnableThresholdConfiguration` in AutosarExt container needs to be enabled:



#### Note

- When using `Adc_EnableWdgNotification()` and `Adc_DisableWdgNotification()`, symbolic channel name defines should be used (e.g. `AdcChannel_0_0`), because these functions assumes that `ChannelId` has the following format: logical channel ID on bits until position defined by `ADC_CHANNEL_SYMBOLIC_NAME_SHIFT_HW_UNIT_ID_U16`, and logical unit ID from that position
- When using `Adc_ConfigureThreshold()`, symbolic threshold control name defines should be used (e.g. `AdcThresholdControl_0_0`), because the function assumes that `ThresholdControlIndex` has the following format: logical threshold ID on bits until position defined by `ADC_THRESHOLD_SYMBOLIC_NAME_SHIFT_HW_UNIT_ID_U16`, and logical unit ID from that position

### 3.6.13 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:

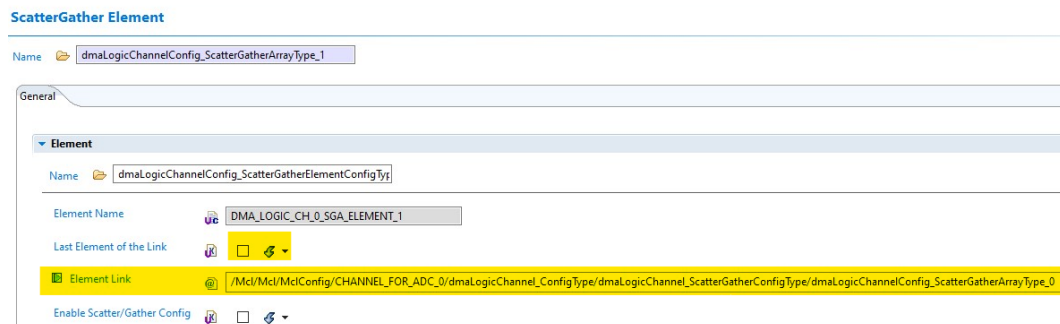
Adc Group Without Interrupts




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

- The Without Interrupt optimization is also supported for groups configured with `ADC_ACCESS_MODE_STREAMING`, with following restrictions: HW unit transfer type is `DMA`, group buffer mode is `CIRCULAR` and group has enabled Streaming Results Reorder (requires Group Streaming Results Reorder to be enabled in General AutosarExt). In this case a second DMA channel is required to be configured (named Select Adc Streaming Dma Channel) and the results are transferred in 2 stages:
  1. Conversion results for 1 sample of all channels are transferred into an internal buffer via the first DMA channel configured with ScatterGather. IMPORTANT NOTE: this channel must be configured as described in [DMA transfer](#), with only one difference: the last ScatterGather element in the array must be linked to the first one, creating a circular scatter gather list. This introduces the limitation that on the same HW unit groups having without interrupt `DISABLED` cannot be used. Also all groups having without interrupt `ENABLED` must have the same number of channels as elements in ScatterGather list.




2. Transfer complete of the last scatter gather element (when last ADC result for the current sample is available) triggers Streaming Dma Channel which transfers results for 1 sample from the internal buffer into the buffer registered by user via `Adc_SetupResultBuffer()`. The ADC driver configures the DMA channel to also support Streaming Number Samples > 1. The configuration required for the Streaming Dma Channel is shown in picture below:


Name  COUNTING\_CHANNEL\_FOR\_ADC\_0





Logic Channel Configuration Global Transfer ScatterGather

**▼ dmaLogicChannel\_GlobalConfigType**


Name  dmaLogicChannel\_GlobalConfigType





**▼ Control**



Name  dmaLogicChannelConfig\_GlobalControlType



Enable Master Id Replication  ☐  Enable Buffered Writes  ☐ 



**▼ Request**

Name  dmaLogicChannelConfig\_GlobalRequestType

Enable DMAMUX Trigger  ☐  Enable DMAMUX Source  ☒ 

DMAMUX0 Source  DMA\_IP\_REQ\_MUX0\_DISABLED 

DMAMUX1 Source  DMA\_IP\_REQ\_MUX1\_DISABLED 

Enable DMA Request  ☐ 

- For hardware triggered groups configured with `ADC_ACCESS_MODE_SINGLE`, multiple channels and transfer type set to DMA, to support multiple occurrences of a trigger, the ScatterGather elements need to be configured the same as for `ADC_ACCESS_MODE_STREAMING` (described above, at point 1). For software triggered groups this is not mandatory. Multiple calls to [Adc\\_StartGroupConversion\(\)](#) are supported in both cases, because the TCD is reconfigured in [Adc\\_StartGroupConversion\(\)](#).

### 3.6.14 Without Dma group

When this feature is disabled and DMA transfer is used, the ADC driver reconfigures the DMA channel at each call to [Adc\\_StartGroupConversion\(\)](#) or [Adc\\_EnableHardwareTrigger\(\)](#). This feature allows complete bypass of the DMA configuration done by ADC driver for the group, so it will not affect other groups for which the ADC driver has already configured the DMA. This feature can be useful for example when starting a normal software group while an injected hardware trigger with `ADC_ACCESS_MODE_STREAMING` is already enabled - to allow the hardware triggered group to execute correctly, the DMA configuration for the software triggered group must be bypassed to not reconfigure the DMA channel of the active hardware triggered group. The results for the software group configured as Without Dma can be read with [Adc\\_ReadGroup\(\)](#), but the channels of the 2 groups must not be overlapped. Otherwise results will not be read, because the valid bit of the result register will be cleared by DMA transfers. The feature can only be enabled if following conditions are met:

- ADC Unit Transfer Type is `ADC_DMA`
- Only for software triggered, oneshot, single access mode groups
- Group Without Interrupts is enabled

### 3.6.15 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()` (ie: update destination address of TCD, enable DMA hardware request) will not be executed for that group so this will need to be handled by the user in the notification
- 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()`.

### 3.6.16 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 that are configured with `ADC_ACCESS_MODE_STREAMING` option. Instead of having an interrupt after every end of chain group conversion, only one or two interrupts will be raised for a stream of N conversions: one after N/2 conversions, and one after all N conversions are completed. With this feature enabled, if a group has more than one channel, a second DMA channel is required, named streaming DMA channel. In this case, the driver will use 2 DMA channels configured: transferring DMA channel and streaming DMA channel. Transferring DMA channel is used for moving data from ADC data register (CDR) to an intermediate buffer. Streaming DMA acts as a hardware counter which has CITER/BITER value equal to group sample as configured by `AdcStreamingNumSamples` parameter. After every end of chain conversion, ADC hardware will trigger a request to transferring DMA channel to complete its major loop. After that, transferring DMA will trigger streaming DMA by a Major Elink request. Streaming DMA will transfer data from intermediate buffer to user buffer and also decrease DMA CITER count by 1. When all ADC samples are completed, the streaming DMA channel will finish its major loop and automatically disable the hardware request from DMA (this is done by disabling DMA in `Adc_Ipw_Adc0DmaTransferCompleteNotification()`). 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 this feature is enabled with a group having only a single channel, driver will only use transferring DMA channel.
- The driver implementation is similar to configuring groups without interrupt, streaming access mode with result reordered. (Referring to [Without Interrupts group](#)) Configuration-time-prerequisites:
- Enable both Adc Enable DMA support and Adc Optimize DMA Streaming Groups:



<p>Adc Optimize OneShot HwTrigger Conversions</p> <p>Adc Presampling Time Once</p> <p>Adc Global Enable DMA support</p>	<p>Adc Enable Double Buffering Optimization</p> <p>Adc Initial Notification Capability</p> <p>Adc BCTU Control Mode Enable DMA support</p>
---	--

- Configure AdcTransferType as ADC\_DMA type and select transferring DMA channel (and also streaming DMA channel if there is at least one ADC group enabled with Optimize DMA streaming groups with more than one ADC channel configured).

Adc Transfer Type	ADC_DMA
Select Dma Channel	/Mcl/Mcl/MclConfig/CHANNEL_FOR_ADC_2
Select Adc Streaming Dma Channel	/Mcl/Mcl/MclConfig/COUNTING_CHANNEL_FOR_ADC_2

- Transferring DMA channel needs to be enabled DMA MUX source (e.g: DMA\_IP\_REQ\_MUX0\_ADC2) and streaming DMA channel needs to have Interrupt Callback selected (e.g: Adc\_Ipw\_Adc2DmaTransferCompleteNotification()) corresponding to the ADC hardware unit used

<p><b>Logic Channel</b></p> <p>Name: CHANNEL_FOR_ADC_2</p> <p>Logic Channel Configuration: Global   Transfer   ScatterGather</p> <p>Logic Channel Name: DMA_LOGIC_CH_2</p> <p>Hardware Instance: DMA_IP_HW_INST_0</p> <p>Hardware Channel: DMA_IP_HW_CH_2</p> <p>Interrupt Callback: NULL_PTR</p> <p>Error Interrupt Callback: NULL_PTR</p> <p>Ecuc Partition Ref: </p> <p>Enable Global Config: <input checked="" type="checkbox"/> Enable Transfer Config: <input type="checkbox"/></p> <p>Enable Scatter/Gather: <input checked="" type="checkbox"/></p>	<p><b>Logic Channel</b></p> <p>Name: CHANNEL_FOR_ADC_2</p> <p>Logic Channel Configuration: Global   Transfer   ScatterGather</p> <p>Request</p> <p>Name: dmaLogicChannelConfig_GlobalRequestType</p> <p>Enable DMAMUX Trigger: <input type="checkbox"/> Enable DMAMUX Source: <input checked="" type="checkbox"/></p> <p>DMAMUX Source: DMA_IP_REQ_MUX0_ADC2</p> <p>DMAMUX Source: DMA_IP_REQ_MUX1_DISABLED</p> <p>Enable DMA Request: <input checked="" type="checkbox"/></p>
<p><b>Logic Channel</b></p> <p>Name: COUNTING_CHANNEL_FOR_ADC_2</p> <p>Logic Channel Configuration: Global   Transfer   ScatterGather</p> <p>Logic Channel Name: DMA_LOGIC_CH_3</p> <p>Hardware Instance: DMA_IP_HW_INST_0</p> <p>Hardware Channel: DMA_IP_HW_CH_3</p> <p>Interrupt Callback: Adc_Ipw_Adc2DmaTransferCompleteNotification</p> <p>Error Interrupt Callback: NULL_PTR</p> <p>Ecuc Partition Ref: </p> <p>Enable Global Config: <input checked="" type="checkbox"/> Enable Transfer Config: <input type="checkbox"/></p> <p>Enable Scatter/Gather: <input type="checkbox"/></p>	<p><b>Logic Channel</b></p> <p>Name: COUNTING_CHANNEL_FOR_ADC_2</p> <p>Logic Channel Configuration: Global   Transfer   ScatterGather</p> <p>Request</p> <p>Name: dmaLogicChannelConfig_GlobalRequestType</p> <p>Enable DMAMUX Trigger: <input type="checkbox"/> Enable DMAMUX Source: <input type="checkbox"/></p> <p>DMAMUX Source: DMA_IP_REQ_MUX0_DISABLED</p> <p>DMAMUX Source: DMA_IP_REQ_MUX1_DISABLED</p> <p>Enable DMA Request: <input type="checkbox"/></p>

- If a group has multiple channels, transferring DMA channel will be configured in Scatter-Gather mode with the number of elements equal to the number of channels in that group. These elements need to be configured as a ring with no last element configured.

Logic Channel

Name CHANNEL\_FOR\_ADC\_2

Logic Channel Configuration Global Transfer ScatterGather

ScatterGather

Index	Name	Element Name	Last Element of the Link
0	dmaLogicChannelConfig_ScatterGatherArrayType_0	DMA_LOGIC_CH_2_SGA_ELEMENT_0	<input type="checkbox"/>
1	dmaLogicChannelConfig_ScatterGatherArrayType_1	DMA_LOGIC_CH_2_SGA_ELEMENT_1	<input type="checkbox"/>

This introduces the limitation that on the same HW unit groups having without interrupt DISABLED cannot be used.

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

Adc (Adc) Mcl (Mcl)

**dmaLogicInstance\_ConfigType**

Name dmaLogicInstance\_ConfigType\_0

Instance

Logic Instance Name DMA\_LOGIC\_INST\_0

Hardware Instance DMA\_IP\_HW\_INST\_0

Debug ☐

Halt After Error ☐

Global Master ID Replication ☐

Ecuc Partition Ref

Round Robin Channel Arbitration ☐

Global Channel linking ☒

**Dma Crc**

Name dmaLogicInstance\_DmaCrc

Enable Swap Bit ☐

Enable Swap Byte ☐

Enable Global ☐

- At group level, Adc Enable Optimize DMA Streaming Groups (and Adc Group Enable Half Interrupt if half interrupt is used) need to be enabled.

Adc Group Streaming Buffer Mode ADC\_STREAM\_BUFFER\_LINEAR

Adc Group Enable Double Buffering ☒

Adc Group Enable Half Interrupt ☒

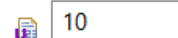
Adc Group Streaming Number Samples 10

- If a group has configured more than one channel, Adc Group Streaming Results Reorder is also required.

Adc Enable Optimize DMA Streaming Groups



Adc Group Streaming Number Samples



Adc Group Streaming Results Reorder



Note

- [Adc\\_SetChannel\(\)](#), [Adc\\_EnableChannel\(\)](#), [Adc\\_DisableChannel\(\)](#) and Optimize one-shot hardware trigger cannot be use concurrently with this feature.

### 3.6.17 TempSense functionality

ADC driver provides an API to interface with the the TempSense module in order to measure the temperature on the chip. The API consists of three functions:

- [Adc\\_TempSenseCalculateTemp\(\)](#) that calculates the temperature based on given data(if read directly using ADC)
- [Adc\\_TempSenseGetTemp\(\)](#) that reads and returns the temperature from the sensor
- [Adc\\_TempSenseSetPowerMode\(\)](#) that turns on or off the module In order to use this functionality, the user must enable `AdcSarEnableTempsenseAPI`. Additionally, the user can configure TempSense to be powered up or down at startup by setting `AdcSarPowerOnTempSense` and modify the voltage supply used for TempSense by updating `AdcSarTempSenseVsupply`.

Configuration-time-prerequisites:

- `AdcSarEnableTempsenseAPI` enabled in AutosarExt container.

Adc Enable Tempsense Api



Adc PowerOn Tempsense



Adc TempSense Voltage Supply (0x0 -> 0x58)



## 3.7 Runtime errors

The driver generates the following DET runtime errors at runtime.

Function	Error Code	Condition triggering the error
<a href="#">Adc_Init()</a>	ADC_E_TIMEOUT	Driver initialization timed out. ADC hardware could not enter power down or idle state.
<a href="#">Adc_DeInit()</a>	ADC_E_TIMEOUT	Timed out. ADC hardware could not enter power down or idle state.
<a href="#">Adc_StartGroupConversion()</a>	ADC_E_TIMEOUT	Timed out. Ongoing conversion could not be aborted.

Function	Error Code	Condition triggering the error
<a href="#">Adc_StopGroupConversion()</a>	ADC_E_TIMEOUT	Timed out. Ongoing conversion could not be aborted.
<a href="#">Adc_SetClockMode()</a>	ADC_E_TIMEOUT	Timed out. ADC hardware could not enter power down or idle state.
<a href="#">Adc_SetClockMode()</a>	ADC_E_UNINIT	Global pointer is NULL.
<a href="#">Adc_SetClockMode()</a>	ADC_E_INVALID_CLOCK_MODE	Clock mode is not correctly set.
<a href="#">Adc_SetHwUnitPowerMode()</a>	ADC_E_TIMEOUT	Timed out. Setting power mode timed out.
<a href="#">Adc_Calibrate()</a>	ADC_E_TIMEOUT	Timed out. Calibration operation timed out.
<a href="#">Adc_SelfTest()</a>	ADC_E_TIMEOUT	Timed out. Self test 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 [AdcHighSpeedEnable](#)
      - Parameter [AdcAltHighSpeedEnable](#)
      - Parameter [AdcPowerDownDelay](#)
      - Parameter [AdcAltPowerDownDelay](#)
      - Parameter [AdcMuxDelay](#)
      - Parameter [AdcAutoClockOff](#)
      - Parameter [AdcBypassSampling](#)
      - Parameter [AdcHwUnitOverwriteEn](#)
      - Parameter [AdcPresamplingInternalSignal0](#)
      - Parameter [AdcPresamplingInternalSignal1](#)
      - Parameter [AdcPresamplingInternalSignal2](#)
      - Parameter [AdcHwUnitUsrOffset](#)
      - Parameter [AdcHwUnitUsrGain](#)
      - Parameter [AdcHwUnitResolution](#)
      - Parameter [AdcHwUnitBypassResolution](#)
      - Parameter [AdcHwUnitDmaClearSource](#)
      - Parameter [AdcSarHwUnitVref](#)
      - Reference [AdcDmaChannelId](#)
      - Reference [AdcCountingDmaChannelId](#)
      - Reference [AdcHwUnitEcucPartitionRef](#)

- Container [SdAdcHwUnitSpecificConfiguration](#)
- Parameter [SdadcDecimaRate](#)
- Parameter [SdadcOutputSetDelay](#)
- Parameter [SdadcFifoThreshold](#)
- Parameter [SdadcCalibSkipped](#)
- Parameter [SdadcCalibAverage](#)
- Container [SdAdcDspssSpecificConfiguration](#)
- Parameter [SdAdcDspssInputThreshold](#)
- Parameter [SdAdcDspssOutputThreshold](#)
- Parameter [SdAdcDspssFIRUpsamplingFactor](#)
- Parameter [SdAdcDspssFIRDownsamplingFactor](#)
- Parameter [SdAdcDspssIIROrder](#)
- Parameter [SdAdcDspssIIRShift](#)
- Parameter [SdAdcDspssCalibrationUse](#)
- Parameter [SdAdcDspssCalibrationGain](#)
- Parameter [SdAdcDspssCalibrationOffset](#)
- Parameter [SdAdcDspssNumberSkippedSamples](#)
- Container [SdAdcDspssFIR Taps](#)
- Parameter [SdAdcDspssFIRCoefficients](#)
- Container [SdAdcDspssIIR Taps](#)
- Parameter [SdAdcDspssIIRCoefficients](#)
- Container [AdcSelfTestThresholdConfiguration](#)
- Parameter [AdcSTAW0RSelfTestHighThresholdValue](#)
- Parameter [AdcSTAW0RSelfTestLowThresholdValue](#)
- Parameter [AdcSTAW1RSelfTestLowThresholdValue](#)
- Parameter [AdcSTAW2RSelfTestLowThresholdValue](#)
- Parameter [AdcSTAW4RSelfTestHighThresholdValue](#)
- Parameter [AdcSTAW5RSelfTestHighThresholdValue](#)
- Container [AdcNormalConvTimings](#)
- Parameter [AdcHardwareAverageEnable](#)
- Parameter [AdcHardwareAverageSelect](#)
- Parameter [AdcSamplingDurationNormal0](#)
- Parameter [AdcSamplingDurationNormal1](#)
- Parameter [AdcSamplingDurationNormal2](#)
- Container [AdcAlternateConvTimings](#)
- Parameter [AdcHardwareAverageEnableAlternate](#)
- Parameter [AdcHardwareAverageSelectAlternate](#)
- Parameter [AdcSamplingDurationAlt0](#)
- Parameter [AdcSamplingDurationAlt1](#)
- Parameter [AdcSamplingDurationAlt2](#)
- 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](#)
- Parameter [AdcEnablePresampling](#)
- Parameter [AdcEnableThresholds](#)
- Parameter [AdcWdogNotification](#)
- Reference [AdcThresholdRegister](#)
- 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](#)
- Reference [AdcGroupHwTriggerSource](#)
- Reference [AdcGroupDefinition](#)
- Reference [AdcGroupEcucPartitionRef](#)
- Container [AdcGroupConversionConfiguration](#)
- Parameter [AdcGroupHardwareAverageEnable](#)
- Parameter [AdcGroupHardwareAverageSelect](#)
- Parameter [AdcSamplingDuration0](#)
- Parameter [AdcSamplingDuration1](#)
- Parameter [AdcSamplingDuration2](#)
- Container [AdcAlternateGroupConvTimings](#)
- Parameter [AdcGroupAltHardwareAverageEnable](#)
- Parameter [AdcGroupAltHardwareAverageSelect](#)
- Parameter [AdcAltGroupSamplingDuration0](#)
- Parameter [AdcAltGroupSamplingDuration1](#)
- Parameter [AdcAltGroupSamplingDuration2](#)
- Container [AdcThresholdControl](#)

- Parameter [AdcThresholdControlRegister](#)
- Parameter [AdcHighThreshold](#)
- Parameter [AdcLowThreshold](#)
- \* Container [AdcHwTrigger](#)
  - Parameter [AdcHwTrigSrc](#)
- \* Container [BctuHwUnit](#)
  - Parameter [BctuHwUnitId](#)
  - Parameter [BctuLogicalUnitId](#)
  - Parameter [BctuLowPowerMode](#)
  - Parameter [BctuGlobalHwTriggers](#)
  - Parameter [BctuNewDataDMAEnableMask](#)
  - Parameter [BctuFifoDmaRawData](#)
  - Parameter [BctuTriggerNotification](#)
  - Container [BctuInternalTrigger](#)
  - Parameter [BctuTriggerLoop](#)
  - Parameter [BctuDataDestination](#)
  - Parameter [BctuHwTriggerEnable](#)
  - Parameter [BctuTriggerConversionMode](#)
  - Parameter [BctuAdcTargetMask](#)
  - Parameter [BctuConversionListStartIndex](#)
  - Reference [BctuTriggerSource](#)
  - Reference [BctuAdcChannelSingle](#)
  - Container [BctuAdcNotifications](#)
  - Parameter [BctuAdcNewDataNotification](#)
  - Parameter [BctuDataOverrunNotification](#)
  - Parameter [BctuListLastConversionNotification](#)
  - Reference [BctuAdcNotificationsAdcIndex](#)
  - Container [BctuListItems](#)
  - Parameter [BctuAdcChannelList](#)
  - Parameter [BctuNextChannelWaitOnTrig](#)
  - Parameter [BctuLastChannel](#)
  - Container [BctuResultFifos](#)
  - Parameter [BctuResultFifoIndex](#)
  - Parameter [BctuWatermarkValue](#)
  - Parameter [BctuFifoNotificationsEnable](#)
  - Parameter [BctuWatermarkNotification](#)
  - Parameter [BctuUnderrunNotification](#)
  - Parameter [BctuOverrunNotification](#)
  - Parameter [BctuFifoDmaEnable](#)
  - Parameter [BctuFifoDmaBuffer](#)
  - Reference [BctuFifoDmaChannelId](#)
- 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 [AdcPowerStateReadyCbRef](#)
- 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 [AdcSarIpDevErrorDetect](#)
  - \* Parameter [SdadIpDevErrorDetect](#)
  - \* Parameter [BctuIpDevErrorDetect](#)
  - \* Parameter [SdadDspssEnable](#)
  - \* Parameter [AdcMulticoreSupport](#)

- \* Parameter [AdcEnableGroupDependentChannelNames](#)
- \* Parameter [AdcBypassAbortChainCheck](#)
- \* Parameter [AdcConvTimeOnce](#)
- \* Parameter [AdcOptimizeOneShotHwTriggerConversions](#)
- \* Parameter [AdcOptimizeDmaStreamingGroups](#)
- \* Parameter [AdcPreSamplingOnce](#)
- \* Parameter [AdcEnableInitialNotification](#)
- \* Parameter [AdcEnableDmaTransferMode](#)
- \* Parameter [AdcUseSoftwareInjectedGroups](#)
- \* Parameter [AdcUseHardwareNormalGroups](#)
- \* Parameter [AdcEnableUserModeSupport](#)
- \* Parameter [AdcSetHwUnitPowerModeApi](#)
- \* Parameter [AdcEnableChDisableChApi](#)
- \* Parameter [AdcGetInjectedConvStatusApi](#)
- \* Parameter [AdcEnableThresholdConfigurationApi](#)
- \* Parameter [AdcEnableCtuTrigAutosarExtApi](#)
- \* Parameter [AdcCtuHardwareTriggerOptimization](#)
- \* Parameter [AdcEnableCtuControlModeApi](#)
- \* Parameter [CtuEnableDmaTransferMode](#)
- \* Parameter [AdcEnableWatchdogApi](#)
- \* Parameter [AdcEnableSetChannel](#)
- \* Parameter [AdcEnableDualClockMode](#)
- \* Parameter [AdcEnableCalibration](#)
- \* Parameter [AdcEnableApplyCalibration](#)
- \* Parameter [AdcEnableSelfTest](#)
- \* Parameter [AdcEnableReadRawDataApi](#)
- \* Parameter [AdcEnableGroupStreamingResultReorder](#)
- \* Parameter [AdcSarEnableTempsenseApi](#)
- \* Parameter [AdcSarPowerOnTempSense](#)
- \* Parameter [AdcSarTempSenseVsupply](#)
- \* Reference [SdadcDspssInitDmaChannelId](#)

### 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](#)
- [BctuHwUnit](#)

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:

- [SdAdcHwUnitSpecificConfiguration](#)
- [AdcSelfTestThresholdConfiguration](#)
- [AdcNormalConvTimings](#)
- [AdcAlternateConvTimings](#)
- [AdcChannel](#)
- [AdcGroup](#)
- [AdcThresholdControl](#)

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', 'ADC2', 'ADC3', 'ADC4', 'ADC5', 'ADC6', 'ADC7_SD0', 'ADC8_SD1', 'ADC9_SD2', 'ADC10_SD3']

## 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

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

## 4.6 Parameter AdcTransferType

Select the Interrupt or Dma transfer Type.

If DMA is required for AdcTransferType, it is recommended to keep AdcWithoutInterrupts as false, otherwise, DMA will not be configured and it will be user responsibility to read the results from registers directly by calling Adc\_ReadGroup.

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.

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	CLK_SRC_0
literals	['CLK_SRC_0']

## 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 module clock frequency.
- 2: ADC module clock frequency / 2.
- 4: ADC module clock frequency / 4.

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	4
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.

- 1: ADC module clock frequency.
- 2: ADC module clock frequency / 2.
- 4: ADC module clock frequency / 4.

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	4
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 module clock frequency.
- 2: ADC module clock frequency / 2.
- 4: ADC module clock frequency / 4.

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

Property	Value
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	2
max	4
min	1

## 4.11 Parameter AdcHighSpeedEnable

Enables/disables high speed conversion or calibration.

The frequency of the conversion clock has to be within the limits defined in the data sheet.

If the module clock frequency is higher than the maximum frequency of the conversion clock allowed during the functional conversion or during the calibration (see the chip data sheet),

then you must configure the ADC conversion clock divider so that the frequency of the conversion clock is within allowed limits.

For some conversion clock frequencies this feature must be enabled. Refer to Reference Manual or data sheet for more details.

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

## 4.12 Parameter AdcAltHighSpeedEnable

Alt configuration: Enables/disables high speed conversion or calibration.

The frequency of the conversion clock has to be within the limits defined in the data sheet.

If the module clock frequency is higher than the maximum frequency of the conversion clock allowed during the functional conversion or during the calibration (see the chip data sheet),



then you must configure the ADC conversion clock divider so that the frequency of the conversion clock is within allowed limits.

For some conversion clock frequencies this feature must be enabled. Refer to Reference Manual or data sheet for more details.

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

### 4.13 Parameter AdcPowerDownDelay

The delay between the power down bit reset and the starting of conversion.

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	15
max	255
min	0

### 4.14 Parameter AdcAltPowerDownDelay

The delay between the power down bit reset and the starting of conversion when ADC runs on low power system frequency.

This node is EDITABLE only if "Adc Set Clock Mode API" is enabled

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	15
max	255
min	0

### 4.15 Parameter AdcMuxDelay

The delay between the external decode signals and the start of the sampling phase.

It is used to take into account the settling time of the external mux when ADC runs on normal system frequency.

The decode signal delay is calculated as  $(DSD \times 1/\text{Frequency of Adc\_Clock})$ .

The DSDR register is 12-bit.

Note: This is an Implementation Specific Parameter.

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	15
max	65535
min	0

## 4.16 Parameter AdcAutoClockOff

Enables/disables the auto-clock-off features.

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

## 4.17 Parameter AdcBypassSampling

Select whether the presampling is followed by the comparison.

Disabled: The presampling will be followed by sampling the selected input.

Enabled: The presampling will be followed by the successive approximation algorithm and the conversion data will be written to the conversion data register of the selected input.

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

## 4.18 Parameter AdcHwUnitOverwriteEn

Specifies whether a conversion data register accepts new data before the current conversion data has been read.

When enabled, the new conversion result overwrites the older data, regardless of the validity of the older conversion data. When  $CDRn[VALID] = 1$ , the conversion data has not been read.

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	true

### 4.19 Parameter AdcPresamplingInternalSignal0

Select the Adc presampling internal voltage for channels 0-31, AdcPresamplingInternalSignal0 shall take the following values:

VREFL - Pre-sample voltage - 0.

VREFH - Pre-sample voltage - 1.

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	VREFL
literals	['VREFL', 'VREFH']

### 4.20 Parameter AdcPresamplingInternalSignal1

Select the Adc presampling internal voltage for channels 32-63, AdcPresamplingInternalSignal0 shall take the follow-

ing values:

VREFL - Pre-sample voltage - 0.

VREFH - Pre-sample voltage - 1.

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	VREFL
literals	['VREFL', 'VREFH']

## 4.21 Parameter AdcPresamplingInternalSignal2

Select the Adc presampling internal voltage for the external channels on current hardware unit:

AdcPresamplingInternalSignal2 shall take the following values:

VREFL - Pre-sample voltage - 0.

VREFH - Pre-sample voltage - 1.

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	VREFL
literals	['VREFL', 'VREFH']

## 4.22 Parameter AdcHwUnitUsrOffset

The ADC can compensate a maximum of  $\pm 15$ LSB at 12b offset by programming OFFSET\_USER field in USROFSGN register.

The field is in 2's complement format and stored with an 8x multiplication. Thus, if a value of  $\pm X$  is stored in this field, it leads to

a final result adjusted with  $\pm (X/8)$  from its unprogrammed version. It is required to keep the 3-LSBs zero to avoid any fractional

subtraction-related errors.

$$\text{ADC\_OUTPUT\_NEW} = \text{ADC\_OUTPUT\_OLD} - (\text{OFFSET\_USER}/8).$$

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

## 4.23 Parameter AdcHwUnitUsrGain

The ADC can compensate a maximum of  $\pm 63$  LSB at 12b gain by programming GAIN\_USER field in USROFSGN register.

The field is in 2's complement format and stored with an 8x multiplication. Thus, if a value of  $\pm Y$  is stored in this field, it leads to a

final result adjusted with  $\pm (Y/8)$  LSB from its unprogrammed version. It is required to keep the 3-LSBs zero to avoid any fractional

subtraction/multiplication related errors.

$$\text{ADC\_OUTPUT\_NEW} = \text{ADC\_OUTPUT\_OLD} * (1 + (\text{GAIN\_USER} / (8 * 4096))).$$

For SDADC, this selects the gain to be applied to the analog input stage.

The effective analog input becomes the input voltage level multiplied by the gain factor.

(Available options are: 1, 2, 4 and 8)

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	65535
min	0

## 4.24 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_14
literals	['RESOLUTION_16', 'RESOLUTION_15', 'RESOLUTION_14', 'RESOLUTION_13', 'RESOLUTION_12', 'RESOLUTION_10', 'RESOLUTION_8']

## 4.25 Parameter AdcHwUnitBypassResolution

Disable resolution processing for results, allowing the API to return the raw result data from CDR register.

Enabling this will allow results read by the API to be on the same bit width as the ones read using DMA, since DMA will always get the data from the CDR register unprocessed.

The resolution of the result will be the same as the one configured above but its number of bits will always be 15.

The threshold values will also need to be given on 15 bits.

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

## 4.26 Parameter AdcHwUnitDmaClearSource

Select the DMA request clearing source.

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	DMA_REQ_CLEAR_ON_ACK
literals	['DMA_REQ_CLEAR_ON_ACK', 'DMA_REQ_CLEAR_ON_READ']

## 4.27 Parameter AdcSarHwUnitVref

Adc Voltage Reference, expressed in fixed point format with 1 bit for the sign, 11 bits for the integer part and 4 bits for the decimal part.

The value must be filled in by the user according to actual board setup. It and will be used for calculating the value measured by TempSense.

Property	Value
type	ECUC-INTEGER-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	53
max	88
min	0

## 4.28 Reference AdcDmaChannelId

ID of the DMA channel used to transfer the data.

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/EcucDefs/Mcl/MclConfig/dmaLogicChannel_Type

## 4.29 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

## Tresos Configuration Plug-in

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

### 4.30 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.

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/EcucPartition

### 4.31 Container SdAdcHwUnitSpecificConfiguration

Vendor specific: This container contains the configuration of the SDADC.

Included subcontainers:

- [SdAdcDspssSpecificConfiguration](#)

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

## 4.32 Parameter SdadcDecimaRate

Vendor specific: This parameter defines the the programmable Decimation Rate to select the over-sampling ratio (OSR).

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	DECIMATION_RATE_1280
literals	['DECIMATION_RATE_120', 'DECIMATION_RATE_140', 'DECIMATION_RATE_160', 'DECIMATION_RATE_180', 'DECIMATION_RATE_200', 'DECIMATION_RATE_220', 'DECIMATION_RATE_240', 'DECIMATION_RATE_280', 'DECIMATION_RATE_320', 'DECIMATION_RATE_360', 'DECIMATION_RATE_376', 'DECIMATION_RATE_400', 'DECIMATION_RATE_440', 'DECIMATION_RATE_480', 'DECIMATION_RATE_560', 'DECIMATION_RATE_640', 'DECIMATION_RATE_720', 'DECIMATION_RATE_800', 'DECIMATION_RATE_880', 'DECIMATION_RATE_960', 'DECIMATION_RATE_1120', 'DECIMATION_RATE_1280', 'DECIMATION_RATE_125']

## 4.33 Parameter SdadcOutputSetDelay

Vendor specific: Defines the delay to qualify the conversion data.

Property	Value
type	ECUC-INTEGER-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	255
max	255
min	0

#### 4.34 Parameter SdadcFifoThreshold

Vendor specific: When the number of datawords in the data FIFO is greater than the value in FTHLD, FIFO full event is flagged.

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	15
min	0

#### 4.35 Parameter SdadcCalibSkipped

Vendor specific: The number of first samples will be skipped to ignore the unstable results before performing actual SDADC calibration.

Property	Value
type	ECUC-INTEGER-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	16
max	255
min	0

## 4.36 Parameter SdadcCalibAverage

Vendor specific: The number of consecutive calibration samples to be read to calculate the average SDADC calibration output.

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	16
max	255
min	1

## 4.37 Container SdAdcDspssSpecificConfiguration

Vendor specific: Dspss configuration for processing (e.g. FIR, FFT) SDADC data.

Included subcontainers:

- [SdAdcDspssFIRTaps](#)
- [SdAdcDspssIIRTaps](#)

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

## 4.38 Parameter SdAdcDspssInputThreshold

Vendor specific: For sample threshold mode, it is also equal to number of samples needed to schedule a thread.

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	20
max	65535
min	1

## 4.39 Parameter SdAdcDspssOutputThreshold

Vendor specific: Defines the number of samples in output buffer required to generate transfer request.

Property	Value
type	ECUC-INTEGER-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	8
max	65535
min	1

#### 4.40 Parameter SdAdcDspssFIRUpsamplingFactor

Vendor specific: Upsampling factor: 1 (no upsampling) or 2.

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	1
max	2
min	1

#### 4.41 Parameter SdAdcDspssFIRDownsamplingFactor

Vendor specific: Downsampling factor.

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



Property	Value
defaultValue	5
max	65535
min	0

#### 4.42 Parameter SdAdcDspssIIROrder

Vendor specific: IIR order.

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	65535
min	0

#### 4.43 Parameter SdAdcDspssIIRShift

Vendor specific: IIR Shift.

