MCAL User Manual for Adc

32-bit TriCore™ AURIX™ TC3xx microcontroller

About this document

Scope and purpose

This User Manual is intended to enable users to integrate the Microcontroller Abstraction Layer (MCAL) software for the TriCore™ AURIX™ family of 32-bit microcontrollers.

This document describes responsibilities of integrator in-charge of integrating MCAL software with the basic software (BSW) stack. This document also provides detailed information on safety, configuration and functions along with examples of usage of significant features.

Note: Detailed information about package installation, safety and other generic information that are common across all modules are provided in MCAL User Manual General.

Intended audience

This document is intended for anyone using the Adc module of the TC3xx MCAL software.

Document conventions

Table 1 Conventions

Convention	Explanation
Bold	Emphasizes heading levels, column headings, table and figure captions, screen names, windows, dialog boxes, menus, sub-menus
Italics	Denotes variable(s) and reference(s)
`Courier New`	Denotes APIs, functions, interrupt handlers, events, data types, error handlers, file/folder names, directories, command line inputs, code snippets
>	Indicates that a cascading sub-menu opens when you select a menu item
[cover parentID=<alpha numeric value>]	Used for traceability completeness. Reader should ignore these.

Reference documents

This User Manual should be read in conjunction with the following documents:

- AURIX™ TC3xx MCAL User Manual General
- Specification of ADC Driver, AUTOSAR_SWS_ADC_Driver, AUTOSAR Release 4.2.2
- Specification of ADC Driver, AUTOSAR_SWS_ADC_Driver, AUTOSAR Release 4.4.0

Table of contents

Table of contents

Table of contents

Table of contents

Table of contents

Table of contents

Table of contents

1 Adc driver

1.1 User information

1.1.1 Description

The ADC driver is responsible for providing standard analog-to-digital conversion services specified by AUTOSAR. The ADC driver provides user interface options to configure the driver parameters as described in the AUTOSAR ADC specification. It also provides additional parameters to configure various functional blocks of EVADC. The EVADC module is the underlying analog converter. The APIs provided by the driver are multicore capable, that is, they may be invoked from several cores simultaneously. The ADC driver supports only the Primary and Secondary clusters of the EVADC, however, the fast-compare channels are not supported.

1.1.2 Hardware-software mapping

This section describes the system view of the ADC driver and peripherals administered by it.

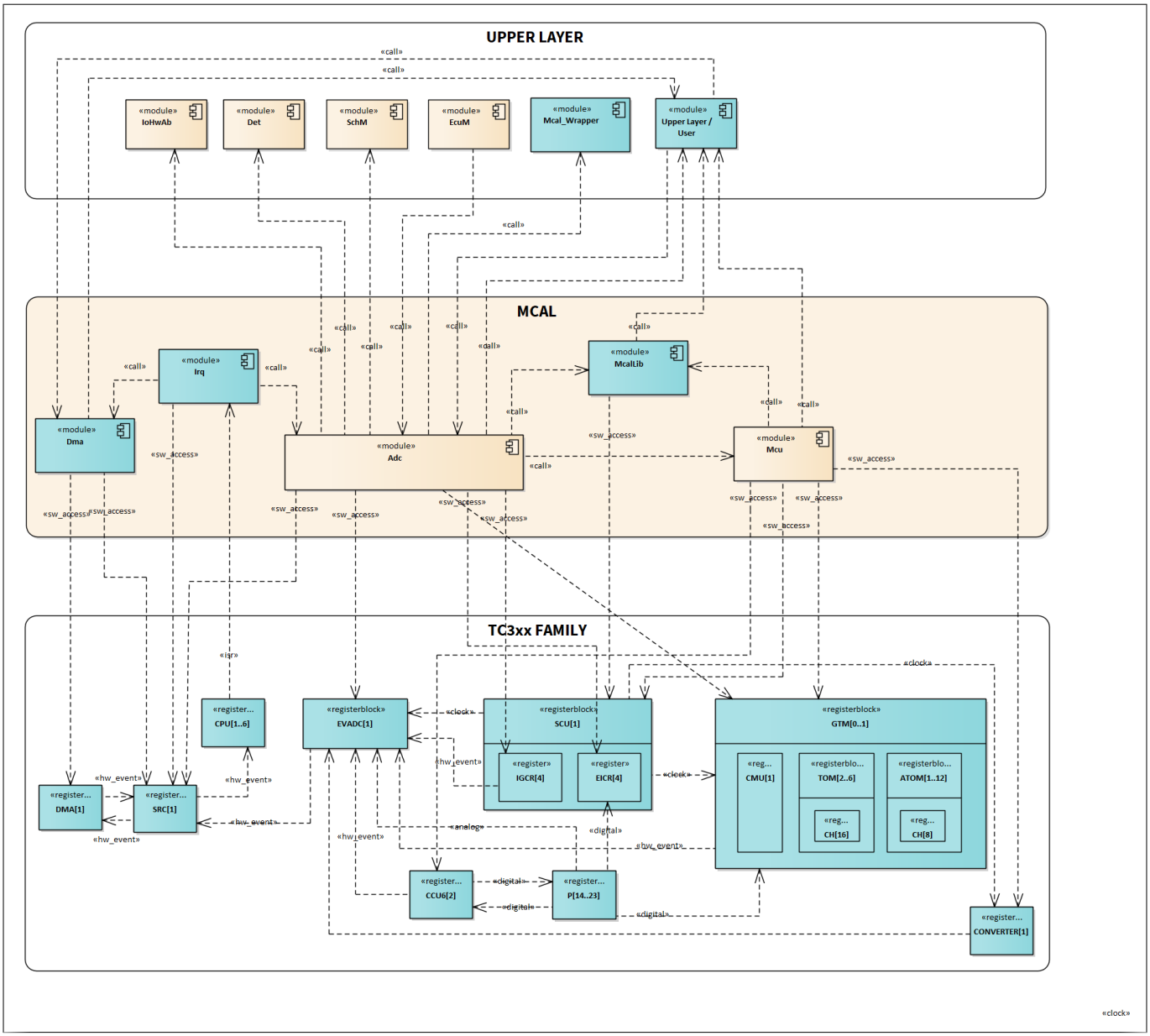

Figure 1 Mapping of hardware-software interfaces

1.1.2.1 EVADC: primary hardware peripheral

Hardware functional features

The ADC driver uses the EVADC IP for converting the analog signals to 12-bit digital values.

The key hardware functional features used by the driver are:

- Software-based trigger
- Timer- and event-based triggers
- Limit checking
- Priority mechanism of the arbiter
- Power saving modes
- Start-up calibration

- Synchronous conversion across hardware groups
- External multiplexer

The unsupported features of the EVADC IP are:

- Wait-for-Read mode
- Conversion result modification modes (Standard Data Reduction mode, Filtering mode and Difference mode)
- Fast compare channels
- FIFO mode of result registers

Users of the hardware

The ADC driver exclusively utilizes the EVADC IP.

Hardware diagnostic features

The supported diagnostic features of the EVADC IP are:

- Broken wire detection
- On-chip supervision signal (through the alias feature)
- Pull-down diagnostics
- Multiplexer diagnostics
- Converter diagnostics (through the alias feature)

Hardware events

The ADC driver uses the following hardware events from the EVADC IP:

- Result event: to trigger transfer of conversion results through DMA IP when the DMA mode of result handling is configured
- Request source event: to trigger transfer of conversion results and update group status when the interrupt mode of result handling is configured
- Channel event: for limit checking enabled AdcChannel groups

1.1.2.2 CCU6: dependent hardware peripheral

Hardware functional features

The ADC driver depends on the CCU6 events to trigger the conversion of an AdcChannel group. The user software may require an AdcChannel group to convert CCU6 events. In such a case, the user software must configure the CCU6 channel directly to generate events. The EVADC trigger line must be configured to the CCU6 event for the ADC channel group through configuration parameters.

Users of the hardware

The T12 or T13 channel of the CCU6 is exclusively used by the CCU6 user. While users of the T12 or T13 channels are many, a channel is allocated to and used by exactly one driver.

Hardware diagnostic features

The SMU alarms configured for the CCU6 are not monitored by the ADC driver.

Hardware events

The ADC driver uses the event generated by the timer channel. The event is used as a trigger for converting the analog channels.

1.1.2.3 CONVERTER: primary hardware peripheral

Hardware functional features

The ADC driver depends on the CONVERTER IP for providing the clock synchronization signal for the EVADC IP. The clock synchronization signal synchronizes the analog clocks of all the EVADC hardware groups.

Users of the hardware

The synchronization signal frequency is used by the ADC and DSADC drivers, however the configuration for generating the signals is done by the MCU driver.

Hardware diagnostic features

The SMU alarms configured for the CONVERTER IP are not monitored by the ADC driver.

Hardware events

Not applicable.

1.1.2.4 DMA: dependent hardware peripheral.

Hardware functional features

The ADC driver depends on the DMA IP for transferring the conversion results in the DMA mode of result handling.

Users of the hardware

The DMA channels are exclusively used by the DMA user. The DMA channels used for the ADC must be reserved and configured by the application through interfaces provided by the DMA.

Hardware diagnostic features

- Move engine error enabled during data transmission
- SMU alarms configured for the DMA are not monitored by the ADC driver

Hardware events

Hardware events from the DMA are not used by the ADC driver.

1.1.2.5 EICR / IGCR: primary hardware peripheral

Hardware functional features

The ADC driver depends on the ERU IP for realizing the hardware-event-based triggers and gating features. The driver uses the trigger event derived from the event trigger logic block of the ERU IP.

The unsupported features of the ERU IP are:

- Pattern detection
- Generation of interrupts based on the trigger events

Users of the hardware

The ERU IP is used by the ADC, DSADC and ICU drivers. The EICR and IGCR channels used by each driver are reserved through the configuration interfaces of the MCU driver. The channel specific SFRs are programmed by the driver. Since multiple channels share common SFRs and to avoid corruption of data for other channels, the driver programs these SFRs atomically with a channel-specific mask. Glitch filter configuration for digital ports is done by the MCU driver.

Hardware diagnostic features

The SMU alarms configured for the ERU IP are not monitored by the ADC driver.

Hardware events

The ADC driver uses the following hardware events from the ERU IP:

- Rising edge event
- Falling edge event

1.1.2.6 GTM: dependent hardware peripheral

Hardware functional features

The ADC driver depends on the GTM IP for realizing the time-based trigger and gating features. The driver uses the compare-match event and the channel output signal for starting and stopping the conversions of an ADC channel group.

Users of the hardware

The GTM IP is used by the ADC, PWM, OCU, WDG, GPT and ICU drivers. The GTM resources used by each driver are reserved through the configuration interface of MCU driver to avoid resource conflict. The ADC driver uses the TOM or ATOM channels for generating the trigger and gating signals. The driver invokes the APIs of the MCU driver to configure all the SFRs related to TOM or ATOM for generating the trigger and gating signals. In case an external PWM signal is used for triggering or gating an AdcChannel group then the user must configure the GTM channels.

Hardware diagnostic features

The SMU alarms configured for the GTM IP are not monitored by the ADC driver.

Hardware events

The ADC driver uses the following hardware events from the GTM:

- Compare-match event: to start the conversion
- Channel output level: to start/stop the conversion based on the timer output

1.1.2.7 PORT: dependent hardware peripheral

Hardware functional features

The analog signals are routed to the EVADC converter through the analog port pads. The external trigger events for the converter are routed through the digital port pad. These are configured and enabled through the PORT driver.

Users of the hardware

The port pads are configured by the PORT driver.

Hardware diagnostic features

Not applicable.

Hardware events

Hardware events from port pads are not used by the ADC driver.

1.1.2.8 SCU: dependent hardware peripheral.

Hardware functional features

The ADC driver depends on the SCU IP for the clock, ENDINIT and reset functionality. The driver requires the fSPB and fADC clock signals for functioning.

Users of the hardware

The SCU IP supplies the clock for all the peripherals and the MCU driver is responsible for configuring the clock tree. To avoid conflicts due to simultaneous writes, update to all the ENDINIT protected registers is performed using the MCALLIB APIs.

Hardware diagnostic features

The SMU alarms configured for the SCU IP are not monitored by the ADC driver.

Hardware events

Hardware events from the SCU IP are not used by the ADC driver.

1.1.2.9 SRC: dependent hardware peripheral

Hardware functional features

The ADC driver depends on the interrupt router for raising an interrupt to the CPU based on the request source event or the channel event, which indicates the end of conversion of a sequence of channels. The ADC driver copies the conversion results from the result registers to the application buffers in the interrupt service routine of these events.

Users of the hardware

The interrupt router is configured either by the IRQ driver or the application software. The ADC driver clears the pending interrupt requests in the SRC register.

Hardware diagnostic features

The SMU alarms configured for the interrupt router are not monitored by the ADC driver.

Hardware events

The interrupt events raised by the interrupt router are serviced by the CPU and the DMA. The ADC driver provides interrupt handlers as software interfaces, which must be invoked from the ISR.

1.1.3 File structure

1.1.3.1 C file structure

This section provides details of the C files of the ADC driver.

