User Manual

for MPC574XG ADC Driver

Document Number: UM35ADCASR4.2 Rev002R1.0.0
Rev. 1.0

Contents

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

Chapter 1
Revision History

Chapter 2
Introduction

Chapter 3
Driver

Chapter 4
Tresos Configuration Plug-in

Chapter 1
Revision History

Table 1-1. Revision History

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

Chapter 2
Introduction

This User Manual describes NXP Semiconductor AUTOSAR Analog to Digital Converter (Adc) for MPC574XG .

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 intented to be used with the following microcontroller devices of NXP Semiconductor .

Table 2-1. MPC574XG Derivatives

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

Table 2-1. MPC574XG Derivatives

	MPC5744C_MAPBGA100, MPC5746B_LQFP176, MPC5746B_MAPBGA256, MPC5746B_MAPBGA100, MPC5744B_LQFP176, MPC5744B_MAPBGA256, MPC5744B_MAPBGA100, MPC5745B_LQFP176, MPC5745B_MAPBGA256, MPC5745B_MAPBGA100

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

2.2 Overview

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

AUTOSAR

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

2.3 About this Manual

This Technical Reference employs the following typographical conventions:

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

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

Notes and warnings are shown as below:

Note

This is a note.

2.4 Acronyms and Definitions

Table 2-2. Acronyms and Definitions

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

2.5 Reference List

Table 2-3. Reference List

#	Title	Version
1	AUTOSAR 4.2 Rev0002Adc Driver Software Specification Document.	R4.2 Rev 002
2	MPC5748G Reference Manual	Rev. 5, 12/2016
3	MPC5746C Reference Manual	Rev. 4, 12/2016
4	MPC5748G_1N81M_Rev.2 (official document) (1N81M)	Jun-16
5	MPC5748G_1N81M_0N78S_Comparison_Summary_v2_0 (internal document) (1N81M, 0N78S)	31.10.2016
6	MPC5746C_1N06M_Rev.4 (official document) (1N06M)	Jul-16
7	MPC5746C_cut1.1_cut2.0_cut2.1_comparison_v0 (internal document) (1N06M, 0N84S, 1N84S)	14-Sep-16
8	C3M_cut2.1_new_errata_20170113 (internal document) (1N84S)	13-Jan-17

Chapter 3
Driver

3.1 Requirements

Requirements for this driver are detailed in the AUTOSAR 4.2 Rev0002Adc 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

- **MPC5748G device families at 176 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:** AN_3, AN_4, AN_7, AN_8, AN_11, AN_12, AN_17, AN_18, AN_19, AN_20, AN_21, AN_32, AN_33, AN_34, AN_35, AN_38, AN_39, AN_42, AN_43, AN_46, AN_47, AN_50, AN_51, AN_52, AN_53, AN_54, AN_55, AN_56, AN_57, AN_58, AN_59, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78, AN_79, AN_80, AN_81,

AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95

- **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_33, AN_35, AN_36, AN_37, AN_38, AN_39, AN_40, AN_41, AN_42, AN_43, AN_44, AN_45, AN_46, AN_47, AN_49, AN_50, AN_51, AN_52, AN_53, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78, AN_79, AN_80, AN_81, AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95

- **MPC5748G device families at 256 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:** AN_3, AN_4, AN_7, AN_8, AN_11, AN_12, AN_17, AN_18, AN_19, AN_20, AN_21, AN_32, AN_33, AN_34, AN_35, AN_38, AN_39, AN_40, AN_41, AN_42, AN_43, AN_44, AN_45, AN_46, AN_47, AN_48, AN_49, AN_50, AN_51, AN_52, AN_53, AN_54, AN_55, AN_56, AN_57, AN_58, AN_59, AN_60, AN_61, AN_62, AN_63, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78, AN_79, AN_80, AN_81, AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95
  - **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_32, AN_33, AN_35, AN_36, AN_37, AN_38, AN_39, AN_40, AN_41, AN_42, AN_43, AN_44, AN_45, AN_46, AN_47, AN_49, AN_50, AN_51, AN_52, AN_53, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78, AN_79, AN_80, AN_81, AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95
- **MPC5748G device families at 324 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_17, AN_18, AN_19, AN_20, AN_21, AN_32, AN_33, AN_34, AN_35, AN_36, AN_37, AN_38, AN_39, AN_40, AN_41, AN_42, AN_43, AN_44, AN_45, AN_46, AN_47, AN_48, AN_49, AN_50, AN_51, AN_52, AN_53, AN_54, AN_55, AN_56, AN_57, AN_58, AN_59, AN_60, AN_61, AN_62, AN_63, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78,

AN_79, AN_80, AN_81, AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95

- **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_32, AN_33, AN_34, AN_35, AN_36, AN_37, AN_38, AN_39, AN_40, AN_41, AN_42, AN_43, AN_44, AN_45, AN_46, AN_47, AN_49, AN_50, AN_51, AN_52, AN_53, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78, AN_79, AN_80, AN_81, AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95

- **MPC5746C device families at 100 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:** AN_7, AN_8, AN_17, AN_18, AN_19, AN_20, AN_21, AN_34, AN_42, AN_43, AN_46, AN_50, AN_53
  - **Adc Physical Channels for ADC HW Unit 1:** AN_1, AN_33, AN_35, AN_36, AN_37, AN_38, AN_39, AN_41, AN_42, AN_43, AN_44, AN_45, AN_46, AN_49, AN_50, AN_51, AN_52, AN_53, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78, AN_79, AN_80, AN_81, AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95
- **MPC5746C device families at 176 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:** AN_3, AN_4, AN_7, AN_8, AN_11, AN_12, AN_17, AN_18, AN_19, AN_20, AN_21, AN_32, AN_33, AN_34, AN_35, AN_38, AN_39, AN_42, AN_43, AN_46, AN_47, AN_50, AN_51, AN_52, AN_53, AN_54, AN_55, AN_56, AN_57, AN_58, AN_59, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78, AN_79, AN_80, AN_81, AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95
  - **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_33, AN_35, AN_36, AN_37, AN_38, AN_39, AN_40, AN_41, AN_42, AN_43, AN_44, AN_45, AN_46, AN_47, AN_49, AN_50, AN_51, AN_52, AN_53, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78, AN_79, AN_80, AN_81, AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95
- **MPC5746C device families at 256 pins:**

- **Adc Physical Channels for ADC HW Unit 0:** AN_3, AN_4, AN_7, AN_8, AN_11, AN_12, AN_17, AN_18, AN_19, AN_20, AN_21, AN_32, AN_33, AN_34, AN_35, AN_38, AN_39, AN_40, AN_41, AN_42, AN_43, AN_44, AN_45, AN_46, AN_47, AN_48, AN_49, AN_50, AN_51, AN_52, AN_53, AN_54, AN_55, AN_56, AN_57, AN_58, AN_59, AN_60, AN_61, AN_62, AN_63, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78, AN_79, AN_80, AN_81, AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95
- **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_32, AN_33, AN_35, AN_36, AN_37, AN_38, AN_39, AN_40, AN_41, AN_42, AN_43, AN_44, AN_45, AN_46, AN_47, AN_49, AN_50, AN_51, AN_52, AN_53, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78, AN_79, AN_80, AN_81, AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95

- **MPC5746C device families at 324 pins:**
  - **Adc Physical Channels for ADC HW Unit 0:** AN_3, AN_4, AN_7, AN_8, AN_11, AN_12, AN_17, AN_18, AN_19, AN_20, AN_21, AN_32, AN_33, AN_34, AN_35, AN_38, AN_39, AN_40, AN_41, AN_42, AN_43, AN_44, AN_45, AN_46, AN_47, AN_48, AN_49, AN_50, AN_51, AN_52, AN_53, AN_54, AN_55, AN_56, AN_57, AN_58, AN_59, AN_60, AN_61, AN_62, AN_63, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78, AN_79, AN_80, AN_81, AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95
  - **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_32, AN_33, AN_35, AN_36, AN_37, AN_38, AN_39, AN_40, AN_41, AN_42, AN_43, AN_44, AN_45, AN_46, AN_47, AN_49, AN_50, AN_51, AN_52, AN_53, AN_64, AN_65, AN_66, AN_67, AN_68, AN_69, AN_70, AN_71, AN_72, AN_73, AN_74, AN_75, AN_76, AN_77, AN_78, AN_79, AN_80, AN_81, AN_82, AN_83, AN_84, AN_85, AN_86, AN_87, AN_88, AN_89, AN_90, AN_91, AN_92, AN_93, AN_94, AN_95

The ADC channel to microcontroller pin mapping can be done using chapter "**32.1.4 ADC channel mapping**" and "**IO_Signal_Description_and_Input_multiplexing_tables.xls**" from the Reference

manual. For example, channel ADC0_S[43] can be found in the xls file, it is connected to pin PN[10]. In order to use ADC0_S[43] in the ADC driver, the corresponding channel AN_0 of unit ADC0 must be selected.

- **Direct hardware triggering (without BCTU)**
  - ADC supports injected hardware triggering directly from PIT module. There are the trigger sources:
    - PIT_2 is connected to ADC0
    - PIT_6 is connected to ADC1
- **Hardware triggering with BCTU**
  - The enhanced BCTU allows synchronisation between an ADC conversion and a timer event. A single event (eMIOS channel or PIT) can trigger a group of ADC channels to be converted.
  - ADC supports two operating mode for BCTU:
    - BCTU trigger mode:
      - On Calypso, there are 96 trigger event inputs that can be used for BCTU trigger mode: BCTU_EMIOS0_0, BCTU_EMIOS0_1, BCTU_EMIOS0_2, BCTU_EMIOS0_3, BCTU_EMIOS0_4, BCTU_EMIOS0_5, BCTU_EMIOS0_6, BCTU_EMIOS0_7, BCTU_EMIOS0_8, BCTU_EMIOS0_9, BCTU_EMIOS0_10, BCTU_EMIOS0_11, BCTU_EMIOS0_12, BCTU_EMIOS0_13, BCTU_EMIOS0_14, BCTU_EMIOS0_15, BCTU_EMIOS0_16, BCTU_EMIOS0_17, BCTU_EMIOS0_18, BCTU_EMIOS0_19, BCTU_EMIOS0_20, BCTU_EMIOS0_21, BCTU_EMIOS0_22, BCTU_PIT_3, BCTU_EMIOS0_24, BCTU_EMIOS0_25, BCTU_EMIOS0_26, BCTU_EMIOS0_27, BCTU_EMIOS0_28, BCTU_EMIOS0_29, BCTU_EMIOS0_30, BCTU_EMIOS0_31, BCTU_EMIOS1_0, BCTU_EMIOS1_1, BCTU_EMIOS1_2, BCTU_EMIOS1_3, BCTU_EMIOS1_4, BCTU_EMIOS1_5, BCTU_EMIOS1_6, BCTU_EMIOS1_7, BCTU_EMIOS1_8, BCTU_EMIOS1_9, BCTU_EMIOS1_10, BCTU_EMIOS1_11, BCTU_EMIOS1_12, BCTU_EMIOS1_13, BCTU_EMIOS1_14, BCTU_EMIOS1_15, BCTU_EMIOS1_16, BCTU_EMIOS1_17, BCTU_EMIOS1_18, BCTU_EMIOS1_19, BCTU_EMIOS1_20, BCTU_EMIOS1_21, BCTU_EMIOS1_22, BCTU_PIT_7, BCTU_EMIOS1_24, BCTU_EMIOS1_25, BCTU_EMIOS1_26, BCTU_EMIOS1_27, BCTU_EMIOS1_28, BCTU_EMIOS1_29, BCTU_EMIOS1_30, BCTU_EMIOS1_31, BCTU_EMIOS2_0, BCTU_EMIOS2_1, BCTU_EMIOS2_2, BCTU_EMIOS2_3,

BCTU_EMIOS2_4, BCTU_EMIOS2_5, BCTU_EMIOS2_6, BCTU_EMIOS2_7, BCTU_EMIOS2_8, BCTU_EMIOS2_9, BCTU_EMIOS2_10, BCTU_EMIOS2_11, BCTU_EMIOS2_12, BCTU_EMIOS2_13, BCTU_EMIOS2_14, BCTU_EMIOS2_15, BCTU_EMIOS2_16, BCTU_EMIOS2_17, BCTU_EMIOS2_18, BCTU_EMIOS2_19, BCTU_EMIOS2_20, BCTU_EMIOS2_21, BCTU_EMIOS2_22, BCTU_PIT_8, BCTU_EMIOS2_24, BCTU_EMIOS2_25, BCTU_EMIOS2_26, BCTU_EMIOS2_27, BCTU_EMIOS2_28, BCTU_EMIOS2_29, BCTU_EMIOS2_30, BCTU_EMIOS2_31.

    - For configuration details, please refer to Form AdcHwTrig

- BCTU control mode:

    - BCTU control mode can be used in ADC driver by using Adc_EnableCtuControlMode NonAutosar functionality. Channels to be converted are not configured in ADC Groups, they are configured directly in the BCTU configuration structure.
    - In BCTU Control mode conversion requests can be generated only by a BCTU trigger. No other normal or injected conversions can be executed in parallel when the BCTU is enabled.
    - On Calypso, there are 96 trigger event inputs that can be used for BCTU control mode: BCTU_EMIOS0_0, BCTU_EMIOS0_1, BCTU_EMIOS0_2, BCTU_EMIOS0_3, BCTU_EMIOS0_4, BCTU_EMIOS0_5, BCTU_EMIOS0_6, BCTU_EMIOS0_7, BCTU_EMIOS0_8, BCTU_EMIOS0_9, BCTU_EMIOS0_10, BCTU_EMIOS0_11, BCTU_EMIOS0_12, BCTU_EMIOS0_13, BCTU_EMIOS0_14, BCTU_EMIOS0_15, BCTU_EMIOS0_16, BCTU_EMIOS0_17, BCTU_EMIOS0_18, BCTU_EMIOS0_19, BCTU_EMIOS0_20, BCTU_EMIOS0_21, BCTU_EMIOS0_22, BCTU_PIT_3, BCTU_EMIOS0_24, BCTU_EMIOS0_25, BCTU_EMIOS0_26, BCTU_EMIOS0_27, BCTU_EMIOS0_28, BCTU_EMIOS0_29, BCTU_EMIOS0_30, BCTU_EMIOS0_31, BCTU_EMIOS1_0, BCTU_EMIOS1_1, BCTU_EMIOS1_2, BCTU_EMIOS1_3, BCTU_EMIOS1_4, BCTU_EMIOS1_5, BCTU_EMIOS1_6, BCTU_EMIOS1_7, BCTU_EMIOS1_8, BCTU_EMIOS1_9, BCTU_EMIOS1_10, BCTU_EMIOS1_11, BCTU_EMIOS1_12, BCTU_EMIOS1_13, BCTU_EMIOS1_14, BCTU_EMIOS1_15, BCTU_EMIOS1_16, BCTU_EMIOS1_17, BCTU_EMIOS1_18, BCTU_EMIOS1_19, BCTU_EMIOS1_20, BCTU_EMIOS1_21, BCTU_EMIOS1_22, BCTU_PIT_7, BCTU_EMIOS1_24, BCTU_EMIOS1_25, BCTU_EMIOS1_26, BCTU_EMIOS1_27, BCTU_EMIOS1_28, BCTU_EMIOS1_29,

BCTU_EMIOS1_30, BCTU_EMIOS1_31, BCTU_EMIOS2_0, BCTU_EMIOS2_1, BCTU_EMIOS2_2, BCTU_EMIOS2_3, BCTU_EMIOS2_4, BCTU_EMIOS2_5, BCTU_EMIOS2_6, BCTU_EMIOS2_7, BCTU_EMIOS2_8, BCTU_EMIOS2_9, BCTU_EMIOS2_10, BCTU_EMIOS2_11, BCTU_EMIOS2_12, BCTU_EMIOS2_13, BCTU_EMIOS2_14, BCTU_EMIOS2_15, BCTU_EMIOS2_16, BCTU_EMIOS2_17, BCTU_EMIOS2_18, BCTU_EMIOS2_19, BCTU_EMIOS2_20, BCTU_EMIOS2_21, BCTU_EMIOS2_22, BCTU_PIT_8, BCTU_EMIOS2_24, BCTU_EMIOS2_25, BCTU_EMIOS2_26, BCTU_EMIOS2_27, BCTU_EMIOS2_28, BCTU_EMIOS2_29, BCTU_EMIOS2_30, BCTU_EMIOS2_31.

    - For configuration details, please refer to Form BCTUHwUnit

- **Multiple Hardware Triggers (MHT)**
    - This feature is only supported for BCTU hardware triggers (trigger mode) .
    - It allows the driver to run more than one hardware triggered group on the same hardware unit in parallel.
    - To use this feature it is required that every MHT marked group shares the same settings on several parameters. The list of parameters that can be different is:
        - Group Id: Mandatory to be different.
        - Group Priority: It can have any value in the domain, it doesn't matter. The priority will be based on the selected BCTU trigger priority.
        - Group Notification: Can be different.
        - Group Buffer Pointer: Recommended to be different to know for each value from which group/channel comes.
        - AdcHwTrigSrc: Must be different for every group. If two or more MHT groups share the same trigger source there will be an error at configuration time.
    - Each such a group (MHT) should have exactly one ADC channel. These ADC channels should be unique across these MHT groups (actually across the MHT subset runtime).

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_00482	N/A	In case the target power state is the same as the current one, no action is executed and the API returns immediately with an E_OK result.	This requirement should follow requirement 478:return E_NOT_OK and ADC_SEQUENCE_ERROR for (Current Power State = Target Power State).
SWS_Adc_00488	N/A	In case development error reporting is activated:The API shall report the DET error ADC_E_POWER_STATE_NOT_SUPPORTED in case this API is called with an unsupported power state or the peripheral does not support low power states at all.	Adc_SetPowerState does not have the Adc_PowerStateType parameter. This parameter is passed and checked (for unsupported power state) through the AdcPreparePowerState function that is called before calling Adc_SetPowerState function.
SWS_Adc_00489	N/A	In case development error reporting is activated:The API shall report the DET error ADC_E_TRANSITION_NOT POSSIBLE in case the requested power state cannot be directly reached from the current power state.	ADC hardware supports only two power states (full power and ow power) and all transitions are valid; there is no need to report ADC_E_TRANSITION_NOT_POSSIBLE error.
SWS_Adc_00495	N/A	In case the target power state is the same as the current one, no action is executed and the API returns immediately with an E_OK result.The responsibility of the preconditions is left to the environment.	This requirement should follow requirement 478: return E_NOT_OK and ADC_SEQUENCE_ERROR for (Current Power State = Target Power State).
SWS_Adc_00498	N/A	In case development error reporting is activated:The API shall report the DET error ADC_E_TRANSITION_NOT POSSIBLE in case the requested power state cannot be directly reached from the current power state.All asynchronous operation needed to reach the target power state can be executed in background in the context of Adc_Main_PowerTransitionManager.	ADC hardware supports only two power states (full power and ow power) and all transitions are valid; there is no need to report ADC_E_TRANSITION_NOT_POSSIBLE error.
SWS_Adc_00479	N/A	Service name: Adc_Main_PowerTransitionManager Syntax: void Adc_Main_PowerTransitionManager(void) Service ID[hex]: 0x14 Description: This API is cyclically called and supervises the power state transitions, checking for the readiness of the module and issuing the callbacks	Asynchronous Power State transition mode is not needed. It's not supported by hardware.

Table continues on the next page...

Table 3-2. ADC Deviations Table (continued)

Requirement	Status	Description	Notes
		IoHwAb_Adc_NotifyReadyForPowerState Mode (see AdcPowerStateReadyCbkRef configuration parameter).	
SWS_Adc_00499	N/A	This API executes any non-immediate action needed to finalize a power state transition requested by Adc_PreparePowerState().	Asynchronous Power State transition mode is not needed. It's not supported by hardware.
SWS_Adc_00500	N/A	The rate of scheduling shall be defined by Adc MainSchedulePeriod and shall be variable, as the function only needs to be called if a transition has been requested.	Asynchronous Power State transition mode is not needed. It's not supported by hardware.
SWS_Adc_00501	N/A	This API shall also issue callback notifications to the eventually registered users (IoHwAbs) as configured, only in case the asynch mode is chosen.	Asynchronous Power State transition mode is not needed. It's not supported by hardware.
SWS_Adc_00502	N/A	In case the ADC module is not initialized, this function shall simply return without any further elaboration. This is needed to avoid to elaborate uninitialized variables. No DET error shall be entered, because this condition can easily be verified during the startup phase (tasks started before the initialization is complete).Rationale: during the startup phase it can happen that the OS already schedules tasks, which call main functions, while some modules are not initialised yet. This is no real error condition, although need handling, i.e. returning without execution.Although the transition state monitoring functionality is mandatory, the implementation of this API is optional, meaning that if the HW allows for other ways to deliver notification and watch the transition state the implementation of this function can be skipped.	Asynchronous Power State transition mode is not needed. It's not supported by hardware.
SWS_Adc_00480	N/A	Service name: IoHwAb_Adc_NotifyReadyForPowerState Syntax: void IoHwAb_Adc_NotifyReadyForPowerState(voi d) Service ID[hex]: 0x70 Sync/Async: Synchronous Reentrancy: Non Reentrant Parameters (in): None Parameters (inout): None Parameters (out): None Return value: None Description: The API shall be invoked by the ADC Driver when the requested power state preparation for mode is completed.	Asynchronous Power State transition mode is not needed. It's not supported by hardware.
ECUC_Adc_00458	N/A	SWS Item ECUC_Adc_00458 : Name AdcPowerStateAsynchTransitionMode Description Enables / disables support of the ADCDriver to the asynchronous power state transition. Multiplicity 0..1 Type EcucBooleanParamDef Default value false Post-Build Variant Multiplicity false Post-Build Variant Value false Multiplicity Configuration Class Pre-compile time X All Variants Link	Asynchronous Power State transition mode is not needed. It's not supported by hardware.

Table continues on the next page...