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	65535
min	0

#### 4.44 Parameter SdAdcDspssCalibrationUse

Vendor specific: Flag indicating the calibration is used or not.

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	true

#### 4.45 Parameter SdAdcDspssCalibrationGain

Vendor specific: Calibration gain. Values have to be in format Q1.14 (range 0 ? 2 represented in 16-bit signed value).

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	16384
max	32767
min	-32768

#### 4.46 Parameter SdAdcDspssCalibrationOffset

Vendor specific: Calibration offset. Values have to be in format Q1.31 (range -1 ? 1 represented in 32-bit signed value).

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	2147483647
min	-2147483648

#### 4.47 Parameter SdAdcDspssNumberSkippedSamples

Vendor specific: Defines the number of sampled skipped for filter processing.

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	50
max	65535
min	0

#### 4.48 Container SdAdcDspssFIRTap

Vendor specific: FIR filter taps array. Values have to be in format Q1.15 (range -1 ? 1 represented in 16-bit signed value).

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	1
upperMultiplicity	128
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE

## 4.49 Parameter SdAdcDspssFIRCoefficients

Vendor specific: FIR Coefficients.

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	32767
min	-32768

## 4.50 Container SdAdcDspssIIRTaps

Vendor specific: Coefficients are stored in the coeff buffer in order: -A2, -A1, B2, B1, B0. Values have to be in format Q1.15 (range -1 ? 1 represented in 16-bit signed value).

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	5
upperMultiplicity	5
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A

## 4.51 Parameter SdAdcDspssIIRCoefficients

Vendor specific: IIR Coefficients.

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	32767
min	-32768

## 4.52 Container AdcSelfTestThresholdConfiguration

These values define the limits for results measured at each step of the self-test algorithms. If measured values are outside these ranges, the self test fails.

When selecting the values, please take into account ADC resolution and reference voltage used.

For more details please refer to reference manual.

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

### 4.53 Parameter AdcSTAW0RSelfTestHighThresholdValue

SelfTest high threshold value for Algorithm S0.

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	13929
max	65535
min	0

### 4.54 Parameter AdcSTAW0RSelfTestLowThresholdValue

SelfTest low threshold value for Algorithm S0.

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	6784
max	65535
min	0

### 4.55 Parameter AdcSTAW1RSelfTestLowThresholdValue

SelfTest low threshold value for Algorithm S1.

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
default Value	16377
max	65535
min	0

## 4.56 Parameter AdcSTAW2RSelfTestLowThresholdValue

SelfTest low threshold value for Algorithm S2.

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
default Value	16377
max	65535
min	0

## 4.57 Parameter AdcSTAW4RSelfTestHighThresholdValue

SelfTest high threshold value for Algorithm C0. Low threshold value is two's-complement of high threshold value and automatically calculated.

Property	Value
type	ECUC-INTEGER-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	100
max	65535
min	0

## 4.58 Parameter AdcSTAW5RSelfTestHighThresholdValue

SelfTest high threshold value for Algorithm C1 to C11. Low threshold value is two's-complement of high threshold value and automatically calculated

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	100
max	65535
min	0

## 4.59 Container AdcNormalConvTimings

Selects Normal values used for programming CTR Conversion Timing Registers in Adc\_SetClockMode API and also when AdcConvTimeOnce option is enabled.

This node is EDITABLE only if "Adc Conversion Time Once" is enabled

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

## 4.60 Parameter AdcHardwareAverageEnable

Enables the hardware average function of the ADC.

This feature is enabled if "Adc Conversion Time Once" from General/AutosarExt is 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

## 4.61 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
defaultValue	SAMPLES_4
literals	['SAMPLES_4', 'SAMPLES_8', 'SAMPLES_16', 'SAMPLES_32']

## 4.62 Parameter AdcSamplingDurationNormal0

Select the Normal Sampling Duration for channels 0-31 when calling `Adc_SetClockMode(ADC_NORMAL)` for CTR0 register.

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	22
max	255
min	8

## 4.63 Parameter AdcSamplingDurationNormal1

Select the Normal Sampling Duration for channels 32-63 when calling `Adc_SetClockMode(ADC_NORMAL)` for CTR1 register.

Property	Value
type	ECUC-INTEGER-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	22
max	255
min	8

## 4.64 Parameter AdcSamplingDurationNormal2

Select the Normal Sampling Duration for channels 64-95 when calling `Adc_SetClockMode(ADC_NORMAL)` for CTR2 register.

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	22
max	255
min	8

## 4.65 Container AdcAlternateConvTimings

Selects Alternate values used in `Adc_SetClockMode` API for programming CTR Conversion Timing Registers.

This container is EDITABLE only if "Adc Conversion Time Once" and "Adc Set Clock Mode API" are enabled. Hardware averaging functionality is also available for configuration if supported.

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

## 4.66 Parameter AdcHardwareAverageEnableAlternate

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

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

## 4.67 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

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	SAMPLES_4
literals	['SAMPLES_4', 'SAMPLES_8', 'SAMPLES_16', 'SAMPLES_32']

## 4.68 Parameter AdcSamplingDurationAlt0

Alternate configuration: Select the Normal Sampling Duration for channels 0-31 when calling `Adc_SetClockMode(ADC_NORMAL)` for CTR0 register.

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	22
max	255
min	8

## 4.69 Parameter AdcSamplingDurationAlt1

Alternate configuration: Select the Normal Sampling Duration for channels 32-63 when calling `Adc_SetClockMode(ADC_NORMAL)` for CTR1 register.

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	22
max	255
min	8

## 4.70 Parameter AdcSamplingDurationAlt2

Alternate configuration: Select the Normal Sampling Duration for channels 64-95 when calling `Adc_SetClockMode(ADC_NORMAL)` for CTR2 register.

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	22
max	255
min	8

## 4.71 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

## 4.72 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

## 4.73 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

Property	Value
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	P0_ChanNum0
literals	['P0_ChanNum0', 'P1_ChanNum1', 'P2_ChanNum2', 'P3_ChanNum3', 'P4_↵_ChanNum4', 'P5_ChanNum5', 'P6_ChanNum6', 'P7_ChanNum7', 'S8_↵ChanNum32', 'S9_ChanNum33', 'S10_ChanNum34', 'S11_ChanNum35', 'S12_↵_ChanNum36', 'S13_ChanNum37', 'S14_ChanNum38', 'S15_ChanNum39', 'S16_ChanNum40', 'S17_ChanNum41', 'S18_ChanNum42', 'S19_ChanNum43', 'S20_ChanNum44', 'S21_ChanNum45', 'S22_ChanNum46', 'S23_ChanNum47', 'BANDGAP_ChanNum48', 'TEMPSENSOR_OUTPUT_ChanNum49', 'VR_↵EFL_ChanNum50', 'VREFH_ChanNum51', 'ANAMUX_OUTPUT_Chan_↵Num54', 'X0_ChanNum64', 'X0_ChanNum65', 'X0_ChanNum66', 'X0_Chan_↵Num67', 'X0_ChanNum68', 'X0_ChanNum69', 'X0_ChanNum70', 'X0_Chan_↵Num71', 'X1_ChanNum72', 'X1_ChanNum73', 'X1_ChanNum74', 'X1_Chan_↵Num75', 'X1_ChanNum76', 'X1_ChanNum77', 'X1_ChanNum78', 'X1_Chan_↵Num79', 'X2_ChanNum80', 'X2_ChanNum81', 'X2_ChanNum82', 'X2_Chan_↵Num83', 'X2_ChanNum84', 'X2_ChanNum85', 'X2_ChanNum86', 'X2_Chan_↵Num87', 'X3_ChanNum88', 'X3_ChanNum89', 'X3_ChanNum90', 'X3_Chan_↵Num91', 'X3_ChanNum92', 'X3_ChanNum93', 'X3_ChanNum94', 'X3_Chan_↵Num95', 'AN0_AN1_ChanNum0', 'AN2_AN3_ChanNum1', 'VCOM1_VCO_↵M1_ChanNum4', 'VCOM2_VCOM2_ChanNum5', 'VREFP_VREFN_Chan_↵Num6', 'VREFN_VREFP_ChanNum7', 'AN0_VCOM0_ChanNum8', 'AN1_↵_VCOM0_ChanNum9', 'AN2_VCOM0_ChanNum10', 'AN3_VCOM0_Chan_↵Num11', 'AN0_VCOM1_ChanNum16', 'AN1_VCOM1_ChanNum17', 'AN2_↵VCOM1_ChanNum18', 'AN3_VCOM1_ChanNum19', 'AN0_VCOM2_Chan_↵Num24', 'AN1_VCOM2_ChanNum25', 'AN2_VCOM2_ChanNum26', 'AN3_↵VCOM2_ChanNum27']

## 4.74 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 'ChanNum' in the selected 'Adc Physical Channel Name'. E.g. ADC\_CH\_0\_ChanNum38 => Channel ID must be 38. 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



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	0
max	1024
min	0

## 4.75 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-INTEGGER-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

## 4.76 Parameter AdcChannelLimitCheck

Enables or disables limit checking for an ADC channel.

This node is EDITABLE only if "Adc Enable Limit Check" is enabled

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

## 4.77 Parameter AdcChannelHighLimit

High limit - used for limit checking. This value depends on the Adc Hw Unit resolution. If AdcHwUnitBypassResolution is enabled, then this value will always be on 15 bits.

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
defaultValue	16383
max	9223372036854775807
min	0

## 4.78 Parameter AdcChannelLowLimit

Low limit - used for limit checking.

Property	Value
type	ECUC-INTEGER-PARAM-DEF

Property	Value
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	0
max	9223372036854775807
min	0

## 4.79 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']

## 4.80 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
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']

## 4.81 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']

## 4.82 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
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	14
max	63
min	1

### 4.83 Parameter AdcChannelSampTime

Sampling time, i.e. the time during which the value is sampled, (in clock cycles) for each channel. 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	22
max	9223372036854775807
min	0

### 4.84 Parameter AdcEnablePresampling

When true, this parameter enables the presampling phase for the selected channel.

The normal operation sequence on the channel: Presampling -> Sampling -> Conversion.

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

## 4.85 Parameter AdcEnableThresholds

When true, this parameter enables the threshold detection feature for the selected channel.

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

## 4.86 Parameter AdcWdogNotification

This function pointer is called whenever the conversion of the channel issued a watchdog interrupt.

The notification takes one uint8 parameter which represents the flags(low/high threshold) that triggered the interrupt.

In order to interpret them please use ADC\_WDG\_... defines.

Property	Value
type	ECUC-FUNCTION-NAME-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	NULL_PTR

## 4.87 Reference AdcThresholdRegister

Select the threshold register which provides the values to be used for upper and lower thresholds.

ADCHwUnits support threshold registers from ADC\_THRESHOLD\_REG\_0 to ADC\_THRESHOLD\_REG\_3.

Note: This is an Implementation Specific Parameter.

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	/TS_T40D34M30I0R0/Adc/AdcConfigSet/AdcHwUnit/AdcThresholdControl

## 4.88 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

## 4.89 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']

## 4.90 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']

## 4.91 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', 'ADC_CONV_TYPE_INJECTED']

## 4.92 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

### 4.93 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

### 4.94 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']

## 4.95 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 CTU or External hardware pins of the controller.

In this controller only Bctu trigger source is supported which is selected by the "Adc Group Hardware Trigger Source" parameter.

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']

## 4.96 Parameter AdcHwTrigSignal

Configures the edge of the hardware trigger signal, i.e. to start the conversion.

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	ADC_HW_TRIG_RISING_EDGE
literals	['ADC_HW_TRIG_BOTH_EDGES', 'ADC_HW_TRIG_RISING_EDGE', 'ADC_HW_TRIG_FALLING_EDGE']

## 4.97 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

## 4.98 Parameter AdcNotification

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

This node is EDITABLE only if "Adc Notification Capability" is enabled

Property	Value
type	ECUC-FUNCTION-NAME-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	NULL_PTR

## 4.99 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.

This feature is enabled if "Adc Notification Capability" from General/Adc General Configuration is enabled.

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

## 4.100 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).

This feature is enabled only if "Adc Group Access Mode" is set to ADC\_ACCESS\_MODE\_STREAMING.

Property	Value
type	ECUC-ENUMERATION-PARAM-DEF

Property	Value
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_STREAM_BUFFER_LINEAR
literals	['ADC_STREAM_BUFFER_LINEAR', 'ADC_STREAM_BUFFER_CIRCULAR']

#### 4.101 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

#### 4.102 Parameter AdcEnableHalfInterrupt

Enable/ Disable the interrupt when half sample complete for optimize DMA streaming groups feature.

The "Adc Optimize DMA Streaming Groups" must be enabled for configuring this feature.

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

### 4.103 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
default Value	1
max	255
min	1

### 4.104 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
defaultValue	false

### 4.105 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
defaultValue	false



## 4.106 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

## 4.107 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

## 4.108 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

## 4.109 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

Property	Value
	VARIANT-PRE-COMPILE: PRE-COMPILE
requiresSymbolicNameValue	False
destination	/TS_T40D34M30I0R0/Adc/AdcConfigSet/AdcHwTrigger

#### 4.110 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

#### 4.111 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/EcucDefs/EcuC/EcucPartitionCollection/EcucPartition

## 4.112 Container AdcGroupConversionConfiguration

Configure the Sampling and Conversion TimeGroup.

Hardware averaging functionality is also available for configuration.

Note: This is an Implementation Specific Parameter.

Included subcontainers:

- None

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

## 4.113 Parameter AdcGroupHardwareAverageEnable

Enables the hardware average function of the ADC.

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

## 4.114 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']

#### 4.115 Parameter AdcSamplingDuration0

Select the Sampling Duration for channels 0-31 from CTR0 register

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	22
max	255
min	8

## 4.116 Parameter AdcSamplingDuration1

Select the Sampling Duration for channels 32-63 from CTR1 register

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	22
max	255
min	8

## 4.117 Parameter AdcSamplingDuration2

Select the Sampling Duration for channels 64-95 from CTR2 register

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	22
max	255
min	8

## 4.118 Container AdcAlternateGroupConvTimings

Selects Alternate values used in Adc\_SetClockMode API for programming CTR Conversion Timing Registers.

This container is EDITABLE only if "Adc Conversion Time Once" is disabled and "Adc Set Clock Mode API" is enabled. Hardware averaging functionality is also available for configuration if supported.

Included subcontainers:

- None

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

#### 4.119 Parameter AdcGroupAltHardwareAverageEnable

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

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

#### 4.120 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']

#### 4.121 Parameter AdcAltGroupSamplingDuration0

Select the Alternate Sampling Duration for channels 0-31 from CTR0 register

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	22
max	255
min	8

#### 4.122 Parameter AdcAltGroupSamplingDuration1

Select the Alternate Sampling Duration for channels 32-63 from CTR1 register

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	22
max	255
min	8

### 4.123 Parameter AdcAltGroupSamplingDuration2

Select the Alternate Sampling Duration for channels 64-95 from CTR2 register

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	22
max	255
min	8

### 4.124 Container AdcThresholdControl

Configure threshold detection feature for the selected channel.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF
lowerMultiplicity	0
upperMultiplicity	4
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE

## 4.125 Parameter AdcThresholdControlRegister

Select the threshold register which provides the values to be used for upper and lower thresholds.

ADCHwUnits support threshold registers from ADC\_THRESHOLD\_REG\_0 to ADC\_THRESHOLD\_REG\_3.

Note: This is an 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	true
valueConfigClasses	VARIANT-POST-BUILD: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	ADC_THRESHOLD_REG_0
literals	['ADC_THRESHOLD_REG_0', 'ADC_THRESHOLD_REG_1', 'ADC_THRESHOLD_REG_2', 'ADC_THRESHOLD_REG_3']

## 4.126 Parameter AdcHighThreshold

Set the value for High Threshold. This value depends on the Adc Hw Unit resolution. If AdcHwUnitBypassResolution is enabled, then this value will always be on 15 bits.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0

Property	Value
upperMultiplicity	1
postBuildVariantMultiplicity	false
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	16383
max	65535
min	0

## 4.127 Parameter AdcLowThreshold

Set the value for Low Threshold.

Property	Value
type	ECUC-INTEGER-PARAM-DEF
origin	NXP
symbolicNameValue	false
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
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	65535
min	0

## 4.128 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

## 4.129 Parameter AdcHwTrigSrc

On this implementation the HW triggers available are from Bctu.

Note: This is an 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: POST-BUILD
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	EXT_TRIG
literals	['BCTU_EMIOS_0_0', 'BCTU_EMIOS_0_1', 'BCTU_EMIOS_0_2', 'BCTU_EMIOS_0_3', 'BCTU_EMIOS_0_4', 'BCTU_EMIOS_0_5', 'BCTU_EMIOS_0_6', 'BCTU_EMIOS_0_7', 'BCTU_EMIOS_0_8', 'BCTU_EMIOS_0_9', 'BCTU_EMIOS_0_10', 'BCTU_EMIOS_0_11', 'BCTU_EMIOS_0_12', 'BCTU_EMIOS_0_13', 'BCTU_EMIOS_0_14', 'BCTU_EMIOS_0_15', 'BCTU_EMIOS_0_16', 'BCTU_EMIOS_0_17', 'BCTU_EMIOS_0_18', 'BCTU_EMIOS_0_19', 'BCTU_EMIOS_0_20', 'BCTU_EMIOS_0_21', 'BCTU_EMIOS_0_22', 'BCTU_eTPU_A_0', 'BCTU_eTPU_A_1', 'BCTU_eTPU_A_2', 'BCTU_eTPU_A_3', 'BCTU_eTPU_A_4', 'BCTU_eTPU_A_5', 'BCTU_eTPU_A_6', 'BCTU_eTPU_A_7', 'BCTU_eTPU_B_0', 'BCTU_eTPU_B_1', 'BCTU_eTPU_B_2', 'BCTU_eTPU_B_3', 'BCTU_eTPU_B_4', 'BCTU_eTPU_B_5', 'BCTU_eTPU_B_6', 'BCTU_eTPU_B_7', 'BCTU_TRGMUX_52', 'BCTU_TRGMUX_56', 'BCTU_TRGMUX_53', 'BCTU_TRGMUX_57', 'BCTU_TRGMUX_54', 'BCTU_TRGMUX_58', 'EXT_TRIG', 'AUX_EXT_TRIG', 'EXT_AND_AUX_EXT_TRIG', 'SDADC_SW_0_TrigNum0', 'SDADC_SW_1_TrigNum1', 'SDADC_SW_2_TrigNum2', 'SDADC_SW_3_TrigNum3', 'SDADC_TRGMUX_28_32_36_40_TrigNum4', 'SDADC_ETPU_1_20_22_24_26_TrigNum5', 'SDADC_ETPU_1_21_23_25_27_TrigNum6', 'SDADC_ETPU_2_20_22_24_26_TrigNum7', 'SDADC_ETPU_2_21_23_25_27_TrigNum8']

## 4.130 Container BctuHwUnit

This container contains configuration of the Bctu unit. This node is EDITABLE only if "Adc CTU Control Mode API" is enabled.

Included subcontainers:

- [BctuInternalTrigger](#)
- [BctuAdcNotifications](#)
- [BctuListItems](#)
- [BctuResultFifos](#)

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

## 4.131 Parameter BctuHwUnitId

Specifies the used BCTU Hardware Unit.

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	1
min	0

### 4.132 Parameter BctuLogicalUnitId

Specifies the Logical id of the Bctu 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
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	0
max	1
min	0

### 4.133 Parameter BctuLowPowerMode

Enable Bctu low power mode. If enabled, during runtime it must be disabled before allowing triggers to start ADC conversions.

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

### 4.134 Parameter BctuGlobalHwTriggers

Enable hardware triggers to start ADC conversions. If disabled only BCTU software triggers can start conversions.

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	true

#### 4.135 Parameter BctuNewDataDMAEnableMask

Bitmask for DMA transfer enable per ADC instance: (1 shifted left n) corresponds to n-th ADC instance connected to the BCTU. The user is responsible for setting up DMA transfer buffer and configuration.

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
default Value	0
max	2047
min	0

#### 4.136 Parameter BctuFifoDmaRawData

Enable Dma Transfer Raw Data of Fifo Result Register (includes Trigger Source, ADC Channel, ADC Number and ADC Data Conversion).

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: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	false

### 4.137 Parameter BctuTriggerNotification

This function pointer is called whenever a BCTU trigger is asserted. NULL value disables the notification.

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

### 4.138 Container BctuInternalTrigger

This container contains the BCTU input trigger configuration parameters

Included subcontainers:

- None

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



### 4.139 Parameter BctuTriggerLoop

Enable/Disable loop trigger conversion. This functionality is disabled by default.

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

### 4.140 Parameter BctuDataDestination

Select the destination of the conversion data.

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: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	BCTU_ADC_DATA_REG
literals	['BCTU_ADC_DATA_REG', 'BCTU_FIFO1', 'BCTU_FIFO2', 'BCTU_FIFO3']

### 4.141 Parameter BctuHwTriggerEnable

Enable/disable initiation of ADC conversions by hw input trigger: true - enabled; false - disabled.

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	true

#### 4.142 Parameter BctuTriggerConversionMode

Select list mode or single mode for conversion. The multiple ADC selection is only valid for multiple parallel conversions LIST functionality.

If more than one ADC is selected for a single conversion, instead of for a LIST, no conversion will be triggered by this register.

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	SINGLE
literals	['SINGLE', 'LIST']

#### 4.143 Parameter BctuAdcTargetMask

Bitmask to select target ADC instance(s): (1 shifted left n) corresponds to n-th ADC instance.

Property	Value
type	ECUC-INTEGER-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	1
max	255
min	1

#### 4.144 Parameter BctuConversionListStartIndex

Start index in the BCTU list, to be enabled when the trigger is asserted. Only enabled if Trigger mode is LIST.

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	31
min	0

#### 4.145 Reference BctuTriggerSource

On this implementation the HW triggers available are from BCTU.

(Note: This is an Implementation Specific Parameter.)

Property	Value
type	ECUC-REFERENCE-DEF

Property	Value
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_T40D34M30I0R0/Adc/AdcConfigSet/AdcHwTrigger

#### 4.146 Reference BctuAdcChannelSingle

ADC channel to be converted when the trigger is asserted in SINGLE Trigger Conversion Mode

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	/AUTOSAR/EcucDefs/Adc/AdcConfigSet/AdcHwUnit/AdcChannel

#### 4.147 Container BctuAdcNotifications

This container allows configuration of the BCTU notifications corresponding to each ADC unit.

Included subcontainers:

- None

Property	Value
type	ECUC-PARAM-CONF-CONTAINER-DEF

Property	Value
lowerMultiplicity	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	true
multiplicityConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE

#### 4.148 Parameter BctuAdcNewDataNotification

This function is called whenever a BCTU new data event is asserted, for the selected ADC.

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

#### 4.149 Parameter BctuDataOverrunNotification

This function is called whenever a BCTU data overrun event is asserted, for the selected ADC.

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

## 4.150 Parameter BctuListLastConversionNotification

This function pointer is called when the last conversion in the list is started, for the selected ADC.

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

## 4.151 Reference BctuAdcNotificationsAdcIndex

ADC instance for which the BCTU notifications are configured.

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	/AUTOSAR/EcucDefs/Adc/AdcConfigSet/AdcHwUnit

## 4.152 Container BctuListItems

This container allows configuration of BCTU conversion list elements.

IMPORTANT NOTE: Command List is generated in the order given by this array list.

Included subcontainers:

- None

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

### 4.153 Parameter BctuAdcChannelList

Selects the physical Hardware Adc Channel for the Conversion List. 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	P0_ChanNum0



Property	Value
literals	['P0_ChanNum0', 'P1_ChanNum1', 'P2_ChanNum2', 'P3_ChanNum3', 'P4_↵ _ChanNum4', 'P5_ChanNum5', 'P6_ChanNum6', 'P7_ChanNum7', 'S8_↵ ChanNum32', 'S9_ChanNum33', 'S10_ChanNum34', 'S11_ChanNum35', 'S12_↵ _ChanNum36', 'S13_ChanNum37', 'S14_ChanNum38', 'S15_ChanNum39', 'S16_ChanNum40', 'S17_ChanNum41', 'S18_ChanNum42', 'S19_ChanNum43', 'S20_ChanNum44', 'S21_ChanNum45', 'S22_ChanNum46', 'S23_ChanNum47', 'BANDGAP_ChanNum48', 'TEMPSENSOR_OUTPUT_ChanNum49', 'VR_↵ EFL_ChanNum50', 'VREFH_ChanNum51', 'ANAMUX_OUTPUT_Chan_↵ Num54', 'X0_ChanNum64', 'X0_ChanNum65', 'X0_ChanNum66', 'X0_Chan_↵ Num67', 'X0_ChanNum68', 'X0_ChanNum69', 'X0_ChanNum70', 'X0_Chan_↵ Num71', 'X1_ChanNum72', 'X1_ChanNum73', 'X1_ChanNum74', 'X1_Chan_↵ Num75', 'X1_ChanNum76', 'X1_ChanNum77', 'X1_ChanNum78', 'X1_Chan_↵ Num79', 'X2_ChanNum80', 'X2_ChanNum81', 'X2_ChanNum82', 'X2_Chan_↵ Num83', 'X2_ChanNum84', 'X2_ChanNum85', 'X2_ChanNum86', 'X2_Chan_↵ Num87', 'X3_ChanNum88', 'X3_ChanNum89', 'X3_ChanNum90', 'X3_Chan_↵ Num91', 'X3_ChanNum92', 'X3_ChanNum93', 'X3_ChanNum94', 'X3_Chan_↵ Num95', 'AN0_AN1_ChanNum0', 'AN2_AN3_ChanNum1', 'VCOM1_VCO_↵ M1_ChanNum4', 'VCOM2_VCOM2_ChanNum5', 'VREFP_VREFN_Chan_↵ Num6', 'VREFN_VREFP_ChanNum7', 'AN0_VCOM0_ChanNum8', 'AN1_↵ _VCOM0_ChanNum9', 'AN2_VCOM0_ChanNum10', 'AN3_VCOM0_Chan_↵ Num11', 'AN0_VCOM1_ChanNum16', 'AN1_VCOM1_ChanNum17', 'AN2_↵ VCOM1_ChanNum18', 'AN3_VCOM1_ChanNum19', 'AN0_VCOM2_Chan_↵ Num24', 'AN1_VCOM2_ChanNum25', 'AN2_VCOM2_ChanNum26', 'AN3_↵ VCOM2_ChanNum27']

#### 4.154 Parameter BctuNextChannelWaitOnTrig

Disable or enable, if the list execution is continuous (for FALSE value) or if the list execution is halted after executing this channel, until the same trigger responsible for start of list is re-applied/re-asserted.

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

## 4.155 Parameter BctuLastChannel

If enabled, this is the last channel converted the current list. Another list can be started from the next position in the BCTU list.

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

## 4.156 Container BctuResultFifos

This container allows configuration of BCTU result FIFOs.

Included subcontainers:

- None

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

## 4.157 Parameter BctuResultFifoIndex

Bctu result FIFO index.

Property	Value
type	ECUC-ENUMERATION-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	FIFO_1
literals	['FIFO_1', 'FIFO_2', 'FIFO_3']

#### 4.158 Parameter BctuWatermarkValue

If the number of active FIFO entries exceeds the watermark level, a DMA or interrupt request is raised.

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	16
min	0

#### 4.159 Parameter BctuFifoNotificationsEnable

Enable FIFO interrupt notifications.

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

#### 4.160 Parameter BctuWatermarkNotification

This function is called if the number of valid FIFO entries is greater than the watermark level. Used for both cases Interrupt and Dma.

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

#### 4.161 Parameter BctuUnderrunNotification

This function is called for FIFO underrun event (FIFO is empty and read attempt is made).

Property	Value
type	ECUC-FUNCTION-NAME-DEF
origin	NXP
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	NULL_PTR

## 4.162 Parameter BctuOverrunNotification

This function is called for FIFO overrun event (FIFO is full and write attempt is made).

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

## 4.163 Parameter BctuFifoDmaEnable

Controls the DMA request that occurs if the number of valid FIFO entries is greater than the watermark level for that FIFO. The Fifo Interrupt should be disabled when actives Dma to prevent conflict Data Transfer.

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

## 4.164 Parameter BctuFifoDmaBuffer

Pointer to the Fifo Data Buffer (destination for conversion results) when BctuFifoDmaEnable is enabled.

Property	Value
type	ECUC-LINKER-SYMBOL-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	BctuDmaFifo

## 4.165 Reference BctuFifoDmaChannelId

Select Logical Dma Channel that will be used to transfer the Fifo Buffer from FIFO Result Data Register to the user fifo buffer.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
lowerMultiplicity	0
upperMultiplicity	1
postBuildVariantMultiplicity	false
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/Mcl/MclConfig/dmaLogicChannel_Type

## 4.166 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

## 4.167 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
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

## 4.168 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



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

## 4.169 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
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PRE-COMPILE
	VARIANT-PRE-COMPILE: PRE-COMPILE
defaultValue	true

## 4.170 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

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

#### 4.171 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
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

#### 4.172 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

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

### 4.173 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
defaultValue	true

### 4.174 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

Property	Value
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	false

### 4.175 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.

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']

## 4.176 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

## 4.177 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']

## 4.178 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

## 4.179 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.

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

## 4.180 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

## 4.181 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

## 4.182 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/EcuDefs/EcuC/EcuPartitionCollection/EcuPartition

### 4.183 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

### 4.184 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

#### 4.185 Parameter AdcPowerStateReadyCbkJef

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

#### 4.186 Container AdcInterrupt

Selects whether the interrupt for each ADC Unit will be enabled. For each Adc HW unit, there are 2 interrupts that can be enabled: the End of Conversion and the Watchdog interrupts. 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	0
upperMultiplicity	Infinite
postBuildVariantMultiplicity	false
multiplicityConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE

#### 4.187 Parameter AdcInterruptSource

The name of the interrupt. 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-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	ADC0_EOC
literals	['ADC0_EOC', 'ADC0_WD', 'ADC1_EOC', 'ADC1_WD', 'ADC2_EOC', 'ADC2_WD', 'ADC3_EOC', 'ADC3_WD', 'ADC4_EOC', 'ADC4_WD', 'ADC5_EOC', 'ADC5_WD', 'ADC6_EOC', 'ADC6_WD', 'ADC7_SD0_FIFOFULL', 'ADC7_SD0_WD', 'ADC8_SD1_FIFOFULL', 'ADC8_SD1_WD', 'ADC9_SD2_FIFOFULL', 'ADC9_SD2_WD', 'ADC10_SD3_FIFOFULL', 'ADC10_SD3_WD']

#### 4.188 Parameter AdcInterruptEnable

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

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

## 4.189 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

## 4.190 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

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

#### 4.191 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

#### 4.192 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	14
max	63
min	1

### 4.193 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

### 4.194 Parameter ArReleaseMajorVersion

Major 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	4
max	4
min	4

### 4.195 Parameter ArReleaseMinorVersion

Minor 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	7
max	7
min	7

#### 4.196 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
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

#### 4.197 Parameter ModuleId

Module ID of this module from Module List.

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	123
max	123
min	123

#### 4.198 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
postBuildVariantMultiplicity	N/A
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	3
max	3
min	3

#### 4.199 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

Property	Value
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

## 4.200 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
multiplicityConfigClasses	N/A
postBuildVariantValue	false
valueConfigClasses	VARIANT-POST-BUILD: PUBLISHED-INFORMATION
	VARIANT-PRE-COMPILE: PUBLISHED-INFORMATION
defaultValue	0
max	0
min	0

## 4.201 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	

## 4.202 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

## 4.203 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

## 4.204 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']

## 4.205 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
default Value	100000
max	4294967295
min	0

## 4.206 Parameter AdcSarIpDevErrorDetect

This parameter Enables / Disables development error detection for Adc Sar.

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

## 4.207 Parameter SdadcIpDevErrorDetect

This parameter Enables / Disables development error detection for Sd 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	false
valueConfigClasses	VARIANT-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
defaultValue	false

## 4.208 Parameter BctuIpDevErrorDetect

This parameter Enables / Disables development error detection for Bctu.

This feature is enabled if Adc CTU Control Mode API from General/AutosarExt is enabled or if Adc Hw Trigger API from General is enabled.

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

## 4.209 Parameter SdadcDspssEnable

This parameter Enables / Disables the Dspss supporting for Sd Adc.

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

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

## 4.210 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 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

## 4.211 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

Property	Value
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

#### 4.212 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
defaultValue	false

#### 4.213 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-PRE-COMPILE: PRE-COMPILE
	VARIANT-POST-BUILD: PRE-COMPILE
default Value	false

## 4.214 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 at most 8 channels configured.

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

## 4.215 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.

This node is EDITABLE only if "Adc Enable DMA support" is enabled

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

### 4.216 Parameter AdcPreSamplingOnce

Implementation Specific Parameter.

Enable/Disable one time setting of the registers.

If Enabled, the setting of the presampling 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



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

## 4.217 Parameter AdcEnableInitialNotification

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

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.

This node is EDITABLE only if "Adc Set Channel API" is enabled.

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

## 4.218 Parameter AdcEnableDmaTransferMode

This parameter 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

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
default Value	false

#### 4.219 Parameter AdcUseSoftwareInjectedGroups

This parameter defines if Software Injected Groups are used in any Hardware Unit, any variant.

It needs to be enabled if Software Injected Groups are needed. If Software Injected Groups are not needed, this parameter should be disabled for code optimizations.

NoteThis 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

#### 4.220 Parameter AdcUseHardwareNormalGroups

This parameter defines if Hardware Normal Groups are used in any Hardware Unit, any variant. It needs to be enabled if Hardware Normal Groups are needed. If Hardware Normal Groups are not needed, this parameter should be disabled for code optimizations. Normal Hardware conversions are not supported on this platform.

NoteThis 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

#### 4.221 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
default Value	false

#### 4.222 Parameter AdcSetHwUnitPowerModeApi

Adds/removes the Autosar Extension implementation API Adc\_SetHwUnitPowerMode() and Adc\_CtuSetPowerMode() from the code.

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

Property	Value
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

### 4.223 Parameter AdcEnableChDisableChApi

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

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

### 4.224 Parameter AdcGetInjectedConvStatusApi

Enable/disable the Autosar Extension API `Adc_GetInjectedConversionStatus()` in ADC driver.

Note: This is an Implementation Specific Parameter.

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

## 4.225 Parameter `AdcEnableThresholdConfigurationApi`

Enable/disable the Autosar Extension APIs `Adc_ConfigureThreshold()` in ADC driver.

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

## 4.226 Parameter `AdcEnableCtuTrigAutosarExtApi`

This is used to enable the Autosar Extension API for the hardware triggered group.

If this parameter is enabled then `Adc_EnableCTUTrigger()`, `Adc_DisableCTUTrigger()` and `Adc_HwResultReadGroup()` will be available in the driver code.

This is an Implementation Specific Parameter.

When this parameter is enabled, the result buffer is no longer to be used to read the results as the result will be directly read from HW registers.

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

This node is EDITABLE only if "Adc Hw Trigger API" is enabled

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

## 4.227 Parameter AdcCtuHardwareTriggerOptimization

When this parameter is enabled, the Bctu channel lists and triggers are configured only once at initialization in order to reduce cpu load when calling the Adc\_EnableHardwareTrigger and Adc\_EnableCtuTrigger APIs.

The Adc\_SetChannel service cannot be used, and the maximum size of groups cannot exceed the size of the entire command list divided by the number of adc hardware units triggered by the CTU/BCTU.

This node is EDITABLE only if "Adc Hw Trigger API" is enabled

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

## 4.228 Parameter AdcEnableCtuControlModeApi

This is used to enable the Autosar Extension API for the enabling and disabling CTU control mode for an ADC unit.

If this parameter is enabled than Adc\_EnableCtuControlMode(), Adc\_DisableCtuControlMode() will be available in the driver code.

When a unit works in CTU control mode, no other conversions shall run in parallel(Adc). The only conversions occurring shall be the ones defined in the CTU configuration.

If AdcEnableCtuControlModeApi is enabled, BCTU must be configured.