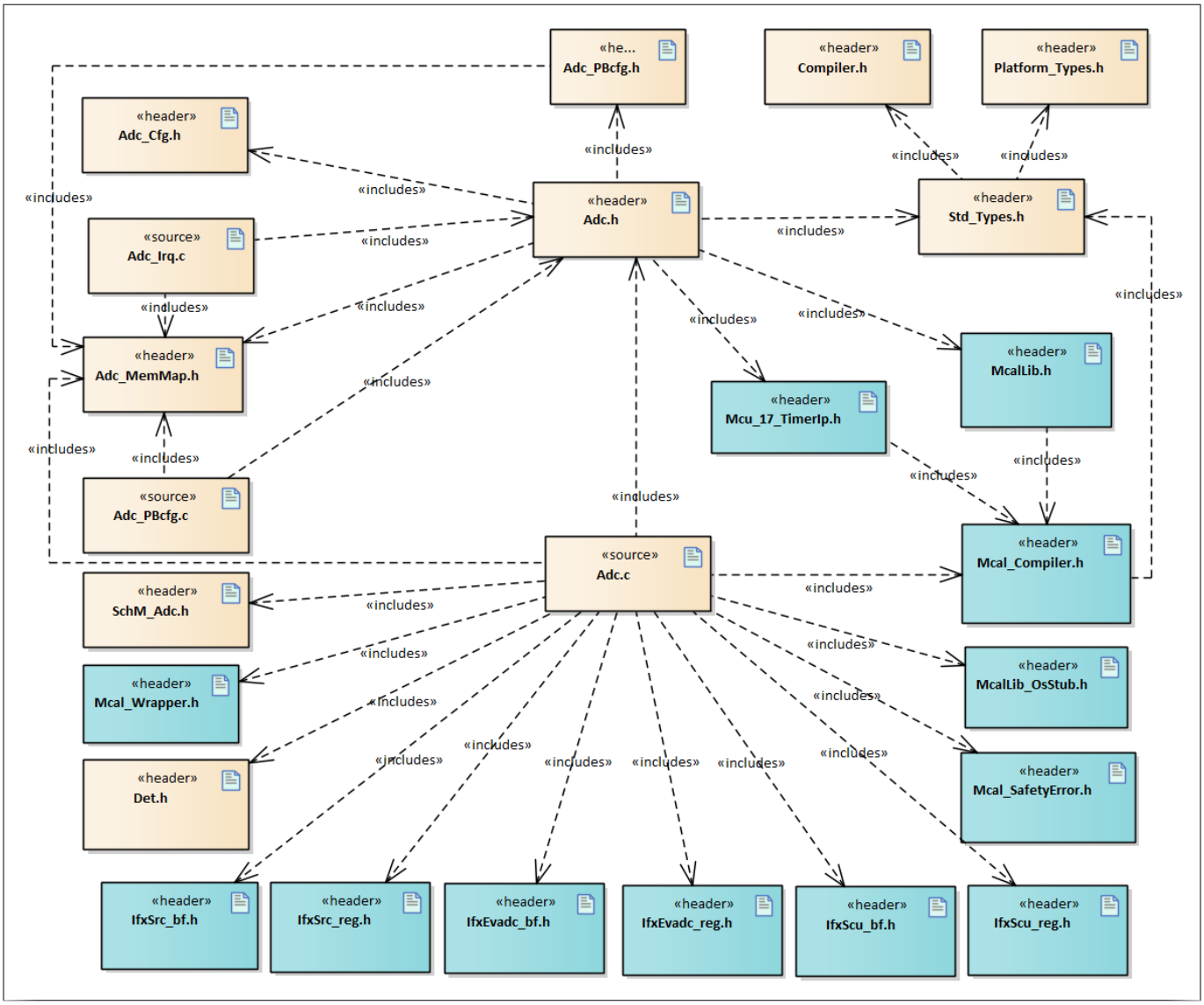

Figure 2 Adc_C_File_Structure-1.png

Table 2 C file structure

File name	Description
`Adc.c`	File (Static) containing implementation of APIs for the ADC driver
`Adc.h`	Header file (Static) defining prototypes of data structures, APIs and interrupt handlers for the ADC driver
`Adc_Cfg.h`	Header file (Generated) containing constants and pre-processor macros as #defines for the ADC driver
`Adc_Irq.c`	Interrupt handler file for the ADC driver
`Adc_MemMap.h`	File (Static) containing the memory section definitions used by the ADC driver
`Adc_PBcfg.c`	File (Generated) containing definition of the configuration data structures for the ADC driver
`Adc_PBcfg.h`	File (Generated) containing declaration of the post-build configuration data structures for the ADC driver

(table continues...)

Table 2 (continued) C file structure

File name	Description
Compiler.h	Provides abstraction from compiler-specific keywords
Det.h	Provides the exported interfaces of Development Error Tracer
IfxEvadc_bf.h	SFR header file for EVADC
IfxEvadc_reg.h	SFR header file for EVADC
IfxScu_bf.h	SFR header file for SCU
IfxScu_reg.h	SFR header file for SCU
IfxSrc_bf.h	SFR header file for Interrupt Controller
IfxSrc_reg.h	SFR header file for Interrupt Controller
McalLib.h	Static header file defining prototypes of data structure and APIs exported by the MCALLIB.
McalLib_OsStub.h	McalLib_OsStub.h provides macros to support user mode of Tricore. This shall be included by other drivers to call OS APIs.
Mcal_Compiler.h	Header file providing abstraction for TriCore™-intrinsic instruction.
Mcal_SafetyError.h	Header file containing the prototype of the API for reporting safety-related errors
Mcal_Wrapper.h	Provides the exported interfaces for Production Error and Runtime Development Errors. Implemented by default to include functions of Dem.h and Det.h files. This file can be modified by the user but function prototype is not user modifiable.
Mcu_17_TimerIp.h	Header file defining prototypes of data structures and APIs of Timer IPs (GTM, CCU6 and GPT12), containing functions such as initialization, enable, interrupt handlers and other services and is included by Mcu_17_TimerIp.c source file
Platform_Types.h	Platform-specific type declaration file as defined by AUTOSAR
SchM_Adc.h	Export Header for SchM functions of the ADC driver
Std_Types.h	Standard type declaration file as defined by AUTOSAR. It is independent of compiler or platform.

1.1.3.2 Code generator plugin files

This section provides details of the code generator plugin files of the ADC driver.

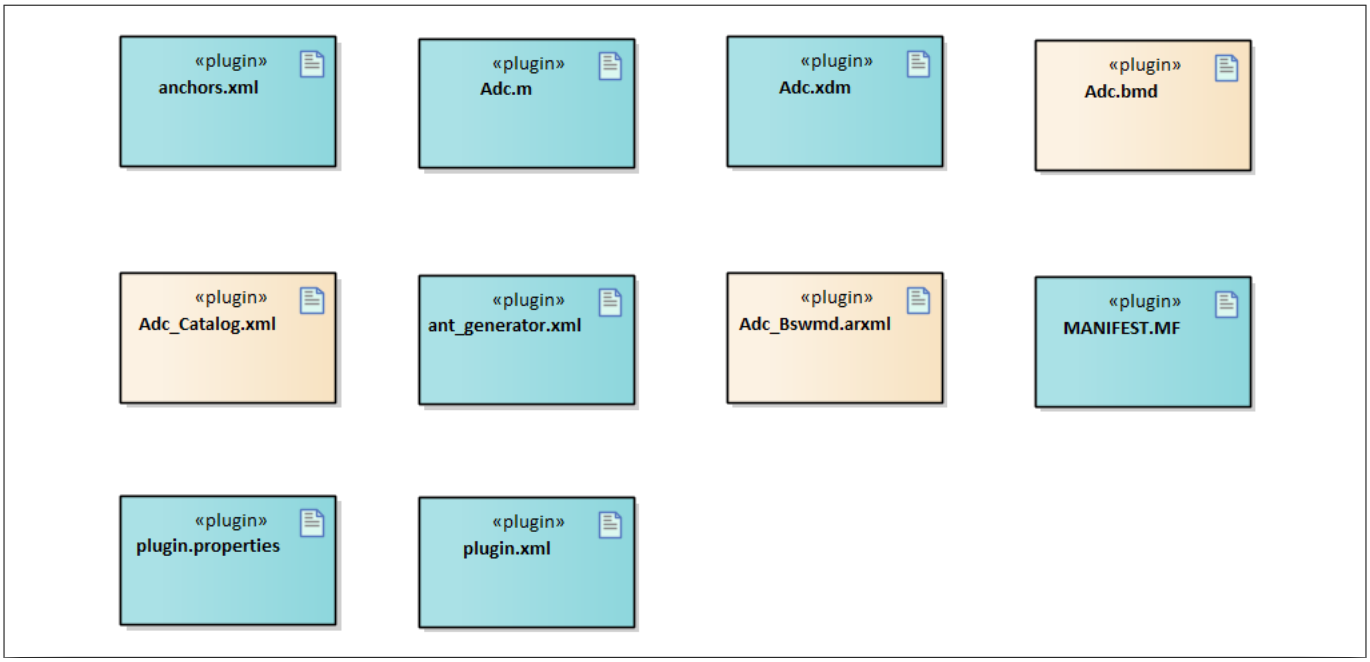

Figure 3 **Adc_Code_Generator_Plugin_Files-1.png**

Table 3 **Code generator plugin files**

File name	Description
`Adc.bmd`	AUTOSAR format XML data model schema file for the ADC driver
`Adc.m`	Code template macro file for the ADC driver
`Adc.xdm`	Tresos format XML data model schema file for the ADC driver
`Adc_Bswmd.arxml`	AUTOSAR format module description file for the ADC driver
`Adc_Catalog.xml`	AUTOSAR format catalog file as per catalog_V3_0_0.ml.xsd for the ADC driver
`MANIFEST.MF`	Tresos plugin support file containing the metadata for the ADC driver
`anchors.xml`	Tresos anchors support file for the ADC driver
`ant_generator.xml`	Tresos support file to generate and rename multiple post-build configurations when using variation point
`plugin.properties`	Tresos plugin support file for the ADC driver
`plugin.xml`	Tresos plugin support file for the ADC driver

1.1.4 Integration hints

This section lists the key points that an integrator or user of the ADC driver must consider.

1.1.4.1 Integration with AUTOSAR stack

This section lists the modules, which are not part of the MCAL, but are required to integrate the ADC driver.

- **EcuM**

  The ECU Manager module is a part of the AUTOSAR stack that manages common aspects of the ECU. Specifically, in the context of the MCAL, the EcuM is used for initialization and de-initialization of the

software drivers. The EcuM module provided in the MCAL package is a stub code and needs to be replaced with a complete EcuM module during the integration phase.

Initialization of ADC:

The initialization of the ADC driver should be invoked from each CPU core, which intends to use the services of the ADC driver. Initialization from the logical master core must be completed prior to invoking the initialization from the slave cores. All the slave cores can execute initialization in parallel. In case all the ADC resources are allocated to the master core, invoking the initialization from the master core alone is sufficient.

EVADC also requires a start-up calibration post-initialization, which must be invoked after initialization on cores is completed. A sample initialization sequence with start-up calibration is depicted in the following figure.

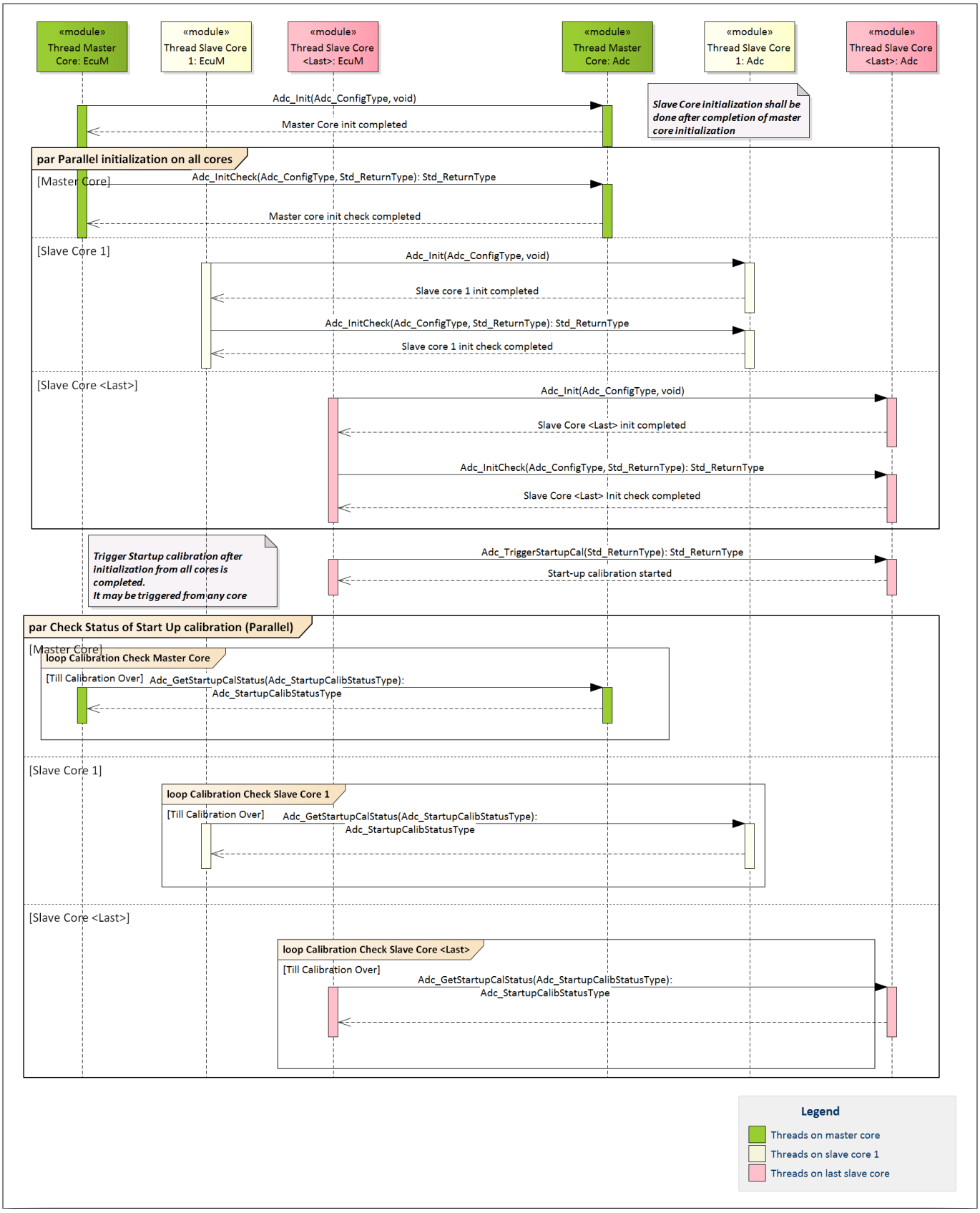

Figure 4 Sample initialization of ADC driver with start-up calibration

De-initialization of ADC:

The de-initialization of the ADC driver should be invoked from each CPU core that used the services of the ADC driver. De-initialization from all the slave cores must be completed prior to invoking the de-initialization from the master core. The slave cores can execute de-initialization in parallel. In case all the ADC resources are allocated to the master core, invoking the de-initialization from the master core alone is sufficient. A sample de-initialization sequence is depicted in the following figure.

