

---

# User Manual

for S32K14X ADC Driver

Document Number: UM2ADCASR4.3 Rev0001R1.0.1  
Rev. 1.0





# Contents

Section number	Title	Page
<b>Chapter 1</b>		
<b>Revision History</b>		
<b>Chapter 2</b>		
<b>Introduction</b>		
2.1	Supported Derivatives.....	15
2.2	Overview.....	15
2.3	About this Manual.....	16
2.4	Acronyms and Definitions.....	16
2.5	Reference List.....	17
<b>Chapter 3</b>		
<b>Driver</b>		
3.1	Requirements.....	19
3.2	Driver Design Summary.....	19
3.3	Hardware resources.....	19
3.4	Deviation from Requirements.....	24
3.5	ADC Driver limitations.....	24
3.6	Driver usage and configuration tips.....	25
3.6.1	Starting a Software Triggered Group.....	27
3.6.2	Stopping a Software Group.....	27
3.6.3	Starting a Hardware Triggered Group.....	27
3.6.4	Stopping a Hardware Group.....	28
3.6.5	Retrieving the conversion results.....	28
3.6.6	Queuing and Priority feature.....	29
3.6.7	Limit checking feature.....	30
3.6.8	DET errors.....	31
3.6.9	DMA transfer.....	31
3.6.10	Calibration.....	32
3.6.11	Changing the Clock and Conversion Timing settings.....	32

Section number	Title	Page
3.6.12	Optimizations.....	33
3.6.12.1	Without Interrupts group.....	33
3.6.12.2	Optimize OneShot HwTrigger Conversions.....	33
3.6.12.3	Enable Double Buffering Optimization.....	34
3.6.12.4	Set Channel Optimization.....	35
3.6.12.5	Conversion Time Once.....	37
3.6.12.6	Bypass Consistency loop.....	38
3.6.13	Programmable Delay Block (PDB).....	39
3.6.13.1	AdcGroupInBacktoBackMode - Enable/Disable the channel conversions occurring in Back to Back mode.....	40
3.6.13.2	AdcGroupUsesChannelDelays - Enable/Disable the usage of channel delays in PDB pretriggers.....	41
3.6.13.3	AdcDelayNextPdb – configuring the delay between PDB channel conversions.....	42
3.6.13.4	PDB Unit settings.....	42
3.7	Runtime Errors.....	43
3.8	Software specification.....	43
3.8.1	Define Reference.....	43
3.8.1.1	Define ADC_VENDOR_ID.....	44
3.8.1.2	Define ADC_MODULE_ID.....	44
3.8.1.3	Define ADC_AR_RELEASE_MAJOR_VERSION.....	44
3.8.1.4	Define ADC_AR_RELEASE_MINOR_VERSION.....	44
3.8.1.5	Define ADC_AR_RELEASE_REVISION_VERSION.....	44
3.8.1.6	Define ADC_SW_MAJOR_VERSION.....	45
3.8.1.7	Define ADC_SW_MINOR_VERSION.....	45
3.8.1.8	Define ADC_SW_PATCH_VERSION.....	45
3.8.1.9	Define ADC_E_UNINIT.....	45
3.8.1.10	Define ADC_E_BUSY.....	46
3.8.1.11	Define ADC_E_IDLE.....	46
3.8.1.12	Define ADC_E_ALREADY_INITIALIZED.....	46
3.8.1.13	Define ADC_E_PARAM_CONFIG.....	46

Section number	Title	Page
3.8.1.14	Define ADC_E_PARAM_POINTER.....	47
3.8.1.15	Define ADC_E_PARAM_GROUP.....	47
3.8.1.16	Define ADC_E_WRONG_CONV_MODE.....	47
3.8.1.17	Define ADC_E_WRONG_TRIGG_SRC.....	47
3.8.1.18	Define ADC_E_NOTIF_CAPABILITY.....	48
3.8.1.19	Define ADC_E_BUFFER_UNINIT.....	48
3.8.1.20	Define ADC_E_NOT_DISENGAGED.....	48
3.8.1.21	Define ADC_E_QUEUE_FULL.....	48
3.8.1.22	Define ADC_E_PARAM_UNIT.....	49
3.8.1.23	Define ADC_E_INVALID_CLOCK_MODE.....	49
3.8.1.24	Define ADC_E_PARAM_CHANNEL.....	49
3.8.1.25	Define ADC_E_BUFFER_UNINIT_LIST.....	49
3.8.1.26	Define ADC_E_WRONG_TRIGG_SRC_LIST.....	50
3.8.1.27	Define ADC_E_QUEUE_FULL_LIST.....	50
3.8.1.28	Define ADC_E_WRONG_CONV_MODE_LIST.....	50
3.8.1.29	Define ADC_E_SET_MODE_LIST.....	50
3.8.1.30	Define ADC_INIT_ID.....	50
3.8.1.31	Define ADC_DEINIT_ID.....	51
3.8.1.32	Define ADC_STARTGROUPCONVERSION_ID.....	51
3.8.1.33	Define ADC_STOPGROUPCONVERSION_ID.....	51
3.8.1.34	Define ADC_VALUEREADGROUP_ID.....	52
3.8.1.35	Define ADC_ENABLEHARDWARETRIGGER_ID.....	52
3.8.1.36	Define ADC_DISABLEHARDWARETRIGGER_ID.....	52
3.8.1.37	Define ADC_ENABLEGROUPNOTIFICATION_ID.....	52
3.8.1.38	Define ADC_DISABLEGROUPNOTIFICATION_ID.....	53
3.8.1.39	Define ADC_GETGROUPSTATUS_ID.....	53
3.8.1.40	Define ADC_GETVERSIONINFO_ID.....	53
3.8.1.41	Define ADC_GETSTREAMLASTPOINTER_ID.....	53
3.8.1.42	Define ADC_SETUPRESULTBUFFER_ID.....	54

Section number	Title	Page
3.8.1.43	Define ADC_SETCLOCKMODE_ID.....	54
3.8.1.44	Define ADC_CALIBRATE_ID.....	54
3.8.1.45	Define ADC_SETCHANNEL_ID.....	54
3.8.2	Enum Reference.....	54
3.8.2.1	Enumeration Adc_GlobalStateType.....	55
3.8.2.2	Enumeration Adc_DualClockModeType.....	55
3.8.2.3	Enumeration Adc_GroupConversionStateType.....	55
3.8.2.4	Enumeration Adc_GroupAccessModeType.....	56
3.8.2.5	Enumeration Adc_GroupConvType.....	56
3.8.2.6	Enumeration Adc_GroupConvModeType.....	57
3.8.2.7	Enumeration Adc_GroupReplacementType.....	57
3.8.2.8	Enumeration Adc_StreamBufferModeType.....	57
3.8.2.9	Enumeration Adc_StatusType.....	58
3.8.2.10	Enumeration Adc_NotificationType.....	58
3.8.2.11	Enumeration Adc_HwTriggerSignalType.....	59
3.8.2.12	Enumeration Adc_TriggerSourceType.....	59
3.8.2.13	Enumeration Adc_HwTriggeringType.....	59
3.8.2.14	Enumeration Adc_ChannelRangeSelectType.....	60
3.8.3	Function Reference.....	60
3.8.3.1	Function Adc_Init.....	61
3.8.3.2	Function Adc_DeInit.....	61
3.8.3.3	Function Adc_SetupResultBuffer.....	62
3.8.3.4	Function Adc_EnableGroupNotification.....	63
3.8.3.5	Function Adc_DisableGroupNotification.....	63
3.8.3.6	Function Adc_EnableHardwareTrigger.....	64
3.8.3.7	Function Adc_DisableHardwareTrigger.....	65
3.8.3.8	Function Adc_StartGroupConversion.....	65
3.8.3.9	Function Adc_StopGroupConversion.....	66
3.8.3.10	Function Adc_GetGroupStatus.....	67

Section number	Title	Page
3.8.3.11	Function Adc_GetStreamLastPointer.....	67
3.8.3.12	Function Adc_GetVersionInfo.....	68
3.8.3.13	Function Adc_Calibrate.....	69
3.8.3.14	Function Adc_SetClockMode.....	70
3.8.3.15	Function Adc_SetChannel.....	70
3.8.3.16	Function Adc_ReadGroup.....	71
3.8.4	Structs Reference.....	72
3.8.4.1	Structure Adc_CalibrationStatusType.....	72
3.8.4.2	Structure Adc_ConfigType.....	73
3.8.4.3	Structure Adc_UnitStatusType.....	73
3.8.4.4	Structure Adc_GroupStatusType.....	74
3.8.4.5	Structure Adc_GroupConfigurationType.....	75
3.8.4.6	Structure Adc_RuntimeGroupChannelType.....	76
3.8.4.7	Structure Adc_ChannelLimitCheckingType.....	77
3.8.4.8	Structure Adc_ValidationResultType.....	77
3.8.5	Types Reference.....	78
3.9	Symbolic Names Disclaimer.....	78

## Chapter 4 Tresos Configuration Plug-in

4.1	Configuration elements of Adc.....	79
4.2	Form IMPLEMENTATION_CONFIG_VARIANT.....	79
4.3	Form AdcGeneral.....	80
4.3.1	AdcDeInitApi (AdcGeneral).....	80
4.3.2	AdcDevErrorDetect (AdcGeneral).....	80
4.3.3	AdcEnableLimitCheck (AdcGeneral).....	81
4.3.4	AdcEnableQueuing (AdcGeneral).....	81
4.3.5	AdcPriorityQueueMaxDepth (AdcGeneral).....	82
4.3.6	AdcEnableStartStopGroupApi (AdcGeneral).....	82
4.3.7	AdcGrpNotifCapability (AdcGeneral).....	82

Section number	Title	Page
4.3.8	AdcHwTriggerApi (AdcGeneral).....	83
4.3.9	AdcPriorityImplementation (AdcGeneral).....	83
4.3.10	AdcReadGroupApi (AdcGeneral).....	83
4.3.11	AdcResultAlignment (AdcGeneral).....	84
4.3.12	AdcVersionInfoApi (AdcGeneral).....	84
4.3.13	AdcTimeout (AdcGeneral).....	85
4.3.14	AdcLowPowerStatesSupport (AdcGeneral).....	85
4.3.15	AdcPowerStateAsynchTransitionMode (AdcGeneral).....	85
4.3.16	Form AdcPowerStateConfig.....	86
4.3.16.1	AdcPowerState (AdcPowerStateConfig).....	86
4.3.16.2	AdcPowerStateReadyCbkJRef (AdcPowerStateReadyCbkJRefConfig).....	87
4.4	Form AdcPublishedInformation.....	87
4.4.1	AdcChannelValueSigned (AdcPublishedInformation).....	88
4.4.2	AdcGroupFirstChannelFixed (AdcPublishedInformation).....	88
4.4.3	AdcMaxChannelResolution (AdcPublishedInformation).....	88
4.5	Form NonAutosar.....	89
4.5.1	AdcEnableGroupDependentChannelNames (NonAutosar).....	89
4.5.2	AdcEnableCalibration (NonAutosar).....	90
4.5.3	AdcEnableInitialNotification (NonAutosar).....	90
4.5.4	AdcConvTimeOnce (NonAutosar).....	91
4.5.5	AdcEnableDoubleBufferingOptimization (NonAutosar).....	91
4.5.6	AdcEnableDualClockMode (NonAutosar).....	92
4.5.7	AdcEnableSetChannel (NonAutosar).....	92
4.5.8	AdcDisableDemReportErrorStatus (NonAutosar).....	93
4.5.9	AdcOptimizeOneShotHwTriggerConversions (NonAutosar).....	93
4.5.10	AdcEnableDmaTrasferMode (NonAutosar).....	94
4.5.11	AdcEnableUserModeSupport (NonAutosar).....	94
4.5.12	AdcContinuousWithoutInterrupt (NonAutosar).....	94
4.5.13	AdcUseHardwareNormalGroups(NonAutosar).....	95



Section number	Title	Page
4.5.14	AdcBypassConsistencyLoop(NonAutosar).....	95
4.6	Form AdcDemEventParameterRefs.....	96
4.6.1	ADC_E_TIMEOUT (AdcDemEventParameterRefs).....	96
4.7	Form AdcInterrupt.....	96
4.7.1	AdcInterruptSource (AdcInterrupt).....	97
4.7.2	AdcInterruptEnable (AdcInterrupt).....	97
4.8	Form AdcConfigSet.....	98
4.8.1	Form AdcHwUnit.....	98
4.8.1.1	AdcTransferType (AdcHwUnit).....	99
4.8.1.2	AdcClockSource (AdcHwUnit).....	99
4.8.1.3	AdcHwUnitId (AdcHwUnit).....	100
4.8.1.4	AdcLogicalUnitId (AdcHwUnit).....	100
4.8.1.5	AdcVoltageReferenceSelection (AdcHwUnit).....	101
4.8.1.6	AdcPrescale (AdcHwUnit).....	101
4.8.1.7	AdcResolution (AdcHwUnit).....	102
4.8.1.8	AdcOffsetCorrectionValue (AdcHwUnit).....	102
4.8.1.9	Form AdcChannel.....	102
4.8.1.9.1	AdcLogicalChannelId (AdcChannel).....	103
4.8.1.9.2	AdcChannelId (AdcChannel).....	103
4.8.1.9.3	AdcChannelLimitCheck (AdcChannel).....	104
4.8.1.9.4	AdcChannelHighLimit (AdcChannel).....	104
4.8.1.9.5	AdcChannelLowLimit (AdcChannel).....	104
4.8.1.9.6	AdcChannelRangeSelect (AdcChannel).....	105
4.8.1.9.7	AdcChannelConvTime (AdcChannel).....	105
4.8.1.9.8	AdcChannelRefVoltsrcHigh (AdcChannel).....	106
4.8.1.9.9	AdcChannelRefVoltsrcLow (AdcChannel).....	106
4.8.1.9.10	AdcChannelResolution (AdcChannel).....	107
4.8.1.9.11	AdcChannelSampTime (AdcChannel).....	107
4.8.1.10	Form AdcGroup.....	107

Section number	Title	Page
4.8.1.10.1	AdcGroupAccessMode (AdcGroup).....	108
4.8.1.10.2	AdcGroupConversionMode (AdcGroup).....	108
4.8.1.10.3	AdcGroupConversionType (AdcGroup).....	109
4.8.1.10.4	AdcGroupId (AdcGroup).....	109
4.8.1.10.5	AdcGroupPriority (AdcGroup).....	110
4.8.1.10.6	AdcGroupReplacement (AdcGroup).....	110
4.8.1.10.7	AdcGroupTriggSrc (AdcGroup).....	110
4.8.1.10.8	AdcHwTrigSignal (AdcGroup).....	111
4.8.1.10.9	AdcHwTrigTimer (AdcGroup).....	111
4.8.1.10.10	AdcNotification (AdcGroup).....	112
4.8.1.10.11	AdcExtraNotification (AdcGroup).....	112
4.8.1.10.12	AdcStreamingBufferMode (AdcGroup).....	112
4.8.1.10.13	AdcEnableDoubleBuffering (AdcGroup).....	113
4.8.1.10.14	AdcEnableHwInterrupt (AdcGroup).....	113
4.8.1.10.15	AdcStreamingNumSamples (AdcGroup).....	114
4.8.1.10.16	AdcWithoutInterrupts (AdcGroup).....	114
4.8.1.10.17	AdcGroupInBacktoBackMode (AdcGroup).....	114
4.8.1.10.18	AdcGroupUsesChannelDelays (AdcGroup).....	115
4.8.1.10.19	AdcDelayNextPdb (AdcGroup).....	115
4.8.1.10.20	AdcPdbPeriodContinuousMode (AdcGroup).....	116
4.8.1.11	Form AdcGroupNormalConversionTimings.....	117
4.8.1.11.1	AdcGroupHardwareAverageEnable (AdcGroupNormalConversionTimings).....	117
4.8.1.11.2	AdcGroupHardwareAverageSelect (AdcGroupNormalConversionTimings).....	118
4.8.1.11.3	AdcGroupSampleTimeDuration (AdcGroupNormalConversionTimings).....	118
4.8.1.11.4	AdcGroupClockDivideSelect (AdcGroupNormalConversionTimings).....	119
4.8.1.12	Form AdcGroupDefinition.....	119
4.8.1.13	Form AdcChannelDelay.....	120
4.8.1.14	Form AdcPdbSettings.....	120
4.8.1.14.1	AdcPdbPrescalerDividerSelect (AdcPdbSettings).....	121

Section number	Title	Page
4.8.1.14.2	AdcPdbMultiplicationFactorSelect (AdcPdbSettings).....	122
4.8.1.14.3	AdcPdbChannelSequenceErrorEnable (AdcPdbSettings).....	122
4.8.1.14.4	AdcPdbErrorNotification (AdcPdbSettings).....	122
4.8.1.15	Form AdcNormalConversionTimings.....	123
4.8.1.15.1	AdcGroupHardwareAverageEnable (AdcNormalConversionTimings).....	123
4.8.1.15.2	AdcGroupHardwareAverageSelect (AdcNormalConversionTimings).....	124
4.8.1.15.3	AdcGroupSampleTimeDuration (AdcNormalConversionTimings).....	124
4.8.1.15.4	AdcGroupClockDivideSelect (AdcNormalConversionTimings).....	124
4.8.1.16	Form AdcAlternateConversionTimings.....	125
4.8.1.16.1	AdcHardwareAverageEnableAlternate (AdcAlternateConversionTimings).....	125
4.8.1.16.2	AdcHardwareAverageSelectAlternate (AdcAlternateConversionTimings).....	126
4.8.1.16.3	AdcSampleTimeDurationAlternate (AdcAlternateConversionTimings).....	126
4.8.1.16.4	AdcClockDivideSelectAlternate (AdcAlternateConversionTimings).....	127
4.9	Form CommonPublishedInformation.....	127
4.9.1	ArReleaseMajorVersion (CommonPublishedInformation).....	128
4.9.2	ArReleaseMinorVersion (CommonPublishedInformation).....	128
4.9.3	ArReleaseRevisionVersion (CommonPublishedInformation).....	129
4.9.4	ModuleId (CommonPublishedInformation).....	129
4.9.5	SwMajorVersion (CommonPublishedInformation).....	130
4.9.6	SwMinorVersion (CommonPublishedInformation).....	130
4.9.7	SwPatchVersion (CommonPublishedInformation).....	131
4.9.8	VendorApiInfix (CommonPublishedInformation).....	131
4.9.9	VendorId (CommonPublishedInformation).....	132
4.10	Configuration element of Mcl (a dependent module).....	132
4.11	Form MclTriggerMux.....	132
4.11.1	TrgMux ADC_0 Input0.....	133
4.11.2	TrgMux ADC_0 Input1.....	133
4.11.3	TrgMux ADC_0 Input2.....	134
4.11.4	TrgMux ADC_0 Input3.....	134

Section number	Title	Page
4.11.5	TrgMux ADC_0 Lock Enabled.....	134
4.11.6	TrgMux ADC_1 Input0.....	135
4.11.7	TrgMux ADC_1 Input1.....	135
4.11.8	TrgMux ADC_1 Input2.....	135
4.11.9	TrgMux ADC_1 Input3.....	136
4.11.10	TrgMux ADC_1 Lock Enabled.....	136
4.11.11	TrgMux PDB0 Input0.....	136
4.11.12	TrgMux PDB0 Lock Enabled.....	137
4.11.13	TrgMux PDB1 Input0.....	137
4.11.14	TrgMux PDB1 Lock Enabled.....	137
4.12	Configuration element of Mcu (a dependent module).....	138
4.13	Form McuSIMConfig.....	138
4.13.1	Form McuChipControlConfiguration.....	138
4.13.1.1	McuEnableAdcSupplyMonitoring (McuChipControlConfiguration).....	139
4.13.1.2	McuAdcSupply (McuChipControlConfiguration).....	139
4.13.1.3	McuPDBBackToBackSelect (McuChipControlConfiguration).....	140
4.13.1.4	McuPTB14InterleaveChannelSelect (McuChipControlConfiguration).....	140
4.13.1.5	McuPTB13InterleaveChannelSelect (McuChipControlConfiguration).....	141
4.13.1.6	McuPTB1InterleaveChannelSelect (McuChipControlConfiguration).....	141
4.13.1.7	McuPTB0InterleaveChannelSelect (McuChipControlConfiguration).....	141
4.13.2	Form McuAdcOptionsConfiguration.....	142
4.13.2.1	McuADC1PreTrigeerSourceSelect (McuAdcOptionsConfiguration).....	142
4.13.2.2	McuADC1SoftwarePreTrigeerSourceSelect (McuAdcOptionsConfiguration).....	143
4.13.2.3	McuADC1TrigeerSourceSelect (McuAdcOptionsConfiguration).....	143
4.13.2.4	McuADC0PreTrigeerSourceSelect (McuAdcOptionsConfiguration).....	144
4.13.2.5	McuADC0SoftwarePreTrigeerSourceSelect (McuAdcOptionsConfiguration).....	144
4.13.2.6	McuADC0TrigeerSourceSelect (McuAdcOptionsConfiguration).....	145
4.13.2.7	McuSoftwareTriggerToTRGMUX (McuAdcOptionsConfiguration).....	145

# Chapter 1

## Revision History

**Table 1-1. Revision History**

Revision	Date	Author	Description
1.0	21/06/2019	NXP MCAL Team	Updated version for ASR 4.3.1S32K14XR1.0.1



# Chapter 2

## Introduction

This User Manual describes NXP Semiconductors AUTOSAR Analog to Digital Converter ( Adc ) for S32K14X .

AUTOSAR Adc driver configuration parameters and deviations from the specification are described in Adc 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 .

**Table 2-1. S32K14X Derivatives**

NXP Semiconductors	s32k148_lqfp144, s32k148_lqfp176, s32k148_mapbga100, s32k146_lqfp144, s32k146_lqfp100, s32k146_lqfp64, s32k146_mapbga100, s32k144_lqfp100, s32k144_lqfp64, s32k144_mapbga100, s32k142_lqfp100, s32k142_lqfp64, s32k118_lqfp48, s32k118_lqfp64, s32k142_lqfp48, s32k144_lqfp48, s32k148_lqfp100
--------------------	--

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

### 2.2 Overview

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

## AUTOSAR

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

## 2.3 About this Manual

This Technical Reference employs the following typographical conventions:

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

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

Notes and warnings are shown as below:

### Note

This is a note.