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

## 4.229 Parameter CtuEnableDmaTransferMode

This parameter enables the possibility to configure DMA transfer for BCTU control mode.

If this parameter is disabled then DMA handling code will be removed at pre-compile time and DMA transfer cannot be configured for any BCTU 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.

This node is EDITABLE only if "Adc CTU Control Mode API" is enabled

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

## 4.230 Parameter AdcEnableWatchdogApi

This parameter globally enables the possibility to use the Adc Watchdog feature. If this parameter is disabled, the Watchdog handling code will be removed at pre-compile time and nothing related to this functionality can be configured in any unit, for any variant. If this parameter is enabled, Analog Watchdog functionality can be configured.

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

## 4.231 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
defaultValue	false



### 4.232 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
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

### 4.233 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

### 4.234 Parameter AdcEnableApplyCalibration

If this parameter has been configured to "TRUE", the Autosar Extension function "`Adc_ApplyCalibration()`" shall

be accessible, otherwise this function shall be removed from the code.

Note

This is an Implementation Specific Parameter.

This feature is only used for SDADC hardware units.

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

### 4.235 Parameter AdcEnableSelfTest

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

NoteThis 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

## 4.236 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
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

## 4.237 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

## 4.238 Parameter AdcSarEnableTempsenseApi

When this parameter is enabled, the Apis for the tempsense are available to be used.

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

## 4.239 Parameter AdcSarPowerOnTempSense

When this parameter is enabled, the TempSense module is powered on.

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

## 4.240 Parameter AdcSarTempSenseVsupply

Voltage Supply of TempSense module, expressed in fixed point format with 1 bit for the sign, 11 bits for the integer part and 4 bits for the decimal part. The value must be filled in by the user according to actual board setup. It and will be used for calculating the value measured by TempSense.

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
default Value	53
max	88
min	0

#### 4.241 Reference SdadcDspssInitDmaChannelId

Logical Id of the DMA channel used to transfer the contents of the XMEM and PMEM when initializing the DSPSS.

Property	Value
type	ECUC-REFERENCE-DEF
origin	NXP
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
requiresSymbolicNameValue	False
destination	/AUTOSAR/EcucDefs/Mcl/MclConfig/dmaLogicChannel_Type



# Chapter 5

## Module Index

### 5.1 Software Specification

Here is a list of all modules:

Adc driver . . . . .	167
Adc Sar IPL . . . . .	223
Bctu IPL . . . . .	262
IP_SDADC . . . . .	282

## 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_NOT_DISENGAGED`  
*One or more ADC group/channel not in IDLE state.*
- `#define ADC_E_POWER_STATE_NOT_SUPPORTED`  
*Unsupported power state request.*
- `#define ADC_E_TRANSITION_NOT_POSSIBLE`  
*Requested power state can not be reached directly.*
- `#define ADC_E_PERIPHERAL_NOT_PREPARED`  
*ADC not prepared for target power state.*
- `#define ADC_E_QUEUE_FULL`  
*The `Adc_StartGroupConversion` and `Adc_EnableHardwareTrigger` services can not queue another conversion (queue is full)*
- `#define ADC_E_SET_HW_UNIT_POWER_MODE`  
*An error occurred when the `Adc_SetHwUnitPowerMode` services is used.*
- `#define ADC_E_PARAM_TRIGGER`  
*Wrong trigger source to be used for the group.*
- `#define ADC_E_WRONG_ENABLE_CH_DISABLE_CH_GROUP`  
*`Adc_EnableChannel` or `Adc_DisableChannel` services called with a wrong channel.*
- `#define ADC_E_WRONG_ENABLE_CH_DISABLE_CH_ID`  
*`Adc_EnableChannel` or `Adc_DisableChannel` services called with a wrong channel identifier (ID).*
- `#define ADC_E_WRONG_CONF_THRHLN_VALUE`  
*`Adc_ConfigureThreshold` service is called using wrong values.*
- `#define ADC_E_PARAM_UNIT`  
*API service called using a wrong ADC unit.*
- `#define ADC_E_WRONG_CTM_TRIGGER`  
*API service called using a wrong CTU trigger.*
- `#define ADC_E_WRONG_CTM_CLCR_TRIGGER`  
*API service called using a wrong CTU CLCR trigger.*
- `#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_E_CONTROL_MODE_DISABLED`  
*Error when a function which works only in CTU Control Mode is called when CTU control mode is disabled.*
- `#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_SETPOWERSTATE_ID`  
API service ID for `Adc_SetPowerState` function.
- `#define ADC_GETCURRENTPOWERSTATE_ID`  
API service ID for `Adc_GetCurrentPowerState` function.
- `#define ADC_GETTARGETPOWERSTATE_ID`  
API service ID for `Adc_GetTargetPowerState` function.
- `#define ADC_PREPAREPOWERSTATE_ID`  
API service ID for `Adc_PrepPowerState` function.
- `#define ADC_HWRESULTREADGROUP_ID`  
API service ID for `Adc_HwResultReadGroup` function.
- `#define ADC_ENABLECTUTRIGGER_ID`  
API service ID for `Adc_EnableCTUTrigge` function.
- `#define ADC_DISABLECTUTRIGGER_ID`  
API service ID for `Adc_DisableCTUTrigger` function.
- `#define ADC_SET_HW_UNIT_POWER_MODE_ID`  
API service ID for `Adc_SetHwUnitPowerMode` function.
- `#define ADC_CTU_SET_POWER_MODE_ID`  
API service ID for `Adc_CtuSetPowerMode` function.
- `#define ADC_SETCLOCKMODE_ID`  
API service ID for `Adc_SetClockMode` function.
- `#define ADC_ENABLE_CHANNEL_ID`  
API service ID for `Adc_EnableChannel` function.
- `#define ADC_DISABLE_CHANNEL_ID`

- API service ID for `Adc_DisableChannel` function.*

  - #define [ADC\\_GETINJECTEDCONVERSIONSTATUS\\_ID](#)

*API service ID for `Adc_GetInjectedConversionStatus` function.*
  - #define [ADC\\_CALIBRATE\\_ID](#)

*API service ID for `Adc_Calibrate` function.*
  - #define [ADC\\_CONFIGURE\\_THRESHOLD\\_ID](#)

*API service ID for `Adc_ConfigureThreshold` function.*
  - #define [ADC\\_CTU\\_SET\\_WRITE\\_TRIG\\_EN\\_MASK\\_ID](#)

*API service ID for `Adc_CtuWriteTriggerEnableMask` function.*
  - #define [ADC\\_CTU\\_SET\\_TRIGGER\\_ENABLE\\_ID](#)

*API service ID for `Adc_CtuSetTriggerEnable` function.*
  - #define [ADC\\_CTU\\_SET\\_TRIGGER\\_ADC\\_CMD\\_ADDRESS\\_ID](#)

*API service ID for `Adc_CtuSetTriggerAdcCmdAddress` function.*
  - #define [ADC\\_CTU\\_SET\\_TRIGGER\\_COMPARE\\_ID](#)

*API service ID for `Adc_CtuSetTriggerCompare` function.*
  - #define [ADC\\_SETCANNEL\\_ID](#)

*API service ID for `Adc_SetChannel` function.*
  - #define [ADC\\_ENABLE\\_CTU\\_CONTROL\\_MODE\\_ID](#)

*API service ID for `Adc_EnableCtuControlMode` function.*
  - #define [ADC\\_DISABLE\\_CTU\\_CONTROL\\_MODE\\_ID](#)

*API service ID for `Adc_DisableCtuControlMode` function.*
  - #define [ADC\\_APPLY\\_CALIBRATION\\_ID](#)

*API service ID for `Adc_ApplyCalibration` 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.*
- Std\_ReturnType [Adc\\_SetHwUnitPowerMode](#) (Adc\_HwUnitType Unit, Adc\_SetPowerModeType SetPowerMode)  
*Set the ADC mode either to powerdown or normal.*
- Std\_ReturnType [Adc\\_CtuSetPowerMode](#) (Adc\_HwUnitType CtuUnit, Adc\_PowerStateType State)  
*Function to set BCTU/CTU power mode.*
- void [Adc\\_EnableCTUTrigger](#) (Adc\_GroupType Group, Adc\_HwTriggerTimerType TriggerSource)  
*Enable the TriggerSource for group selected by Group parameter.*
- void [Adc\\_DisableCTUTrigger](#) (Adc\_GroupType Group, Adc\_HwTriggerTimerType TriggerSource)  
*Disable the TriggerSource for group selected by Group parameter.*
- Std\_ReturnType [Adc\\_HwResultReadGroup](#) (Adc\_GroupType Group, Adc\_ValueGroupType \*DataPtr)  
*Read the result of the hardware triggered groups conversion result.*
- void [Adc\\_EnableChannel](#) (Adc\_GroupType Group, Adc\_ChannelType Channel)  
*Enable an individual channel configured in SW-triggered (non-injected) ADC Group at initialization.*
- void [Adc\\_DisableChannel](#) (Adc\_GroupType Group, Adc\_ChannelType Channel)  
*Disable an individual channel configured in SW-triggered (non-injected) ADC Group at initialization.*
- [Adc\\_StatusType](#) [Adc\\_GetInjectedConversionStatus](#) (Adc\_HwUnitType Unit)  
*Get the injected conversions status.*
- void [Adc\\_Calibrate](#) (Adc\_HwUnitType Unit, Adc\_CalibrationStatusType \*pStatus)  
*Executes high accuracy calibration of a ADC HW unit.*
- Std\_ReturnType [Adc\\_ApplyCalibration](#) (Adc\_HwUnitType Unit, const uint32 BufferLength, const sint16 \*const UncalibratedBuffer, sint16 \*const CalibratedBuffer)  
*Calibrate the converted data of the SDADC.*
- Std\_ReturnType [Adc\\_SelfTest](#) (Adc\_HwUnitType Unit)  
*Executes hardware Self Test of a ADC HW unit.*
- void [Adc\\_ConfigureThreshold](#) (Adc\_ThresholdControlIndexType ThresholdControlIndex, Adc\_WdgThresholdValueType LowValue, Adc\_WdgThresholdValueType HighValue)  
*Function to reconfigure High and Low thresholds for a given threshold control index.*
- void [Adc\\_EnableWdgNotification](#) (Adc\_ChannelType ChannelId)  
*Enable notification of a channel that has watchdog functionality configured at initialization.*
- void [Adc\\_DisableWdgNotification](#) (Adc\_ChannelType ChannelId)  
*Disable notification of a channel that has watchdog functionality configured at initialization.*
- Std\_ReturnType [Adc\\_SetClockMode](#) (Adc\_SelectPrescalerType Prescaler)  
*Set the ADC clock prescaler if available and modify the conversion timings.*
- Std\_ReturnType [Adc\\_CtuWriteTriggerEnableMask](#) (Adc\_CtuTrigSrcType Trigger, uint8 ThcrValue)  
*Function to set the trigger handler control registers of the CTU IPL.*
- Std\_ReturnType [Adc\\_CtuSetTriggerEnable](#) (Adc\_CtuTrigSrcType Trigger, boolean Enable)

*Function to set or clear the bit Tx\_E of the trigger handler control registers of the CTU IPL.*

- Std\_ReturnType [Adc\\_CtuSetTriggerAdcCmdAddress](#) (Adc\_CtuTrigSrcType Trigger, uint8 CldrValue)

*Function to set the cmd list handler control registers of the CTU IPL.*

- Std\_ReturnType [Adc\\_CtuSetTriggerCompare](#) (Adc\_CtuTrigSrcType Trigger, uint16 CompareValue)

*Function to set the trigger compare registers of the CTU IPL.*

- void [Adc\\_SetChannel](#) (Adc\_GroupType Group, const [Adc\\_GroupDefType](#) \*Channel, Adc\_ChannelIndex↵Type NumberOfChannel)

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

- Std\_ReturnType [Adc\\_SetPowerState](#) (Adc\_PowerStateRequestResultType \*Result)

*Enters the already prepared power state.*

- Std\_ReturnType [Adc\\_GetCurrentPowerState](#) (Adc\_PowerStateType \*CurrentPowerState, Adc\_Power↵StateRequestResultType \*Result)

*Get the current power state of the ADC HW unit.*

- Std\_ReturnType [Adc\\_GetTargetPowerState](#) (Adc\_PowerStateType \*TargetPowerState, Adc\_PowerState↵RequestResultType \*Result)

*Get the target power state of the ADC HW unit.*

- Std\_ReturnType [Adc\\_PreparePowerState](#) (Adc\_PowerStateType PowerState, Adc\_PowerStateRequest↵ResultType \*Result)

*Starts the needed process to allow the ADC HW module to enter the requested power state.*

- void [Adc\\_EnableCtuControlMode](#) (Adc\_HwUnitType Unit)

*Function to enable CTU control mode for an ADC unit.*

- void [Adc\\_DisableCtuControlMode](#) (Adc\_HwUnitType Unit)

*Function to disable CTU control mode for an ADC unit.*

- void [Adc\\_CtuEnableHwTrigger](#) (Adc\_CtuTrigSrcType TrigSource)

*Function to enable CTU hardware trigger.*

- void [Adc\\_CtuDisableHwTrigger](#) (Adc\_CtuTrigSrcType TrigSource)

*Function to disable CTU hardware trigger.*

- void [Adc\\_CtuStartConversion](#) (Adc\_CtuTrigSrcType TrigSource)

*Function to start CTU conversion.*

- Adc\_ValueGroupType [Adc\\_CtuReadConvData](#) (Adc\_HwUnitType AdcUnit)

*Function to read BCTU conversion data.*

- void [Adc\\_CtuReadConvResult](#) (Adc\_HwUnitType AdcUnit, [Adc\\_CtuResultType](#) \*pResult)

*Function to read BCTU conversion result.*

- void [Adc\\_CtuStopLoopConversions](#) (Adc\_CtuTrigSrcType TrigSource)

*Function to stop CTU loop conversion.*

- void [Adc\\_CtuReadFifoData](#) (Adc\_CtuFifoIdxType FifoIdx, uint16 \*Data, uint8 DataLength)

*Function to read CTU conversion data from FIFO.*

- void [Adc\\_CtuReadFifoResult](#) (Adc\_CtuFifoIdxType FifoIdx, [Adc\\_CtuFifoResultType](#) \*Result, uint8 ResultLength)

*Function to read CTU conversion results from FIFO.*

- void [Adc\\_CtuSetFifoWatermark](#) (Adc\_CtuFifoIdxType FifoIdx, uint8 Watermark)

*Function to set CTU FIFO watermark.*

- void [Adc\\_CtuEnableNotification](#) (Adc\_CtuNotificationType Notification)

*Function to enable CTU notification.*

- void [Adc\\_CtuDisableNotification](#) (Adc\_CtuNotificationType Notification)

*Function to disable CTU notification.*

- void [Adc\\_CtuSetList](#) (Adc\_HwUnitType CtuUnit, const [Adc\\_CtuListItemType](#) \*const ListItemsArray, const uint8 NumItems, const uint8 ListStartPosition)

*Function to set CTU list.*

- void [Adc\\_CtuSetListPointer](#) (Adc\_CtuTrigSrcType TrigSource, Adc\_CtuListPtrType ListPtr)

*Reconfigure the list pointer of a triggered CTU source.*

- uint16 [Adc\\_TempSenseCalculateTemp](#) (Adc\_HwUnitType Unit, const uint16 TempSenseAdcConvData)

*Function to calculate temperature on chip from provided data.*

- Std\_ReturnType [Adc\\_TempSenseGetTemp](#) (Adc\_HwUnitType Unit, uint16 \*const TempSenseVal)

*Function to get the temperature on chip directly.*

- Std\_ReturnType [Adc\\_TempSenseSetPowerMode](#) (Adc\_SetPowerModeType SetPowerMode)

*Set the Tempsense mode either to powerdown or normal.*

## 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 <a href="#">Adc_StatusType</a>	Conversion	Group status.

## Data Fields

Type	Name	Description
volatile <a href="#">Adc_GroupConversionStateType</a>	AlreadyConverted	Group was previously converted or not.
<a href="#">Adc_HwTriggeringType</a>	HwTriggering	hw trigger enabled/disabled
<a href="#">Adc_NotificationType</a>	Notification	notification enabled/disabled
volatile <a href="#">Adc_StreamNumSampleType</a>	ResultIndex	index into streaming buffer that is currently being filled
<a href="#">Adc_ValueGroupType</a> *	ResultsBufferPtr	Pointer to user result buffer array.
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 <a href="#">Adc_QueueIndexType</a>	SwNormalQueueIndex	Filled slots in the queue.
volatile <a href="#">Adc_GroupType</a>	SwNormalQueue[(1U)]	Queued groups indexes, always executing Queue[0].
volatile <a href="#">Adc_GroupType</a>	OngoingHwGroup	Ongoing hardware group ID.
volatile <a href="#">Adc_GroupType</a>	SwInjectedQueue[1U]	The depth of the software injected queue.
volatile <a href="#">Adc_QueueIndexType</a>	SwInjectedQueueIndex	Filled slots in the Sw injected queue.
boolean	CtuControlOngoing	Indicates Ctu control mode is ongoing.

**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\_NOT\_DISENGAGED

```
#define ADC_E_NOT_DISENGAGED
```

One or more ADC group/channel not in IDLE state.

Definition at line 188 of file Adc.h.

#### 6.1.3.13 ADC\_E\_POWER\_STATE\_NOT\_SUPPORTED

```
#define ADC_E_POWER_STATE_NOT_SUPPORTED
```

Unsupported power state request.

Definition at line 193 of file Adc.h.

### 6.1.3.14 ADC\_E\_TRANSITION\_NOT\_POSSIBLE

```
#define ADC_E_TRANSITION_NOT_POSSIBLE
```

Requested power state can not be reached directly.

Definition at line 198 of file Adc.h.

### 6.1.3.15 ADC\_E\_PERIPHERAL\_NOT\_PREPARED

```
#define ADC_E_PERIPHERAL_NOT_PREPARED
```

ADC not prepared for target power state.

Definition at line 203 of file Adc.h.

### 6.1.3.16 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.

### 6.1.3.17 ADC\_E\_SET\_HW\_UNIT\_POWER\_MODE

```
#define ADC_E_SET_HW_UNIT_POWER_MODE
```

An error occurred when the `Adc_SetHwUnitPowerMode` services is used.

Definition at line 216 of file Adc.h.

### 6.1.3.18 ADC\_E\_PARAM\_TRIGGER

```
#define ADC_E_PARAM_TRIGGER
```

Wrong trigger source to be used for the group.

Definition at line 223 of file Adc.h.

#### 6.1.3.19 ADC\_E\_WRONG\_ENABLE\_CH\_DISABLE\_CH\_GROUP

```
#define ADC_E_WRONG_ENABLE_CH_DISABLE_CH_GROUP
```

Adc\_EnableChannel or Adc\_DisableChannel services called with a wrong channel.

Definition at line 230 of file Adc.h.

#### 6.1.3.20 ADC\_E\_WRONG\_ENABLE\_CH\_DISABLE\_CH\_ID

```
#define ADC_E_WRONG_ENABLE_CH_DISABLE_CH_ID
```

Adc\_EnableChannel or Adc\_DisableChannel services called with a wrong channel identifier (ID).

Definition at line 234 of file Adc.h.

#### 6.1.3.21 ADC\_E\_WRONG\_CONF\_THRHLN\_VALUE

```
#define ADC_E_WRONG_CONF_THRHLN_VALUE
```

Adc\_ConfigureThreshold service is called using wrong values.

Definition at line 241 of file Adc.h.

#### 6.1.3.22 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.

#### 6.1.3.23 ADC\_E\_WRONG\_CTU\_TRIGGER

```
#define ADC_E_WRONG_CTU_TRIGGER
```

API service called using a wrong CTU trigger.

Definition at line 253 of file Adc.h.

### 6.1.3.24 ADC\_E\_WRONG\_CTU\_CLCR\_TRIGGER

```
#define ADC_E_WRONG_CTU_CLCR_TRIGGER
```

API service called using a wrong CTU CLCR trigger.

Definition at line 257 of file Adc.h.

### 6.1.3.25 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.

### 6.1.3.26 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.

### 6.1.3.27 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.

### 6.1.3.28 ADC\_E\_CONTROL\_MODE\_DISABLED

```
#define ADC_E_CONTROL_MODE_DISABLED
```

Error when a function which works only in CTU Control Mode is called when CTU control mode is disabled.

Definition at line 283 of file Adc.h.

#### 6.1.3.29 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.

#### 6.1.3.30 ADC\_DEINIT\_ID

```
#define ADC_DEINIT_ID
```

API service ID for Adc\_DeInit function.

Definition at line 307 of file Adc.h.

#### 6.1.3.31 ADC\_STARTGROUPCONVERSION\_ID

```
#define ADC_STARTGROUPCONVERSION_ID
```

API service ID for Adc\_StartGroupConversion function.

Definition at line 312 of file Adc.h.

#### 6.1.3.32 ADC\_STOPGROUPCONVERSION\_ID

```
#define ADC_STOPGROUPCONVERSION_ID
```

API service ID for Adc\_StopGroupConversion function.

Definition at line 317 of file Adc.h.

#### 6.1.3.33 ADC\_VALUEREADGROUP\_ID

```
#define ADC_VALUEREADGROUP_ID
```

API service ID for Adc\_ReadGroup function.

Definition at line 322 of file Adc.h.

### 6.1.3.34 ADC\_ENABLEHARDWARETRIGGER\_ID

```
#define ADC_ENABLEHARDWARETRIGGER_ID
```

API service ID for `Adc_EnableHardwareTrigger` function.

Definition at line 327 of file `Adc.h`.

### 6.1.3.35 ADC\_DISABLEHARDWARETRIGGER\_ID

```
#define ADC_DISABLEHARDWARETRIGGER_ID
```

API service ID for `Adc_DisableHardwareTrigger` function.

Definition at line 332 of file `Adc.h`.

### 6.1.3.36 ADC\_ENABLEGROUPNOTIFICATION\_ID

```
#define ADC_ENABLEGROUPNOTIFICATION_ID
```

API service ID for `Adc_EnableGroupNotification` function.

Definition at line 337 of file `Adc.h`.

### 6.1.3.37 ADC\_DISABLEGROUPNOTIFICATION\_ID

```
#define ADC_DISABLEGROUPNOTIFICATION_ID
```

API service ID for `Adc_DisableGroupNotification` function.

Definition at line 342 of file `Adc.h`.

### 6.1.3.38 ADC\_GETGROUPSTATUS\_ID

```
#define ADC_GETGROUPSTATUS_ID
```

API service ID for `Adc_GetGroupStatus` function.

Definition at line 347 of file `Adc.h`.



#### 6.1.3.39 ADC\_GETVERSIONINFO\_ID

```
#define ADC_GETVERSIONINFO_ID
```

API service ID for Adc\_GetVersionInfo function.

Definition at line 352 of file Adc.h.

#### 6.1.3.40 ADC\_GETSTREAMLASTPOINTER\_ID

```
#define ADC_GETSTREAMLASTPOINTER_ID
```

API service ID for Adc\_GetStreamLastPointer function.

Definition at line 357 of file Adc.h.

#### 6.1.3.41 ADC\_SETUPRESULTBUFFER\_ID

```
#define ADC_SETUPRESULTBUFFER_ID
```

API service ID for Adc\_SetupResultBuffer function.

Definition at line 362 of file Adc.h.

#### 6.1.3.42 ADC\_SETPOWERSTATE\_ID

```
#define ADC_SETPOWERSTATE_ID
```

API service ID for Adc\_SetPowerState function.

Definition at line 368 of file Adc.h.

#### 6.1.3.43 ADC\_GETCURRENTPOWERSTATE\_ID

```
#define ADC_GETCURRENTPOWERSTATE_ID
```

API service ID for Adc\_GetCurrentPowerState function.

Definition at line 373 of file Adc.h.

### 6.1.3.44 ADC\_GETTARGETPOWERSTATE\_ID

```
#define ADC_GETTARGETPOWERSTATE_ID
```

API service ID for Adc\_GetTargetPowerState function.

Definition at line 378 of file Adc.h.

### 6.1.3.45 ADC\_PREPAREPOWERSTATE\_ID

```
#define ADC_PREPAREPOWERSTATE_ID
```

API service ID for Adc\_PreparePowerState function.

Definition at line 383 of file Adc.h.

### 6.1.3.46 ADC\_HWRESULTREADGROUP\_ID

```
#define ADC_HWRESULTREADGROUP_ID
```

API service ID for Adc\_HwResultReadGroup function.

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

Definition at line 394 of file Adc.h.

### 6.1.3.47 ADC\_ENABLECTUTRIGGER\_ID

```
#define ADC_ENABLECTUTRIGGER_ID
```

API service ID for Adc\_EnableCTUTrigge function.

Definition at line 398 of file Adc.h.

### 6.1.3.48 ADC\_DISABLECTUTRIGGER\_ID

```
#define ADC_DISABLECTUTRIGGER_ID
```

API service ID for Adc\_DisableCTUTrigger function.

Definition at line 402 of file Adc.h.

#### 6.1.3.49 ADC\_SET\_HW\_UNIT\_POWER\_MODE\_ID

```
#define ADC_SET_HW_UNIT_POWER_MODE_ID
```

API service ID for `Adc_SetHwUnitPowerMode` function.

Definition at line 409 of file `Adc.h`.

#### 6.1.3.50 ADC\_CTU\_SET\_POWER\_MODE\_ID

```
#define ADC_CTU_SET_POWER_MODE_ID
```

API service ID for `Adc_CtuSetPowerMode` function.

Definition at line 414 of file `Adc.h`.

#### 6.1.3.51 ADC\_SETCLOCKMODE\_ID

```
#define ADC_SETCLOCKMODE_ID
```

API service ID for `Adc_SetClockMode` function.

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

#### 6.1.3.52 ADC\_ENABLE\_CHANNEL\_ID

```
#define ADC_ENABLE_CHANNEL_ID
```

API service ID for `Adc_EnableChannel` function.

Definition at line 429 of file `Adc.h`.

#### 6.1.3.53 ADC\_DISABLE\_CHANNEL\_ID

```
#define ADC_DISABLE_CHANNEL_ID
```

API service ID for `Adc_DisableChannel` function.

Definition at line 433 of file `Adc.h`.

### 6.1.3.54 ADC\_GETINJECTEDCONVERSIONSTATUS\_ID

```
#define ADC_GETINJECTEDCONVERSIONSTATUS_ID
```

API service ID for `Adc_GetInjectedConversionStatus` function.

Definition at line 440 of file `Adc.h`.

### 6.1.3.55 ADC\_CALIBRATE\_ID

```
#define ADC_CALIBRATE_ID
```

API service ID for `Adc_Calibrate` function.

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

### 6.1.3.56 ADC\_CONFIGURE\_THRESHOLD\_ID

```
#define ADC_CONFIGURE_THRESHOLD_ID
```

API service ID for `Adc_ConfigureThreshold` function.

Definition at line 458 of file `Adc.h`.

### 6.1.3.57 ADC\_CTU\_SET\_WRITE\_TRIG\_EN\_MASK\_ID

```
#define ADC_CTU_SET_WRITE_TRIG_EN_MASK_ID
```

API service ID for `Adc_CtuWriteTriggerEnableMask` function.

Definition at line 465 of file `Adc.h`.

### 6.1.3.58 ADC\_CTU\_SET\_TRIGGER\_ENABLE\_ID

```
#define ADC_CTU_SET_TRIGGER_ENABLE_ID
```

API service ID for `Adc_CtuSetTriggerEnable` function.

Definition at line 469 of file `Adc.h`.

**6.1.3.59 ADC\_CTU\_SET\_TRIGGER\_ADC\_CMD\_ADDRESS\_ID**

```
#define ADC_CTU_SET_TRIGGER_ADC_CMD_ADDRESS_ID
```

API service ID for `Adc_CtuSetTriggerAdcCmdAddress` function.

Definition at line 473 of file `Adc.h`.

**6.1.3.60 ADC\_CTU\_SET\_TRIGGER\_COMPARE\_ID**

```
#define ADC_CTU_SET_TRIGGER_COMPARE_ID
```

API service ID for `Adc_CtuSetTriggerCompare` function.

Definition at line 477 of file `Adc.h`.

**6.1.3.61 ADC\_SETCHANNEL\_ID**

```
#define ADC_SETCHANNEL_ID
```

API service ID for `Adc_SetChannel` function.

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

**6.1.3.62 ADC\_ENABLE\_CTU\_CONTROL\_MODE\_ID**

```
#define ADC_ENABLE_CTU_CONTROL_MODE_ID
```

API service ID for `Adc_EnableCtuControlMode` function.

Definition at line 491 of file `Adc.h`.

**6.1.3.63 ADC\_DISABLE\_CTU\_CONTROL\_MODE\_ID**

```
#define ADC_DISABLE_CTU_CONTROL_MODE_ID
```

API service ID for `Adc_DisableCtuControlMode` function.

Definition at line 495 of file `Adc.h`.

### 6.1.3.64 ADC\_APPLY\_CALIBRATION\_ID

```
#define ADC_APPLY_CALIBRATION_ID
```

API service ID for `Adc_ApplyCalibration` function.

Definition at line 531 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 )
```

## Module Documentation

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\_SetHwUnitPowerMode()**

```
Std_ReturnType Adc_SetHwUnitPowerMode (
    Adc_HwUnitType Unit,
    Adc_SetPowerModeType SetPowerMode )
```

Set the ADC mode either to powerdown or normal.

Set the ADC either to powerdown or normal mode.

## Parameters

in	<i>Unit</i>	Adc unit used. Recommended to use generated define for Adc Logical Unit Id
in	<i>SetPowerMode</i>	Power mode to set: normal or powerdown.

## Returns

Std\_ReturnType Standard return type. E\_OK: Transition successful. E\_NOT\_OK: Transition unsuccessful.

### 6.1.6.16 `Adc_CtuSetPowerMode()`

```
Std_ReturnType Adc_CtuSetPowerMode (
    Adc_HwUnitType CtuUnit,
    Adc_PowerStateType State )
```

Function to set BCTU/CTU power mode.

Set BCTU/CTU power mode.

Parameters

in	<i>CtuUnit</i>	Bctu/Ctu hardware unit. Recommended to use generated define for Bctu/Ctu Logical Unit Id.
in	<i>State</i>	Power state to be set

Returns

Std\_ReturnType Standard return type. E\_OK: Transition successful. E\_NOT\_OK: Transition unsuccessful.

### 6.1.6.17 `Adc_EnableCTUTrigger()`

```
void Adc_EnableCTUTrigger (
    Adc_GroupType Group,
    Adc_HwTriggerTimerType TriggerSource )
```

Enable the TriggerSource for group selected by Group parameter.

This Autosar Extension API is used to enable any one of the configured TriggerSource of the Group. When this Autosar Extension API is used to enable the trigger source the CTU interrupt will be disabled by the driver. So user has to call the Autosar Extension API `Adc_HwResultReadGroup` to read the converted result from the ADC hardware register.

Parameters

in	<i>Group</i>	Index of group.
in	<i>TriggerSource</i>	Trigger source to be used for the group. (Configuration file should contain it for that group).

Returns

void

#### 6.1.6.18 `Adc_DisableCTUTrigger()`

```
void Adc_DisableCTUTrigger (
    Adc_GroupType Group,
    Adc_HwTriggerTimerType TriggerSource )
```

Disable the TriggerSource for group selected by Group parameter.

This Autosar Extension API is used to disable the already enabled TriggerSource of the Group.

Parameters

in	<i>Group</i>	Index of group.
in	<i>TriggerSource</i>	Trigger source to be disabled for the group. (Configuration file should contain it for that group).

Returns

void

#### 6.1.6.19 `Adc_HwResultReadGroup()`

```
Std_ReturnType Adc_HwResultReadGroup (
    Adc_GroupType Group,
    Adc_ValueGroupType * DataPtr )
```

Read the result of the hardware triggered groups conversion result.

This Autosar Extension API is used to read the result of the hardware triggered groups conversion result from the ADC hardware register in this case the CTU interrupt will be disabled for the group. The VALID bit C←DR register will be cleared automatically when we read the results from the channel data register. If the user calls Autosar Extension function [Adc\\_HwResultReadGroup\(\)](#) once again before the next conversion takes place, the [Adc\\_HwResultReadGroup\(\)](#) returns E\_NOT\_OK.

Parameters

in	<i>Group</i>	Index of group.
in	<i>DataPtr</i>	Pointer to a buffer which will be filled by the conversion results.

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.

### 6.1.6.20 Adc\_EnableChannel()

```
void Adc_EnableChannel (
    Adc_GroupType Group,
    Adc_ChannelType Channel )
```

Enable an individual channel configured in SW-triggered (non-injected) ADC Group at initialization.

Enable an individual channel configured in SW-triggered (non-injected) ADC Group at initialization Use generated symbolic channel name defines (e.g. `AdcChannel_0_0`), because function assumes `ChannelId` to be in following format: Logical channel id on bits until position defined by `ADC_CHANNEL_SYMBOLIC_NAME_SHIFT_↔ HW_UNIT_ID_U16`, and for the rest the Logical Unit Id The driver will not update the values in result buffers corresponding to disabled channels, keeping in the buffer the last results from when the channel was enabled.

Parameters

in	<i>Adc_GroupType</i>	Group logical ID or group symbolic name
in	<i>Adc_ChannelType</i>	Symbolic name of channel

Returns

void.

### 6.1.6.21 Adc\_DisableChannel()

```
void Adc_DisableChannel (
    Adc_GroupType Group,
    Adc_ChannelType Channel )
```

Disable an individual channel configured in SW-triggered (non-injected) ADC Group at initialization.

Disable an individual channel configured in SW-triggered (non-injected) ADC Group at initialization Use generated symbolic channel name defines (e.g. `AdcChannel_0_0`), because function assumes `ChannelId` to be in following format: Logical channel id on bits until position defined by `ADC_CHANNEL_SYMBOLIC_NAME_SHIFT_↔ HW_UNIT_ID_U16`, and for the rest the Logical Unit Id The driver will not update the values in result buffers corresponding to disabled channels, keeping in the buffer the last results from when the channel was enabled.

Parameters

in	<i>Adc_GroupType</i>	Group logical ID or group symbolic name
in	<i>Adc_ChannelType</i>	Symbolic name of channel

Returns

void.

### 6.1.6.22 Adc\_GetInjectedConversionStatus()

```
Adc_StatusType Adc_GetInjectedConversionStatus (
    Adc_HwUnitType Unit )
```

Get the injected conversions status.

This function checks if an injected conversion (HW,SW) is ongoing

Parameters

in	<i>Unit</i>	Adc unit used. Recommended to use generated define for Adc Logical Unit Id
----	-------------	--

Returns

Adc\_StatusType Status of the ADC HW unit. ADC\_IDLE: SW,HW Injection or Hardware Trigger group are idle. ADC\_BUSY: SW,HW Injection or Hardware Trigger is in progress.

### 6.1.6.23 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

**6.1.6.24   Adc\_ApplyCalibration()**

```
Std_ReturnType Adc_ApplyCalibration (
    Adc_HwUnitType Unit,
    const uint32 BufferLength,
    const sint16 *const UncalibratedBuffer,
    sint16 *const CalibratedBuffer )
```

Calibrate the converted data of the SDADC.

This function applies calibration to the uncalibrated converted data acquired by DMA or Sdadc\_Ip\_GetRawConv↔DataFifo function. In the case uncalibrated converted data acquired by DMA, this function should be called in the DMA callback. NOTE: this function is only used for SDADC hardware units.

Parameters

in	<i>Unit</i>	Adc unit used. Recommended to use generated define for Adc Logical Unit Id.
in	<i>BufferLength</i>	The number of entries in the uncalibrated data buffer.
in	<i>UncalibratedBuffer</i>	The pointer to the converted data array which contains uncalibrated data.
out	<i>CalibratedBuffer</i>	The pointer to the data array where the calibrated data will be stored.

Returns

Std\_ReturnType. Status of the apply calibration result. E\_OK: Calibration applied successfully. E\_NOT↔\_OK: Invalid parameter given.

**6.1.6.25   Adc\_SelfTest()**