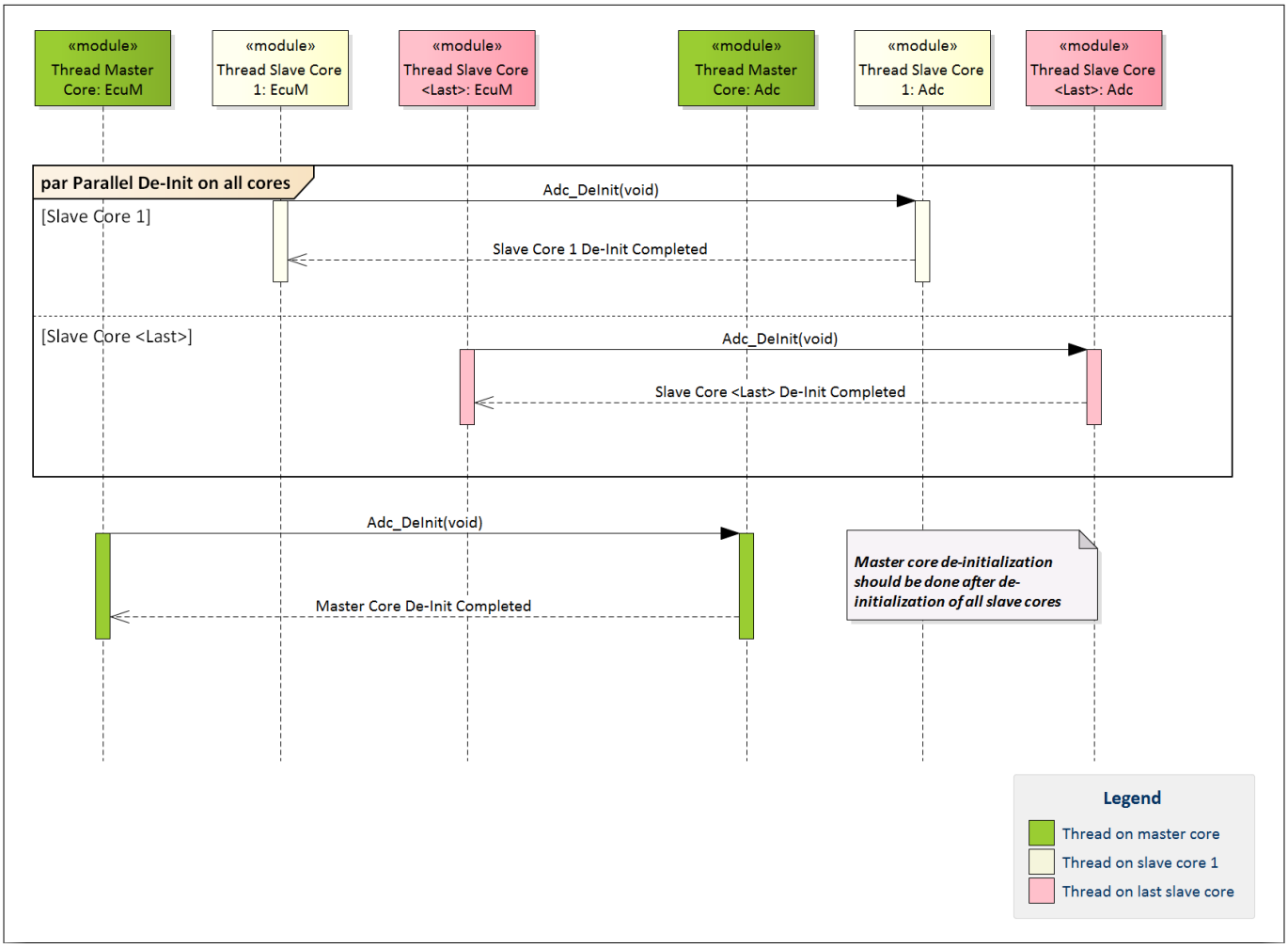

Figure 5 Sample de-initialization of ADC driver

- **Memory mapping**

  Memory mapping is a concept from AUTOSAR that allows relocation of text, variables, constants and configuration data to user-specific memory regions. To achieve this, all the relocatable elements of the driver are encapsulated in different memory-section macros. These macros are defined in the `Adc_MemMap.h` file.

  The `Adc_MemMap.h` file is provided in the MCAL package as a stub code. The integrator must place the appropriate compiler pragmas within the memory-section macros. The pragmas ensure that the elements

are relocated to the correct memory region. A sample implementation listing the memory-section macros is as follows.

```
/**** GLOBAL RAM DATA -- NON-CACHED LMU ****/
if defined ADC_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
 /*****User pragmas here for Non-cached LMU*****/
 #undef ADC_START_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined ADC_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
 #ifdef _TASKING_C_TRICORE_
 /*****User pragmas here for Non-cached LMU*****/
 #undef ADC_STOP_SEC_VAR_CLEARED_ASIL_B_GLOBAL_UNSPECIFIED
 #undef MEMMAP_ERROR

/**** CORE[x] CONFIG DATA -- PF[x] ****/ /*[x] = 0..5*/
#elif defined ADC_START_SEC_CONFIG_DATA_ASIL_B_CORE[x]_UNSPECIFIED
 /*****User pragmas here for PF[x]*****/
 #undef ADC_START_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
 #undef MEMMAP_ERROR
#elif defined ADC_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
 /*****User pragmas here for PF[x]*****/
 #undef ADC_STOP_SEC_CONFIG_DATA_ASIL_B_CORE0_UNSPECIFIED
 #undef MEMMAP_ERROR

/**** CODE -- PF[x] ****/
#elif defined ADC_START_SEC_CODE_ASIL_B_GLOBAL
 /*****User pragmas here for PF[x]*****/
 #undef ADC_START_SEC_CODE_ASIL_B_GLOBAL
 #undef MEMMAP_ERROR
#elif defined ADC_STOP_SEC_CODE_ASIL_B_GLOBAL
 /*****User pragmas here for PF[x]*****/
 #undef ADC_STOP_SEC_CODE_ASIL_B_GLOBAL
 #undef MEMMAP_ERROR

#endif

#if defined MEMMAP_ERROR
#error "Adc_MemMap.h, wrong pragma command"
#endif
```

- **DET**

  The DET module is a part of the AUTOSAR stack that handles all the development errors reported by the BSW modules. The ADC driver reports all the development errors to the DET module through the `Det_ReportError()` API. The user of the ADC driver must process all the errors reported to the DET module through the `Det_ReportError()` API.

  The `Det.h` and `Det.c` files are provided in the MCAL package as a stub code and needs to be replaced with a complete DET module during the integration phase.

- **Mcal_Wrapper**

  This Driver performs reporting of the Production and Runtime errors. The Handling of the reported errors shall be done by the user. The `Mcal_Wrapper_Det_ReportRuntimeError()` API,

`Mcal_Wrapper_Dem_SetEventStatus()` API and `Mcal_Wrapper_Dem_ReportErrorStatus()` API are provided in the Mcal_Wrapper.c and Mcal_Wrapper.h files as a stub code, and can be updated by the integrator to handle the reported errors. The files Mcal_Wrapper.c and Mcal_Wrapper.h are user modifiable but function prototype is not user modifiable and by default the Mcal Wrapper function shall calls AUTOSAR DEM and DET Modules.

The user of the ADC driver shall process all the production errors (fail/pass) and Runtime errors reported to the Mcal_Wrapper module. The interface used for reporting production error In AUTOSAR version 4.2.2 is `Mcal_Wrapper_Dem_ReportErrorStatus()` and for AUTOSAR version 4.4.0 is `Mcal_Wrapper_Dem_SetEventStatus()`, for reporting Runtime error `Mcal_Wrapper_Det_ReportRuntimeError()` API is used. The Mcal_Wrapper.c and Mcal_Wrapper.h files are provided in the MCAL package as a stub code and can be replaced with a user specific production and Runtime error handling module/s during the integration phase.

- **SchM**

  The SchM module is a part of the RTE that manages the BSW Scheduler. The ADC driver uses the exclusive areas defined in the `SchM_Adc.h` file to protect the SFRs and variables from concurrent accesses from different threads. The SchMs identified for the ADC driver are:

  - KernelData

  - SrcRegAccess

  The `SchM_Adc.h` and `SchM_Adc.c` files are provided in the MCAL package as an example code and needs to updated by the integrator. The user must implement the SchM functions defined by the ADC driver as **suspend / resume** of interrupts for the CPU on which the API is invoked. A sample implementation of the SchM functions is as follows:

```
/**** Sample implementation of SchM_Adc.c ****/
#include "Os.h"

void SchM_Enter_Adc_KernelData(void)
{
 /* Start of Critical Section */
 SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Adc_KernelData(void)
{
 /* End of Critical Section */
 ResumeAllInterrupts(); /* Resume CPU core interrupt */
}

void SchM_Enter_Adc_SrcRegAccess(void)
{
 /* Start of Critical Section */
 SuspendAllInterrupts(); /* Suspend CPU core interrupt */
}

void SchM_Exit_Adc_SrcRegAccess(void)
{
 /* End of Critical Section */
 ResumeAllInterrupts(); /* Resume CPU core interrupt */
}
```

- **Safety error**

  The ADC driver will report all the detected safety errors through the `Mcal_ReportSafetyError()` API.

  The driver performs only detection and reporting of the safety errors. The handling of the reported errors should be carried out by the user. The `Mcal_ReportSafetyError()` API is provided in the `Mcal_SafetyError.c` and `Mcal_SafetyError.h` files as a stub code, and must be updated by the integrator to handle the reported errors.

  Note: All DET errors are also reported as safety errors (error code used is same as DET).

- **Notifications and callbacks**

  The driver does not implement any notifications. However, it does report the completion of a group conversion through notification functions. These notification functions can be configured by the user in the EB tresos for each AdcChannelGroup separately.

- **Operating system (OS)**

  The OS or the application must ensure correct type of service and interrupt priority is configured in the SR register. Enabling and disabling of interrupts must also be managed by the OS or application.

  The OS files provided by the MCAL package are only an example code and must be updated by the integrator with the actual OS files for the desired function.

  Note: The ADC driver updates the SRC registers of EVADC (`MODULE_SRC.VADC.G[].SRx`) to clear the pending interrupt requests.

1.1.4.2 Multicore and Resource Manager

The ADC driver supports the execution of its APIs in parallel from all CPU cores. The user should allocate resources of the EVADC to the CPU cores at pre-compile time using the Resource Manager module. The following are the key points to be considered with respect to multicore in the driver:

- ADC hardware groups can be allocated to the CPU cores at pre-compile time. For example, EVADC_G0 and EVADC_G1 can be allocated to core-2, EVADC_G9 to core-3 and EVADC_G3 to core-0
- An AdcChannel group belongs to a particular hardware group. Hence, all the AdcChannel groups belonging to a hardware group, get allocated to the same core as the hardware group
- It must be ensured that AdcChannel group passed as a parameter while invoking an API belongs to the same core on which the API is invoked
- DETs will be raised in case APIs are invoked with mismatch of core and AdcChannel group
- Interrupts raised by a hardware group must be serviced by the CPU core to which the hardware group has been allocated to
- Locating of constants, variables and configuration data to the correct memory space should be done by the user. Memory sections are marked GLOBAL(common to all cores) and CORE[x](specific to a CPU core). The following should be considered by the user to ensure better performance of the driver:

  Code section

  The executable code of the ADC driver is placed under single MemMap section. It can be relocated to any PFlash region.

  Data section

  The RAM variable memory sections marked as specific to a core should be re-located to the DSPR/DLMU of the same core. The sections marked as global should be relocated to the non-cached LMU region. In devices with no LMU, non-cached DSPR can be used.

  Configuration data and constants:

  The configuration data sections marked as specific to a core should be re-located to the PFlash of the same core. The sections marked as global should be relocated to the PFlash of the master core.

Note: Relocating code, data or constants to a distant memory region would impact execution timings.

Note: If the driver operates from single (master) core, all the sections may be relocated to the PFlash/DSPR/DLMU of the same CPU core.

1.1.4.3 MCU support

The ADC driver is dependent on the MCU driver for the clock configuration and timer IP-related services. The initialization of the ADC driver must be started only after completion of the MCU initialization. The following must be considered while configuring the MCU driver in the EB tresos:

- CONVCTRL block must be used if more than one analog converters are used by the application. The configuration and programming of the CONVCTRL block is managed directly by the MCU driver.
- AdcChannel groups may require conversions triggered by a timer event generated by the GTM. The ADC driver uses the services of the MCU to configure the GTM-related trigger events during runtime. The GTM channels used by the ADC driver must be reserved in the MCU configuration for exclusive use by the ADC.
- AdcChannel groups may require conversions triggered by a hardware event generated for EICR-IGCR. The EICR-IGCR channels used by the ADC driver must be reserved in the MCU configuration for exclusive use by the ADC.

1.1.4.4 Port support

The PORT driver configures the port pins of the entire microcontroller. The user must configure port pins used by the ADC driver through the PORT configuration and initialize the port pins prior to invoking the ADC initialization.

1.1.4.5 DMA support

The ADC driver may be configured such that the conversion results are directly transferred from the result register to the application buffers through the DMA move engines. The APIs and configuration parameters of the DMA driver may be used to achieve this. Enabling the DMA mode is a module-wide feature. When enabled, priority and queuing are not supported.

The result register event from EVADC triggers a service request, which is serviced by the DMA. The DMA move engine transfers the conversion results from result registers to the application buffers. Result event will be triggered on completion of conversion of the last channel of the AdcChannel group. A request source event is also triggered, which is serviced by the CPU and updates the status of the AdcChannel group.

The user must ensure the following points, while using this mode:

- Configuration to enable the DMA-based result transfer must be done through the EB tresos parameter: AdcResultHandlingImplementation.
- DMA channels intended to be used for the ADC driver must be reserved and configured through the DMA driver in the EB tresos.
- Consecutive result registers must be allocated to channels of an AdcChannel group in the EB tresos.
- ADC driver does not configure the DMA channels. The user of the ADC should invoke proper DMA APIs to start/stop the DMA channels before starting/stopping an AdcChannel group.
- Address space 0xD and 0xC should not be used for DMA-related usage. The MemMap sections allocating memory in the scratch pad RAM should always generate global addresses instead of local addresses.
- Since the Data CRC and Address CRC features of the DMA are not used for the ADC driver, the user should ensure that while using the DMA mode a plausibility check of the conversion result is performed either by redundancy or by other means.