## 2.4 Acronyms and Definitions

**Table 2-2. Acronyms and Definitions**

Term	Definition
ADC	Analog to Digital Converter
API	Application Programming Interface
ASM	Assembler
AUTOSAR	Automotive Open System Architecture
BSMI	Basic Software Make file Interface
CAN	Controller Area Network
C/CPP	C and C++ Source Code
CS	Chip Select
CTU	Cross Trigger Unit

*Table continues on the next page...*



**Table 2-2. Acronyms and Definitions (continued)**

Term	Definition
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

**Table 2-3. Reference List**

#	Title	Version
1	Specification of Adc Driver	AUTOSAR Release 4.3.1
2	S32K14X Reference Manual	Reference Manual, Rev. 9, 9/2018
3	S32K142 Mask Set Errata for Mask 0N33V (0N33V)	30/11/2017
4	S32K144 Mask Set Errata for Mask 0N57U (0N57U)	30/11/2017
5	S32K146 Mask Set Errata for Mask 0N73V (0N73V)	30/11/2017
6	S32K148 Mask Set Errata for Mask 0N20V (0N20V)	25/10/2018
7	S32K118 Mask Set Errata for Mask 0N97V (0N97V)	07/01/2019



## Chapter 3 Driver

### 3.1 Requirements

Requirements for this driver are detailed in the AUTOSAR 4.3 Rev0001Adc Driver Software Specification document (See Table [Reference List](#) ).

### 3.2 Driver Design Summary

The ADC Driver initializes and controls the internal Analogue 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 analogue 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

- **S32K118 device families at 48 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **S32K118 device families at 64 pins:**

- **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **S32K142 device families at 48 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
  - **Adc Physical Channels for ADC HW Unit 1:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_8,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **S32K142 device families at 64 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
  - **Adc Physical Channels for ADC HW Unit 1:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_10,AN\_11,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **S32K142 device families at 100 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
  - **Adc Physical Channels for ADC HW Unit 1:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **S32K144 device families at 48 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
  - **Adc Physical Channels for ADC HW Unit 1:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_8,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **S32K144 device families at 64 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
  - **Adc Physical Channels for ADC HW Unit 1:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_10,AN\_11,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **S32K144 device families at 100 pins:**

- **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **Adc Physical Channels for ADC HW Unit 1:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **S32K146 device at 64 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
  - **Adc Physical Channels for ADC HW Unit 1:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_10,AN\_11,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **S32K146 device at 100 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
  - **Adc Physical Channels for ADC HW Unit 1:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **S32K146 device at 144 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,AN\_16,AN\_17,AN\_18,AN\_19,AN\_20,AN\_21,AN\_22,AN\_23,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
  - **Adc Physical Channels for ADC HW Unit 1:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,AN\_16,AN\_17,AN\_18,AN\_19,AN\_20,AN\_21,AN\_22,AN\_23,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **S32K148 device at 100 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
  - **Adc Physical Channels for ADC HW Unit 1:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL
- **S32K148 device at 144 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:**  
AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,INT\_AN\_0,BAND\_GAP,VREFH,VREFL

11,AN\_12,AN\_13,AN\_14,AN\_15,AN\_16,AN\_17,AN\_18,AN\_19,AN\_20,AN\_21,AN\_22,AN\_23,AN\_24,AN\_25,AN\_26,AN\_27,AN\_28,AN\_29,AN\_30,AN\_31,INT\_AN\_0,BAND\_GAP,VREFH,VREFL

- **Adc Physical Channels for ADC HW Unit 1:**

AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,AN\_16,AN\_17,AN\_18,AN\_19,AN\_20,AN\_21,AN\_22,AN\_23,AN\_24,AN\_25,AN\_26,AN\_27,AN\_28,AN\_29,AN\_30,AN\_31,INT\_AN\_0,BAND\_GAP,VREFH,VREFL

- **S32K148 device at 176 pins:**

- **Adc Physical Channels for ADC HW Unit 0:**

AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,AN\_16,AN\_17,AN\_18,AN\_19,AN\_20,AN\_21,AN\_22,AN\_23,AN\_24,AN\_25,AN\_26,AN\_27,AN\_28,AN\_29,AN\_30,AN\_31,INT\_AN\_0,BAND\_GAP,VREFH,VREFL

- **Adc Physical Channels for ADC HW Unit 1:**

AN\_0,AN\_1,AN\_2,AN\_3,AN\_4,AN\_5,AN\_6,AN\_7,AN\_8,AN\_9,AN\_10,AN\_11,AN\_12,AN\_13,AN\_14,AN\_15,AN\_16,AN\_17,AN\_18,AN\_19,AN\_20,AN\_21,AN\_22,AN\_23,AN\_24,AN\_25,AN\_26,AN\_27,AN\_28,AN\_29,AN\_30,AN\_31,INT\_AN\_0,BAND\_GAP,VREFH,VREFL

The following picture in Reference Manual - chapter **42.4.2 ADC Status and Control Register 1 (SC1A - aSC1P)** provides ADC pin channel mapping details.

5-0 ADCH	<p>Input channel select</p> <p>Selects one of the input channels.</p> <p><b>NOTE:</b> Some of the input channel options in the bitfield-setting descriptions might not be available for your chip. For the actual ADC channel assignments for your device, see the chip-specific information.</p> <p>The successive approximation converter subsystem is turned off when the channel bits are all set (i.e. ADCH set to all 1s). This feature allows explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional single conversion from being performed. It is not necessary to set ADCH to all 1s to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.</p> <p>000000b - External channel 0 is selected as input.  000001b - External channel 1 is selected as input.  000010b - External channel 2 is selected as input.  000011b - External channel 3 is selected as input.  000100b - External channel 4 is selected as input.  000101b - External channel 5 is selected as input.  000110b - External channel 6 is selected as input.  000111b - External channel 7 is selected as input.  001000b - External channel 8 is selected as input.  001001b - External channel 9 is selected as input.  001010b - External channel 10 is selected as input.  001011b - External channel 11 is selected as input.  001100b - External channel 12 is selected as input.  001101b - External channel 13 is selected as input.  001110b - External channel 14 is selected as input.  001111b - External channel 15 is selected as input.  010000b - Reserved  010001b - Reserved  010010b - Reserved  010011b - Reserved  010100b - Reserved  010101b - Internal channel 0 is selected as input.  010110b - Internal channel 1 is selected as input.  010111b - Internal channel 2 is selected as input.  011000b - Reserved  011001b - Reserved  011010b - Reserved  011011b - Band Gap  011100b - Internal channel 3 is selected as input.  011101b - <math>V_{REFSH}</math> is selected as input. Voltage reference selected is determined by SC2[REFSEL].  011110b - <math>V_{REFSL}</math> is selected as input. Voltage reference selected is determined by SC2[REFSEL].  011111b - Reserved  100000b - External channel 16 is selected as input.  100001b - External channel 17 is selected as input.  100010b - External channel 18 is selected as input.  100011b - External channel 19 is selected as input.  100100b - External channel 20 is selected as input.  100101b - External channel 21 is selected as input.  100110b - External channel 22 is selected as input.  100111b - External channel 23 is selected as input.  101000b - External channel 24 is selected as input.  101001b - External channel 25 is selected as input.  101010b - External channel 26 is selected as input.  101011b - External channel 27 is selected as input.  101100b - External channel 28 is selected as input.  101101b - External channel 29 is selected as input.  101110b - External channel 30 is selected as input.  101111b - External channel 31 is selected as input.  110000b - Reserved  110001b - Reserved  110010b - Reserved  110011b - Reserved  110100b - Reserved  110101b - Reserved  110110b - Reserved  110111b - Reserved  111000b - Reserved  111001b - Reserved  111010b - Reserved  111011b - Reserved  111100b - Reserved  111101b - Reserved  111110b - Reserved  111111b - Module is disabled</p>
-------------	---

Figure 3-1. ADC\_SC1n.ADCH field descriptions

- Direct hardware triggering

- ADC supports normal hardware triggering directly from PDB. Hardware trigger source must be configured in Mcl.

### 3.4 Deviation from Requirements

The driver deviates from the AUTOSAR Adc Driver software specification in some places. The table identifies the AUTOSAR requirements that are not fully implemented, implemented differently, or out of scope for the Adc Driver. Table [Table 3-1](#) provides Status column description.

**Table 3-1. Deviations Status Column Description**

Term	Definition
N/A	Not available
N/T	Not testable
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 driver.

**Table 3-2. ADC Deviations Table**

Requirement	Status	Description	Notes
SWS_Adc_00460	N/A	These requirements are not applicable to this specification.	Not a requirement.

### 3.5 ADC Driver limitations

Hardware triggered groups and software triggered groups cannot function in parallel - not supported by ADC hardware on S32K1XX.

The Adc hardware unit id should be the same for one logical unit id in all variants to ensure that the data for channel limit checking configuration is generated correctly.

If the channels assignment are different between variants, the symbolic names will be inconsistent.

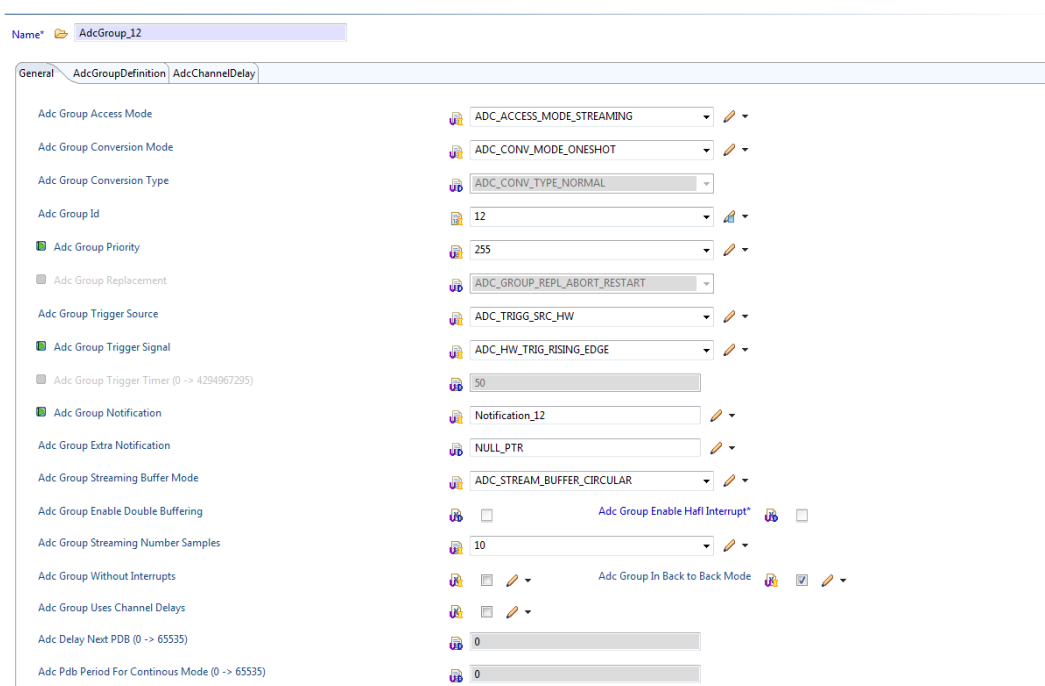


Buffer will be overflowed if PDB trigger period between each channel is too short. Minimum required period is depending on each derivative. It is recommended to use default delay period.

## 3.6 Driver usage and configuration tips

All Adc driver APIs use Adc Groups as parameters. An Adc Group is a collection of one or more Adc Hardware channels that is defined at configuration time. All the channels in a Group will be converted when used with the standard Adc APIs.

The Group configuration is described in below picture:



The parameter Adc Group ID will be generated as symbolic name and can be used as parameter of all Group – handling Adc APIs.

### Some general prerequisite APIs need to be invoked prior to any conversion:

\* **Adc\_Init:** The function *Adc\_Init* shall initialize the ADC hardware units and driver according to the configuration set referenced by *ConfigPtr*.

Prototype: `void Adc_Init(const Adc_ConfigType *ConfigPtr);`

\* **Adc\_SetupResultBuffer:** The function *Adc\_SetupResultBuffer* shall initialize the result buffer pointer of the selected Group with the address value passed as parameter.

**Prototype:** `Std_ReturnType Adc_SetupResultBuffer(Adc_GroupType Group, Adc_ValueGroupType *DataBufferPtr);`

The user has to provide application result buffers for the ADC group results. One buffer is required for each group. The buffer size depends on the number of group channels, the group access mode and from the number of streaming samples, if streaming access mode is selected. Before starting a group conversion, the user has to initialize the group result pointer using API function *Adc\_SetupResultBuffer* which initializes the group result pointer to point to the specified application result buffer.

Refer to chapter *ADC Buffer Access Mode Example* in **AUTOSAR\_SWS\_ADCDriver** to see more details.

### **Optional standard Autosar APIs:**

\* **Adc\_EnableGroupNotification:** The function *Adc\_EnableGroupNotification* shall enable the notification mechanism for the requested ADC channel group. After calling this API, every successful group conversion will result in a call to a user defined notification function. After calling *Adc\_Init*, notifications are disabled by default. Stopping a group conversion will also disable the notifications, if enabled.

**Prototype:** `void Adc_EnableGroupNotification(Adc_GroupType Group);`

#### **Configuration-time-prerequisites:**

- Enable general notification capability for Adc driver in AdcGeneral container:

Adc Notification Capability



- Enable and configure the notification function for each group where it's intended to be used:

Adc Group Notification

Adc\_Notification\_G0

The notifications can be stopped at any time by calling *Adc\_DisableGroupNotification*.

**Prototype:** `void Adc_DisableGroupNotification (Adc_GroupType Group);`

\* **Adc\_GetGroupStatus:** The function *Adc\_GetGroupStatus* will return the conversion status of the requested ADC channel group. The status can be one of ADC\_IDLE, ADC\_BUSY, ADC\_COMPLETED, ADC\_STREAM\_COMPLETED, depending on the group type and its runtime condition. For more details about the conversion status of Adc Groups, see the relevant sequence diagrams in *AUTOSAR\_SWS\_ADCDriver.pdf*.

**Prototype:** `Adc_StatusType Adc_GetGroupStatus (Adc_GroupType Group);`

\* **Adc\_DeInit:** The function *Adc\_DeInit* returns all ADC HW Units to a state comparable to their power on reset state. *Adc\_Init* can then be called again with different configuration settings.

Prototype: `void Adc_DeInit (void);`

Configuration-time-prerequisites:

- Enable *Adc\_DeInit* API in *AdcGeneral* container:

[Adc\\_DeInit API](#)



### 3.6.1 Starting a Software Triggered Group

A software Group will be started immediately when calling **Adc\_StartGroupConversion API**.

Prototype: `void Adc_StartGroupConversion(Adc_GroupType Group);`

Configuration-time-prerequisites:

- Enable *AdcEnableStartStopGroupApi* in *AdcGeneral* container:

[Adc\\_StartStopGroup API](#)



### 3.6.2 Stopping a Software Group

Depending on the Group type and queue configuration, the Group can be stopped by calling **Adc\_StopGroupConversion API**, or it will be implicitly stopped and removed from the queue. Groups configured with streaming conversion, linear buffer, or one-shot conversion, single access will be implicitly stopped.

Prototype: `void Adc_StopGroupConversion(Adc_GroupType Group);`

### 3.6.3 Starting a Hardware Triggered Group

A Hardware Triggered Group can be prepared for conversion by calling **Adc\_EnableHardwareTrigger API**. Groups used with this API need to have a hardware trigger source associated since configuration time. Each hardware platform supports

several sources of hardware triggers for ADC conversions and any one of them can be selected in `Adc` driver configuration. For a full list of HW trigger sources, see **chapter 3.3: Hardware resources**.

The actual `Adc` conversion will be started when the associated HW trigger event occurs, after calling the **`Adc_EnableHardwareTrigger` Api**.

Prototype: `void Adc_EnableHardwareTrigger(Adc_GroupType Group)`

Configuration-time-prerequisites:

- Enable `AdcHwTriggerApi` in `AdcGeneral` container:

[Adc Hw Trigger API](#)



### 3.6.4 Stopping a Hardware Group

Depending on the Group type and Queue configuration, the Group can be stopped by `Adc_DisableHardwareTrigger` API, or it will be implicitly stopped and removed from the Queue (streaming conversions with linear buffer).

Prototype: `void Adc_DisableHardwareTrigger(Adc_GroupType Group);`

### 3.6.5 Retrieving the conversion results

A direct access from the ADC API functions to the ADC hardware result register is not supported from the ADC driver. There are 2 ADC APIs can be used for retrieving the conversion results:

\* **`Adc_GetStreamLastPointer`**: The function `Adc_GetStreamLastPointer` shall set the pointer, passed as parameter (*PtrToSamplePtr*) to point in the ADC result buffer to the latest result of the first group channel of the last completed conversion round, and return the number of valid samples per channel stored in the ADC result buffer.

Prototype: `Adc_StreamNumSampleType Adc_GetStreamLastPointer(Adc_GroupType Group, Adc_ValueGroupType** PtrToSamplePtr);`

Configuration-time-prerequisites: None.

\* **`Adc_ReadGroup`**: The function `Adc_ReadGroup` shall read the group conversion result of the last completed conversion round of the requested group and stores the channel values starting at the *DataBufferPtr* address.

Prototype: Std\_ReturnType Adc\_ReadGroup(Adc\_GroupType Group, Adc\_ValueGroupType\* DataBufferPtr);

Configuration-time-prerequisites:

- The AdcReadGroupApi parameter needs to be enabled in AdcGeneral container:

Adc\_ReadGroup API



Refer to chapters *Adc\_GetStreamLastPointer* Usage and *Adc\_ReadGroup* Usage in **AUTOSAR\_SWS\_ADCDriver** for more details.

### 3.6.6 Queuing and Priority feature

Adc driver provides a Priority mechanism and a Queuing mechanism for sequencing the Group conversions.

If Priority is enabled, starting a group with higher priority while a group with lower priority is running will cause the running group to be aborted, and the new group to be started instead. If Priority is disabled, the groups are served in “first come, first served” order.

If Queueing is enabled, multiple conversion requests can be stored in the queue, if the ADC hardware is busy converting a previously started group.

The size of the queue is configurable via *AdcPriorityQueueMaxDepth* for Software Groups. For Hardware Groups, the depth of the queue is fixed to 1 because AUTOSAR specification does not allow queuing of Hardware triggered conversion requests. If Priority is also enabled, the groups will be sorted based on configured priority, otherwise “first come first served” order is used. When the group at the top of the queue completes its execution (either stopped implicitly or via Adc API), the next group in the queue is automatically started, if a pending request exists.

Please see sequence diagrams for Priority and Queuing in **AUTOSAR\_SWS\_ADCDriver** for more details.

Configuration-time-prerequisites:

- Enable *AdcEnableQueueing* in AdcGeneral container:

Adc Queue



- Configure the number of *AdcPriorityQueueMaxDepth* in AdcGeneral container:


Adc Max Queue Depth





- Configure *AdcPriorityImplementation* in AdcGeneral container:

Adc Priority Mechanism




ADC\_PRIORITY\_NONE

ADC\_PRIORITY\_HW

ADC\_PRIORITY\_HW\_SW

ADC\_PRIORITY\_NONE



### 3.6.7 Limit checking feature

Adc driver provides Limit checking mechanism allows users to configure a flexible range for checking the result of the conversion, which will be taken for updating the user specified ADC result buffer only when they are in are in configured range.

If limit checking is active for an ADC Channel, only ADC conversion results, which are in the configured range, are taken into account for updating the user specified ADC result buffer. The configured range is combination of the *AdcChannelLowLimit*, *AdcChannelHighLimit* and *AdcChannelRangeSelect*.

Please refer to the requirements for LimitChecking feature in Autosar Specification of ADC driver to see more details.

Configuration-time-prerequisites:

- Enable *AdcEnableLimitCheck* in AdcGeneral container:

Adc Enable Limit Check


☒


- Enable *AdcChannelLimitCheck* in AdcChannel container:


 Adc Channel Limit Check


☒


- Configure the *AdcChannelHighLimit* and *AdcChannelLowLimit* in AdcChannel container:


 Adc Channel Low Limit





 Adc Channel High Limit








### 3.6.8 DET errors

Detected development errors shall be reported to the *Det\_ReportError* service of the Development Error Tracer (DET) if the preprocessor switch *AdcDevErrorDetect* is set.

Please refer to *chapter 7.6: Error detection* in **AUTOSAR\_SWS\_ADCDriver** for more details.

Configuration-time-prerequisites:

- Enable *AdcDevErrorDetect* in AdcGeneral container:

Adc Development Error Detection   

### 3.6.9 DMA transfer

Adc driver provides the DMA transfer mechanism for transferring directly the conversion results 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.



If DMA is used, user must not run SW and HW groups at the same time on the same HW unit because the same DMA channel will be used for both. The buffer containing conversion results should be declared within **NO\_CACHEABLE** section.

Configuration-time-prerequisites:

- *AdcEnableDmaTransferMode* needs to be enabled in NonAutosar container:

Adc Global Enable DMA Transfer   

- Configured *AdcTransferType* as ADC\_DMA in AdcHwUnit container:

General	AdcChannel	AdcGroup
<p>Adc Transfer Type  ADC_DMA </p>		

- In Mcl module, enable *MclDMAChannelEnable* and configure *DmaSource* to the corresponding *Adc Unit*. The *MclDmaTransferCompletionNotif* needs to be configured as the corresponding interrupt in *Adc driver*. For more details, see *chapter 5.3 - ISR to configure within OS – dependencies*.



### 3.6.10 Calibration

*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.

*Adc\_Calibrate API* will return in *pStatus* parameter the status *E\_OK/E\_NOT\_OK* depending if calibration was successful or not. If the hardware supports it, the status of individual BIST steps will also be reported. Please refer chapter 42.5.6 Calibration function in Reference Manual of the microcontroller to see more details.

*Prototype:* `void Adc_Calibrate (VAR (Adc_HwUnitType, AUTOMATIC) Unit,  
P2VAR (Adc_CalibrationStatusType, AUTOMATIC, ADC_APPL_DATA) pStatus);`

*Configuration-time-prerequisites:*

- Enable *AdcEnableCalibration* in *NonAutosar* container:



### 3.6.11 Changing the Clock and Conversion Timing settings

*Adc driver* provides API *Adc\_SetClockMode* for changing the internal settings of the clock in *ADC module*.