```
Std_ReturnType Adc_SelfTest (
    Adc_HwUnitType Unit )
```

Executes hardware Self Test of a ADC HW unit.

This function checks if the ADC HW unit is functioning correctly

Parameters

in	<i>Unit</i>	Adc unit used. Recommended to use generated define for Adc Logical Unit Id
----	-------------	--

Returns

Std\_ReturnType. Status of the ADC HW unit Self Test.

### 6.1.6.26 `Adc_ConfigureThreshold()`

```
void Adc_ConfigureThreshold (
    Adc_ThresholdControlIndexType ThresholdControlIndex,
    Adc_WdgThresholdValueType LowValue,
    Adc_WdgThresholdValueType HighValue )
```

Function to reconfigure High and Low thresholds for a given threshold control index.

This function is used to reconfigure High and Low thresholds for a given threshold control index. Use generated symbolic name defines of threshold register (e.g. `AdcThresholdControl_0_0`), because function assumes `ThresholdControlIndex` to be in following format: Logical threshold id on bits until position defined by `ADC_THRESHOLD_SYMBOLIC_NAME_SHIFT_HW_UNIT_ID_U16`, and for the rest the Logical Unit Id

Parameters

in	<i>ThresholdControlIndex</i>	Symbolic name of threshold control
in	<i>LowValue</i>	Low threshold value of the threshold control
in	<i>HighValue</i>	High threshold value of the threshold control

Returns

void

### 6.1.6.27 `Adc_EnableWdgNotification()`

```
void Adc_EnableWdgNotification (
    Adc_ChannelType ChannelId )
```

Enable notification of a channel that has watchdog functionality configured at initialization.

Enable notification of a channel that has watchdog functionality configured at initialization Use generated symbolic channel name defines (e.g. `AdcChannel_0_0`), because function assumes `ChannelId` to be in following format: Logical channel id on bits until position defined by `ADC_CHANNEL_SYMBOLIC_NAME_SHIFT_HW_UNIT_ID_U16`, and for the rest the Logical Unit Id

Parameters

in	<i>Adc_ChannelType</i>	Symbolic name of channel
----	------------------------	--------------------------

Returns

void.

### 6.1.6.28 `Adc_DisableWdgNotification()`

```
void Adc_DisableWdgNotification (
    Adc_ChannelType ChannelId )
```

Disable notification of a channel that has watchdog functionality configured at initialization.

Disable notification of a channel that has watchdog functionality configured at initialization Use generated symbolic channel name defines (e.g. `AdcChannel_0_0`), because function assumes `ChannelId` to be in following format: Logical channel id on bits until position defined by `ADC_CHANNEL_SYMBOLIC_NAME_SHIFT_HW_UNIT_ID_↔U16`, and for the rest the Logical Unit Id

Parameters

in	<i>Adc_ChannelType</i>	Symbolic name of channel
----	------------------------	--------------------------

Returns

void.

### 6.1.6.29 `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.

### 6.1.6.30 `Adc_CtuWriteTriggerEnableMask()`

```
Std_ReturnType Adc_CtuWriteTriggerEnableMask (
    Adc_CtuTrigSrcType Trigger,
    uint8 ThcrValue )
```



Function to set the trigger handler control registers of the CTU IPL.

Full configurability of CTU THCR1 & THCR2 registers of the CTU IPL. This function has in input the CTU trigger (0...7) and the mask (8 bit) to enable the External Trigger and Timer output for that Trigger, without considering the ADC command output enable. See from the RM the THCR1 and THCR2 registers for the right Thcr\_value to use.

Note

The function Service ID[hex]: 0x35.

Parameters

in	<i>Trigger</i>	Index of the trigger: 0...7.
in	<i>ThcrValue</i>	THCRx mask value (only with bits for Ext. Trigger and Timer) for the selected input trigger.

Returns

Std\_ReturnType E\_OK or E\_NOT\_OK. E\_OK In case of successful settings. E\_NOT\_OK In case of unsuccessful settings.

#### 6.1.6.31 Adc\_CtuSetTriggerEnable()

```
Std_ReturnType Adc_CtuSetTriggerEnable (
    Adc_CtuTrigSrcType Trigger,
    boolean Enable )
```

Function to set or clear the bit Tx\_E of the trigger handler control registers of the CTU IPL.

Full configurability of CTU THCR1 & THCR2 registers of the CTU IPL. This function has in input the CTU trigger (0...7) and a Boolean to set to clear the right trigger bit. See from the RM the THCR1 and THCR2 registers for the right API use.

Parameters

in	<i>Trigger</i>	Index of the trigger: 0...7.
in	<i>Enable</i>	if True, the bit Tx_E shall be set to 1, 0 otherwise

Note

The function Service ID[hex]: 0x36.

Returns

Std\_ReturnType E\_OK or E\_NOT\_OK. E\_OK In case of successful settings. E\_NOT\_OK In case of unsuccessful settings.

6.1.6.32    **Adc\_CtuSetTriggerAdcCmdAddress()**

```
Std_ReturnType Adc_CtuSetTriggerAdcCmdAddress (
    Adc_CtuTrigSrcType Trigger,
    uint8 ClcrValue )
```

Function to set the cmd list handler control registers of the CTU IPL.

Full configurability of CLCR1 & CLCR2 registers of the CTU IPL. This function has in input the CTU trigger (0...7) and the position of the first command in the command list. See from the RM the CLCR1 and CLCR2 registers for the right API use.

Parameters

in	<i>Trigger</i>	Index of the trigger: 0...7.
in	<i>ClcrValue</i>	Position of the first command

Note

The function Service ID[hex]: 0x37.

Returns

Std\_ReturnType E\_OK or E\_NOT\_OK. E\_OK In case of successful settings. E\_NOT\_OK In case of unsuccessful settings.

6.1.6.33    **Adc\_CtuSetTriggerCompare()**

```
Std_ReturnType Adc_CtuSetTriggerCompare (
    Adc_CtuTrigSrcType Trigger,
    uint16 CompareValue )
```

Function to set the trigger compare registers of the CTU IPL.

Full configurability of CTU TxCR registers of the CTU IPL. This function has in input the CTU trigger (0...7) and the comparator value See from the RM the TxCR trigger compare registers for the right compare\_value to use.

Note

The function Service ID[hex]: 0x38.

## Parameters

in	<i>Trigger</i>	Index of the trigger: 0...7.
in	<i>CompareValue</i>	value to set in the register TxCR.

## Returns

Std\_ReturnType E\_OK or E\_NOT\_OK. E\_OK In case of successful settings. E\_NOT\_OK In case of unsuccessful settings.

**6.1.6.34 Adc\_SetChannel()**

```
void Adc_SetChannel (
    Adc_GroupType Group,
    const Adc_GroupDefType * Channel,
    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.

### 6.1.6.35 Adc\_SetPowerState()

```
Std_ReturnType Adc_SetPowerState (
    Adc_PowerStateRequestResultType * Result )
```

Enters the already prepared power state.

This API configures the Adc module so that it enters the already prepared power state, chosen between a predefined set of configured ones.

Parameters

out	<i>Result</i>	Pointer to a variable to store the result of this function
-----	---------------	--

Returns

Std\_ReturnType Standard return type. E\_OK: Power Mode changed. E\_NOT\_OK: Request rejected.

### 6.1.6.36 Adc\_GetCurrentPowerState()

```
Std_ReturnType Adc_GetCurrentPowerState (
    Adc_PowerStateType * CurrentPowerState,
    Adc_PowerStateRequestResultType * Result )
```

Get the current power state of the ADC HW unit.

This API returns the current power state of the ADC HW unit.

Parameters

out	<i>CurrentPowerState</i>	The current power mode of the ADC HW Unit is returned in this parameter
out	<i>Result</i>	Pointer to a variable to store the result of this function

Returns

Std\_ReturnType Standard return type. E\_OK: Mode could be read. E\_NOT\_OK: Service is rejected.

### 6.1.6.37 `Adc_GetTargetPowerState()`

```
Std_ReturnType Adc_GetTargetPowerState (
    Adc_PowerStateType * TargetPowerState,
    Adc_PowerStateRequestResultType * Result )
```

Get the target power state of the ADC HW unit.

This API returns the target power state of the ADC HW unit.

Parameters

out	<i>TargetPowerState</i>	The Target power mode of the ADC HW Unit is returned in this parameter.
out	<i>Result</i>	Pointer to a variable to store the result of this function.

Returns

Std\_ReturnType Standard return type. E\_OK: Mode could be read. E\_NOT\_OK: Service is rejected.

### 6.1.6.38 `Adc_PreparePowerState()`

```
Std_ReturnType Adc_PreparePowerState (
    Adc_PowerStateType PowerState,
    Adc_PowerStateRequestResultType * Result )
```

Starts the needed process to allow the ADC HW module to enter the requested power state.

This API starts the needed process to allow the ADC HW module to enter the requested power state.

Parameters

in	<i>PowerState</i>	The target power state intended to be attained.
out	<i>Result</i>	Pointer to a variable to store the result of this function.

Returns

Std\_ReturnType Standard return type. E\_OK: Mode could be read. E\_NOT\_OK: Service is rejected.

### 6.1.6.39 `Adc_EnableCtuControlMode()`

```
void Adc_EnableCtuControlMode (
    Adc_HwUnitType Unit )
```

Function to enable CTU control mode for an ADC unit.

Enable CTU control mode for an ADC unit. This function to enable CTU control mode for Adc. When a unit works in CTU control mode, no other conversions shall run in parallel(Adc). The only conversions occurring shall be the ones defined in the CTU configuration.

Parameters

in	<i>Unit</i>	Adc unit used. Recommended to use generated define for Adc Logical Unit Id (e.g: AdcHwUnit_0)
----	-------------	---

Returns

void

### 6.1.6.40 Adc\_DisableCtuControlMode()

```
void Adc_DisableCtuControlMode (
    Adc_HwUnitType Unit )
```

Function to disable CTU control mode for an ADC unit.

Disable CTU control mode for an ADC unit. This function to disable CTU control mode for Adc. The other Adc conversions can run in software trigger normal mode, software trigger injected mode or hardware trigger mode.

Parameters

in	<i>Unit</i>	Adc unit used. Recommended to use generated define for Adc Logical Unit Id (e.g: AdcHwUnit_0)
----	-------------	---

Returns

void

### 6.1.6.41 Adc\_CtuEnableHwTrigger()

```
void Adc_CtuEnableHwTrigger (
    Adc_CtuTrigSrcType TrigSource )
```

Function to enable CTU hardware trigger.

Enable CTU hardware trigger.

## Parameters

in	<i>TrigSource</i>	Symbolic name of source trigger index (e.g CtuHwUnit_0_CtuTriggerCfg_0; BctuHwUnit_0_BctuInternalTrigger_0) Symbolic name format: <ul style="list-style-type: none"> <li>• Bit fields [15-8]: The CTU Logical Unit Id</li> <li>• Bit fields [7-0]: Trigger Control index in CTU unit.</li> </ul>
----	-------------------	--

## Returns

void

**6.1.6.42   Adc\_CtuDisableHwTrigger()**

```
void Adc_CtuDisableHwTrigger (
    Adc_CtuTrigSrcType TrigSource )
```

Function to disable CTU hardware trigger.

Disable CTU hardware trigger.

## Parameters

in	<i>TrigSource</i>	Symbolic name of source trigger index (e.g CtuHwUnit_0_CtuTriggerCfg_0; BctuHwUnit_0_BctuInternalTrigger_0) Symbolic name format: <ul style="list-style-type: none"> <li>• Bit fields [15-8]: The CTU Logical Unit Id</li> <li>• Bit fields [7-0]: Trigger Control index in CTU unit.</li> </ul>
----	-------------------	--

## Returns

void

**6.1.6.43   Adc\_CtuStartConversion()**

```
void Adc_CtuStartConversion (
    Adc_CtuTrigSrcType TrigSource )
```

Function to start CTU conversion.

Start CTU conversion.

Parameters

in	<i>TrigSource</i>	Symbolic name of source trigger index (e.g CtuHwUnit_0_CtuTriggerCfg_0; BctuHwUnit_0_BctuInternalTrigger_0) Symbolic name format: <ul style="list-style-type: none"><li>• Bit fields [15-8]: The CTU Logical Unit Id</li><li>• Bit fields [7-0]: Trigger Control index in CTU unit.</li></ul>
----	-------------------	---

Returns

void

6.1.6.44    **Adc\_\_CtuReadConvData()**

```
Adc_ValueGroupType Adc_CtuReadConvData (
    Adc_HwUnitType AdcUnit )
```

Function to read BCTU conversion data.

Read CTU conversion data.

Parameters

in	<i>AdcUnit</i>	Adc HW unit. Recommended to use generated define for Adc Logical Unit Id.
----	----------------	---

Returns

Adc\_ValueGroupType Conversion result.

6.1.6.45    **Adc\_\_CtuReadConvResult()**

```
void Adc_CtuReadConvResult (
    Adc_HwUnitType AdcUnit,
    Adc_CtuResultType * pResult )
```

Function to read BCTU conversion result.

Read BCTU conversion result.



## Parameters

in	<i>AdcUnit</i>	Adc HW unit. Recommended to use generated define for Adc Logical Unit Id.
out	<i>pResult</i>	Adc result structure.

## Returns

void

**6.1.6.46   Adc\_CtuStopLoopConversions()**

```
void Adc_CtuStopLoopConversions (
    Adc_CtuTrigSrcType TrigSource )
```

Function to stop CTU loop conversion.

Stop CTU loop conversions.

## Parameters

in	<i>TrigSource</i>	Symbolic name of source trigger index (e.g CtuHwUnit_0_CtuTriggerCfg_0; BctuHwUnit_0_BctuInternalTrigger_0) Symbolic name format: <ul style="list-style-type: none"> <li>• Bit fields [15-8]: The CTU Logical Unit Id</li> <li>• Bit fields [7-0]: Trigger Control index in CTU unit.</li> </ul>
----	-------------------	--

## Returns

void

**6.1.6.47   Adc\_CtuReadFifoData()**

```
void Adc_CtuReadFifoData (
    Adc_CtuFifoIdxType FifoIdx,
    uint16 * Data,
    uint8 DataLength )
```

Function to read CTU conversion data from FIFO.

Read CTU conversion data from FIFO.

### Parameters

in	<i>FifoIdx</i>	Symbolic name of CTU FIFO index (e.g: CtuHwUnit_0_CtuResultFifos_0; BctuHwUnit_0_BctuResultFifos_0) Symbolic name format: <ul style="list-style-type: none"> <li>• Bit fields [15-8]: The CTU Logical Unit Id</li> <li>• Bit fields [7-0]: FIFO index in CTU unit.</li> </ul>
out	<i>Data</i>	Pointer to pre-allocated result array.
in	<i>DataLength</i>	Max amount of results to be read.

### Returns

void

#### 6.1.6.48 Adc\_CtuReadFifoResult()

```
void Adc_CtuReadFifoResult (
    Adc_CtuFifoIdxType FifoIdx,
    Adc_CtuFifoResultType * Result,
    uint8 ResultLength )
```

Function to read CTU conversion results from FIFO.

Read CTU conversion results from FIFO.

### Parameters

in	<i>FifoIdx</i>	Symbolic name of CTU FIFO index (e.g: CtuHwUnit_0_CtuResultFifos_0; BctuHwUnit_0_BctuResultFifos_0) Symbolic name format: <ul style="list-style-type: none"> <li>• Bit fields [15-8]: The CTU Logical Unit Id</li> <li>• Bit fields [7-0]: FIFO index in CTU unit.</li> </ul>
out	<i>Result</i>	Pointer to pre-allocated result array.
in	<i>ResultLength</i>	Max amount of results to be read.

### Returns

void

#### 6.1.6.49 `Adc_CtuSetFifoWatermark()`

```
void Adc_CtuSetFifoWatermark (
    Adc_CtuFifoIdxType FifoIdx,
    uint8 Watermark )
```

Function to set CTU FIFO watermark.

Set CTU FIFO watermark.

Parameters

in	<i>FifoIdx</i>	Symbolic name of CTU FIFO index (e.g: CtuHwUnit_0__CtuResultFifos_0; BctuHwUnit_0__BctuResultFifos_0) Symbolic name format: <ul style="list-style-type: none"> <li>• Bit fields [15-8]: The CTU Logical Unit Id</li> <li>• Bit fields [7-0]: FIFO index in CTU unit.</li> </ul>
in	<i>Watermark</i>	Watermark value.

Returns

void

#### 6.1.6.50 `Adc_CtuEnableNotification()`

```
void Adc_CtuEnableNotification (
    Adc_CtuNotificationType Notification )
```

Function to enable CTU notification.

Enable CTU notification.

Parameters

in	<i>Notification</i>	Notification to be enabled. Note: CTU logical mask must be 'OR'ed together with other notification masks (e.g: ADC_IPW_CTU_NOTIF_CTU_LOGICAL_0   ADC_IPW_CTU_NOTIF_FIFO0_OVERFLOW)
----	---------------------	--

Returns

void

Module Documentation

6.1.6.51    **Adc\_CtuDisableNotification()**

```
void Adc_CtuDisableNotification (
    Adc_CtuNotificationType Notification )
```

Function to disable CTU notification.

Disable CTU notification.

Parameters

in	<i>Notification</i>	Notification to be disabled. Note: CTU logical mask must be 'OR'ed together with other notification masks (e.g: ADC_IPW_CTU_NOTIF_CTU_LOGICAL_0   ADC_IPW_CTU_NOTIF_FIFO0_OVERFLOW)
----	---------------------	---

Returns

void

6.1.6.52    **Adc\_CtuSetList()**

```
void Adc_CtuSetList (
    Adc_HwUnitType CtuUnit,
    const Adc_CtuListItemType *const ListItemsArray,
    const uint8 NumItems,
    const uint8 ListStartPosition )
```

Function to set CTU list.

Reconfigure the CTU list of conversions in CTU Control Mode.

Parameters

in	<i>CtuUnit</i>	Symbolic names of CTU Hardware units. (CTU logical unit ID)
in	<i>ListItemsArray</i>	Pointer to list items array to be set.
in	<i>NumItems</i>	Number of items in the array.
in	<i>ListStartPosition</i>	Start position of the list.

Returns

void

### 6.1.6.53 `Adc_CtuSetListPointer()`

```
void Adc_CtuSetListPointer (
    Adc_CtuTrigSrcType TrigSource,
    Adc_CtuListPtrType ListPtr )
```

Reconfigure the list pointer of a triggered CTU source.

Reconfigure the list pointer of a trigger source in CTU Control Mode.

Parameters

in	<i>TrigSource</i>	Symbolic name of source trigger index (e.g <code>CtuHwUnit_0_CtuTriggerCfg_0</code> ; <code>BctuHwUnit_0_BctuInternalTrigger_0</code> ) Symbolic name format: <ul style="list-style-type: none"> <li>• Bit fields [15-8]: The CTU Logical Unit Id</li> <li>• Bit fields [7-0]: Trigger Control index in CTU unit.</li> </ul>
in	<i>ListPtr</i>	Position of the first channel of the command list.

Returns

void

### 6.1.6.54 `Adc_TempSenseCalculateTemp()`

```
uint16 Adc_TempSenseCalculateTemp (
    Adc_HwUnitType Unit,
    const uint16 TempSenseAdcConvData )
```

Function to calculate temperature on chip from provided data.

Calculates temperature on chip from provided data.

Parameters

in	<i>Unit</i>	Adc unit used. Recommended to use generated define for Adc Logical Unit Id.
in	<i>TempSenseConvData</i>	Data measured on the ADC internal channel for TempSense. (1 bit for the sign, 11 bits for the integer part and 4 bits for the decimal part)

Returns

uint16 Temperature value on chip.in degrees C, expressed in fixed point format.

### 6.1.6.55 Adc\_TempSenseGetTemp()

```
Std_ReturnType Adc_TempSenseGetTemp (
    Adc_HwUnitType Unit,
    uint16 *const TempSenseVal )
```

Function to get the temperature on chip directly.

This function starts a normal software conversion with one-shot mode on tempsense channel and calculates the temperature on chip from the data conversion. The function is synchronous: waits until the ADC conversion completes or timeout occurs.

Parameters

in	<i>Unit</i>	Adc unit used. Recommended to use generated define for Adc Logical Unit Id.
out	<i>TempSenseVal</i>	Temperature value on chip in degrees C, expressed in fixed point format. (1 bit for the sign, 11 bits for the integer part and 4 bits for the decimal part)

Returns

Std\_ReturnType Standard return type. E\_OK: temperature read successful E\_NOT\_OK: operation failed

### 6.1.6.56 Adc\_TempSenseSetPowerMode()

```
Std_ReturnType Adc_TempSenseSetPowerMode (
    Adc_SetPowerModeType SetPowerMode )
```

Set the Tempsense mode either to powerdown or normal.

Set the Tempsense mode either to powerdown or normal.

Parameters

in	<i>SetPowerMode</i>	Power mode to set: normal or powerdown.
----	---------------------	---

Returns

Std\_ReturnType Successful/Unsuccessful transition.

## 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 Sar IPL

### 6.2.1 Detailed Description Adc Sar HW module.

Adc Sar IP layer hardware module.

#### Data Structures

- struct [Adc\\_Sar\\_Ip\\_ChanConfigType](#)  
*Defines the channel configuration. [More...](#)*
- struct [Adc\\_Sar\\_Ip\\_ClockConfigType](#)  
*Defines the ADC clock configuration. [More...](#)*
- struct [Adc\\_Sar\\_Ip\\_WdgThresholdType](#)  
*Defines the upper and lower thresholds for analog watchdog. [More...](#)*
- struct [Adc\\_Sar\\_Ip\\_ChanResultType](#)  
*Defines the data regarding a conversion, beyond the conversion data. [More...](#)*
- struct [Adc\\_Sar\\_Ip\\_ChansIdxMaskType](#)  
*Defines configuration of channels in a chain. [More...](#)*
- struct [Adc\\_Sar\\_Ip\\_SelfTestThresholdType](#)  
*Defines configuration of self-test threshold values. [More...](#)*
- struct [Adc\\_Sar\\_Ip\\_ConfigType](#)  
*Defines the module configuration. [More...](#)*
- struct [Adc\\_Sar\\_Ip\\_StateStructType](#)  
*Structure used to store runtime info. [More...](#)*

#### Macros

- `#define ADC_SAR_IP_NOTIF_FLAG_NORMAL_ENDCHAIN`  
*Macros for status and notification flags.*
- `#define ADC_SAR_IP_CHAN_NOTIF_EOC`  
*Macros for channel notifications.*
- `#define ADC_SAR_IP_WDG_HIGH_FLAG`  
*Macros for watchdog registers.*

#### Types Reference

- typedef void [Adc\\_Sar\\_Ip\\_WdgNotificationType](#)(const uint16 ChanIdx, const uint8 Flags)  
*Defines the watchdog notification header.*

## Enum Reference

- enum [Adc\\_Sar\\_Ip\\_StatusType](#)  
*ADC\_SAR status return type.*
- enum [Adc\\_Sar\\_Ip\\_ConvModeType](#)  
*Conversion mode selection (One-shot or Scan)*
- enum [Adc\\_Sar\\_Ip\\_ClockSelType](#)  
*Converter input clock.*
- enum [Adc\\_Sar\\_Ip\\_ExtTriggerEdgeType](#)  
*External Trigger selection.*
- enum [Adc\\_Sar\\_Ip\\_ExtTriggerSourceType](#)  
*External Trigger Source Enable.*
- enum [Adc\\_Sar\\_Ip\\_ConvChainType](#)  
*Conversion chain selection.*
- enum [Adc\\_Sar\\_Ip\\_DataAlignedType](#)  
*Data alignment selection.*
- enum [Adc\\_Sar\\_Ip\\_ClearSourceType](#)  
*Clear DMA source.*
- enum [Adc\\_Sar\\_Ip\\_PresamplingSourceType](#)  
*Presampling Voltage selection.*
- enum [Adc\\_Sar\\_Ip\\_ChainGroupType](#)  
*Channel group selection.*
- enum [Adc\\_Sar\\_Ip\\_AvgSelectType](#)  
*Averaging selection.*
- enum [Adc\\_Sar\\_Ip\\_Resolution](#)  
*Adc group conversion resolution.*

## Function Reference

- uint32 [Adc\\_Sar\\_Ip\\_GetDataAddress](#) (uint32 u32Instance, uint32 u32ChannelIndex)  
*Return the address of the specified data register.*
- [Adc\\_Sar\\_Ip\\_StatusType](#) [Adc\\_Sar\\_Ip\\_Init](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_ConfigType](#) \*const pConfig)  
*Initialize ADC\_SAR module.*
- [Adc\\_Sar\\_Ip\\_StatusType](#) [Adc\\_Sar\\_Ip\\_Deinit](#) (const uint32 u32Instance)  
*Deinitialize ADC\_SAR module.*
- void [Adc\\_Sar\\_Ip\\_ChainConfig](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_ChansIdxMaskType](#) \*const pChansIdxMask, const [Adc\\_Sar\\_Ip\\_ConvChainType](#) pChainType)  
*Configures the converter chains the given configuration structure.*
- void [Adc\\_Sar\\_Ip\\_EnableChannel](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_ConvChainType](#) pChainType, const uint32 u32ChnIdx)  
*Enable a channel.*
- void [Adc\\_Sar\\_Ip\\_DisableChannel](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_ConvChainType](#) pChainType, const uint32 u32ChnIdx)  
*Disable a channel.*
- void [Adc\\_Sar\\_Ip\\_SetResolution](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_Resolution](#) eResolution)



*Set conversion resolution.*

- void [Adc\\_Sar\\_Ip\\_StartConversion](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_ConvChainType](#) pChainType)

*Start conversion.*

- uint32 [Adc\\_Sar\\_Ip\\_GetStatusFlags](#) (const uint32 u32Instance)

*Get the status flags.*

- void [Adc\\_Sar\\_Ip\\_ClearStatusFlags](#) (const uint32 u32Instance, const uint32 u32Mask)

*Clear the status flags.*

- [Adc\\_Sar\\_Ip\\_StatusType](#) [Adc\\_Sar\\_Ip\\_SelfTest](#) (const uint32 u32Instance)

*Self test.*

- uint32 [Adc\\_Sar\\_Ip\\_GetConvDataToArray](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_ConvChainType](#) pChainType, const uint32 u32Length, uint16 \*const pResults)

*Get conversion results for a conversion chain.*

- uint32 [Adc\\_Sar\\_Ip\\_GetConvResultsToArray](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_ConvChainType](#) pChainType, const uint32 u32Length, [Adc\\_Sar\\_Ip\\_ChanResultType](#) \*const pResults)

*Get conversion results for a conversion chain with extended information.*

- uint16 [Adc\\_Sar\\_Ip\\_GetConvData](#) (const uint32 u32Instance, const uint32 u32ChnIdx)

*Return the result of the conversion.*

- void [Adc\\_Sar\\_Ip\\_GetConvResult](#) (const uint32 u32Instance, const uint32 u32ChnIdx, const [Adc\\_Sar\\_Ip\\_ConvChainType](#) pChainType, [Adc\\_Sar\\_Ip\\_ChanResultType](#) \*const pResult)

*Return the result and the status of the conversion.*

- [Adc\\_Sar\\_Ip\\_StatusType](#) [Adc\\_Sar\\_Ip\\_DoCalibration](#) (const uint32 u32Instance)

*Perform Calibration of the ADC.*

- [Adc\\_Sar\\_Ip\\_StatusType](#) [Adc\\_Sar\\_Ip\\_Powerup](#) (const uint32 u32Instance)

*Power up the ADC.*

- [Adc\\_Sar\\_Ip\\_StatusType](#) [Adc\\_Sar\\_Ip\\_Powerdown](#) (const uint32 u32Instance)

*Power down the ADC.*

- void [Adc\\_Sar\\_Ip\\_EnableNotifications](#) (const uint32 u32Instance, const uint32 u32NotificationMask)

*Enable ADC interrupts.*

- void [Adc\\_Sar\\_Ip\\_DisableNotifications](#) (const uint32 u32Instance, const uint32 u32NotificationMask)

*Disable ADC interrupts.*

- void [Adc\\_Sar\\_Ip\\_EnableChannelNotifications](#) (const uint32 u32Instance, const uint32 u32ChnIdx, const uint32 u32Mask)

*Enable ADC interrupt for a channel.*

- void [Adc\\_Sar\\_Ip\\_DisableChannelNotifications](#) (const uint32 u32Instance, const uint32 u32ChnIdx, const uint32 u32Mask)

*Disable ADC interrupt for a channel.*

- [Adc\\_Sar\\_Ip\\_StatusType](#) [Adc\\_Sar\\_Ip\\_SetClockMode](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_ClockConfigType](#) \*const pConfig)

*Set the ADC clocks.*

- void [Adc\\_Sar\\_Ip\\_SetSampleTimes](#) (const uint32 u32Instance, const uint8 \*const aSampleTimes)

*Set the sample times.*

- void [Adc\\_Sar\\_Ip\\_SetAveraging](#) (const uint32 u32Instance, const boolean bAvgEn, const [Adc\\_Sar\\_Ip\\_AvgSelectType](#) eAvgSel)

*Configure averaging.*

- void [Adc\\_Sar\\_Ip\\_AbortConversion](#) (const uint32 u32Instance)

*Abort ongoing conversion.*

- [Adc\\_Sar\\_Ip\\_StatusType](#) [Adc\\_Sar\\_Ip\\_AbortChain](#) (const uint32 u32Instance, const boolean bBlocking, const boolean bAllowRestart)  
*Abort ongoing chain conversion.*
- void [Adc\\_Sar\\_Ip\\_SetPresamplingSource](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_ChanGroupType](#) p← ChanGroup, const [Adc\\_Sar\\_Ip\\_PresamplingSourceType](#) pPresampleSource)  
*Set the Presampling Source for the channel group.*
- void [Adc\\_Sar\\_Ip\\_EnableChannelPresampling](#) (const uint32 u32Instance, const uint32 u32ChnIdx)  
*Enable Presampling on one channel.*
- void [Adc\\_Sar\\_Ip\\_DisableChannelPresampling](#) (const uint32 u32Instance, const uint32 u32ChnIdx)  
*Disable Presampling on one channel.*
- void [Adc\\_Sar\\_Ip\\_EnablePresampleConversion](#) (const uint32 u32Instance)  
*Enable Conversion Presampled Data.*
- void [Adc\\_Sar\\_Ip\\_DisablePresampleConversion](#) (const uint32 u32Instance)  
*Disable Conversion of Presampled Data.*
- void [Adc\\_Sar\\_Ip\\_EnableDma](#) (const uint32 u32Instance)  
*Enable DMA Requests.*
- void [Adc\\_Sar\\_Ip\\_DisableDma](#) (const uint32 u32Instance)  
*Disable DMA Requests.*
- void [Adc\\_Sar\\_Ip\\_EnableChannelDma](#) (const uint32 u32Instance, const uint32 u32ChnIdx)  
*Enable DMA on one channel.*
- void [Adc\\_Sar\\_Ip\\_DisableChannelDma](#) (const uint32 u32Instance, const uint32 u32ChnIdx)  
*Disable DMA on one channel.*
- void [Adc\\_Sar\\_Ip\\_DisableChannelDmaAll](#) (const uint32 u32Instance)  
*Disable DMA on all channels.*
- void [Adc\\_Sar\\_Ip\\_SetDmaClearSource](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_ClearSourceType](#) p← DmaClear)  
*Set DMA Request Clear Source.*
- void [Adc\\_Sar\\_Ip\\_SetWdgThreshold](#) (const uint32 u32Instance, const uint8 u8RegisterIdx, const [Adc\\_Sar\\_Ip\\_WdgThresholdType](#) \*const pThresholdValues)  
*Configure watchdog threshold register.*
- void [Adc\\_Sar\\_Ip\\_SetConversionMode](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_ConvModeType](#) e← ConvMode)  
*Configure conversion mode.*
- void [Adc\\_Sar\\_Ip\\_SetExternalTrigger](#) (const uint32 u32Instance, const [Adc\\_Sar\\_Ip\\_ExtTriggerEdgeType](#) eTrggerEdge, const [Adc\\_Sar\\_Ip\\_ExtTriggerSourceType](#) eTrggerSrc)  
*Configure external trigger.*

## CALBISTREG - Control And Calibration Status

- #define [ADC\\_SAR\\_IP\\_CALBISTREG\\_TEST\\_EN\\_MASK](#)
- #define [ADC\\_SAR\\_IP\\_CALBISTREG\\_TEST\\_EN\(x\)](#)
- #define [ADC\\_SAR\\_IP\\_CALBISTREG\\_TEST\\_FAIL\\_MASK](#)
- #define [ADC\\_SAR\\_IP\\_CALBISTREG\\_AVG\\_EN\\_MASK](#)
- #define [ADC\\_SAR\\_IP\\_CALBISTREG\\_AVG\\_EN\(x\)](#)
- #define [ADC\\_SAR\\_IP\\_CALBISTREG\\_NR\\_SMPL\\_MASK](#)
- #define [ADC\\_SAR\\_IP\\_CALBISTREG\\_NR\\_SMPL\(x\)](#)
- #define [ADC\\_SAR\\_IP\\_CALBISTREG\\_C\\_T\\_BUSY\\_MASK](#)
- #define [ADC\\_SAR\\_IP\\_CALBISTREG\\_TSAMP\\_MASK](#)

## 6.2.2 Data Structure Documentation

### 6.2.2.1 struct Adc\_Sar\_Ip\_ChanConfigType

Defines the channel configuration.

This structure is used to configure channels

Implements : Adc\_Sar\_Ip\_ChanConfigType\_Class

Definition at line 326 of file Adc\_Sar\_Ip\_Types.h.

### 6.2.2.2 struct Adc\_Sar\_Ip\_ClockConfigType

Defines the ADC clock configuration.

This structure is used to the ADC\_SAR clock

Implements : Adc\_Sar\_Ip\_ClockConfigType\_Class

Definition at line 347 of file Adc\_Sar\_Ip\_Types.h.

Data Fields

Type	Name	Description
<a href="#">Adc_Sar_Ip_ClockSelType</a>	ClkSelect	Selected clock
boolean	HighSpeedConvEn	
uint8	SampleTimeArr[ADC_SAR_IP_NUM_GROUPS]	Sample time for each channel group
uint8	PowerDownDelay	Delay before entering Power Down
boolean	AvgEn	
<a href="#">Adc_Sar_Ip_AvgSelectType</a>	AvgSel	

### 6.2.2.3 struct Adc\_Sar\_Ip\_WdgThresholdType

Defines the upper and lower thresholds for analog watchdog.

This structure is used to configure the analog watchdog threshold registers.

Implements : Adc\_Sar\_Ip\_WdgThresholdType\_Class

Definition at line 374 of file Adc\_Sar\_Ip\_Types.h.

### Data Fields

Type	Name	Description
uint8	WdgIndex	Watchdog threshold register index
uint16	LowThreshold	Lower threshold
uint16	HighThreshold	Upper threshold
boolean	LowThresholdIntEn	Enable interrupt when lower threshold exceeded
boolean	HighThresholdIntEn	Enable interrupt when upper threshold exceeded

#### 6.2.2.4 struct Adc\_Sar\_Ip\_CharResultType

Defines the data regarding a conversion, beyond the conversion data.

This structure is used to return information about conversions beyond just conversion data

Implements : Adc\_Sar\_Ip\_CharResultType\_Class

Definition at line 391 of file Adc\_Sar\_Ip\_Types.h.

### Data Fields

Type	Name	Description
uint8	AdcChnIdx	ADC Channel Index
boolean	ValidFlag	Data Valid Flag
boolean	OverWrittenFlag	Data Overwritten Flag
uint16	ConvData	Conversion Data

#### 6.2.2.5 struct Adc\_Sar\_Ip\_ChansIdxMaskType

Defines configuration of channels in a chain.

This structure is used to configure channels in chain.

Implements : Adc\_Sar\_Ip\_ChansIdxMaskType\_Class

Definition at line 405 of file Adc\_Sar\_Ip\_Types.h.