1.1.4.6 Interrupt connections

The interrupt connections of the ADC driver are described in this section.

- **Result handling in interrupt mode**

  Conversion result transfer in interrupt is selected when AdcResultHandlingImplementation is equal to ADC_INTERRUPT_MODE_RESULT_HANDLING. In this mode, the conversion results are transferred from result registers to the application buffers in the ISR of the Request Source Event. Each request source of an ADC hardware group is assigned a dedicated service request line. The channel event generated as a result of the limit check feature is also assigned a dedicated service request line. The following figure depicts the interrupt connections required by the ADC driver.

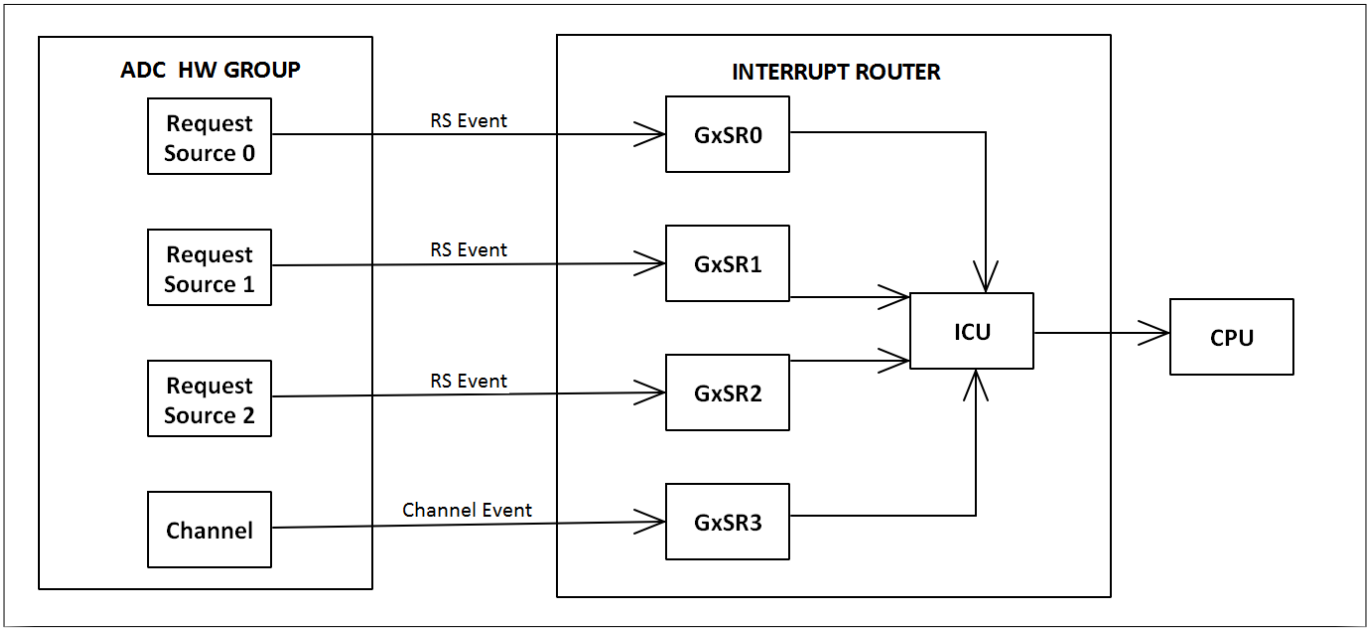

Figure 6 Result handling in the interrupt mode

Invoking the interrupt handlers provided by the driver must be done by the user. A sample invocation for EVADC_G0 and EVADC_G5 is shown as follows:

```
#include “Adc.h”
/**AdcResultHandlingImplementation = ADC_INTERRUPT_MODE_RESULT_HANDLING**/

/*************EVADC_G0*************/
/*********SRC_VADC_G0_SR1*********/
ISR(ADC0SR0_ISR)
{
 ENABLE(); /* Enable interrupts */
 /*The interrupt handler should be called from SRN0 of EVADC_G0*/
 Adc_RS0EventInterruptHandler(0u); /*0 indicates the HW group EVADC_G0*/
}

/*********SRC_VADC_G0_SR1*********/
ISR(ADC0SR1_ISR)
{
 ENABLE(); /* Enable interrupts */
 /*The interrupt handler should be called from SRN1 of EVADC_G0*/
 Adc_RS1EventInterruptHandler(0u); /*0 indicates the HW group EVADC_G0*/
}

/*********SRC_VADC_G0_SR2*********/
ISR(ADC0SR2_ISR)
{
 ENABLE(); /* Enable interrupts */
 /*The interrupt handler should be called from SRN2 of EVADC_G0*/
 Adc_RS2EventInterruptHandler(0u); /*0 indicates the HW group EVADC_G0*/
}

/*********SRC_VADC_G0_SR3*********/
ISR(ADC0SR3_ISR)
{
 ENABLE(); /* Enable interrupts */
 /*The interrupt handler should be called from SRN3 of EVADC_G0*/
 Adc_ChEventInterruptHandler(0u); /*0 indicates the HW group EVADC_G0*/
}

/*************EVADC_G5*************/
/*********SRC_VADC_G5_SR0*********/
ISR(ADC5SR0_ISR)
{
 ENABLE(); /* Enable interrupts */
 /*The interrupt handler should be called from SRN0 of EVADC_G5 */
 Adc_RS0EventInterruptHandler(5u); /*5 indicates the HW group EVADC_G5*/
}

/*********SRC_VADC_G5_SR1*********/
ISR(ADC5SR1_ISR)
{
 ENABLE(); /* Enable interrupts */
 /*The interrupt handler should be called from SRN1 of EVADC_G5 */
```

```
 Adc_RS1EventInterruptHandler(5u); /*5 indicates the HW group EVADC_G5*/
}

/*********SRC_VADC_G5_SR2*********/
ISR(ADC5SR2_ISR)
{
 ENABLE(); /* Enable interrupts */
 /*The interrupt handler should be called from SRN2 of EVADC_G5 */
 Adc_RS2EventInterruptHandler(5u); /*5 indicates the HW group EVADC_G5*/
}

/*********SRC_VADC_G5_SR3*********/
ISR(ADC5SR3_ISR)
{
 ENABLE(); /* Enable interrupts */
 /*The interrupt handler should be called from SRN3 of EVADC_G5 */
 Adc_ChEventInterruptHandler(5u); /*5 indicates the HW group EVADC_G5*/
}
```

- **Result handling in DMA mode:**
  Conversion result transfer through DMA is selected when AdcResultHandlingImplementation is equal to ADC_DMA_MODE_RESULT_HANDLING. In this mode, the conversion results are transferred from result registers to the application buffers by a DMA move engine. The result register event triggers a service request which is serviced by the DMA. The following figure represents the interrupt connectivity.

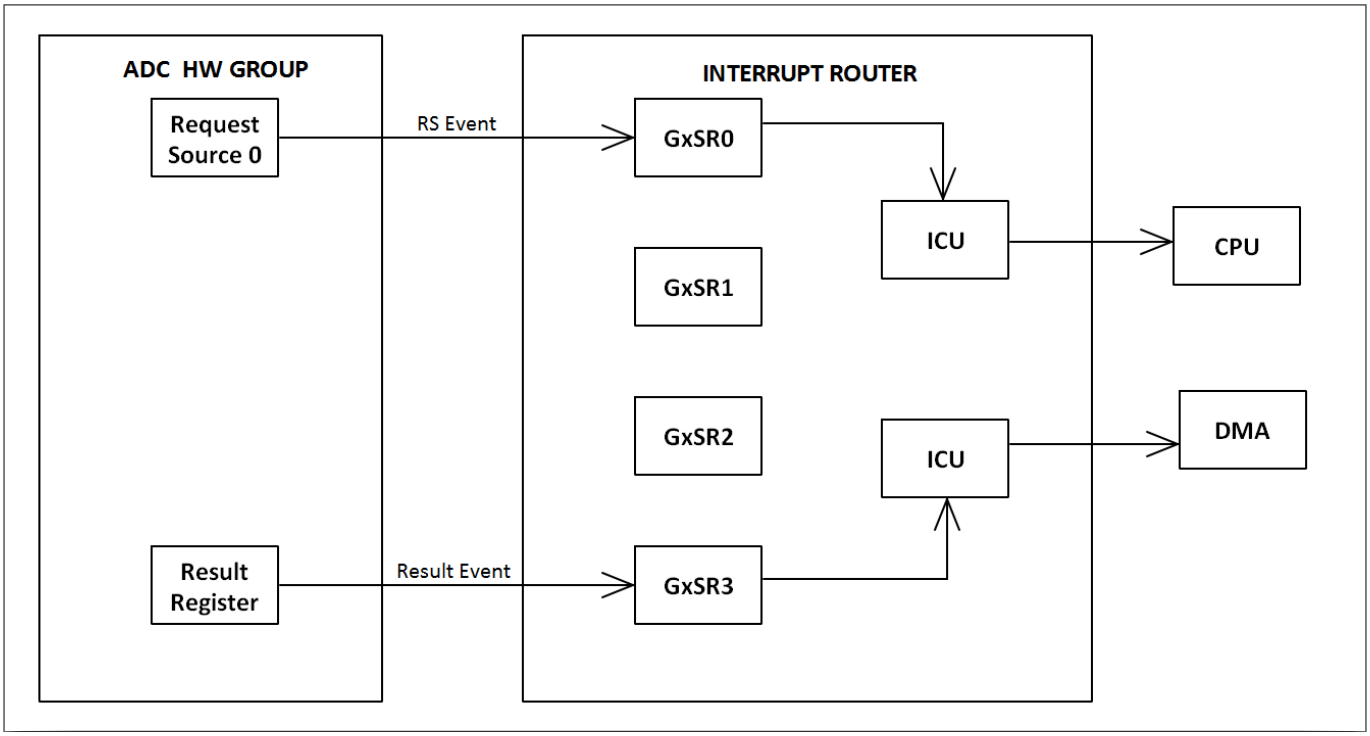

Figure 7 Result handling in the DMA mode

Invoking the interrupt handlers provided by the driver must be done by the user. A sample invocation for EVADC_G0 and EVADC_G5 is shown as follows:

```
#include “Adc.h”
/** AdcResultHandlingImplementation = ADC_DMA_MODE_RESULT_HANDLING **/
/************EVADC_G0*************/
/*********SRC_VADC_G0_SR0*********/
ISR(ADC0SR0_ISR)
{
 ENABLE(); /* Enable interrupts */
 /*The interrupt handler should be called from SRN0 of EVADC_G0 */
 Adc_RS0EventInterruptHandler(0u); /*0 indicates the HW group EVADC_G0*/
}
/**** Service Request 3 is service by DMA ****/
/**** Other Service Requests are unused ****/

/************EVADC_G5*************/
/*********SRC_VADC_G5_SR0*********/
ISR(ADC5SR0_ISR)
{
 ENABLE(); /* Enable interrupts */
 /*The interrupt handler should be called from SRN0 of EVADC_G5*/
 Adc_RS0EventInterruptHandler(5u); /*5 indicates the HW group EVADC_G5*/
}
/**** Service Request 3 is service by DMA ****/
/**** Other Service Requests are unused ****/
```

1.1.4.7 Example usage

The following are some of the key use cases of the ADC driver.

Note: Refer to the comments in the code snippets for additional information.

Initialization of the driver

The code sequence for initializing the ADC driver is as follows:

```
/*
Configuration values mandatory for below code snippet:-
1. AdcResultHandlingImplementation = ADC_DMA_MODE_RESULT_HANDLING: Then Dma_Init() is required
prior to use of runtime ADC services.
2. AdcStartupCalibApi = TRUE: Then Adc_GetStartupCalStatus() and Adc_TriggerStartupCal() can be
invoked
*/

#include “Adc.h”
#include “Mcu.h”
#include “Port.h”
#include “Dma.h”
#include “Irq.h”


/* MCU and GTM Initialization */
Mcu_Init(&Mcu_Config);
Mcu_InitClock(0U);
while(Mcu_GetPllStatus() != MCU_PLL_LOCKED);
Mcu_DistributePllClock();

/* Port Initialization */
Port_Init(&Port_Config);

#if(ADC_RESULT_HANDLING_IMPLEMENTATION == ADC_DMA_MODE_RESULT_HANDLING)
/* Dma Initialization, Only if DMA mode of Result
 handling is used */
Dma_Init(&Dma_Config);
#endif

/* ADC Initialization */
Adc_Init(&Adc_Config);

/* ADC Startup Calibration */
Adc_TriggerStartupCal();
/* Wait till the Start Calibration is over*/
while(Adc_GetStartupCalStatus() != ADC_STARTUP_CALIB_OVER);

/* Initialize the SRPN and TOS for used interrupts */
IrqAdc_Init();

/* Enable Interrupts for used ADC HW units(x) */
SRC_VADC_Gx_SR0.B.SRE = 1U;
SRC_VADC_Gx_SR1.B.SRE = 1U;
SRC_VADC_Gx_SR2.B.SRE = 1U;


#if((ADC_RESULT_HANDLING_IMPLEMENTATION == ADC_DMA_MODE_RESULT_HANDLING) ||
(ADC_ENABLE_LIMIT_CHECK == STD_ON))
SRC_VADC_Gx_SR3.B.SRE = 1U;
#endif
```

```
/* Further APIs of ADC driver can be called now */
```

Start software-triggered conversion

The code sequence for starting a software triggered group conversion is as follows:

```
/*
Configuration values mandatory for below code snippet:-
1. AdcEnableStartStopGroupApi = True
2. AdcGrpNotifCapability = True
3. AdcResultHandlingImplementation = ADC_INTERRUPT_MODE_RESULT_HANDLING
*/

#include “Adc.h”
Adc_GroupType Group ;
Adc_ValueGroupType DataBufferPtr[10]; //Assuming buffer size of 10 is sufficient for the
AdcChannel group being depicted
Std_ReturnType lRetVal;

/*AdcConf_AdcGroup_AdcXGroup_Y is a valid SW group ID macro
 generated in Adc_Cfg.h */
Group = AdcConf_AdcGroup_AdcXGroup_Y;

lRetVal = Adc_SetupResultBuffer(Group, &DataBufferPtr[0]);

if(lRetVal != E_NOT_OK)
{
 Adc_EnableGroupNotification(Group);
 Adc_StartGroupConversion(Group);
}
else
{
 /*Could not setup result buffer*/
}
```