Table 3-2. ADC Deviations Table (continued)

Requirement	Status	Description	Notes
		time -- Post-build time -- Value Configuration Class Pre-compile time X All Variants Link time -- Post-build time -- Scope / Dependency scope: local dependency: This parameter shall only be configured if the parameter AdcLowPowerStatesSupport is set to true.	
ECUC_Adc_00459	N/A	SWS Item ECUC_Adc_00459 : Container Name AdcPowerStateConfig Description Each instance of this parameter defines a power state and the callback to be called when this power state is reached. Configuration Parameters	Asynchronous Power State transition mode is not needed. It's not supported by hardware.
ECUC_Adc_00461	N/A	SWS Item ECUC_Adc_00461 : Name AdcPowerState Description Each instance of this parameter describes a different power state supported by the ADC HW. It should be defined by the HW supplier and used by the ADCDriver to reference specific HW configurations which set the ADC HW module in the referenced power state. At least the power mode corresponding to full power state shall be always configured. Multiplicity 1 Type EcucIntegerParamDef (Symbolic Name generated for this parameter) Range 0 .. 18446744073709551615 Default value -- Post-Build Variant Value false Value Configuration Class Pre-compile time X All Variants Link time -- Post-build time -- Scope / Dependency scope: local dependency: This parameter shall only be configured if the parameter AdcLowPowerStatesSupport is set to true.	Asynchronous Power State transition mode is not needed. It's not supported by hardware.
ECUC_Adc_00460	N/A	SWS Item ECUC_Adc_00460 : Name AdcPowerStateReadyCbkRef Description Each instance of this parameter contains a reference to a power mode callback defined in a CDD or IoHwAbs component. Multiplicity 1 Type EcucFunctionNameDef Default value -- maxLength -- minLength -- regularExpression -- Post-Build Variant Value false Value Configuration Class Pre-compile time Link time Post-build time Scope / Dependency scope: local dependency: This parameter shall only be configured if the parameter AdcLowPowerStatesSupport is set to true. No Included Containers	Asynchronous Power State transition mode is not needed. It's not supported by hardware.

3.5 ADC Driver limitations

After a BCTU triggered coversion has been enabled(ADC is in BCTU control mode), software conversions cannot be started on the same ADC unit until BCTU is disabled.

Configuration limitation: When configuring DMA channels to be used with BCTU, all BCTU configurations(in PostBuild configuration mode) must address only the first configuration of MCL driver

When using DMA transfer mode, HW triggered groups should not be used in parallel with any other groups (SW triggered with DMA transfer or SW triggered without interrupts)

When using Watchdog feature, if two or more channels share same ADC_THRHLR register, they must have the same limitation.

In BCTU control mode, if LIST mode is used, it is possible get incorrect data from the driver because of timing constraints between ADC and BCTU. To avoid this problem, conversion timing registers(ADC_CTRx) should be configured with maximum value and set once at initialization time.

If an ADC group is configured to run at high frequency, with DMA transfer in any other mode than one shot, software trigger, a great number of triggers to DMA will be issued, and it's possible that a residual trigger remains after the group is stopped. The next time DMA is enabled for ADC, a transfer may occur before DMA is triggered again.

When Pre-Sampling is enabled, the ADC output may be unreliable because there is an error occurred in Analog to Digital conversion. To avoid this problem, be sure that sampling period is greater than or equal to 375nS, this is controlled via INPSAMP bits of ADC_CTR register. In addition, for ADC1, having another way: select pre-sample voltage ADC_1_PSCR[PREVALn] = 3 (VDD_HV_ADC1_REF). Note: This driver limitation only occurs for Calypso 6M.

If development error detection for the ADC module is enabled , If the Adc_StopGroupConversion is called to stop a software normal group in the first place of software normal queue whille a software injected group is ongoing, the function Adc_StopGroupConversion shall raise development error ADC_E_BUSY and return without any action.

If Multiple Configuration Sets are used for ADC driver, and Limit checking feature is used for some Adc channels, the user must ensure consistency of limit checking configurations by following these restrictions:

- the same set of ADC units must be defined in all configurations, and ADC HW units must be mapped to the same ADC logical hardware unit ids.

- in all configurations, all Adc units should have the same number of channels configured.

- in all configurations, the limit checking parameters (AdcChannelLimitCheck, AdcChannelHighLimit, AdcChannelLowLimit, AdcChannelRangeSelect) should be configured with the same values.

The conversion User Buffer and User Callback function for each Bctu/Ctu input trigger must have a unique name.

Adc_SetChannel API cannot be used for a group if DMA transfer is configured for the associated ADC unit.

Adc_SetChannel API should not be used on Adc Groups that are used also with Adc_EnableChannel / Adc_DisableChannel.

Adc_SetChannel API should not be used on Adc Channels that have enabled the Limit Checking feature.

Adc_SetChannel API should not be used on Adc Groups that have enabled the Double Buffering feature. For such groups there is no interrupt after every conversion, so there is no guarantee that the configuration will be updated in a timely manner. Also, the configuration for such groups is limited to 1 channel and this restriction must be respected.

If ADC_SET_ADC_PRESAMPLE_ONCE is OFF, the presampling configuration of the channels will not match the list of channels updated by Adc_SetChannel API, but the list of channels in the original configuration.

3.6 Driver usage and configuration tips

None

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
...	...	...

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 4.2 Rev0002 .

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	2

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	2

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	0

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

Name	ADC_E_UNINIT
Initializer	((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

Name	ADC_E_BUSY
Initializer	((uint8)0x0BU)

3.8.1.11 Define ADC_E_IDLE

Adc module is in idle state.

Table 3-14. Define ADC_E_IDLE Description

Name	ADC_E_IDLE
Initializer	((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

Name	ADC_E_ALREADY_INITIALIZED
Initializer	((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

Name	ADC_E_PARAM_CONFIG
Initializer	((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

Name	ADC_E_PARAM_POINTER
Initializer	((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

Name	ADC_E_PARAM_GROUP
Initializer	((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

Name	ADC_E_WRONG_CONV_MODE
Initializer	((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

Name	ADC_E_WRONG_TRIGG_SRC
Initializer	((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

Name	ADC_E_NOTIF_CAPABILITY
Initializer	((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

Name	ADC_E_BUFFER_UNINIT
Initializer	((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

Name	ADC_E_NOT_DISENGAGED
Initializer	((uint8)0x1AU)

3.8.1.21 Define ADC_E_POWER_STATE_NOT_SUPPORTED

Unsupported power state request.

Table 3-24. Define ADC_E_POWER_STATE_NOT_SUPPORTED Description

Name	ADC_E_POWER_STATE_NOT_SUPPORTED
Initializer	((uint8)0x1BU)

3.8.1.22 Define ADC_E_TRANSITION_NOT_POSSIBLE

Requested power state can not be reached directly.

Table 3-25. Define ADC_E_TRANSITION_NOT_POSSIBLE Description

Name	ADC_E_TRANSITION_NOT_POSSIBLE
Initializer	((uint8)0x1CU)

3.8.1.23 Define ADC_E_PERIPHERAL_NOT_PREPARED

ADC not prepared for target power state.

Table 3-26. Define ADC_E_PERIPHERAL_NOT_PREPARED Description

Name	ADC_E_PERIPHERAL_NOT_PREPARED
Initializer	((uint8)0x1DU)

3.8.1.24 Define ADC_E_QUEUE_FULL

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

Table 3-27. Define ADC_E_QUEUE_FULL Description

Name	ADC_E_QUEUE_FULL
Initializer	((uint8)0x20U)

3.8.1.25 Define ADC_E_SET_MODE

An error occurred when the Adc_SetMode services is used.

Table 3-28. Define ADC_E_SET_MODE Description

Name	ADC_E_SET_MODE
Initializer	((uint8)0x21U)

3.8.1.26 Define ADC_E_PARAM_TRIGGER

Wrong trigger source to be used for the group.

Table 3-29. Define ADC_E_PARAM_TRIGGER Description

Name	ADC_E_PARAM_TRIGGER
Initializer	((uint8)0x22U)

3.8.1.27 Define ADC_E_WRONG_ENABLE_CH_DISABLE_CH_GROUP

Adc_EnableChannel or Adc_DisableChannel services called with a wrong channel.

Violates: Identifier exceeds 31 characters

Table 3-30. Define ADC_E_WRONG_ENABLE_CH_DISABLE_CH_GROUP Description

Name	ADC_E_WRONG_ENABLE_CH_DISABLE_CH_GROUP
Initializer	((uint8)0x23U)

3.8.1.28 Define ADC_E_WRONG_ENABLE_CH_DISABLE_CH_ID

Adc_EnableChannel or Adc_DisableChannel services called with a wrong channel identifier (ID).

Violates: Identifier exceeds 31 characters

Table 3-31. Define ADC_E_WRONG_ENABLE_CH_DISABLE_CH_ID Description

Name	ADC_E_WRONG_ENABLE_CH_DISABLE_CH_ID
Initializer	((uint8)0x24U)

3.8.1.29 Define ADC_E_WRONG_CONF_THRHLD_GROUP

Adc_ConfigureThreshold service is called for a wrong group.

Table 3-32. Define ADC_E_WRONG_CONF_THRHLD_GROUP Description

Name	ADC_E_WRONG_CONF_THRHLD_GROUP
Initializer	((uint8)0x25U)

3.8.1.30 Define ADC_E_WRONG_CONF_THRHLD_VALUE

Adc_ConfigureThreshold service is called using wrong values.

Table 3-33. Define ADC_E_WRONG_CONF_THRHLD_VALUE Description

Name	ADC_E_WRONG_CONF_THRHLD_VALUE
Initializer	((uint8)0x26U)

3.8.1.31 Define ADC_E_PARAM_UNIT

API service called using a wrong ADC unit.

Table 3-34. Define ADC_E_PARAM_UNIT Description

Name	ADC_E_PARAM_UNIT

Table continues on the next page...

Table 3-34. Define ADC_E_PARAM_UNIT Description (continued)

Initializer	((uint8)0x27U)

3.8.1.32 Define ADC_E_WRONG_CTUV2_TRIGGER

API service called using a wrong CTU trigger.

Table 3-35. Define ADC_E_WRONG_CTUV2_TRIGGER Description

Name	ADC_E_WRONG_CTUV2_TRIGGER
Initializer	((uint8)0x28U)

3.8.1.33 Define ADC_E_WRONG_CTUV2_CLCR_TRIGGER

API service called using a wrong CTU CLCR trigger.

Table 3-36. Define ADC_E_WRONG_CTUV2_CLCR_TRIGGER Description

Name	ADC_E_WRONG_CTUV2_CLCR_TRIGGER
Initializer	((uint8)0x29U)

3.8.1.34 Define ADC_E_INVALID_CLOCK_MODE

Adc_SetClockMode service called using an invalid clock mode.

Table 3-37. Define ADC_E_INVALID_CLOCK_MODE Description

Name	ADC_E_INVALID_CLOCK_MODE
Initializer	((uint8)0x2AU)

3.8.1.35 Define ADC_E_PARAM_CHANNEL

Adc_SetChannel service called using an invalid channel list.

Table 3-38. Define ADC_E_PARAM_CHANNEL Description

Name	ADC_E_PARAM_CHANNEL
Initializer	((uint8)0x2BU)

3.8.1.36 Define ADC_E_BUFFER_UNINIT_LIST

Table 3-39. Define ADC_E_BUFFER_UNINIT_LIST Description

Name	ADC_E_BUFFER_UNINIT_LIST
Initializer	((uint32)0x00000001U)

3.8.1.37 Define ADC_E_WRONG_TRIGG_SRC_LIST

Table 3-40. Define ADC_E_WRONG_TRIGG_SRC_LIST Description

Name	ADC_E_WRONG_TRIGG_SRC_LIST
Initializer	((uint32)0x00000002U)

3.8.1.38 Define ADC_E_QUEUE_FULL_LIST

Table 3-41. Define ADC_E_QUEUE_FULL_LIST Description

Name	ADC_E_QUEUE_FULL_LIST
Initializer	((uint32)0x00000004U)

3.8.1.39 Define ADC_E_WRONG_CONV_MODE_LIST

Table 3-42. Define ADC_E_WRONG_CONV_MODE_LIST Description

Name	ADC_E_WRONG_CONV_MODE_LIST
Initializer	((uint32)0x00000008U)

3.8.1.40 Define ADC_E_WRONG_ENABLE_CH_DISABLE_CH_GROUP_LIST

Violates: Identifier exceeds 31 characters

Table 3-43. Define ADC_E_WRONG_ENABLE_CH_DISABLE_CH_GROUP_LIST Description

Name	ADC_E_WRONG_ENABLE_CH_DISABLE_CH_GROUP_LIST
Initializer	((uint32)0x00000010U)

3.8.1.41 Define ADC_E_WRONG_ENABLE_CH_DISABLE_CH_ID_LIST

Violates: Identifier exceeds 31 characters

Table 3-44. Define ADC_E_WRONG_ENABLE_CH_DISABLE_CH_ID_LIST Description

Name	ADC_E_WRONG_ENABLE_CH_DISABLE_CH_ID_LIST
Initializer	((uint32)0x00000020U)

3.8.1.42 Define ADC_E_SET_MODE_LIST

Table 3-45. Define ADC_E_SET_MODE_LIST Description

Name	ADC_E_SET_MODE_LIST
Initializer	((uint32)0x00000040U)

3.8.1.43 Define ADC_INIT_ID

API service ID for Adc_Init function.

Details:

All AUTOSAR API's service IDs

Table 3-46. Define ADC_INIT_ID Description

Name	ADC_INIT_ID
Initializer	0x00U

3.8.1.44 Define ADC_DEINIT_ID

API service ID for Adc_DeInit function.

Table 3-47. Define ADC_DEINIT_ID Description

Name	ADC_DEINIT_ID
Initializer	0x01U

3.8.1.45 Define ADC_STARTGROUPCONVERSION_ID

API service ID for Adc_StartGroupConversion function.

Table 3-48. Define ADC_STARTGROUPCONVERSION_ID Description

Name	ADC_STARTGROUPCONVERSION_ID
Initializer	0x02U

3.8.1.46 Define ADC_STOPGROUPCONVERSION_ID

API service ID for Adc_StopGroupConversion function.

Table 3-49. Define ADC_STOPGROUPCONVERSION_ID Description

Name	ADC_STOPGROUPCONVERSION_ID
Initializer	0x03U

3.8.1.47 Define ADC_VALUEREADGROUP_ID

API service ID for Adc_ReadGroup function.

Table 3-50. Define ADC_VALUEREADGROUP_ID Description

Name	ADC_VALUEREADGROUP_ID
Initializer	0x04U

3.8.1.48 Define ADC_ENABLEHARDWARETRIGGER_ID

API service ID for Adc_EnableHardwareTrigger function.

Table 3-51. Define ADC_ENABLEHARDWARETRIGGER_ID Description

Name	ADC_ENABLEHARDWARETRIGGER_ID
Initializer	0x05U

3.8.1.49 Define ADC_DISABLEHARDWARETRIGGER_ID

API service ID for Adc_DisableHardwareTrigger function.

Table 3-52. Define ADC_DISABLEHARDWARETRIGGER_ID Description

Name	ADC_DISABLEHARDWARETRIGGER_ID
Initializer	0x06U

3.8.1.50 Define ADC_ENABLEGROUPNOTIFICATION_ID

API service ID for Adc_EnableGroupNotification function.

Table 3-53. Define ADC_ENABLEGROUPNOTIFICATION_ID Description

Name	ADC_ENABLEGROUPNOTIFICATION_ID
Initializer	0x07U

3.8.1.51 Define ADC_DISABLEGROUPNOTIFICATION_ID

API service ID for Adc_DisableGroupNotification function.

Table 3-54. Define ADC_DISABLEGROUPNOTIFICATION_ID Description

Name	ADC_DISABLEGROUPNOTIFICATION_ID
Initializer	0x08U

3.8.1.52 Define ADC_GETGROUPSTATUS_ID

API service ID for Adc_GetGroupStatus function.

Table 3-55. Define ADC_GETGROUPSTATUS_ID Description

Name	ADC_GETGROUPSTATUS_ID
Initializer	0x09U

3.8.1.53 Define ADC_GETVERSIONINFO_ID

API service ID for Adc_GetVersionInfo function.

Table 3-56. Define ADC_GETVERSIONINFO_ID Description

Name	ADC_GETVERSIONINFO_ID
Initializer	0x0AU

3.8.1.54 Define ADC_GETSTREAMLASTPOINTER_ID

API service ID for Adc_GetStreamLastPointer function.

Table 3-57. Define ADC_GETSTREAMLASTPOINTER_ID Description

Name	ADC_GETSTREAMLASTPOINTER_ID
Initializer	0x0BU

3.8.1.55 Define ADC_SETUPRESULTBUFFER_ID

API service ID for Adc_SetupResultBuffer function.

Table 3-58. Define ADC_SETUPRESULTBUFFER_ID Description

Name	ADC_SETUPRESULTBUFFER_ID
Initializer	0x0CU

3.8.1.56 Define ADC_SETPOWERSTATE_ID

API service ID for Adc_SetPowerState function.

Table 3-59. Define ADC_SETPOWERSTATE_ID Description

Name	ADC_SETPOWERSTATE_ID
Initializer	0x10U

3.8.1.57 Define ADC_GETCURRENTPOWERSTATE_ID

API service ID for Adc_GetCurrentPowerState function.

Table 3-60. Define ADC_GETCURRENTPOWERSTATE_ID Description

Name	ADC_GETCURRENTPOWERSTATE_ID
Initializer	0x11U

3.8.1.58 Define ADC_GETTARGETPOWERSTATE_ID

API service ID for Adc_GetTargetPowerState function.

Table 3-61. Define ADC_GETTARGETPOWERSTATE_ID Description

Name	ADC_GETTARGETPOWERSTATE_ID
Initializer	0x12U

3.8.1.59 Define ADC_PREPAREPOWERSTATE_ID

API service ID for Adc_PreparePowerState function.

Table 3-62. Define ADC_PREPAREPOWERSTATE_ID Description

Name	ADC_PREPAREPOWERSTATE_ID
Initializer	0x13U

3.8.1.60 Define ADC_HWRESULTREADGROUP_ID

API service ID for Adc_HwResultReadGroup function.

Details:

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

Table 3-63. Define ADC_HWRESULTREADGROUP_ID Description

Name	ADC_HWRESULTREADGROUP_ID
Initializer	0x20U

3.8.1.61 Define ADC_ENABLECTUTRIGGER_ID

API service ID for Adc_EnableCTUTrigge function.

Table 3-64. Define ADC_ENABLECTUTRIGGER_ID Description

Name	ADC_ENABLECTUTRIGGER_ID
Initializer	0x21U

3.8.1.62 Define ADC_DISABLECTUTRIGGER_ID

API service ID for Adc_DisableCTUTrigger function.

Table 3-65. Define ADC_DISABLECTUTRIGGER_ID Description

Name	ADC_DISABLECTUTRIGGER_ID
Initializer	0x22U

3.8.1.63 Define ADC_SETMODE_ID

API service ID for Adc_SetMode function.

Table 3-66. Define ADC_SETMODE_ID Description

Name	ADC_SETMODE_ID
Initializer	0x23U

3.8.1.64 Define ADC_SETCLOCKMODE_ID

API service ID for Adc_SetClockMode function.

Table 3-67. Define ADC_SETCLOCKMODE_ID Description

Name	ADC_SETCLOCKMODE_ID
Initializer	0x24U

3.8.1.65 Define ADC_ENABLE_CHANNEL_ID

API service ID for Adc_EnableChannel function.

Table 3-68. Define ADC_ENABLE_CHANNEL_ID Description

Name	ADC_ENABLE_CHANNEL_ID
Initializer	0x25U

3.8.1.66 Define ADC_DISABLE_CHANNEL_ID

API service ID for Adc_DisableChannel function.

Table 3-69. Define ADC_DISABLE_CHANNEL_ID Description

Name	ADC_DISABLE_CHANNEL_ID
Initializer	0x26U

3.8.1.67 Define ADC_GETINJECTEDCONVERSIONSTATUS_ID

API service ID for Adc_GetInjectedConversionStatus function.

Table 3-70. Define ADC_GETINJECTEDCONVERSIONSTATUS_ID Description

Name	ADC_GETINJECTEDCONVERSIONSTATUS_ID
Initializer	0x27U

3.8.1.68 Define ADC_CALIBRATE_ID

API service ID for Adc_Calibrate function.

Table 3-71. Define ADC_CALIBRATE_ID Description

Name	ADC_CALIBRATE_ID
Initializer	0x28U

3.8.1.69 Define ADC_CONFIGURE_THRESHOLD_ID

API service ID for Adc_ConfigureThreshold function.

Table 3-72. Define ADC_CONFIGURE_THRESHOLD_ID Description

Name	ADC_CONFIGURE_THRESHOLD_ID
Initializer	0x2AU

3.8.1.70 Define ADC_ENABLECTUCONTROLMODE_ID

API service ID for Adc_EnableCtuControlMode function.

Table 3-73. Define ADC_ENABLECTUCONTROLMODE_ID Description

Name	ADC_ENABLECTUCONTROLMODE_ID
Initializer	(0x2FU)

3.8.1.71 Define ADC_DISABLECTUCONTROLMODE_ID

API service ID for Adc_DisableCtuControlMode function.

Table 3-74. Define ADC_DISABLECTUCONTROLMODE_ID Description

Name	ADC_DISABLECTUCONTROLMODE_ID
Initializer	(0x30U)

3.8.1.72 Define ADC_SETCHANNEL_ID

API service ID for Adc_SetChannel function.

Table 3-75. Define ADC_SETCHANNEL_ID Description

Name	ADC_SETCHANNEL_ID
Initializer	(0x31U)

3.8.2 Enum Reference

Enumeration of all constants supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.2 Rev0002 .

3.8.2.1 Enumeration Adc_DualClockModeType

Group Access Mode type.

Details:

Used for value received by Tressos interface configuration.

Table 3-76. Enumeration Adc_DualClockModeType Values

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

3.8.2.2 Enumeration Adc_GroupConversionStateType

ADC group already converted type.

Details:

Used to differentiate if group is already converted or not.

Table 3-77. Enumeration Adc_GroupConversionStateType Values

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

3.8.2.3 Enumeration Adc_GroupAccessModeType

Adc group access Mode.

Details:

Used for value received by Tressos interface configuration.

Implements: Adc_GroupAccessModeType_enumeration

Table 3-78. Enumeration Adc_GroupAccessModeType Values

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

3.8.2.4 Enumeration Adc_GroupConvType

Adc group conversion.

Details:

Used for value received by Tressos interface configuration.

Table 3-79. Enumeration Adc_GroupConvType Values

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

3.8.2.5 Enumeration Adc_GroupConvModeType

Adc Group conversion mode.

Details:

Used for value received by Tressos interface configuration.

Implements: Adc_GroupConvModeType_enumeration

Table 3-80. Enumeration Adc_GroupConvModeType Values

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

3.8.2.6 Enumeration Adc_GroupReplacementType

Adc group replacement.

Details:

Used for value received by Tressos interface configuration.