### Data Fields

Type	Name	Description
uint32	ChanMaskArr[ADC_SAR_IP_NUM_GROUP_CHAN]	Bit-mask used to configure channels in chain

### 6.2.2.6 struct Adc\_Sar\_Ip\_SelfTestThresholdType

Defines configuration of self-test threshold values.

This structure is used to configure self-test threshold values.

Implements : Adc\_Sar\_Ip\_SelfTestThresholdType\_Class

Definition at line 418 of file Adc\_Sar\_Ip\_Types.h.

### 6.2.2.7 struct Adc\_Sar\_Ip\_ConfigType

Defines the module configuration.

This structure is used to configure the ADC\_SAR module

Implements : Adc\_Sar\_Ip\_ConfigType\_Class

Definition at line 452 of file Adc\_Sar\_Ip\_Types.h.

#### Data Fields

- [Adc\\_Sar\\_Ip\\_ConvModeType](#) ConvMode
- [Adc\\_Sar\\_Ip\\_Resolution](#) AdcResolution
- boolean [BypassResolution](#)
- [Adc\\_Sar\\_Ip\\_ClockSelType](#) ClkSelect
- [Adc\\_Sar\\_Ip\\_ClockSelType](#) CalibrationClkSelect
- [Adc\\_Sar\\_Ip\\_ExtTriggerEdgeType](#) InjectedEdge
- [Adc\\_Sar\\_Ip\\_ExtTriggerEdgeType](#) ExtTrigger
- boolean [NormalExtTrgEn](#)
- boolean [NormalAuxExtTrgEn](#)
- uint8 [SampleTimeArr](#) [ADC\_SAR\_IP\_NUM\_GROUP\_CHAN]
- [Adc\\_Sar\\_Ip\\_PresamplingSourceType](#) [PresamplingSourceArr](#) [ADC\_SAR\_IP\_NUM\_GROUP\_CHAN]
- boolean [AutoClockOff](#)
- boolean [OverwriteEnable](#)
- [Adc\\_Sar\\_Ip\\_DataAlignedType](#) DataAlign
- uint16 [DecodeDelay](#)
- uint8 [PowerDownDelay](#)
- const [Adc\\_Sar\\_Ip\\_SelfTestThresholdType](#) \* [SelfTestThresholdConfig](#)
- uint32 [ChanMaskNormal](#) [ADC\_SAR\_IP\_NUM\_GROUP\_CHAN]
- uint32 [ChanMaskInjected](#) [ADC\_SAR\_IP\_NUM\_GROUP\_CHAN]

#### 6.2.2.7.1 Field Documentation

### 6.2.2.7.1.1 ConvMode [Adc\\_Sar\\_Ip\\_ConvModeType](#) ConvMode

Conversion Mode (One-shot or Scan)

Definition at line 454 of file [Adc\\_Sar\\_Ip\\_Types.h](#).

### 6.2.2.7.1.2 AdcResolution [Adc\\_Sar\\_Ip\\_Resolution](#) AdcResolution

Adc resolution

Definition at line 456 of file [Adc\\_Sar\\_Ip\\_Types.h](#).

### 6.2.2.7.1.3 BypassResolution [boolean](#) BypassResolution

Bypass Adc resolution processing for the result

Definition at line 457 of file [Adc\\_Sar\\_Ip\\_Types.h](#).

### 6.2.2.7.1.4 ClkSelect [Adc\\_Sar\\_Ip\\_ClockSelType](#) ClkSelect

Clock input

Definition at line 459 of file [Adc\\_Sar\\_Ip\\_Types.h](#).

### 6.2.2.7.1.5 CalibrationClkSelect [Adc\\_Sar\\_Ip\\_ClockSelType](#) CalibrationClkSelect

Clock input for calibration

Definition at line 460 of file [Adc\\_Sar\\_Ip\\_Types.h](#).

### 6.2.2.7.1.6 InjectedEdge [Adc\\_Sar\\_Ip\\_ExtTriggerEdgeType](#) InjectedEdge

Injected Trigger selection

Definition at line 469 of file [Adc\\_Sar\\_Ip\\_Types.h](#).

**6.2.2.7.1.7 ExtTrigger** `Adc_Sar_Ip_ExtTriggerEdgeType` ExtTrigger

External Trigger selection

Definition at line 472 of file Adc\_Sar\_Ip\_Types.h.

**6.2.2.7.1.8 NormalExtTrgEn** `boolean` NormalExtTrgEn

Enables normal trigger source

Definition at line 473 of file Adc\_Sar\_Ip\_Types.h.

**6.2.2.7.1.9 NormalAuxExtTrgEn** `boolean` NormalAuxExtTrgEn

Enables auxiliary normal trigger source

Definition at line 475 of file Adc\_Sar\_Ip\_Types.h.

**6.2.2.7.1.10 SampleTimeArr** `uint8` SampleTimeArr[ADC\_SAR\_IP\_NUM\_GROUP\_CHAN]

Sample time for each channel group

Definition at line 480 of file Adc\_Sar\_Ip\_Types.h.

**6.2.2.7.1.11 PresamplingSourceArr** `Adc_Sar_Ip_PresamplingSourceType` PresamplingSourceArr[ADC\_SAR\_IP\_NUM\_GROUP\_CHAN]

Presampling sources for each channel group

Definition at line 484 of file Adc\_Sar\_Ip\_Types.h.

**6.2.2.7.1.12 AutoClockOff** `boolean` AutoClockOff

Enable Auto Clock Off

Definition at line 486 of file Adc\_Sar\_Ip\_Types.h.

### 6.2.2.7.1.13 **OverwriteEnable** `boolean OverwriteEnable`

Overwrite new conversion data over old data

Definition at line 487 of file `Adc_Sar_Ip_Types.h`.

### 6.2.2.7.1.14 **DataAlign** `Adc_Sar_Ip_DataAlignedType DataAlign`

Data alignment in conversion result register

Definition at line 488 of file `Adc_Sar_Ip_Types.h`.

### 6.2.2.7.1.15 **DecodeDelay** `uint16 DecodeDelay`

Delay for decoding Input MUX channels

Definition at line 490 of file `Adc_Sar_Ip_Types.h`.

### 6.2.2.7.1.16 **PowerDownDelay** `uint8 PowerDownDelay`

Delay before entering Power Down

Definition at line 492 of file `Adc_Sar_Ip_Types.h`.

### 6.2.2.7.1.17 **SelfTestThresholdConfig** `const Adc_Sar_Ip_SelfTestThresholdType* SelfTestThresholdConfig`

Self test threshold configuration

Definition at line 497 of file `Adc_Sar_Ip_Types.h`.

### 6.2.2.7.1.18 **ChanMaskNormal** `uint32 ChanMaskNormal[ADC_SAR_IP_NUM_GROUP_CHAN]`

Bit-mask used to configure Normal Chain

Definition at line 508 of file `Adc_Sar_Ip_Types.h`.



**6.2.2.7.1.19 ChanMaskInjected** `uint32 ChanMaskInjected[ADC_SAR_IP_NUM_GROUP_CHAN]`

Bit-mask used to configure Injected Chain

Definition at line 509 of file `Adc_Sar_Ip_Types.h`.

**6.2.2.8 struct Adc\_Sar\_Ip\_StateStructType**

Structure used to store runtime info.

This structure is used to store ADC SAR runtime info

Implements : `Adc_Sar_Ip_StateStructType_Class`

Definition at line 547 of file `Adc_Sar_Ip_Types.h`.

**Data Fields**

- boolean `InitStatus`
- `Adc_Sar_Ip_DataAlignedType` `DataAlign`
- `Adc_Sar_Ip_ClockSelType` `CalibrationClkSelect`
- boolean `BypassResolution`
- `uint32` `ChanWdgThresholdIndex` `[ADC_SAR_IP_CDR_COUNT]`

**6.2.2.8.1 Field Documentation****6.2.2.8.1.1 InitStatus** `boolean InitStatus`

Check if the driver was initialized.

Definition at line 549 of file `Adc_Sar_Ip_Types.h`.

**6.2.2.8.1.2 DataAlign** `Adc_Sar_Ip_DataAlignedType DataAlign`

Data alignment in conversion result register

Definition at line 550 of file `Adc_Sar_Ip_Types.h`.

**6.2.2.8.1.3 CalibrationClkSelect** `Adc_Sar_Ip_ClockSelType CalibrationClkSelect`

Clock input for calibration

Definition at line 551 of file `Adc_Sar_Ip_Types.h`.

### 6.2.2.8.1.4 BypassResolution `boolean BypassResolution`

Adc raw data result

Definition at line 553 of file `Adc_Sar_Ip_Types.h`.

### 6.2.2.8.1.5 ChanWdgThresholdIndex `uint32 ChanWdgThresholdIndex[ADC_SAR_IP_CDR_COUNT]`

Channel index to threshold index mapping

Definition at line 556 of file `Adc_Sar_Ip_Types.h`.

## 6.2.3 Macro Definition Documentation

### 6.2.3.1 ADC\_SAR\_IP\_NOTIF\_FLAG\_NORMAL\_ENDCHAIN

```
#define ADC_SAR_IP_NOTIF_FLAG_NORMAL_ENDCHAIN
```

Macros for status and notification flags.

These flags map to internal hardware flags in different registers, but are grouped together for convenience.

Definition at line 113 of file `Adc_Sar_Ip.h`.

### 6.2.3.2 ADC\_SAR\_IP\_CHAN\_NOTIF\_EOC

```
#define ADC_SAR_IP_CHAN_NOTIF_EOC
```

Macros for channel notifications.

These notification flags map to internal hardware flags in different registers, but are grouped together for convenience.

Definition at line 157 of file `Adc_Sar_Ip.h`.

### 6.2.3.3 ADC\_SAR\_IP\_WDG\_HIGH\_FLAG

```
#define ADC_SAR_IP_WDG_HIGH_FLAG
```

Macros for watchdog registers.

These macros help decode and compose bit mask for watchdog functionality

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

## 6.2.4 Types Reference

### 6.2.4.1 Adc\_Sar\_Ip\_WdgNotificationType

```
typedef void Adc_Sar_Ip_WdgNotificationType(const uint16 ChanIdx, const uint8 Flags)
```

Defines the watchdog notification header.

This header is used for out of range watchdog notification callbacks u8Flags will contain which WDG flag was triggered(low and/or high). These are defined by ADC\_SAR\_IP\_WDG\_LOW\_FLAG and ADC\_SAR\_IP\_WDG\_HIGH\_FLAG

Implements : Adc\_Sar\_Ip\_WdgChanNotificationType\_Class

Definition at line 316 of file Adc\_Sar\_Ip\_Types.h.

## 6.2.5 Enum Reference

### 6.2.5.1 Adc\_Sar\_Ip\_StatusType

```
enum Adc_Sar_Ip_StatusType
```

ADC\_SAR status return type.

This enum is used as return type

Implements : Adc\_Sar\_Ip\_StatusType\_Class

Enumerator

ADC_SAR_IP_STATUS_SUCCESS	Function completed successfully
ADC_SAR_IP_STATUS_ERROR	Function didn't complete successfully
ADC_SAR_IP_STATUS_TIMEOUT	Function timed out

Definition at line 101 of file Adc\_Sar\_Ip\_Types.h.

### 6.2.5.2 Adc\_Sar\_Ip\_ConvModeType

```
enum Adc_Sar_Ip_ConvModeType
```

Conversion mode selection (One-shot or Scan)

This enum is used to configure the conversion mode

Implements : Adc\_Sar\_Ip\_ConvModeType\_Class

Enumerator

ADC_SAR_IP_CONV_MODE_ONESHOT	One-shot conversion mode
ADC_SAR_IP_CONV_MODE_SCAN	Scan conversion mode

Definition at line 115 of file Adc\_Sar\_Ip\_Types.h.

### 6.2.5.3 Adc\_Sar\_Ip\_ClockSelType

enum `Adc_Sar_Ip_ClockSelType`

Converter input clock.

This enum is used to configure the converter input clock

Implements : `Adc_Sar_Ip_ClockSelType_Class`

Enumerator

ADC_SAR_IP_CLK_FULL_BUS	Adc module clock
ADC_SAR_IP_CLK_HALF_BUS	Adc module clock/2

Definition at line 128 of file Adc\_Sar\_Ip\_Types.h.

### 6.2.5.4 Adc\_Sar\_Ip\_ExtTriggerEdgeType

enum `Adc_Sar_Ip_ExtTriggerEdgeType`

External Trigger selection.

This enum is used to configure the external trigger

Implements : `Adc_Sar_Ip_ExtTriggerEdgeType_Class`

Enumerator

ADC_SAR_IP_EXT_TRIG_EDGE_DISABLED	Injected trigger disabled
ADC_SAR_IP_EXT_TRIG_EDGE_FALLING	Injected trigger on Falling Edge
ADC_SAR_IP_EXT_TRIG_EDGE_RISING	Injected trigger on Rising Edge

Definition at line 162 of file Adc\_Sar\_Ip\_Types.h.

#### 6.2.5.5 Adc\_Sar\_Ip\_ExtTriggerSourceType

```
enum Adc_Sar_Ip_ExtTriggerSourceType
```

External Trigger Source Enable.

This enum is used to enable the external trigger source to start the conversion.

Implements : Adc\_Sar\_Ip\_ExtTriggerSourceType\_Class

Enumerator

ADC_SAR_IP_NORMAL_EXT_TRIG	Enables normal trigger input
ADC_SAR_IP_AUX_NORMAL_EXT_TRIG	Enables auxiliary normal trigger input
ADC_SAR_IP_ALL_NORMAL_EXT_TRIG	Enables normal and auxiliary trigger inputs
ADC_SAR_IP_INJECTED_EXT_TRIG	Enables injection trigger input

Definition at line 176 of file Adc\_Sar\_Ip\_Types.h.

#### 6.2.5.6 Adc\_Sar\_Ip\_ConvChainType

```
enum Adc_Sar_Ip_ConvChainType
```

Conversion chain selection.

This enum is used to configure type of the conversion

Implements : Adc\_Sar\_Ip\_ConvChainType\_Class

Enumerator

ADC_SAR_IP_CONV_CHAIN_NORMAL	Selects the "Normal" Conversion Chain
ADC_SAR_IP_CONV_CHAIN_INJECTED	Selects the "Injected" Conversion Chain
ADC_SAR_IP_CONV_CHAIN_CTU	Selects the "CTU" Conversion Chain

Definition at line 198 of file Adc\_Sar\_Ip\_Types.h.

### 6.2.5.7 Adc\_Sar\_Ip\_DataAlignedType

enum `Adc_Sar_Ip_DataAlignedType`

Data alignment selection.

This enum is used to configure data alignment

Implements : `Adc_Sar_Ip_DataAlignedType_Class`

Enumerator

<code>ADC_SAR_IP_DATA_ALIGNED_RIGHT</code>	Measured data is right-aligned
<code>ADC_SAR_IP_DATA_ALIGNED_LEFT</code>	Measured data is left-aligned

Definition at line 212 of file `Adc_Sar_Ip_Types.h`.

### 6.2.5.8 Adc\_Sar\_Ip\_ClearSourceType

enum `Adc_Sar_Ip_ClearSourceType`

Clear DMA source.

This enum is used to configure source used to clear a DMA request

Implements : `Adc_Sar_Ip_ClearSourceType_Class`

Enumerator

<code>ADC_SAR_IP_DMA_REQ_CLEAR_ON_ACK</code>	Clear DMA Request on Ack from DMA Controller
<code>ADC_SAR_IP_DMA_REQ_CLEAR_ON_READ</code>	Clear DMA Request on read of Data Registers

Definition at line 225 of file `Adc_Sar_Ip_Types.h`.

### 6.2.5.9 Adc\_Sar\_Ip\_PresamplingSourceType

enum `Adc_Sar_Ip_PresamplingSourceType`

Presampling Voltage selection.

This enum is used to configure the presampling voltage

Implements : `Adc_Sar_Ip_PresamplingSourceType_Class`

Enumerator

ADC_SAR_IP_PRESAMPLE_VREFL	Presampling from VREFL
ADC_SAR_IP_PRESAMPLE_VREFH	Presampling from VREFH

Definition at line 238 of file Adc\_Sar\_Ip\_Types.h.

#### 6.2.5.10 Adc\_Sar\_Ip\_ChanGroupType

enum [Adc\\_Sar\\_Ip\\_ChanGroupType](#)

Channel group selection.

This enum is used to select the group of ADC channels

Implements : Adc\_Sar\_Ip\_ChanGroupType\_Class

Enumerator

ADC_SAR_IP_CHAN_GROUP↵ _0	Channels Group (0-31)
ADC_SAR_IP_CHAN_GROUP↵ _1	Channels Group (32-63)

Definition at line 255 of file Adc\_Sar\_Ip\_Types.h.

#### 6.2.5.11 Adc\_Sar\_Ip\_AvgSelectType

enum [Adc\\_Sar\\_Ip\\_AvgSelectType](#)

Averaging selection.

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

Implements : Adc\_Sar\_Ip\_AvgSelectType\_Class

Enumerator

ADC_SAR_IP_AVG_4_CONV	4 conversions per conversion data
ADC_SAR_IP_AVG_8_CONV	8 conversions per conversion data
ADC_SAR_IP_AVG_16_CONV	16 conversions per conversion data
ADC_SAR_IP_AVG_32_CONV	32 conversions per conversion data

Definition at line 272 of file `Adc_Sar_Ip_Types.h`.

### 6.2.5.12 Adc\_Sar\_Ip\_Resolution

```
enum Adc_Sar_Ip_Resolution
```

Adc group conversion resolution.

Used for value received by Tressos interface configuration.

Enumerator

ADC_SAR_IP_RESOLUTION_14	14-bit per conversion data
ADC_SAR_IP_RESOLUTION_12	12-bit per conversion data
ADC_SAR_IP_RESOLUTION_10	10-bit per conversion data
ADC_SAR_IP_RESOLUTION_8	8-bit per conversion data

Definition at line 286 of file `Adc_Sar_Ip_Types.h`.

## 6.2.6 Function Reference

### 6.2.6.1 Adc\_Sar\_Ip\_GetDataAddress()

```
uint32 Adc_Sar_Ip_GetDataAddress (
    uint32 u32Instance,
    uint32 u32ChannelIndex )
```

Return the address of the specified data register.

This function returns the address of the specified data register

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u32ChannelIndex</i>	- adc channel of the Hw unit

Returns

status:

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



### 6.2.6.2 Adc\_Sar\_Ip\_Init()

```
Adc_Sar_Ip_StatusType Adc_Sar_Ip_Init (
    const uint32 u32Instance,
    const Adc_Sar_Ip_ConfigType *const pConfig )
```

Initialize ADC\_SAR module.

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

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>pConfig</i>	- configuration struct pointer

Returns

status:

- ADC\_SAR\_IP\_STATUS\_SUCCESS: init successful
- ADC\_SAR\_IP\_STATUS\_TIMEOUT: init step timed out

### 6.2.6.3 Adc\_Sar\_Ip\_Deinit()

```
Adc_Sar_Ip_StatusType Adc_Sar_Ip_Deinit (
    const uint32 u32Instance )
```

Deinitialize ADC\_SAR module.

This function resets the ADC internal registers to default values.

Parameters

in	<i>u32Instance</i>	- ADC instance number
----	--------------------	-----------------------

Returns

status:

- ADC\_SAR\_IP\_STATUS\_SUCCESS: deinit successful
- ADC\_SAR\_IP\_STATUS\_TIMEOUT: deinit step timed out

6.2.6.4    **Adc\_Sar\_Ip\_ChainConfig()**

```
void Adc_Sar_Ip_ChainConfig (
    const uint32 u32Instance,
    const Adc_Sar_Ip_ChansIdxMaskType *const pChansIdxMask,
    const Adc_Sar_Ip_ConvChainType pChainType )
```

Configures the converter chains the given configuration structure.

This function configures the ADC Normal and Injected Chains with the options provided in the structure.

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>pChansIdxMask</i>	- channel configuration structure
in	<i>pChainType</i>	- conversion chain (Normal or Injected)

Returns

void

6.2.6.5    **Adc\_Sar\_Ip\_EnableChannel()**

```
void Adc_Sar_Ip_EnableChannel (
    const uint32 u32Instance,
    const Adc_Sar_Ip_ConvChainType pChainType,
    const uint32 u32ChnIdx )
```

Enable a channel.

This function enables a channel in a specified conversion chain

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>pChainType</i>	- conversion chain (Normal or Injected)
in	<i>u32ChnIdx</i>	- channel 0

Returns

void

### 6.2.6.6 Adc\_Sar\_Ip\_DisableChannel()

```
void Adc_Sar_Ip_DisableChannel (
    const uint32 u32Instance,
    const Adc_Sar_Ip_ConvChainType pChainType,
    const uint32 u32ChnIdx )
```

Disable a channel.

This function disables a channel in a specified conversion chain

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>pChainType</i>	- conversion chain (Normal or Injected)
in	<i>u32ChnIdx</i>	- channel 0

Returns

void

### 6.2.6.7 Adc\_Sar\_Ip\_SetResolution()

```
void Adc_Sar_Ip_SetResolution (
    const uint32 u32Instance,
    const Adc_Sar_Ip_Resolution eResolution )
```

Set conversion resolution.

This function sets the conversion resolution (number of bits per conversion data)

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>eResolution</i>	- conversion resolution

Returns

void

### 6.2.6.8 Adc\_Sar\_Ip\_StartConversion()

```
void Adc_Sar_Ip_StartConversion (
    const uint32 u32Instance,
    const Adc_Sar_Ip_ConvChainType pChainType )
```

Start conversion.

This function starts a conversion channel (Normal or Injected)

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>pChainType</i>	- conversion chain (Normal or Injected)

Returns

void

### 6.2.6.9 Adc\_Sar\_Ip\_GetStatusFlags()

```
uint32 Adc_Sar_Ip_GetStatusFlags (
    const uint32 u32Instance )
```

Get the status flags.

This function returns the status flags of the ADC.

Parameters

in	<i>u32Instance</i>	- ADC instance number
----	--------------------	-----------------------

Returns

the status flag bit-mask

### 6.2.6.10 Adc\_Sar\_Ip\_ClearStatusFlags()

```
void Adc_Sar_Ip_ClearStatusFlags (
    const uint32 u32Instance,
    const uint32 u32Mask )
```

Clear the status flags.

This function clears the status flags of the ADC.

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u32Mask</i>	- bit-mask of flags to clear

Returns

void

#### 6.2.6.11 Adc\_Sar\_Ip\_SelfTest()

```
Adc_Sar_Ip_StatusType Adc_Sar_Ip_SelfTest (
    const uint32 u32Instance )
```

Self test.

This function executes a self test on the ADC instance.

Parameters

in	<i>u32Instance</i>	- ADC instance number
----	--------------------	-----------------------

Returns

status:

- ADC\_SAR\_IP\_STATUS\_SUCCESS: self testing successful
- ADC\_SAR\_IP\_STATUS\_TIMEOUT: self testing step timed out

#### 6.2.6.12 Adc\_Sar\_Ip\_GetConvDataToArray()

```
uint32 Adc_Sar_Ip_GetConvDataToArray (
    const uint32 u32Instance,
    const Adc_Sar_Ip_ConvChainType pChainType,
    const uint32 u32Length,
    uint16 *const pResults )
```

Get conversion results for a conversion chain.

This function gets the conversion results for the selected Conversion Chain.

### Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>pChainType</i>	- conversion chain (Normal, Injected or CTU)
in	<i>u32Length</i>	- the length of the buffer
out	<i>pResults</i>	- the output buffer

### Returns

the number of values written in the buffer (max length)

#### 6.2.6.13 Adc\_Sar\_Ip\_GetConvResultsToArray()

```
uint32 Adc_Sar_Ip_GetConvResultsToArray (
    const uint32 u32Instance,
    const Adc_Sar_Ip_ConvChainType pChainType,
    const uint32 u32Length,
    Adc_Sar_Ip_ChanResultType *const pResults )
```

Get conversion results for a conversion chain with extended information.

This function gets the conversion results for the selected Conversion Chain, with extended information about each conversion result (channel index, valid an overwritten properties and conversion data). This function should be used in case of configurations with overlapping channel lists in different chains, resulting in overwrite of conversion data when a higher priority chain is executed before all data was read.

### Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>pChainType</i>	- conversion chain (Normal, Injected or CTU)
in	<i>u32Length</i>	- the length of the buffer
out	<i>pResults</i>	- the output buffer

### Returns

the number of values written in the buffer (max length)

#### 6.2.6.14 Adc\_Sar\_Ip\_GetConvData()

```
uint16 Adc_Sar_Ip_GetConvData (
    const uint32 u32Instance,
    const uint32 u32ChnIdx )
```

Return the result of the conversion.

This function returns the result of the conversion for a single channel

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u32ChnIdx</i>	- channel 0

Returns

conversion data

#### 6.2.6.15 Adc\_Sar\_Ip\_GetConvResult()

```
void Adc_Sar_Ip_GetConvResult (
    const uint32 u32Instance,
    const uint32 u32ChnIdx,
    const Adc_Sar_Ip_ConvChainType pChainType,
    Adc_Sar_Ip_ChanResultType *const pResult )
```

Return the result and the status of the conversion.

This function returns the result and the status of the conversion for a single channel

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u32ChnIdx</i>	- channel 0
in	<i>pChainType</i>	- conversion chain (Normal, Injected or CTU)
out	<i>pResult</i>	- pointer to the buffer where the result will be written

Returns

void

#### 6.2.6.16 Adc\_Sar\_Ip\_DoCalibration()

```
Adc_Sar_Ip_StatusType Adc_Sar_Ip_DoCalibration (
    const uint32 u32Instance )
```

Perform Calibration of the ADC.

## Module Documentation

This function performs a calibration of the ADC. The maximum input clock frequency for the ADC is 80 MHz, checked with assertions if DEV\_ASSERT is enabled. After calibration, the ADC is left in Powerup state (PWDN bit is clear).



## Parameters

in	<i>u32Instance</i>	- ADC instance number
----	--------------------	-----------------------

## Returns

the calibration result

- ADC\_SAR\_IP\_STATUS\_SUCCESS: calibration successful
- ADC\_SAR\_IP\_STATUS\_TIMEOUT: calibration step timed out
- ADC\_SAR\_IP\_STATUS\_ERROR: calibration failed

**6.2.6.17   Adc\_Sar\_Ip\_Powerup()**

```
Adc_Sar_Ip_StatusType Adc_Sar_Ip_Powerup (
    const uint32 u32Instance )
```

Power up the ADC.

This function enables the ADC (disables the Power Down feature).

## Parameters

in	<i>u32Instance</i>	- ADC instance number
----	--------------------	-----------------------

## Returns

status:

- ADC\_SAR\_IP\_STATUS\_SUCCESS: power up successful
- ADC\_SAR\_IP\_STATUS\_TIMEOUT: power up failed

**6.2.6.18   Adc\_Sar\_Ip\_Powerdown()**

```
Adc_Sar_Ip_StatusType Adc_Sar_Ip_Powerdown (
    const uint32 u32Instance )
```

Power down the ADC.

This function disables the ADC (enables the Power Down feature).

### Parameters

in	<i>u32Instance</i>	- ADC instance number
----	--------------------	-----------------------

### Returns

status:

- ADC\_SAR\_IP\_STATUS\_SUCCESS: power down successful
- ADC\_SAR\_IP\_STATUS\_TIMEOUT: power down failed

#### 6.2.6.19 Adc\_Sar\_Ip\_EnableNotifications()

```
void Adc_Sar_Ip_EnableNotifications (
    const uint32 u32Instance,
    const uint32 u32NotificationMask )
```

Enable ADC interrupts.

This function enables ADC interrupts.

### Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u32NotificationMask</i>	- mask of interrupts to enable (of status flags)

### Returns

void

#### 6.2.6.20 Adc\_Sar\_Ip\_DisableNotifications()

```
void Adc_Sar_Ip_DisableNotifications (
    const uint32 u32Instance,
    const uint32 u32NotificationMask )
```

Disable ADC interrupts.

This function disables ADC interrupts.

## Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u32NotificationMask</i>	- mask of interrupts to disable (of status flags)

## Returns

void

**6.2.6.21   Adc\_Sar\_Ip\_EnableChannelNotifications()**

```
void Adc_Sar_Ip_EnableChannelNotifications (
    const uint32 u32Instance,
    const uint32 u32ChnIdx,
    const uint32 u32Mask )
```

Enable ADC interrupt for a channel.

This function enables interrupt generation on End of Conversion and watchdog events for a single channel. The mask parameter can be set using the defines provided in the driver header file that have the pattern ADC\_SAR\_IP\_CHAN\_NOTIF\_... e.g. ADC\_SAR\_IP\_CHAN\_NOTIF\_EOC.

## Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u32ChnIdx</i>	- channel 0
in	<i>u32Mask</i>	- mask selecting targeted events

## Returns

void

**6.2.6.22   Adc\_Sar\_Ip\_DisableChannelNotifications()**

```
void Adc_Sar_Ip_DisableChannelNotifications (
    const uint32 u32Instance,
    const uint32 u32ChnIdx,
    const uint32 u32Mask )
```

Disable ADC interrupt for a channel.

This function disables interrupt generation on End of Conversion and watchdog events for a single channel. The mask parameter can be set using the defines provided in the driver header file that have the pattern ADC\_SAR\_IP\_CHAN\_NOTIF\_... e.g. ADC\_SAR\_IP\_CHAN\_NOTIF\_EOC.

### Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u32ChnIdx</i>	- channel 0
in	<i>u32Mask</i>	- mask selecting targeted events

### Returns

void

#### 6.2.6.23 Adc\_Sar\_Ip\_SetClockMode()

```
Adc_Sar_Ip_StatusType Adc_Sar_Ip_SetClockMode (  
    const uint32 u32Instance,  
    const Adc_Sar_Ip_ClockConfigType *const pConfig )
```

Set the ADC clocks.

This function initializes the ADC clock configuration.

### Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>pConfig</i>	- the clock configuration

### Returns

status:

- ADC\_SAR\_IP\_STATUS\_SUCCESS: set successful
- ADC\_SAR\_IP\_STATUS\_TIMEOUT: power up or down sequence timed out

#### 6.2.6.24 Adc\_Sar\_Ip\_SetSampleTimes()

```
void Adc_Sar_Ip_SetSampleTimes (  
    const uint32 u32Instance,  
    const uint8 *const aSampleTimes )
```

Set the sample times.

This function sets the sample times for each channel group. Note: aSampleTimes must contain the sample times in order of the channel groups, e.g. sample time for the channel group 1 must be on the first position.

## Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>aSampleTimes</i>	- array with size ADC_SAR_IP_NUM_GROUP_CHAN containing sample times for each channel group

## Returns

void

**6.2.6.25 Adc\_Sar\_Ip\_SetAveraging()**

```
void Adc_Sar_Ip_SetAveraging (
    const uint32 u32Instance,
    const boolean bAvgEn,
    const Adc_Sar_Ip_AvgSelectType eAvgSel )
```

Configure averaging.

This function enables averaging and selects the number of conversions to average. The mask parameter should be set using the Adc\_Sar\_Ip\_AvgSelectType enum elements that have the pattern ADC\_SAR\_IP\_AVG\_... e.g. ADC\_SAR\_IP\_AVG\_4\_CONV.

## Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>bAvgEn</i>	- enable or disable averaging
in	<i>eAvgSel</i>	- selects number of conversions to average

## Returns

void

**6.2.6.26 Adc\_Sar\_Ip\_AbortConversion()**

```
void Adc_Sar_Ip_AbortConversion (
    const uint32 u32Instance )
```

Abort ongoing conversion.

This function aborts an ongoing conversion.

### Parameters

in	<i>u32Instance</i>	- ADC instance number
----	--------------------	-----------------------

### Returns

void

#### 6.2.6.27 `Adc_Sar_Ip_AbortChain()`

```
Adc_Sar_Ip_StatusType Adc_Sar_Ip_AbortChain (
    const uint32 u32Instance,
    const boolean bBlocking,
    const boolean bAllowRestart )
```

Abort ongoing chain conversion.

This function aborts an ongoing chain of conversions. If scan(continuous) mode is selected, then bAllowRestart selects whether the abort command restarts the chain or stops it. For one shot conversion, this should be FALSE.

### Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>bBlocking</i>	- wait and check that the ongoing conversion has stopped
in	<i>bAllowRestart</i>	- restart continuous conversion instead of aborting

### Returns

status:

- `ADC_SAR_IP_STATUS_SUCCESS`: abort successful
- `ADC_SAR_IP_STATUS_TIMEOUT`: ongoing conversion could not be stopped

void

#### 6.2.6.28 `Adc_Sar_Ip_SetPresamplingSource()`

```
void Adc_Sar_Ip_SetPresamplingSource (
    const uint32 u32Instance,
    const Adc_Sar_Ip_ChanGroupType pChanGroup,
    const Adc_Sar_Ip_PresamplingSourceType pPresampleSource )
```

Set the Presampling Source for the channel group.

This function configures the Presampling Source for a channel group.

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>pChanGroup</i>	- the channel group
in	<i>pPresampleSource</i>	- the presampling source

Returns

void

#### 6.2.6.29 Adc\_Sar\_Ip\_EnableChannelPresampling()

```
void Adc_Sar_Ip_EnableChannelPresampling (
    const uint32 u32Instance,
    const uint32 u32ChnIdx )
```

Enable Presampling on one channel.

This function enables the Presampling on one channel of the ADC.

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u32ChnIdx</i>	- channel 0

Returns

void

#### 6.2.6.30 Adc\_Sar\_Ip\_DisableChannelPresampling()

```
void Adc_Sar_Ip_DisableChannelPresampling (
    const uint32 u32Instance,
    const uint32 u32ChnIdx )
```

Disable Presampling on one channel.

This function disables the Presampling on one channel of the ADC.

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u32ChnIdx</i>	- channel 0

Returns

void

### 6.2.6.31 Adc\_Sar\_Ip\_EnablePresampleConversion()

```
void Adc_Sar_Ip_EnablePresampleConversion (
    const uint32 u32Instance )
```

Enable Conversion Presampled Data.

This function enables bypass of the Sampling Phase, resulting in a conversion of the presampled data. This is available only for channels that have presampling enabled.

Parameters

in	<i>u32Instance</i>	- ADC instance number
----	--------------------	-----------------------

Returns

void

### 6.2.6.32 Adc\_Sar\_Ip\_DisablePresampleConversion()

```
void Adc_Sar_Ip_DisablePresampleConversion (
    const uint32 u32Instance )
```

Disable Conversion of Presampled Data.

This function disables Sampling Phase bypass.

Parameters

in	<i>u32Instance</i>	- ADC instance number
----	--------------------	-----------------------

Returns

void



**6.2.6.33    Adc\_Sar\_Ip\_EnableDma()**

```
void Adc_Sar_Ip_EnableDma (
    const uint32 u32Instance )
```

Enable DMA Requests.

This function enables requests to DMA from ADC

Parameters

in	<i>u32Instance</i>	- ADC instance number
----	--------------------	-----------------------

Returns

void

**6.2.6.34    Adc\_Sar\_Ip\_DisableDma()**

```
void Adc_Sar_Ip_DisableDma (
    const uint32 u32Instance )
```

Disable DMA Requests.

This function disables requests to DMA from ADC

Parameters

in	<i>u32Instance</i>	- ADC instance number
----	--------------------	-----------------------

Returns

void

**6.2.6.35    Adc\_Sar\_Ip\_EnableChannelDma()**

```
void Adc_Sar_Ip_EnableChannelDma (
    const uint32 u32Instance,
    const uint32 u32ChnIdx )
```

Enable DMA on one channel.

This function enables DMA requests triggered by End of Conversion event from a selected channel.

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u32ChnIdx</i>	- channel 0

Returns

void

### 6.2.6.36 `Adc_Sar_Ip_DisableChannelDma()`

```
void Adc_Sar_Ip_DisableChannelDma (  
    const uint32 u32Instance,  
    const uint32 u32ChnIdx )
```

Disable DMA on one channel.

This function disables DMA requests triggered by End of Conversion event from a selected channel.