The conversion timing settings can also be adjusted when this API is called, per *Unit* or per *Group*, depending on the configuration of *Adc Conversion Time Once* parameter – see the associated chapter for more details.



**Prototype:** Std\_ReturnType Adc\_SetClockMode (VAR(Adc\_DualClockModeType, AUTOMATIC)  
eClockMode) ;

**Configuration-time-prerequisites:**

- Enable AdcEnableDualClockMode in NonAutosar container:

Adc Set Clock Mode API\*



## 3.6.12 Optimizations

The following sections contains driver software optimizations.

### 3.6.12.1 Without Interrupts group

ADC driver provides without interrupt 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) and the application can retrieve the results by calling *Adc\_ReadGroup()*. The user configured result buffer is no longer used to store the results, instead they will be directly read from HW registers, if the flags indicate that the group conversion was completed.

**Configuration-time-prerequisites:**

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

Adc Group Without Interrupts\*



**Note:** When without interrupt is enable for a group, the group access mode must be **ADC\_ACCESS\_MODE\_SINGLE**.

### 3.6.12.2 Optimize OneShot HwTrigger Conversions

ADC driver provides a configuration option to enable maximum speed optimizations for conversion processing of Adc Groups configured with ADC\_CONV\_MODE\_ONESHOT and ADC\_TRIGG\_SRC\_HW.

When this parameter is enabled, no other type of Adc Group may be configured.

Configuration-time-prerequisites:

- Enable AdcOptimizeOneShotHwTriggerConversions in NonAutosar container:

Adc Optimize OneShot HwTrigger Conversions\*    ▾

### 3.6.12.3 Enable Double Buffering Optimization

The ADC driver provides an optional configuration parameter for reducing the number of interrupts required for processing the conversions of Adc Groups and are configured as **ADC\_ACCESS\_MODE\_STREAMING**. Instead of having an interrupt after every sample group conversion, only one interrupt will be raised for a stream of N conversion after all N sample are complete. In addition, one more interrupt will be raised after N/2 sample is complete through enable "Adc Group Enable Half Interrupt" in Adc group.

Adc Group Enable Double Buffering\*



Adc Group Enable Half Interrupt\*






This option is available only when DMA transfer is used for all hardware unit.

**Note:** When this feature is used, Adc Optimize OneShot HwTrigger Conversions and Adc Set Channel API must be deactivated due to driver limitations.

Configuration-time-prerequisites:

- Enable AdcEnableDoubleBufferingOptimization in NonAutosar container:

Adc Enable Double Buffering Optimization\*    ▾

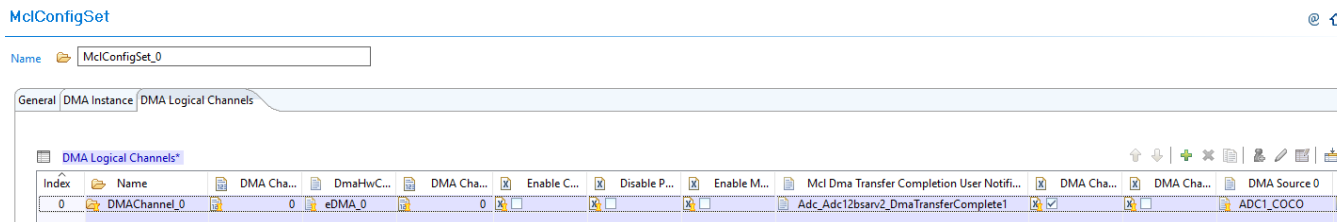
- Enable AdcEnableDoubleBuffering in AdcGroup container:

Adc Group Enable Double Buffering\*



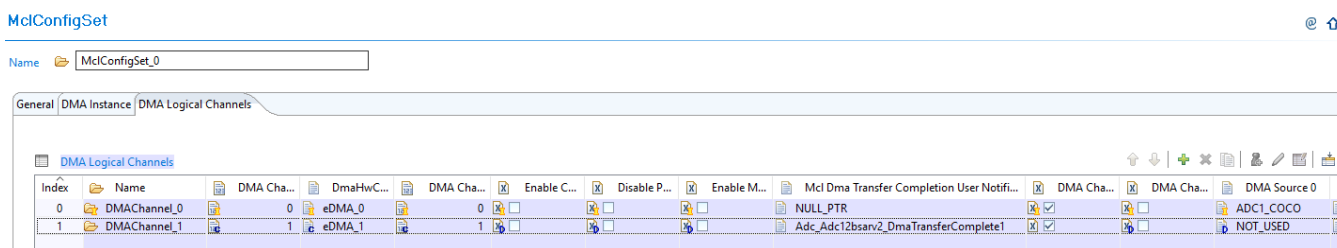
**In this case of one channel**

- For use double Buffering for group one channel, the DMA configuration is similar to normal DMA group (without double buffering).

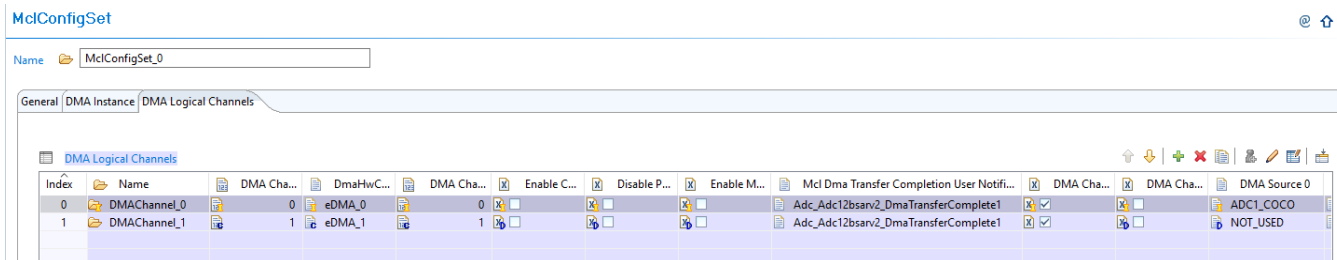


### In this case of more than one channel

- For use double Buffering for group more than one channel, we need to use 2 DMA channel for one hardware unit which have group double buffer more than one channel, channel 1 use to transfer, channel 2 raised interrupt. Interrupt must be enable for DMA channel 2.



### In this case of one channel/normal dma groups and more than one channel



The MclDmaTransferCompletionNotif needs to be configured as the corresponding interrupt in Adc driver for both DMA channel related to ADC hardware unit

**Note:** In case Adc config with 2 HW Unit use DMA, if only one HW Unit contain group enable double buffer with number of channel > 1, 3 DMA channel need to use, if both two HW Unit contain group Enable double buffer with number channel > 1, 4 DMA channel need to use.

### 3.6.12.4 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.

**Prototype:** `void Adc_SetChannel( Adc_GroupType Group, Adc_GroupDefType pChannel, uint16 pDelays, uint32 u32Mask, Adc_ChannelIndexType NumberOfChannel);`

#### Configuration-time-prerequisites:

- The `AdcEnableSetChannel` parameter needs to be enabled in NonAutosar container:

Adc Set Channel API\*



- The `AdcEnableInitialNotification` parameter needs to be enabled in NonAutosar 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 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` HW registers. So the very next conversion will not be affected by the changes, but the second next conversion will. To overcome this drawback, a separate, Non-Autosar notification can be configured to be

used with `Adc_SetChannel` feature – the `Adc` Initial notification. It can be used by the user application to call `Adc_SetChannel` API before `Adc` driver updates the HW configuration for the next conversion, avoiding the limitation of Autosar notification.

### 3.6.12.5 Conversion Time Once

ADC driver provides Conversion time once feature to enable/disable one time configuration of the settings that affect configuration time. If this feature is enabled, the timing relevant 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 groups 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 `NonAutosar` container:

`Adc Conversion Time Once*`



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

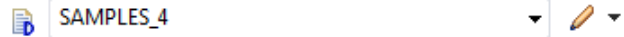
## ▼ AdcNormalConversionTimings

Name\*  AdcNormalConversionTimings

Adc Hardware Average Enable\*



Adc Hardware Average Select\*




Adc Sample Time Duration (2 -&gt; 256)\*



Adc Clock Divide Select\*



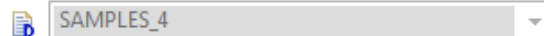
## ▼ AdcAlternateConversionTimings

Name\*  AdcAlternateConversionTimings

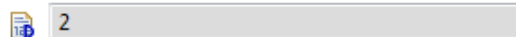
Adc Hardware Average Enable\*



Adc Hardware Average Select\*



Adc Sample Time Duration (2 -&gt; 256)\*



Adc Clock Divide Select\*



### 3.6.12.6 Bypass Consistency loop

Adc driver provides a mechanism called Bypass Consistency Loop to increase the execution speed when stopping a group. Normally, the Adc driver polls the status of the HW until the conversion is confirmed as stopped. If this setting is used, the status polling is skipped, and the application must make sure to not call an ADC service before the HW reaches the correct state.

Configuration-time-prerequisites:

- Enable AdcBypassConsistencyLoop in NonAutosar container. When this parameter is enabled, Queuing and Priority features must be disabled. This feature is applicable only for groups that have Without Interrupts feature configured:

Adc Bypass Consistency Loop\*



**Note:** When without interrupt is enable for a group, the group access mode must be **ADC\_ACCESS\_MODE\_SINGLE**.

### 3.6.13 Programmable Delay Block (PDB)

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

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

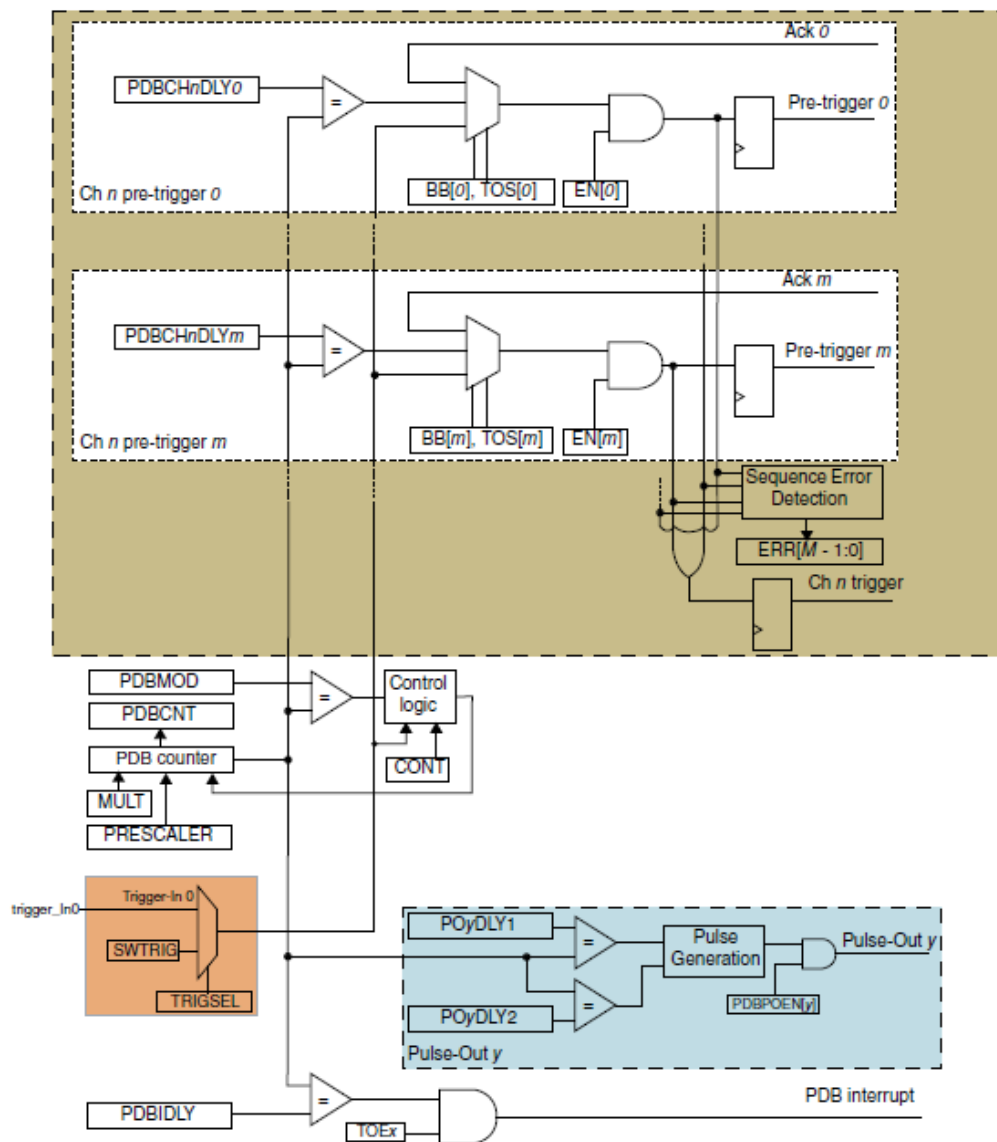
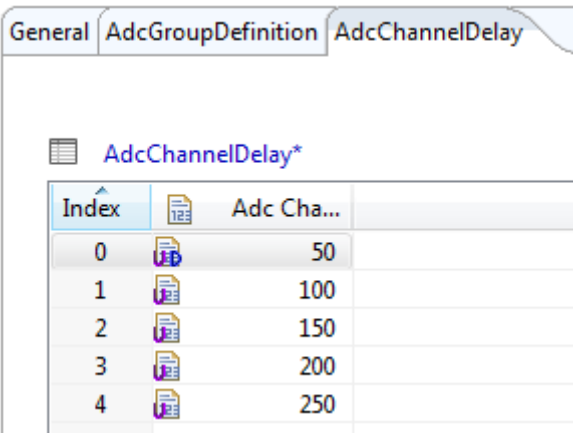


Figure 3-2. PDB block diagram

The PDB pretriggers can work in either Back to Back Mode or work as individual Delays for channel conversions, and the Adc driver offers configuration options for each Adc Group, defining how the PDB should be configured to work:  
AdcGroupInBacktoBackMode, AdcGroupUsesChannelDelays, AdcDelayNextPdb.

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

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



Index	Adc Cha...
0	50
1	100
2	150
3	200
4	250

**3.6.13.1 AdcGroupInBacktoBackMode - Enable/Disable the channel conversions occurring in Back to Back mode.**



When this parameter is enabled, the PDB pretriggers will be configured to work in back to back mode: PDB\_CH1n\_C1[BB] bits are set for all the group channels, except the first one.

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

The Adc driver can convert only up to N channels at once, where N = number of PDB Channels X 8 pretriggers each; for groups larger than that, Adc driver will split the group in sub-groups of N and convert them with sub-group conversions. Interrupt processing is needed between the sub-groups for register re-configuration. The N channels are



converted using one or more PDB channels (each controlling 8 pretriggers), but the different channels do not work in back to back mode together, so a delay is needed for the 2nd, 3rd PDB channels, otherwise PDB errors will occur. Adc driver will configure this delay value when Back to back mode is used, with the value defined by the user in **AdcDelayNextPdb** parameter. The sequences of 8 channels associated to a PDB channel will happen in back to back mode.

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

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

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

### 3.6.13.2 AdcGroupUsesChannelDelays - Enable/Disable the usage of channel delays in PDB pretriggers.

Adc Group Uses Channel Delays\*



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

The Adc driver can convert only up to N channels at once, where N = number of PDB Channels X 8 pretriggers each; for groups larger than that, Adc driver will split the group in sub-groups of N and convert them with sub-group conversions. Interrupt processing is needed between the sub-groups for register re-configuration. The timing of the channel conversions will be accurate only for the first N channels; for the next ones, the on top of the configured delay, the delay of previous conversions and interrupt processing will be implicitly added. For this reason, it's safe to configure a delay of 0 for the N+1 channel. But the delay values for each group of N need to be increasing, so that the channel conversions occur in the order they were defined in the group.

If **AdcGroupInBacktoBackMode** is also configured for the Adc Group, the user can configure a delay only for the first channel conversion, and the rest will be converted in back to back mode (see description of **AdcGroupInBacktoBackMode**). The delay value defined by the user in **AdcDelayNextPdb** parameter will be used for 2nd, 3rd PDB channel to avoid PDB internal errors.

### 3.6.13.3 AdcDelayNextPdb – configuring the delay between PDB channel conversions

Adc Delay Next PDB (0 -> 65535)\*

1000

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

### 3.6.13.4 PDB Unit settings

There is a PDB sub-container for configuring general PDB unit settings.

▼ AdcPdbSettings

Name AdcPdbSettings

---

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

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

PDB Sequence Error Interrupt Enable ☐

PDB Sequence Error Notification NULL\_PTR

AdcPdbPrescalerDividerSelect and AdcPdbMultiplicationFactorSelect can be used to fine tune the frequency at which PDB counter works. These values will be written into PRESCALER and MULT bitfields of PDB\_SC register during initialization. The PDB counter uses the peripheral clock divided by the product of a factor (selected by MULT field) and an integer factor (set by PRESCALAR field), or in other words, (peripheral clock)/(MULT x PRESCALAR). For more details about the implementation of these bitfields in the hardware please check the Reference Manual.

Setting AdcPdbChannelSequenceErrorEnable enables the PDB sequence error interrupt - any of the PDB channel sequence error flags will generate a PDB sequence error interrupt. The sequence error flags can be set as result of various invalid scenarios involving the PDB, most usual being that one pretrigger becomes asserted and requests a conversion to ADC unit while another conversion is still ongoing. When one of these flags is set, the PDB unit will be stuck and not request any conversions until the flags are cleared. The flags will be cleared by PDB sequence error interrupt (if enabled), and an optional notification function will be called to signal this event to the user. The notification function can be configured in AdcPdbErrorNotification field.

## 3.7 Runtime Errors

The driver generates the following DEM errors at runtime.

**Table 3-3. Runtime Errors**

Function	Error Code	Condition triggering the error
Adc_StartGroupConversion	Adc_E_Timeout	Timeout expired. Ongoing conversion is not aborted.
Adc_StopGroupConversion	Adc_E_Timeout	Timeout expired. Ongoing conversion is not aborted.
Adc_Calibrate	Adc_E_Timeout	Timeout expired. Calibrate not success.
Adc_Pdb_ChannelSequenceError	Adc_E_Timeout	Timeout expired. DMA does not transfer successfully

## 3.8 Software specification

The following sections contains driver software specifications.

## 3.8.1 Define Reference

Constants supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.3 Rev0001 .

### 3.8.1.1 Define ADC\_VENDOR\_ID

Table 3-4. Define ADC\_VENDOR\_ID Description

Name	ADC_VENDOR_ID
Initializer	43

### 3.8.1.2 Define ADC\_MODULE\_ID

Table 3-5. Define ADC\_MODULE\_ID Description

Name	ADC_MODULE_ID
Initializer	123

### 3.8.1.3 Define ADC\_AR\_RELEASE\_MAJOR\_VERSION

Table 3-6. Define ADC\_AR\_RELEASE\_MAJOR\_VERSION Description

Name	ADC_AR_RELEASE_MAJOR_VERSION
Initializer	4

### 3.8.1.4 Define ADC\_AR\_RELEASE\_MINOR\_VERSION

Table 3-7. Define ADC\_AR\_RELEASE\_MINOR\_VERSION Description

Name	ADC_AR_RELEASE_MINOR_VERSION
Initializer	3

### 3.8.1.5 Define ADC\_AR\_RELEASE\_REVISION\_VERSION

**Violates:** The compiler/linker shall be checked to ensure that 31 character significance and case sensitivity are supported for external identifiers

**Table 3-8. Define ADC\_AR\_RELEASE\_REVISION\_VERSION**  
Description

Name	ADC_AR_RELEASE_REVISION_VERSION
Initializer	1

### 3.8.1.6 Define ADC\_SW\_MAJOR\_VERSION

**Table 3-9. Define ADC\_SW\_MAJOR\_VERSION**  
Description

Name	ADC_SW_MAJOR_VERSION
Initializer	1

### 3.8.1.7 Define ADC\_SW\_MINOR\_VERSION

**Table 3-10. Define ADC\_SW\_MINOR\_VERSION** Description

Name	ADC_SW_MINOR_VERSION
Initializer	0

### 3.8.1.8 Define ADC\_SW\_PATCH\_VERSION

**Table 3-11. Define ADC\_SW\_PATCH\_VERSION**  
Description

Name	ADC_SW_PATCH_VERSION
Initializer	1

### 3.8.1.9 Define ADC\_E\_UNINIT

API service used without Adc module initialization.

#### Details:

All error codes

**Violates:** Identifier clash Development errors. The following errors shall be detectable by the ADC module depending on its configuration (development / production mode).

**Table 3-12. Define ADC\_E\_UNINIT Description**

<b>Name</b>	ADC_E_UNINIT
<b>Initializer</b>	((uint8)0x0AU)

### 3.8.1.10 Define ADC\_E\_BUSY

Adc module is busy with a running operation.

**Table 3-13. Define ADC\_E\_BUSY Description**

<b>Name</b>	ADC_E_BUSY
<b>Initializer</b>	((uint8)0x0BU)

### 3.8.1.11 Define ADC\_E\_IDLE

Adc module is in idle state.

**Table 3-14. Define ADC\_E\_IDLE Description**

<b>Name</b>	ADC_E_IDLE
<b>Initializer</b>	((uint8)0x0CU)

### 3.8.1.12 Define ADC\_E\_ALREADY\_INITIALIZED

The ADC module is already initilized.

**Table 3-15. Define ADC\_E\_ALREADY\_INITIALIZED Description**

<b>Name</b>	ADC_E_ALREADY_INITIALIZED
<b>Initializer</b>	((uint8)0x0DU)

### 3.8.1.13 Define ADC\_E\_PARAM\_CONFIG

The ADC module is not properly configured.