Implements: Adc_GroupReplacementType_enumeration

Table 3-81. Enumeration Adc_GroupReplacementType Values

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

3.8.2.7 Enumeration Adc_StreamBufferModeType

Adc group streaming buffer mode.

Details:

Used for value received by Tressos interface configuration.

Implements: Adc_StreamBufferModeType_enumeration

Table 3-82. Enumeration Adc_StreamBufferModeType Values

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

3.8.2.8 Enumeration Adc_StatusType

ADC group status.

Details:

ADC group enumeration type.

Implements: Adc_StatusType_enumeration

Table 3-83. Enumeration Adc_StatusType Values

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

3.8.2.9 Enumeration Adc_NotificationType

ADC group notification.

Details:

Indicates if notification is enabled for the group.

Table 3-84. Enumeration Adc_NotificationType Values

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

3.8.2.10 Enumeration Adc_HwTriggerSignalType

Adc hardware trigger edge.

Details:

Used for value received by Tressos interface configuration.

Implements: Adc_HwTriggerSignalType_enumeration

Table 3-85. Enumeration Adc_HwTriggerSignalType Values

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

3.8.2.11 Enumeration Adc_TriggerSourceType

Adc hardware trigger source.

Details:

Used for value received by Tressos interface configuration.

Implements: Adc_TriggerSourceType_enumeration

Table 3-86. Enumeration Adc_TriggerSourceType Values

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

3.8.2.12 Enumeration Adc_HwTriggeringType

Adc Hardware trigger.

Details:

Indicates if hardware trigger is enabled for group.

Table 3-87. Enumeration Adc_HwTriggeringType Values

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

3.8.2.13 Enumeration Adc_MhtGroupType

Adc MHT group.

Details:

Indicates if the hardware triggered group is regular or MHT type.

Table 3-88. Enumeration Adc_MhtGroupType Values

Name	Initializer	Description
ADC_REGULAR_GROUP_TYPE	0U	Indicate that the group is part-of a regular groups.
ADC_MHT_GROUP_TYPE		Indicate that the group is part-of the MHT groups subset.

3.8.2.14 Enumeration Adc_SetModeType

Set mode values.

Details:

Indicates all the values to set the Adc mode.

Table 3-89. Enumeration Adc_SetModeType Values

Name	Initializer	Description
ADC_NORMAL_MODE	0U	Normal mode.
ADC_POWER_DOWN_MODE		Power down mode.

3.8.2.15 Enumeration Adc_ChannelRangeSelectType

Range select values.

Details:

Indicates which range select is used.

Implements: Adc_ChannelRangeSelectType_enumeration

Table 3-90. Enumeration Adc_ChannelRangeSelectType Values

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

3.8.2.16 Enumeration Adc_PowerStateType

Power state type.

Details:

Power state currently active or set as target power state.

Implements: Adc_PowerStateType_enumeration

Table 3-91. Enumeration Adc_PowerStateType Values

Name	Initializer	Description
ADC_FULL_POWER	0U	Adc full power mode.
ADC_LOW_POWER		Adc low power mode.
ADC_NODEFINE_POWER		Adc no define power mode.

3.8.2.17 Enumeration Adc_PowerStateRequestResultType

Result of power state type.

Details:

Result of the requests related to power state transitions.

Implements: Adc_PowerStateRequestResultType_enumeration

Table 3-92. Enumeration Adc_PowerStateRequestResultType Values

Name	Initializer	Description
ADC_SERVICE_ACCEPTED	0U	Power state change executed.
ADC_NOT_INIT		Module not initialized.
ADC_SEQUENCE_ERROR		Wrong API call sequence.
ADC_HW_FAILURE		The HW module has a failure which prevents it to enter the required power state.
ADC_POWER_STATE_NOT_SUPP		Module does not support the requested power state.

Table continues on the next page...

Table 3-92. Enumeration Adc_PowerStateRequestResultType Values (continued)

Name	Initializer	Description
ADC_TRANS_NOT_POSSIBLE		Module cannot transition directly from the current power state to the requested power state.

3.8.3 Function Reference

Functions of all functions supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.2 Rev0002 .

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 3-93. 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_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 3-94. 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 3-95. Adc_SetupResultBuffer Return Values

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

3.8.3.3 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);`

3.8.3.4 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 3-96. Adc_StartGroupConversion Arguments

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

3.8.3.5 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 3-97. Adc_StopGroupConversion Arguments

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

3.8.3.6 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 3-98. 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 3-99. 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 occured.

3.8.3.7 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 3-100. Adc_EnableHardwareTrigger Arguments

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

3.8.3.8 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 3-101. Adc_DisableHardwareTrigger Arguments

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

3.8.3.9 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 3-102. Adc_EnableGroupNotification Arguments

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

3.8.3.10 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 3-103. Adc_DisableGroupNotification Arguments

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

3.8.3.11 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 3-104. Adc_GetGroupStatus Arguments

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

Table 3-105. Adc_GetGroupStatus Return Values

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

3.8.3.12 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 3-106. Adc_GetStreamLastPointer Arguments

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

Table 3-107. Adc_GetStreamLastPointer Return Values

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

3.8.3.13 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 3-108. Adc_GetVersionInfo Arguments

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

Table 3-109. Adc_GetVersionInfo Return Values

Name	Description
structure	In case of no errors.

3.8.3.14 Function Adc_SetPowerState

Enters the already prepared power state.

Details:

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

Return: Std_ReturnType Standard return type.

Post: Enters the already prepared power state.

Note

The function Autosar Service ID[hex]: 0x10.Synchronous.Non Re-entrant function.

Implements: Adc_SetPowerState_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_SetPowerState(Adc_PowerStateRequestResultType *Result);`

Table 3-110. Adc_SetPowerState Arguments

Type	Name	Direction	Description
`Adc_PowerStateRequestResultType*`	Result	**output**	Pointer to a variable to store the result of this function.

Table 3-111. Adc_SetPowerState Return Values

Name	Description
E_OK:	Power Mode changed.
E_NOT_OK:	Request rejected.

3.8.3.15 Function Adc_GetCurrentPowerState

Get the current power state of the ADC HW unit.

Details:

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

Return: Std_ReturnType Standard return type.

Post: Returns the current power state of the ADC HW unit.

Note

The function Autosar Service ID[hex]: 0x11.Synchronous.Non Re-entrant function.

Implements: Adc_GetCurrentPowerState_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_GetCurrentPowerState(Adc_PowerStateType *CurrentPowerState, Adc_PowerStateRequestResultType *Result);`

Table 3-112. Adc_GetCurrentPowerState Arguments

Type	Name	Direction	Description
`Adc_PowerStateType*`	CurrentPowerState	**output**	The current power mode of the ADC HW Unit is returned in this parameter.
`Adc_PowerStateRequestResultType*`	Result	**output**	Pointer to a variable to store the result of this function.

Table 3-113. Adc_GetCurrentPowerState Return Values

Name	Description
E_OK:	Mode could be read.
E_NOT_OK:	Service is rejected.

3.8.3.16 Function Adc_GetTargetPowerState

Get the target power state of the ADC HW unit.

Details:

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

Return: Std_ReturnType Standard return type.

Post: Returns the target power state of the ADC HW unit.

Note

The function Autosar Service ID[hex]: 0x12.Synchronous.Non Re-entrant function.

Implements: Adc_GetTargetPowerState_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_GetTargetPowerState(Adc_PowerStateType *TargetPowerState, Adc_PowerStateRequestResultType *Result);`

Table 3-114. Adc_GetTargetPowerState Arguments

Type	Name	Direction	Description
`Adc_PowerStateType*`	TargetPowerState	**output**	The Target power mode of the ADC HW Unit is returned in this parameter.
`Adc_PowerStateRequestResultType*`	Result	**output**	Pointer to a variable to store the result of this function.

Table 3-115. Adc_GetTargetPowerState Return Values

Name	Description
E_OK:	Mode could be read.
E_NOT_OK:	Service is rejected.

3.8.3.17 Function Adc_PreparePowerState

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

Details:

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

Return: Std_ReturnType Standard return type.

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

Note

The function Autosar Service ID[hex]: 0x13.Synchronous.Non Re-entrant function.

Implements: Adc_PreparePowerState_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_PreparePowerState(Adc_PowerStateType PowerState, Adc_PowerStateRequestResultType *Result);`

Table 3-116. Adc_PreparePowerState Arguments

Type	Name	Direction	Description
`Adc_PowerStateType`	PowerState	**input**	The target power state intended to be attained.
`Adc_PowerStateRequestResultType*`	Result	**output**	Pointer to a variable to store the result of this function.

Table 3-117. Adc_PreparePowerState Return Values

Name	Description
E_OK:	Mode could be read.
E_NOT_OK:	Service is rejected.

3.8.3.18 Function Adc_SetMode

Set the ADC mode either to powerdown or normal.

Details:

Set the ADC either to powerdown or normal mode.

Return: Std_ReturnType Standard return type.

Note

The function Non Autosar Service ID[hex]: 0x10.Synchronous.Non Re-entrant function.

Implements: Adc_SetMode_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_SetMode(Adc_SetModeType SetMode);`

Table 3-118. Adc_SetMode Arguments

Type	Name	Direction	Description
`Adc_SetModeType`	SetMode	**input**	Normal or powerdown mode.

Table 3-119. Adc_SetMode Return Values

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

3.8.3.19 Function Adc_EnableCTUTrigger

Enable the TriggerSource for group selected by Group parameter.

Details:

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

Return: void.

Note

The function Service ID[hex]: 0x0E.Synchronous.Non Re-entrant function.

Implements: Adc_EnableCTUTrigger_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_EnableCTUTrigger(Adc_GroupType Group, Adc_HwTriggerTimerType TriggerSource);`

Table 3-120. Adc_EnableCTUTrigger Arguments

Type	Name	Direction	Description
`Adc_GroupType`	Group	**input**	Index of group.
`Adc_HwTriggerTimerType`	TriggerSource	**input**	Trigger source to be used for the group. (Configuration file should contain it for that group).

3.8.3.20 Function Adc_DisableCTUTrigger

Disable the TriggerSource for group selected by Group parameter.

Details:

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

Return: void.

Note

The function Service ID[hex]: 0x0F.Synchronous.Non Re-entrant function.

Implements: Adc_DisableCTUTrigger_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_DisableCTUTrigger(Adc_GroupType Group, Adc_HwTriggerTimerType TriggerSource);`

Table 3-121. Adc_DisableCTUTrigger Arguments

Type	Name	Direction	Description
`Adc_GroupType`	Group	**input**	Index of group.
`Adc_HwTriggerTimerType`	TriggerSource	**input**	Trigger source to be disabled for the group. (Configuration file should contain it for that group).

3.8.3.21 Function Adc_HwResultReadGroup

Read the result of the hardware triggered groups conversion result.

Details:

This non Autosar API is used to read the result of the hardware triggered groups conversion result from the ADC hardware register in this case the CTU interrupt will be disabled for the group. The VALID bit CDR register will be cleared automatically when we read the results from the channel data register. If the user calls non-Autosar function`Adc_HwResultReadGroup()` once again before the next conversion takes place, the`Adc_HwResultReadGroup()` returns E_NOT_OK.

Return: Std_ReturnType Standard return type.

Note

The function Service ID[hex]: 0x0D.Synchronous.Non Re-entrant function.

Implements: Adc_HwResultReadGroup_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_HwResultReadGroup(Adc_GroupType Group, Adc_ValueGroupType *DataPtr);`

Table 3-122. Adc_HwResultReadGroup Arguments

Type	Name	Direction	Description
`Adc_GroupType`	Group	**input**	Index of group.
	pDataPtr	**input**	Pointer to a buffer which will be filled by the conversion results.

Table 3-123. Adc_HwResultReadGroup 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.

3.8.3.22 Function Adc_EnableChannel

Enable a channel inside a group.

Details:

This function allows to active a channel assigned to a group for SW normal conversion

Return: void.

Note

The function Service ID[hex]: 0x12.Synchronous.Re-entrant function.

Implements: Adc_EnableChannel_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_EnableChannel(Adc_GroupType Group, Adc_ChannelType Channel);`

Table 3-124. Adc_EnableChannel Arguments

Type	Name	Direction	Description
`Adc_GroupType`	Group	**input**	Symbolic name of group.
`Adc_ChannelType`	Channel	**input**	Symbolic name of channel.

3.8.3.23 Function Adc_DisableChannel

Disable a channel inside a group.

Details:

This function allows to de-active a channel assigned to a group for SW normal conversion

Return: void.

Note

The function Service ID[hex]: 0x13.Synchronous.Re-entrant function.

Implements: Adc_DisableChannel_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_DisableChannel(Adc_GroupType Group, Adc_ChannelType Channel);`

Table 3-125. Adc_DisableChannel Arguments

Type	Name	Direction	Description
`Adc_GroupType`	Group	**input**	Symbolic name of group.
`Adc_ChannelType`	Channel	**input**	Symbolic name of channel.

3.8.3.24 Function Adc_GetInjectedConversionStatus

Get the injected conversions status.

Details:

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

Return: Adc_StatusType Status of the ADC HW unit.

Note

The function Service ID[hex]: 0x14.Synchronous.Re-entrant function.

Implements: Adc_GetInjectedConversionStatus_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_GetInjectedConversionStatus(Adc_HwUnitType Unit);`

Table 3-126. Adc_GetInjectedConversionStatus Arguments

Type	Name	Direction	Description
	unit	**input**	ADC Unit id.

Table 3-127. Adc_GetInjectedConversionStatus Return Values

Name	Description
ADC_IDLE:	SW,HW Injection or Hardware Trigger group are idle.
ADC_BUSY:	SW,HW Injection or Hardware Trigger is inprogress.

3.8.3.25 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 3-128. 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.

3.8.3.26 Function Adc_ConfigureThreshold

Function to reconfigure High and Low thresholds for a given group.

Details:

This Non Autosar API is used to reconfigure High and Low thresholds for a given group.

Note

The function Service ID[hex]: 0x14.

Return: Std_ReturnType E_OK or E_NOT_OK.

Implements: Adc_ConfigureThreshold_Activity

Violates: All declarations and definitions of objects or

Prototype: `Std_ReturnType Adc_ConfigureThreshold(Adc_GroupType GroupId, Adc_ValueGroupType LowValue, Adc_ValueGroupType HighValue);`

Table 3-129. Adc_ConfigureThreshold Arguments

Type	Name	Direction	Description
`Adc_GroupType`	GroupId	**input**	Index of group.
`Adc_ValueGroupType`	LowValue	**input**	Low threshold value for channels in the group.
`Adc_ValueGroupType`	HighValue	**input**	High threshold value for channels in the group.

Table 3-130. Adc_ConfigureThreshold Return Values

Name	Description
E_OK	In case of successful Configure Threshold.
E_NOT_OK	In case of unsuccessful ConfigureThreshold.

3.8.3.27 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 3-131. Adc_SetClockMode Arguments

Type	Name	Direction	Description
	eClockMode	**input**	Normal or Alternate mode.

Table 3-132. Adc_SetClockMode Return Values

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

3.8.3.28 Function Adc_EnableCtuControlMode

Function to enable CTU control mode for an ADC unit.

Details:

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

Note

The function Service ID[hex]: 0x39.

Implements: Adc_EnableCtuControlMode_Activity

Violates: internal linkage vs external linkage.

Prototype: `void Adc_EnableCtuControlMode(Adc_HwUnitType Unit);`

Table 3-133. Adc_EnableCtuControlMode Arguments

Type	Name	Direction	Description
`Adc_HwUnitType`	Unit	**input**	ADC Unit Id.

3.8.3.29 Function Adc_DisableCtuControlMode

Function to disable CTU control mode for an ADC unit.

Details:

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

Note

The function Service ID[hex]: 0x4A.

Implements: Adc_DisableCtuControlMode_Activity

Violates: internal linkage vs external linkage.

Prototype: `void Adc_DisableCtuControlMode(Adc_HwUnitType Unit);`

Table 3-134. Adc_DisableCtuControlMode Arguments

Type	Name	Direction	Description
`Adc_HwUnitType`	Unit	**input**	ADC Unit Id.

3.8.3.30 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 uint16 u16Mask, const Adc_ChannelIndexType NumberOfChannel);`

Table 3-135. Adc_SetChannel Arguments

Type	Name	Direction	Description
const `Adc_GroupType`	Group	**input**	Group Id.
const `Adc_GroupDefType*`	pChannel	**input**	Pointer to channel list array.
const `Adc_ChannelIndexType`	NumberOfChannel	**input**	Number of Channel.

3.8.4 Structs Reference

Data structures supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.2 Rev0002 .

3.8.4.1 Structure Adc_CalibrationStatusType

Structure for calibration status.

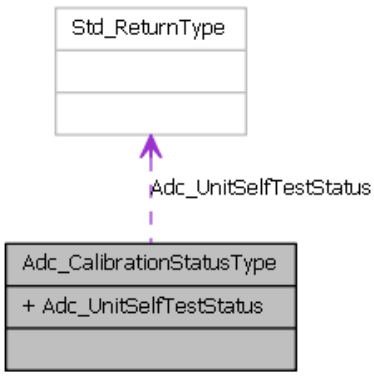

Figure 3-1. Struct Adc_CalibrationStatusType

Declaration:

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

Table 3-136. 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_Adcdig_HwUnitConfigurationType* pAdc,
                                    const Adc_GroupConfigurationType* pGroups,
                                    const Adc_Adcdig_ChannelConfigurationType** pChannels,
                                    Adc_GroupType GroupCount,
```

```
                const Adc_Adcdig_MultiConfigType Misc
} Adc_ConfigType;
```

Table 3-137. 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_GroupConfigurationType

Structure for group configuration.

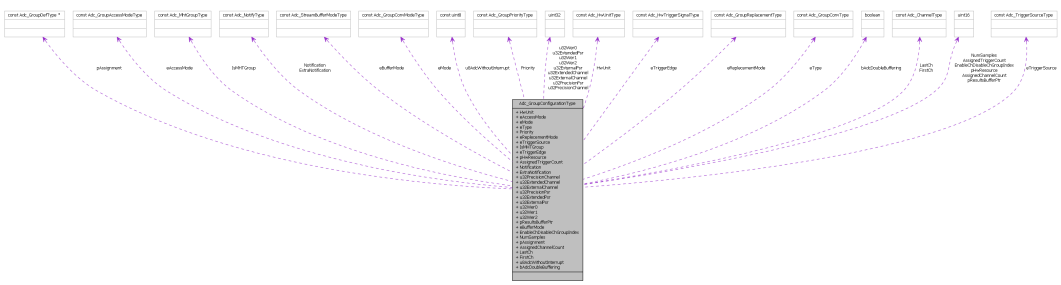

Figure 3-2. Struct Adc_GroupConfigurationType

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 eReplecementMode,
          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 3-138. Structure Adc_GroupConfigurationType member description

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

3.8.4.4 Structure Adc_Adcdig_ChannelConfigurationType

Structure for channel configuration.

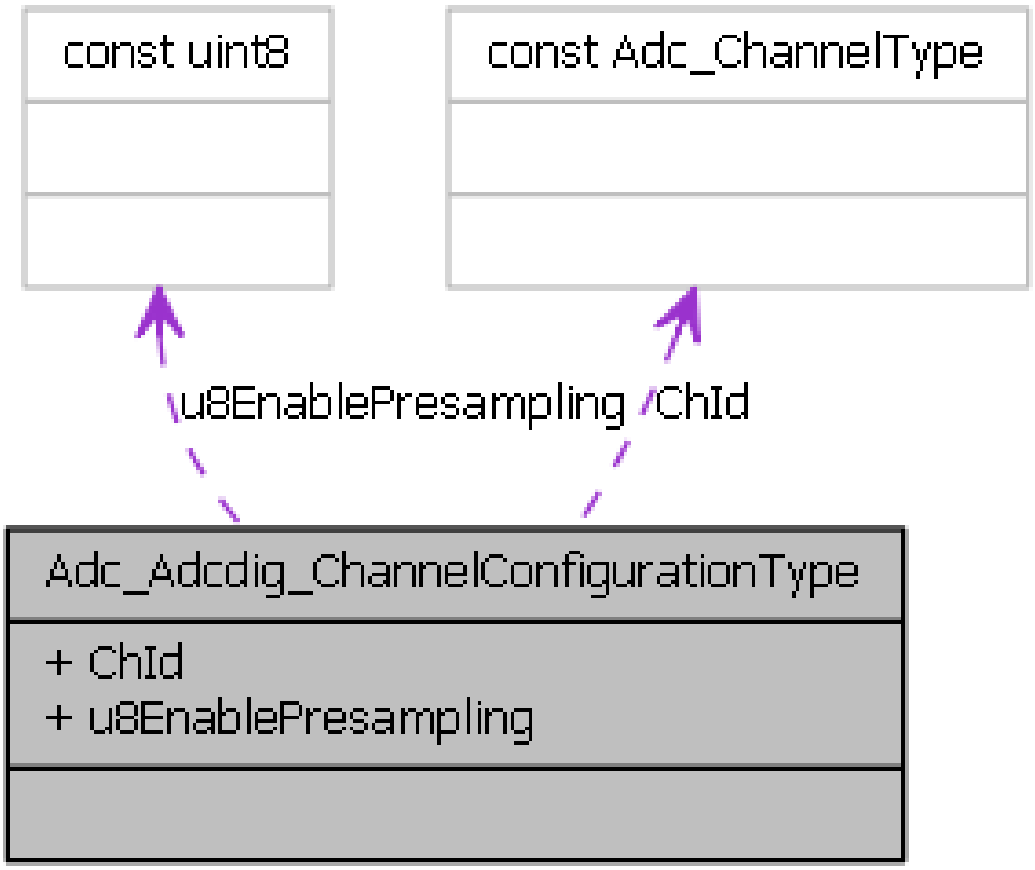

Figure 3-3. Struct Adc_Adcdig_ChannelConfigurationType

Declaration:

```
typedef struct
                        {
                          const Adc_ChannelType ChId,
                                  const uint8 u8ThReg,
                                  const Adc_WdgNotifyType WdogNotification,
                                  const uint8 u8RegIndex,
                                  const uint32 u32Mask,
                                  const uint8 u8EnablePresampling
                        } Adc_Adcdig_ChannelConfigurationType;
```

Table 3-139. Structure Adc_Adcdig_ChannelConfigurationType member description

Member	Description
ChId	Channel Id.
u8ThReg	Threshold register configured.
WdogNotification	Wdg notification function pointer.
u8RegIndex	Channel descriptions for the WDG interrupts.
u32Mask	Channel descriptions Mask for the WDG interrupts.
u8EnablePresampling	status to indicate the presampling state

3.8.4.5 Structure Adc_Adcdig_ChannelLimitCheckingType

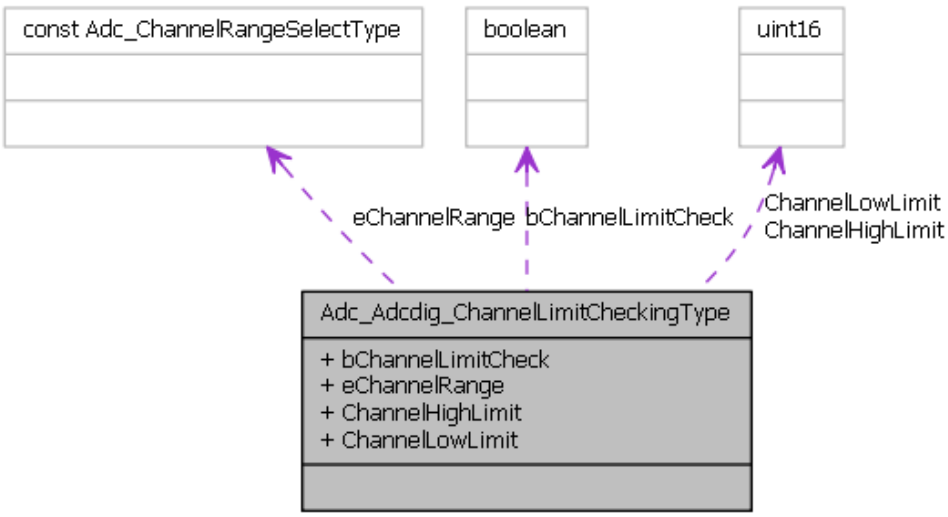

Figure 3-4. Struct Adc_Adcdig_ChannelLimitCheckingType

Declaration:

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

Table 3-140. Structure Adc_Adcdig_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.

3.8.4.6 Structure Adc_Adcdig_HwUnitConfigurationType

Structure for Adc hardware unit configuration.

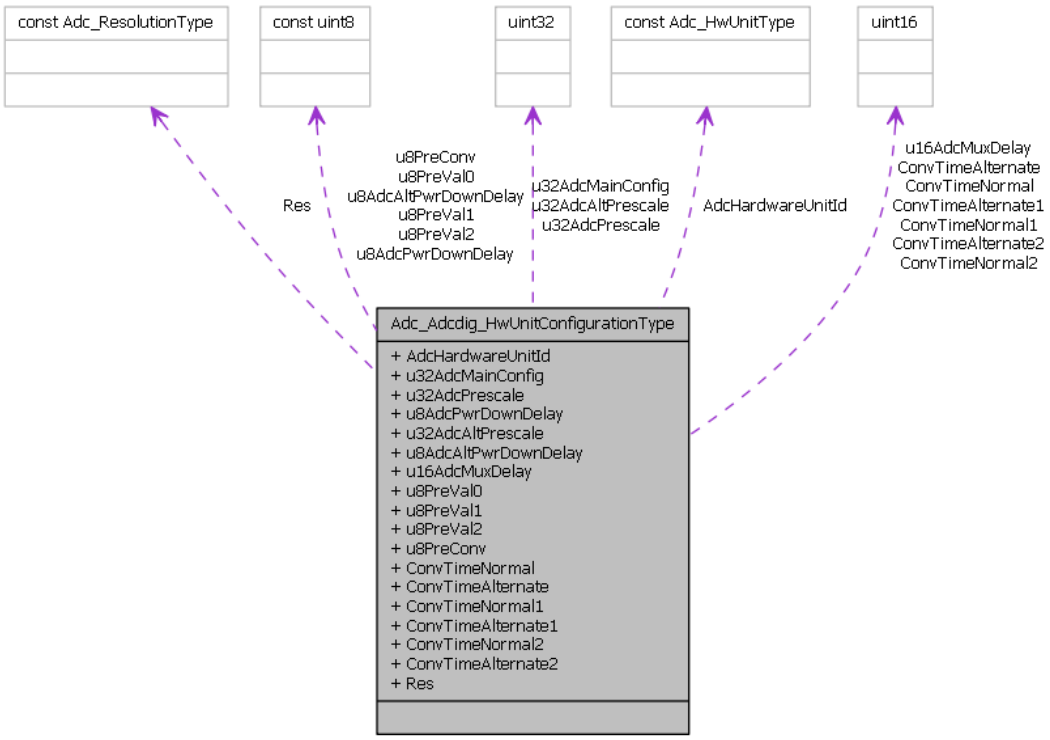

Figure 3-5. Struct Adc_Adcdig_HwUnitConfigurationType

Declaration:

```
typedef struct
                          {
                            const Adc_HwUnitType AdcHardwareUnitId,
                                    const uint32 u32AdcMainConfig,
                                    const uint32 u32AdcPrescale,
                                    const uint8 u8AdcPwrDownDelay,
                                    const uint32 u32AdcAltPrescale,
                                    const uint8 u8AdcAltPwrDownDelay,
                                    const uint8 u8PreVal0,
                                    const uint8 u8PreVal1,
                                    const uint8 u8PreConv,
                                    const Adc_ConversionTimeType ConvTimeNormal,
                                    const Adc_ConversionTimeType ConvTimeAlternate,
                                    const Adc_ConversionTimeType ConvTimeNormal1,
                                    const Adc_ConversionTimeType ConvTimeAlternate1,
                                    const Adc_ResolutionType Res,
                                    const uint8 ConfiguredThCount,
                                    const Adc_Adcdig_ThresholdConfigurationType*
pThConfigured
                          } Adc_Adcdig_HwUnitConfigurationType;
```

Table 3-141. Structure Adc_Adcdig_HwUnitConfigurationType member description

Member	Description
AdcHardwareUnitId	Adc hardware unit id.
u32AdcMainConfig	Prescaler of normal mode.
u32AdcPrescale	Power down delay.
u8AdcPwrDownDelay	Prescaler of alternate mode.
u32AdcAltPrescale	Power down delay for low power system frequency.

Table continues on the next page...

Table 3-141. Structure Adc_Adcdig_HwUnitConfigurationType member description (continued)

Member	Description
u8AdcAltPwrDownDelay	
u8PreVal0	presampling control register - preval0 field Internal voltage selection for presampling, targeting internal precision channels.
u8PreVal1	presampling control register - preconv field
u8PreConv	Conversion time CTR0.
ConvTimeNormal	
ConvTimeAlternate	Conversion time CTR1.
ConvTimeNormal1	
ConvTimeAlternate1	Resolution of the ADC hardware.
Res	Maximum of Threshold registers configured.
ConfiguredThCount	Pointer to Threshold registers configuration.
pThConfigured	

3.8.4.7 Structure Adc_Adcdig_MultiConfigType

Miscellaneous configuration structure.

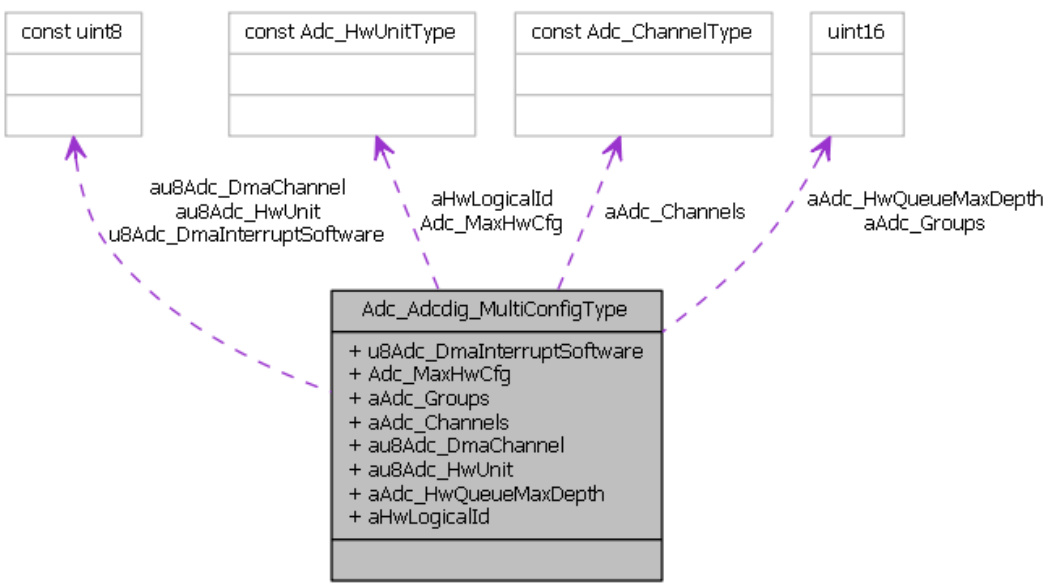

Figure 3-6. Struct Adc_Adcdig_MultiConfigType

Declaration:

```
typedef struct
                        {
                          const uint8 u8Adc_DmaInterruptSoftware[ADC_MAX_HW_UNITS],
                                  const Adc_HwUnitType Adc_MaxHwCfg,
                                  const Adc_GroupType aAdc_Groups[ADC_MAX_HW_UNITS],
```

```
                                    const Adc_ChannelType aAdc_Channels[ADC_MAX_HW_UNITS],
                                    const uint8 au8Adc_DmaChannel[ADC_MAX_HW_UNITS],
                                    const uint8 au8Adc_HwUnit[ADC_MAX_HW_UNITS],
                                    const Adc_QueueIndexType
aAdc_HwQueueMaxDepth[ADC_MAX_HW_UNITS],
                                    const Adc_HwUnitType aHwLogicalId[ADC_MAX_HW_UNITS]
                                } Adc_Adcdig_MultiConfigType;
```

Table 3-142. Structure Adc_Adcdig_MultiConfigType member description

Member	Description
u8Adc_DmaInterruptSoftware	DMA or Interrupt driven.
Adc_MaxHwCfg	Number of the maximum hw units.
aAdc_Groups	Number of groups configured for each unit.
aAdc_Channels	Number of channels of the hw unit x.
au8Adc_DmaChannel	If dma driven then indicates the dma channel number for HW UNITS.
au8Adc_HwUnit	If unit x is active STD_ON/STD_OFF.
aAdc_HwQueueMaxDepth	Maximum depth of the hw queue for each unit.
aHwLogicalId	

3.8.4.8 Structure Adc_Adcdig_ThresholdConfigurationType

Structure for watchdog threshold control configuration.

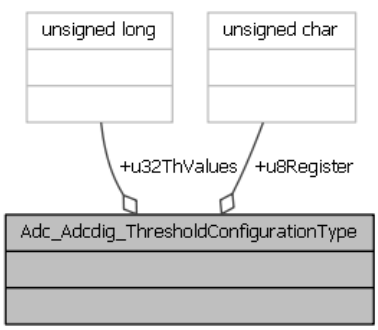

Figure 3-7. Struct Adc_Adcdig_ThresholdConfigurationType

Declaration:

```
typedef struct
                                {
                                  const uint8 u8Register,
                                          const uint32 u32ThValues
                                } Adc_Adcdig_ThresholdConfigurationType;
```

Table 3-143. Structure Adc_Adcdig_ThresholdConfigurationType member description

Member	Description
u8Register	Threshold register configured.
u32ThValues	Threshold value.

3.8.4.9 Structure Adc_Bctu_ConfigType

Main configuration structure for BCTU.

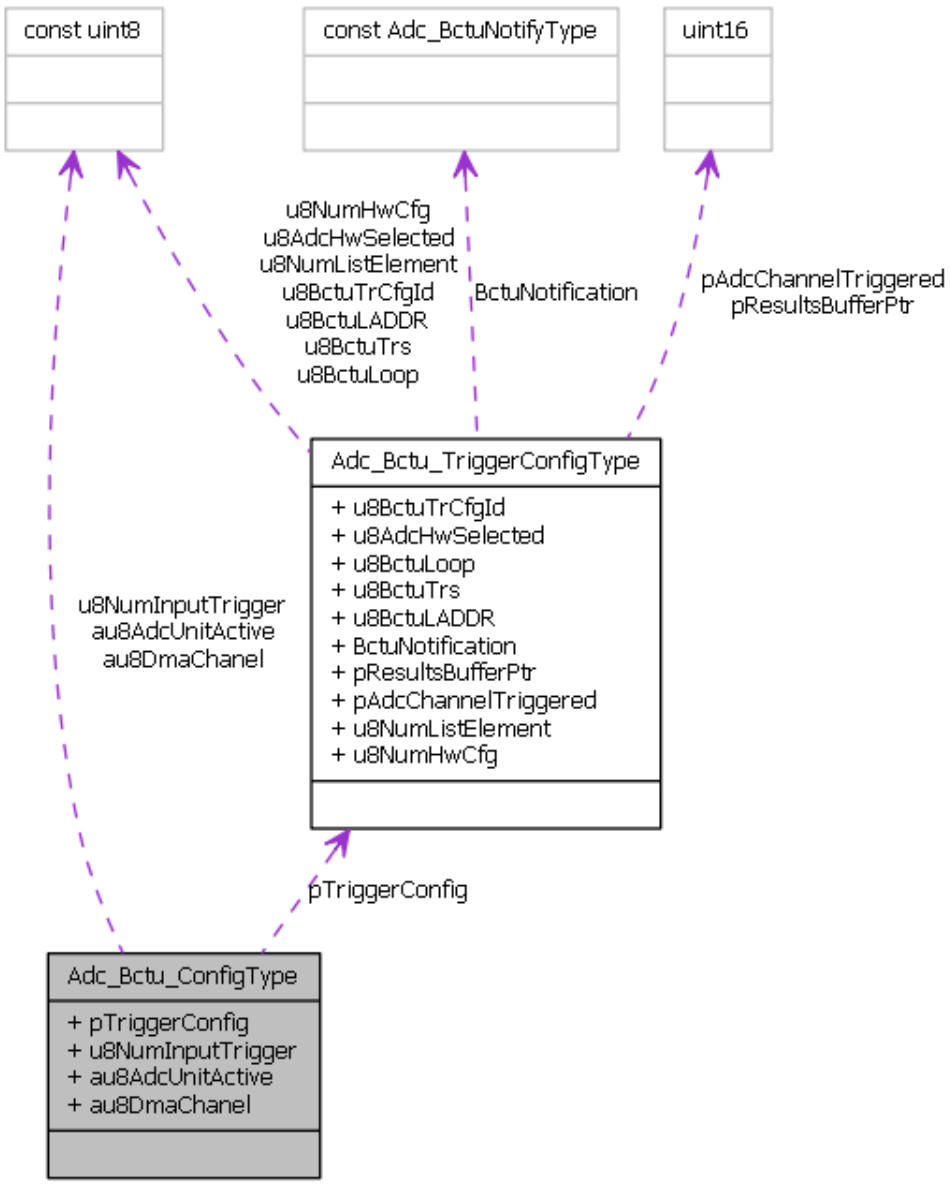

Figure 3-8. Struct Adc_Bctu_ConfigType

Declaration:

```
typedef struct
                        {
                          const Adc_Bctu_TriggerConfigType* pTriggerConfig,
                                  const uint8 u8NumInputTrigger,
                                  const uint8 au8AdcUnitActive[ADC_MAX_HW_UNITS],
                                  const uint8 au8DmaChanel[ADC_MAX_HW_UNITS]
                        } Adc_Bctu_ConfigType;
```

Table 3-144. Structure Adc_Bctu_ConfigType member description

Member	Description
pTriggerConfig	Pointer to trigger configure array.
u8NumInputTrigger	Number of input trigger.
au8AdcUnitActive	Adc unit is ON/OFF.
au8DmaChanel	Dma channel assigned for Adc unit.

3.8.4.10 Structure Adc_Bctu_TriggerConfigType

Configuration structure for input trigger.

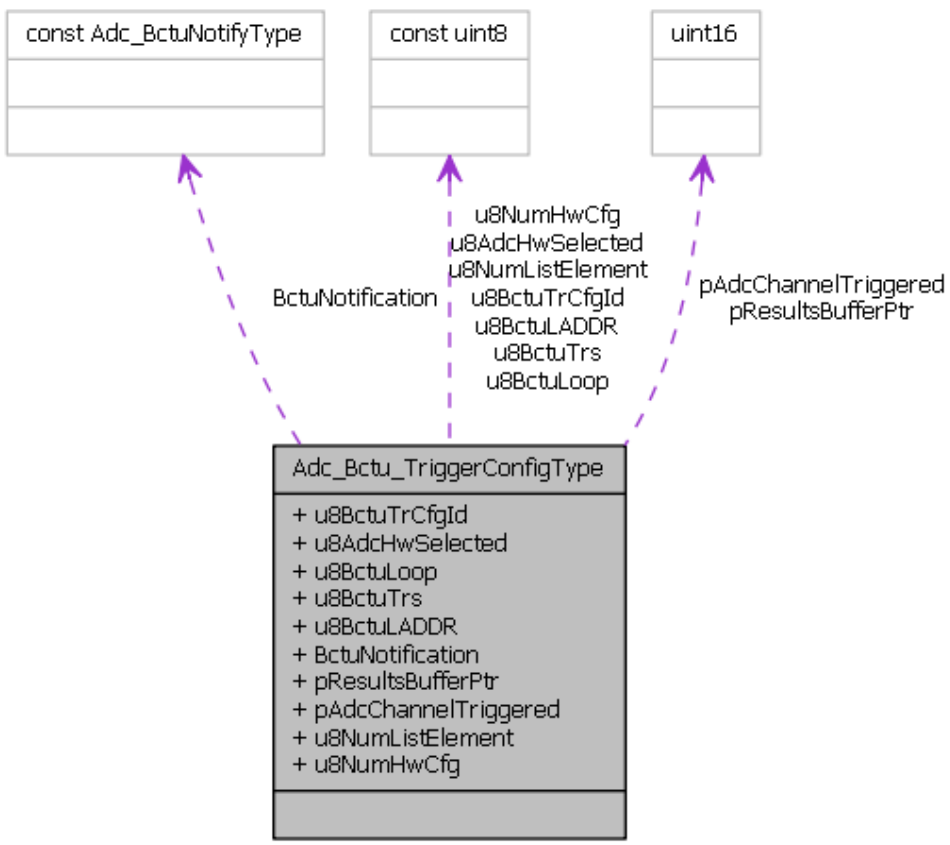

Figure 3-9. Struct Adc_Bctu_TriggerConfigType

Declaration:

```
typedef struct
                        {
                          const uint8 u8BctuTrCfgId,
                                  const uint8 u8AdcHwSelected,
                                  const uint8 u8BctuLoop,
                                  const uint8 u8BctuTrs,
                                  const uint8 u8BctuLADDR,
                                  const Adc_BctuNotifyType BctuNotification,
```