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u32ChnIdx</i>	- channel 0

Returns

void

### 6.2.6.37 `Adc_Sar_Ip_DisableChannelDmaAll()`

```
void Adc_Sar_Ip_DisableChannelDmaAll (  
    const uint32 u32Instance )
```

Disable DMA on all channels.

This function disables DMA requests triggered by End of Conversion event from all channels.

Parameters

in	<i>u32Instance</i>	- ADC instance number
----	--------------------	-----------------------

Returns

void

#### 6.2.6.38 Adc\_Sar\_Ip\_SetDmaClearSource()

```
void Adc_Sar_Ip_SetDmaClearSource (
    const uint32 u32Instance,
    const Adc_Sar_Ip_ClearSourceType pDmaClear )
```

Set DMA Request Clear Source.

This function selects the DMA Request Flag Clear Source.

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>pDmaClear</i>	- the clear source for DMA Requests (Ack from DMA Controller or read of data registers)

Returns

void

#### 6.2.6.39 Adc\_Sar\_Ip\_SetWdgThreshold()

```
void Adc_Sar_Ip_SetWdgThreshold (
    const uint32 u32Instance,
    const uint8 u8RegisterIdx,
    const Adc_Sar_Ip_WdgThresholdType *const pThresholdValues )
```

Configure watchdog threshold register.

This function configures the high/low thresholds for a certain register.

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>u8RegisterIdx</i>	- the index of the register
in	<i>pThresholdValues</i>	- the threshold values

Returns

void

6.2.6.40    **Adc\_Sar\_Ip\_SetConversionMode()**

```
void Adc_Sar_Ip_SetConversionMode (
    const uint32 u32Instance,
    const Adc_Sar_Ip_ConvModeType eConvMode )
```

Configure conversion mode.

This function configures the used conversion mode. Note: The selected ADC instance must be in the IDLE state.

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>eConvMode</i>	- selected conversion mode

Returns

void

6.2.6.41    **Adc\_Sar\_Ip\_SetExternalTrigger()**

```
void Adc_Sar_Ip_SetExternalTrigger (
    const uint32 u32Instance,
    const Adc_Sar_Ip_ExtTriggerEdgeType eTriggerEdge,
    const Adc_Sar_Ip_ExtTriggerSourceType eTrggerSrc )
```

Configure external trigger.

This function configures the external trigger.

Parameters

in	<i>u32Instance</i>	- ADC instance number
in	<i>eTriggerEdge</i>	- selected external trigger type
in	<i>eTrggerSrc</i>	- selects normal, auxiliary normal or injected trigger

Returns

void

## 6.3 Bctu IPL

### 6.3.1 Detailed Description

Bctu HW module.

Bctu IP layer module.

### Data Structures

- struct [Bctu\\_Ip\\_AdcNotificationType](#)  
*Structure for configuring Adc notification. [More...](#)*
- struct [Bctu\\_Ip\\_ListItemConfigType](#)  
*Structure for storing channel information. [More...](#)*
- struct [Bctu\\_Ip\\_ListItemType](#)  
*Structure for storing channel information at runtime. [More...](#)*
- struct [Bctu\\_Ip\\_FifoConfigType](#)  
*Structure for configuring BCTU FIFO. [More...](#)*
- struct [Bctu\\_Ip\\_FifoNotificationType](#)  
*Store BCTU FIFO notifications. [More...](#)*
- struct [Bctu\\_Ip\\_TrigConfigType](#)  
*Structure for configuring a trigger configuration register. [More...](#)*
- struct [Bctu\\_Ip\\_ResultType](#)  
*Structure for storing full information of the conversion result. [More...](#)*
- struct [Bctu\\_Ip\\_FifoResultType](#)  
*Structure for storing full information of the conversion result stored in the FIFO. [More...](#)*
- struct [Bctu\\_Ip\\_ConfigType](#)  
*Structure for configuring global BCTU functionalities. [More...](#)*
- struct [Bctu\\_Ip\\_StateStructType](#)  
*Structure used to store runtime info. [More...](#)*

### Macros

- `#define BCTU_IP_NOTIF_FIFO1`  
*Macros for notification flags.*
- `#define BCTU_IP_STATUS_TRG`  
*Macros for status flags.*

### Enum Reference

- enum [Bctu\\_Ip\\_StatusType](#)  
*BCTU status return type.*
- enum [Bctu\\_Ip\\_TrigDataDestType](#)  
*Enumeration for selecting data destination.*
- enum [Bctu\\_Ip\\_TrigType](#)  
*Enumeration for selecting trigger type.*

## Function Reference

- void [Bctu\\_Ip\\_Init](#) (const uint32 u32Instance, const [Bctu\\_Ip\\_ConfigType](#) \*const pConfig)  
*Configure global functionalities of a BCTU instance.*
- [Bctu\\_Ip\\_StatusType](#) [Bctu\\_Ip\\_Deinit](#) (const uint32 u32Instance)  
*Reset the BCTU registers to default values.*
- void [Bctu\\_Ip\\_SetGlobalTriggerEn](#) (const uint32 u32Instance, const boolean bState)  
*Set the state of the Global Trigger Enable flag.*
- void [Bctu\\_Ip\\_SetLowPowerMode](#) (const uint32 u32Instance, const boolean bState)  
*Set the BCTU operating mode: normal or low power.*
- uint32 [Bctu\\_Ip\\_GetStatusFlags](#) (const uint32 u32Instance)  
*Get the value of all BCTU status flags.*
- void [Bctu\\_Ip\\_ClearStatusFlags](#) (const uint32 u32Instance, const uint32 u32Mask)  
*Clear selected BCTU status flags.*
- [Bctu\\_Ip\\_StatusType](#) [Bctu\\_Ip\\_ConfigTrigger](#) (const uint32 u32Instance, const [Bctu\\_Ip\\_TrigConfigType](#) \*const pTrigConfig)  
*Configures a trigger configuration register for an individual (non-list) conversion.*
- [Bctu\\_Ip\\_StatusType](#) [Bctu\\_Ip\\_SetTriggerChnListAddr](#) (const uint32 u32Instance, const uint8 u8TrigIdx, const boolean bList, const uint8 u8ChnOrListAddr)  
*Configures trigger resolution and either the channel or first address of the command list associated with the trigger.*
- void [Bctu\\_Ip\\_EnableHwTrigger](#) (const uint32 u32Instance, const uint8 u8TrigIdx)  
*Clears any eventual pending hw input signals already occurred on the selected trigger and enables new occurrences of the hw input signals to start new conversions.*
- void [Bctu\\_Ip\\_DisableHwTrigger](#) (const uint32 u32Instance, const uint8 u8TrigIdx)  
*Prevent the selected BCTU trigger to start new conversions on the occurrence of an external input signal or new conversions in an on-going conversion loop (loop flag enabled).*
- void [Bctu\\_Ip\\_StopLoopConversions](#) (const uint32 u32Instance, const uint8 u8TrigIdx)  
*Stop a series of loop conversions initiated by the selected BCTU trigger.*
- void [Bctu\\_Ip\\_ConfigChanList](#) (const uint32 u32Instance, const uint8 u8StartListElemIdx, const [Bctu\\_Ip\\_ListItemType](#) \*const pConvListItems, const uint8 u8NumItems)  
*Set an array of ADC channel structures in the conversion list.*
- void [Bctu\\_Ip\\_ConfigChanListSimple](#) (const uint32 u32Instance, const uint8 u8StartListElemIdx, const uint8 \*const pConvListItems, const uint8 u8NumItems)  
*Set an array of ADC channel numbers in the conversion list.*
- void [Bctu\\_Ip\\_SwTriggerConversion](#) (const uint32 u32Instance, const uint8 u8TrigIdx)  
*Triggers a conversion or list of conversions associated with a trigger index, if write protection is disabled.*
- uint16 [Bctu\\_Ip\\_GetConvData](#) (const uint32 u32Instance, const uint8 u8AdcIdx)  
*Return the current conversion result data corresponding to an ADC instance.*
- void [Bctu\\_Ip\\_GetConvResult](#) (const uint32 u32Instance, const uint8 u8AdcIdx, [Bctu\\_Ip\\_ResultType](#) \*const pResult)  
*Return the full conversion result information corresponding to an ADC instance.*
- uint16 [Bctu\\_Ip\\_GetFifoData](#) (const uint32 u32Instance, const uint8 u8FifoIdx)  
*Return the conversion result data stored in the index in a FIFO.*
- void [Bctu\\_Ip\\_GetFifoResult](#) (const uint32 u32Instance, const uint8 u8FifoIdx, [Bctu\\_Ip\\_FifoResultType](#) \*const pResult)  
*Return the full conversion result information stored in a FIFO.*
- void [Bctu\\_Ip\\_EnableNotifications](#) (const uint32 u32Instance, const uint32 u32NotificationMask)  
*Enables notifications.*

- void [Bctu\\_Ip\\_DisableNotifications](#) (const uint32 u32Instance, const uint32 u32NotificationMask)  
*Disables notifications.*
- void [Bctu\\_Ip\\_SetFifoWatermark](#) (const uint32 u32Instance, const uint8 u8FifoIdx, const uint8 u8Watermark)  
*Sets watermark value for the selected FIFO.*
- uint8 [Bctu\\_Ip\\_GetFifoCount](#) (const uint32 u32Instance, const uint8 u8FifoIdx)  
*Get the counter value of the given FIFO.*

### 6.3.2 Data Structure Documentation

#### 6.3.2.1 struct Bctu\_Ip\_AdcNotificationType

Structure for configuring Adc notification.

This structure is used for configuring Adc notification

Implements : Bctu\_Ip\_AdcNotificationType\_Class

Definition at line 142 of file Bctu\_Ip\_Types.h.

#### 6.3.2.2 struct Bctu\_Ip\_ListItemConfigType

Structure for storing channel information.

This structure is used for storing channel information.

Implements : Bctu\_Ip\_ListItemConfigType\_Class

Definition at line 155 of file Bctu\_Ip\_Types.h.

#### 6.3.2.3 struct Bctu\_Ip\_ListItemType

Structure for storing channel information at runtime.

This structure is used for storing channel information at runtime.

Implements : Bctu\_Ip\_ListItemType\_Class

Definition at line 168 of file Bctu\_Ip\_Types.h.

#### 6.3.2.4 struct Bctu\_Ip\_FifoConfigType

Structure for configuring BCTU FIFO.

This structure is used for configuring the BCTU FIFO.

Implements : Bctu\_Ip\_FifoConfigType\_Class

Definition at line 180 of file Bctu\_Ip\_Types.h.



### 6.3.2.5 struct Bctu\_Ip\_FifoNotificationType

Store BCTU FIFO notifications.

This structure is used for storing BCTU FIFO notifications

Implements : Bctu\_Ip\_FifoNotificationType\_Class

Definition at line 203 of file Bctu\_Ip\_Types.h.

### 6.3.2.6 struct Bctu\_Ip\_TrigConfigType

Structure for configuring a trigger configuration register.

This structure is used for configuring a trigger configuration register (TRGCFG)

Implements : Bctu\_Ip\_TrigConfigType\_Class

Definition at line 216 of file Bctu\_Ip\_Types.h.

Data Fields

Type	Name	Description
uint8	TrigIndex	
boolean	LoopEn	Select if a trigger (hw or sw) starts a loop of conversions.
<a href="#">Bctu_Ip_TrigDataDestType</a>	DataDest	
boolean	HwTriggersEn	Enable/disable initiation of ADC conversions by hw input trigger: true - enabled; false - disabled.
<a href="#">Bctu_Ip_TrigType</a>	TrigType	
uint8	AdcTargetMask	Bitmask to select target ADC instance(s): (1 << n) corresponds to n-th ADC instance.
uint8	AdcChanOrListStart	

### 6.3.2.7 struct Bctu\_Ip\_ResultType

Structure for storing full information of the conversion result.

This structure is used for storing full information of the conversion result

Implements : Bctu\_Ip\_ResultType\_Class

Definition at line 233 of file Bctu\_Ip\_Types.h.

### Data Fields

Type	Name	Description
uint8	TriggerIdx	Index of the trigger which started the conversion.
uint8	ChanIdx	Index of the ADC channel which executed the conversion.
uint16	AdcData	Conversion result data.
boolean	ListFlag	Conversion initiated from a LIST: true / false.
boolean	LastFlag	Last conversion in a LIST: true / false

### 6.3.2.8 struct Bctu\_Ip\_FifoResultType

Structure for storing full information of the conversion result stored in the FIFO.

This structure is used for storing full information of the conversion result stored in the FIFO

Implements : Bctu\_Ip\_FifoResultType\_Class

Definition at line 248 of file Bctu\_Ip\_Types.h.

### Data Fields

Type	Name	Description
uint8	TriggerIdx	Index of the trigger which started the conversion.
uint8	ChanIdx	Index of the ADC channel which executed the conversion.
uint8	AdcNum	Index of the ADC instance which executed the conversion.
uint16	AdcData	Conversion result data.

### 6.3.2.9 struct Bctu\_Ip\_ConfigType

Structure for configuring global BCTU functionalities.

This structure is used for configuring global BCTU functionalities

Implements : Bctu\_Ip\_ConfigType\_Class

Definition at line 262 of file Bctu\_Ip\_Types.h.

### Data Fields

- boolean [LowPowerModeEn](#)
- boolean [GlobalHwTriggersEn](#)
- uint8 [NewDataDmaEnMask](#)
- void(\* [TriggerNotification](#) )(void)

### 6.3.2.9.1 Field Documentation

#### 6.3.2.9.1.1 LowPowerModeEn `boolean LowPowerModeEn`

Low power operating mode: true - low power; false - normal operating mode.

Definition at line 263 of file `Bctu_Ip_Types.h`.

#### 6.3.2.9.1.2 GlobalHwTriggersEn `boolean GlobalHwTriggersEn`

HW triggers globally enabled/disabled: true - enabled; false - all hw triggers disabled (BCTU in freeze mode).

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

#### 6.3.2.9.1.3 NewDataDmaEnMask `uint8 NewDataDmaEnMask`

Bitmask for DMA transfer enable per ADC instance:  $(1 \ll n)$  corresponds to n-th ADC instance.

Definition at line 265 of file `Bctu_Ip_Types.h`.

#### 6.3.2.9.1.4 TriggerNotification `void(* TriggerNotification) (void)`

TRGF

Definition at line 267 of file `Bctu_Ip_Types.h`.

### 6.3.2.10 struct Bctu\_Ip\_StateStructType

Structure used to store runtime info.

This structure is used to store BCTU runtime info

Implements : `Bctu_Ip_StateStructType_Class`

Definition at line 288 of file `Bctu_Ip_Types.h`.

#### Data Fields

- boolean [InitStatus](#)
- void(\* [TriggerNotification](#) )(void)

### 6.3.2.10.1 Field Documentation

#### 6.3.2.10.1.1 **InitStatus** `boolean InitStatus`

Check if the driver was initialized.

Definition at line 289 of file `Bctu_Ip_Types.h`.

#### 6.3.2.10.1.2 **TriggerNotification** `void(* TriggerNotification) (void)`

TRGF

Definition at line 290 of file `Bctu_Ip_Types.h`.

### 6.3.3 Macro Definition Documentation

#### 6.3.3.1 **BCTU\_IP\_NOTIF\_FIFO1**

```
#define BCTU_IP_NOTIF_FIFO1
```

Macros for notification flags.

Define bit masks to check for enabling and disabling notification.

Definition at line 111 of file `Bctu_Ip.h`.

#### 6.3.3.2 **BCTU\_IP\_STATUS\_TRG**

```
#define BCTU_IP_STATUS_TRG
```

Macros for status flags.

These flags map to internal hardware flags in different registers, but are grouped together for convenience.

Definition at line 140 of file `Bctu_Ip.h`.

### 6.3.4 Enum Reference

#### 6.3.4.1 **Bctu\_Ip\_StatusType**

```
enum Bctu_Ip_StatusType
```

BCTU status return type.

This structure is used as return type

Implements : `Bctu_Ip_StatusType_Class`

Enumerator

BCTU_IP_STATUS_SUCCESS	Function completed successfully
BCTU_IP_STATUS_ERROR	Function didn't complete successfully
BCTU_IP_STATUS_TIMEOUT	Function timed out
BCTU_IP_STATUS_BUSY	Functionality busy

Definition at line 101 of file Bctu\_Ip\_Types.h.

#### 6.3.4.2 Bctu\_Ip\_TrigDataDestType

```
enum Bctu_Ip_TrigDataDestType
```

Enumeration for selecting data destination.

This enumeration selects the data destination

Implements : Bctu\_Ip\_TrigDataDestType\_Class

Definition at line 116 of file Bctu\_Ip\_Types.h.

#### 6.3.4.3 Bctu\_Ip\_TrigType

```
enum Bctu_Ip_TrigType
```

Enumeration for selecting trigger type.

This enumeration selects the trigger type

Implements : Bctu\_Ip\_TrigType\_Class

Definition at line 130 of file Bctu\_Ip\_Types.h.

### 6.3.5 Function Reference

#### 6.3.5.1 Bctu\_Ip\_Init()

```
void Bctu_Ip_Init (
    const uint32 u32Instance,
    const Bctu_Ip_ConfigType *const pConfig )
```

Configure global functionalities of a BCTU instance.

This function configures the global functionalities of a BCTU instance.

### Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>pConfig</i>	- pointer to the input configuration structure

### Returns

void

#### 6.3.5.2 Bctu\_Ip\_Deinit()

```
Bctu_Ip_StatusType Bctu_Ip_Deinit (  
    const uint32 u32Instance )
```

Reset the BCTU registers to default values.

This function attempts to reset the BCTU registers to default values, in the given timeout interval. Trigger config registers cannot be modified while there is an ongoing conversion, so the function waits for the conversion(s) to finish and returns an error code if timeout occurs. It globally disables all triggers, leaving the BCTU in configuration mode.

### Parameters

in	<i>u32Instance</i>	- BCTU instance number
----	--------------------	------------------------

### Returns

status:  
- BCTU\_IP\_STATUS\_SUCCESS: reset succeeded  
- BCTU\_IP\_STATUS\_TIMEOUT: reset did not complete because an active conversion did not finish execution within the provided timeout interval

#### 6.3.5.3 Bctu\_Ip\_SetGlobalTriggerEn()

```
void Bctu_Ip_SetGlobalTriggerEn (  
    const uint32 u32Instance,  
    const boolean bState )
```

Set the state of the Global Trigger Enable flag.

This function sets the state of the Global Trigger Enable flag. When triggers are disabled globally, the BCTU is in configuration mode, which allows configuring the trigger config registers freely, without generating any spurious ADC conversion triggers. Enabling the flag will move BCTU out of configuration mode and will enable all trigger inputs which are already individually enabled from the trigger config registers.

## Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>bState</i>	- true: globally enable triggers false: globally disable triggers - enter configuration mode

## Returns

void

**6.3.5.4 Bctu\_Ip\_SetLowPowerMode()**

```
void Bctu_Ip_SetLowPowerMode (
    const uint32 u32Instance,
    const boolean bState )
```

Set the BCTU operating mode: normal or low power.

This function sets the BCTU operating mode: normal or low power. In low power mode all trigger inputs are ignored and any pending triggers remain active.

## Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>bState</i>	- true: enable low power mode false: enable normal mode

## Returns

void

**6.3.5.5 Bctu\_Ip\_GetStatusFlags()**

```
uint32 Bctu_Ip_GetStatusFlags (
    const uint32 u32Instance )
```

Get the value of all BCTU status flags.

This function returns the value of all BCTU status flags.

Parameters

in	<i>u32Instance</i>	- BCTU instance number
----	--------------------	------------------------

Returns

state of the selected flags

6.3.5.6 Bctu\_Ip\_ClearStatusFlags()

```
void Bctu_Ip_ClearStatusFlags (
    const uint32 u32Instance,
    const uint32 u32Mask )
```

Clear selected BCTU status flags.

This function clears selected BCTU status flags. The mask parameter should be set using BCTU\_IP\_STATUS\_... defines.

Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u32Mask</i>	- selected flags mask

Returns

void

6.3.5.7 Bctu\_Ip\_ConfigTrigger()

```
Bctu_Ip_StatusType Bctu_Ip_ConfigTrigger (
    const uint32 u32Instance,
    const Bctu_Ip_TrigConfigType *const pTrigConfig )
```

Configures a trigger configuration register for an individual (non-list) conversion.

This function attempts to configure a trigger configuration register for an individual (non-list) conversion. If there is an on-going conversion started via target trigger, it only disables the trigger (will ignore any new triggers) and returns error code.

If succeeds, implicitly clears any pending previous input trigger event.

Note: when executing a list, the conversion is considered active until the conversion marked as "last" completes.



## Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>pTrigConfig</i>	- BCTU trigger configuration

## Returns

status:

- BCTU\_IP\_STATUS\_SUCCESS: configured successfully
- BCTU\_IP\_STATUS\_BUSY: configuration was not written because there is a triggered active conversion

**6.3.5.8 Bctu\_Ip\_SetTriggerChnListAddr()**

```

Bctu_Ip_StatusType Bctu_Ip_SetTriggerChnListAddr (
    const uint32 u32Instance,
    const uint8 u8TrigIdx,
    const boolean bList,
    const uint8 u8ChnOrListAddr )

```

Configures trigger resolution and either the channel or first address of the command list associated with the trigger.

This function attempts to configure trigger resolution and the register for either the channel or the first address of the command list associated with the trigger. If there is an on-going conversion started via target trigger, it only disables the trigger (will ignore any new triggers) and returns error code.

If succeeds, implicitly clears any pending previous input trigger event.

## Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8TrigIdx</i>	- index of the trigger register
in	<i>bList</i>	- value of trigger resolution register: false for single conversion and true for conversion list
in	<i>u8ChnOrListAddr</i>	- channel index if bList is false or conversion list start address if bList is true

## Returns

status:

- BCTU\_IP\_STATUS\_SUCCESS: configured successfully
- BCTU\_IP\_STATUS\_BUSY: configuration was not written because there is a triggered active conversion

### 6.3.5.9 Bctu\_Ip\_EnableHwTrigger()

```
void Bctu_Ip_EnableHwTrigger (
    const uint32 u32Instance,
    const uint8 u8TrigIdx )
```

Clears any eventual pending hw input signals already occurred on the selected trigger and enables new occurrences of the hw input signals to start new conversions.

This function clears any eventual pending hw input signals already occurred on the selected trigger and enables new occurrences of the hw input signals to start new conversions.

Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8TrigIdx</i>	- index of the trigger register

Returns

void

### 6.3.5.10 Bctu\_Ip\_DisableHwTrigger()

```
void Bctu_Ip_DisableHwTrigger (
    const uint32 u32Instance,
    const uint8 u8TrigIdx )
```

Prevent the selected BCTU trigger to start new conversions on the occurrence of an external input signal or new conversions in an on-going conversion loop (loop flag enabled).

This function prevents the selected BCTU trigger to start new conversions on the occurrence of an hw input signal or new conversions in an on-going conversion loop (loop flag enabled). After calling this function, new conversions may still be started by BCTU software trigger. Note: the function does NOT stop an on-going conversion.

Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8TrigIdx</i>	- index of the trigger register

Returns

void

### 6.3.5.11 Bctu\_Ip\_StopLoopConversions()

```
void Bctu_Ip_StopLoopConversions (
    const uint32 u32Instance,
    const uint8 u8TrigIdx )
```

Stop a series of loop conversions initiated by the selected BCTU trigger.

This function stops a series of loop conversions initiated by the selected BCTU trigger. Note0: the function does NOT stop an on-going conversion. Note1: the function assumes that no HW trigger occurs on the selected trigger during its execution - the reset & restore operation on the LOOP flag is not atomic! Occurrence of a HW trigger may start a new conversion with an inconsistent value for the LOOP flag. For stopping HW triggered loops of conversions consider also using Bctu\_Ip\_DisableHwTrigger.

Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8TrigIdx</i>	- index of the trigger register

Returns

void

### 6.3.5.12 Bctu\_Ip\_ConfigChanList()

```
void Bctu_Ip_ConfigChanList (
    const uint32 u32Instance,
    const uint8 u8StartListElemIdx,
    const Bctu_Ip_ListItemType *const pConvListItems,
    const uint8 u8NumItems )
```

Set an array of ADC channel structures in the conversion list.

This function sets the conversion list using an array of ListItemType structures. Also marks the element corresponding to the last channel in the array, as last element in the list. Implicitly wrap-around of the array onto the list, if (listSize - startElemIdx) < numArrayElems.

Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8StartListElemIdx</i>	- index in the list corresponding to the first element in the array
in	<i>pConvListItems</i>	- pointer to the first element in the ADC channel structure input array
in	<i>u8NumItems</i>	- number of elements in the ADC channel numbers input array (required to be >0)

Returns

void

### 6.3.5.13 Bctu\_Ip\_ConfigChanListSimple()

```
void Bctu_Ip_ConfigChanListSimple (
    const uint32 u32Instance,
    const uint8 u8StartListElemIdx,
    const uint8 *const pConvListItems,
    const uint8 u8NumItems )
```

Set an array of ADC channel numbers in the conversion list.

This function sets an array of ADC channel numbers in the conversion list and marks the element corresponding to the last channel in the array, as last element in the list. Other configurable fields of a conversion element, like NEXT\_CH\_WAIT\_ON\_TRIG\_y is set to false/0. Implicitly wrap-around of the array onto the list, if (listSize - startElemIdx) < numArrayElems.

Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8StartListElemIdx</i>	- index in the list corresponding to the first element in the array
in	<i>pConvListItems</i>	- pointer to the first element in the ADC channel array
in	<i>u8NumItems</i>	- number of elements in the ADC channel numbers input array (required to be >0)

Returns

void

### 6.3.5.14 Bctu\_Ip\_SwTriggerConversion()

```
void Bctu_Ip_SwTriggerConversion (
    const uint32 u32Instance,
    const uint8 u8TrigIdx )
```

Triggers a conversion or list of conversions associated with a trigger index, if write protection is disabled.

This function triggers a conversion or list of conversions associated with a trigger index, if write protection is disabled. If write protection is enabled, the function will not trigger the conversion.

It is recommended that this function is used with HW triggers disabled for the target trigger index, to avoid ambiguity regarding source of trigger (hw or sw) for conversion results.

The BCTU HW will ignore attempts to trigger a new conversion or list, while there is a pending conversion or, in case of an active conversion list, before the last conversion is started.

Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8TrigIdx</i>	- index of the trigger register

Returns

void

#### 6.3.5.15 Bctu\_Ip\_GetConvData()

```
uint16 Bctu_Ip_GetConvData (
    const uint32 u32Instance,
    const uint8 u8AdcIdx )
```

Return the current conversion result data corresponding to an ADC instance.

This function returns the current conversion result data corresponding to an ADC instance. Note: In order to return the result with the correct resolution, the ADC\_SAR instance used must still be initialized when calling this function.

Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8AdcIdx</i>	- index of the target ADC module

Returns

uint16 - conversion result

#### 6.3.5.16 Bctu\_Ip\_GetConvResult()

```
void Bctu_Ip_GetConvResult (
    const uint32 u32Instance,
    const uint8 u8AdcIdx,
    Bctu_Ip_ResultType *const pResult )
```

Return the full conversion result information corresponding to an ADC instance.

This function returns the full conversion result information corresponding to an ADC instance. Note: In order to return the result with the correct resolution, the ADC\_SAR instance used must still be initialized when calling this function.

### Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8AdcIdx</i>	- index of the target ADC module
out	<i>pResult</i>	- pointer to the structure to be populated with the conversion result info

### Returns

void

#### 6.3.5.17 Bctu\_Ip\_GetFifoData()

```
uint16 Bctu_Ip_GetFifoData (
    const uint32 u32Instance,
    const uint8 u8FifoIdx )
```

Return the conversion result data stored in the index in a FIFO.

This function returns the conversion result data that is stored in the BCTU result FIFO at the given index. Note: In order to return the result with the correct resolution, the ADC\_SAR instance used must still be initialized when calling this function.

### Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8FifoIdx</i>	- index of the selected FIFO

### Returns

uint16 - FIFO content

#### 6.3.5.18 Bctu\_Ip\_GetFifoResult()

```
void Bctu_Ip_GetFifoResult (
    const uint32 u32Instance,
    const uint8 u8FifoIdx,
    Bctu_Ip_FifoResultType *const pResult )
```

Return the full conversion result information stored in a FIFO.

This function returns the full conversion result information that is stored in the BCTU result FIFO at the given index. Note: In order to return the result with the correct resolution, the ADC\_SAR instance used must still be initialized when calling this function.

## Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8FifoIdx</i>	- index of the selected FIFO
out	<i>pResult</i>	- pointer to the structure to be populated with the FIFO result info

## Returns

void

**6.3.5.19 Bctu\_Ip\_EnableNotifications()**

```
void Bctu_Ip_EnableNotifications (
    const uint32 u32Instance,
    const uint32 u32NotificationMask )
```

Enables notifications.

This function enables notifications selected by the given mask on the selected BCTU instance. The notifications mask parameter should be set using BCTU\_IP\_NOTIF\_... defines.

## Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u32NotificationMask</i>	- mask selecting targeted events

## Returns

void

**6.3.5.20 Bctu\_Ip\_DisableNotifications()**

```
void Bctu_Ip_DisableNotifications (
    const uint32 u32Instance,
    const uint32 u32NotificationMask )
```

Disables notifications.

This function disables notifications selected by the given mask on the selected BCTU instance. The notifications mask parameter should be set using BCTU\_IP\_NOTIF\_... defines.

Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u32NotificationMask</i>	- mask selecting targeted events

Returns

void

6.3.5.21 Bctu\_Ip\_SetFifoWatermark()

```
void Bctu_Ip_SetFifoWatermark (
    const uint32 u32Instance,
    const uint8 u8FifoIdx,
    const uint8 u8Watermark )
```

Sets watermark value for the selected FIFO.

This function sets the watermark value for the selected FIFO.

Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8FifoIdx</i>	- index of the selected FIFO
in	<i>u8Watermark</i>	- watermark value

Returns

void

6.3.5.22 Bctu\_Ip\_GetFifoCount()

```
uint8 Bctu_Ip_GetFifoCount (
    const uint32 u32Instance,
    const uint8 u8FifoIdx )
```

Get the counter value of the given FIFO.

This function gets the counter value of the given FIFO.



## Parameters

in	<i>u32Instance</i>	- BCTU instance number
in	<i>u8FifoIdx</i>	- index of the selected FIFO

## Returns

uint8 - FIFO counter value

## 6.4 IP\_SDADC

### 6.4.1 Detailed Description

#### Data Structures

- struct [DSPSS\\_CFSADC\\_FIRParamsType](#)
- struct [DSPSS\\_CFSADC\\_IIRParamsType](#)
- struct [DSPSS\\_CFSADC\\_CalibParamsType](#)
- struct [DSPSS\\_CFSADC\\_StandardParamsType](#)
- struct [DSPSS\\_CFSADC\\_ThreadParamsType](#)
- struct [DSPSS\\_CFSADC\\_ThreadDescriptorType](#)
- struct [DSPSS\\_ThreadConfigType](#)
- struct [Sdadc\\_Ip\\_ConfigType](#)

*Defines the converter configuration. [More...](#)*

- struct [Sdadc\\_Ip\\_StateStructType](#)

*Structure used to store runtime info. [More...](#)*

#### Macros

- `#define` [DSPSS\\_VENDOR\\_ID\\_TYPES](#)
- `#define` [DSPSS\\_CFSADC\\_MAX\\_FIR\\_SIZE](#)
- `#define` [DSPSS\\_CFSADC\\_MAX\\_IIR\\_SIZE](#)
- `#define` [DSPSS\\_CFSADC\\_STATUS\\_INACTIVE](#)
- `#define` [DSPSS\\_CFSADC\\_PARAMS\\_IDLE](#)
- `#define` [DSPSS\\_DMA\\_NONE](#)
- `#define` [DSPSS\\_CFSADC\\_OPTIMIZATION\\_NONE](#)
- `#define` [DSPSS\\_SAMPLE\\_THRESHOLD\\_MODE](#)
- `#define` [DSPSS\\_HALT\\_ACK\\_WAIT](#)
- `#define` [DSPSS\\_M7\\_WRITE\\_OWNER](#)
- `#define` [DSPSS\\_THREAD\\_ID0](#)
- `#define` [DSPSS\\_TRANSFER\\_DMA](#)
- `#define` [DSPSS\\_OK](#)
- `#define` [DSPSS\\_DSP\\_ERROR\\_INTR\\_ENABLE](#)
- `#define` [SDADC\\_IP\\_FLAG\\_DATA\\_FIFO\\_EMPTY](#)

*Macros for status flags.*

- `#define` [SDADC\\_IP\\_FLAG\\_WDG\\_UPPER\\_THRES\\_CROSS\\_OVER](#)
- `#define` [SDADC\\_IP\\_FLAG\\_WDG\\_LOWER\\_THRES\\_CROSS\\_OVER](#)
- `#define` [SDADC\\_IP\\_FLAG\\_CONVERTED\\_DATA\\_VALID](#)
- `#define` [SDADC\\_IP\\_FLAG\\_DATA\\_FIFO\\_OVERRUN](#)
- `#define` [SDADC\\_IP\\_FLAG\\_DATA\\_FIFO\\_FULL](#)
- `#define` [SDADC\\_IP\\_EVENT\\_FIFO\\_FULL](#)

*Macros for setting DMA and Interrupt request generating.*

- `#define` [SDADC\\_IP\\_EVENT\\_WDOG\\_CROSSOVER](#)
- `#define` [SDADC\\_IP\\_EVENT\\_FIFO\\_OVERRUN](#)
- `#define` [SDADC\\_IP\\_CHANNEL\\_ENUM\\_MODE\\_MASK](#)

- *The position of the mode bit in the input channel enum.*
- `#define SDADC_IP_CHANNEL_ENUM_VCOMSEL_MASK`
- *The position of the common voltage bias selection bits in the input channel enum.*
- `#define SDADC_IP_CHANNEL_ENUM_CHAN_MASK`
- *The position of the analog channel selection bits in the input channel enum.*
- `#define SDADC_IP_WDG_HIGH_FLAG`
- *Macros for watchdog registers.*
- `#define SDADC_IP_CHAN_SELECT(mode, vcom, chan)`
- *The macro to decode and compose bit mask for the input channel enum Enum value format:*
- `#define SDADC_IP_TRIGGER_DISABLE`
- *These macros are used to select which input source is used for hardware-triggered conversions.*
- `#define SDADC_IP_SW_0_TRIGGER`
- `#define SDADC_IP_SW_1_TRIGGER`
- `#define SDADC_IP_SW_2_TRIGGER`
- `#define SDADC_IP_SW_3_TRIGGER`
- `#define SDADC_IP_TRGMUX_28_32_36_40_TRIGGER`
- `#define SDADC_IP_ETPU_1_20_22_24_26_TRIGGER`
- `#define SDADC_IP_ETPU_1_21_23_25_27_TRIGGER`
- `#define SDADC_IP_ETPU_2_20_22_24_26_TRIGGER`
- `#define SDADC_IP_ETPU_2_21_23_25_27_TRIGGER`

## Types Reference

- `typedef uint8 Sdadc_Ip_TriggerSelectType`  
*Trigger Source Selection.*
- `typedef void Sdadc_Ip_WdgNotificationType(const uint16 ChanIdx, const uint8 Flags)`  
*Defines the watchdog notification header.*

## Enum Reference

- `enum Sdadc_Ip_StatusType`  
*SDADC status return type.*
- `enum Sdadc_Ip-DecimationRateType`  
*Programmable Decimation Rate.*
- `enum Sdadc_Ip_InputGainType`  
*Programmable Gain.*
- `enum Sdadc_Ip_TriggerEdgeType`  
*Trigger Edge Selection.*
- `enum Sdadc_Ip_ResolutionType`  
*Output data format selection.*
- `enum Sdadc_Ip_ChannelType`  
*Analog Channel Selection.*