**Table 3-16. Define ADC\_E\_PARAM\_CONFIG Description**

<b>Name</b>	ADC_E_PARAM_CONFIG
<b>Initializer</b>	((uint8)0x0EU)

### 3.8.1.14 Define ADC\_E\_PARAM\_POINTER

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

**Table 3-17. Define ADC\_E\_PARAM\_POINTER Description**

<b>Name</b>	ADC_E_PARAM_POINTER
<b>Initializer</b>	((uint8)0x14U)

### 3.8.1.15 Define ADC\_E\_PARAM\_GROUP

API service used with an invalid ADC group.

**Table 3-18. Define ADC\_E\_PARAM\_GROUP Description**

<b>Name</b>	ADC_E_PARAM_GROUP
<b>Initializer</b>	((uint8)0x15U)

### 3.8.1.16 Define ADC\_E\_WRONG\_CONV\_MODE

API service used with an invalid ADC Conversion Mode.

**Table 3-19. Define ADC\_E\_WRONG\_CONV\_MODE Description**

<b>Name</b>	ADC_E_WRONG_CONV_MODE
<b>Initializer</b>	((uint8)0x16U)

### 3.8.1.17 Define ADC\_E\_WRONG\_TRIGG\_SRC

API service used with an invalid ADC Trigger Source.

**Table 3-20. Define ADC\_E\_WRONG\_TRIGG\_SRC Description**

<b>Name</b>	ADC_E_WRONG_TRIGG_SRC
<b>Initializer</b>	((uint8)0x17U)

### 3.8.1.18 Define ADC\_E\_NOTIF\_CAPABILITY

Check the notification capability of a group.

**Table 3-21. Define ADC\_E\_NOTIF\_CAPABILITY Description**

<b>Name</b>	ADC_E_NOTIF_CAPABILITY
<b>Initializer</b>	((uint8)0x18U)

### 3.8.1.19 Define ADC\_E\_BUFFER\_UNINIT

API service used without initializing the buffer.

**Table 3-22. Define ADC\_E\_BUFFER\_UNINIT Description**

<b>Name</b>	ADC_E_BUFFER_UNINIT
<b>Initializer</b>	((uint8)0x19U)

### 3.8.1.20 Define ADC\_E\_NOT\_DISENGAGED

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

**Table 3-23. Define ADC\_E\_NOT\_DISENGAGED Description**

<b>Name</b>	ADC_E_NOT_DISENGAGED
<b>Initializer</b>	((uint8)0x1AU)

### 3.8.1.21 Define ADC\_E\_QUEUE\_FULL

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



**Table 3-24. Define ADC\_E\_QUEUE\_FULL  
Description**

<b>Name</b>	ADC_E_QUEUE_FULL
<b>Initializer</b>	((uint8)0x20U)

### 3.8.1.22 Define ADC\_E\_PARAM\_UNIT

API service called using a wrong ADC unit.

**Table 3-25. Define ADC\_E\_PARAM\_UNIT Description**

<b>Name</b>	ADC_E_PARAM_UNIT
<b>Initializer</b>	((uint8)0x27U)

### 3.8.1.23 Define ADC\_E\_INVALID\_CLOCK\_MODE

Adc\_SetClockMode service called using an invalid clock mode.

**Table 3-26. Define ADC\_E\_INVALID\_CLOCK\_MODE Description**

<b>Name</b>	ADC_E_INVALID_CLOCK_MODE
<b>Initializer</b>	((uint8)0x2AU)

### 3.8.1.24 Define ADC\_E\_PARAM\_CHANNEL

Adc\_SetChannel service called using an invalid channel list.

**Table 3-27. Define ADC\_E\_PARAM\_CHANNEL Description**

<b>Name</b>	ADC_E_PARAM_CHANNEL
<b>Initializer</b>	((uint8)0x2BU)

### 3.8.1.25 Define ADC\_E\_BUFFER\_UNINIT\_LIST

**Table 3-28. Define ADC\_E\_BUFFER\_UNINIT\_LIST  
Description**

<b>Name</b>	ADC_E_BUFFER_UNINIT_LIST
-------------	--------------------------

*Table continues on the next page...*

**Table 3-28. Define ADC\_E\_BUFFER\_UNINIT\_LIST Description  
(continued)**

<b>Initializer</b>	((uint32)0x00000001U)
--------------------	-----------------------

### 3.8.1.26 Define ADC\_E\_WRONG\_TRIGG\_SRC\_LIST

**Table 3-29. Define ADC\_E\_WRONG\_TRIGG\_SRC\_LIST  
Description**

<b>Name</b>	ADC_E_WRONG_TRIGG_SRC_LIST
<b>Initializer</b>	((uint32)0x00000002U)

### 3.8.1.27 Define ADC\_E\_QUEUE\_FULL\_LIST

**Table 3-30. Define ADC\_E\_QUEUE\_FULL\_LIST  
Description**

<b>Name</b>	ADC_E_QUEUE_FULL_LIST
<b>Initializer</b>	((uint32)0x00000004U)

### 3.8.1.28 Define ADC\_E\_WRONG\_CONV\_MODE\_LIST

**Table 3-31. Define ADC\_E\_WRONG\_CONV\_MODE\_LIST  
Description**

<b>Name</b>	ADC_E_WRONG_CONV_MODE_LIST
<b>Initializer</b>	((uint32)0x00000008U)

### 3.8.1.29 Define ADC\_E\_SET\_MODE\_LIST

**Table 3-32. Define ADC\_E\_SET\_MODE\_LIST Description**

<b>Name</b>	ADC_E_SET_MODE_LIST
<b>Initializer</b>	((uint32)0x00000040U)

### 3.8.1.30 Define ADC\_INIT\_ID

API service ID for Adc\_Init function.

#### Details:

All AUTOSAR API's service IDs

**Table 3-33. Define ADC\_INIT\_ID Description**

<b>Name</b>	ADC_INIT_ID
<b>Initializer</b>	0x00U

### 3.8.1.31 Define ADC\_DEINIT\_ID

API service ID for Adc\_DeInit function.

**Table 3-34. Define ADC\_DEINIT\_ID Description**

<b>Name</b>	ADC_DEINIT_ID
<b>Initializer</b>	0x01U

### 3.8.1.32 Define ADC\_STARTGROUPCONVERSION\_ID

API service ID for Adc\_StartGroupConversion function.

**Table 3-35. Define ADC\_STARTGROUPCONVERSION\_ID Description**

<b>Name</b>	ADC_STARTGROUPCONVERSION_ID
<b>Initializer</b>	0x02U

### 3.8.1.33 Define ADC\_STOPGROUPCONVERSION\_ID

API service ID for Adc\_StopGroupConversion function.

**Table 3-36. Define ADC\_STOPGROUPCONVERSION\_ID Description**

<b>Name</b>	ADC_STOPGROUPCONVERSION_ID
<b>Initializer</b>	0x03U

### 3.8.1.34 Define ADC\_VALUEREADGROUP\_ID

API service ID for Adc\_ReadGroup function.

**Table 3-37. Define ADC\_VALUEREADGROUP\_ID Description**

<b>Name</b>	ADC_VALUEREADGROUP_ID
<b>Initializer</b>	0x04U

### 3.8.1.35 Define ADC\_ENABLEHARDWARETRIGGER\_ID

API service ID for Adc\_EnableHardwareTrigger function.

**Table 3-38. Define ADC\_ENABLEHARDWARETRIGGER\_ID Description**

<b>Name</b>	ADC_ENABLEHARDWARETRIGGER_ID
<b>Initializer</b>	0x05U

### 3.8.1.36 Define ADC\_DISABLEHARDWARETRIGGER\_ID

API service ID for Adc\_DisableHardwareTrigger function.

**Table 3-39. Define ADC\_DISABLEHARDWARETRIGGER\_ID Description**

<b>Name</b>	ADC_DISABLEHARDWARETRIGGER_ID
<b>Initializer</b>	0x06U

### 3.8.1.37 Define ADC\_ENABLEGROUPNOTIFICATION\_ID

API service ID for Adc\_EnableGroupNotification function.

**Table 3-40. Define ADC\_ENABLEGROUPNOTIFICATION\_ID Description**

<b>Name</b>	ADC_ENABLEGROUPNOTIFICATION_ID
<b>Initializer</b>	0x07U

### 3.8.1.38 Define ADC\_DISABLEGROUPNOTIFICATION\_ID

API service ID for Adc\_DisableGroupNotification function.

**Table 3-41. Define ADC\_DISABLEGROUPNOTIFICATION\_ID Description**

<b>Name</b>	ADC_DISABLEGROUPNOTIFICATION_ID
<b>Initializer</b>	0x08U

### 3.8.1.39 Define ADC\_GETGROUPSTATUS\_ID

API service ID for Adc\_GetGroupStatus function.

**Table 3-42. Define ADC\_GETGROUPSTATUS\_ID Description**

<b>Name</b>	ADC_GETGROUPSTATUS_ID
<b>Initializer</b>	0x09U

### 3.8.1.40 Define ADC\_GETVERSIONINFO\_ID

API service ID for Adc\_GetVersionInfo function.

**Table 3-43. Define ADC\_GETVERSIONINFO\_ID Description**

<b>Name</b>	ADC_GETVERSIONINFO_ID
<b>Initializer</b>	0x0AU

### 3.8.1.41 Define ADC\_GETSTREAMLASTPOINTER\_ID

API service ID for Adc\_GetStreamLastPointer function.

**Table 3-44. Define ADC\_GETSTREAMLASTPOINTER\_ID Description**

<b>Name</b>	ADC_GETSTREAMLASTPOINTER_ID
<b>Initializer</b>	0x0BU

### 3.8.1.42 Define ADC\_SETUPRESULTBUFFER\_ID

API service ID for Adc\_SetupResultBuffer function.

**Table 3-45. Define ADC\_SETUPRESULTBUFFER\_ID Description**

<b>Name</b>	ADC_SETUPRESULTBUFFER_ID
<b>Initializer</b>	0x0CU

### 3.8.1.43 Define ADC\_SETCLOCKMODE\_ID

API service ID for Adc\_SetClockMode function.

**Table 3-46. Define ADC\_SETCLOCKMODE\_ID Description**

<b>Name</b>	ADC_SETCLOCKMODE_ID
<b>Initializer</b>	0x24U

### 3.8.1.44 Define ADC\_CALIBRATE\_ID

API service ID for Adc\_Calibrate function.

**Table 3-47. Define ADC\_CALIBRATE\_ID Description**

<b>Name</b>	ADC_CALIBRATE_ID
<b>Initializer</b>	0x28U

### 3.8.1.45 Define ADC\_SETCHANNEL\_ID

API service ID for Adc\_SetChannel function.

**Table 3-48. Define ADC\_SETCHANNEL\_ID Description**

<b>Name</b>	ADC_SETCHANNEL_ID
<b>Initializer</b>	(0x31U)

## 3.8.2 Enum Reference

Enumeration of all constants supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.3 Rev0001 .

### 3.8.2.1 Enumeration Adc\_GlobalStateType

ADC driver status.

#### Details:

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

**Table 3-49. Enumeration Adc\_GlobalStateType Values**

Name	Initializer	Description
ADC_STATE_UNINIT	0U	Hardware unit Un-Initializing state
ADC_STATE_BUSY	1U	Hardware unit Busy state
ADC_STATE_IDLE	2U	Hardware unit Idle state

### 3.8.2.2 Enumeration Adc\_DualClockModeType

Group Access Mode type.

#### Details:

Used for value received by Tressos interface configuration.

**Table 3-50. Enumeration Adc\_DualClockModeType Values**

Name	Initializer	Description
ADC_NORMAL	0U	Normal mode.
ADC_ALTERNATE	1U	Alternate mode.

### 3.8.2.3 Enumeration Adc\_GroupConversionStateType

ADC group already converted type.

#### Details:

Used to differentiate if group is already converted or not.

**Table 3-51. Enumeration Adc\_GroupConversionStateType Values**

Name	Initializer	Description
ADC_NOT_YET_CONVERTED	0U	Group not yet converted.
ADC_ALREADY_CONVERTED	1U	Group is already converted.

### 3.8.2.4 Enumeration Adc\_GroupAccessModeType

Adc group access Mode.

#### Details:

Used for value received by Tressos interface configuration.

**Table 3-52. Enumeration Adc\_GroupAccessModeType Values**

Name	Initializer	Description
ADC_ACCESS_MODE_SINGLE	0U	Single access mode.
ADC_ACCESS_MODE_STREAMING	1U	Streaming access mode.

### 3.8.2.5 Enumeration Adc\_GroupConvType

Adc group conversion.

#### Details:

Used for value received by Tressos interface configuration.



**Table 3-53. Enumeration Adc\_GroupConvType Values**

Name	Initializer	Description
ADC_CONV_TYPE_NORMAL	0U	Normal conversion mode.
ADC_CONV_TYPE_INJECTED	1U	Injected conversion mode.

### 3.8.2.6 Enumeration Adc\_GroupConvModeType

Adc Group conversion mode.

#### Details:

Used for value received by Tressos interface configuration.

**Table 3-54. Enumeration Adc\_GroupConvModeType Values**

Name	Initializer	Description
ADC_CONV_MODE_ONESHOT	0U	One shot.
ADC_CONV_MODE_CONTINUOUS	1U	Continuous conversion mode.

### 3.8.2.7 Enumeration Adc\_GroupReplacementType

Adc group replacement.

#### Details:

Used for value received by Tressos interface configuration.

**Table 3-55. Enumeration Adc\_GroupReplacementType Values**

Name	Initializer	Description
ADC_GROUP_REPL_ABORT_RESTART	0U	Abort and restart of group.
ADC_GROUP_REPL_SUSPEND_RESUME	1U	Suspend and resuming of group.

### 3.8.2.8 Enumeration Adc\_StreamBufferModeType

Adc group streaming buffer mode.

#### Details:

Used for value received by Tressos interface configuration.

**Table 3-56. Enumeration Adc\_StreamBufferModeType Values**

Name	Initializer	Description
ADC_STREAM_BUFFER_LINEAR	0U	Linear streaming.
ADC_STREAM_BUFFER_CIRCULAR	1U	Circular streaming.

### 3.8.2.9 Enumeration Adc\_StatusType

ADC group status.

#### Details:

ADC group enumeration type.

**Table 3-57. Enumeration Adc\_StatusType Values**

Name	Initializer	Description
ADC_IDLE	0U	Group is in IDLE state.
ADC_BUSY	1U	Group is in BUSY state.
ADC_COMPLETED	2U	Group is in COMPLETED state.
ADC_STREAM_COMPLETED	3U	Group is in STREAM_COMPLETED state.

### 3.8.2.10 Enumeration Adc\_NotificationType

ADC group notification.

#### Details:

Indicates if notification is enabled for the group.

**Table 3-58. Enumeration Adc\_NotificationType Values**

Name	Initializer	Description
ADC_NOTIFICATION_DISABLED	0U	Notification is disabled.
ADC_NOTIFICATION_ENABLED	1U	Notification is enabled.

### 3.8.2.11 Enumeration Adc\_HwTriggerSignalType

Adc hardware trigger edge.

#### Details:

Used for value received by Tressos interface configuration.

**Table 3-59. Enumeration Adc\_HwTriggerSignalType Values**

Name	Initializer	Description
ADC_HW_TRIG_RISING_EDGE	0U	Rising edge.
ADC_HW_TRIG_FALLING_EDGE	1U	Falling edge.
ADC_HW_TRIG_BOTH_EDGES	2U	Falling and rising edge.

### 3.8.2.12 Enumeration Adc\_TriggerSourceType

Adc hardware trigger source.

#### Details:

Used for value received by Tressos interface configuration.

**Table 3-60. Enumeration Adc\_TriggerSourceType Values**

Name	Initializer	Description
ADC_TRIGG_SRC_SW	0U	Software triggered.
ADC_TRIGG_SRC_HW	1U	Hardware triggered.

### 3.8.2.13 Enumeration Adc\_HwTriggeringType

Adc Hardware trigger.

#### Details:

Indicates if hardware trigger is enabled for group.

**Table 3-61. Enumeration Adc\_HwTriggeringType Values**

Name	Initializer	Description
ADC_HWTRIGGER_DISABLED	0U	Hardware trigger is disabled.
ADC_HWTRIGGER_ENABLED	1U	Hardware trigger is enabled.

### 3.8.2.14 Enumeration Adc\_ChannelRangeSelectType

Range select values.

#### Details:

Indicates which range select is used.

**Table 3-62. Enumeration Adc\_ChannelRangeSelectType Values**

Name	Initializer	Description
ADC_RANGE_ALWAYS	0U	Complete range - independent from channel limit settings.
ADC_RANGE_BETWEEN	1U	Range between low limit and high limit - high limit value included.
ADC_RANGE_NOT_BETWEEN	2U	Range above high limit or below low limit - low limit value included.
ADC_RANGE_NOT_OVER_HIGH	3U	Range below high limit - high limit value included.
ADC_RANGE_NOT_UNDER_LOW	4U	Range above low limit.
ADC_RANGE_OVER_HIGH	5U	Range above high limit.
ADC_RANGE_UNDER_LOW	6U	Below low limit - low limit value included.

### 3.8.3 Function Reference

Functions of all functions supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.3 Rev0001 .

#### 3.8.3.1 Function Adc\_Init

Initializes the ADC hardware unit and the driver.

**Details:**

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

**Return:** void.

**Post:** Initializes the driver.

**Note**

The function Autosar Service ID[hex]: 0x00.Synchronous.Non  
Re-entrant function.

**Implements:** Adc\_Init\_Activity

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Violates:** Only preprocessor statements and comments before "#include"

**Violates:** Repeated include file MemMap.h

**Prototype:** void Adc\_Init(const Adc\_ConfigType \*ConfigPtr);

**Table 3-63. Adc\_Init Arguments**

Type	Name	Direction	Description
	pConfigPtr	input	Pointer to configuration set in Variant PB (Variant PC requires a NULL_PTR).

#### 3.8.3.2 Function Adc\_DeInit

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

**Details:**

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

**Return:** void.

**Note**

The function Autosar Service ID[hex]: 0x01.Synchronous.Non  
Re-entrant function.

**Implements:** Adc\_DeInit\_Activity

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** void Adc\_DeInit(void);

### 3.8.3.3 Function Adc\_SetupResultBuffer

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

**Details:**

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.

**Return:** Std\_ReturnType Standard return type.

**Note**

The function Autosar Service ID[hex]: 0x0C.Synchronous.Re-  
entrant function.

**Implements:** Adc\_SetupResultBuffer\_Activity

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** Std\_ReturnType Adc\_SetupResultBuffer(Adc\_GroupType Group, Adc\_ValueGroupType  
\*DataBufferPtr);

**Table 3-64. Adc\_SetupResultBuffer Arguments**

Type	Name	Direction	Description
Adc_GroupType	Group	input	Numeric ID of requested ADC channel group.
	pDataBufferPtr	input	Pointer to result data buffer.

**Table 3-65. Adc\_SetupResultBuffer Return Values**

Name	Description
E_OK:	Result buffer pointer initialized correctly.
E_NOT_OK:	Operation failed or development error occurred.

### 3.8.3.4 Function Adc\_EnableGroupNotification

Enables the notification mechanism for the requested ADC channel group.

#### **Details:**

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

**Return:** void.

#### **Note**

The function Autosar Service ID[hex]: 0x07.Synchronous.Re-entrant function.

**Implements:** Adc\_EnableGroupNotification\_Activity

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** void Adc\_EnableGroupNotification(Adc\_GroupType Group);

**Table 3-66. Adc\_EnableGroupNotification Arguments**

Type	Name	Direction	Description
Adc_GroupType	Group	input	Numeric ID of requested ADC channel group.

### 3.8.3.5 Function `Adc_DisableGroupNotification`

Disables the notification mechanism for the requested ADC channel group.

**Details:**

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

**Return:** void.

**Note**

The function Autosar Service ID[hex]: 0x08.Synchronous.Re-entrant function.

**Implements:** `Adc_DisableGroupNotification_Activity`

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** `void Adc_DisableGroupNotification(Adc_GroupType Group);`

**Table 3-67. `Adc_DisableGroupNotification` Arguments**

Type	Name	Direction	Description
<code>Adc_GroupType</code>	Group	input	Numeric ID of requested ADC channel group.

### 3.8.3.6 Function `Adc_EnableHardwareTrigger`

Enables the hardware trigger for the requested ADC Channel group.

**Details:**

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.

**Return:** void.

**Note**

The function Autosar Service ID[hex]: 0x05.Synchronous.Re-entrant function.

**Implements:** `Adc_EnableHardwareTrigger_Activity`



**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** `void Adc_EnableHardwareTrigger(Adc_GroupType Group);`

**Table 3-68. Adc\_EnableHardwareTrigger Arguments**

Type	Name	Direction	Description
Adc_GroupType	Group	input	Numeric ID of requested ADC channel group.

### 3.8.3.7 Function Adc\_DisableHardwareTrigger

Disables the hardware trigger for the requested ADC Channel group.

**Details:**

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

**Return:** void.

#### Note

The function Autosar Service ID[hex]: 0x06.Synchronous.Re-entrant function.

**Implements:** Adc\_DisableHardwareTrigger\_Activity

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** `void Adc_DisableHardwareTrigger(Adc_GroupType Group);`

**Table 3-69. Adc\_DisableHardwareTrigger Arguments**

Type	Name	Direction	Description
Adc_GroupType	Group	input	Numeric ID of requested ADC channel group.

### 3.8.3.8 Function Adc\_StartGroupConversion

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

**Details:**

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

**Return:** void.

**Note**

The function Autosar Service ID[hex]: 0x02.Asynchronous.Re-entrant function.

**Implements:** Adc\_StartGroupConversion\_Activity

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** void Adc\_StartGroupConversion(Adc\_GroupType Group);

**Table 3-70. Adc\_StartGroupConversion Arguments**

Type	Name	Direction	Description
Adc_GroupType	Group	input	Numeric ID of requested ADC channel group.

### 3.8.3.9 Function Adc\_StopGroupConversion

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

**Details:**

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

**Return:** void.

**Note**

The function Autosar Service ID[hex]: 0x03.Synchronous.Re-entrant function.

**Implements:** Adc\_StopGroupConversion\_Activity

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** void Adc\_StopGroupConversion(Adc\_GroupType Group);

**Table 3-71. Adc\_StopGroupConversion Arguments**

Type	Name	Direction	Description
Adc_GroupType	Group	input	Numeric ID of requested ADC channel group.

### 3.8.3.10 Function Adc\_GetGroupStatus

Returns the conversion status of the requested ADC Channel group.