```
                                    Adc_Bctu_DataType** pResultsBufferPtr,
                                    const Adc_Bctu_ListChannelType* pAdcChannelTriggered,
                                    const uint8 u8NumListElement,
                                    const uint8 u8NumHwCfg
                            } Adc_Bctu_TriggerConfigType;
```

Table 3-145. Structure Adc_Bctu_TriggerConfigType member description

Member	Description
u8BctuTrCfgId	BCTU Trigger configuration number.
u8AdcHwSelected	BCTU ADC HW selection.
u8BctuLoop	BCTU Loop bit.
u8BctuTrs	BCTU trigger resolution.
u8BctuLADDR	BCTU channel value or addr of the first LIST position.
BctuNotification	Bctu notification function pointer.
pResultsBufferPtr	pointer to user result buffer array
pAdcChannelTriggered	pointer to list channel array
u8NumListElement	Number of element in the LIST.
u8NumHwCfg	Number of Adc unit configured.

3.8.5 Types Reference

Types supported by the driver are as per AUTOSAR Adc Driver software specification Version 4.2 Rev0002 .

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
- AdcConfigSet
- AdcGeneral
- AdcPublishedInformation
- CommonPublishedInformation
- NonAutosar
- AdcDemEventParameterRefs
- AdcInterrupt

Table 4-1. Revision table

Revision	Date
4.6.0	2014-10-31

4.2 Form IMPLEMENTATION_CONFIG_VARIANT

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

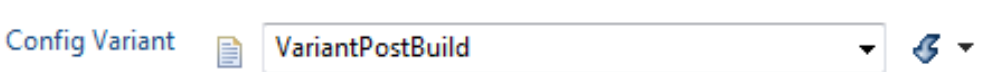

Figure 4-1. Tresos Plugin snapshot for IMPLEMENTATION_CONFIG_VARIANT form.

Table 4-2. 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.

Included forms :

- Form AdcPowerStateConfig

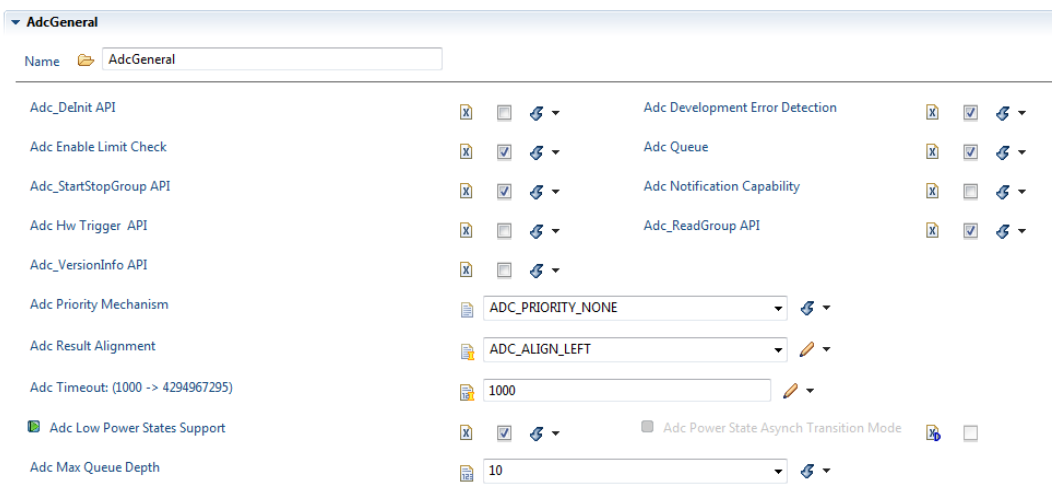

Figure 4-2. Tresos Plugin snapshot for AdcGeneral form.

4.3.1 AdcDeInitApi (AdcGeneral)

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

Table 4-3. Attribute AdcDeInitApi (AdcGeneral) detailed description

Property	Value
Label	Adc_DeInit API
Type	BOOLEAN

Table continues on the next page...

Table 4-3. Attribute AdcDeInitApi (AdcGeneral) detailed description (continued)

Property	Value
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

4.3.2 AdcDevErrorDetect (AdcGeneral)

Enable/Disable Development Error Detection.

Table 4-4. 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 4-5. 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 4-6. Attribute AdcEnableQueuing (AdcGeneral) detailed description

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

4.3.5 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

4.3.6 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
Type	BOOLEAN
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	true

4.3.7 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

4.3.8 AdcReadGroupApi (AdcGeneral)

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

Table 4-10. Attribute AdcReadGroupApi (AdcGeneral) detailed description

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

4.3.9 AdcVersionInfoApi (AdcGeneral)

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

Table 4-11. Attribute AdcVersionInfoApi (AdcGeneral) detailed description

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

4.3.10 AdcPriorityImplementation (AdcGeneral)

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

Table 4-12. 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

4.3.11 AdcResultAlignment (AdcGeneral)

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

Table 4-13. 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 AdcTimeout (AdcGeneral)

This is a timeout value which is used to wait till - the conversion is not aborted - ADC hardware is not entered in power down state - ADC hardware is not entered in idle state 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	Custom
Symbolic Name	false
Default	65535
Invalid	Range <=4294967295 >=1000

4.3.13 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.

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

4.3.14 AdcPowerStateAsynchTransitionMode (AdcGeneral)

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

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

Property	Value
Label	Adc Power State Asynch Transition Mode
Type	BOOLEAN
Origin	AUTOSAR_ECUC

Table continues on the next page...

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

Property	Value
Symbolic Name	false
Default	false

4.3.15 AdcPriorityQueueMaxDepth (AdcGeneral)

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

Table 4-17. Attribute AdcPriorityQueueMaxDepth (AdcGeneral) detailed description

Property	Value
Label	Adc Max Queue Depth
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	1
Invalid	Range <=65535 >=1

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.

Is included by form : Form AdcGeneral

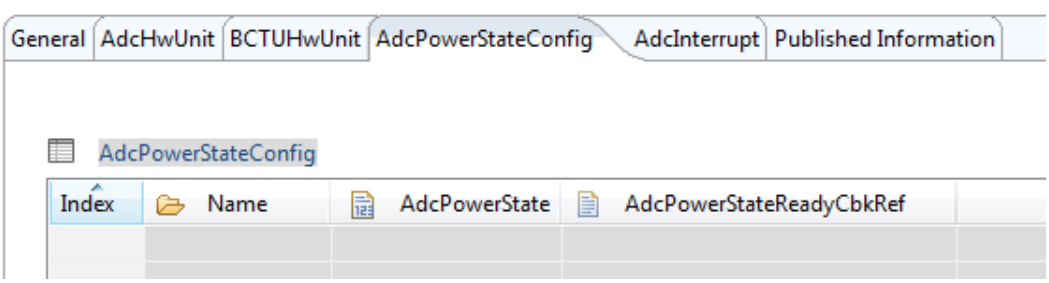

Figure 4-3. Tresos Plugin snapshot for AdcPowerStateConfig form.

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.

Table 4-18. Attribute AdcPowerState (AdcPowerStateConfig) detailed description

Property	Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	true
Default	0

4.3.16.2 AdcPowerStateReadyCbkRef (AdcPowerStateConfig)

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

Table 4-19. Attribute AdcPowerStateReadyCbkRef (AdcPowerStateConfig) detailed description

Property	Value
Type	FUNCTION-NAME
Origin	AUTOSAR_ECUC
Symbolic Name	false
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.

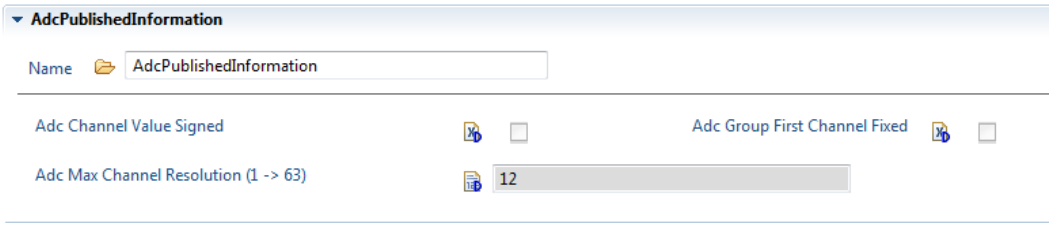

Figure 4-4. Tresos Plugin snapshot for AdcPublishedInformation form.

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 4-20. 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 4-21. 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 4-22. 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 CommonPublishedInformation

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

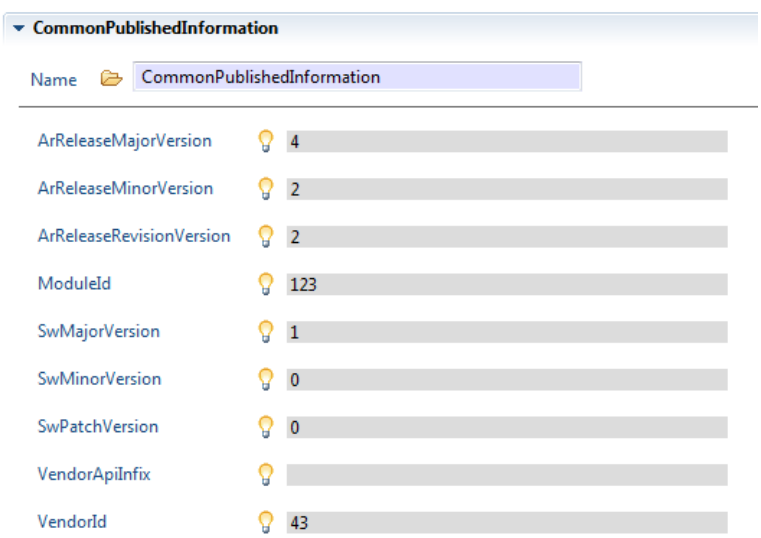

Figure 4-5. Tresos Plugin snapshot for CommonPublishedInformation form.

4.5.1 ArReleaseMajorVersion (CommonPublishedInformation)

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

Table 4-23. Attribute ArReleaseMajorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	AUTOSAR Major Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	4
Invalid	Range >=4 <=4

4.5.2 ArReleaseMinorVersion (CommonPublishedInformation)

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

Table 4-24. Attribute ArReleaseMinorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	AUTOSAR Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	2
Invalid	Range >=2 <=2

4.5.3 ArReleaseRevisionVersion (CommonPublishedInformation)

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

Table 4-25. Attribute ArReleaseRevisionVersion (CommonPublishedInformation) detailed description

Property	Value
Label	AUTOSAR Release Revision Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	2
Invalid	Range >=2 <=2

4.5.4 ModuleId (CommonPublishedInformation)

Module ID of this module from Module List.

Table 4-26. Attribute ModuleId (CommonPublishedInformation) detailed description

Property	Value
Label	Module Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	123
Invalid	Range >=123 <=123

4.5.5 SwMajorVersion (CommonPublishedInformation)

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

Table 4-27. Attribute SwMajorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	Software Major Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false

Table continues on the next page...

Table 4-27. Attribute SwMajorVersion (CommonPublishedInformation) detailed description (continued)

Property	Value
Default	1
Invalid	Range >=1 <=1

4.5.6 SwMinorVersion (CommonPublishedInformation)

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

Table 4-28. Attribute SwMinorVersion (CommonPublishedInformation) detailed description

Property	Value
Label	Software Minor Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range >=0 <=0

4.5.7 SwPatchVersion (CommonPublishedInformation)

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

Table 4-29. Attribute SwPatchVersion (CommonPublishedInformation) detailed description

Property	Value
Label	Software Patch Version
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range >=0 <=0

4.5.8 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 Can_Write defined in the SWS will translate to Can_123_v11r456Write. This parameter is mandatory for all modules with upper multiplicity > 1. It shall not be used for modules with upper multiplicity =1.

Table 4-30. Attribute VendorApiInfix (CommonPublishedInformation) detailed description

Property	Value
Label	Vendor Api Infix
Type	STRING_LABEL
Origin	Custom
Symbolic Name	false
Default	
Enable	false

4.5.9 VendorId (CommonPublishedInformation)

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

Table 4-31. Attribute VendorId (CommonPublishedInformation) detailed description

Property	Value
Label	Vendor Id
Type	INTEGER_LABEL
Origin	Custom
Symbolic Name	false
Default	43
Invalid	Range >=43 <=43

4.6 Form NonAutosar

Non Autosar API settings.

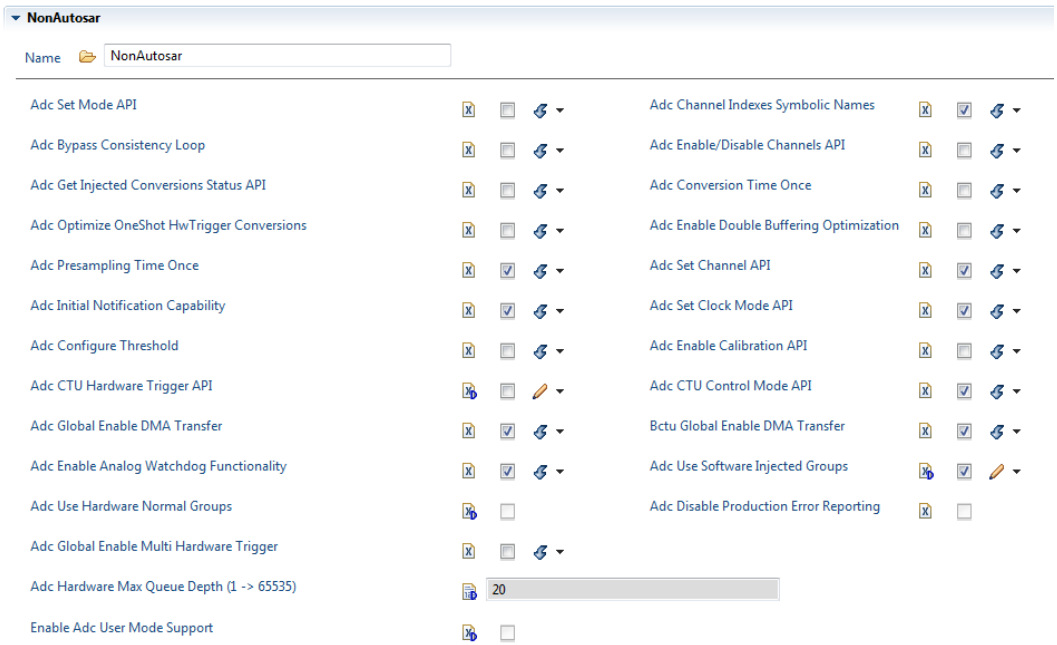

Figure 4-6. Tresos Plugin snapshot for NonAutosar form.

4.6.1 AdcSetModeApi (NonAutosar)

Adds/removes the non-autosar implementation api Adc_SetModeApi() from the code.

Table 4-32. Attribute AdcSetModeApi (NonAutosar) detailed description

Property	Value
Label	Adc Set Mode API
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.2 AdcEnableGroupDependentChannelNames (NonAutosar)

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 4-33. Attribute AdcEnableGroupDependentChannelNames (NonAutosar) detailed description

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

4.6.3 AdcBypassConsistencyLoop (NonAutosar)

This is used to increase ADC performances. If checked 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 4-34. Attribute AdcBypassConsistencyLoop (NonAutosar) detailed description

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

4.6.4 AdcEnableChDisableChApi (NonAutosar)

Note

Enable/disable the non-autosar implementation api(s) Adc_EnableChannel() and Adc_DisableChannel() in ADC driver.

This is an Implementation Specific Parameter.

Table 4-35. Attribute AdcEnableChDisableChApi (NonAutosar) detailed description

Property	Value
Label	Adc Enable/Disable Channels API
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.5 AdcGetInjectedConvStatusApi (NonAutosar)

Note

Enable/disable the non-autosar API Adc_GetInjectedConversionStatus() in ADC driver.

This is an Implementation Specific Parameter.

Table 4-36. Attribute AdcGetInjectedConvStatusApi (NonAutosar) detailed description

Property	Value
Label	Adc Get Injected Conversions Status API
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.6 AdcConvTimeOnce (NonAutosar)

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 4-37. Attribute AdcConvTimeOnce (NonAutosar) detailed description

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

4.6.7 AdcOptimizeOneShotHwTriggerConversions (NonAutosar)

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 configrued.

Table 4-38. Attribute AdcOptimizeOneShotHwTriggerConversions (NonAutosar) detailed description

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

4.6.8 AdcEnableDoubleBufferingOptimization (NonAutosar)

Implementation Specific Parameter. Enable/Disable The Adc driver double buffering optimization for HW Triggered groups, OneShot, Streamin access, Circular Buffer. Also, all groups that enable this feature must have at only 1 channels configrued.

Table 4-39. Attribute AdcEnableDoubleBufferingOptimization (NonAutosar) detailed description

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

4.6.9 AdcPreSamplingOnce (NonAutosar)

Implementation Specific Parameter. Enable/Disable one time setting of the registers. If Enabled, the setting of the presampling time registers will be done only once in Adc_Init() function for the configured hardware unit.

Table 4-40. Attribute AdcPreSamplingOnce (NonAutosar) detailed description

Property	Value
Label	Adc Presampling Time Once
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.10 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 4-41. Attribute AdcEnableSetChannel (NonAutosar) detailed description

Property	Value
Label	Adc Set Channel API
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.11 AdcEnableInitialNotification (NonAutosar)

Note

Enable/disable an extra notification to be called for each Adc Group conversion. This feature is intended to be used together with Adc_SetChannel service. The initial notification can be used by the user application to call Adc_SetChannel API before Adc driver updates the HW configuration for the next conversion.

Table 4-42. Attribute AdcEnableInitialNotification (NonAutosar) detailed description

Property	Value
Label	Adc Initial Notification Capability
Type	BOOLEAN
Origin	Custom

Table continues on the next page...

Table 4-42. Attribute AdcEnableInitialNotification (NonAutosar) detailed description (continued)

Property	Value
Symbolic Name	false
Default	false

4.6.12 AdcEnableDualClockMode (NonAutosar)

Adds/removes the Dual Clock mode service Adc_SetClockMode from the code. Also it enables the programmation of Conversion Timing registers in Adc_SetClockMode.

Table 4-43. Attribute AdcEnableDualClockMode (NonAutosar) detailed description

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

4.6.13 AdcEnableThresholdConfiguration (NonAutosar)

Note

Enable/disable the non-autosar APIs Adc_ConfigureThreshold() in ADC driver.

This is an Implementation Specific Parameter.

Table 4-44. Attribute AdcEnableThresholdConfiguration (NonAutosar) detailed description

Property	Value
Label	Adc Configure Threshold
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.14 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 is an Implementation Specific Parameter.

Table 4-45. Attribute AdcEnableCalibration (NonAutosar) detailed description

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

4.6.15 AdcEnableCtuTrigNonAutosarApi (NonAutosar)

This is used to enable the non Autosar API for the hardware triggered group. If this parameter is enabled than Adc_EnableCTUTrigger(), Adc_DisableCTUTrigger() and Adc_HwResultReadGroup() will be available in the driver code. This is an Implementation Specific Parameter. When this parameter is enabled, the result buffer is no longer to be used to read the results as the result will be directly read from HW registers. When this parameter is disabled, normal functionlaity shall be executed.

Table 4-46. Attribute AdcEnableCtuTrigNonAutosarApi (NonAutosar) detailed description

Property	Value
Label	Adc CTU Hardware Trigger API
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.16 AdcEnableCtuControlModeApi (NonAutosar)

Note

This is used to enable the non Autosar API for the enabling and disabling CTU control mode for an ADC unit.. If this parameter is enabled than Adc_EnableCtuControlMode(), Adc_DisableCtuControlMode() will be available in the driver code. When a unit works in CTU control mode, no other conversions shall run in parallel(Adc). The only conversions occurring shall be the ones defined in the CTU configuration.

If AdcEnableCtuControlModeApi is enabled, BCTU must be configured. This is an Implementation Specific Parameter.

Table 4-47. Attribute AdcEnableCtuControlModeApi (NonAutosar) detailed description

Property	Value
Label	Adc CTU Control Mode API
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.17 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 configure 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 4-48. Attribute AdcEnableDmaTrasferMode (NonAutosar) detailed description

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

4.6.18 BctuEnableDmaTrasferMode (NonAutosar)

Note