## Function Reference

- DSPSS\_StatusType [DSPSS\\_Init](#) (void)
- void [DSPSS\\_DspInit](#) (void)
- DSPSS\_StatusType [DSPSS\\_Deinit](#) (void)
- DSPSS\_StatusType [DSPSS\\_FirmwareVersionGet](#) (uint16 \*const FirmwareVersion, const uint32 Length)
- DSPSS\_StatusType [DSPSS\\_ThreadConfigure](#) (const DSPSS\_ThreadIdType Id, const [DSPSS\\_ThreadConfigType](#) \*const Config)
- DSPSS\_StatusType [DSPSS\\_ThreadsInitialize](#) (void)
- DSPSS\_StatusType [DSPSS\\_Start](#) (void)
- DSPSS\_StatusType [DSPSS\\_Stop](#) (void)
- DSPSS\_StatusType [DSPSS\\_ThreadsCalibrationSet](#) (const DSPSS\_ThreadIdType Id, const [DSPSS\\_CFSADC\\_CalibParams](#) CalibrationParams)
- DSPSS\_CFSADC\_ThreadStatusType [DSPSS\\_ThreadsStatusGet](#) (const DSPSS\_ThreadIdType Id)
- uint32 [DSPSS\\_CoreBufferRead](#) (const DSPSS\_ThreadIdType Id, const uint32 Length, sint16 \*const Data)
- void [DSPSS\\_FlushOutputBuffer](#) (const DSPSS\_ThreadIdType Id, const DSPSS\_TransferMethodType TransferMethod)
- void [DSPSS\\_DspStartResetRoutine](#) (void)
- void [DSPSS\\_DspAssertReset](#) (void)
- void [DSPSS\\_ThreadSchedulingConfigure](#) (DSPSS\_SchedulingModeType SchedMode)
- void [DSPSS\\_ThreadSchedulingSampleThresholdMode](#) (const DSPSS\_NextReqHaltAckWaitModeType ContrlMode)
- void [DSPSS\\_ThreadSchedulingClockThresholdMode](#) (const DSPSS\_NextReqHaltAckWaitModeType ContrlMode, const DSPSS\_ThreadIdType SchdSeq1, const DSPSS\_ThreadIdType SchdSeq2, const DSPSS\_ThreadIdType SchdSeq3, const DSPSS\_ThreadIdType SchdSeq4)
- void [DSPSS\\_ThreadPCInitSet](#) (const DSPSS\_ThreadIdType Id, const uint16 PCInit)
- void [DSPSS\\_ThreadInputBufferConfigure](#) (const DSPSS\_ThreadIdType Id, const uint16 StartAddress, const uint16 EndAddress)
- void [DSPSS\\_ThreadThresholdSet](#) (const DSPSS\_ThreadIdType Id, const uint16 Threshold)
- void [DSPSS\\_ThreadEnable](#) (const DSPSS\_ThreadIdType Id)
- void [DSPSS\\_ThreadSuspend](#) (const DSPSS\_ThreadIdType Id)
- uint32 [DSPSS\\_ThreadGetOutputBufferStart](#) (const DSPSS\_ThreadIdType Id)
- uint32 [DSPSS\\_ThreadGetOutputBufferLength](#) (const DSPSS\_ThreadIdType Id)
- void [DSPSS\\_DspCoreBufferConfigure](#) (const DSPSS\_ThreadIdType Id, const uint16 StartAddr, const uint16 EndAddr, const DSPSS\_DspCoreBufferOwnerType Owner)
- void [DSPSS\\_DspCoreBufferEnable](#) (const DSPSS\_ThreadIdType Id)
- void [DSPSS\\_DspCoreBufferDisable](#) (const DSPSS\_ThreadIdType Id)
- uint16 [DSPSS\\_DspCoreBufferCurrReadPtrGet](#) (const DSPSS\_ThreadIdType Id)
- uint32 [DSPSS\\_DspCoreBufferOverflowCheck](#) (const DSPSS\_ThreadIdType Id)
- void [DSPSS\\_DspCoreBufferOverflowClear](#) (const DSPSS\_ThreadIdType Id)
- void [DSPSS\\_InterruptEnable](#) (const uint32 Id, const DSPSS\_InterruptTypeType InterruptMask)
- void [DSPSS\\_InterruptDisable](#) (const uint32 Id, const DSPSS\_InterruptTypeType InterruptMask)
- void [DSPSS\\_DmaReadBufferConfigure](#) (const DSPSS\_CFSADC\_DmaChannelType DmaChannel, const uint16 StartAddr, const uint16 EndAddr)
- void [DSPSS\\_DmaReadBufferEnable](#) (const DSPSS\_CFSADC\_DmaChannelType DmaChannel)
- void [DSPSS\\_DmaReadBufferDisable](#) (const DSPSS\_CFSADC\_DmaChannelType DmaChannel)
- boolean [DSPSS\\_DmaReadBufferWrapPtrCheck](#) (const DSPSS\_CFSADC\_DmaChannelType DmaChannel)
- Sdadc\_Ip\_StatusType [Sdadc\\_Ip\\_Init](#) (const uint32 Instance, const [Sdadc\\_Ip\\_ConfigType](#) \*const Config)  
*Configures the hardware instance with the given configuration structure.*
- void [Sdadc\\_Ip\\_Deinit](#) (const uint32 Instance)

- *Deinitialize SDADC module.*
- void [Sdadc\\_Ip\\_Powerup](#) (const uint32 Instance)
- *Power up the SDADC.*
- void [Sdadc\\_Ip\\_Powerdown](#) (const uint32 Instance)
- *Power down the SDADC.*
- void [Sdadc\\_Ip\\_ReloadConversion](#) (const uint32 Instance)
- *Reload SDADC conversion.*
- void [Sdadc\\_Ip\\_SwTriggerConversion](#) (const uint32 Instance)
- *Generate software trigger.*
- void [Sdadc\\_Ip\\_SetWdgThreshold](#) (const uint32 Instance, const uint16 UpperThreshold, const uint16 LowerThreshold)
- *Configure watchdog threshold register.*
- void [Sdadc\\_Ip\\_EnableWatchdog](#) (const uint32 Instance)
- *Enables the watchdog monitor.*
- void [Sdadc\\_Ip\\_DisableWatchdog](#) (const uint32 Instance)
- *Disables the watchdog monitor.*
- void [Sdadc\\_Ip\\_EnableWraparound](#) (const uint32 Instance)
- *Enable Wrap Around mode.*
- void [Sdadc\\_Ip\\_DisableWraparound](#) (const uint32 Instance)
- *Disable Wrap Around mode.*
- void [Sdadc\\_Ip\\_SetInputChannel](#) (const uint32 Instance, const [Sdadc\\_Ip\\_ChannelType](#) Channel)
- *Select input analog channel.*
- void [Sdadc\\_Ip\\_FlushFifo](#) (const uint32 Instance)
- *Flush data FIFO.*
- uint32 [Sdadc\\_Ip\\_GetConvData](#) (const uint32 Instance, const uint32 Length, sint16 \*const Data)
- *Gets the conversion data.*
- uint32 [Sdadc\\_Ip\\_GetRawConvDataFifo](#) (const uint32 Instance, const uint32 Length, sint16 \*const Data)
- *Gets the raw conversion data in FIFO.*
- uint32 [Sdadc\\_Ip\\_GetStatusFlags](#) (const uint32 Instance)
- *Get the status flags.*
- void [Sdadc\\_Ip\\_ClearStatusFlags](#) (const uint32 Instance, const uint32 Mask)
- *Clear the status flags.*
- void [Sdadc\\_Ip\\_EnableDmaEvents](#) (const uint32 Instance, const uint32 EventsMask)
- *Enables SDADC DMA request generation.*
- void [Sdadc\\_Ip\\_EnableInterruptEvents](#) (const uint32 Instance, const uint32 EventsMask)
- *Enables SDADC interrupt request generation.*
- void [Sdadc\\_Ip\\_DisableInterruptEvents](#) (const uint32 Instance, const uint32 EventsMask)
- *Disable SDADC DMA and interrupt request generation.*
- [Sdadc\\_Ip\\_StatusType](#) [Sdadc\\_Ip\\_DoCalibration](#) (const uint32 Instance)
- *Perform Calibration of the SDADC.*
- void [Sdadc\\_Ip\\_ApplyCalibration](#) (const uint32 Instance, const uint32 BufferLength, const sint16 \*const UncalibratedBuffer, sint16 \*const CalibratedBuffer)
- *Calibrate the converted data of the SDADC.*
- void [Sdadc\\_Ip\\_EnableHwTrigger](#) (const uint32 Instance)
- *Enable the hardware trigger.*
- void [Sdadc\\_Ip\\_DisableHwTrigger](#) (const uint32 Instance)
- *Disable the hardware trigger.*

- void [Sdadc\\_Ip\\_SetHwTrigger](#) (const uint32 Instance, const [Sdadc\\_Ip\\_TriggerEdgeType](#) TriggerEdge, const [Sdadc\\_Ip\\_TriggerSelectType](#) TriggerSrc)  
*Setting the input hardware trigger.*
- uint32 [Sdadc\\_Ip\\_GetDataAddress](#) (const uint32 Instance)  
*Return the address of the converted data.*

## Variables

- DSPSS\_CFSADC\_OptimizationLevelType [OptimizationLevel](#)
- uint16 [UpsamplingFactor](#)
- uint16 [DownsamplingFactor](#)
- uint16 [NbTaps](#)
- sint16 [Taps](#) [(128U)]
- uint16 [Order](#)
- sint16 [Taps](#) [(5U)]
- uint16 [Padd](#)
- sint16 [UseCalibration](#)
- sint16 [Gain](#)
- sint32 [Offset](#)
- DSPSS\_CFSADC\_FIRParamsType [FirParams](#)
- DSPSS\_CFSADC\_IIRParamsType [IirParams](#)
- DSPSS\_CFSADC\_CalibParamsType [CalibParams](#)
- uint16 [NbSkippedSamples](#)
- uint16 [Padd](#)
- DSPSS\_CFSADC\_StandardParamsType [StandardParams](#)
- volatile DSPSS\_CFSADC\_ThreadStatusType [Status](#)
- volatile DSPSS\_CFSADC\_ConfigMessageType [ConfigMessage](#)
- volatile uint16 [Counter](#)
- DSPSS\_CFSADC\_DmaChannelType [InputDMAChannel](#)
- DSPSS\_CFSADC\_DmaChannelType [OutputDMAChannel](#)
- uint16 [EntryPoint](#)
- uint16 [Stack](#)
- uint16 [InputThreshold](#)
- uint16 [BufferInputStart](#)
- uint16 [BufferInputLength](#)
- uint16 [OutputThreshold](#)
- uint16 [BufferOutputStart](#)
- uint16 [BufferOutputLength](#)
- uint16 [CoreMsgQueueStart](#)
- uint16 [CoreMsgQueueLength](#)
- uint16 [WorkAreaStart](#)
- uint16 [WorkAreaLength](#)
- uint16 [Padd](#)
- DSPSS\_CFSADC\_ThreadParamsType [ThreadParams](#)
- DSPSS\_SchedulingModeType [SchedMode](#)
- uint16 [ThreadThreshold](#)
- uint16 [InputThreshold](#)
- uint16 [OutputThreshold](#)
- DSPSS\_DspCoreBufferOwnerType [Owner](#)
- DSPSS\_TransferMethodType [InputTransferMethod](#)
- DSPSS\_TransferMethodType [OutputTransferMethod](#)

## 6.4.2 Data Structure Documentation

### 6.4.2.1 struct DSPSS\_CFSDADC\_FIRParamsType

FIR filter config filled by M7, interpreted and managed by CoolFlux

Definition at line 200 of file DSPSS\_Types.h.

### 6.4.2.2 struct DSPSS\_CFSDADC\_IIRParamsType

IIR filter config filled by M7, interpreted and managed by CoolFlux

Definition at line 213 of file DSPSS\_Types.h.

### 6.4.2.3 struct DSPSS\_CFSDADC\_CalibParamsType

Calibration config filled by M7, interpreted and managed by CoolFlux

Definition at line 225 of file DSPSS\_Types.h.

### 6.4.2.4 struct DSPSS\_CFSDADC\_StandardParamsType

Parameters for standard filtering using FIR, IIR and calibration

Definition at line 236 of file DSPSS\_Types.h.

### 6.4.2.5 struct DSPSS\_CFSDADC\_ThreadParamsType

Thread parameters specific to each use-case

Definition at line 249 of file DSPSS\_Types.h.

### 6.4.2.6 struct DSPSS\_CFSDADC\_ThreadDescriptorType

Type definition that contains the thread configuration.

Definition at line 258 of file DSPSS\_Types.h.

### 6.4.2.7 struct DSPSS\_ThreadConfigType

Type definition that contains the thread configuration set by M7.

Definition at line 375 of file DSPSS\_Types.h.

### 6.4.2.8 struct Sdadc\_Ip\_ConfigType

Defines the converter configuration.

This structure is used to configure the SDADC converter

Implements : [Sdadc\\_Ip\\_ConfigType](#)

Definition at line 409 of file Sdadc\_Ip\_Types.h.

#### Data Fields

- [Sdadc\\_Ip\\_InputGainType](#) InputGain
- [Sdadc\\_Ip\\_DecimationRateType](#) DecimaRate
- uint8 OutputSetDelay
- [Sdadc\\_Ip\\_ResolutionType](#) Resolution
- uint8 CalibSkipped
- uint8 CalibAverage
- [Sdadc\\_Ip\\_TriggerSelectType](#) TrigSelect
- [Sdadc\\_Ip\\_TriggerEdgeType](#) TrigEdge
- const [Sdadc\\_Ip\\_ChannelType](#) \* ChannelConfigsPtr
- boolean WrapAroundEnable
- uint8 Wraparound
- boolean FifoEnable
- boolean FifoOverwrite
- uint8 FifoThreshold
- boolean WdgEnable
- uint16 WdgUpperThreshold
- uint16 WdgLowerThreshold
- boolean StopInDebug
- const [DSPSS\\_ThreadConfigType](#) \* DspssConfig
- void(\* [FifoFullNotification](#) )(void)
- void(\* [FifoOverrunNotification](#) )(void)
- [Sdadc\\_Ip\\_WdgNotificationType](#) \* WdgCrossOverNotification

#### 6.4.2.8.1 Field Documentation

##### 6.4.2.8.1.1 InputGain [Sdadc\\_Ip\\_InputGainType](#) InputGain

Programmable Gain

Definition at line 411 of file Sdadc\_Ip\_Types.h.



**6.4.2.8.1.2 DecimaRate** `Sdadc_Ip_DecimationRateType` DecimaRate

Programmable Decimation Rate

Definition at line 412 of file Sdadc\_Ip\_Types.h.

**6.4.2.8.1.3 OutputSetDelay** `uint8` OutputSetDelay

Output Settling Delay

Definition at line 413 of file Sdadc\_Ip\_Types.h.

**6.4.2.8.1.4 Resolution** `Sdadc_Ip_ResolutionType` Resolution

Output data format

Definition at line 414 of file Sdadc\_Ip\_Types.h.

**6.4.2.8.1.5 CalibSkipped** `uint8` CalibSkipped

The number of first samples will be skipped to ignore the unstable results before performing actual SDADC calibration

Definition at line 416 of file Sdadc\_Ip\_Types.h.

**6.4.2.8.1.6 CalibAverage** `uint8` CalibAverage

The number of consecutive calibration samples to be read to calculate the average SDADC calibration output

Definition at line 417 of file Sdadc\_Ip\_Types.h.

**6.4.2.8.1.7 TrigSelect** `Sdadc_Ip_TriggerSelectType` TrigSelect

Trigger input selection

Definition at line 419 of file Sdadc\_Ip\_Types.h.

### 6.4.2.8.1.8 TrigEdge `Sdadc_Ip_TriggerEdgeType` TrigEdge

Trigger edge selection, this member is not influent if trigger selection is disabled

Definition at line 420 of file `Sdadc_Ip_Types.h`.

### 6.4.2.8.1.9 ChannelConfigsPtr `const Sdadc_Ip_ChannelType*` ChannelConfigsPtr

Select analog input is connected to SDADC terminals. In Wrap Around mode, this indicates the initial entry value for the first loop. Only the first element will be used by `Sdadc_Ip_Init()`

Definition at line 423 of file `Sdadc_Ip_Types.h`.

### 6.4.2.8.1.10 WrapAroundEnable `boolean` WrapAroundEnable

Enable Wrap-Around mode

Definition at line 427 of file `Sdadc_Ip_Types.h`.

### 6.4.2.8.1.11 Wraparound `uint8` Wraparound

In Wrap Around mode, this indicates the maximum value of the wraparound counter

Definition at line 428 of file `Sdadc_Ip_Types.h`.

### 6.4.2.8.1.12 FifoEnable `boolean` FifoEnable

Enable Fifo

Definition at line 430 of file `Sdadc_Ip_Types.h`.

### 6.4.2.8.1.13 FifoOverwrite `boolean` FifoOverwrite

Fifo Over Write Enable

Definition at line 431 of file `Sdadc_Ip_Types.h`.

**6.4.2.8.1.14 FifoThreshold** `uint8 FifoThreshold`

Fifo Threshold

Definition at line 432 of file `Sdadc_Ip_Types.h`.

**6.4.2.8.1.15 WdgEnable** `boolean WdgEnable`

Enable Wdg monitor

Definition at line 434 of file `Sdadc_Ip_Types.h`.

**6.4.2.8.1.16 WdgUpperThreshold** `uint16 WdgUpperThreshold`

Wdg Upper Threshold Value

Definition at line 435 of file `Sdadc_Ip_Types.h`.

**6.4.2.8.1.17 WdgLowerThreshold** `uint16 WdgLowerThreshold`

Wdg Lower Threshold Value

Definition at line 436 of file `Sdadc_Ip_Types.h`.

**6.4.2.8.1.18 StopInDebug** `boolean StopInDebug`

Enable stopping the SDADC conversions when the chip enters debug mode

Definition at line 438 of file `Sdadc_Ip_Types.h`.

**6.4.2.8.1.19 DspssConfig** `const DSPSS\_ThreadConfigType* DspssConfig`

Dspss configuration for processing (e.g. FIR, FFT) SDADC data (set as `NULL_PTR` to disable)

Definition at line 441 of file `Sdadc_Ip_Types.h`.

### 6.4.2.8.1.20 FifoFullNotification `void(* FifoFullNotification) (void)`

Fifo full event

Definition at line 445 of file `Sdadc_Ip_Types.h`.

### 6.4.2.8.1.21 FifoOverrunNotification `void(* FifoOverrunNotification) (void)`

Fifo overrun event

Definition at line 446 of file `Sdadc_Ip_Types.h`.

### 6.4.2.8.1.22 WdgCrossOverNotification `Sdadc_Ip_WdgNotificationType* WdgCrossOverNotification`

Wdg cross over event

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

## 6.4.2.9 struct Sdadc\_Ip\_StateStructType

Structure used to store runtime info.

This structure is used to store SDADC runtime info

Implements : `Sdadc_Ip_StateStructType_Class`

Definition at line 458 of file `Sdadc_Ip_Types.h`.

Data Fields

Type	Name	Description
const <code>Sdadc_Ip_ConfigType *</code>	Config	Point to user's configuration structure (Also used to check if the driver was initialized)

## 6.4.3 Macro Definition Documentation

### 6.4.3.1 DSPSS\_VENDOR\_ID\_TYPES

```
#define DSPSS_VENDOR_ID_TYPES
```

This file contains register definitions and macros for easy access to their bit fields.

This file assumes LITTLE endian system.

Definition at line 60 of file DSPSS\_Types.h.

#### 6.4.3.2 DSPSS\_CFSADC\_MAX\_FIR\_SIZE

```
#define DSPSS_CFSADC_MAX_FIR_SIZE
```

Maximum number of taps for filters Maximum number of taps for FIR filter

Definition at line 140 of file DSPSS\_Types.h.

#### 6.4.3.3 DSPSS\_CFSADC\_MAX\_IIR\_SIZE

```
#define DSPSS_CFSADC_MAX_IIR_SIZE
```

Maximum number of taps for IIR filter

Definition at line 141 of file DSPSS\_Types.h.

#### 6.4.3.4 DSPSS\_CFSADC\_STATUS\_INACTIVE

```
#define DSPSS_CFSADC_STATUS_INACTIVE
```

Enumeration identifying the thread status.

Definition at line 151 of file DSPSS\_Types.h.

#### 6.4.3.5 DSPSS\_CFSADC\_PARAMS\_IDLE

```
#define DSPSS_CFSADC_PARAMS_IDLE
```

Enumeration identifying the configuration message status.

Definition at line 167 of file DSPSS\_Types.h.

### 6.4.3.6 DSPSS\_DMA\_NONE

```
#define DSPSS_DMA_NONE
```

Enumeration identifying the different DMA channels.

Definition at line 176 of file DSPSS\_Types.h.

### 6.4.3.7 DSPSS\_CFSADC\_OPTIMIZATION\_NONE

```
#define DSPSS_CFSADC_OPTIMIZATION_NONE
```

Enumeration identifying the different optimization levels.

Definition at line 191 of file DSPSS\_Types.h.

### 6.4.3.8 DSPSS\_SAMPLE\_THRESHOLD\_MODE

```
#define DSPSS_SAMPLE_THRESHOLD_MODE
```

Enumeration values for the different scheduling mode

Definition at line 296 of file DSPSS\_Types.h.

### 6.4.3.9 DSPSS\_HALT\_ACK\_WAIT

```
#define DSPSS_HALT_ACK_WAIT
```

Enumeration identifying the different controlling modes of thread scheduling in sample threshold mode.

Definition at line 306 of file DSPSS\_Types.h.

### 6.4.3.10 DSPSS\_M7\_WRITE\_OWNER

```
#define DSPSS_M7_WRITE_OWNER
```

Enumeration values for the read and write pointer owners of DSP-Core buffer

Definition at line 315 of file DSPSS\_Types.h.

**6.4.3.11 DSPSS\_THREAD\_ID0**

```
#define DSPSS_THREAD_ID0
```

Enumeration identifying the different threads.

Definition at line 324 of file DSPSS\_Types.h.

**6.4.3.12 DSPSS\_TRANSFER\_DMA**

```
#define DSPSS_TRANSFER_DMA
```

Enumeration identifying the different transfer types. NOTE: For input all three are possible For output only first and third are possible

Definition at line 338 of file DSPSS\_Types.h.

**6.4.3.13 DSPSS\_OK**

```
#define DSPSS_OK
```

Enumeration values for return codes

Definition at line 348 of file DSPSS\_Types.h.

**6.4.3.14 DSPSS\_DSP\_ERROR\_INTR\_ENABLE**

```
#define DSPSS_DSP_ERROR_INTR_ENABLE
```

Enumeration values for the different interrupt type.

Definition at line 360 of file DSPSS\_Types.h.

**6.4.3.15 SDADC\_IP\_FLAG\_DATA\_FIFO\_EMPTY**

```
#define SDADC_IP_FLAG_DATA_FIFO_EMPTY
```

Macros for status flags.

These flags map to internal hardware flags in the status flag register. ORing these macros to clear multiple flags. DATA FIFO is empty

Definition at line 135 of file Sdadc\_Ip\_Types.h.

### 6.4.3.16 SDADC\_IP\_FLAG\_WDG\_UPPER\_THRES\_CROSS\_OVER

```
#define SDADC_IP_FLAG_WDG_UPPER_THRES_CROSS_OVER
```

Watchdog Upper Threshold is cross Over

Definition at line 136 of file Sdadc\_Ip\_Types.h.

### 6.4.3.17 SDADC\_IP\_FLAG\_WDG\_LOWER\_THRES\_CROSS\_OVER

```
#define SDADC_IP_FLAG_WDG_LOWER_THRES_CROSS_OVER
```

Watchdog Lower Threshold is cross Over

Definition at line 137 of file Sdadc\_Ip\_Types.h.

### 6.4.3.18 SDADC\_IP\_FLAG\_CONVERTED\_DATA\_VALID

```
#define SDADC_IP_FLAG_CONVERTED_DATA_VALID
```

Converted Data is Valid

Definition at line 138 of file Sdadc\_Ip\_Types.h.

### 6.4.3.19 SDADC\_IP\_FLAG\_DATA\_FIFO\_OVERRUN

```
#define SDADC_IP_FLAG_DATA_FIFO_OVERRUN
```

Data FIFO is overrun

Definition at line 139 of file Sdadc\_Ip\_Types.h.

### 6.4.3.20 SDADC\_IP\_FLAG\_DATA\_FIFO\_FULL

```
#define SDADC_IP_FLAG_DATA_FIFO_FULL
```

Data FIFO is full

Definition at line 140 of file Sdadc\_Ip\_Types.h.



#### 6.4.3.21 SDADC\_IP\_EVENT\_FIFO\_FULL

```
#define SDADC_IP_EVENT_FIFO_FULL
```

Macros for setting DMA and Interrupt request generating.

These macros are directly mapped to bits in the RSER register. ORing these macros to setting DMA and Interrupt request generating of multiple events Data FIFO Full Event

Definition at line 163 of file Sdadc\_Ip\_Types.h.

#### 6.4.3.22 SDADC\_IP\_EVENT\_WDOG\_CROSSOVER

```
#define SDADC_IP_EVENT_WDOG_CROSSOVER
```

WDG Threshold Cross Over Event

Definition at line 164 of file Sdadc\_Ip\_Types.h.

#### 6.4.3.23 SDADC\_IP\_EVENT\_FIFO\_OVERRUN

```
#define SDADC_IP_EVENT_FIFO_OVERRUN
```

Data FIFO Full Event, only used to enable interrupt

Definition at line 165 of file Sdadc\_Ip\_Types.h.

#### 6.4.3.24 SDADC\_IP\_CHANNEL\_ENUM\_MODE\_MASK

```
#define SDADC_IP_CHANNEL_ENUM_MODE_MASK
```

The position of the mode bit in the input channel enum.

Definition at line 169 of file Sdadc\_Ip\_Types.h.

### 6.4.3.25 SDADC\_IP\_CHANNEL\_ENUM\_VCOMSEL\_MASK

```
#define SDADC_IP_CHANNEL_ENUM_VCOMSEL_MASK
```

The position of the common voltage bias selection bits in the input channel enum.

Definition at line 173 of file Sdadc\_Ip\_Types.h.

### 6.4.3.26 SDADC\_IP\_CHANNEL\_ENUM\_CHAN\_MASK

```
#define SDADC_IP_CHANNEL_ENUM_CHAN_MASK
```

The position of the analog channel selection bits in the input channel enum.

Definition at line 177 of file Sdadc\_Ip\_Types.h.

### 6.4.3.27 SDADC\_IP\_WDG\_HIGH\_FLAG

```
#define SDADC_IP_WDG_HIGH_FLAG
```

Macros for watchdog registers.

These macros help decode and compose bit mask for watchdog functionality

Definition at line 186 of file Sdadc\_Ip\_Types.h.

### 6.4.3.28 SDADC\_IP\_CHAN\_SELECT

```
#define SDADC_IP_CHAN_SELECT(  
    mode,  
    vcom,  
    chan )
```

The macro to decode and compose bit mask for the input channel enum Enum value format:

- Bit field [5] : Mode Selection
- Bit fields [4-3]: Common Voltage Bias Selection
- Bit fields [2-0]: Analog Channel Selection

Definition at line 195 of file Sdadc\_Ip\_Types.h.

#### 6.4.3.29 SDADC\_IP\_TRIGGER\_DISABLE

```
#define SDADC_IP_TRIGGER_DISABLE
```

These macros are used to select which input source is used for hardware-triggered conversions.

Trigger is disabled. The configured instance will not be triggered by any trigger source

Definition at line 200 of file Sdadc\_Ip\_Types.h.

#### 6.4.3.30 SDADC\_IP\_SW\_0\_TRIGGER

```
#define SDADC_IP_SW_0_TRIGGER
```

SDADC\_IP\_0 output software trigger is selected. The configured instance will be triggered by SDADC0 SW trigger output

Definition at line 202 of file Sdadc\_Ip\_Types.h.

#### 6.4.3.31 SDADC\_IP\_SW\_1\_TRIGGER

```
#define SDADC_IP_SW_1_TRIGGER
```

SDADC\_IP\_1 output software trigger is selected. The configured instance will be triggered by SDADC1 SW trigger output

Definition at line 203 of file Sdadc\_Ip\_Types.h.

#### 6.4.3.32 SDADC\_IP\_SW\_2\_TRIGGER

```
#define SDADC_IP_SW_2_TRIGGER
```

SDADC\_IP\_2 output software trigger is selected. The configured instance will be triggered by SDADC2 SW trigger output

Definition at line 204 of file Sdadc\_Ip\_Types.h.

### 6.4.3.33 SDADC\_IP\_SW\_3\_TRIGGER

```
#define SDADC_IP_SW_3_TRIGGER
```

SDADC\_IP\_3 output software trigger is selected. The configured instance will be triggered by SDADC3 SW trigger output

Definition at line 205 of file Sdadc\_Ip\_Types.h.

### 6.4.3.34 SDADC\_IP\_TRGMUX\_28\_32\_36\_40\_TRIGGER

```
#define SDADC_IP_TRGMUX_28_32_36_40_TRIGGER
```

TRGMUX output port 28, 32, 36, 40 trigger is selected to correspond to SDADC instance 0, 1, 2, 3. The configured instance will be triggered by the corresponding TRGMUX output port

Definition at line 208 of file Sdadc\_Ip\_Types.h.

### 6.4.3.35 SDADC\_IP\_ETPU\_1\_20\_22\_24\_26\_TRIGGER

```
#define SDADC_IP_ETPU_1_20_22_24_26_TRIGGER
```

ETPU\_1 channel 20, 22, 24, 26 output trigger is selected to correspond to SDADC instance 0, 1, 2, 3. The configured instance will be triggered by the corresponding ETPU\_1 channel output

Definition at line 211 of file Sdadc\_Ip\_Types.h.

### 6.4.3.36 SDADC\_IP\_ETPU\_1\_21\_23\_25\_27\_TRIGGER

```
#define SDADC_IP_ETPU_1_21_23_25_27_TRIGGER
```

ETPU\_1 channel 21, 23, 25, 27 output trigger is selected to correspond to SDADC instance 0, 1, 2, 3. The configured instance will be triggered by the corresponding ETPU\_1 channel output

Definition at line 214 of file Sdadc\_Ip\_Types.h.

### 6.4.3.37 SDADC\_IP\_ETPU\_2\_20\_22\_24\_26\_TRIGGER

```
#define SDADC_IP_ETPU_2_20_22_24_26_TRIGGER
```

ETPU\_2 channel 20, 22, 24, 26 output trigger is selected to correspond to SDADC instance 0, 1, 2, 3. The configured instance will be triggered by the corresponding ETPU\_2 channel output

Definition at line 217 of file Sdadc\_Ip\_Types.h.

### 6.4.3.38 SDADC\_IP\_ETPU\_2\_21\_23\_25\_27\_TRIGGER

```
#define SDADC_IP_ETPU_2_21_23_25_27_TRIGGER
```

ETPU\_2 channel 21, 23, 25, 27 output trigger is selected to correspond to SDADC instance 0, 1, 2, 3. The configured instance will be triggered by the corresponding ETPU\_2 channel output

Definition at line 220 of file Sdadc\_Ip\_Types.h.

## 6.4.4 Types Reference

### 6.4.4.1 Sdadc\_Ip\_TriggerSelectType

```
typedef uint8 Sdadc_Ip_TriggerSelectType
```

Trigger Source Selection.

This enum is used to select which input is used for hardware-triggered conversions.

Implements : Sdadc\_Ip\_TriggerSelectType\_Class

Definition at line 303 of file Sdadc\_Ip\_Types.h.

### 6.4.4.2 Sdadc\_Ip\_WdgNotificationType

```
typedef void Sdadc_Ip_WdgNotificationType(const uint16 ChanIdx, const uint8 Flags)
```

Defines the watchdog notification header.

This header is used for out of range watchdog notification callbacks u8Flags will contain which WDG flag was triggered(low and/or high). These are defined by SDADC\_IP\_WDG\_LOW\_FLAG and SDADC\_IP\_WDG\_HIGH\_FLAG

Implements : Sdadc\_Ip\_WdgNotificationType\_Class

Definition at line 399 of file Sdadc\_Ip\_Types.h.

## 6.4.5 Enum Reference

### 6.4.5.1 Sdadc\_Ip\_StatusType

```
enum Sdadc_Ip_StatusType
```

SDADC status return type.

This enum is used as return type

Implements : Sdadc\_Ip\_StatusType\_Class

Enumerator

SDADC_IP_STATUS_SUCCESS	Function completed successfully
SDADC_IP_STATUS_ERROR	Function didn't complete successfully
SDADC_IP_STATUS_TIMEOUT	Function timed out

Definition at line 234 of file Sdadc\_Ip\_Types.h.

### 6.4.5.2 Sdadc\_Ip\_DecimationRateType

```
enum Sdadc_Ip_DecimationRateType
```

Programmable Decimation Rate.

This enum is used to configure the programmable Decimation Rate to select the over-sampling ratio (OSR) to be applied to support different passbands with a fixed input sampling clock. When external modulator is selected, the output data rate is independent of this field and is fixed to  $fs\_ext/128$

Implements : Sdadc\_Ip\_DecimationRateType\_Class

Enumerator

SDADC_IP_DECIMATION_RATE_120	OSR is 120
SDADC_IP_DECIMATION_RATE_140	OSR is 140
SDADC_IP_DECIMATION_RATE_160	OSR is 160
SDADC_IP_DECIMATION_RATE_180	OSR is 180
SDADC_IP_DECIMATION_RATE_200	OSR is 200
SDADC_IP_DECIMATION_RATE_220	OSR is 220
SDADC_IP_DECIMATION_RATE_240	OSR is 240
SDADC_IP_DECIMATION_RATE_280	OSR is 280
SDADC_IP_DECIMATION_RATE_320	OSR is 320
SDADC_IP_DECIMATION_RATE_360	OSR is 360
SDADC_IP_DECIMATION_RATE_376	OSR is 376
SDADC_IP_DECIMATION_RATE_400	OSR is 400
SDADC_IP_DECIMATION_RATE_440	OSR is 440
SDADC_IP_DECIMATION_RATE_480	OSR is 480
SDADC_IP_DECIMATION_RATE_560	OSR is 560
SDADC_IP_DECIMATION_RATE_640	OSR is 640
SDADC_IP_DECIMATION_RATE_720	OSR is 720
SDADC_IP_DECIMATION_RATE_800	OSR is 800
SDADC_IP_DECIMATION_RATE_880	OSR is 880
SDADC_IP_DECIMATION_RATE_960	OSR is 960
SDADC_IP_DECIMATION_RATE_1120	OSR is 1120
SDADC_IP_DECIMATION_RATE_1280	OSR is 1280
SDADC_IP_DECIMATION_RATE_125	OSR is 125 (special case. CIC decimation rate is 25 and FIR is suppose to do decimation by 5)

Definition at line 251 of file Sdadc\_Ip\_Types.h.

### 6.4.5.3 Sdadc\_Ip\_InputGainType

```
enum Sdadc_Ip_InputGainType
```

Programmable Gain.

This enum is used to select the gain to be applied to the analog input stage of SDADC. The effective analog input becomes the input voltage level multiplied by the gain factor.

Implements : Sdadc\_Ip\_InputGainType\_Class

Enumerator

SDADC_IP_INPUT_GAIN↔ _1	Input gain is 1
SDADC_IP_INPUT_GAIN↔ _2	Input gain is 2
SDADC_IP_INPUT_GAIN↔ _4	Input gain is 4
SDADC_IP_INPUT_GAIN↔ _8	Input gain is 8

Definition at line 287 of file Sdadc\_Ip\_Types.h.

### 6.4.5.4 Sdadc\_Ip\_TriggerEdgeType

```
enum Sdadc_Ip_TriggerEdgeType
```

Trigger Edge Selection.

This enum is used to select the edge of input hardware trigger

Implements : Sdadc\_Ip\_TriggerEdgeType\_Class

Enumerator

SDADC_IP_TRIGGER_FALLING_EDGE	Falling edge
SDADC_IP_TRIGGER_RISING_EDGE	Rising edge
SDADC_IP_TRIGGER_BOTH_EDGE	Both edges

Definition at line 313 of file Sdadc\_Ip\_Types.h.

#### 6.4.5.5 Sdadc\_Ip\_ResolutionType

enum [Sdadc\\_Ip\\_ResolutionType](#)

Output data format selection.