**Details:**

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

**Return:** Adc\_StatusType Conversion status for the requested group.

**Note**

The function Autosar Service ID[hex]: 0x09.Synchronous.Re-entrant function.

**Implements:** Adc\_GetGroupStatus\_Activity

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** Adc\_StatusType Adc\_GetGroupStatus(Adc\_GroupType Group);

**Table 3-72. Adc\_GetGroupStatus Arguments**

Type	Name	Direction	Description
Adc_GroupType	Group	input	Numeric ID of requested ADC channel group.

**Table 3-73. Adc\_GetGroupStatus Return Values**

Name	Description
ADC_IDLE	In case of errors.
conversion	Status in case of no errors.

### 3.8.3.11 Function `Adc_GetStreamLastPointer`

Returns the number of valid samples per channel.

#### Details:

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).

**Return:** `Adc_StreamNumSampleType` Number of valid samples per channel.

#### **Note**

The function Autosar Service ID[hex]:  
0x0B.Synchronous.Reentrant function.

**Implements:** `Adc_GetStreamLastPointer_Activity`

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** `Adc_StreamNumSampleType Adc_GetStreamLastPointer(Adc_GroupType Group, Adc_ValueGroupType **PtrToSamplePtr);`

**Table 3-74. `Adc_GetStreamLastPointer` Arguments**

Type	Name	Direction	Description
<code>Adc_GroupType</code>	Group	input	Numeric ID of requested ADC channel group.
<code>Adc_ValueGroupType**</code>	PtrToSamplePtr	output	Pointer to result buffer pointer.

**Table 3-75. `Adc_GetStreamLastPointer` Return Values**

Name	Description
0	In case of errors.
>0	Number of valid samples per channel.

### 3.8.3.12 Function `Adc_GetVersionInfo`

Returns the version information of this module.

**Details:**

Returns the version information of this module.

**Note**

The function Autosar Service ID[hex]:  
0x0A.Synchronous.Reentrant function.

**Implements:** Adc\_GetVersionInfo\_Activity

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Violates:** The identifiers used in the declaration and definition of a function shall be identical

**Prototype:** `void Adc_GetVersionInfo(Std_VersionInfoType *versionInfo);`

**Table 3-76. Adc\_GetVersionInfo Arguments**

Type	Name	Direction	Description
	pVersionInfo	output	Pointer to where to store the version information of this module.

**Table 3-77. Adc\_GetVersionInfo Return Values**

Name	Description
structure	In case of no errors.

### 3.8.3.13 Function Adc\_Calibrate

Executes high accuracy calibration of a ADC HW unit.

**Details:**

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

**Return:** void.

**Note**

The function Service ID[hex]: 0x15.

**Implements:** Adc\_Calibrate\_Activity

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** void Adc\_Calibrate(Adc\_HwUnitType Unit, Adc\_CalibrationStatusType \*pStatus);

**Table 3-78. Adc\_Calibrate Arguments**

Type	Name	Direction	Description
Adc_HwUnitType	Unit	input	ADC Unit Id.
Adc_CalibrationStatusType*	pStatus	input	Status of the ADC HW unit calibration and list of failed and passed tests.

### 3.8.3.14 Function Adc\_SetClockMode

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

**Details:**

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

**Return:** Std\_ReturnType Standard return type.

**Implements:** Adc\_SetClockMode\_Activity

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** Std\_ReturnType Adc\_SetClockMode(Adc\_DualClockModeType ClockMode);

**Table 3-79. Adc\_SetClockMode Arguments**

Type	Name	Direction	Description
	eClockMode	input	Normal or Alternate mode.

**Table 3-80. Adc\_SetClockMode Return Values**

Name	Description
E_OK:	In case of successful settings.
E_NOT_OK:	In case of unsuccessful settings.

### 3.8.3.15 Function Adc\_SetChannel

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

#### Details:

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

#### **Note**

The function Service ID[hex]: 0x4B.

**Implements:** Adc\_SetChannel\_Activity

**Violates:** internal linkage vs external linkage.

**Prototype:** void Adc\_SetChannel(const Adc\_GroupType Group, const Adc\_GroupDefType \*pChannel, const uint16 \*pDelays, const uint32 u32Mask, const Adc\_ChannelIndexType NumberOfChannel);

**Table 3-81. Adc\_SetChannel Arguments**

Type	Name	Direction	Description
constAdc_GroupType	Group	input	Group Id.
constAdc_GroupDefType*	pChannel	input	Pointer to channel list array.
constAdc_ChannelIndexType	NumberOfChannel	input	Number of Channel.

### 3.8.3.16 Function Adc\_ReadGroup

Reads the group conversion results.

#### Details:

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).

**Return:** Std\_ReturnType Standard return type.

**Pre:** Preconditions as text description. Optional tag.

**Post:** Postconditions as text description. Optional tag.

**Note**

The function Autosar Service ID[hex]:  
0x04.Synchronous.Reentrant function.

**Implements:** Adc\_ReadGroup\_Activity

**Violates:** All declarations and definitions of objects or functions at file scope shall have internal linkage unless external linkage is required

**Prototype:** Std\_ReturnType Adc\_ReadGroup(Adc\_GroupType Group, Adc\_ValueGroupType \*DataBufferPtr);

**Table 3-82. Adc\_ReadGroup Arguments**

Type	Name	Direction	Description
Adc_GroupType	Group	input	Numeric ID of requested ADC Channel group.
	pDataBufferPtr	input	ADC results of all channels of the selected group are stored in the data buffer addressed with the pointer.

**Table 3-83. Adc\_ReadGroup Return Values**

Name	Description
E_OK:	Results are available and written to the data buffer.
E_NOT_OK:	No results are available or development error occurred.

## 3.8.4 Structs Reference

Data structures supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.3 Rev0001 .

### 3.8.4.1 Structure Adc\_CalibrationStatusType

Structure for calibration status.

**Declaration:**

```
typedef struct
{
    Std_ReturnType Adc_UnitSelfTestStatus
} Adc_CalibrationStatusType;
```



**Table 3-84. Structure Adc\_CalibrationStatusType member description**

Member	Description
Adc_UnitSelfTestStatus	Unit calibration result status.

### 3.8.4.2 Structure Adc\_ConfigType

Structure for Configuration data.

#### Details:

Data structure containing the set of configuration parameters required for initializing the ADC Driver and ADC HW Unit(s).

**Implements:** Adc\_ConfigType\_struct

#### **Declaration:**

```
typedef struct
{
    const Adc_Adc12bsarv2_HwUnitConfigurationType* pAdc,
    const Adc_GroupConfigurationType* pGroups,
    const Adc_Adc12bsarv2_ChannelConfigurationType** pChannels,
    Adc_GroupType GroupCount,
    const Adc_Adc12bsarv2_MultiConfigType Misc
} Adc_ConfigType;
```

**Table 3-85. Structure Adc\_ConfigType member description**

Member	Description
pAdc	Hw unit configurations.
pGroups	Group configurations.
pChannels	Channel configurations.
GroupCount	Total number of groups.
Misc	Miscellaneous configuration parameters.

### 3.8.4.3 Structure Adc\_UnitStatusType

Structure for hardware unit status.

#### Details:

This structure contains the HW unit status information.

**Declaration:**

```
typedef struct
{
    Adc_QueueIndexType SwNormalQueueIndex,
    Adc_GroupType SwNormalQueue[ADC_QUEUE_MAX_DEPTH_MAX],
    Adc_GroupType HwInjectedQueue[ADC_HW_QUEUE],
    Adc_QueueIndexType HwInjectedQueueIndex,
    Adc_GroupType HwNormalQueue[ADC_HW_QUEUE],
    Adc_QueueIndexType HwNormalQueueIndex,
    Adc_MhtGroupType eHwQueueGroupType,
    Adc_GroupType SwInjectedQueue[1],
    Adc_QueueIndexType SwInjectedQueueIndex,
    boolean bCtuControlOngoing,
    Adc_ChannelIndexType NumSegment,
    uint8 u8Sc1Used
} Adc_UnitStatusType;
```

**Table 3-86. Structure Adc\_UnitStatusType member description**

Member	Description
SwNormalQueueIndex	Filled slots in the queue.
SwNormalQueue	Queued groups indexes, always executing Queue[0].
HwInjectedQueue	The depth of the hardware injected queue.
HwInjectedQueueIndex	Filled slots in the Hw injected queue.
HwNormalQueue	The depth of the hardware normal queue.
HwNormalQueueIndex	Filled slots in the Hw normal queue.
eHwQueueGroupType	When != 0 indicate MHT groups in queue, else regular.
SwInjectedQueue	The depth of the software injected queue.
SwInjectedQueueIndex	Filled slots in the Sw injected queue.
bCtuControlOngoing	Indicates Ctu control mode is ongoing
NumSegment	Indicates the number of channel segments of group
u8Sc1Used	Indicates the number of SC1 register is used.

**3.8.4.4 Structure Adc\_GroupStatusType**

Structure for group status.

**Details:**

This structure contains the group status information.

**Declaration:**

```
typedef struct
{
    Adc_StatusType eConversion,
    Adc_GroupConversionStateType eAlreadyConverted,
    Adc_HwTriggeringType eHwTriggering,
```

```

        Adc_HwTriggeringType eCtuTriggering,
        Adc_NotificationType eNotification,
        Adc_StreamNumSampleType ResultIndex,
        boolean bLimitCheckFailed
    } Adc_GroupStatusType;

```

**Table 3-87. Structure Adc\_GroupStatusType member description**

Member	Description
eConversion	Group status.
eAlreadyConverted	Group was previously converted or not.
eHwTriggering	hw trigger enabled/disabled
eCtuTriggering	CTU trigger enabled/disabled.
eNotification	notification enabled/disabled
ResultIndex	index into streaming buffer that is currently being filled
bLimitCheckFailed	check limit check fail

### 3.8.4.5 Structure Adc\_GroupConfigurationType

Structure for group configuration.

#### Declaration:

```

typedef struct
{
    const Adc_HwUnitType HwUnit,
    const Adc_GroupAccessModeType eAccessMode,
    const Adc_GroupConvModeType eMode,
    const Adc_GroupConvType eType,
    const Adc_GroupPriorityType Priority,
    const Adc_GroupReplacementType eReplacemementMode,
    const Adc_TriggerSourceType eTriggerSource,
    const Adc_MhtGroupType IsMHTGroup,
    const Adc_HwTriggerSignalType eTriggerEdge,
    const Adc_HwTriggerTimerType* pHwResource,
    const Adc_HwTriggerTimerType AssignedTriggerCount,
    const Adc_NotifyType Notification,
    const Adc_NotifyType ExtraNotification,
    const uint32 u32PrecisionChannel,
    const uint32 u32PrecisionPsr,
    const uint32 u32Wer0,
    Adc_ValueGroupType** pResultsBufferPtr,
    const Adc_StreamBufferModeType eBufferMode,
    const Adc_GroupType EnableChDisableChGroupIndex,
    const Adc_StreamNumSampleType NumSamples,
    const Adc_GroupDefType* pAssignment,
    const Adc_ChannelIndexType AssignedChannelCount,
    const Adc_ConversionTimeType ConvTime,
    const Adc_ConversionTimeType ConvTime1,
    const Adc_ConversionTimeType AltConvTime,
    const Adc_ConversionTimeType AltConvTime1,
    const Adc_ChannelType LastCh,
    const Adc_ChannelType FirstCh,
    const uint8 u8AdcWithoutInterrupt,
    const boolean bAdcDoubleBuffering
} Adc_GroupConfigurationType;

```

**Table 3-88. Structure `Adc_GroupConfigurationType` member description**

Member	Description
<code>HwUnit</code>	Hw unit to which the group belongs to.
<code>eAccessMode</code>	Access Mode.
<code>eMode</code>	Conversion Mode (OneShot/Continuous).
<code>eType</code>	Conversion type (Normal/Injected).
<code>Priority</code>	Priority of group.
<code>eReplcementMode</code>	Replacement Mode.
<code>eTriggerSource</code>	Hw/Sw trigger.
<code>IsMHTGroup</code>	Indicate the group type (Regular or MHT).
<code>eTriggerEdge</code>	Hardware trigger edge.
<code>pHwResource</code>	Resource of the selected hw trigger.
<code>AssignedTriggerCount</code>	Number of trigger source assigned to the group.
<code>Notification</code>	Pointer to notification function.
<code>ExtraNotification</code>	Pointer to extra notification function.
<code>u32PrecisionChannel</code>	ANP0_31, Precision configured channels.
<code>u32PrecisionPsr</code>	Presampling for Precision channels.
<code>u32Wer0</code>	Wer0 for precision channels.
<code>pResultsBufferPtr</code>	pointer to user result buffer array
<code>eBufferMode</code>	Buffer Mode.
<code>EnableChDisableChGroupIndex</code>	Group's index if it has the support to enable/disable channel.
<code>NumSamples</code>	Number of samples.
<code>pAssignment</code>	Assigned channels to group.
<code>AssignedChannelCount</code>	Number of channels.
<code>ConvTime</code>	Conversion time.
<code>ConvTime1</code>	Conversion time CTR1.
<code>AltConvTime</code>	Alternate Conversion time.
<code>AltConvTime1</code>	Alternate conversion time CTR1.
<code>LastCh</code>	Last channel configured.
<code>FirstCh</code>	First channel configured.
<code>u8AdcWithoutInterrupt</code>	Enables or Disables the ADC and DMA interrupts.
<code>bAdcDoubleBuffering</code>	Enables or Disables the ADC double buffering feature.

### 3.8.4.6 Structure `Adc_RuntimeGroupChannelType`

#### Declaration:

```
typedef struct
{
    constAdc_GroupDefType* pChannel,
    const uint16 * pul6Delays,
    uint32 u32Mask,
    Adc_ChannelIndexType ChannelCount,
```

```

        boolean bRuntimeUpdated
    } Adc_RuntimeGroupChannelType;

```

**Table 3-89. Structure Adc\_RuntimeGroupChannelType member description**

Member	Description
pChannel	Run time assigned channels to group.
pu16Delays	Run time assigned delay of channels group.
u32Mask	Mask per channel - to be updated or not.
ChannelCount	Run time number of channels.
bRuntimeUpdated	Indicates whether the configuration has been updated or not.

### 3.8.4.7 Structure Adc\_ChannelLimitCheckingType

#### Declaration:

```

typedef struct
{
    const boolean bChannelLimitCheck,
    const Adc_ChannelRangeSelectType eChannelRange,
    const Adc_ValueGroupType ChannelHighLimit,
    const Adc_ValueGroupType ChannelLowLimit
} Adc_ChannelLimitCheckingType;

```

**Table 3-90. Structure Adc\_ChannelLimitCheckingType member description**

Member	Description
bChannelLimitCheck	Channel limit checking feature.
eChannelRange	Range conversion.
ChannelHighLimit	High limit channel conversion value.
ChannelLowLimit	Low limit channel conversion value.

### 3.8.4.8 Structure Adc\_ValidationResultType

Structure for validation results.

#### Details:

This structure contains the validation information

#### Declaration:

```

typedef struct
{
    boolean bEndValidations,
    Std_ReturnType ValidParams
} Adc_ValidationResultType;

```

**Table 3-91. Structure Adc\_ValidationResultType member description**

Member	Description
bEndValidations	Signal if validation ended.
ValidParams	Return status.

### 3.8.5 Types Reference

Types supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.3 Rev0001 .

## 3.9 Symbolic Names Disclaimer

All containers having the symbolic name tag set as true in the Autosar schema will generate defines like:

```
#define <Container_Short_Name> <Container_ID>
```

For this reason it is forbidden to duplicate the name of such containers across the MCAL configuration, 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 Adc Driver. The most of the parameters are described below.

### 4.1 Configuration elements of Adc

Included forms :

- IMPLEMENTATION\_CONFIG\_VARIANT
- AdcGeneral
- AdcPublishedInformation
- NonAutosar
- AdcDemEventParameterRefs
- AdcInterrupt
- AdcConfigSet
- CommonPublishedInformation

### 4.2 Form IMPLEMENTATION\_CONFIG\_VARIANT

Configuration classes. Enable the parameters that are editable for specific configuration classes

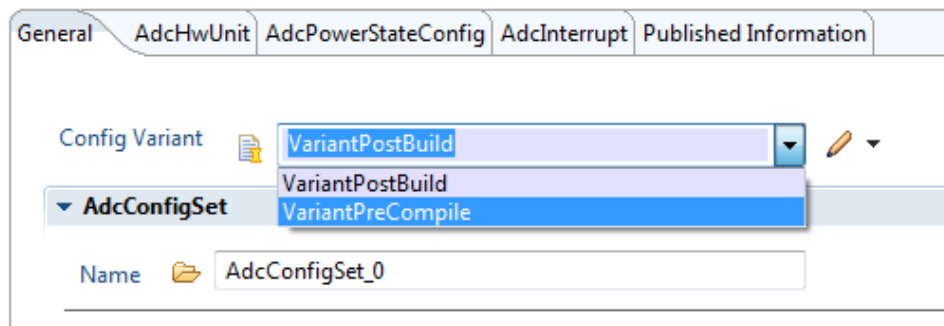


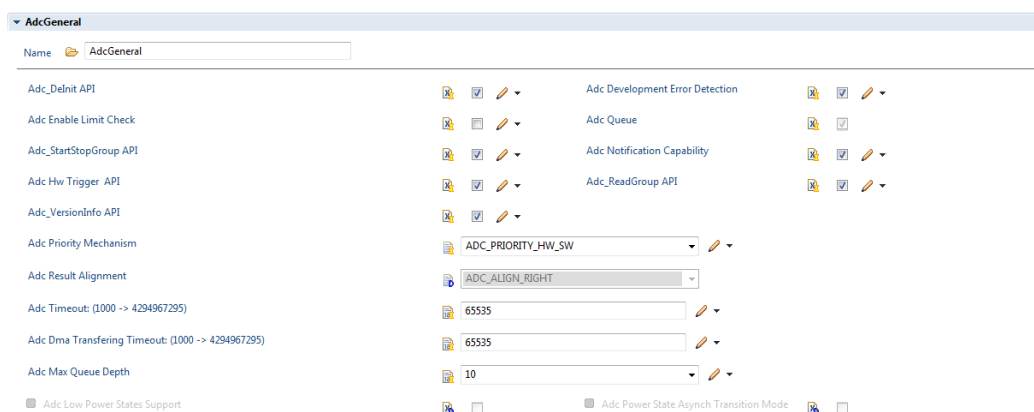
Figure 4-1. Tresos Plugin snapshot for IMPLEMENTATION\_CONFIG\_VARIANT form.

**Table 4-1. Attribute IMPLEMENTATION\_CONFIG\_VARIANT detailed description**

Property	Value
Label	Config Variant
Type	ENUMERATION
Default	VariantPostBuild
Range	VariantPostBuild VariantPreCompile

## 4.3 Form AdcGeneral

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

**Figure 4-2. TRESOS Plugin snapshot for AdcGeneral form.**

### 4.3.1 AdcDeInitApi (AdcGeneral)

Adds/removes the service Adc\_DeInit() from the code.

**Table 4-2. Attribute AdcDeInitApi (AdcGeneral) detailed description**

Property	Value
Label	Adc_DeInit API
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true



### 4.3.2 AdcDevErrorDetect (AdcGeneral)

Enable/Disable Development Error Detection.

**Table 4-3. Attribute AdcDevErrorDetect (AdcGeneral) detailed description**

Property	Value
Label	Adc Development Error Detection
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.3 AdcEnableLimitCheck (AdcGeneral)

Enable/disable limit checking feature in the ADC driver.

**Table 4-4. Attribute AdcEnableLimitCheck (AdcGeneral) detailed description**

Property	Value
Label	Adc Enable Limit Check
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.4 AdcEnableQueuing (AdcGeneral)

Enable/Disable the Queue. Note that if AdcPriorityImplementation=ADC\_PRIORITY\_HW\_SW this field is always enabled.

**Table 4-5. Attribute AdcEnableQueuing (AdcGeneral) detailed description**

Property	Value
Label	Adc Queue
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.5 AdcPriorityQueueMaxDepth (AdcGeneral)

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

**Table 4-6. Attribute AdcPriorityQueueMaxDepth (AdcGeneral) detailed description**

Property	Value
Label	Adc Max Queue Depth
Type	INTEGER
Origin	NXP
Symbolic Name	false
Default	1
Invalid	Range <div>&lt;=1024</div> <div>&gt;=1</div>

### 4.3.6 AdcEnableStartStopGroupApi (AdcGeneral)

Adds/removes the services Adc\_StartGroupConversion() and Adc\_StopGroupConversion from the code.

**Table 4-7. Attribute AdcEnableStartStopGroupApi (AdcGeneral) detailed description**

Property	Value
Label	Adc_StartStopGroup API
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.7 AdcGrpNotifCapability (AdcGeneral)

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

**Table 4-8. Attribute AdcGrpNotifCapability (AdcGeneral) detailed description**

Property	Value
Label	Adc Notification Capability

*Table continues on the next page...*

**Table 4-8. Attribute AdcGrpNotifCapability (AdcGeneral) detailed description (continued)**

Property	Value
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.8 AdcHwTriggerApi (AdcGeneral)

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

**Table 4-9. Attribute AdcHwTriggerApi (AdcGeneral) detailed description**

Property	Value
Label	Adc Hw Trigger API
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

### 4.3.9 AdcPriorityImplementation (AdcGeneral)

Select the Priority mechanism. In this version the `ADC_PRIORITY_HW` isn't used.

**Table 4-10. Attribute AdcPriorityImplementation (AdcGeneral) detailed description**

Property	Value
Label	Adc Priority Mechanism
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_PRIORITY_NONE
Range	ADC_PRIORITY_HW ADC_PRIORITY_HW_SW ADC_PRIORITY_NONE

### 4.3.10 AdcReadGroupApi (AdcGeneral)

Adds / removes the service Adc\_ReadGroup() from the code.

**Table 4-11. Attribute AdcReadGroupApi (AdcGeneral) detailed description**

Property	Value
Label	Adc_ReadGroup API
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.11 AdcResultAlignment (AdcGeneral)

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

This feature is not supported by S32K14X's hardware.