This parameter globally enables the possibility to configure DMA transfer for BCTU control mode. If this parameter is disabled then DMA handling code will be removed at pre-compile time and DMA transfer cannot be configure for any BCTU 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 4-49. Attribute BctuEnableDmaTrasferMode (NonAutosar) detailed description

Property	Value
Label	Bctu Global Enable DMA Transfer
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.19 AdcEnableWatchdogFunctionality (NonAutosar)

Note

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

This is an Implementation Specific Parameter.

Table 4-50. Attribute AdcEnableWatchdogFunctionality (NonAutosar) detailed description

Property	Value
Label	Adc Enable Analog Watchdog Functionality
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.20 AdcUseSoftwareInjectedGroups (NonAutosar)

Note

This parameter defines if Software Injected Groups are used in any Hardware Unit, any variant. It needs to be enabled if Software Injected Groups are needed. If Software Injected Groups are not needed, this parameter should be disabled - for code optimizations.

This is an Implementation Specific Parameter.

Table 4-51. Attribute AdcUseSoftwareInjectedGroups (NonAutosar) detailed description

Property	Value
Label	Adc Use Software Injected Groups
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.21 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.Normal Hardware conversions are not supported on this platform.

This is an Implementation Specific Parameter.

Table 4-52. Attribute AdcUseHardwareNormalGroups (NonAutosar) detailed description

Property	Value
Label	Adc Use Hardware Normal Groups
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.22 AdcDisableDemReportErrorStatus (NonAutosar)

Enable/Disable Dem error reporting.

Table 4-53. Attribute AdcDisableDemReportErrorStatus (NonAutosar) detailed description

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

4.6.23 AdcEnableMultiHardwareTrigger (NonAutosar)

Note

This parameter globally enables the possibility to configure multi hardware trigger. If this parameter is disabled then multi hardware trigger handling code will be removed at pre-compile time and multi hardware cannot be configured for any Group in any variant.

This is an Implementation Specific Parameter.

Table 4-54. Attribute AdcEnableMultiHardwareTrigger (NonAutosar) detailed description

Property	Value
Label	Adc Global Enable Multi Hardware Trigger
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.6.24 AdcHardwareQueueMaxDepth (NonAutosar)

Maximum depth of hardware queue used in case of multi hardware trigger used.

Table 4-55. Attribute AdcHardwareQueueMaxDepth (NonAutosar) detailed description

Property	Value
Label	Adc Hardware Max Queue Depth
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	1
Invalid	Range

Table 4-55. Attribute AdcHardwareQueueMaxDepth (NonAutosar) detailed description

Property	Value
	<=65535 >=1

4.6.25 AdcEnableUserModeSupport (NonAutosar)

When this parameter is enabled, the Adc module will adapt to run from User Mode.

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

Table 4-56. Attribute AdcEnableUserModeSupport (NonAutosar) detailed description

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

4.7 Form AdcDemEventParameterRefs

Container for the references to DemEventParameter elements which shall be invoked using the Dem_ReportErrorStatus 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.

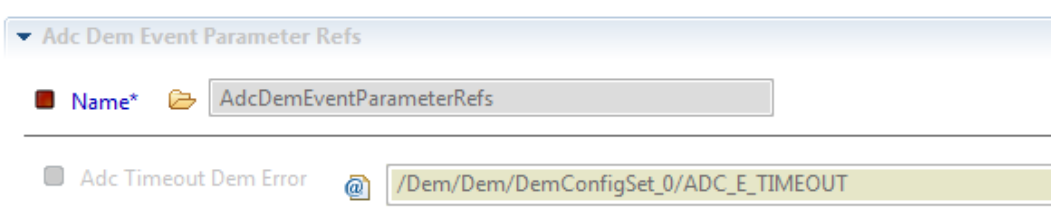

Figure 4-7. Tresos Plugin snapshot for AdcDemEventParameterRefs form.

4.7.1 ADC_E_TIMEOUT (AdcDemEventParameterRefs)

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

Table 4-57. Attribute ADC_E_TIMEOUT (AdcDemEventParameterRefs) detailed description

Property	Value
Label	Adc Timeout Dem Error
Type	SYMBOLIC-NAME-REFERENCE
Origin	Custom

4.8 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 Watchdong interrupts. These settings are used for optimizing the code size by removing the interrupt handling code for interrupts that are not needed. .

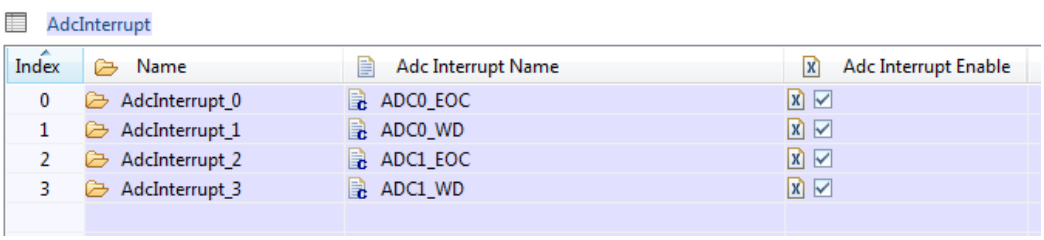

Figure 4-8. Tresos Plugin snapshot for AdcInterrupt form.

4.8.1 AdcInterruptSource (AdcInterrupt)

The name of the interrupt.

Note: Implementation Specific Parameter.

Table 4-58. Attribute AdcInterruptSource (AdcInterrupt) detailed description

Property	Value
Label	Adc Interrupt Name
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

4.8.2 AdcInterruptEnable (AdcInterrupt)

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

Table 4-59. Attribute AdcInterruptEnable (AdcInterrupt) detailed description

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

4.9 Form AdcConfigSet

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

Included forms :

- Form AdcHwUnit
- Form BCTUHwUnit

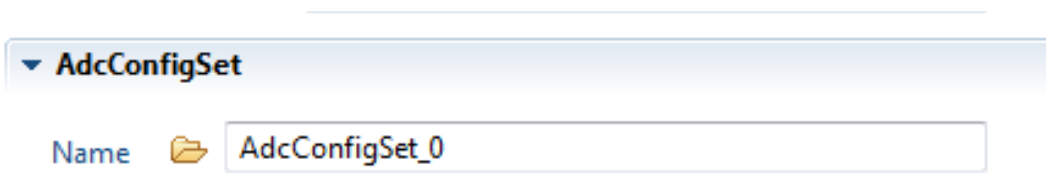

Figure 4-9. Tresos Plugin snapshot for AdcConfigSet form.

4.9.1 Form AdcHwUnit

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

Is included by form : Form AdcConfigSet

Included forms :

- Form AdcNormalConvTimings
- Form AdcAlternateConvTimings

- Form AdcChannel
- Form AdcGroup
- Form AdcThresholdControl

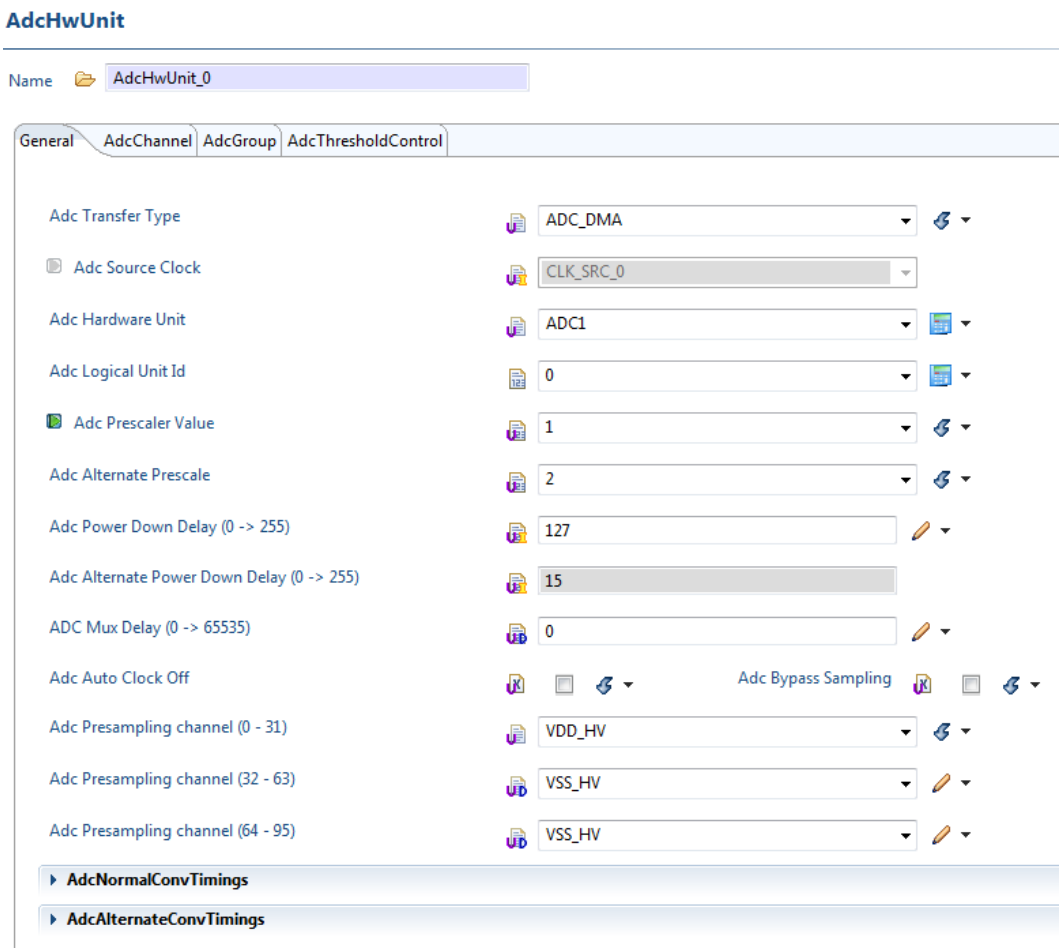

Figure 4-10. Tresos Plugin snapshot for AdcHwUnit form.

4.9.1.1 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 responsibilty to read the results from registers directly by calling Adc_ReadGroup

Table 4-60. Attribute AdcTransferType (AdcHwUnit) detailed description

Property	Value
Label	Adc Transfer Type

Table continues on the next page...

Table 4-60. Attribute AdcTransferType (AdcHwUnit) detailed description (continued)

Property	Value
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	ADC_INTERRUPT
Range	ADC_DMA ADC_INTERRUPT

4.9.1.2 AdcClockSource (AdcHwUnit)

This value should be selected in the MCU Driver. This parameter is not used by the current implementation.

Table 4-61. Attribute AdcClockSource (AdcHwUnit) detailed description

Property	Value
Label	Adc Source Clock
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	CLK_SRC_0
Range	CLK_SRC_0

4.9.1.3 AdcHwUnitId (AdcHwUnit)

Specifies the used ADC Hardware Unit.

Table 4-62. Attribute AdcHwUnitId (AdcHwUnit) detailed description

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

4.9.1.4 AdcLogicalUnitId (AdcHwUnit)

Specifies the Logical id of the Hardware Unit.

Table 4-63. Attribute AdcLogicalUnitId (AdcHwUnit) detailed description

Property	Value
Label	Adc Logical Unit Id
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <ecu:get('Adc.AdcConfigSet.AdcHwUnit') >=0

4.9.1.5 AdcPrescale (AdcHwUnit)

The Prescaler value for NORMAL mode. Only 1 or 2 are allowed.

- 1: ADC clock frequency is equal to bus clock.
- 2: ADC clock frequency is half of bus clock.

Table 4-64. Attribute AdcPrescale (AdcHwUnit) detailed description

Property	Value
Label	Adc Prescaler Value
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	1
Invalid	Range <=2 >=1 <=65535 >=0

4.9.1.6 AdcAltPrescale (AdcHwUnit)

The Prescaler value for ALTERNATE mode. Only 1 or 2 are allowed.

- 1: ADC clock frequency is equal to bus clock.
- 2: ADC clock frequency is half of bus clock.

Table 4-65. Attribute AdcAltPrescale (AdcHwUnit) detailed description

Property	Value
Label	Adc Alternate Prescale
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	2
Invalid	Range <=2 >=1

4.9.1.7 AdcPowerDownDelay (AdcHwUnit)

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

Table 4-66. Attribute AdcPowerDownDelay (AdcHwUnit) detailed description

Property	Value
Label	Adc Power Down Delay
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	15
Invalid	Range >=0 <=255

4.9.1.8 AdcAltPowerDownDelay (AdcHwUnit)

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

Table 4-67. Attribute AdcAltPowerDownDelay (AdcHwUnit) detailed description

Property	Value
Label	Adc Alternate Power Down Delay
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	15

Table continues on the next page...

Table 4-67. Attribute AdcAltPowerDownDelay (AdcHwUnit) detailed description (continued)

Property	Value
Invalid	Range >=0 <=255

4.9.1.9 AdcMuxDelay (AdcHwUnit)

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

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

The decode signal delay is calculated as (DSD x 1/Frequency of Adc_Clock).

The DSDR register is 12-bit.

Note: This is an Implementation Specific Parameter.

Table 4-68. Attribute AdcMuxDelay (AdcHwUnit) detailed description

Property	Value
Label	ADC Mux Delay
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	15
Invalid	Range >=0 <=65535

4.9.1.10 AdcAutoClockOff (AdcHwUnit)

Enables/disables the auto-clock-off features.

Table 4-69. Attribute AdcAutoClockOff (AdcHwUnit) detailed description

Property	Value
Label	Adc Auto Clock Off
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.9.1.11 AdcBypassSampling (AdcHwUnit)

When true, this parameter bypasses the sampling phase for all the presampling enabled channels, for the current hardware unit. The normal operation sequence on the presampling enabled channels will be: Presampling -> Conversion. Sampling will be bypassed and conversion of presampled data will be done.

Table 4-70. Attribute AdcBypassSampling (AdcHwUnit) detailed description

Property	Value
Label	Adc Bypass Sampling
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.9.1.12 AdcPresamplingInternalSignal0 (AdcHwUnit)

Select the Adc presampling internal voltage for the internal precision channels on current hardware unit: AdcPresamplingInternalSignal0 shall take the following values:

- VSS_HV - Presample voltage 1: VSS_HV_ADC0 (SAR ADC 0 ground), VSS_HV_ADC1 (SAR ADC 1 ground).
- VDD_HV - Presample voltage 2: VDD_HV_ADC0/8 (SAR ADC 0 supply divided by 8), VDD_HV_ADC1/8 (SAR ADC 1 supply divided by 8).
- VREFL - Presample voltage 3: VREFL.
- VDD_HV_REF_ADC1 - Presample voltage 4: VDD_HV_ADC1_REF ADC1 (SAR ADC 1 reference high).

Table 4-71. Attribute AdcPresamplingInternalSignal0 (AdcHwUnit) detailed description

Property	Value
Label	Adc Presampling channel (0 - 31)
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	VSS_HV
Range	VSS_HV VDD_HV VREFL VDD_HV_REF_ADC1

4.9.1.13 AdcPresamplingInternalSignal1 (AdcHwUnit)

Select the Adc presampling internal voltage for the extended channels or temparature sensor channel on current hardware unit: AdcPresamplingInternalSignal1 shall take the following values:

- VSS_HV - Presample voltage 1: VSS_HV_ADC0 (SAR ADC 0 ground), VSS_HV_ADC1 (SAR ADC 1 ground).
- VDD_HV - Presample voltage 2: VDD_HV_ADC0/8 (SAR ADC 0 supply divided by 8), VDD_HV_ADC1/8 (SAR ADC 1 supply divided by 8).
- VREFL - Presample voltage 3: VREFL.
- VDD_HV_REF_ADC1 - Presample voltage 4: VDD_HV_ADC1_REF ADC1 (SAR ADC 1 reference high).

Table 4-72. Attribute AdcPresamplingInternalSignal1 (AdcHwUnit) detailed description

Property	Value
Label	Adc Presampling channel (32 - 63)
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	VSS_HV
Range	VSS_HV VDD_HV VREFL VDD_HV_REF_ADC1

4.9.1.14 AdcPresamplingInternalSignal2 (AdcHwUnit)

Select the Adc presampling internal voltage for the external channels on current hardware unit: AdcPresamplingInternalSignal2 shall take the following values:

- VSS_HV - Presample voltage 1: VSS_HV_ADC0 (SAR ADC 0 ground), VSS_HV_ADC1 (SAR ADC 1 ground).
- VDD_HV - Presample voltage 2: VDD_HV_ADC0/8 (SAR ADC 0 supply divided by 8), VDD_HV_ADC1/8 (SAR ADC 1 supply divided by 8).
- VREFL - Presample voltage 3: VREFL.
- VDD_HV_REF_ADC1 - Presample voltage 4: VDD_HV_ADC1_REF ADC1 (SAR ADC 1 reference high).

Table 4-73. Attribute AdcPresamplingInternalSignal2 (AdcHwUnit) detailed description

Property	Value
Label	Adc Presampling channel (64 - 95)
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	VSS_HV
Range	VSS_HV VDD_HV VREFL VDD_HV_REF_ADC1

4.9.1.15 Form AdcChannel

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

Is included by form : Form AdcHwUnit

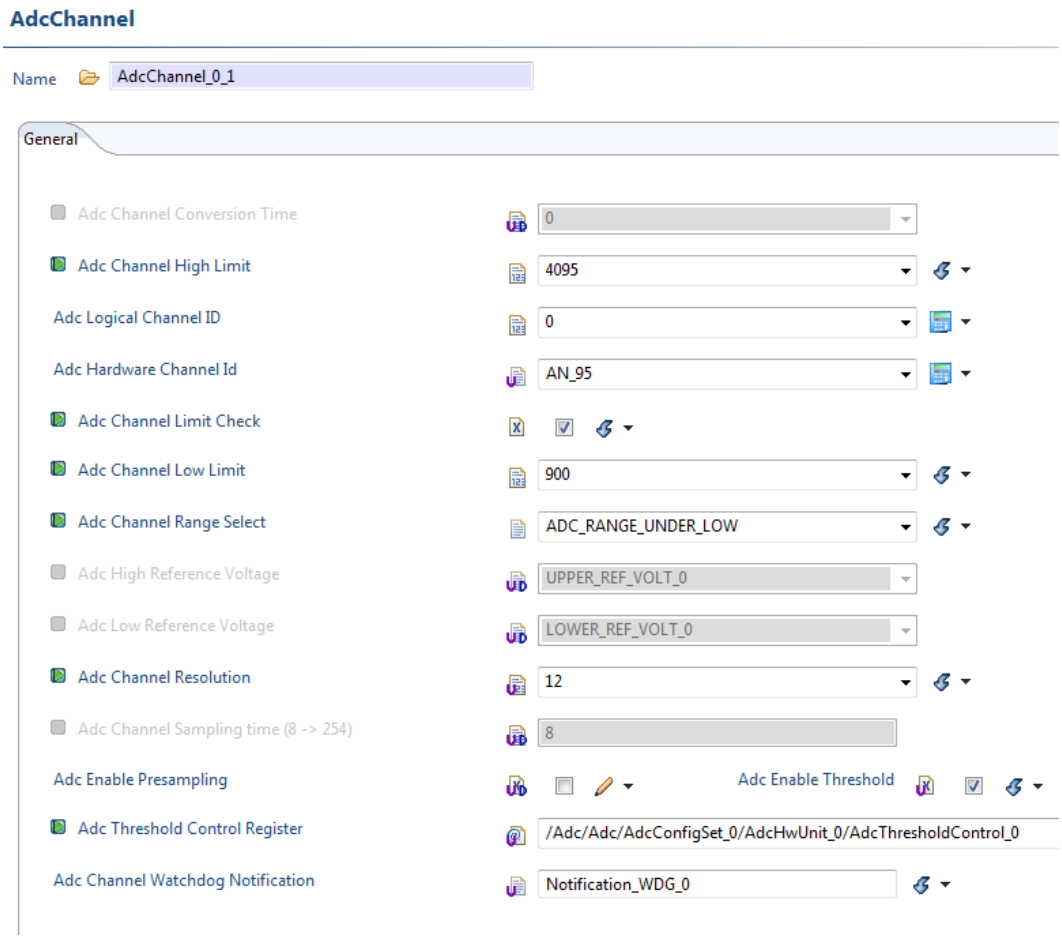

Figure 4-11. Tresos Plugin snapshot for AdcChannel form.

4.9.1.15.1 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 4-74. Attribute AdcChannelConvTime (AdcChannel) detailed description

Property	Value
Label	Adc Channel Conversion Time
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0

4.9.1.15.2 AdcChannelHighLimit (AdcChannel)

High limit - used for limit checking.

Table 4-75. Attribute AdcChannelHighLimit (AdcChannel) detailed description

Property	Value
Label	Adc Channel High Limit
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0

4.9.1.15.3 AdcLogicalChannelId (AdcChannel)

This is the logical Id of the ADC channel.

Table 4-76. Attribute AdcLogicalChannelId (AdcChannel) detailed description

Property	Value
Label	Adc Logical Channel ID
Type	INTEGER
Origin	Custom
Symbolic Name	false
Invalid	Range <=1024 >=0

4.9.1.15.4 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 4-77. Attribute AdcChannelId (AdcChannel) detailed description

Property	Value
Label	Adc Hardware Channel Id
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false

4.9.1.15.5 AdcChannelLimitCheck (AdcChannel)

Enables or disables limit checking for an ADC channel.

Table 4-78. Attribute AdcChannelLimitCheck (AdcChannel) detailed description

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

4.9.1.15.6 AdcChannelLowLimit (AdcChannel)

Low limit - used for limit checking.

Table 4-79. Attribute AdcChannelLowLimit (AdcChannel) detailed description

Property	Value
Label	Adc Channel Low Limit
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0

4.9.1.15.7 AdcChannelRangeSelect (AdcChannel)

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

Table 4-80. Attribute AdcChannelRangeSelect (AdcChannel) detailed description

Property	Value
Label	Adc Channel Range Select
Type	ENUMERATION
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

Table 4-80. Attribute AdcChannelRangeSelect (AdcChannel) detailed description

Property	Value
	ADC_RANGE_OVER_HIGH ADC_RANGE_UNDER_LOW

4.9.1.15.8 AdcChannelRefVoltsrcHigh (AdcChannel)

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

Table 4-81. Attribute AdcChannelRefVoltsrcHigh (AdcChannel) detailed description

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

4.9.1.15.9 AdcChannelRefVoltsrcLow (AdcChannel)

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

Table 4-82. Attribute AdcChannelRefVoltsrcLow (AdcChannel) detailed description

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

4.9.1.15.10 AdcChannelResolution (AdcChannel)

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

Table 4-83. Attribute AdcChannelResolution (AdcChannel) detailed description

Property	Value
Label	Adc Channel Resolution
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	10
Invalid	Range <=63 >=1

4.9.1.15.11 AdcChannelSampTime (AdcChannel)

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

Table 4-84. Attribute AdcChannelSampTime (AdcChannel) detailed description

Property	Value
Label	Adc Channel Sampling time
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	8
Invalid	Range >=8 <=254

4.9.1.15.12 AdcEnablePresampling (AdcChannel)

When true, this parameter enables the presampling phase for the selected channel. The normal operation sequence on the channel: Presampling -> Sampling -> Conversion.

Table 4-85. Attribute AdcEnablePresampling (AdcChannel) detailed description

Property	Value
Label	Adc Enable Presampling
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.9.1.15.13 AdcEnableThresholds (AdcChannel)

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

Table 4-86. Attribute AdcEnableThresholds (AdcChannel) detailed description

Property	Value
Label	Adc Enable Threshold
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.9.1.15.14 AdcWdogNotification (AdcChannel)