Stop software-triggered conversion

The code sequence for stopping a software-triggered group conversion is as follows:

```
/*
Configuration values mandatory for below code snippet:-
1. AdcEnableStartStopGroupApi = True
*/

#include “Adc.h”
Adc_GroupType Group ;

/*AdcConf_AdcGroup_AdcXGroup_Y is a valid SW group ID macro
generated in Adc_Cfg.h */
Group = AdcConf_AdcGroup_AdcXGroup_Y;

/*Make sure Group has already been started by calling
Adc_StopGroupConversion API*/

Adc_StopGroupConversion(Group);
```

Enable hardware trigger for conversion

The code sequence for enabling a hardware trigger for a group conversion is as follows:

```
/*
Configuration values mandatory for below code snippet:-
1.AdcHwTriggerApi = True
2.AdcEnableStartStopGroupApi = True
3.AdcResultHandlingImplementation = ADC_INTERRUPT_MODE_RESULT_HANDLING
*/

#include “Adc.h”
Adc_GroupType Group ;
Adc_ValueGroupType DataBufferPtr[10]; //Assuming buffer size of 10 is sufficient for the
AdcChannel group being depicted
Std_ReturnType lRetVal;

/*AdcConf_AdcGroup_AdcXGroup_Y is a valid HW group ID macro generated in Adc_Cfg.h */

Group = AdcConf_AdcGroup_AdcXGroup_Y;

lRetVal = Adc_SetupResultBuffer(Group, &DataBufferPtr[0]);

if(lRetVal != E_NOT_OK)
{
 Adc_EnableGroupNotification(Group);
 Adc_EnableHardwareTrigger(Group);
}
else
{
 /*Could not setup result buffer*/
}
```

Disable hardware trigger for conversion

The code sequence for disabling a hardware trigger for a group conversion is as follows:

```
/*
Configuration values mandatory for below code snippet:-
1. AdcHwTriggerApi = True
*/
#include “Adc.h”
Adc_GroupType Group ;

/*AdcConf_AdcGroup_AdcXGroup_Y is a valid HW group ID */
Group = AdcConf_AdcGroup_AdcXGroup_Y;

/*Make sure Group has already been started by calling Adc_
EnableHardwareTrigger API*/

/* Disable the HW trigger */
Adc_DisableHardwareTrigger(Group);
```

Enable hardware trigger for conversion with DMA result handling

The code sequence for enabling a hardware trigger for a group conversion in the DMA mode of result handling is as follows:

```
/*
Configuration values mandatory for below code snippet:-
1. AdcHwTriggerApi = True
2. AdcEnableStartStopGroupApi = True
3. AdcResultHandlingImplementation = ADC_DMA_MODE_RESULT_HANDLING
*/

#include “Adc.h”
#include “Dma.h”
Adc_GroupType Group ;

/*AdcConf_AdcGroup_AdcXGroup_Y is a valid HW group ID macro
generated in Adc_Cfg.h */

Group = AdcConf_AdcGroup_AdcXGroup_Y;
/* DMA channel initialization is already completed as part of Dma_Init*/
/* Update the source address to SFR address of result register and destination address to RAM
buffer */
Dma_ChUpdate(0U, PointerToChannelUpdateStruct, NULL_PTR)
/* Enable HW trigger for the DMA Channel used */
Dma_ChEnableHardwareTrigger(0U);

/* Enable notification and HW trigger for the Group*/
Adc_EnableGroupNotification(Group);
Adc_EnableHardwareTrigger(Group);
```

Read results

The code sequence for reading results of completed conversions is as follows:

```
/*
Configuration values mandatory for below code snippet:-
1. AdcReadGroupApi = True
2. AdcResultHandlingImplementation = ADC_INTERRUPT_MODE_RESULT_HANDLING
*/

#include “Adc.h”
Adc_GroupType Group ;
Std_ReturnType Read_Err;
Adc_StatusType lRetVal;
Adc_ValueGroupType DataBufferPtr[10]; //Assuming buffer size of 10 is sufficient for the
AdcChannel group being depicted

/*AdcConf_AdcGroup_AdcXGroup_Y is a valid group ID macro generated in Adc_Cfg.h */

Group = AdcConf_AdcGroup_AdcXGroup_Y;

lRetVal = Adc_GetGroupStatus(Group);

if((lRetVal == ADC_STREAM_COMPLETED) || (lRetVal == ADC_COMPLETED))
{
 Read_Err = Adc_ReadGroup (Group, &DataBufferPtr[0]);
 if(Read_Err != E_NOT_OK)
 {
 /*Adc read group is successful*/
 }
}
else
{
 /*Results not ready*/
}
```

De-initialization of the driver

The code sequence for de-initialization of the driver is as follows:

```
/*
Configuration values mandatory for below code snippet:-
1. AdcDeInitApi = True
*/
#include “Adc.h”

/*Stop all the currently converting groups before calling Adc De-init so that De-Initialization
can go on successfully */

/* ADC De-Initialization */
Adc_DeInit();
```

Configuring an ADC channel group for GTM-based trigger and gate:

ADC channel groups conversions can be triggered on a timer event from the GTM Channel. The conversion for ADC channels in such a group occur every time a GTM timer event is detected. To address multiple application use cases the ADC driver provides multiple ways to configure the GTM channels as follows:

- **Time-based trigger using GTM**

  In this mode, the ADC driver configures the GTM channel (through the APIs of MCU) every time the `Adc_EnableHardwareTrigger()` API is called for a group having GTM as a trigger source.

  - The GTM channel indented to be used must be exclusively reserved in the MCU driver through the configuration parameters under the McuGtmAllocationConf container.

  - The multiplexer for GTM to EVADC trigger connections must be configured in the MCU driver through the configuration parameters listed under the GtmTriggerForAdc container.

  - The clock, related configuration of GTM must be done through the MCU driver configuration.

  - The AdcHwTriggerApi configuration parameter must be set to TRUE.

  - The correct GTM trigger line must be selected in the AdcGroup/AdcHwExtTrigSelect parameter.

  - The edge of the external request signal must be selected in the AdcGroup/AdcHwTrigSignal parameter.

  - The GTM timer and trigger interval must be configured in the container AdcGroup/ GtmTriggerTimerConfig.

- **PWM signal-based trigger using GTM**

  In this mode, the ADC driver relies upon the application or another driver to configure the GTM channel that generates the PWM signal.

  - The GTM timer indented for use must be configured by the user outside the ADC driver.

  - The multiplexer for GTM to EVADC trigger connections must be configured in the MCU driver through the configuration parameters listed under the GtmTriggerForAdc container.

  - The clock, related configuration of GTM must be done through the MCU driver configuration.

  - The AdcHwTriggerApi configuration parameter must be set to true.

  - The correct GTM trigger line must be selected in the AdcGroup/AdcHwExtTrigSelect parameter.

  - The edge of the external request signal must be selected in the AdcGroup/AdcHwTrigSignal parameter.

Similarly for gate signals, the GTM channel may be configured through the ADC driver, application or any other driver based on the use cases.

Configuring an ADC group for synchronous conversion

The ADC driver supports synchronous conversion of channels across the ADC hardware groups. This feature can be enabled through the `AdcSyncConvEnable, AdcSyncConvMode` and `AdcSyncConvChannelEnable` configuration parameters. The `AdcSyncConvMode` parameter determines the master and slave hardware groups. When an ADC channel group is triggered on the master hardware group then all the channels marked for synchronous conversion through `AdcSyncConvChannelEnable` will be triggered on the slave hardware groups also.

The driver configures the slave analog channels (CHCTR) with the same properties as that of the master analog channel. Hence, the input class used for the slave analog channels must have the same properties as the input class used for the master analog channel.

It is recommended to use the global input classes for the master channel, so that the slave is also configured with the same input class, eliminating different conversion properties. The conversion results for the channels of the slave hardware group will be stored in the application buffer after the conversion results for the channels of the master hardware group.

The buffer layout is explained in the diagram for the following scenario.

- Three channels are configured in an ADC channel group: CH9, CH1, and CH2

- Two streaming samples for each channel

- Kernel K0 synchronization master, kernel K1 and K2 synchronization slaves

- Channels CH9 and CH1 enabled for synchronous conversion

Note: If a channel on a slave kernel is converting or queued for conversion, the groups from the master kernel having the same slave channel configured as synchrounous cannot be queued or started. Such a scenario triggers a DET.

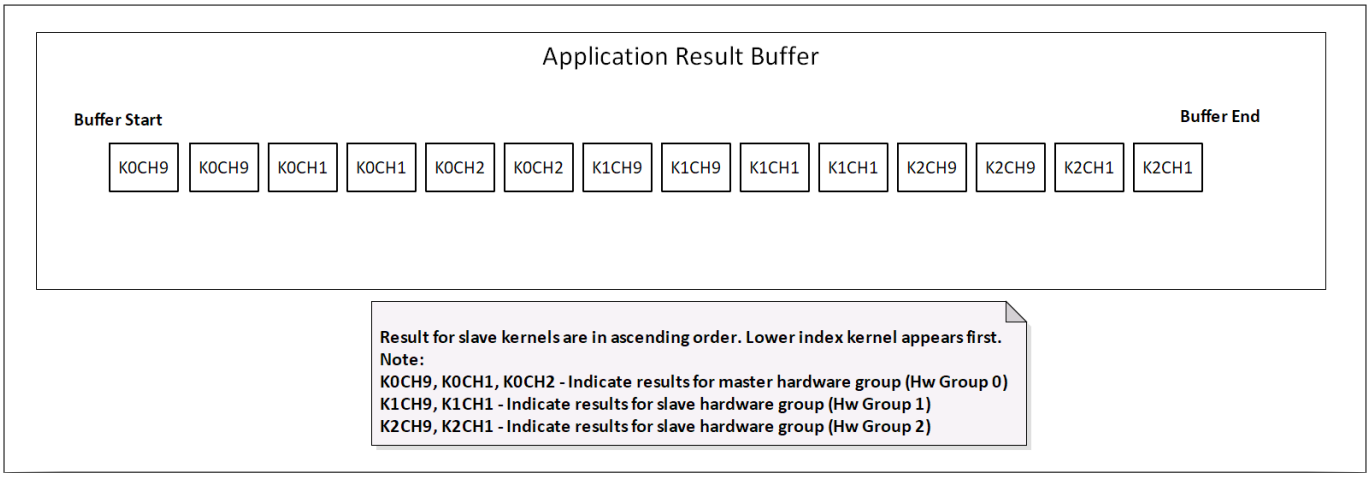

Figure 8 Synchronous conversion

1.1.5 Key architectural considerations

1.1.5.1 Modes of operation: priority and queuing

The ADC driver supports four modes of operation related to priority and queuing. The modes can be changed at pre-compile time by assigning appropriate value to AdcEnableQueuing and AdcPriorityImplementation. These modes are as follows:

- **Priority and queuing disabled**

  In this mode of operation, only one AdcChannel group can execute on a particular ADC hardware group at a given time. Groups can be scheduled in parallel on different ADC hardware groups.

  Configuration settings

  - AdcEnableQueuing: False

  - AdcPriorityImplementation: ADC_PRIORITY_NONE

- **Priority disabled and queuing enabled**

  The ADC driver maintains one software queue per ADC hardware group in this mode of operation. The software queue operates in the FCFS mode. An AdcChannel group cannot be placed in the software queue, if it is already present in the queue or currently converting or waiting for hardware trigger.

  Configuration settings

  - AdcEnableQueuing: True

  - AdcPriorityImplementation: ADC_PRIORITY_NONE

- **Hardware priority**

  The prioritization mechanism used in this mode is completely based on the priority of the Request Sources (hardware arbiter). Request sources have fixed priority: RS0 has a priority level of 0 (lowest), RS1 has a priority level of 1, and RS2 has a priority level of 2 (highest). Since the ADC hardware supports only three priority levels, a mapping with AdcChannel group priority is established as follows:

  - AdcChannel group priority number 0..253 maps to hardware priority 0 (lowest) and executes only on RS0

  - AdcChannel group priority 254 maps to hardware priority 1 and executes only on RS1

  - AdcChannel group priority 255 maps to hardware priority 2 (highest) and executes only on RS2

In this mode of operation, the AdcChannel group will be placed in the software queue if another AdcChannel group with the same priority is currently executing. The software queue operates in the FCFS mode. Hence, groups with the same priority in the queue will be executed in the order of the request. An AdcChannel group cannot be placed in the software queue, if it is already present in the queue or currently converting or triggered by hardware mechanism or sharing an analog channel / result register with another executing groups. Hardware triggered groups should never be placed in any software queue.

Configuration settings

- AdcEnableQueuing: False
- AdcPriorityImplementation: ADC_PRIORITY_HW

- **Hardware software priority**

  This is the full-fledged prioritization mechanism using both hardware and software priority mechanisms. Request sources do not have a fixed priority in this mode. The AdcChannel groups can be allocated priority levels from 0 to 255 (highest). In this mode of operation, the ADC driver maintains a prioritized software queue per ADC hardware group. An AdcChannel group cannot be placed in the software queue, if it is already present in the queue or currently converting or triggered by hardware mechanism or sharing a analog channel/result register with another executing groups. Hardware-triggered groups should never be placed in any software queue.

  Configuration settings

  - AdcEnableQueuing: False
  - AdcPriorityImplementation: ADC_PRIORITY_HW_SW

1.1.5.2 Modes of operation: result handling

The ADC driver supports two modes of operation related to result handling. The modes can be changed at pre-compile time by assigning appropriate value to AdcResultHandlingImplementation. The modes are listed as follows:

- **Interrupt-based result handling**

  In this mode, the conversion results are transferred from the result registers to the application buffers in the ISR of the Request Source Event.

  Each request source of an ADC hardware group is assigned a dedicated service request line.

  The channel event generated as a result of the limit check feature is also assigned a dedicated service request line.