**Table 4-12. Attribute AdcResultAlignment (AdcGeneral) detailed description**

Property	Value
Label	Adc Result Alignment
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_ALIGN_RIGHT
Range	ADC_ALIGN_LEFT ADC_ALIGN_RIGHT

### 4.3.12 AdcVersionInfoApi (AdcGeneral)

Adds / removes the service Adc\_GetVersionInfo() from the code.

**Table 4-13. Attribute AdcVersionInfoApi (AdcGeneral) detailed description**

Property	Value
Label	Adc_VersionInfo API
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

### 4.3.13 AdcTimeout (AdcGeneral)

This is a timeout value which is used to wait till - ADC hardware is disabled - Conversion completed in calibration feature. If the Status is not updated then after this timeout the ADC\_E\_TIMEOUT production error will be reported and the rest of the functionality will be skipped.

**Table 4-14. Attribute AdcTimeout (AdcGeneral) detailed description**

Property	Value
Label	Adc Timeout:
Type	INTEGER
Origin	NXP
Symbolic Name	false
Default	65535
Invalid	Range <=4294967295 >=1000

### 4.3.14 AdcLowPowerStatesSupport (AdcGeneral)

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. This parameter is disabled, there is no power management support implemented for this platform.

This is not used by the current implementation

**Table 4-15. Attribute AdcLowPowerStatesSupport (AdcGeneral) detailed description**

Property	Value
Label	Adc Low Power States Support
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

### 4.3.15 AdcPowerStateAsynchTransitionMode (AdcGeneral)

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

This is not used by the current implementation

**Table 4-16. Attribute AdcPowerStateAsynchTransitionMode (AdcGeneral) detailed description**

Property	Value
Label	Adc Power State Asynch Transition Mode
Optional	True
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

### 4.3.16 Form AdcPowerStateConfig

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

This is not used by the current implementation

**Is included by form :** [Form AdcGeneral](#)

#### 4.3.16.1 AdcPowerState (AdcPowerStateConfig)

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.

This is not used by the current implementation

**Table 4-17. Attribute AdcPowerState (AdcPowerStateConfig) detailed description**

Property	Value
Type	INTEGER

*Table continues on the next page...*

**Table 4-17. Attribute AdcPowerState (AdcPowerStateConfig) detailed description (continued)**

Property	Value
Origin	AUTOSAR_ECUC
Symbolic Name	true
Default	0
Invalid	Range $\leq 18446744073709551615$ $\geq 0$

#### 4.3.16.2 AdcPowerStateReadyCbkRef (AdcPowerStateReadyCbkRefConfig)

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.

This is not used by the current implementation

**Table 4-18. Attribute AdcPowerStateReadyCbkRef (AdcPowerStateReadyCbkRefConfig) detailed description**

Property	Value
Type	FUNCTION-NAME
Origin	AUTOSAR_ECUC
Symbolic Name	true
Default	NULL_PTR

## 4.4 Form 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.

▼ AdcPublishedInformation

Name AdcPublishedInformation

---

Adc Channel Value Signed false

Adc Group First Channel Fixed false

Adc Max Channel Resolution 12

**Figure 4-3. Tresos Plugin snapshot for AdcPublishedInformation form.**

#### 4.4.1 AdcChannelValueSigned (AdcPublishedInformation)

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.

**Table 4-19. Attribute AdcChannelValueSigned (AdcPublishedInformation) detailed description**

Property	Value
Label	Adc Channel Value Signed
Type	BOOLEAN_LABEL
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

#### 4.4.2 AdcGroupFirstChannelFixed (AdcPublishedInformation)

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.

**Table 4-20. Attribute AdcGroupFirstChannelFixed (AdcPublishedInformation) detailed description**

Property	Value
Label	Adc Group First Channel Fixed
Type	BOOLEAN_LABEL
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false



### 4.4.3 AdcMaxChannelResolution (AdcPublishedInformation)

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

**Table 4-21. Attribute AdcMaxChannelResolution (AdcPublishedInformation) detailed description**

Property	Value
Label	Adc Max Channel Resolution
Type	INTEGER_LABEL
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	12
Invalid	Range <=63 >=1

## 4.5 Form NonAutosar

Non Autosar API settings.

The screenshot shows the 'NonAutosar' configuration form in the Tresos Plugin. It features a search bar at the top with the text 'NonAutosar'. Below the search bar, there are two columns of configuration options, each with a name, a status icon (checkbox or circle), and a dropdown menu (arrow). The options are as follows:

Option Name	Status	Action
Adc Channel Indexes Symbolic Names	<input type="checkbox"/>	Dropdown
Adc Enable Calibration API	<input checked="" type="checkbox"/>	Dropdown
Adc Initial Notification Capability	<input type="checkbox"/>	Dropdown
Adc Conversion Time Once	<input checked="" type="checkbox"/>	Dropdown
Adc Enable Double Buffering Optimization	<input type="checkbox"/>	Dropdown
Adc Use Hardware Normal Groups	<input checked="" type="checkbox"/>	Dropdown
Adc Bypass Consistency Loop	<input type="checkbox"/>	Dropdown
Adc Set Clock Mode API	<input type="checkbox"/>	Dropdown
Adc Set Channel API	<input type="checkbox"/>	Dropdown
Adc Disable Production Error Reporting	<input type="checkbox"/>	Dropdown
Adc Optimize OneShot HwTrigger Conversions	<input type="checkbox"/>	Dropdown
Adc Global Enable DMA Transfer	<input checked="" type="checkbox"/>	Dropdown
Enable Adc User Mode Support	<input type="checkbox"/>	Dropdown
Adc Continuous Without Interrupt Uses	<input type="checkbox"/>	Dropdown

**Figure 4-4. Tresos Plugin snapshot for NonAutosar form.**

### 4.5.1 AdcEnableGroupDependentChannelNames (NonAutosar)

#### Note

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.

**Table 4-22. Attribute `AdcEnableGroupDependentChannelNames` (NonAutosar) detailed description**

Property	Value
Label	Adc Channel Indexes Symbolic Names
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

## 4.5.2 `AdcEnableCalibration` (NonAutosar)

### Note

If this parameter has been configured to "TRUE", the Non-Autosar function "`Adc_Calibrate()`" shall be accessible, otherwise this function shall be removed from the code. This functionality is not available for this platform (not supported by HW).

This is an Implementation Specific Parameter.

**Table 4-23. Attribute `AdcEnableCalibration` (NonAutosar) detailed description**

Property	Value
Label	Adc Enable Calibration API
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false
Enable	false

## 4.5.3 `AdcEnableInitialNotification` (NonAutosar)

### Note

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

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

**Table 4-24. Attribute `AdcEnableInitialNotification (NonAutosar)` detailed description**

Property	Value
Label	Adc Initial Notification Capability
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

#### 4.5.4 `AdcConvTimeOnce (NonAutosar)`

##### Note

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.

**Table 4-25. `AdcConvTimeOnce. (NonAutosar)` detailed description**

Property	Value
Label	Adc Conversion Time Once
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

#### 4.5.5 `AdcEnableDoubleBufferingOptimization (NonAutosar)`

##### Note

Implementation Specific Parameter. Enable/Disable The Adc driver double buffering optimization for Streaming access mode.

**Table 4-26. Attribute AdcEnableDoubleBufferingOptimization (NonAutosar) detailed description**

Property	Value
Label	Adc Enable Double Buffering Optimization
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

### 4.5.6 AdcEnableDualClockMode (NonAutosar)

#### Note

Adds/removes the Dual Clock mode service Adc\_SetClockMode from the code. Also it enables the programming of Conversion Timing registers in Adc\_SetClockMode.

**Table 4-27. AdcEnableDualClockMode. (NonAutosar) detailed description**

Property	Value
Label	Adc Set Clock Mode API
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false
Enable	false

### 4.5.7 AdcEnableSetChannel (NonAutosar)

#### Note

If this parameter has been configured to "TRUE", the Non-Autosar function "Adc\_SetChannel()" shall be accessible, otherwise this function shall be removed from the code. This is an Implementation Specific Parameter.

**Table 4-28. Attribute AdcEnableSetChannel (NonAutosar) detailed description**

Property	Value
Label	Adc Set Channel API
Type	BOOLEAN
Origin	NXP

*Table continues on the next page...*

**Table 4-28. Attribute AdcEnableSetChannel (NonAutosar) detailed description (continued)**

Property	Value
Symbolic Name	false
Default	false

## 4.5.8 AdcDisableDemReportErrorStatus (NonAutosar)

### Note

Enable/Disable Dem error reporting.

**Table 4-29. AdcDisableDemReportErrorStatus. (NonAutosar) detailed description**

Property	Value
Label	Adc Disable Production Error Reporting
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false
Enable	false

## 4.5.9 AdcOptimizeOneShotHwTriggerConversions (NonAutosar)

### Note

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.

**Table 4-30. AdcOptimizeOneShotHwTriggerConversions. (NonAutosar) detailed description**

Property	Value
Label	Adc Optimize OneShot HwTrigger Conversions
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

## 4.5.10 AdcEnableDmaTrasferMode (NonAutosar)

### Note

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

**Table 4-31. AdcEnableDmaTrasferMode. (NonAutosar) detailed description**

Property	Value
Label	Adc Global Enable DMA Transfer
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

## 4.5.11 AdcEnableUserModeSupport (NonAutosar)

### Note

When this parameter is enabled, the Adc module will adapt to run from User Mode. Note: No special measures need to be taken to run ADC driver from user mode; The Adc driver code can be executed at any time from both supervisor and user mode.

**Table 4-32. AdcEnableUserModeSupport. (NonAutosar) detailed description**

Property	Value
Label	Enable Adc User Mode Support
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

## 4.5.12 AdcContinuousWithoutInterrupt (NonAutosar)

### Note

This parameter globally enables the possibility to configure continuous without interrupt. This is an Implementation Specific Parameter.

**Table 4-33. AdcContinuousWithoutInterrupt. (NonAutosar) detailed description**

Property	Value
Label	Adc Continuous Without Interrupt Uses
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

### 4.5.13 AdcUseHardwareNormalGroups(NonAutosar)

#### Note

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. This is an Implementation Specific Parameter.

**Table 4-34. AdcUseHardwareNormalGroups. (NonAutosar) detailed description**

Property	Value
Label	Double Buffering Config More Than One Channel
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

### 4.5.14 AdcBypassConsistencyLoop(NonAutosar)

#### Note

This is used to increase ADC performances. If checking the HW-SW coherency is no longer guaranteed by the driver, the user must make sure he does not call a ADC service before the HW reaches the correct state.

**Table 4-35. AdcBypassConsistencyLoop. (NonAutosar) detailed description**

Property	Value
Label	Adc Bypass Consistency Loop
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

## 4.6 Form AdcDemEventParameterRefs

Container for the references to DemEventParameter elements which shall be invoked using the API Dem\_SetEventStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.

**Figure 4-5. Tresos Plugin snapshot for AdcDemEventParameterRefs form.**

### 4.6.1 ADC\_E\_TIMEOUT (AdcDemEventParameterRefs)

Reference to configured DEM event to report "Timeout failure".

**Table 4-36. Attribute ADC\_E\_TIMEOUT (AdcDemEventParameterRefs) detailed description**

Property	Value
Label	Adc Timeout Dem Error
Type	SYMBOLIC-NAME-REFERENCE
Origin	NXP
Optional	true



## 4.7 Form 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.

Figure 4-6. Tresos Plugin snapshot for AdcInterrupt form.

### 4.7.1 AdcInterruptSource (AdcInterrupt)

The name of the interrupt.

- Note: Implementation Specific Parameter.

Table 4-37. Attribute AdcInterruptSource (AdcInterrupt) detailed description

Property	Value
Label	Adc Interrupt Name
Type	STRING
Origin	NXP
Symbolic Name	false

### 4.7.2 AdcInterruptEnable (AdcInterrupt)

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

**Table 4-38. Attribute AdcInterruptEnable (AdcInterrupt) detailed description**

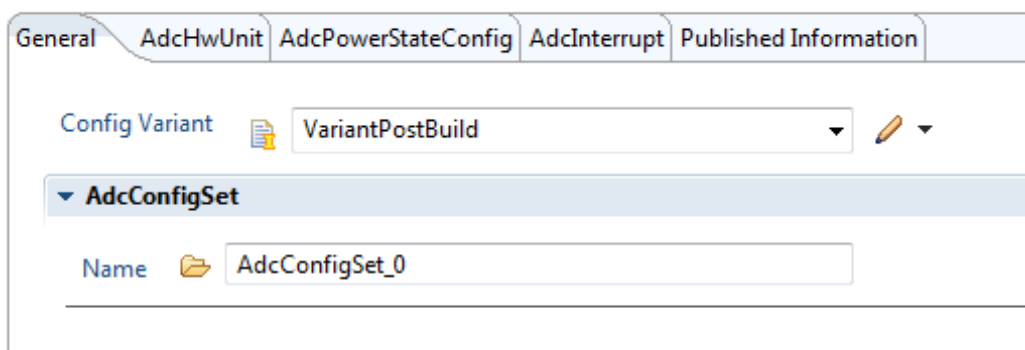
Property	Value
Label	Adc Interrupt Enable
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

## 4.8 Form AdcConfigSet

This is the base container that contains the post-build selectable configuration parameters

**Included forms :**

- [Form AdcHwUnit](#)



**Figure 4-7. TRESOS Plugin snapshot for AdcConfigSet form.**

### 4.8.1 Form AdcHwUnit

This container contains the Driver configuration (parameters) depending on grouping of channels.

**Is included by form :** [Form AdcConfigSet](#)

**Included forms :**

- [Form AdcChannel](#)
- [Form AdcGroup](#)
- [Form AdcPdbSettings](#)
- [Form AdcNormalConversionTimings](#)
- [Form AdcAlternateConversionTimings](#)

AdcHwUnit

Name

AdcHwUnit\_1

General

AdcChannel

AdcGroup

Adc Transfer Type

ADC\_INTERRUPT

Adc Source Clock

ADC\_ALTCLK1

Adc Hardware Unit

ADC1

Adc Logical Unit Id

0

Adc Voltage Reference Selection

VREFH\_VREFL

Adc Prescaler Value (0 -> 65535)

1

Adc Resolution

BITS\_12

Adc Offset Correction Value (0 -> 255)

0

AdcPdbSettings

AdcNormalConversionTimings

AdcAlternateConversionTimings

Figure 4-8. Tresos Plugin snapshot for AdcHwUnit form.

4.8.1.1 AdcTransferType (AdcHwUnit)

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

Table 4-39. Attribute AdcTransferType (AdcHwUnit) detailed description

Property	Value
Label	Adc Transfer Type
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_INTERRUPT
Range	ADC_DMA ADC_INTERRUPT

### 4.8.1.2 AdcClockSource (AdcHwUnit)

Not used. This value should be selected in the MCU Driver.

**Table 4-40. Attribute AdcClockSource (AdcHwUnit) detailed description**

Property	Value
Label	Adc Source Clock
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_ALTCLK1
Enable	false
Range	ADC_ALTCLK1 ADC_ALTCLK2 ADC_ALTCLK3 ADC_ALTCLK4

### 4.8.1.3 AdcHwUnitId (AdcHwUnit)

Specifies the used ADC Hardware Unit.

**Table 4-41. Attribute AdcHwUnitId (AdcHwUnit) detailed description**

Property	Value
Label	Adc Hardware Unit
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Range	ADC0 ADC1

### 4.8.1.4 AdcLogicalUnitId (AdcHwUnit)

Specifies the used ADC Hardware Unit.

**Table 4-42. Attribute AdcLogicalUnitId (AdcHwUnit) detailed description**

Property	Value
Label	Adc Logical Unit Id
Type	INTEGER

*Table continues on the next page...*

**Table 4-42. Attribute AdcLogicalUnitId (AdcHwUnit) detailed description (continued)**

Property	Value
Origin	AUTOSAR_ECUC
Symbolic Name	false
Invalid	<=1 >=0

#### 4.8.1.5 AdcVoltageReferenceSelection (AdcHwUnit)

Selects the voltage reference source used for conversions. Refer to the chip configuration chapter's ADC section for details of these field definitions for this chip.

- VREFH\_VREFL - Selects VREFH/VREFL as reference voltage.
- VALTH\_VALTL - Selects VALTH\_VREFL as reference voltage.

**Table 4-43. Attribute AdcVoltageReferenceSelection (AdcHwUnit) detailed description**

Property	Value
Label	Adc Voltage Reference Selection
Type	STRING
Origin	NXP
Symbolic Name	false
Default	VREFH_VREFL
Range	VREFH_VREFL VALTH_VREFL

#### 4.8.1.6 AdcPrescale (AdcHwUnit)

Not used, use the AdcClockDivideSelect instead.

**Table 4-44. Attribute AdcPrescale (AdcHwUnit) detailed description**

Property	Value
Label	Adc Prescaler Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	2
Invalid	Range <=65535 >=0

#### 4.8.1.7 AdcResolution (AdcHwUnit)

Used to set the ADC resolution mode.

- BITS\_8 - 8-bit conversion.
- BITS\_10 - 10-bit conversion.
- BITS\_12 - 12-bit conversion.

**Table 4-45. Attribute AdcResolution (AdcHwUnit) detailed description**

Property	Value
Label	Adc Resolution
Type	STRING
Origin	NXP
Symbolic Name	false
Default	BITS_12
Range	BITS_8 BITS_10 BITS_12

#### 4.8.1.8 AdcOffsetCorrectionValue (AdcHwUnit)

Value to be written in the ADC\_OFS(Offset correction value register). This value will be added with the converted result value to generate final result to be loaded into ADC\_Rn.

**Table 4-46. Attribute AdcOffsetCorrectionValue (AdcHwUnit) detailed description**


Property	Value
Label	Adc Offset Correction Value
Type	INTEGER
Origin	NXP
Symbolic Name	false
Default	0
Range	>=0 <=255

#### 4.8.1.9 Form AdcChannel

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



**Is included by form :** [Form AdcHwUnit](#)



**AdcChannel**


Name  AdcChannel\_0\_0



---



**General**



Adc Logical Channel ID  0 



Adc Hardware Channel Id  AN\_0 



☒ Adc Channel Limit Check  ☐



☒ Adc Channel High Limit  2 



☒ Adc Channel Low Limit  1 



☒ Adc Channel Range Select  ADC\_RANGE\_UNDER\_LOW 

☐ Adc Channel Conversion Time (0 -> 9223372036854775807)  0 

☐ Adc High Reference Voltage  UPPER\_REF\_VOLT\_0 

☐ Adc Low Reference Voltage  LOWER\_REF\_VOLT\_0 

☐ Adc Channel Resolution (1 -> 63)  12 

☐ Adc Channel Sampling time (8 -> 254)  8 

**Figure 4-9. Tresos Plugin snapshot for AdcChannel form.**

#### 4.8.1.9.1 AdcLogicalChannelId (AdcChannel)

This is the logical Id of the ADC channel.

**Table 4-47. Attribute AdcLogicalChannelId (AdcChannel) detailed description**

Property	Value
Label	Adc Logical Channel ID
Type	INTEGER
Origin	NXP
Symbolic Name	false
Invalid	Range <div>&lt;=1024</div> <div>&gt;=0</div>

#### 4.8.1.9.2 AdcChannelId (AdcChannel)

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.

**Table 4-48. Attribute AdcChannelId (AdcChannel) detailed description**

Property	Value
Label	Adc Hardware Channel Id

*Table continues on the next page...*

**Table 4-48. Attribute AdcChannelId (AdcChannel) detailed description (continued)**

Property	Value
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false

#### 4.8.1.9.3 AdcChannelLimitCheck (AdcChannel)

Enables or disables limit checking for an ADC channel.

**Table 4-49. Attribute AdcChannelLimitCheck (AdcChannel) detailed description**

Property	Value
Label	Adc Channel Limit Check
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	false

#### 4.8.1.9.4 AdcChannelHighLimit (AdcChannel)

High limit - used for limit checking.

**Table 4-50. Attribute AdcChannelHighLimit (AdcChannel) detailed description**

Property	Value
Label	Adc Channel High Limit
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0
Invalid	Range <=9223372036854775807 >=0

#### 4.8.1.9.5 AdcChannelLowLimit (AdcChannel)

Low limit - used for limit checking.



**Table 4-51. Attribute AdcChannelLowLimit (AdcChannel) detailed description**

Property	Value
Label	Adc Channel Low Limit
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0
Invalid	Range <=9223372036854775807 >=0

#### 4.8.1.9.6 AdcChannelRangeSelect (AdcChannel)

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

**Table 4-52. Attribute AdcChannelRangeSelect (AdcChannel) detailed description**

Property	Value
Label	Adc Channel Range Select
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_RANGE_ALWAYS
Range	ADC_RANGE_ALWAYS ADC_RANGE_BETWEEN ADC_RANGE_NOT_BETWEEN ADC_RANGE_NOT_OVER_HIGH ADC_RANGE_NOT_UNDER_LOW ADC_RANGE_OVER_HIGH ADC_RANGE_UNDER_LOW

#### 4.8.1.9.7 AdcChannelConvTime (AdcChannel)

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.

This parameter is not used by the current implementation.

**Table 4-53. Attribute AdcChannelConvTime (AdcChannel) detailed description**

Property	Value
Label	Adc Channel Conversion Time

*Table continues on the next page...*

**Table 4-53. Attribute AdcChannelConvTime (AdcChannel) detailed description (continued)**

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0
Enable	false
Invalid	Range <=9223372036854775807 >=0

#### 4.8.1.9.8 AdcChannelRefVoltsrcHigh (AdcChannel)

Upper reference voltage source for each channel. This parameter is not used by the current implementation.

**Table 4-54. Attribute AdcChannelRefVoltsrcHigh (AdcChannel) detailed description**

Property	Value
Label	Adc High Reference Voltage
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	UPPER_REF_VOLT_0
Enable	false
Range	UPPER_REF_VOLT_0

#### 4.8.1.9.9 AdcChannelRefVoltsrcLow (AdcChannel)

Lower reference voltage source for each channel. This parameter is not used by the current implementation.

**Table 4-55. Attribute AdcChannelRefVoltsrcLow (AdcChannel) detailed description**

Property	Value
Label	Adc Low Reference Voltage
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	LOWER_REF_VOLT_0
Enable	false
Range	LOWER_REF_VOLT_0

#### 4.8.1.9.10 AdcChannelResolution (AdcChannel)

Channel Resolution in bits of converted value. It's fixed to 12 bits.