This enum is used to control the output data in CDR.

Implements : Sdadc\_Ip\_ResolutionType\_Class

Enumerator

SDADC_IP_RESOLUTION_15	Output data is 15-bit unsigned
SDADC_IP_RESOLUTION_16	Output data is signed and sign extended to 16 bits

Definition at line 327 of file Sdadc\_Ip\_Types.h.

#### 6.4.5.6 Sdadc\_Ip\_ChannelType

enum [Sdadc\\_Ip\\_ChannelType](#)

Analog Channel Selection.

This enum is used to select analog input channel

Bit fields format:

- Bit fields [5] : Mode Selection (MODE)
- Bit fields [4-3]: Common Voltage Bias Selection (VCOMSEL)
- Bit fields [2-0]: Analog Channel Selection (ANCHSEL)

Implements : Sdadc\_Ip\_ChannelType\_Class

Enumerator

SDADC_IP_CHAN_AN0_AN1	Differential input mode AN0 to INP and AN1 to INM
SDADC_IP_CHAN_AN2_AN3	AN2 to INP and AN3 to INM
SDADC_IP_CHAN_AN4_AN5	AN4 to INP and AN5 to INM
SDADC_IP_CHAN_AN6_AN7	AN6 to INP and AN7 to INM
SDADC_IP_CHAN_VCOM1_VCOM1	VCOM1_HV to both INP and INM
SDADC_IP_CHAN_VCOM2_VCOM2	VCOM2_HV to both INP and INM



Enumerator

SDADC_IP_CHAN_VREFP_VREFN	VREFP to INP and VREFN to INM
SDADC_IP_CHAN_VREFN_VREFP	VREFN to INP and VREFP to INM Single-ended input mode, AN[x] is connected to INP terminal, VCOM0 is connected to INM terminal
SDADC_IP_CHAN_AN7_VCOM0	Single-ended input mode, AN[x] is connected to INP terminal, VCOM1 is connected to INM terminal
SDADC_IP_CHAN_AN7_VCOM1	Single-ended input mode, AN[x] is connected to INP terminal, VCOM2 is connected to INM terminal

Definition at line 346 of file Sdadc\_Ip\_Types.h.

## 6.4.6 Function Reference

### 6.4.6.1 DSPSS\_Init()

```
DSPSS_StatusType DSPSS_Init (
    void )
```

Initializes the DSPSS by loading the FW init PMEM and default configuration into XMEM.

Returns

Return error code

### 6.4.6.2 DSPSS\_DspInit()

```
void DSPSS_DspInit (
    void )
```

DSP initialization

### 6.4.6.3 DSPSS\_Deinit()

```
DSPSS_StatusType DSPSS_Deinit (
    void )
```

This function allows the user to de-initialize the DSPSS

Returns

Return error code

### 6.4.6.4 DSPSS\_FirmwareVersionGet()

```
DSPSS_StatusType DSPSS_FirmwareVersionGet (
    uint16 *const FirmwareVersion,
    const uint32 Length )
```

This function get the information about CFSDADC firmware

### Parameters

<i>FirmwareVersion</i>	The pointer to the address that will contain the CFSDADC firmware build tag
<i>Length</i>	The length of the string that will contain the CFSDADC firmware build tag

### Returns

Return error code

#### 6.4.6.5 DSPSS\_ThreadConfigure()

```
DSPSS_StatusType DSPSS_ThreadConfigure (  
    const DSPSS_ThreadIdType Id,  
    const DSPSS_ThreadConfigType *const Config )
```

Configure the DSPSS thread

### Parameters

<i>Id</i>	Thread id
<i>Thread</i>	Thread configuration information
<i>SchedMode</i>	Scheduling mode

### Returns

Return error code

#### 6.4.6.6 DSPSS\_ThreadsInitialize()

```
DSPSS_StatusType DSPSS_ThreadsInitialize (  
    void )
```

Initialize the threads

### Returns

Return error code

#### 6.4.6.7 DSPSS\_Start()

```
DSPSS_StatusType DSPSS_Start (
    void )
```

Start running threads on DSPSS

Returns

Return error code

#### 6.4.6.8 DSPSS\_Stop()

```
DSPSS_StatusType DSPSS_Stop (
    void )
```

Stop running threads on DSPSS

Returns

Return error code

#### 6.4.6.9 DSPSS\_ThreadsCalibrationSet()

```
DSPSS_StatusType DSPSS_ThreadsCalibrationSet (
    const DSPSS_ThreadIdType Id,
    const DSPSS_CFSADC_CalibParamsType CalibrationParams )
```

Set calibration parameters

Parameters

<i>Id</i>	Thread id
<i>Calibration</i>	Calibration information

Returns

Return error code

### 6.4.6.10 DSPSS\_ThreadsStatusGet()

```
DSPSS_CFSDADC_ThreadStatusType DSPSS_ThreadsStatusGet (
    const DSPSS_ThreadIdType Id )
```

Get thread running status

Parameters

in	<i>Id</i>	Thread id
----	-----------	-----------

Returns

thread running status

### 6.4.6.11 DSPSS\_CoreBufferRead()

```
uint32 DSPSS_CoreBufferRead (
    const DSPSS_ThreadIdType Id,
    const uint32 Length,
    sint16 *const Data )
```

Read from DSP-Core buffer

Parameters

in	<i>Id</i>	Thread id
in	<i>Length</i>	The length of data array
in	<i>Value</i>	Pointer to the values read from DSP-Core buffer

Returns

The actual number of values read from the buffer

### 6.4.6.12 DSPSS\_FlushOutputBuffer()

```
void DSPSS_FlushOutputBuffer (
    const DSPSS_ThreadIdType Id,
    const DSPSS_TransferMethodType TransferMethod )
```

Flush output buffer to discard all the previous processed data.

Parameters

in	<i>Id</i>	Thread id
in	<i>TransferMethod</i>	The transfer type on output buffer (DMA or Core buffer)

Returns

void

**6.4.6.13 DSPSS\_DspStartResetRoutine()**

```
void DSPSS_DspStartResetRoutine (
    void )
```

Start reset routine

**6.4.6.14 DSPSS\_DspAssertReset()**

```
void DSPSS_DspAssertReset (
    void )
```

Assert reset

**6.4.6.15 DSPSS\_ThreadSchedulingConfigure()**

```
void DSPSS_ThreadSchedulingConfigure (
    DSPSS_SchedulingModeType SchedMode )
```

Program the scheduling sequence. Change of scheduling modes is possible only after the DSPSS reset.

Parameters

<i>SchedMode</i>	Scheduling mode
------------------	-----------------

**6.4.6.16 DSPSS\_ThreadSchedulingSampleThresholdMode()**

```
void DSPSS_ThreadSchedulingSampleThresholdMode (
    const DSPSS_NextReqHaltAckWaitModeType ContrlMode )
```

Program the scheduling to Sample Threshold mode. Change of scheduling modes is possible only after the DSPSS reset.

### Parameters

<i>ContrlMode</i>	Controlling mode of thread scheduling.
-------------------	--

#### 6.4.6.17 DSPSS\_ThreadSchedulingClockThresholdMode()

```
void DSPSS_ThreadSchedulingClockThresholdMode (
    const DSPSS_NextReqHaltAckWaitModeType ContrlMode,
    const DSPSS_ThreadIdType SchdSeq1,
    const DSPSS_ThreadIdType SchdSeq2,
    const DSPSS_ThreadIdType SchdSeq3,
    const DSPSS_ThreadIdType SchdSeq4 )
```

Program the scheduling to Clock Threshold mode. Change of scheduling modes is possible only after the DSPSS reset.

### Parameters

<i>ContrlMode</i>	Controlling mode of thread scheduling.
<i>SchdSeq1</i>	First thread that will be scheduled in clock threshold mode.
<i>SchdSeq2</i>	Second thread that will be scheduled in clock threshold mode.
<i>SchdSeq3</i>	Third thread that will be scheduled in clock threshold mode.
<i>SchdSeq4</i>	Fourth thread that will be scheduled in clock threshold mode.

#### 6.4.6.18 DSPSS\_ThreadPCInitSet()

```
void DSPSS_ThreadPCInitSet (
    const DSPSS_ThreadIdType Id,
    const uint16 PCInit )
```

Program the PC\_INIT values for every thread in Hardware scheduler of DSP sub-system. The PCINIT value for each of the thread should be in multiples of 8 PMEM locations.

### Parameters

<i>Id</i>	Thread id
<i>PCInit</i>	PC initial value for the threads of DSP-0

#### 6.4.6.19 DSPSS\_ThreadInputBufferConfigure()

```
void DSPSS_ThreadInputBufferConfigure (
    const DSPSS_ThreadIdType Id,
    const uint16 StartAddress,
    const uint16 EndAddress )
```

Program the start and end address memory location for respective threads in memory. Specifies the start and end addresses of the SDADC input buffer to write the SDADC data in the XMEM.

Parameters

<i>Id</i>	Thread id that will be configured.
<i>StartAddress</i>	Start address for the thread in data memory.
<i>EndAddress</i>	End address (byte address aligned) location for the thread in data memory.

#### 6.4.6.20 DSPSS\_ThreadThresholdSet()

```
void DSPSS_ThreadThresholdSet (
    const DSPSS_ThreadIdType Id,
    const uint16 Threshold )
```

Set thread threshold

Parameters

<i>Id</i>	Thread id that will be configured.
<i>Threshold</i>	The number of ADC samples in sample mode, after which the corresponding DSP thread is scheduled for operation or number of clock cycles for which a thread stays in operation in Clock-Cycle mode after which a new thread is scheduled.

#### 6.4.6.21 DSPSS\_ThreadEnable()

```
void DSPSS_ThreadEnable (
    const DSPSS_ThreadIdType Id )
```

Enable the thread.

Parameters

<i>Id</i>	Thread id.
-----------	------------

### 6.4.6.22 DSPSS\_ThreadSuspend()

```
void DSPSS_ThreadSuspend (
    const DSPSS_ThreadIdType Id )
```

Suspend the thread.

Parameters

<i>Id</i>	Thread id.
-----------	------------

### 6.4.6.23 DSPSS\_ThreadGetOutputBufferStart()

```
uint32 DSPSS_ThreadGetOutputBufferStart (
    const DSPSS_ThreadIdType Id )
```

Get the start address of output buffer.

Parameters

<i>Id</i>	Identifier of the thread.
-----------	---------------------------

Returns

Current start address of output buffer space inside XMEM.

### 6.4.6.24 DSPSS\_ThreadGetOutputBufferLength()

```
uint32 DSPSS_ThreadGetOutputBufferLength (
    const DSPSS_ThreadIdType Id )
```

Get the length in byte of output buffer.

Parameters

<i>Id</i>	Identifier of the thread.
-----------	---------------------------



Returns

length of the output buffer for the specific thread.

#### 6.4.6.25 DSPSS\_DspCoreBufferConfigure()

```
void DSPSS_DspCoreBufferConfigure (
    const DSPSS_ThreadIdType Id,
    const uint16 StartAddr,
    const uint16 EndAddr,
    const DSPSS_DspCoreBufferOwnerType Owner )
```

Configure the buffer address in XMEM by programming the start and end address and the read and write owners

Parameters

<i>Id</i>	Identifier of the thread.
<i>StartAddr</i>	Start address of the DSP_CORE circular buffer.
<i>EndAddr</i>	End address of the DSP_CORE circular buffer.
<i>Owner</i>	Read and write owners of the buffers.

#### 6.4.6.26 DSPSS\_DspCoreBufferEnable()

```
void DSPSS_DspCoreBufferEnable (
    const DSPSS_ThreadIdType Id )
```

Enable the DSP-Core buffer.

Parameters

<i>Id</i>	Identifier of the thread.
-----------	---------------------------

#### 6.4.6.27 DSPSS\_DspCoreBufferDisable()

```
void DSPSS_DspCoreBufferDisable (
    const DSPSS_ThreadIdType Id )
```

Disable the DSP-Core buffer.

### Parameters

<i>Id</i>	Identifier of the thread.
-----------	---------------------------

#### 6.4.6.28 DSPSS\_DspCoreBufferCurrReadPtrGet()

```
uint16 DSPSS_DspCoreBufferCurrReadPtrGet (  
    const DSPSS_ThreadIdType Id )
```

Get the current read pointer

### Parameters

<i>Id</i>	Identifier of the thread.
-----------	---------------------------

### Returns

Current read pointer value of DSP\_CORE buffer space inside XMEM.

#### 6.4.6.29 DSPSS\_DspCoreBufferOverflowCheck()

```
uint32 DSPSS_DspCoreBufferOverflowCheck (  
    const DSPSS_ThreadIdType Id )
```

Check overflow of write pointer inside DSP-CORE buffer

### Parameters

<i>Id</i>	Identifier of the thread.
-----------	---------------------------

### Returns

Overflow bit check.

#### 6.4.6.30 DSPSS\_DspCoreBufferOverflowClear()

```
void DSPSS_DspCoreBufferOverflowClear (  
    const DSPSS_ThreadIdType Id )
```

Clear overrrflow bit

Parameters

<i>Id</i>	Identifier of the thread.
-----------	---------------------------

#### 6.4.6.31 DSPSS\_InterruptEnable()

```
void DSPSS_InterruptEnable (
    const uint32 Id,
    const DSPSS_InterruptTypeType InterruptMask )
```

Enable an interrupt

Parameters

<i>Id</i>	Thread or dma id
<i>InterruptMask</i>	Type of the interrupt

#### 6.4.6.32 DSPSS\_InterruptDisable()

```
void DSPSS_InterruptDisable (
    const uint32 Id,
    const DSPSS_InterruptTypeType InterruptMask )
```

Disable an interrupt

Parameters

<i>Id</i>	Thread or dma id
<i>InterruptMask</i>	Type of the interrupt

#### 6.4.6.33 DSPSS\_DmaReadBufferConfigure()

```
void DSPSS_DmaReadBufferConfigure (
    const DSPSS_CFSDADC_DmaChannelType DmaChannel,
    const uint16 StartAddr,
    const uint16 EndAddr )
```

Configure the dma read buffer

Parameters

<i>DmaChannel</i>	Dma channel id
<i>StartAddr</i>	Start address of the DMA_READ circular buffer.
<i>EndAddr</i>	End address of the DMA_READ circular buffer.

### 6.4.6.34 DSPSS\_DmaReadBufferEnable()

```
void DSPSS_DmaReadBufferEnable (  
    const DSPSS_CFSDADC_DmaChannelType DmaChannel )
```

Enable the dma read buffer

Parameters

<i>DmaChannel</i>	Dma channel id
-------------------	----------------

### 6.4.6.35 DSPSS\_DmaReadBufferDisable()

```
void DSPSS_DmaReadBufferDisable (  
    const DSPSS_CFSDADC_DmaChannelType DmaChannel )
```

Disable the dma read buffer

Parameters

<i>DmaChannel</i>	Dma channel id
-------------------	----------------

### 6.4.6.36 DSPSS\_DmaReadBufferWrapPtrCheck()

```
boolean DSPSS_DmaReadBufferWrapPtrCheck (  
    const DSPSS_CFSDADC_DmaChannelType DmaChannel )
```

Check wrap status of the read pointer

Parameters

<i>DmaChannel</i>	Identifier of the DMA channel.
-------------------	--------------------------------

Returns

DMA-READ buffer wrap status check.

#### 6.4.6.37 Sdadc\_Ip\_Init()

```
Sdadc_Ip_StatusType Sdadc_Ip_Init (
    const uint32 Instance,
    const Sdadc_Ip_ConfigType *const Config )
```

Configures the hardware instance with the given configuration structure.

This function configures the SDADC hardware instance with the options provided in the structure. Module does not start by default after calling this function. The [Sdadc\\_Ip\\_ReloadConversion\(\)](#) or [Sdadc\\_Ip\\_SwTriggerConversion\(\)](#) function must be called to start conversions. The maximum input clock frequency for the SDADC is 40 MHz

Parameters

in	<i>Instance</i>	The SDADC instance number
in	<i>Config</i>	Configuration structure pointer

#### 6.4.6.38 Sdadc\_Ip\_Deinit()

```
void Sdadc_Ip_Deinit (
    const uint32 Instance )
```

Deinitialize SDADC module.

This function returns the SDADC internal registers to their reset values.

Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

#### 6.4.6.39 Sdadc\_Ip\_Powerup()

```
void Sdadc_Ip_Powerup (
    const uint32 Instance )
```

Power up the SDADC.

This function enables the SDADC block.

### Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

#### 6.4.6.40 Sdadc\_Ip\_Powerdown()

```
void Sdadc_Ip_Powerdown (
    const uint32 Instance )
```

Power down the SDADC.

This function disables the SDADC, SDADC internal modulator placed in low consumption mode. Note: This function also clears RSER[DFFDIRE] to ensure safe operation.

### Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

#### 6.4.6.41 Sdadc\_Ip\_ReloadConversion()

```
void Sdadc_Ip_ReloadConversion (
    const uint32 Instance )
```

Reload SDADC conversion.

This function resets SDADC internal modulator to start a fresh conversion. When the input trigger is disabled, this function must be call after changing converter configuration(gain, input channel, trigger, watchdog...).

### Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

#### 6.4.6.42 Sdadc\_Ip\_SwTriggerConversion()

```
void Sdadc_Ip_SwTriggerConversion (
    const uint32 Instance )
```

Generate software trigger.

This function generates the trigger event output which can be used for triggering conversions.

Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

#### 6.4.6.43 Sdadc\_Ip\_SetWdgThreshold()

```
void Sdadc_Ip_SetWdgThreshold (
    const uint32 Instance,
    const uint16 UpperThreshold,
    const uint16 LowerThreshold )
```

Configure watchdog threshold register.

This function configures the High/Low thresholds for the Watchdog monitor.

Parameters

in	<i>Instance</i>	The SDADC instance number
in	<i>UpperThreshold</i>	Watchdog upper threshold value
in	<i>LowerThreshold</i>	Watchdog lower threshold value

#### 6.4.6.44 Sdadc\_Ip\_EnableWatchdog()

```
void Sdadc_Ip_EnableWatchdog (
    const uint32 Instance )
```

Enables the watchdog monitor.

This function enables the Watchdog monitor.

Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

#### 6.4.6.45 Sdadc\_Ip\_DisableWatchdog()

```
void Sdadc_Ip_DisableWatchdog (
    const uint32 Instance )
```

Disables the watchdog monitor.

This function disables the Watchdog monitor.

Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

### 6.4.6.46 Sdadc\_Ip\_EnableWraparound()

```
void Sdadc_Ip_EnableWraparound (
    const uint32 Instance )
```

Enable Wrap Around mode.

This function enables the wraparound mechanism for conversion.

Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

### 6.4.6.47 Sdadc\_Ip\_DisableWraparound()

```
void Sdadc_Ip_DisableWraparound (
    const uint32 Instance )
```

Disable Wrap Around mode.

This function disables the wraparound mechanism for conversion.

Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

### 6.4.6.48 Sdadc\_Ip\_SetInputChannel()

```
void Sdadc_Ip_SetInputChannel (
    const uint32 Instance,
    const Sdadc_Ip_ChannelType Channel )
```



Select input analog channel.

This function configures the connectivity of analog inputs to either positive or negative polarity terminals of the SDADC. If wraparound mode is enabled, this function supports to configure initial entry value for the first loop of the wraparound sequence.

Parameters

in	<i>Instance</i>	The SDADC instance number
in	<i>Channel</i>	The input channel selection

#### 6.4.6.49 Sdadc\_Ip\_FlushFifo()

```
void Sdadc_Ip_FlushFifo (
    const uint32 Instance )
```

Flush data FIFO.

This function flushes data FIFO, all data in the FIFO will be erased.

Note:

- With DSPSS disabled: SDADC FIFO will be flushed.
- With DSPSS enabled : DSPSS output buffer will be flushed.

Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

#### 6.4.6.50 Sdadc\_Ip\_GetConvData()

```
uint32 Sdadc_Ip_GetConvData (
    const uint32 Instance,
    const uint32 Length,
    sint16 *const Data )
```

Gets the conversion data.

This function retrieves the converted data and writes it in the given data array. The data will be consecutive popped out the FIFO until the FIFO is empty or the data array is full, so the data array length should be big enough to contain all data.

Note:

- With DSPSS disabled: The data will be read from the SDADC hardware FIFO and they are calibrated automatically. In this case, please use the `Sdadc_Ip_GetRawConvDataFifo` function to get raw data (uncalibrated data).
- With DSPSS enabled : The data will be read from the DSP core buffer located in XMEM.

### Parameters

in	<i>Instance</i>	The SDADC instance number
in	<i>Length</i>	The length of data array
out	<i>Data</i>	The data array which contains the converted data

### Returns

The actual number of read values

#### 6.4.6.51 Sdadc\_Ip\_GetRawConvDataFifo()

```
uint32 Sdadc_Ip_GetRawConvDataFifo (  
    const uint32 Instance,  
    const uint32 Length,  
    sint16 *const Data )
```

Gets the raw conversion data in FIFO.

This function retrieves the raw (uncalibrated) converted data and writes it in the given data array. The data will be consecutive popped out the FIFO until the FIFO is empty or the data array is full, so the data array length should be big enough to contain all data.

### Parameters

in	<i>Instance</i>	The SDADC instance number
in	<i>Length</i>	The length of data array
out	<i>Data</i>	The data array which contains the converted data

### Returns

The actual number of values read from the FIFO

#### 6.4.6.52 Sdadc\_Ip\_GetStatusFlags()

```
uint32 Sdadc_Ip_GetStatusFlags (  
    const uint32 Instance )
```

Get the status flags.

This function returns the status flags of the SDADC. Bitwise AND the returned value with the SDADC\_IP\_FLAG\_ defines to get a specific status flag.

Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

Returns

The status flags

6.4.6.53 Sdadc\_Ip\_ClearStatusFlags()

```
void Sdadc_Ip_ClearStatusFlags (
    const uint32 Instance,
    const uint32 Mask )
```

Clear the status flags.

This function clears the status flags that are set to '1' in the mask. The mask input parameter can be set using SDADC\_IP\_FLAG\_ defines.

Parameters

in	<i>Instance</i>	The SDADC instance number
in	<i>Mask</i>	Bit-mask of flags to clear <ul style="list-style-type: none"><li>For example:<ul style="list-style-type: none"><li>With mask = SDADC_IP_FLAG_DATA_FIFO_FULL to clear Data FIFO Full Flag(DFFF).</li><li>With mask = (SDADC_IP_FLAG_DATA_FIFO_FULL   SDADC_IP_FLAG_DATA_FIFO_OVERRUN) to clear Data FIFO Full Flag(DFFF) and Data FIFO Overrun Flag(DFORF).</li></ul></li></ul>

6.4.6.54 Sdadc\_Ip\_EnableDmaEvents()

```
void Sdadc_Ip_EnableDmaEvents (
    const uint32 Instance,
    const uint32 EventsMask )
```

Enables SDADC DMA request generation.

This function enables SDADC DMA requests that are set to '1' in the EventsMask. The EventsMask input parameter can be set using SDADC\_IP\_EVENT\_ defines. Bitwise OR the macro defines to enable multiple DMA requests.

Parameters

in	<i>Instance</i>	The SDADC instance number
in	<i>EventsMask</i>	The bit-mask of SDADC events to enable DMA request generation

- For example: To enable DMA request generation of Data FIFO Full Event, the mask must be "SDADC\_IP\_EVENT\_FIFO\_FULL".

### 6.4.6.55 Sdadc\_Ip\_EnableInterruptEvents()

```
void Sdadc_Ip_EnableInterruptEvents (
    const uint32 Instance,
    const uint32 EventsMask )
```

Enables SDADC interrupt request generation.

This function enables SDADC interrupt requests that are set to '1' in the EventsMask. The EventsMask input parameter can be set using SDADC\_IP\_EVENT\_ defines. Bitwise OR the macro defines to enable multiple interrupt requests.

Parameters

in	<i>Instance</i>	The SDADC instance number
in	<i>EventsMask</i>	The bit-mask of SDADC events to enable interrupt request generation

- For example: To enable interrupt request generation of Data FIFO Full Event, the mask must be "SDADC\_IP\_EVENT\_FIFO\_FULL".

### 6.4.6.56 Sdadc\_Ip\_DisableInterruptEvents()

```
void Sdadc_Ip_DisableInterruptEvents (
    const uint32 Instance,
    const uint32 EventsMask )
```

Disable SDADC DMA and interrupt request generation.

This function disable SDADC DMA and interrupt requests that are set to '1' in the EventsMask. The EventsMask input parameter can be set using SDADC\_IP\_EVENT\_ defines. Bitwise OR the macro defines to disable multiple interrupt/DMA requests.

## Parameters

in	<i>Instance</i>	The SDADC instance number
in	<i>EventsMask</i>	The bit-mask of SDADC event to disable DMA and interrupt request generation

- For example: To disable DMA and interrupt request generation of Data FIFO Full Event, the mask must be "SDADC\_IP\_EVENT\_FIFO\_FULL".

**6.4.6.57 Sdadc\_Ip\_DoCalibration()**

```
Sdadc_Ip_StatusType Sdadc_Ip_DoCalibration (
    const uint32 Instance )
```

Perform Calibration of the SDADC.

This function performs a offset and gain calibration of the SDADC. Calibration should be done before using the SDADC module or after the operating conditions (particularly Vref, input gain) change significantly. The measured offset is going be used to nullify the offset error in the data conversion. The offset calibration must be performed for each input gain changing since it is expected to vary with input gain configuration of SDADC. The measured gain value is going be used to nullify the gain errors in the data conversion.

## Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

## Returns

the calibration result

- SDADC\_IP\_STATUS\_SUCCESS: calibration successful
- SDADC\_IP\_STATUS\_TIMEOUT: calibration step timed out
- SDADC\_IP\_STATUS\_ERROR: calibration failed

**6.4.6.58 Sdadc\_Ip\_ApplyCalibration()**

```
void Sdadc_Ip_ApplyCalibration (
    const uint32 Instance,
    const uint32 BufferLength,
    const sint16 *const UncalibratedBuffer,
    sint16 *const CalibratedBuffer )
```

Calibrate the converted data of the SDADC.

This function applies calibration to the uncalibrated converted data acquired by DMA or `Sdadc_Ip_GetRawConvertedDataFifo` function. In the case uncalibrated converted data gathered by DMA, this function should be called in the DMA callback.

## Parameters

in	<i>Instance</i>	The SDADC instance number
in	<i>BufferLength</i>	The number of entries in the uncalibrated data buffer.
in	<i>UncalibratedBuffer</i>	The pointer to the converted data array which contains uncalibrated data.
out	<i>CalibratedBuffer</i>	The pointer to the data array where the calibrated data will be stored.

**6.4.6.59 Sdadc\_Ip\_EnableHwTrigger()**

```
void Sdadc_Ip_EnableHwTrigger (
    const uint32 Instance )
```

Enable the hardware trigger.

This function enables the hardware trigger functionality

## Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

**6.4.6.60 Sdadc\_Ip\_DisableHwTrigger()**

```
void Sdadc_Ip_DisableHwTrigger (
    const uint32 Instance )
```

Disable the hardware trigger.

This function disables the hardware trigger functionality

## Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

**6.4.6.61 Sdadc\_Ip\_SetHwTrigger()**

```
void Sdadc_Ip_SetHwTrigger (
    const uint32 Instance,
```

```
const Sdadc_Ip_TriggerEdgeType TriggerEdge,
const Sdadc_Ip_TriggerSelectType TriggerSrc )
```

Setting the input hardware trigger.

This function configures the input used for hardware-triggered conversions.

Parameters

in	<i>Instance</i>	The SDADC instance number
in	<i>TriggerEdge</i>	The edge of the hardware trigger signal
in	<i>TriggerSrc</i>	The hardware trigger source to be selected

6.4.6.62 Sdadc\_Ip\_GetDataAddress()

```
uint32 Sdadc_Ip_GetDataAddress (
    const uint32 Instance )
```

Return the address of the converted data.

This function returns the address of the converted data which is used to configure DMA transfer.

- With DSPSS disabled: returns the address of SDADC FIFO.
- With DSPSS enabled : returns the address of DSPSS output buffer.

Parameters

in	<i>Instance</i>	The SDADC instance number
----	-----------------	---------------------------

Returns

The address where the converted data will be stored.

6.4.7 Variable Documentation

6.4.7.1 OptimizationLevel

```
DSPSS_CFSDADC_OptimizationLevelType OptimizationLevel
```

Optimization level for different filters

Definition at line 202 of file DSPSS\_Types.h.



#### 6.4.7.2 UpsamplingFactor

uint16 UpsamplingFactor

Upsampling factor: 1 (no upsampling) or 2

Definition at line 203 of file DSPSS\_Types.h.

#### 6.4.7.3 DownsamplingFactor

uint16 DownsamplingFactor

Downsampling factor

Definition at line 204 of file DSPSS\_Types.h.

#### 6.4.7.4 NbTaps

uint16 NbTaps

Nr of taps

Definition at line 205 of file DSPSS\_Types.h.

#### 6.4.7.5 Taps [1/2]

sint16 Taps[(128U)]

Taps array

Definition at line 206 of file DSPSS\_Types.h.

#### 6.4.7.6 Order

uint16 Order

Current usecase is 2nd order, biquad

Definition at line 215 of file DSPSS\_Types.h.

### 6.4.7.7 Taps [2/2]

```
sint16 Taps[ (5U) ]
```

Taps array

Definition at line 217 of file DSPSS\_Types.h.

### 6.4.7.8 Padd [1/3]

```
uint16 Padd
```

Padding for alignment

Definition at line 218 of file DSPSS\_Types.h.

### 6.4.7.9 UseCalibration

```
sint16 UseCalibration
```

Flag indicating the calibration is used or not

Definition at line 227 of file DSPSS\_Types.h.

### 6.4.7.10 Gain

```
sint16 Gain
```

Calibration gain

Definition at line 228 of file DSPSS\_Types.h.

### 6.4.7.11 Offset

```
sint32 Offset
```

Calibration offset

Definition at line 229 of file DSPSS\_Types.h.

#### 6.4.7.12 FirParams

`DSPSS_CFSADC_FirParamsType` FirParams

FIR filter configuration

Definition at line 238 of file DSPSS\_Types.h.

#### 6.4.7.13 IirParams

`DSPSS_CFSADC_IirParamsType` IirParams

IIR filter configuration

Definition at line 239 of file DSPSS\_Types.h.

#### 6.4.7.14 CalibParams

`DSPSS_CFSADC_CalibParamsType` CalibParams

Calibration configuration

Definition at line 240 of file DSPSS\_Types.h.

#### 6.4.7.15 NbSkippedSamples

`uint16` NbSkippedSamples

Number of sampled skipped for filter processing

Definition at line 241 of file DSPSS\_Types.h.

#### 6.4.7.16 Padd [2/3]

`uint16` Padd

Padding for alignment

Definition at line 242 of file DSPSS\_Types.h.

### 6.4.7.17 StandardParams

`DSPSS_CFSDADC_StandardParamsType` StandardParams

Standard parameters

Definition at line 251 of file DSPSS\_Types.h.

### 6.4.7.18 Status

`volatile DSPSS_CFSDADC_ThreadStatusType` Status

Thread status

Definition at line 260 of file DSPSS\_Types.h.

### 6.4.7.19 ConfigMessage

`volatile DSPSS_CFSDADC_ConfigMessageType` ConfigMessage

Thread message for parameter configuration

Definition at line 261 of file DSPSS\_Types.h.

### 6.4.7.20 Counter

`volatile uint16` Counter

Thread running counter

Definition at line 262 of file DSPSS\_Types.h.

### 6.4.7.21 InputDMAChannel

`DSPSS_CFSDADC_DmaChannelType` InputDMAChannel

Specifies usage of DMA channel for SDASC

Definition at line 264 of file DSPSS\_Types.h.

#### 6.4.7.22 OutputDMAChannel

DSPSS\_CFSDADC\_DmaChannelType OutputDMAChannel

Specifies usage of DMA channel or Core buffer on output

Definition at line 265 of file DSPSS\_Types.h.

#### 6.4.7.23 EntryPoint

uint16 EntryPoint

PC initial value for thread

Definition at line 267 of file DSPSS\_Types.h.

#### 6.4.7.24 Stack

uint16 Stack

Thread stack

Definition at line 268 of file DSPSS\_Types.h.

#### 6.4.7.25 InputThreshold [1/2]

uint16 InputThreshold

The number of samples used for thread threshold (to schedule a thread)

Definition at line 270 of file DSPSS\_Types.h.

#### 6.4.7.26 BufferInputStart

uint16 BufferInputStart

Start address for the thread in data memory.

Definition at line 271 of file DSPSS\_Types.h.

### 6.4.7.27 BufferInputLength

`uint16 BufferInputLength`

Size in bytes of the input buffer for the thread in data memory

Definition at line 272 of file DSPSS\_Types.h.

### 6.4.7.28 OutputThreshold [1/2]

`uint16 OutputThreshold`

The number of samples to be transferred/provided in the minor loop (to generate transfer request - OVF event)

Definition at line 274 of file DSPSS\_Types.h.

### 6.4.7.29 BufferOutputStart

`uint16 BufferOutputStart`

Start address for circular buffer used for filter output data

Definition at line 275 of file DSPSS\_Types.h.

### 6.4.7.30 BufferOutputLength

`uint16 BufferOutputLength`

Size in bytes of the circular buffer used for filter output data

Definition at line 276 of file DSPSS\_Types.h.

### 6.4.7.31 CoreMsgQueueStart

`uint16 CoreMsgQueueStart`

Start address for circular buffer used for DSP-HOST communication

Definition at line 278 of file DSPSS\_Types.h.

#### 6.4.7.32 CoreMsgQueueLength

uint16 CoreMsgQueueLength

Size in bytes of the circular buffer used for DSP-HOST communication

Definition at line 279 of file DSPSS\_Types.h.

#### 6.4.7.33 WorkAreaStart

uint16 WorkAreaStart

Area for filter state, scratch buffers

Definition at line 281 of file DSPSS\_Types.h.

#### 6.4.7.34 WorkAreaLength

uint16 WorkAreaLength

Size in bytes of the area for filter state, scratch buffers

Definition at line 282 of file DSPSS\_Types.h.

#### 6.4.7.35 Padd [3/3]

uint16 Padd

Padding for alignment

Definition at line 284 of file DSPSS\_Types.h.

#### 6.4.7.36 ThreadParams

[DSPSS\\_CFSADC\\_ThreadParamsType](#) ThreadParams

Thread parameters specific to each use-case

Definition at line 285 of file DSPSS\_Types.h.

### 6.4.7.37 SchedMode

DSPSS\_SchedulingModeType SchedMode

Thread scheduling mode

Definition at line 377 of file DSPSS\_Types.h.

### 6.4.7.38 ThreadThreshold

uint16 ThreadThreshold

Number of clock cycles for which a thread is scheduled (Clock Threshold Mode)

Definition at line 378 of file DSPSS\_Types.h.

### 6.4.7.39 InputThreshold [2/2]

uint16 InputThreshold

The number of samples used for thread threshold

Definition at line 379 of file DSPSS\_Types.h.

### 6.4.7.40 OutputThreshold [2/2]

uint16 OutputThreshold

The number of samples to be transferred/provided in the minor loop

Definition at line 380 of file DSPSS\_Types.h.

### 6.4.7.41 Owner

DSPSS\_DspCoreBufferOwnerType Owner

Read and write pointer owners of DSP-Core buffer

Definition at line 381 of file DSPSS\_Types.h.



#### 6.4.7.42 InputTransferMethod

DSPSS\_TransferMethodType InputTransferMethod

Specifies usage of DMA or SDADC transfer to the input buffer

Definition at line 382 of file DSPSS\_Types.h.

#### 6.4.7.43 OutputTransferMethod

DSPSS\_TransferMethodType OutputTransferMethod

Specifies usage of DMA or Core buffer transfer on output buffer

Definition at line 383 of file DSPSS\_Types.h.

**How to Reach Us:**

**Home Page:**

[nxp.com](http://nxp.com)

**Web Support:**

[nxp.com/support](http://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](http://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.