  Configuration settings

  - AdcResultHandlingImplementation: ADC_INTERRUPT_MODE_RESULT_HANDLING

- **DMA-based result handling**

  In this mode, the conversion results are transferred from the result registers to the application buffers by a DMA move engine. When the DMA mode is enabled, priority and queuing are not supported (restricted through the tresos). Also, since the results are transferred to the application through the DMA, APIs related to the result buffer management and result reporting will not be available (restricted through the tresos). For more information, refer to the *DMA support* section.

  Configuration settings

  - AdcResultHandlingImplementation: ADC_DMA_MODE_RESULT_HANDLING

1.1.5.3 Reentrancy of APIs

The ADC driver functions are reentrant if the functions are called for different ADC channel groups. The reentrancy of the API functions applies only if the caller takes care that there is no simultaneous invocation of

different APIs for the same ADC Channel group. This is applicable for all API functions taking ADC Channel group as an input argument.

1.1.5.4 Accessing shared SFR

The ADC driver updates the SFR related to Service Request node and ERU. These SFR may be updated by the application software also. Hence, these updates must be done in a critical section or atomically.

- **Accessing ERU registers**: The ADC driver updates the MODULE_SCU.EICR and MODULE_SCU.IGCR registers to configure the trigger and gating signals. The update to these registers are done atomically by the driver. Any update to the MODULE_SCU.EICR and MODULE_SCU.IGCR by the application should be performed atomically, if the same register is used by the ADC driver also. This is required since two ERU channels share a common register. Therefore, update for one channel should not corrupt the ongoing write for another channel.
- **Accessing SRC registers**: The ADC driver updates the SRC registers of EVADC (MODULE_SRC.VADC.G[x].SRy) to clear the pending interrupt requests. The update to this register is done under a critical section using SchM locks. The application should update the MODULE_SRC.VADC.G[x].SRy under the SchM locks using the `SchM_Enter_Adc_SrcRegAccess()` and `SchM_Exit_Adc_SrcRegAccess()` functions. This will avoid corruption of the register, if updated by the driver and the application simultaneously.

1.1.5.5 Hardware trigger and gate mechanism

The ADC driver supports various hardware trigger and gate mechanisms to control the start or stop of the conversions.

- **GTM-based trigger and gate signal**

  The ADC driver supports the trigger and gate through the GTM-TOM and GTM-ATOM channels. The driver provides the following flexibility to the user for configuring the GTM channel for trigger and gate signals:

  - GTM channel can be configured through the tresos configuration parameters provided in the ADC driver. In such a case, the driver invokes APIs provided by the MCU driver to initialize the configured GTM channel at runtime.

  - Configuration and initialization of the GTM channels for trigger and gate signal must be done by the user if the parameters related to the GTM trigger and gate signals are not configured inside the ADC driver.
- **ERU-based trigger and gate signal**

  The ADC driver supports the trigger and gate through the ERU channels. The ERU channel must be configured through the tresos configuration parameters provided in the ADC driver. The initialization of the configured ERU channels is done at runtime by the ADC driver for the groups using ERU for trigger or gating.

1.1.5.6 EMUX feature

The ADC driver supports the EMUX feature of the EVADC. In this EMUX feature, the driver supports the sequence mode of operation. The driver supports the EMUX feature only when No priority mechanism is available and Synchronous conversion of hardware is stand-alone. The layout of result is adapted to the AUTOSAR recommended result buffer layout, and is only extended for channels of EMUX. The driver also provides the flexibility to the user for configuring the available hardware units to EMUX interfaces and to configure the EMUX channel for each AdcChannel group independently.

The EVADC hardware provides 3 EMUX select output lines. These pins need to be configured as output EMUX pins using the Port driver to use them as EMUX select pins. The ADC driver handles the copying of conversion results from result registers to the application buffer based on the number of external channels configured to an external 1-out-of-8 multiplexer.

Configuration settings:

- AdcEmuxEnable: TRUE

1.1.5.7 Safety features

The ADC driver supports the following safety features of the EVADC:

- **Broken Wire Detection**

  The ADC driver supports this safety feature on all the Request sources, hence this safety feature is supported on all the priority mechanisms.

- **Pull-Down Diagnostics**

  The ADC driver supports this safety feature only on Request source Q2 as per hardware restriction, hence the mapping of AdcChannel Group on Request source Q2 is supported only on hardware priority mechanism when AdcChannel group's priority level is 255.

  The EVADC hardware supports the Pull-Down Diagnostics feature only on selected analog input channels. If an analog input channel selected for Pull-Down Diagnostic belongs to an IO port, then the port pin needs to be configured using the PORT driver as prescribed in the hardware family description.

- **Multiplexer Diagnostics**

  The ADC driver supports this safety feature only on Request source Q2 as per hardware restriction, hence the mapping of AdcChannel Group on Request source Q2 is supported only on hardware priority mechanism when AdcChannel group's priority level is 255.

  The EVADC hardware supports the Multiplexer Diagnostics feature only on selected analog input channels. If an analog input channel selected for Multiplexer Diagnostic belongs to an IO port, then the port pin needs to be configured using the PORT driver as prescribed in the hardware family description.

- **Converter Diagnostics**

  The ADC driver supports this safety feature only on Request source Q2 as per hardware restriction, hence the mapping of AdcChannel Group on Request source Q2 is supported only on hardware priority mechanism when AdcChannel group's priority level is 255. This safety feature is supported only on non-existent channel number(channel 24), hence the driver uses the Alias feature to redirect channel 0 or 1 to channel 24.

Note: The driver provides the flexibility to the user for configuring all the diagnostic features at AdcChannel level independently based on the hardware derivative.

1.1.5.8 Alias Feature

The ADC driver supports the Alias feature of EVADC IP. The Alias feature allows user to configure conversion of any other ADC channels in place of channel CH0 and CH1.

Alias to channel CH0

ADC driver converts the channel CH4 instead of channel CH0 when CH4 is configured as alias to CH0, similar configuration is applicable for parameter AdcChannel1Alias.

Configuration settings:

- AdcGroupDefinition: Channel CH0

- AdcChannel0Alias: Channel CH4

In some hardware derivatives, channel CH0 and CH1 are not available. In such cases, dummy channel ACH0 and ACH1 are added only to support Alias feature. Since these channels are not physically accessible to user, an error check is provided to restrict usage of these channels only for Alias feature. However, there is no error check provided if configured Alias channels are physically present or not. So, it is user's responsibility to configure only the available channels for Alias feature.

1.2 Assumptions of Use (AoU)

The AoU for the ADC driver are as follows.

- **InitCheck Sequence**

User shall invoke Adc_InitCheck to ensure the initialization is done correctly.

The parameter AdcInitCheckApi shall be enabled and the user of ADC shall call InitCheck function before the execution of any runtime API (except GetVersionInfo) but after completion of ADC initialization sequence.
[cover parentID ADC={537E39DB-00D7-4267-A92A-92D45C0A1022}]

- **ConfigPtr passed to InitCheck**

User of ADC shall ensure that InitCheck is invoked with the same ConfigPtr that is used in Init.
[cover parentID ADC={C0A634B2-F373-47bf-BDF2-5FA3985F76B0}]

- **DMA channel initialization sequence**

The users of the ADC driver shall ensure that when the DMA mode of result handling is used, the DMA channel is setup/initialized before starting the ADC conversions. Also ensure that the DMA channel is stopped/deinitialized after stopping the ADC conversion.
[cover parentID ADC={F0B13815-84DD-4cc7-9A20-9542C8C85D2F}]

- **Priority of hardware trigger group**

The users of the ADC driver shall ensure that the priority of hardware-triggered group is higher compared to the software-triggered group in the continuous conversion mode. This ensures that hardware triggers are not missed when higher priority software-triggered groups are executing.
[cover parentID ADC={F5565B70-99D3-4fcd-80EA-B54CDFB19754}]

- **Groups using ERU as hardware trigger**

ADC channel groups having the same ERU (EICR/IGCR) channel as trigger/gating source shall not be triggered simultaneously. The driver does not perform any check if the ERU channel requested currently is already in use by another ADC channel group.

[cover parentID ADC={C8281063-A66F-4217-B299-21D0F17F33C9}]

- **Groups using GTM as hardware trigger**

ADC channel groups having the same GTM (TOM/ATOM) channel as trigger/gating source shall not be triggered simultaneously. The driver does not perform any check if the GTM channel requested currently is already in use by another ADC channel group.
[cover parentID ADC={1CB59FC5-2B01-4dbe-B658-454EE965A81F}]

- **Start-up calibration invocation-1**

The users of the ADC driver shall ensure that no conversion is ongoing before triggering the start-up calibration operation or started during the start-up calibration process. The status of the start-up calibration shall be checked through the Adc_GetStartupCalStatus API.
[cover parentID ADC={CE7E70BA-A387-4930-A6A3-0FF2BCF49550}]

- **Start-up calibration invocation-2**

The users of the ADC driver shall ensure that the Adc_TriggerStartupCal() API is invoked after completing the initialization and initialization check on all the cores.
[cover parentID ADC={E27FCBC4-A8EC-4edf-859A-95654C24D655}]

- **Valid pointer and result buffer pointers passed**

The users of the ADC driver shall ensure a valid pointer and result buffer pointer with the required size is passed to achieve the expected behavior. The driver does not perform any check on the size of the buffers and pointers.
[cover parentID ADC={094F8CCC-0436-4bde-A6A0-7A93DD4A5055}]

1.3 Reference information

1.3.1 Configuration interfaces

Supported configuration variant: Post-Build

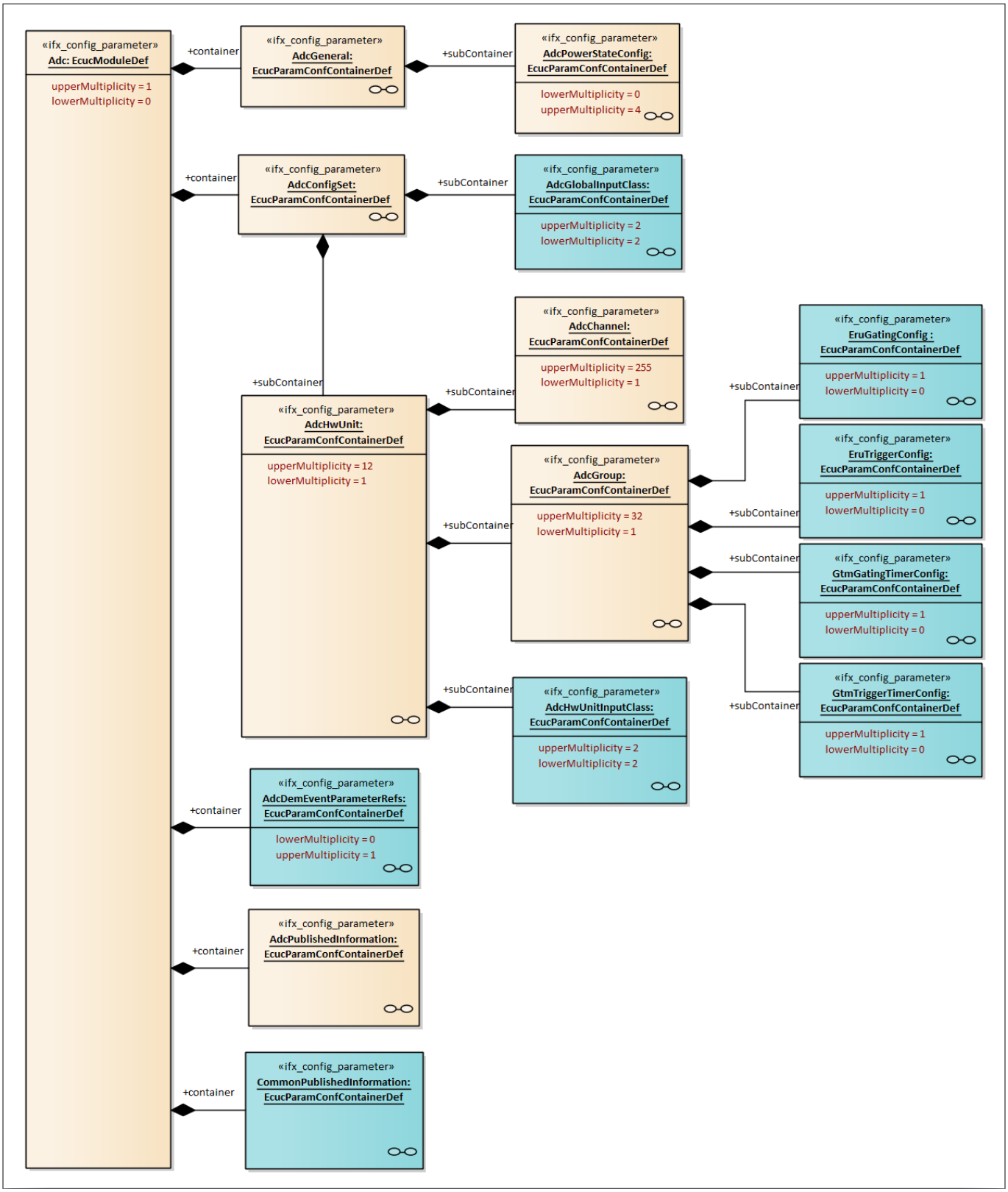

Figure 9 Container hierarchy along with their configuration parameters

1.3.1.1 Container: Adc

Configuration of the ADC module

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.2 Container: AdcChannel

The container contains the channel configuration (parameters). The short-name of all AdcChannels must be unique across the AdcHwUnits. Same AdcChannel can be assigned to multiple AdcGroups belonging to the same AdcHwUnit.

The upper multiplicity of this container is restricted to a maximum of 255 logical channels per ADC hardware group.

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

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.2.1 AdcAnChannelNum

Table 4 Specification for AdcAnChannelNum

Name	`AdcAnChannelNum`		
Description	The parameter defines the analog channel number as per the hardware. The possible range of values for analog channels is dependent on the value of the AdcHwUnitId parameter. The default value of this parameter is the first physical channel for each hardware group.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	G[x]CH[y]: where x=Hardware Unit ID y=Channel Number		
Default value	G[x]CH[y]		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcHwUnitId		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.2 AdcBWDEnable

Table 5 Specification for AdcBWDEnable

Name	`AdcBWDEnable`
Description	The parameter enables or disables the broken wire detection feature for the channel. The default value of this parameter is set to FALSE to minimize the executable code size.