**Table 4-56. Attribute AdcChannelResolution (AdcChannel) detailed description**

Property	Value
Label	Adc Channel Resolution
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	12
Invalid	Range <div>&lt;=63</div> <div>&gt;=1</div>

#### 4.8.1.9.11 AdcChannelSampTime (AdcChannel)

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

**Table 4-57. Attribute AdcChannelSampTime (AdcChannel) detailed description**

Property	Value
Label	Adc Channel Sampling time
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	8
Enable	false
Invalid	Range <div>&gt;=8</div> <div>&lt;=254</div>

#### 4.8.1.10 Form AdcGroup

This container contains the Group configuration (parameters).

**Is included by form :** [Form AdcHwUnit](#)

**Included forms :**

- [Form AdcGroupNormalConversionTimings](#)

## Form AdcConfigSet

- [Form AdcGroupDefinition](#)
- [Form AdcChannelDelay](#)

Figure 4-10. TRESOS Plugin snapshot for AdcGroup form.

### 4.8.1.10.1 AdcGroupAccessMode (AdcGroup)

Type of access mode to group conversion results.

Table 4-58. Attribute AdcGroupAccessMode (AdcGroup) detailed description

Property	Value
Label	Adc Group Access Mode
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_ACCESS_MODE_SINGLE
Range	ADC_ACCESS_MODE_SINGLE ADC_ACCESS_MODE_STREAMING

### 4.8.1.10.2 AdcGroupConversionMode (AdcGroup)

Type of Conversion mode of the channel group.

**Table 4-59. Attribute AdcGroupConversionMode (AdcGroup) detailed description**

Property	Value
Label	Adc Group Conversion Mode
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_CONV_MODE_ONESHOT
Range	ADC_CONV_MODE_CONTINUOUS ADC_CONV_MODE_ONESHOT

#### 4.8.1.10.3 AdcGroupConversionType (AdcGroup)

Normal or Injected conversion type.

**Table 4-60. Attribute AdcGroupConversionType (AdcGroup) detailed description**

Property	Value
Label	Adc Group Conversion Type
Type	STRING
Origin	NXP
Symbolic Name	false
Default	ADC_CONV_TYPE_NORMAL
Range	ADC_CONV_TYPE_NORMAL ADC_CONV_TYPE_INJECTED

#### 4.8.1.10.4 AdcGroupId (AdcGroup)

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.

**Table 4-61. Attribute AdcGroupId (AdcGroup) detailed description**

Property	Value
Label	Adc Group Id
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	true
Invalid	Range ≤1023 ≥0

#### 4.8.1.10.5 AdcGroupPriority (AdcGroup)

Priority level of the AdcGroup. This item is ignored if Adc/AdcGeneral/AdcPriorityImplementation is defined to ADC\_PRIORITY\_NONE.

**Table 4-62. Attribute AdcGroupPriority (AdcGroup) detailed description**

Property	Value
Label	Adc Group Priority
Optional	true
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0
Invalid	Range <=255 >=0

#### 4.8.1.10.6 AdcGroupReplacement (AdcGroup)

Replacement mechanism used on ADC group level, if a group conversion is interrupted by a group which has a higher priority. It's fixed to Abort/Restart

**Table 4-63. Attribute AdcGroupReplacement (AdcGroup) detailed description**

Property	Value
Label	Adc Group Replacement
Optional	true
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_GROUP_REPL_ABORT_RESTART
Enable	false
Range	ADC_GROUP_REPL_ABORT_RESTART ADC_GROUP_REPL_SUSPEND_RESUME

#### 4.8.1.10.7 AdcGroupTriggSrc (AdcGroup)

Type of source event that starts a group conversion. It's possible to select Hw or Sw trigger. In case of HW trigger source the actual HW trigger source will be selected by the "AdcHwTrigSrc" parameter and it can be any of the supported HW trigger sources available for this controller.

**Table 4-64. Attribute AdcGroupTriggSrc (AdcGroup) detailed description**

Property	Value
Label	Adc Group Trigger Source
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_TRIGG_SRC_SW
Range	ADC_TRIGG_SRC_HW ADC_TRIGG_SRC_SW

#### 4.8.1.10.8 AdcHwTrigSignal (AdcGroup)

Configures on which edge of the hardware trigger signal the driver should reach, i.e. start the conversion.

**Table 4-65. Attribute AdcHwTrigSignal (AdcGroup) detailed description**

Property	Value
Label	Adc Group Trigger Signal
Optional	true
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_HW_TRIG_RISING_EDGE
Range	ADC_HW_TRIG_FALLING_EDGE ADC_HW_TRIG_RISING_EDGE

#### 4.8.1.10.9 AdcHwTrigTimer (AdcGroup)

Reload value of the ADC module embedded timer. Isn't used on this version.

**Table 4-66. Attribute AdcHwTrigTimer (AdcGroup) detailed description**

Property	Value
Label	Adc Group Trigger Timer
Optional	true
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0
Enable	false
Invalid	Range

**Table 4-66. Attribute AdcHwTrigTimer (AdcGroup) detailed description**

Property	Value
	$\leq 4294967295$ $\geq 0$

#### 4.8.1.10.10 AdcNotification (AdcGroup)

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

**Table 4-67. Attribute AdcNotification (AdcGroup) detailed description**

Property	Value
Label	Adc Group Notification
Optional	true
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	NULL_PTR

#### 4.8.1.10.11 AdcExtraNotification (AdcGroup)

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.

**Table 4-68. Attribute AdcExtraNotification (AdcGroup) detailed description**

Property	Value
Label	Adc Group Extra Notification
Optional	true
Type	STRING
Origin	NXP
Symbolic Name	false
Default	NULL_PTR

#### 4.8.1.10.12 AdcStreamingBufferMode (AdcGroup)

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 ring buffer (wraps around if the end of the stream buffer is reached).



**Table 4-69. Attribute AdcStreamingBufferMode (AdcGroup) detailed description**

Property	Value
Label	Adc Group Streaming Buffer Mode
Type	STRING
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_STREAM_BUFFER_LINEAR
Range	ADC_STREAM_BUFFER_CIRCULAR ADC_STREAM_BUFFER_LINEAR

#### 4.8.1.10.13 AdcEnableDoubleBuffering (AdcGroup)

Enable/ Disable the Double Buffering feature for Adc group conversions. this Parameter can be configured only for groups configured with ADC\_ACCESS\_MODE\_STREAMING Access Mode, and only if ADC\_DMA is configured as the transfer method for the Adc Unit. When this parameter is Disabled, normal functionlaity shall be executed. Note: This is an Implementation Specific Parameter.

**Table 4-70. Attribute AdcEnableDoubleBuffering (AdcGroup) detailed description**

Property	Value
Label	Adc Group Enable Double Buffering
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false
Readonly	false

#### 4.8.1.10.14 AdcEnableHafIInterrupt (AdcGroup)

Enable/ Disable the interrupt when hafI sample complete for double buffering feature. Double buffering must be enable for configure this feature.

**Table 4-71. Attribute AdcEnableHafIInterrupt (AdcGroup) detailed description**

Property	Value
Label	Adc Group Enable HafI Interrupt
Type	BOOLEAN
Origin	NXP
Symbolic Name	false

*Table continues on the next page...*

**Table 4-71. Attribute AdcEnableHafInterrupt (AdcGroup) detailed description (continued)**

Property	Value
Default	false
Readonly	false

#### 4.8.1.10.15 AdcStreamingNumSamples (AdcGroup)

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

**Table 4-72. Attribute AdcStreamingNumSamples (AdcGroup) detailed description**

Property	Value
Label	Adc Group Streaming Number Samples
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	1
Invalid	Range <=1024 >=1

#### 4.8.1.10.16 AdcWithoutInterrupts (AdcGroup)

Enable/ Disable the occurring of ADC Interrupts and Reading of the group conversion results periodically without interrupts. A) When this parameter is enabled, interrupts are disabled . The conversion will run without software intervention (no interrupt generated anymore) and application can read the results only by calling Adc\_ReadGroup(). B) When this parameter is enabled, the result buffer is no longer to be used to read the results as the results will be directly read from HW registers. C) data transfer using DMA cannot be used when this parameter is enabled When this parameter is Disabled, normal functionlaity shall be executed. Note: This is an Implementation Specific Parameter.

**Table 4-73. Attribute AdcWithoutInterrupts (AdcGroup) detailed description**

Property	Value
Label	Adc Group Without Interrupts
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

#### 4.8.1.10.17 AdcGroupInBacktoBackMode (AdcGroup)

Enable/ Disable the channel conversions occurring in Back to Back mode. When this feature is enabled, PDB\_CH1n\_C1[BB] bits will be set for the group channels, from the second to the last. The first channel conversion complete will trigger the PDB channel second pretrigger, and so on until the last channel of the group. This ensures that the channel conversions occur in sequence. If this feature is not enabled for a group, all pretriggers of the PDB channel will be triggered at the same time. The coherence of channel conversions must be ensured via the delay configured for each pretrigger. If AdcGroupUsesChannelDelays is also configured for this group, the user can configure a delay for the first channel conversion, the rest will be converted in back to back mode (no delays).

**Table 4-74. Attribute AdcGroupInBacktoBackMode (AdcGroup) detailed description**

Property	Value
Label	Adc Group In Back to Back Mode
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

#### 4.8.1.10.18 AdcGroupUsesChannelDelays (AdcGroup)

Enable/ Disable the usage of channel delays. When this feature is enabled, PDB\_CH1nDLYx will be set with the values configured in AdcChannelDelay list for each channel. If AdcGroupInBacktoBackMode is set, the group conversions will work even without using channel delays.

**Table 4-75. Attribute AdcGroupUsesChannelDelays (AdcGroup) detailed description**

Property	Value
Label	Adc Group Uses Channel Delays
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

#### 4.8.1.10.19 AdcDelayNextPdb (AdcGroup)

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

**Table 4-76. Attribute AdcDelayNextPdb (AdcGroup) detailed description**

Property	Value
Label	Adc Delay Next PDB
Optional	true
Type	INTEGER
Origin	NXP
Symbolic Name	false
Default	0
Enable	false
Invalid	Range <div>&lt;=65535</div> <div>&gt;=0</div>

#### 4.8.1.10.20 AdcPdbPeriodContinuousMode (AdcGroup)

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

**Table 4-77. Attribute AdcPdbPeriodContinuousMode (AdcGroup) detailed description**

Property	Value
Label	Adc Pdb Period For Continous Mode
Optional	true
Type	INTEGER
Origin	NXP

*Table continues on the next page...*

**Table 4-77. Attribute AdcPdbPeriodContinuousMode (AdcGroup) detailed description (continued)**

Property	Value
Symbolic Name	false
Default	0
Enable	false
Invalid	Range <=65535 >=0

#### 4.8.1.11 Form AdcGroupNormalConversionTimings

Selects Normal values used for programming clock frequency, conversion timing and hardware averaging used at initialization and also by Adc\_SetClockMode API when it is called with parameter ADC\_NORMAL.

Is included by form : [Form AdcGroup](#)

**Figure 4-11. Tresos Plugin snapshot for AdcGroupNormalConversionTimings form.**

##### 4.8.1.11.1 AdcGroupHardwareAverageEnable (AdcGroupNormalConversionTimings)

Enables the hardware average function of the ADC.

**Table 4-78. Attribute AdcGroupHardwareAverageEnable (AdcGroupNormalConversionTimings) detailed description**

Property	Value
Label	Adc Hardware Average Enable
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

#### 4.8.1.11.2 AdcGroupHardwareAverageSelect (AdcGroupNormalConversionTimings)

Determines how many ADC conversions will be averaged to create the ADC average result. This functionality is activated when ADCx\_SC3[AVGE] = 1.

- SAMPLES\_4 - 4 samples averaged.
- SAMPLES\_8 - 8 samples averaged.
- SAMPLES\_16 - 16 samples averaged.
- SAMPLES\_32 - 32 samples averaged.

**Table 4-79. Attribute AdcGroupHardwareAverageSelect  
(AdcGroupNormalConversionTimings) detailed description**

Property	Value
Label	Adc Hardware Average Select
Type	STRING
Origin	NXP
Symbolic Name	false
Default	SAMPLES_4
Range	SAMPLES_4 SAMPLES_8 SAMPLES_16 SAMPLES_32

#### 4.8.1.11.3 AdcGroupSampleTimeDuration (AdcGroupNormalConversionTimings)

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

**Table 4-80. Attribute AdcGroupSampleTimeDuration  
(AdcGroupNormalConversionTimings) detailed description**

Property	Value
Label	Adc Sample Time Duration
Type	INTEGER
Origin	NXP
Symbolic Name	false
Default	2
Invalid	Range ≤256 ≥2

#### 4.8.1.11.4 AdcGroupClockDivideSelect (AdcGroupNormalConversionTimings)

Selects the divide ratio used by the ADC to generate the internal clock ADCK.

- DIV\_NONE - Input clock
- DIV\_2 - Input clock / 2
- DIV\_4 - Input clock / 4
- DIV\_8 - Input clock / 8

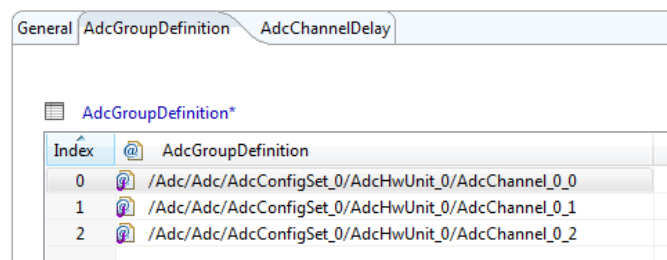
**Table 4-81. Attribute AdcGroupClockDivideSelect (AdcGroupNormalConversionTimings) detailed description**

Property	Value
Label	Adc Clock Divide Select
Type	STRING
Origin	NXP
Symbolic Name	false
Default	DIV_NONE
Range	DIV_NONE DIV_2 DIV_4 DIV_8

#### 4.8.1.12 Form AdcGroupDefinition

Is included by form : [Form AdcGroup](#)

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



**Figure 4-12. Tresos Plugin snapshot for AdcGroupDefinition form.**

**Table 4-82. Attribute AdcGroupDefinition (AdcGroupDefinition) detailed description**

Property	Value
Type	REFERENCE
Origin	AUTOSAR_ECUC

4.8.1.13 Form AdcChannelDelay

Is included by form : [Form AdcGroup](#)

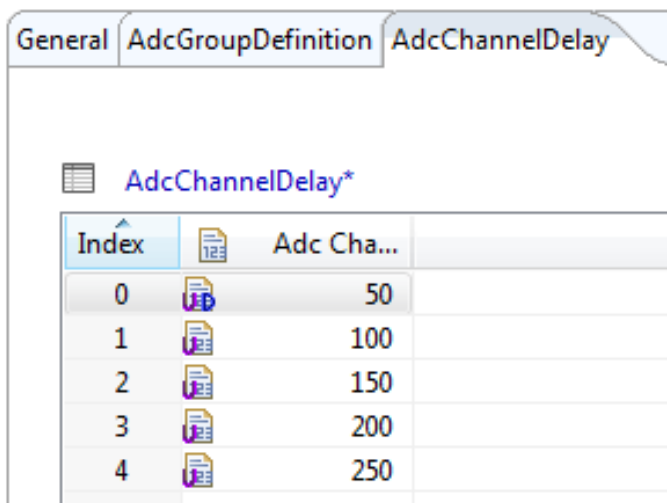


Figure 4-13. Tresos Plugin snapshot for AdcChannelDelay form.

Table 4-83. Attribute AdcChannelDelay (AdcChannelDelay) detailed description

Property	Value
Type	INTEGER
Origin	NXP
Label	Adc Channel Delay
Default	50
Invalid	Range <=65535 >=0

4.8.1.14 Form AdcPdbSettings

Confiugration for the PDB unit associated with this Adc hardware unit.

Is included by form : [Form AdcHwUnit](#)



AdcPdbSettings

Name AdcPdbSettings

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

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

PDB Sequence Error Interrupt Enable ☒

PDB Sequence Error Notification NULL\_PTR

**Figure 4-14. Tressos Plugin snapshot for AdcPdbSettings form.**

#### 4.8.1.14.1 AdcPdbPrescalerDividerSelect (AdcPdbSettings)

Determines the prescaler used for the PDB counter.

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

**Table 4-84. Attribute AdcPdbPrescalerDividerSelect (AdcPdbSettings) detailed description**

Property	Value
Label	Adc Pdb Prescaler Divider Select
Type	INTEGER
Origin	NXP
Symbolic Name	false
Default	0
Invalid	Range <div>&lt;=7</div> <div>&gt;=0</div>

#### 4.8.1.14.2 AdcPdbMultiplicationFactorSelect (AdcPdbSettings)

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

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

**Table 4-85. Attribute AdcPdbMultiplicationFactorSelect (AdcPdbSettings) detailed description**

Property	Value
Label	Adc Pdb Multiplication Factor Select for Prescaler
Type	INTEGER
Origin	NXP
Symbolic Name	false
Default	0
Invalid	Range <div>&lt;=3</div> <div>&gt;=0</div>

#### 4.8.1.14.3 AdcPdbChannelSequenceErrorEnable (AdcPdbSettings)

Enables the PDB sequence error interrupt. When this field is set, any of the PDB channel sequence error flags generates a PDB sequence error interrupt.

**Table 4-86. Attribute AdcPdbChannelSequenceErrorEnable (AdcPdbSettings) detailed description**

Property	Value
Label	PDB Sequence Error Interrupt Enable
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

#### 4.8.1.14.4 AdcPdbErrorNotification (AdcPdbSettings)

Callback function for PDB channel sequence error. This function pointer is called everytime when PDB channel sequence error occurred.

**Table 4-87. Attribute AdcPdbErrorNotification (AdcPdbSettings) detailed description**

Property	Value
Label	PDB Sequence Error Notification
Type	STRING
Origin	NXP
Symbolic Name	false
Default	NULL_PTR

#### 4.8.1.15 Form AdcNormalConversionTimings

Selects Normal values used for programming clock frequency, conversion timing and hardware averaging used at initialization and also by Adc\_SetClockMode API when it is called with parameter ADC\_NORMAL.

Is included by form : [Form AdcHwUnit](#)

**Figure 4-15. Tresos Plugin snapshot for AdcNormalConversionTimings form.**

##### 4.8.1.15.1 AdcGroupHardwareAverageEnable (AdcNormalConversionTimings)

Enables the hardware average function of the ADC.

**Table 4-88. Attribute AdcGroupHardwareAverageEnable (AdcNormalConversionTimings) detailed description**

Property	Value
Label	Adc Hardware Average Enable
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

#### 4.8.1.15.2 AdcGroupHardwareAverageSelect (AdcNormalConversionTimings)

Determines how many ADC conversions will be averaged to create the ADC average result. This functionality is activated when ADC\_CS3[AVGE] = 1.

- SAMPLES\_4 - 4 samples averaged.
- SAMPLES\_8 - 8 samples averaged.
- SAMPLES\_16 - 16 samples averaged.
- SAMPLES\_32 - 32 samples averaged.

**Table 4-89. Attribute AdcGroupHardwareAverageSelect (AdcNormalConversionTimings) detailed description**

Property	Value
Label	Adc Hardware Average Select
Type	STRING
Origin	NXP
Symbolic Name	false
Default	SAMPLES_4
Range SAMPLES_4 SAMPLES_8 SAMPLES_16 SAMPLES_32	

#### 4.8.1.15.3 AdcGroupSampleTimeDuration (AdcNormalConversionTimings)

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

**Table 4-90. Attribute AdcGroupSampleTimeDuration (AdcNormalConversionTimings) detailed description**

Property	Value
Label	Adc Sample Time Duration
Type	INTEGER
Origin	NXP
Symbolic Name	false
Default	2
Invalid	Range ≤256 ≥2

#### 4.8.1.15.4 AdcGroupClockDivideSelect (AdcNormalConversionTimings)

Selects the divide ratio used by the ADC to generate the internal clock ADCK.

- DIV\_NONE - Input clock
- DIV\_2 - Input clock / 2
- DIV\_4 - Input clock / 4
- DIV\_8 - Input clock / 8

**Table 4-91. Attribute AdcGroupClockDivideSelect (AdcNormalConversionTimings) detailed description**

Property	Value
Label	Adc Clock Divide Select
Type	STRING
Origin	NXP
Symbolic Name	false
Default	DIV_NONE
Range	DIV_NONE DIV_2 DIV_4 DIV_8

#### 4.8.1.16 Form AdcAlternateConversionTimings

Selects Alternate values used for programming clock frequency, conversion timing and hardware averaging used by Adc\_SetClockMode API when it is called with parameter ADC\_ALTERNATE. This is available when AdcEnableDualClockMode is enabled.

Is included by form : [Form AdcHwUnit](#)

**Figure 4-16. Tresos Plugin snapshot for AdcAlternateConversionTimings form.**

#### 4.8.1.16.1 AdcHardwareAverageEnableAlternate (AdcAlternateConversionTimings)

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

**Table 4-92. Attribute AdcHardwareAverageEnableAlternate  
(AdcAlternateConversionTimings) detailed description**

Property	Value
Label	Adc Hardware Average Enable
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

#### 4.8.1.16.2 AdcHardwareAverageSelectAlternate (AdcAlternateConversionTimings)

Determines how many ADC conversions will be averaged to create the ADC average result. This functionality is activated when ADC\_CS3[AVGE] = 1.

- SAMPLES\_4 - 4 samples averaged.
- SAMPLES\_8 - 8 samples averaged.
- SAMPLES\_16 - 16 samples averaged.
- SAMPLES\_32 - 32 samples averaged.

**Table 4-93. Attribute AdcHardwareAverageSelectAlternate  
(AdcAlternateConversionTimings) detailed description**

Property	Value
Label	Adc Hardware Average Select
Type	STRING
Origin	NXP
Symbolic Name	false
Default	SAMPLES_4
Range	SAMPLES_4 SAMPLES_8 SAMPLES_16 SAMPLES_32

#### 4.8.1.16.3 AdcSampleTimeDurationAlternate (AdcAlternateConversionTimings)

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

**Table 4-94. Attribute AdcSampleTimeDurationAlternate (AdcAlternateConversionTimings) detailed description**

Property	Value
Label	Adc Sample Time Duration
Type	INTEGER
Origin	NXP
Symbolic Name	false
Default	2
Invalid	Range <=256 >=2

#### 4.8.1.16.4 AdcClockDivideSelectAlternate (AdcAlternateConversionTimings)

Selects the divide ratio used by the ADC to generate the internal clock ADCK.