This function pointer is called everytime when the conversion of the channel caused a watchdog interrupt.

Table 4-87. Attribute AdcWdogNotification (AdcChannel) detailed description

Property	Value
Label	Adc Channel Watchdog Notification
Type	FUNCTION-NAME
Origin	Custom
Symbolic Name	false
Default	NULL_PTR

4.9.1.15.15 AdcThresholdRegister (AdcChannel)

Select the threshold register which provides the values to be used for upper and lower thresholds. ADCHwUnits support threshold registers from ADC_THRESHOLD_REG_0 to ADC_THRESHOLD_REG_3. Note: This is an Implementation Specific Parameter.

Table 4-88. Attribute AdcThresholdRegister (AdcChannel) detailed description

Property	Value
Label	Adc Threshold Control Register
Type	REFERENCE
Origin	Custom

4.9.1.16 Form AdcGroup

This container contains the Group configuration (parameters).

Is included by form : Form AdcHwUnit

Included forms :

- Form AdcGroupConversionConfiguration
- Form AdcAlternateGroupConvTimings

- Form AdcGroupDefinition
- Form AdcHwTrig

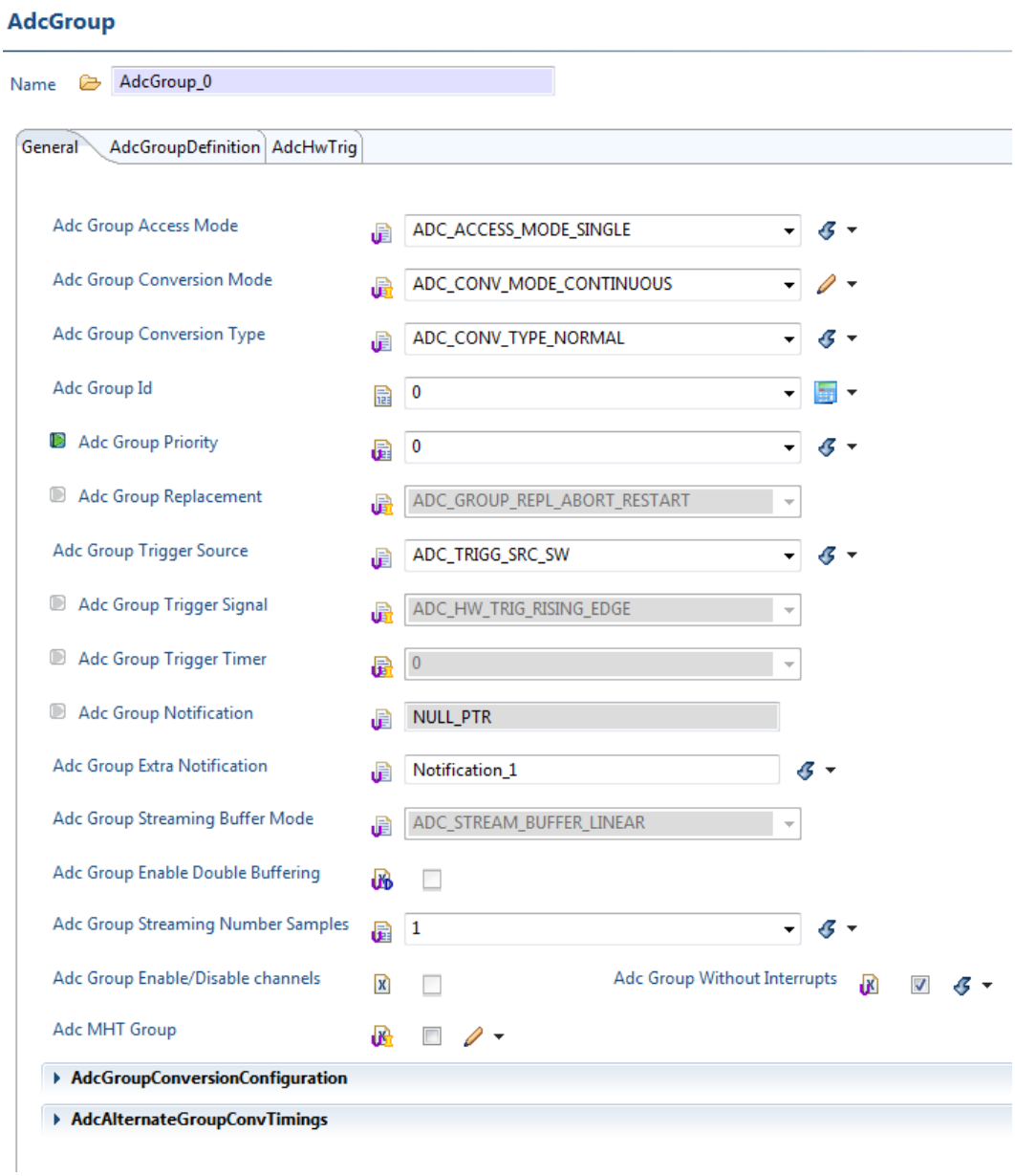

Figure 4-12. Tresos Plugin snapshot for AdcGroup form.

4.9.1.16.1 AdcGroupAccessMode (AdcGroup)

Type of access mode to group conversion results.

Table 4-89. Attribute AdcGroupAccessMode (AdcGroup) detailed description

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

4.9.1.16.2 AdcGroupConversionMode (AdcGroup)

Type of Conversion mode of the channel group.

Table 4-90. Attribute AdcGroupConversionMode (AdcGroup) detailed description

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

4.9.1.16.3 AdcGroupConversionType (AdcGroup)

Normal or Injected conversion type. Note: Calypso hardware does not support normal hardware triggered conversions. Only injected conversion type can be configured for hardware triggered groups.

Table 4-91. Attribute AdcGroupConversionType (AdcGroup) detailed description

Property	Value
Label	Adc Group Conversion Type
Type	ENUMERATION
Origin	Custom

Table continues on the next page...

Table 4-91. Attribute AdcGroupConversionType (AdcGroup) detailed description (continued)

Property	Value
Symbolic Name	false
Default	ADC_CONV_TYPE_NORMAL
Range	ADC_CONV_TYPE_NORMAL ADC_CONV_TYPE_INJECTED

4.9.1.16.4 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 4-92. Attribute AdcGroupId (AdcGroup) detailed description

Property	Value
Label	Adc Group Id
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	true
Invalid	XPath Range <=1023 >=0

4.9.1.16.5 AdcGroupPriority (AdcGroup)

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

Table 4-93. Attribute AdcGroupPriority (AdcGroup) detailed description

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

4.9.1.16.6 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 4-94. Attribute AdcGroupReplacement (AdcGroup) detailed description

Property	Value
Label	Adc Group Replacement
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_GROUP_REPL_ABORT_RESTART
Range	ADC_GROUP_REPL_ABORT_RESTART ADC_GROUP_REPL_SUSPEND_RESUME

4.9.1.16.7 AdcGroupTriggSrc (AdcGroup)

Type of source event that starts a group conversion. It' s possible select Hw or Sw trigger. In case of Hw trigger the trigger source can be from the CTU or External hardware pins of the controller. In this controller only CTU trigger source is supported which is selected by the "AdcHwTrigSrc" parameter.

Table 4-95. Attribute AdcGroupTriggSrc (AdcGroup) detailed description

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

4.9.1.16.8 AdcHwTrigSignal (AdcGroup)

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

Table 4-96. Attribute AdcHwTrigSignal (AdcGroup) detailed description

Property	Value
Label	Adc Group Trigger Signal

Table continues on the next page...

Table 4-96. Attribute AdcHwTrigSignal (AdcGroup) detailed description (continued)

Property	Value
Type	ENUMERATION
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	ADC_HW_TRIG_RISING_EDGE
Range	ADC_HW_TRIG_FALLING_EDGE ADC_HW_TRIG_RISING_EDGE

4.9.1.16.9 AdcHwTrigTimer (AdcGroup)

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

Table 4-97. Attribute AdcHwTrigTimer (AdcGroup) detailed description

Property	Value
Label	Adc Group Trigger Timer
Type	INTEGER
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	0

4.9.1.16.10 AdcNotification (AdcGroup)

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

Table 4-98. Attribute AdcNotification (AdcGroup) detailed description

Property	Value
Label	Adc Group Notification
Type	FUNCTION-NAME
Origin	AUTOSAR_ECUC
Symbolic Name	false
Default	NULL_PTR

4.9.1.16.11 AdcExtraNotification (AdcGroup)

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

Table 4-99. Attribute AdcExtraNotification (AdcGroup) detailed description

Property	Value
Label	Adc Group Extra Notification
Type	FUNCTION-NAME
Origin	Custom
Symbolic Name	false
Default	NULL_PTR

4.9.1.16.12 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 4-100. Attribute AdcStreamingBufferMode (AdcGroup) detailed description

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

4.9.1.16.13 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. This feature supported only for Groups which is configured as hardware trigger

Table 4-101. Attribute AdcEnableDoubleBuffering (AdcGroup) detailed description

Property	Value
Label	Adc Group Enable Double Buffering
Type	BOOLEAN
Origin	Custom

Table continues on the next page...

Table 4-101. Attribute AdcEnableDoubleBuffering (AdcGroup) detailed description (continued)

Property	Value
Symbolic Name	false
Default	false

4.9.1.16.14 AdcStreamingNumSamples (AdcGroup)

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

Table 4-102. 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 <=255 >=1

4.9.1.16.15 AdcEnableChDisableChGroup (AdcGroup)

Note

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

Max.no of Groups with this feature enabled, should be configured are 254 if the configuration parameter AdcEnableChDisableChApi is enabled in NonAutosar container"/> This is an Implementation Specific Parameter.

Table 4-103. Attribute AdcEnableChDisableChGroup (AdcGroup) detailed description

Property	Value
Label	Adc Group Enable/Disable channels
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.9.1.16.16 AdcWithoutInterrupts (AdcGroup)

Enable/ Disable the occuring 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 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. When this parameter is Disabled, normal functionlaity shall be executed. Note: This is an Implementation Specific Parameter.

Table 4-104. Attribute AdcWithoutInterrupts (AdcGroup) detailed description

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

4.9.1.16.17 AdcMultipleHardwareTriggerGroup (AdcGroup)

If is checked that means that this group is part of the MHT groups subset. MHT - Multiple Hardware Trigger. That meaning a subset of groups can be active at a given time as Hardware Triggered groups. To use this feature it is required that every MHT marked group share the same settings on several parameters. The list of parameters that can be different is: - Group Id - actually it should be different. - Group Priority - it can have any value in the domain, it doesnt matter. The priority will be based on the selected CTU trigger priority. - Group Notification - can be different. - Group Buffer Pointer - recommended to be different to know which value from which group/channel comes. - AdcHwTrigSrc - should be different for every group. If two or more MHT groups share the same trigger source it won't work. Each such a grup (MHT) should have only and only one ADC channel! These ADC channel should be unique across these MHT groups (actually they should be unique across the MHT subset runtime!).

Table 4-105. Attribute AdcMultipleHardwareTriggerGroup (AdcGroup) detailed description

Property	Value
Label	Adc MHT Group
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.9.1.16.18 AdcGroupDefinition (AdcGroupDefinition)

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

Table 4-106. Attribute AdcGroupDefinition (AdcGroupDefinition) detailed description

Property	Value
Type	REFERENCE
Origin	AUTOSAR_ECUC

4.9.1.16.19 Form AdcGroupConversionConfiguration

Configure the Sampling and Conversion TimeGroup.

Is included by form : Form AdcGroup

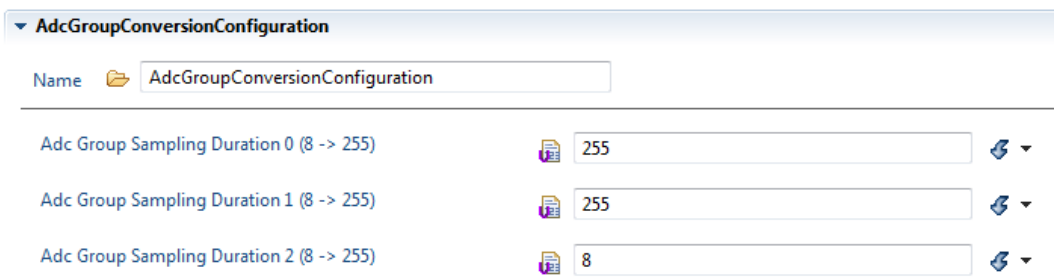

Figure 4-13. Tresos Plugin snapshot for AdcGroupConversionConfiguration form.

4.9.1.16.19.1 AdcSamplingDuration (AdcGroupConversionConfiguration)

Select the Sampling Duration for channels 0-31. It's possibile summarize the Conversion Time using the following formula: Total_conversion_time = ([(ST + CT + DP) * chain_length] + TPT) cycles of AD_clk, where

- Sample phase time (ST): Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing. Registers ADC_CTRx(x= 0...2). The value in the register represents units of cycle of the AD_clk (minum value is 8).
- Compare phase Time (CT): ((n + 1) 4) cycles of AD_clk. n = operating resolution
- Data Processing Time(DP): 2 cycles of AD_clk.
- Trigger Processing time (TPT): 2 clock cycles of bus clock.

Table 4-107. Attribute AdcSamplingDuration (AdcGroupConversionConfiguration) detailed description

Property	Value
Label	Adc Group Sampling Duration 0
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	8
Invalid	Range >=8 <=255

4.9.1.16.19.2 AdcSamplingDuration1 (AdcGroupConversionConfiguration)

Select the Sampling Duration for channels 32-63. It's possibile summarize the Conversion Time using the following formula: Total_conversion_time = ([(ST + CT + DP) * chain_length] + TPT) cycles of AD_clk, where

- Sample phase time (ST): Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing. Registers ADC_CTRx(x= 0...2). The value in the register represents units of cycle of the AD_clk (minum value is 8).
- Compare phase Time (CT): ((n + 1) 4) cycles of AD_clk. n = operating resolution
- Data Processing Time(DP): 2 cycles of AD_clk.
- Trigger Processing time (TPT): 2 clock cycles of bus clock.

Table 4-108. Attribute AdcSamplingDuration1 (AdcGroupConversionConfiguration) detailed description

Property	Value
Label	Adc Group Sampling Duration 1
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	8
Invalid	Range >=8 <=255

4.9.1.16.19.3 AdcSamplingDuration2 (AdcGroupConversionConfiguration)

Select the Sampling Duration for channels 64-95. It's possibile summarize the Conversion Time using the following formula: Total_conversion_time = ([(ST + CT + DP) * chain_length] + TPT) cycles of AD_clk, where

- Sample phase time (ST): Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing. Registers ADC_CTRx(x= 0...2). The value in the register represents units of cycle of the AD_clk (minum value is 8).
- Compare phase Time (CT): ((n + 1) 4) cycles of AD_clk. n = operating resolution
- Data Processing Time(DP): 2 cycles of AD_clk.
- Trigger Processing time (TPT): 2 clock cycles of bus clock.

Table 4-109. Attribute AdcSamplingDuration2 (AdcGroupConversionConfiguration) detailed description

Property	Value
Label	Adc Group Sampling Duration 2
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	8
Invalid	Range >=8 <=255

4.9.1.16.20 Form AdcAlternateGroupConvTimings

Selects Alternate values used for prorgamming CTR Conversion Timing Registers in Adc_SetClockMode API. This is available when AdcEnableDualClockMode has been enabled and AdcConvTimeOnce has been disable

Is included by form : Form AdcGroup

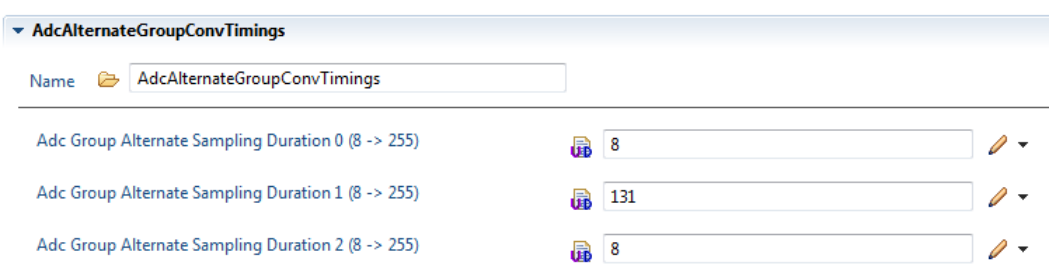

Figure 4-14. Tresos Plugin snapshot for AdcAlternateGroupConvTimings form.

4.9.1.16.20.1 AdcAltGroupSamplingDuration (AdcAlternateGroupConvTimings)

Select the Alternate Sampling Duration for channels 0-31. It's possibile summarize the Conversion Time using the following formula: Total_conversion_time = ([(ST + CT + DP) * chain_length] + TPT) cycles of AD_clk, where

- Sample phase time (ST): Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing Registers ADC_CTRx(x= 0...2). The value in the register represents units of cycle of the AD_clk (minum value is 8).
- Compare phase Time (CT): ((n + 1) 4) cycles of AD_clk. n = operating resolution
- Data Processing Time(DP): 2 cycles of AD_clk.
- Trigger Processing time (TPT): 2 clock cycles of bus clock.

Table 4-110. Attribute AdcAltGroupSamplingDuration (AdcAlternateGroupConvTimings) detailed description

Property	Value
Label	Adc Group Alternate Sampling Duration 0
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	8
Invalid	Range >=8 <=255

4.9.1.16.20.2 AdcAltGroupSamplingDuration1 (AdcAlternateGroupConvTimings)

Select the Alternate Sampling Duration for channels 32-63. It's possibile summarize the Conversion Time using the following formula: Total_conversion_time = ([(ST + CT + DP) * chain_length] + TPT) cycles of AD_clk, where

- Sample phase time (ST): Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing Registers ADC_CTRx(x= 0...2). The value in the register represents units of cycle of the AD_clk (minum value is 8).
- Compare phase Time (CT): ((n + 1) 4) cycles of AD_clk. n = operating resolution
- Data Processing Time(DP): 2 cycles of AD_clk.
- Trigger Processing time (TPT): 2 clock cycles of bus clock.

Table 4-111. Attribute AdcAltGroupSamplingDuration1 (AdcAlternateGroupConvTimings) detailed description

Property	Value
Label	Adc Group Alternate Sampling Duration 1
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	8
Invalid	Range >=8 <=255

4.9.1.16.20.3 AdcAltGroupSamplingDuration2 (AdcAlternateGroupConvTimings)

Select the Alternate Sampling Duration for channels 64-95. It's possibile summarize the Conversion Time using the following formula: Total_conversion_time = ([(ST + CT + DP) * chain_length] + TPT) cycles of AD_clk, where

- Sample phase time (ST): Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing Registers ADC_CTRx(x= 0...2). The value in the register represents units of cycle of the AD_clk (minum value is 8).
- Compare phase Time (CT): ((n + 1) 4) cycles of AD_clk. n = operating resolution
- Data Processing Time(DP): 2 cycles of AD_clk.
- Trigger Processing time (TPT): 2 clock cycles of bus clock.

Table 4-112. Attribute AdcAltGroupSamplingDuration2 (AdcAlternateGroupConvTimings) detailed description

Property	Value
Label	Adc Group Alternate Sampling Duration 2
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	8
Invalid	Range >=8 <=255

4.9.1.16.21 Form AdcGroupDefinition

Is included by form : Form AdcGroup

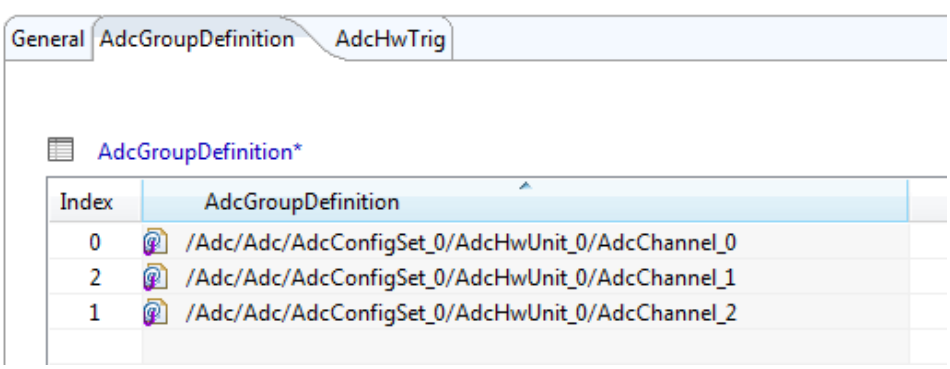

Figure 4-15. Tresos Plugin snapshot for AdcGroupDefinition form.

4.9.1.16.21.1 AdcGroupDefinition (AdcGroupDefinition)

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

Table 4-113. Attribute AdcGroupDefinition (AdcGroupDefinition) detailed description

Property	Value
Type	REFERENCE
Origin	AUTOSAR_ECUC

4.9.1.16.22 Form AdcHwTrig

This container contains the Hardware trigger source configured for the group.

Is included by form : Form AdcGroup

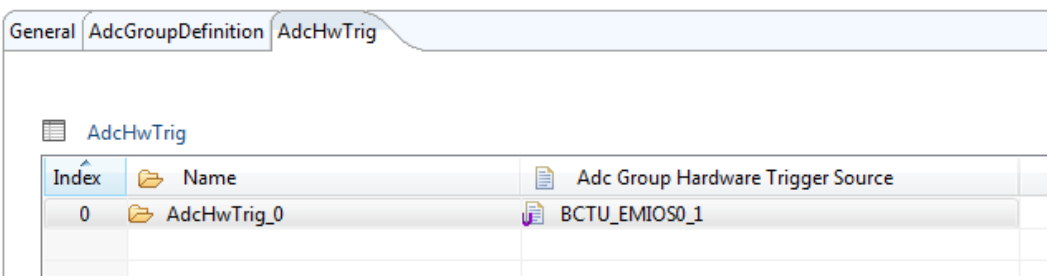

Figure 4-16. Tresos Plugin snapshot for AdcHwTrig form.

4.9.1.16.22.1 AdcHwTrigSrc (AdcHwTrig)

On this implementation there is one possibilities to select hardware triggers: BCTU. (Note: This is an Implementation Specific Parameter.)

Table 4-114. Attribute AdcHwTrigSrc (AdcHwTrig) detailed description

Property	Value
Label	Adc Group Hardware Trigger Source
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	PIT_2

4.9.1.17 Form AdcThresholdControl

Configure threshold detection feature for the selected channel.

Is included by form : Form AdcHwUnit

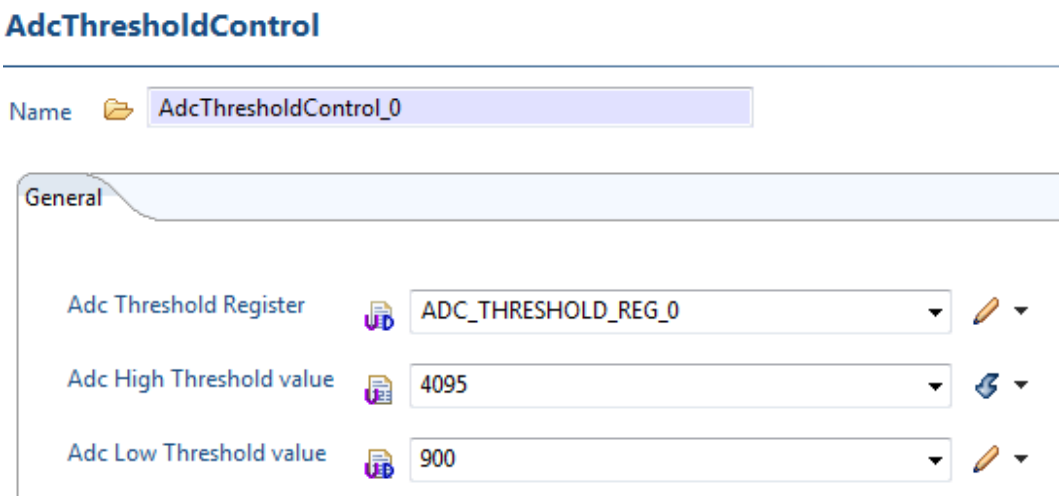

Figure 4-17. Tresos Plugin snapshot for AdcThresholdControl form.

4.9.1.17.1 AdcThresholdControlRegister (AdcThresholdControl)

Select the threshold register which provides the values to be used for upper and lower thresholds. ADCHwUnits support threshold registers from ADC_THRESHOLD_REG_0 to ADC_THRESHOLD_REG_3. Note: This is an Implementation Specific Parameter.