(table continues...)

Table 5 (continued) Specification for AdcBWDEnable

Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcConverterDiagnosticEnable, AdcMultiplexerDiagnosticEnable, AdcPullDownDiagnosticEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.3 AdcBWDPrechargeLevel

Table 6 Specification for AdcBWDPrechargeLevel

Name	`AdcBWDPrechargeLevel`		
Description	The parameter determines the VAREF/VAGND pre-charging for the broken wire detection channel. The configuration of pre-charging for the broken wire detection channel is editable only when the AdcBWDEnable parameter is enabled.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_BWD_PRECH_VAGND: VAGND is used for pre-charging. ADC_BWD_PRECH_VAREF: VAREF is used for pre-charging.		
Default value	ADC_BWD_PRECH_VAGND		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcBWDEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.4 AdcChannelConvTime

Table 7 Specification for AdcChannelConvTime

Name	`AdcChannelConvTime`

(table continues...)

Table 7 (continued) Specification for AdcChannelConvTime

Description	The parameter defines the conversion time during which the analog value is converted into its digital representation for each channel. The AdcInputClassSelection parameter controls the channel conversion time. Therefore, this parameter is disabled in tresos and cannot be edited. *Note: The maximum value of this parameter is specified based on the maximum value generated from arxml.*		
Multiplicity	0..1	**Type**	EcucIntegerParamDef
Range	0 - 9223372036854775807		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.5 AdcChannelHighLimit

Table 8 Specification for AdcChannelHighLimit

Name	`AdcChannelHighLimit`		
Description	The parameter defines the upper limit used for limit checking. The default and maximum value of this parameter is added based on the 12-bit ADC converters value supported by the hardware. *Note: This parameter is configurable only if the AdcChannelLimitCheck parameter is set to TRUE and the AdcChannelRangeSelect parameter is neither equal to ADC_RANGE_UNDER_LOW nor to ADC_RANGE_NOT_UNDER_LOW.* *Note: The AdcChannelHighLimit parameter value has to be greater than or equal to the AdcChannelLowLimit parameter value.*		
Multiplicity	0..1	**Type**	EcucIntegerParamDef
Range	0 - 4095		
Default value	4095		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcChannelLowLimit, AdcChannelRangeSelect, AdcChannelLimitCheck		

(table continues...)

Table 8 (continued) Specification for AdcChannelHighLimit

Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.2.6 AdcChannelId

Table 9 Specification for AdcChannelId

Name	`AdcChannelId`		
Description	The parameter defines the numeric ID of the channel, which is the logical ID for the channel. It must be in consecutive sequence starting from zero for each ADC hardware unit. A symbolic name is generated for each channel. *Note: The maximum number of logical channels per hardware group is 255 (ID 0 to ID 254).*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 254		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.7 AdcChannelLimitCheck

Table 10 Specification for AdcChannelLimitCheck

Name	`AdcChannelLimitCheck`		
Description	The parameter enables or disables the limit checking feature for the current channel. This parameter is set to FALSE by default and it can be TRUE only if the AdcEnableLimitCheck parameter is set to TRUE. The ADC channels, with limit checking feature enabled, have to be assigned to a AdcGroup which consists exactly of one AdcChannel (itself). *Note: As per AUTOSAR, this is a pre-compile parameter. Since all the channel related parameters are post-build, it is simpler software design to implement this parameter also as post-build instead of pre-compile.*		
Multiplicity	0..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE

(table continues...)

Table 10 (continued) Specification for AdcChannelLimitCheck

Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcEnableLimitCheck		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.8 AdcChannelLowLimit

Table 11 Specification for AdcChannelLowLimit

Name	`AdcChannelLowLimit`		
Description	The parameter defines the lower limit used for limit checking. The default and maximum value of this parameter is added based on the 12-bit ADC converters value supported by the hardware. *Note: This parameter is configurable only if the AdcChannelLimitCheck parameter is set to TRUE and the AdcChannelRangeSelect parameter is neither equal to ADC_RANGE_OVER_HIGH nor to ADC_RANGE_NOT_OVER_HIGH.* *Note: The AdcChannelLowLimit parameter value has to be less than or equal to the AdcChannelHighLimit parameter value.*		
Multiplicity	0..1	**Type**	EcucIntegerParamDef
Range	0 - 4095		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcChannelHighLimit, AdcChannelRangeSelect, AdcChannelLimitCheck		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.9 AdcChannelRangeSelect

Table 12 Specification for AdcChannelRangeSelect

Name	`AdcChannelRangeSelect`

(table continues...)

Table 12 (continued) Specification for AdcChannelRangeSelect

Description	The parameter defines which conversion values are taken into account related to the limits defined with the AdcChannelLowLimit and AdcChannelHighLimit parameter. The default value of this parameter is set to ADC_RANGE_ALWAYS which effectively considers all the values as valid, as this is the behavior when limit check is disabled. *Note: This parameter is configurable only if the AdcChannelLimitCheck parameter is set to TRUE.*		
Multiplicity	0..1	**Type**	EcucEnumerationParamDef
Range	ADC_RANGE_ALWAYS: 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: Range below low limit, low limit value included.		
Default value	ADC_RANGE_ALWAYS		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcChannelLimitCheck		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.10 AdcChannelRefVoltsrcHigh

Table 13 Specification for AdcChannelRefVoltsrcHigh

Name	`AdcChannelRefVoltsrcHigh`		
Description	The parameter defines the upper reference voltage source for each channel. *Note:For Channel 0, value of this parameter should always be ADC_USE_VREF.* *Note: The AdcChannelRefVoltsrcHigh parameter cannot have value as ADC_USES_CH0, if the AdcAnChannelNum parameter contains a channel with fixed reference.*		
Multiplicity	0..1	**Type**	EcucEnumerationParamDef
Range	ADC_USES_CH0: Use channel CH0 as a reference voltage for conversion. ADC_USES_VREF: Use the analog reference voltage as a reference voltage for conversion.		
Default value	ADC_USES_VREF		

(table continues...)

Table 13 (continued) Specification for AdcChannelRefVoltsrcHigh

Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcAnChannelNum		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.11 AdcChannelRefVoltsrcLow

Table 14 Specification for AdcChannelRefVoltsrcLow

Name	`AdcChannelRefVoltsrcLow`		
Description	The parameter defines the ground reference voltage source for each channel. The default value of this parameter is set to ADC_USES_VAGND as only EVADC analog ground pin is supported by the hardware.		
Multiplicity	0..1	**Type**	EcucEnumerationParamDef
Range	ADC_USES_VAGND: Use the analog reference ground as a reference voltage for conversion.		
Default value	ADC_USES_VAGND		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.12 AdcChannelResolution

Table 15 Specification for AdcChannelResolution

Name	`AdcChannelResolution`		
Description	The parameter defines the channel resolution in bits. The converters operates only in 12-bit mode and hence the driver will update the result buffers with 12-bit conversion values only. Hence, this parameter will be un-editable is tresos and holds a default value of 12.		
Multiplicity	0..1	**Type**	EcucIntegerParamDef
Range	1 - 63		
Default value	12		

(table continues...)

Table 15 (continued) Specification for AdcChannelResolution

Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.13 AdcChannelSampTime

Table 16 Specification for AdcChannelSampTime

Name	`AdcChannelSampTime`		
Description	The parameter defines configuration of sampling time during which the value is sampled for each channel. The AdcInputClassSelection parameter controls the channel sampling time. Therefore, the AdcChannelSampTime parameter is disabled in tresos and cannot be edited. *Note: Maximum value of this parameter is specified based on the maximum value generated from arxml.*		
Multiplicity	0..1	**Type**	EcucIntegerParamDef
Range	0 - 9223372036854775807		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.14 AdcConverterDiagnosticEnable

Table 17 Specification for AdcConverterDiagnosticEnable

Name	`AdcConverterDiagnosticEnable`

(table continues...)

Table 17 (continued) Specification for AdcConverterDiagnosticEnable

Description	The parameter enables or disables the converter diagnostic feature for the channel. The default value of this parameter is set to FALSE to minimize the executable code size. *Note: This parameter is configurable only when the AdcPriorityImplementation parameter is ADC_PRIORITY_HW.* *Note: This parameter is configurable only when all other safety features are disabled.* *Note: This parameter is configurable only when the channel configured in the AdcAnChannelNum parameter supports the aliasing feature to alias the channels to a non-existent channel 24.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcBWDEnable, AdcPullDownDiagnosticEnable, AdcPriorityImplementation, AdcMultiplexerDiagnosticEnable, AdcAnChannelNum		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.15 AdcConverterDiagnosticsLevel

Table 18 Specification for AdcConverterDiagnosticsLevel

Name	`AdcConverterDiagnosticsLevel`		
Description	The parameter determines the converter diagnostic level for the channel. This parameter is editable only when the AdcConverterDiagnosticEnable parameter is set to TEUE. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_CD_PULL_DEVICE_HALF_VDDM: 1/2 of VDDM is used for converter diagnostic. ADC_CD_PULL_DEVICE_TWOTHIRD_VDDM: 2/3 of VDDM is used for converter diagnostic. ADC_CD_PULL_DEVICE_VDDM: VDDM is used for converter diagnostic. ADC_CD_PULL_DEVICE_VSSM: VSSM is used for converter diagnostic.		
Default value	ADC_CD_PULL_DEVICE_VDDM		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

(table continues...)

Table 18 (continued) Specification for AdcConverterDiagnosticsLevel

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcConverterDiagnosticEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.16 AdcInputClassSelection

Table 19 Specification for AdcInputClassSelection

Name	`AdcInputClassSelection`		
Description	The parameter defines the input class for the analog channel. Input class determines the channel conversion properties such as sample time and conversion mode for each channel. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_GLOBAL_CLASS_0: Global Class 0 is used ADC_GLOBAL_CLASS_1: Global Class 1 is used ADC_HWUNIT_CLASS_0: Hardware Unit Class 0 is used ADC_HWUNIT_CLASS_1: Hardware Unit Class 1 is used		
Default value	ADC_HWUNIT_CLASS_0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.17 AdcMultiplexerDiagnosticEnable

Table 20 Specification for AdcMultiplexerDiagnosticEnable

Name	`AdcMultiplexerDiagnosticEnable`

(table continues...)

Table 20 (continued) Specification for AdcMultiplexerDiagnosticEnable

Description	The parameter enables or disables the multiplexer diagnostic feature for the channel. The default value of this parameter is set to FALSE to minimize the executable code size. *Note: This parameter is configurable only when the AdcPriorityImplementation parameter is ADC_PRIORITY_HW.* *Note: This parameter is configurable only when all other safety features are disabled* *Note: This parameter is configurable only when the channel configured in the AdcAnChannelNum parameter supports the Multiplexer diagnostic feature.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcBWDEnable, AdcPullDownDiagnosticEnable, AdcAnChannelNum, AdcPriorityImplementation, AdcConverterDiagnosticEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.18 AdcMultiplexerDiagnosticLevel

Table 21 Specification for AdcMultiplexerDiagnosticLevel

Name	`AdcMultiplexerDiagnosticLevel`		
Description	The parameter determines the multiplexer diagnostic level for the channel. This parameter is editable only when the AdcMultiplexerDiagnosticEnable parameter is set to TRUE.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_MD_PULL_DOWN: Pull-down is used for multiplexer diagnostic. ADC_MD_PULL_UP: Pull-up is used for multiplexer diagnostic.		
Default value	ADC_MD_PULL_DOWN		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcMultiplexerDiagnosticEnable		

(table continues...)

Table 21 (continued) Specification for AdcMultiplexerDiagnosticLevel

Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.2.19 AdcPullDownDiagnosticEnable

Table 22 Specification for AdcPullDownDiagnosticEnable

Name	`AdcPullDownDiagnosticEnable`		
Description	The parameter enables or disables the pull-down diagnostic feature for the channel. The default value of this parameter is set to FALSE to minimize the executable code size. *Note: This parameter is configurable only when the AdcPriorityImplementation parameter is ADC_PRIORITY_HW.* *Note: This parameter is configurable only when all other safety features are disabled.* *Note: This parameter is configurable only when the channel configured in the AdcAnChannelNum parameter supports the Pull-down diagnostic feature.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcConverterDiagnosticEnable, AdcMultiplexerDiagnosticEnable, AdcBWDEnable, AdcAnChannelNum, AdcPriorityImplementation		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.2.20 AdcSyncConvChannelEnable

Table 23 Specification for AdcSyncConvChannelEnable

Name	`AdcSyncConvChannelEnable`		
Description	The parameter enables the synchronous A to D conversion for the current channel. Conversion of this channel will trigger a simultaneous conversion for the analog channel with the same number on the slave hardware groups also. The default value of this parameter is set to FALSE. *Note: This parameter is editable only for channels of AdcHwUnit configured as ADC_SYNC_MASTER.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef

(table continues...)

Table 23 (continued) Specification for AdcSyncConvChannelEnable

Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcChannelLimitCheck, AdcSyncConvEnable, AdcSyncConvMode		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.3 Container: AdcConfigSet

Container contains the configuration parameters and sub-containers of the ADC module.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.3.1 AdcSyncClockDisable

Table 24 Specification for AdcSyncClockDisable

Name	`AdcSyncClockDisable`		
Description	The parameter determines whether the analog clock is generated in synchronized mode or unsynchronized mode. The Converter Control (CONVCTRL) clock is configured by the MCU driver. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: This parameter writes into the common SFRs of all the kernels and can be implemented as post-build. Hence it is represented inside the AdcConfigSet container.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

(table continues...)

Table 24 (continued) Specification for AdcSyncClockDisable

Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.3.2 AdcSystemClock

Table 25 Specification for AdcSystemClock

Name	`AdcSystemClock`		
Description	The parameter refers to the system clock configured by the MCU driver. With this system clock, the reload value corresponding to the configured timer is calculated for the timer triggered AdcChannel groups. *Note: This parameter writes into common SFRs of all the kernels and can be implemented as post-build. Hence it is represented inside the AdcConfigSet container.*		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: McuClockReferencePointConfig		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4 Container: AdcDemEventParameterRefs

Container lists the production errors supported by the ADC driver. This container must be present when safety check is enabled.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.4.1 AdcClcFailureNotification

Table 26 Specification for AdcClcFailureNotification

Name	`AdcClcFailureNotification`		
Description	The parameter defines whether CLC failure production error notification is enabled or not. This parameter must be present when safety check is enabled.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		

(table continues...)