- DIV\_NONE - Input clock
- DIV\_2 - Input clock / 2
- DIV\_4 - Input clock / 4
- DIV\_8 - Input clock / 8

**Table 4-95. Attribute AdcClockDivideSelectAlternate (AdcAlternateConversionTimings) detailed description**


Property	Value
Label	Adc Clock Divide Select
Type	STRING
Origin	NXP
Symbolic Name	false
Default	DIV_NONE
Range	DIV_NONE DIV_2 DIV_4 DIV_8

## 4.9 Form CommonPublishedInformation


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

General AdcHwUnit AdcPowerStateConfig AdcInterrupt Published Information


▼ CommonPublishedInformation

Name  CommonPublishedInformation


ArReleaseMajorVersion

 4


ArReleaseMinorVersion

 3


ArReleaseRevisionVersion

 1


ModuleId

 123


SwMajorVersion

 1


SwMinorVersion

 0

SwPatchVersion

 1

VendorApilInfix



VendorId


 43

Figure 4-17. Tresos Plugin snapshot for CommonPublishedInformation form.

4.9.1 ArReleaseMajorVersion (CommonPublishedInformation)

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

Table 4-96. Attribute ArReleaseMajorVersion (CommonPublishedInformation) detailed description

Property	Value
Type	INTEGER_LABEL
Origin	NXP
Symbolic Name	false
Default	4
Invalid	Range >=4 <=4



### 4.9.2 ArReleaseMinorVersion (CommonPublishedInformation)

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

**Table 4-97. Attribute ArReleaseMinorVersion (CommonPublishedInformation) detailed description**

Property	Value
Type	INTEGER_LABEL
Origin	NXP
Symbolic Name	false
Default	3
Invalid	Range >=3 <=3

### 4.9.3 ArReleaseRevisionVersion (CommonPublishedInformation)

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

**Table 4-98. Attribute ArReleaseRevisionVersion (CommonPublishedInformation) detailed description**

Property	Value
Type	INTEGER_LABEL
Origin	NXP
Symbolic Name	false
Default	1
Invalid	Range >=1 <=1

### 4.9.4 ModuleId (CommonPublishedInformation)

Module ID of this module from Module List.

**Table 4-99. Attribute ModuleId (CommonPublishedInformation) detailed description**

Property	Value
Type	INTEGER_LABEL
Origin	NXP
Symbolic Name	false
Default	123
Invalid	Range <div>&gt;=123</div> <div>&lt;=123</div>

### 4.9.5 SwMajorVersion (CommonPublishedInformation)

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

**Table 4-100. Attribute SwMajorVersion (CommonPublishedInformation) detailed description**

Property	Value
Type	INTEGER_LABEL
Origin	NXP
Symbolic Name	false
Default	1
Invalid	Range <div>&gt;=1</div> <div>&lt;=1</div>

### 4.9.6 SwMinorVersion (CommonPublishedInformation)

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

**Table 4-101. Attribute SwMinorVersion (CommonPublishedInformation) detailed description**

Property	Value
Type	INTEGER_LABEL
Origin	NXP
Symbolic Name	false
Default	0

*Table continues on the next page...*

**Table 4-101. Attribute SwMinorVersion (CommonPublishedInformation) detailed description (continued)**

Property	Value
Invalid	Range >=0 <=0

### 4.9.7 SwPatchVersion (CommonPublishedInformation)

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

**Table 4-102. Attribute SwPatchVersion (CommonPublishedInformation) detailed description**

Property	Value
Type	INTEGER_LABEL
Origin	NXP
Symbolic Name	false
Default	1
Invalid	Range >=1 <=1

### 4.9.8 VendorApiInfix (CommonPublishedInformation)

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><Api name from SWS>. E.g. assuming that the VendorId of the implementor is 123 and the implementer chose a VendorApiInfix of "v11r456" a api name Adc\_Init defined in the SWS will translate to Adc\_123\_v11r456Init. This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

**Table 4-103. Attribute VendorApiInfix (CommonPublishedInformation) detailed description**

Property	Value
Type	STRING_LABEL

*Table continues on the next page...*

**Table 4-103. Attribute VendorApilnfix (CommonPublishedInformation) detailed description (continued)**

Property	Value
Origin	NXP
Symbolic Name	false
Default	
Enable	false

### 4.9.9 VendorId (CommonPublishedInformation)

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

**Table 4-104. Attribute VendorId (CommonPublishedInformation) detailed description**

Property	Value
Type	INTEGER_LABEL
Origin	NXP
Symbolic Name	false
Default	43
Invalid	Range >=43 <=43

## 4.10 Configuration element of Mcl (a dependent module)

Included form :

- TriggerMux

### 4.11 Form MclTriggerMux

Configuration of TriggerMux which refer to Adc.

▼ TriggerMux

Name

TriggerMux

TrgMux ADC_0 Input0	<div><div></div><div>PDB0_CH0</div><div>▼</div><div></div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux ADC_0 Input1	<div><div></div><div>INPUT_DISABLED</div><div>▼</div><div></div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux ADC_0 Input2	<div><div></div><div>INPUT_DISABLED</div><div>▼</div><div></div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux ADC_0 Input3	<div><div></div><div>INPUT_DISABLED</div><div>▼</div><div></div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux ADC_1 Lock Enabled	<div><div></div><div><div></div></div><div></div><div>▼</div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux ADC_1 Input0	<div><div></div><div>PDB1_CH0</div><div>▼</div><div></div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux ADC_1 Input1	<div><div></div><div>INPUT_DISABLED</div><div>▼</div><div></div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux ADC_1 Input2	<div><div></div><div>INPUT_DISABLED</div><div>▼</div><div></div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux ADC_1 Input3	<div><div></div><div>INPUT_DISABLED</div><div>▼</div><div></div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux ADC_1 Lock Enabled	<div><div></div><div><div></div></div><div></div><div>▼</div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux PDB0 Input0	<div><div></div><div>LPIT_CH0</div><div>▼</div><div></div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux PDB0 Lock Enabled	<div><div></div><div><div></div></div><div></div><div>▼</div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux PDB1 Input0	<div><div></div><div>LPIT_CH0</div><div>▼</div><div></div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux PDB1 Lock Enabled	<div><div></div><div><div></div></div><div></div><div>▼</div></div>	<div><div></div><div></div><div>▼</div></div>
TrgMux FLEXIO Input0	<div><div></div><div>INPUT_DISABLED</div><div>▼</div><div></div></div>	<div><div></div><div></div><div>▼</div></div>

Figure 4-18. Tresos Plugin snapshot for MclTriggerMux form (which refer to Adc)

4.11.1 TrgMux ADC\_0 Input0

Used to configure the Adc0 input 0.

Table 4-105. Attribute TrgMuxAdc0Input0 detailed description

Property	Value
Label	TrgMux ADC_0 Input0
Type	STRING
Origin	NXP
Default	INPUT_DISABLED

### 4.11.2 TrgMux ADC\_0 Input1

Used to configure the Adc0 input 1.

**Table 4-106. Attribute TrgMuxAdc0Input1 detailed description**

Property	Value
Label	TrgMux ADC_0 Input1
Type	STRING
Origin	NXP
Default	INPUT_DISABLED

### 4.11.3 TrgMux ADC\_0 Input2

Used to configure the Adc0 input 2.

**Table 4-107. Attribute TrgMuxAdc0Input2 detailed description**

Property	Value
Label	TrgMux ADC_0 Input2
Type	STRING
Origin	NXP
Default	INPUT_DISABLED

### 4.11.4 TrgMux ADC\_0 Input3

Used to configure the Adc0 input 3.

**Table 4-108. Attribute TrgMuxAdc0Input3 detailed description**

Property	Value
Label	TrgMux ADC_0 Input3
Type	STRING
Origin	NXP
Default	INPUT_DISABLED

### 4.11.5 TrgMux ADC\_0 Lock Enabled

Configures if register Adc0 must be locked(read-only).

**Table 4-109. Attribute TrgMuxAdc0LockEn detailed description**

Property	Value
Label	TrgMux ADC_0 Lock Enabled
Type	BOOLEAN
Origin	NXP
Default	false

### 4.11.6 TrgMux ADC\_1 Input0

Used to configure the Adc1 input 0.

**Table 4-110. Attribute TrgMuxAdc1Input0 detailed description**

Property	Value
Label	TrgMux ADC_1 Input0
Type	STRING
Origin	NXP
Default	INPUT_DISABLED

### 4.11.7 TrgMux ADC\_1 Input1

Used to configure the Adc1 input 1.

**Table 4-111. Attribute TrgMuxAdc1Input1 detailed description**

Property	Value
Label	TrgMux ADC_1 Input1
Type	STRING
Origin	NXP
Default	INPUT_DISABLED

### 4.11.8 TrgMux ADC\_1 Input2

Used to configure the Adc1 input 2.

**Table 4-112. Attribute TrgMuxAdc1Input2 detailed description**

Property	Value
Label	TrgMux ADC_1 Input2
Type	STRING
Origin	NXP
Default	INPUT_DISABLED

### 4.11.9 TrgMux ADC\_1 Input3

Used to configure the Adc1 input 3.

**Table 4-113. Attribute TrgMuxAdc1Input3 detailed description**

Property	Value
Label	TrgMux ADC_1 Input3
Type	STRING
Origin	NXP
Default	INPUT_DISABLED

### 4.11.10 TrgMux ADC\_1 Lock Enabled

Configures if register Adc1 must be locked(read-only).

**Table 4-114. Attribute TrgMuxAdc1LockEn detailed description**

Property	Value
Label	TrgMux ADC_1 Lock Enabled
Type	BOOLEAN
Origin	NXP
Default	false

### 4.11.11 TrgMux PDB0 Input0

Used to configure the Pdb0 input 0.



**Table 4-115. Attribute TrgMuxPdb0Input0 detailed description**

Property	Value
Label	TrgMux PDB0 Input0
Type	STRING
Origin	NXP
Default	INPUT_DISABLED

### 4.11.12 TrgMux PDB0 Lock Enabled

Configures if register Pdb0 must be locked(read-only).

**Table 4-116. Attribute TrgMuxPdb0LockEn detailed description**

Property	Value
Label	TrgMux PDB0 Lock Enabled
Type	BOOLEAN
Origin	NXP
Default	false

### 4.11.13 TrgMux PDB1 Input0

Used to configure the Pdb1 input 0.

**Table 4-117. Attribute TrgMuxPdb1Input0 detailed description**

Property	Value
Label	TrgMux PDB1 Input0
Type	STRING
Origin	NXP
Default	INPUT_DISABLED

### 4.11.14 TrgMux PDB1 Lock Enabled

Configures if register Pdb1 must be locked(read-only).

**Table 4-118. Attribute TrgMuxPdb1LockEn detailed description**

Property	Value
Label	TrgMux PDB1 Lock Enabled
Type	BOOLEAN
Origin	NXP
Default	false

## 4.12 Configuration element of Mcu (a dependent module)

Included form :

- McuSIMConfig
- McuSIMClockConfig

## 4.13 Form McuSIMConfig

Configures SIM\_ADCOPT, SIM\_CHIPCTL, SIM\_FCFG1, SIM\_FTMOPT0, SIM\_FTMOPT1 and SIM\_MISCTRL registers.

Note: Implementation specific Container.

Included forms :

- [Form McuChipControlConfiguration](#)
- [Form McuAdcOptionsConfiguration](#)

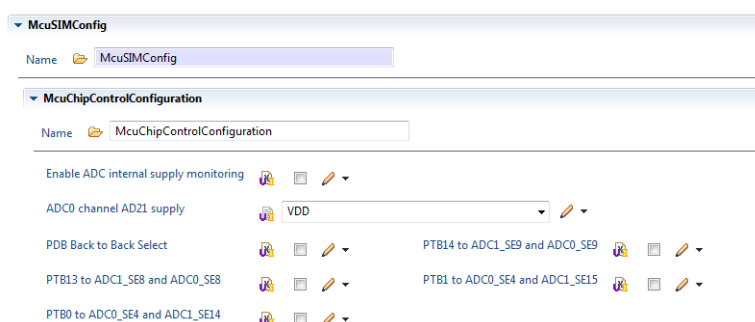


**Figure 4-19. Tressos Plugin snapshot for McuSIMConfig form.**

### 4.13.1 Form McuChipControlConfiguration

This container contains the configuration for the SIM\_CHIPCTL register.

Is included by form : [Form McuSIMConfig](#)



**Figure 4-20. Tressos Plugin snapshot for McuChipControlConfiguration form.**

#### 4.13.1.1 McuEnableAdcSupplyMonitoring (McuChipControlConfiguration)

SIM\_CHIPCTL[ADC\_SUPPLYEN] - Enable for internal supply monitoring on ADC0 channel AD21.

**Table 4-119. Attribute McuEnableAdcSupplyMonitoring (McuChipControlConfiguration) detailed description**

Property	Value
Label	Enable ADC internal supply monitoring
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

#### 4.13.1.2 McuAdcSupply (McuChipControlConfiguration)

SIM\_CHIPCTL[ADC\_SUPPLY] - Internal supplies monitored on ADC0 channel AD21.

**Table 4-120. Attribute McuAdcSupply (McuChipControlConfiguration) detailed description**

Property	Value
Label	ADC0 channel AD21 supply
Type	ENUMERATION
Origin	NXP
Symbolic Name	false
Default	VDD

*Table continues on the next page...*

**Table 4-120. Attribute McuAdcSupply (McuChipControlConfiguration) detailed description (continued)**

Property	Value
Range	VDD VDDA VREFH VDD_3V VDD_FLASH_3V VDD_LV

#### 4.13.1.3 McuPDBBackToBackSelect (McuChipControlConfiguration)

SIM\_CHIPCTL[PDB\_BB\_SEL] - PDB back-to-back select Selects ADC COCO source as pdb back-to-back mode. unchecked - PDB0 channel 0 back-to-back operation with ADC0 COCO[7:0]; PDB1 channel 0 back-to-back operation with ADC1 COCO[7:0]; PDB2 channel 0 back-to-back operation with ADC2 COCO[7:0]. checked - Channel 0 of PDB0,PDB1 back-to-back operation with COCO[7:0] of ADC0, ADC1 Note: Implementation Specific Parameter.

**Table 4-121. Attribute McuPDBBackToBackSelect (McuChipControlConfiguration) detailed description**

Property	Value
Label	PDB Back to Back Select
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

#### 4.13.1.4 McuPTB14InterleaveChannelSelect (McuChipControlConfiguration)

SIM\_CHIPCTL[INTERLEAVE\_SEL] - ADC interleave channel select

**Table 4-122. Attribute McuPTB14InterleaveChannelSelect (McuChipControlConfiguration) detailed description**

Property	Value
Label	PTB14 to ADC1_SE9 and ADC0_SE9
Type	BOOLEAN
Origin	NXP

*Table continues on the next page...*

**Table 4-122. Attribute McuPTB14InterleaveChannelSelect (McuChipControlConfiguration) detailed description (continued)**

Property	Value
Symbolic Name	false
Default	false

#### 4.13.1.5 McuPTB13InterleaveChannelSelect (McuChipControlConfiguration)

SIM\_CHIPCTL[INTERLEAVE\_SEL] - ADC interleave channel select

**Table 4-123. Attribute McuPTB13InterleaveChannelSelect (McuChipControlConfiguration) detailed description**

Property	Value
Label	PTB13 to ADC1_SE8 and ADC0_SE8
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

#### 4.13.1.6 McuPTB1InterleaveChannelSelect (McuChipControlConfiguration)

SIM\_CHIPCTL[INTERLEAVE\_SEL] - ADC interleave channel select

**Table 4-124. Attribute McuPTB1InterleaveChannelSelect (McuChipControlConfiguration) detailed description**

Property	Value
Label	PTB1 to ADC0_SE4 and ADC1_SE15
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

### 4.13.1.7 McuPTB0InterleaveChannelSelect (McuChipControlConfiguration)

SIM\_CHIPCTL[INTERLEAVE\_SEL] - ADC interleave channel select

**Table 4-125. Attribute McuPTB0InterleaveChannelSelect (McuChipControlConfiguration) detailed description**

Property	Value
Label	PTB0 to ADC0_SE4 and ADC1_SE14
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

### 4.13.2 Form McuAdcOptionsConfiguration

This container contains the configuration for the SIM\_ADCOPT and SIM\_MISCTRL registers.

Is included by form : [Form McuSIMConfig](#)

The screenshot shows the 'McuAdcOptionsConfiguration' form in the TRESOS Plugin. It features a 'Name' field with the value 'McuAdcOptionsConfiguration'. Below this, there are several configuration rows, each with a label, a dropdown menu, and an edit icon (pencil). The rows are:

- ADC1 pre-trigger source: PDB\_PRE\_TRIGGER
- ADC1 software pre-trigger source: SW\_PRE\_TRIGGER\_DISABLE
- ADC1 trigger source: PDB
- ADC0 pre-trigger source: PDB\_PRE\_TRIGGER
- ADC0 software pre-trigger source: SW\_PRE\_TRIGGER\_DISABLE
- ADC0 trigger source: PDB
- Software Trigger to TRGMUX: (checkbox icon)

**Figure 4-21. TRESOS Plugin snapshot for McuAdcOptionsConfiguration form.**

#### 4.13.2.1 McuADC1PreTrigeeerSourceSelect (McuAdcOptionsConfiguration)

SIM\_ADCOPT[ADC1PRETRGSE] - ADC1 pre-trigger source select

**Table 4-126. Attribute McuADC1PreTrigeerSourceSelect (McuAdcOptionsConfiguration) detailed description**

Property	Value
Label	ADC1 pre-trigger source
Type	ENUMERATION
Origin	NXP
Symbolic Name	false
Default	PDB_PRE_TRIGGER
Range	PDB_PRE_TRIGGER TRGMUX_PRE_TRIGGER SOFTWARE_PRE_TRIGGER

#### 4.13.2.2 McuADC1SoftwarePreTrigeerSourceSelect (McuAdcOptionsConfiguration)

SIM\_ADCOPT[ADC1SWPRETRG] - ADC1 software pre-trigger sources

**Table 4-127. Attribute McuADC1SoftwarePreTrigeerSourceSelect (McuAdcOptionsConfiguration) detailed description**

Property	Value
Label	ADC1 software pre-trigger source
Type	ENUMERATION
Origin	NXP
Symbolic Name	false
Default	SW_PRE_TRIGGER_DISABLE
Range	SW_PRE_TRIGGER_DISABLE SW_PRE_TRIGGER_0 SW_PRE_TRIGGER_1 SW_PRE_TRIGGER_2 SW_PRE_TRIGGER_3

#### 4.13.2.3 McuADC1TrigeerSourceSelect (McuAdcOptionsConfiguration)

SIM\_ADCOPT[ADC1TRGSEL] - ADC1 trigger source select

**Table 4-128. Attribute McuADC1TrigeerSourceSelect (McuAdcOptionsConfiguration) detailed description**

Property	Value
Label	ADC1 trigger source
Type	ENUMERATION
Origin	NXP
Symbolic Name	false
Default	PDB
Range	PDB TRGMUX

#### 4.13.2.4 McuADC0PreTrigeerSourceSelect (McuAdcOptionsConfiguration)

SIM\_ADCOPT[ADC0PRETRGSE] - ADC0 pre-trigger source select

**Table 4-129. Attribute McuADC0PreTrigeerSourceSelect (McuAdcOptionsConfiguration) detailed description**

Property	Value
Label	ADC0 pre-trigger source
Type	ENUMERATION
Origin	NXP
Symbolic Name	false
Default	PDB_PRE_TRIGGER
Range	PDB_PRE_TRIGGER TRGMUX_PRE_TRIGGER SOFTWARE_PRE_TRIGGER

#### 4.13.2.5 McuADC0SoftwarePreTrigeerSourceSelect (McuAdcOptionsConfiguration)

SIM\_ADCOPT[ADC0SWPRETRG] - ADC0 software pre-trigger sources

**Table 4-130. Attribute McuADC0SoftwarePreTrigeerSourceSelect (McuAdcOptionsConfiguration) detailed description**

Property	Value
Label	ADC0 software pre-trigger source
Type	ENUMERATION

*Table continues on the next page...*



**Table 4-130. Attribute McuADC0SoftwarePreTrigeeerSourceSelect (McuAdcOptionsConfiguration) detailed description (continued)**

Property	Value
Origin	NXP
Symbolic Name	false
Default	SW_PRE_TRIGGER_DISABLE
Range	SW_PRE_TRIGGER_DISABLE SW_PRE_TRIGGER_0 SW_PRE_TRIGGER_1 SW_PRE_TRIGGER_2 SW_PRE_TRIGGER_3

#### 4.13.2.6 McuADC0TrigeeerSourceSelect (McuAdcOptionsConfiguration)

SIM\_ADCOPT[ADC2TRGSEL] - ADC0 trigger source select

**Table 4-131. Attribute McuADC0TrigeeerSourceSelect (McuAdcOptionsConfiguration) detailed description**

Property	Value
Label	ADC0 trigger source
Type	ENUMERATION
Origin	NXP
Symbolic Name	false
Default	PDB
Range	PDB TRGMUX

#### 4.13.2.7 McuSoftwareTriggerToTRGMUX (McuAdcOptionsConfiguration)

SIM\_MISCTRL[SW\_TRG] - Software Trigger bit to TRGMUX

Note: Implementation Specific Parameter.

**Table 4-132. Attribute McuSoftwareTriggerToTRGMUX (McuAdcOptionsConfiguration) detailed description**

Property	Value
Label	Software Trigger to TRGMUX

*Table continues on the next page...*

**Table 4-132. Attribute McuSoftwareTriggerToTRGMUX (McuAdcOptionsConfiguration) detailed description (continued)**

Property	Value
Type	BOOLEAN
Origin	NXP
Symbolic Name	false
Default	false

**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).

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

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. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. 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.

© 2019 NXP B.V.

Document Number UM2ADCASR4.3 Rev0001R1.0.1  
Revision 1.0