Table 4-115. Attribute AdcThresholdControlRegister (AdcThresholdControl) detailed description

Property	Value
Label	Adc Threshold Register
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	ADC_THRESHOLD_REG_0
Range	ADC_THRESHOLD_REG_0 ADC_THRESHOLD_REG_1 ADC_THRESHOLD_REG_2 ADC_THRESHOLD_REG_3 ADC_THRESHOLD_REG_4 ADC_THRESHOLD_REG_5

4.9.1.17.2 AdcHighThreshold (AdcThresholdControl)

Set the value for High Threshold.

Table 4-116. Attribute AdcHighThreshold (AdcThresholdControl) detailed description

Property	Value
Label	Adc High Threshold value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	4095
Invalid	Range >=0 <=4095

4.9.1.17.3 AdcLowThreshold (AdcThresholdControl)

Set the value for Low Threshold.

Table 4-117. Attribute AdcLowThreshold (AdcThresholdControl) detailed description

Property	Value
Label	Adc Low Threshold value
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	0
Invalid	Range >=0 <=4095

4.9.1.18 Form AdcNormalConvTimings

Selects Normal values used for prorgamming CTR Conversion Timing Registers in Adc_SetClockMode API and also when AdcConvTimeOnce option is enabled. This is available when AdcEnableDualClockMode or AdcConvTimeOnce have been enabled

Is included by form : Form AdcHwUnit

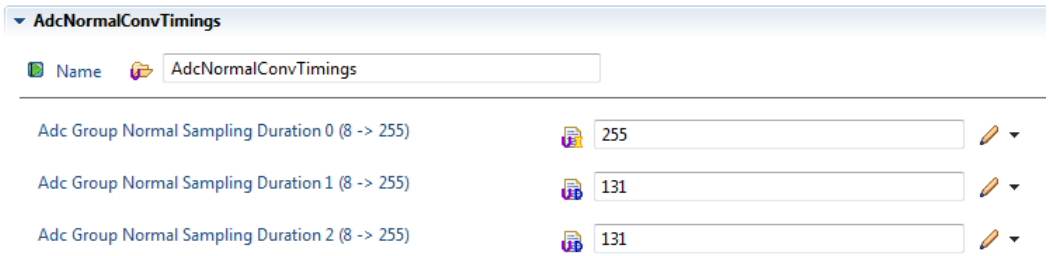

Figure 4-18. Tresos Plugin snapshot for AdcNormalConvTimings form.

4.9.1.18.1 AdcSamplingDurationNormal (AdcNormalConvTimings)

Selects the Normal INPSAMP value of the CTR0 register used when calling Adc_SetClockMode(ADC_NORMAL). The total conversion time in terms of ipg clock can be calculated with following equation for Normal and Injected Conversion-Mode: Total_conversion_time = ([(ST + CT + DP) * chain_length] + TPT) cycles of AD_clk, where

- Sample phase time (ST): Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing. Registers ADC_CTRx(x= 0...2). The value in the register represents units of cycle of the AD_clk (minum value is 8).
- Compare phase Time (CT): ((n + 1) 4) cycles of AD_clk. n = operating resolution
- Data Processing Time(DP): 2 cycles of AD_clk.
- Trigger Processing time (TPT): 2 clock cycles of bus clock.

Table 4-118. Attribute AdcSamplingDurationNormal (AdcNormalConvTimings) detailed description

Property	Value
Label	Adc Group Normal Sampling Duration 0
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	8
Invalid	Range >=8 <=255

4.9.1.18.2 AdcSamplingDurationNormal1 (AdcNormalConvTimings)

Selects the Normal INPSAMP value of the CTR1 register used when calling Adc_SetClockMode(ADC_NORMAL). The total conversion time in terms of ipg clock can be calculated with following equation for Normal and Injected Conversion-Mode: Total_conversion_time = ([(ST + CT + DP) * chain_length] + TPT) cycles of AD_clk, where

- Sample phase time (ST): Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing. Registers ADC_CTRx(x= 0...2). The value in the register represents units of cycle of the AD_clk (minum value is 8).
- Compare phase Time (CT): ((n + 1) 4) cycles of AD_clk. n = operating resolution
- Data Processing Time(DP): 2 cycles of AD_clk.
- Trigger Processing time (TPT): 2 clock cycles of bus clock.

Table 4-119. Attribute AdcSamplingDurationNormal1 (AdcNormalConvTimings) detailed description

Property	Value
Label	Adc Group Normal Sampling Duration 1
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	8
Invalid	Range >=8 <=255

4.9.1.18.3 AdcSamplingDurationNormal2 (AdcNormalConvTimings)

Selects the Normal INPSAMP value of the CTR2 register used when calling Adc_SetClockMode(ADC_NORMAL). The total conversion time in terms of ipg clock can be calculated with following equation for Normal and Injected Conversion-Mode: Total_conversion_time = ([(ST + CT + DP) * chain_length] + TPT) cycles of AD_clk, where

- Sample phase time (ST): Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing. Registers ADC_CTRx(x= 0...2). The value in the register represents units of cycle of the AD_clk (minum value is 8).
- Compare phase Time (CT): ((n + 1) 4) cycles of AD_clk. n = operating resolution
- Data Processing Time(DP): 2 cycles of AD_clk.
- Trigger Processing time (TPT): 2 clock cycles of bus clock.

Table 4-120. Attribute AdcSamplingDurationNormal2 (AdcNormalConvTimings) detailed description

Property	Value
Label	Adc Group Normal Sampling Duration 2
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	8

Table continues on the next page...

Table 4-120. Attribute AdcSamplingDurationNormal2 (AdcNormalConvTimings) detailed description (continued)

Property	Value
Invalid	Range >=8 <=255

4.9.1.19 Form AdcAlternateConvTimings

Selects Alternate values used in Adc_SetClockMode API for prorgamming CTR Conversion Timing Registers. This container is available only when AdcEnableDualClockMode has been enabled

Is included by form : Form AdcHwUnit

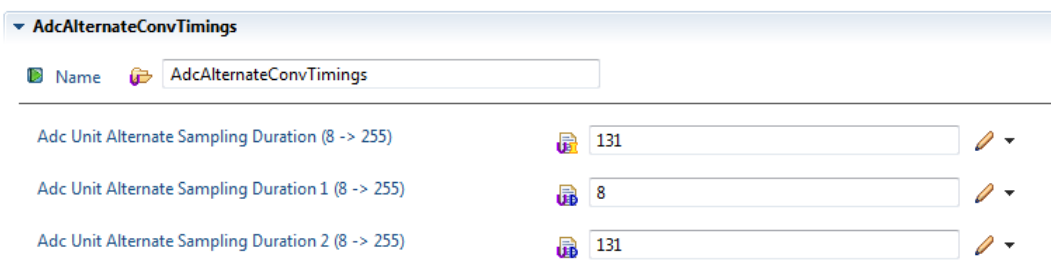

Figure 4-19. Tresos Plugin snapshot for AdcAlternateConvTimings form.

4.9.1.19.1 AdcSamplingDurationAlt (AdcAlternateConvTimings)

Selects the Alternate INPSAMP value of the CTR0 register used when calling Adc_SetClockMode(ADC_ALTERNATE). The total conversion time in terms of ipg clock can be calculated with following equation for Normal and Injected Conversion-Mode: Total_conversion_time = ([(ST + CT + DP) * chain_length] + TPT) cycles of AD_clk, where

- Sample phase time (ST): Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing.Registers ADC_CTRx(x= 0...2). The value in the register represents units of cycle of the AD_clk (minum value is 8).
- Compare phase Time (CT): ((n + 1) 4) cycles of AD_clk. n = operating resolution
- Data Processing Time(DP): 2 cycles of AD_clk.
- Trigger Processing time (TPT): 2 clock cycles of bus clock.

Table 4-121. Attribute AdcSamplingDurationAlt (AdcAlternateConvTimings) detailed description

Property	Value
Label	Adc Unit Alternate Sampling Duration
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	8
Invalid	Range >=8 <=255

4.9.1.19.2 AdcSamplingDurationAlt1 (AdcAlternateConvTimings)

Selects the Alternate INPSAMP value of the CTR1 register used when calling Adc_SetClockMode(ADC_ALTERNATE). The total conversion time in terms of ipg clock can be calculated with following equation for Normal and Injected Conversion-Mode: Total_conversion_time = ([(ST + CT + DP) * chain_length] + TPT) cycles of AD_clk, where

- Sample phase time (ST): Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing. Registers ADC_CTRx(x= 0...2). The value in the register represents units of cycle of the AD_clk (minum value is 8).
- Compare phase Time (CT): ((n + 1) 4) cycles of AD_clk. n = operating resolution
- Data Processing Time(DP): 2 cycles of AD_clk.
- Trigger Processing time (TPT): 2 clock cycles of bus clock.

Table 4-122. Attribute AdcSamplingDurationAlt1 (AdcAlternateConvTimings) detailed description

Property	Value
Label	Adc Unit Alternate Sampling Duration 1
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	8
Invalid	Range >=8 <=255

4.9.1.19.3 AdcSamplingDurationAlt2 (AdcAlternateConvTimings)

Selects the Alternate INPSAMP value of the CTR2 register used when calling Adc_SetClockMode(ADC_ALTERNATE). The total conversion time in terms of ipg clock can be calculated with following equation for Normal and Injected Conversion-Mode: Total_conversion_time = ([(ST + CT + DP) * chain_length] + TPT) cycles of AD_clk, where

- Sample phase time (ST): Sample time duration is controlled by the INPSAMP[7:0] field of Conversion timing. Registers ADC_CTRx(x= 0...2). The value in the register represents units of cycle of the AD_clk (minum value is 8).
- Compare phase Time (CT): ((n + 1) 4) cycles of AD_clk. n = operating resolution
- Data Processing Time(DP): 2 cycles of AD_clk.
- Trigger Processing time (TPT): 2 clock cycles of bus clock.

Table 4-123. Attribute AdcSamplingDurationAlt2 (AdcAlternateConvTimings) detailed description

Property	Value
Label	Adc Unit Alternate Sampling Duration 2
Type	INTEGER
Origin	Custom
Symbolic Name	false
Default	8
Invalid	Range >=8 <=255

4.9.2 Form BCTUHwUnit

This container contains configuration of the BCTU unit.

Is included by form : Form AdcConfigSet

Included forms :

- Form BCTU_InputTrigger

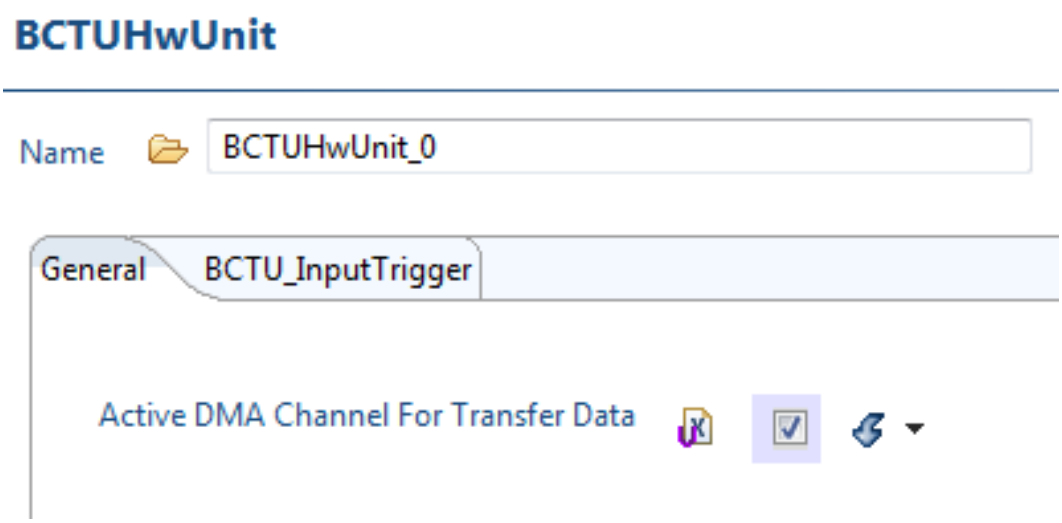

Figure 4-20. Tresos Plugin snapshot for BCTUHwUnit form.

4.9.2.1 BCTUDMAChannelEnable (BCTUHwUnit)

Use DMA channel for transfer conversion results or not.

Table 4-124. Attribute BCTUDMAChannelEnable (BCTUHwUnit) detailed description

Property	Value
Label	Active DMA Channel For Transfer Data
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.9.2.2 Form BCTU_InputTrigger

This container contains the BCTU input trigger configuration parameters as follows:

InputTrigger 0 - eMIOS_0 Channel_0

InputTrigger 1 - eMIOS_0 Channel_1

InputTrigger 2 - eMIOS_0 Channel_2

InputTrigger 3 - eMIOS_0 Channel_3

InputTrigger 4 - eMIOS_0 Channel_4

InputTrigger 5 - eMIOS_0 Channel_5

InputTrigger 6 - eMIOS_0 Channel_6

InputTrigger 7 - eMIOS_0 Channel_7

InputTrigger 8 - eMIOS_0 Channel_8

InputTrigger 9 - eMIOS_0 Channel_9

InputTrigger 10 - eMIOS_0 Channel_10

InputTrigger 11 - eMIOS_0 Channel_11

InputTrigger 12 - eMIOS_0 Channel_12

InputTrigger 13 - eMIOS_0 Channel_13

InputTrigger 14 - eMIOS_0 Channel_14

InputTrigger 15 - eMIOS_0 Channel_15

InputTrigger 16 - eMIOS_0 Channel_16

InputTrigger 17 - eMIOS_0 Channel_17

InputTrigger 18 - eMIOS_0 Channel_18

InputTrigger 19 - eMIOS_0 Channel_19

InputTrigger 20 - eMIOS_0 Channel_20

InputTrigger 21 - eMIOS_0 Channel_21

InputTrigger 22 - eMIOS_0 Channel_22

InputTrigger 23 - PIT_RTI PIT_3

InputTrigger 24 - eMIOS_0 Channel_24

InputTrigger 25 - eMIOS_0 Channel_25

InputTrigger 26 - eMIOS_0 Channel_26

InputTrigger 27 - eMIOS_0 Channel_27

InputTrigger 28 - eMIOS_0 Channel_28

InputTrigger 29 - eMIOS_0 Channel_29

InputTrigger 30 - eMIOS_0 Channel_30

InputTrigger 31 - eMIOS_0 Channel_31

InputTrigger 32 - eMIOS_1 Channel_0

InputTrigger 33 - eMIOS_1 Channel_1

InputTrigger 34 - eMIOS_1 Channel_2

InputTrigger 35 - eMIOS_1 Channel_3

InputTrigger 36 - eMIOS_1 Channel_4

InputTrigger 37 - eMIOS_1 Channel_5

InputTrigger 38 - eMIOS_1 Channel_6

InputTrigger 39 - eMIOS_1 Channel_7

InputTrigger 40 - eMIOS_1 Channel_8

InputTrigger 41 - eMIOS_1 Channel_9

InputTrigger 42 - eMIOS_1 Channel_10

InputTrigger 43 - eMIOS_1 Channel_11

InputTrigger 44 - eMIOS_1 Channel_12

InputTrigger 45 - eMIOS_1 Channel_13

InputTrigger 46 - eMIOS_1 Channel_14

InputTrigger 47 - eMIOS_1 Channel_15

InputTrigger 48 - eMIOS_1 Channel_16

InputTrigger 49 - eMIOS_1 Channel_17

InputTrigger 50 - eMIOS_1 Channel_18

InputTrigger 51 - eMIOS_1 Channel_19

InputTrigger 52 - eMIOS_1 Channel_20

InputTrigger 53 - eMIOS_1 Channel_21

InputTrigger 54 - eMIOS_1 Channel_22

InputTrigger 55 - PIT_RTI PIT_7

InputTrigger 56 - eMIOS_1 Channel_24

InputTrigger 57 - eMIOS_1 Channel_25

InputTrigger 58 - eMIOS_1 Channel_26

InputTrigger 59 - eMIOS_1 Channel_27

InputTrigger 60 - eMIOS_1 Channel_28

InputTrigger 61 - eMIOS_1 Channel_29

InputTrigger 62 - eMIOS_1 Channel_30

InputTrigger 63 - eMIOS_1 Channel_31

InputTrigger 64 - eMIOS_2 Channel_0

InputTrigger 65 - eMIOS_2 Channel_1

InputTrigger 66 - eMIOS_2 Channel_2

InputTrigger 67 - eMIOS_2 Channel_3

InputTrigger 68 - eMIOS_2 Channel_4

InputTrigger 69 - eMIOS_2 Channel_5

InputTrigger 70 - eMIOS_2 Channel_6

InputTrigger 71 - eMIOS_2 Channel_7

InputTrigger 72 - eMIOS_2 Channel_8

InputTrigger 73 - eMIOS_2 Channel_9

InputTrigger 74 - eMIOS_2 Channel_10

InputTrigger 75 - eMIOS_2 Channel_11

InputTrigger 76 - eMIOS_2 Channel_12

InputTrigger 77 - eMIOS_2 Channel_13

InputTrigger 78 - eMIOS_2 Channel_14

InputTrigger 79 - eMIOS_2 Channel_15

InputTrigger 80 - eMIOS_2 Channel_16

InputTrigger 81 - eMIOS_2 Channel_17

InputTrigger 82 - eMIOS_2 Channel_18

InputTrigger 83 - eMIOS_2 Channel_19

InputTrigger 84 - eMIOS_2 Channel_20

InputTrigger 85 - eMIOS_2 Channel_21

InputTrigger 86 - eMIOS_2 Channel_22

InputTrigger 87 - PIT_RTI PIT_8

InputTrigger 88 - eMIOS_2 Channel_24Channel_24

InputTrigger 89 - eMIOS_2 Channel_25

InputTrigger 90 - eMIOS_2 Channel_26

InputTrigger 91 - eMIOS_2 Channel_27

InputTrigger 92 - eMIOS_2 Channel_28

InputTrigger 93 - eMIOS_2 Channel_29

InputTrigger 94 - eMIOS_2 Channel_30

InputTrigger 95 - eMIOS_2 Channel_31

Is included by form : Form BCTUHwUnit

Included forms :

- Form AdcChannelTriggered

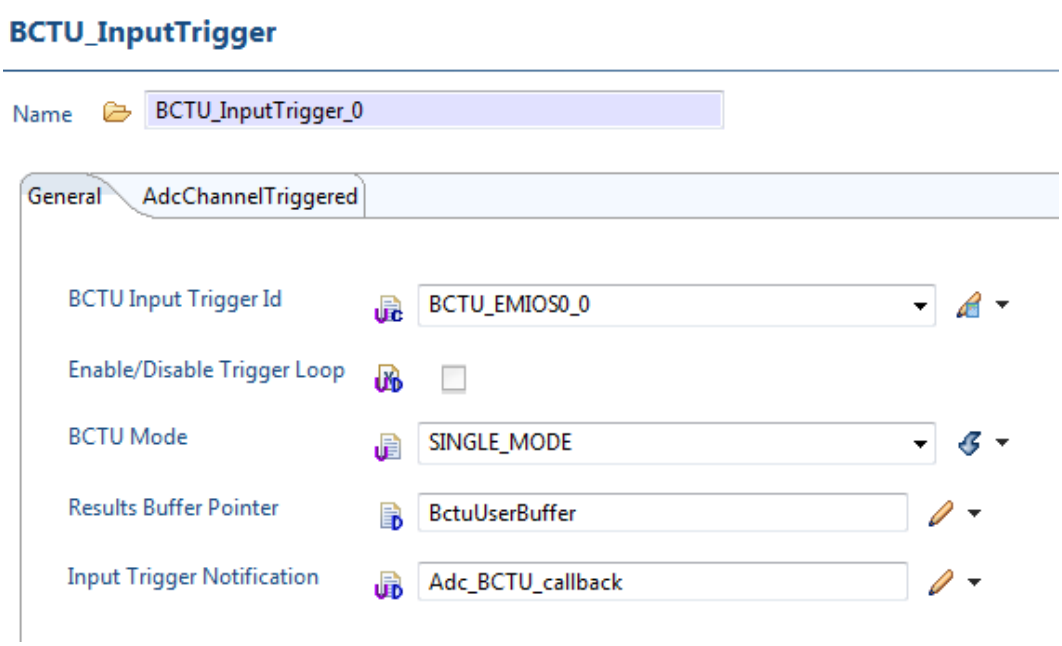

Figure 4-21. Tresos Plugin snapshot for BCTU_InputTrigger form.

4.9.2.2.1 BCTUInputTriggerID (BCTU_InputTrigger)

This parameter defines the assignment of the input trigger source to trigger BCTU

Table 4-125. Attribute BCTUInputTriggerID (BCTU_InputTrigger) detailed description

Property	Value
Label	BCTU Input Trigger Id
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

4.9.2.2.2 BCTUTriggerLoop (BCTU_InputTrigger)

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

Table 4-126. Attribute BCTUTriggerLoop (BCTU_InputTrigger) detailed description

Property	Value
Label	Enable/Disable Trigger Loop
Type	BOOLEAN
Origin	Custom
Symbolic Name	false
Default	false

4.9.2.2.3 BCTUMode (BCTU_InputTrigger)

Select list mode or single mode for conversion. The multiple ADC selection is only valid for multiple parallel conversions LIST functionality. If more than one ADC is selected for a single conversion, instead of for a LIST, no conversion will be triggered by this register.

Table 4-127. Attribute BCTUMode (BCTU_InputTrigger) detailed description

Property	Value
Label	BCTU Mode
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	SINGLE_MODE
Range	SINGLE_MODE LIST_MODE

4.9.2.2.4 BCTUUserBuffer (BCTU_InputTrigger)

Pointer to the Data Buffer (destination for conversion results).

Table 4-128. Attribute BCTUUserBuffer (BCTU_InputTrigger) detailed description

Property	Value
Label	Results Buffer Pointer
Type	LINKER-SYMBOL
Origin	Custom
Symbolic Name	false
Default	BctuUserBuffer

4.9.2.2.5 BCTUUserCallback (BCTU_InputTrigger)

This function pointer is called everytime when the data available for single mode and last channel ending for LIST mode.

Table 4-129. Attribute BCTUUserCallback (BCTU_InputTrigger) detailed description

Property	Value
Label	Input Trigger Notification
Type	FUNCTION-NAME
Origin	Custom
Symbolic Name	false
Default	NULL_PTR

4.9.2.3 Form AdcChannelTriggered

Note

This container contains Adc channels triggered by the Input trigger.

If List mode is enabled and more than one Adc hardware unit are selected (Multiple Parallel Conversions), If n ADC are selected the LIST is interpreted in groups of n elements, being the order from 0 through n.

Is included by form : Form BCTU_InputTrigger

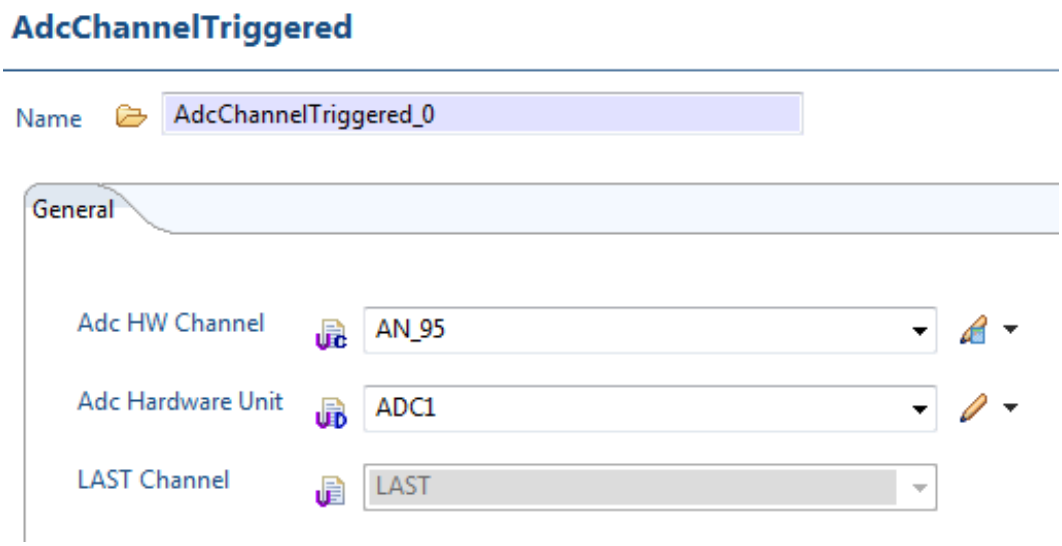

Figure 4-22. Tresos Plugin snapshot for AdcChannelTriggered form.

4.9.2.3.1 AdcChannel (AdcChannelTriggered)

Selects the physical Hardware Adc Channel. Note: Range of the ADC Channels depends on the selected package.

Table 4-130. Attribute AdcChannel (AdcChannelTriggered) detailed description

Property	Value
Label	Adc HW Channel
Type	ENUMERATION
Origin	Custom
Symbolic Name	false

4.9.2.3.2 ADCHWUNIT (AdcChannelTriggered)

Specifies the used ADC Hardware Unit.

Table 4-131. Attribute ADCHWUNIT (AdcChannelTriggered) detailed description

Property	Value
Label	Adc Hardware Unit
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	ADC0
Range	ADC0 ADC1

4.9.2.3.3 COMMAND (AdcChannelTriggered)

Indicate that this command is the last/not last in the sequence, if last channel, ADC will be stop.

Table 4-132. Attribute COMMAND (AdcChannelTriggered) detailed description

Property	Value
Label	LAST Channel
Type	ENUMERATION
Origin	Custom
Symbolic Name	false
Default	LAST
Range	NOT_LAST LAST

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number UM35ADCASR4.2 Rev002R1.0.0
Revision 1.0

AUTOSAR