Table 26 (continued) Specification for AdcClcFailureNotification

Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	AdcSafetyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.4.2 AdcConvStopTimeNotification

Table 27 Specification for AdcConvStopTimeNotification

Name	`AdcConvStopTimeNotification`		
Description	The parameter determines whether conversion stop failure production error notification is enabled or not. This parameter must be present when safety check is enabled.		
Multiplicity	0..1	**Type**	EcucSymbolicNameReferenceDef
Range	Reference to Node: DemEventParameter		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	IFX	**Scope**	LOCAL
Dependency	AdcSafetyEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5 Container: AdcGeneral

Container contains all the general configuration parameters for the ADC driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.5.1 AdcDeInitApi

Table 28 Specification for AdcDeInitApi

Name	`AdcDeInitApi`
Description	The parameter adds or removes the Adc_DeInit() API from the code. The default value of this parameter is set to FALSE to minimize the executable code size.

(table continues...)

Table 28 (continued) Specification for AdcDeInitApi

Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.2 AdcDevErrorDetect

Table 29 Specification for AdcDevErrorDetect

Name	`AdcDevErrorDetect`		
Description	The parameter enables or disables the Default Error Tracer (DET) detection and reporting. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.3 AdcEcucPartitionRef

Table 30 Specification for AdcEcucPartitionRef

Name	`AdcEcucPartitionRef`

(table continues...)

Table 30 (continued) Specification for AdcEcucPartitionRef

Description	The parameter maps the ADC driver to zero or multiple ECUC partitions to make the driver API available in the according partition. *Note: The parameter support is added only for AUTOSAR schema compliance. This parameter is not used in code generation logic, hence this parameter is made editable false.*		
Multiplicity	0..*	**Type**	EcucReferenceDef
Range	Reference to Node: EcucPartition		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.5.4 AdcEmuxEnable

Table 31 Specification for AdcEmuxEnable

Name	`AdcEmuxEnable`		
Description	The parameter enables or disables the EMUX conversions. The default value of this parameter is set to FALSE to minimize the executable code size. *Note: EMUX feature is supported only when the AdcPriorityImplementation parameter is set to ADC_PRIORITY_NONE.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcPriorityImplementation		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.5 AdcEmuxGroupInterface0

Table 32 Specification for AdcEmuxGroupInterface0

Name	`AdcEmuxGroupInterface0`		
Description	The parameter is used to route the external multiplexer control signals of the available ADC hardware unit to the EMUX interface 0. The lowest ADC hardware unit available is selected as the default value for this parameter. *Note: The value of the AdcEmuxGroupInterface0 and AdcEmuxGroupInterface1 parameters should not be the same ADC hardware unit.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_HWUNIT[x]: [x] = list of available Hardware units.		
Default value	ADC_HWUNIT[x]		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcEmuxGroupInterface1, AdcEmuxEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.6 AdcEmuxGroupInterface1

Table 33 Specification for AdcEmuxGroupInterface1

Name	`AdcEmuxGroupInterface1`		
Description	The parameter is used to route the external multiplexer control signals of the available ADC hardware unit to the EMUX interface 1. The second lowest ADC hardware unit available is selected as the default value for this parameter. *Note: The value of the AdcEmuxGroupInterface0 and AdcEmuxGroupInterface1 parameters should not be the same ADC hardware unit.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_HWUNIT[x]: [x] = list of available Hardware units.		
Default value	ADC_HWUNIT[x]		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL

(table continues...)

Table 33 (continued) Specification for AdcEmuxGroupInterface1	
Dependency	AdcEmuxGroupInterface0, AdcEmuxEnable
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.5.7 AdcEnableLimitCheck

Table 34 Specification for AdcEnableLimitCheck

Name	`AdcEnableLimitCheck`		
Description	The parameter enables or disables the limit checking feature in the ADC driver. The default value of this parameter is set to FALSE to minimize the executable code size. *Note: The AdcEnableLimitCheck parameter cannot be set to TRUE, when the AdcResultHandlingImplementation parameter is set to ADC_DMA_MODE_RESULT_HANDLING.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcResultHandlingImplementation		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.8 AdcEnableQueuing

Table 35 Specification for AdcEnableQueuing

Name	`AdcEnableQueuing`		
Description	The parameter determines whether the queuing mechanism is active or not. The default value of this parameter is set to FALSE to minimize the executable code size. *Note: The parameter is editable and evaluated only when the AdcPriorityImplementation parameter is set as ADC_PRIORITY_NONE, the AdcEnableStartStopGroupApi parameter is set as TRUE and the AdcResultHandlingImplementation parameter is ADC_INTERRUPT_MODE_RESULT_HANDLING.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		

(table continues...)

Table 35 (continued) Specification for AdcEnableQueuing

Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcResultHandlingImplementation, AdcEnableStartStopGroupApi, AdcPriorityImplementation		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.9 AdcEnableStartStopGroupApi

Table 36 Specification for AdcEnableStartStopGroupApi

Name	`AdcEnableStartStopGroupApi`		
Description	The parameter determines if the software trigger mechanisms (Adc_StartGroupConversion() and Adc_StopGroupConversion ()) are available at runtime. When configured as TRUE, these APIs are available at runtime. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.10 AdcGrpNotifCapability

Table 37 Specification for AdcGrpNotifCapability

Name	`AdcGrpNotifCapability`

(table continues...)

Table 37 (continued) Specification for AdcGrpNotifCapability

Description	The parameter determines if the group notification mechanisms (Adc_EnableGroupNotification() and Adc_DisableGroupNotification()) are available at runtime. When configured as TRUE, these APIs are available at runtime. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.11 AdcHwTriggerApi

Table 38 Specification for AdcHwTriggerApi

Name	`AdcHwTriggerApi`		
Description	The parameter determines if the hardware trigger mechanisms (Adc_EnableHardwareTrigger() and Adc_DisableHardwareTrigger()) are available at runtime. When configured as TRUE, these APIs are available at runtime. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.12 AdcInitCheckApi

Table 39 Specification for AdcInitCheckApi

Name	AdcInitCheckApi		
Description	The parameter adds or removes the Adc_InitCheck() API from the code. When configured as TRUE, the API is available at runtime. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.13 AdcInitDeInitApiMode

Table 40 Specification for AdcInitDeInitApiMode

Name	AdcInitDeInitApiMode		
Description	The parameter defines the privilege mode in which the initialization and deinitialization APIs would operate. Since the ADC driver accesses the SFRs, it is efficient to operate the ADC driver in supervisory mode than the USER1 mode. Hence, the default mode of operation is the supervisory mode.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_MCAL_SUPERVISOR: Operating mode used is Supervisory. ADC_MCAL_USER1: Operating mode used is USER1.		
Default value	ADC_MCAL_SUPERVISOR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

(table continues...)

Table 40	(continued) Specification for AdcInitDeInitApiMode
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.5.14 AdcKernelEcucPartitionRef

Table 41 Specification for AdcKernelEcucPartitionRef

Name	`AdcKernelEcucPartitionRef`		
Description	The parameter maps the ADC kernel to zero or one ECUC partitions to assign the driver kernel to a certain core. The ECUC partition referenced is a subset of the ECUC partitions where the ADC driver is mapped to. *Note: The parameter support is added only for AUTOSAR schema compliance. This parameter is not used in code generation logic, hence this parameter is made editable false.*		
Multiplicity	0..1	**Type**	EcucReferenceDef
Range	Reference to Node: EcucPartition		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.5.15 AdcLowPowerStatesSupport

Table 42 Specification for AdcLowPowerStatesSupport

Name	`AdcLowPowerStatesSupport`		
Description	The parameter adds or removes all power state management related APIs (Adc_SetPowerState, Adc_GetCurrentPowerState, Adc_GetTargetPowerState, Adc_PreparePowerState) from the code. When configured as TRUE, these APIs are available at runtime. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	0..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE

(table continues...)

Table 42 (continued) Specification for AdcLowPowerStatesSupport

Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.16 AdcMaxChConvTimeCount

Table 43 Specification for AdcMaxChConvTimeCount

Name	`AdcMaxChConvTimeCount`		
Description	The parameter determines the maximum channel conversion time amongst all the configured channels. This parameter value is used by the ADC driver while trying to stop an ongoing conversion. This will make sure that the currently converting channel completes the conversion after a stop request and EVADC hardware unit goes to idle state before a new conversion is started by the driver. The default value of this parameter is set to 0 as default values of the AdcPriorityImplementation parameter is ADC_PRIORITY_NONE and the AdcEnableQueuing parameter is FALSE. If value of either of these parameters is modified, the value of AdcMaxChConvTimeCount shall be set to be greater than the default time to stop the conversion. The default stop time is based on the input class configuration of the hardware. *Note: In case of production error ADC_E_CONV_STOP_TIME_FAILURE, user is expected to increase the timeout value in this configuration parameter.* *Note: Range of this parameter is from 16 to 16962 when the AdcPriorityImplementation parameter is set to ADC_PRIORITY_HW or ADC_PRIORITY_HW_SW or the AdcEnableQueuing parameter is set to TRUE.* *Note: The value of this parameter is 0 when the AdcPriorityImplementation parameter is set to ADC_PRIORITY_NONE and the AdcEnableQueuing parameter is set to FALSE.* *Note: An error message will be raised if value entered in this parameter is not in the specified range.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 16962		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcEnableQueuing, AdcPriorityImplementation		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.17 AdcMultiCoreErrorDetect

Table 44 Specification for AdcMultiCoreErrorDetect

Name	`AdcMultiCoreErrorDetect`		
Description	The parameter enables or disables the multicore related default error tracer (DET) detection and reporting. It is applicable only when DETs are enabled. When set to TRUE, detection and reporting of multicore related errors is enabled. The default value of this parameter is set to FALSE to minimize the executable code size. *Note: An error message is raised if this parameter is set to TRUE when the device is a single-core device.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcDevErrorDetect		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.18 AdcPowerStateAsynchTransitionMode

Table 45 Specification for AdcPowerStateAsynchTransitionMode

Name	`AdcPowerStateAsynchTransitionMode`		
Description	The parameter enables or disables the support of asynchronous power state transition. *Note: Since power state transitions are always synchronous, this parameter is disabled in tresos.*		
Multiplicity	0..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	FALSE
Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile

(table continues...)

Table 45 (continued) Specification for AdcPowerStateAsynchTransitionMode

Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.19 AdcPriorityImplementation

Table 46 Specification for AdcPriorityImplementation

Name	`AdcPriorityImplementation`		
Description	The parameter determines whether a priority mechanism for prioritization of the conversion requests is available.Two types of prioritization mechanisms can be selected. - The hardware prioritization mechanism (ADC_PRIORITY_HW) uses the ADC hardware features for prioritization. - The mixed hardware and software prioritization mechanism (ADC_PRIORITY_HW_SW) uses the ADC hardware features and software logic for prioritization of AdcChannel groups. The group priorities for software triggered groups are typically configured with lower priority levels than the group priorities for hardware triggered groups.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_PRIORITY_HW: Only hardware priority mechanism is available ADC_PRIORITY_HW_SW: Both hardware and software priority mechanisms are available ADC_PRIORITY_NONE: Priority mechanism is not available		
Default value	ADC_PRIORITY_NONE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.20 AdcReadGroupApi

Table 47 Specification for AdcReadGroupApi

Name	`AdcReadGroupApi`
Description	The parameter adds or removes the Adc_ReadGroup() API from the code. When set to TRUE, the API is available at runtime. The default value of this parameter is set to FALSE to minimize the executable code size. *Note: The AdcReadGroupApiparameter shall be FALSE, when AdcResultHandlingImplementation parameter is set to ADC_DMA_MODE_RESULT_HANDLING.*

(table continues...)

Table 47 (continued) Specification for AdcReadGroupApi

Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcResultHandlingImplementation		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.21 AdcResultAlignment

Table 48 Specification for AdcResultAlignment

Name	`AdcResultAlignment`		
Description	The parameter determines whether the raw ADC results in the ADC result buffer are left aligned or right aligned. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_ALIGN_LEFT: Raw results are left aligned ADC_ALIGN_RIGHT: Raw results are right aligned		
Default value	ADC_ALIGN_RIGHT		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.22 AdcResultHandlingImplementation

Table 49 Specification for AdcResultHandlingImplementation

Name	`AdcResultHandlingImplementation`

(table continues...)

Table 49 (continued) Specification for AdcResultHandlingImplementation

Description	The parameter determines the result handling mechanism for the ADC driver.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_DMA_MODE_RESULT_HANDLING: Conversion results are transferred using a DMA channel. ADC_INTERRUPT_MODE_RESULT_HANDLING: Conversion results are transferred in the Request Source ISR.		
Default value	ADC_INTERRUPT_MODE_RESULT_HANDLING		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcPriorityImplementation		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.23 AdcRunTimeErrorDetect

Table 50 Specification for AdcRunTimeErrorDetect

Name	`AdcRunTimeErrorDetect`		
Description	The parameter enables or disables the runtime error detection and reporting. The default value of this parameter is set to TRUE to ensure the runtime error detection during the product lifecycle. *Note: Detection and reporting of runtime error should be enabled, if safety is enabled. An error message is raised if this parameter is set to FALSE when the AdcSafetyEnable parameter is set to TRUE.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcSafetyEnable		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.5.24 AdcRuntimeApiMode

Table 51 Specification for AdcRuntimeApiMode

Name	`AdcRuntimeApiMode`		
Description	The parameter defines the privilege mode, in which the runtime APIs would operate. Since the ADC driver accesses the SFRs, it is efficient to operate the ADC driver in supervisory mode. Hence, the default mode of operation is supervisory mode. The AdcRuntimeApiMode parameter shall be configured as User-1 mode if the AdcInitDeInitApiMode parameter is configured as User-1 mode.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_MCAL_SUPERVISOR: Operating mode used is Supervisory. ADC_MCAL_USER1: Operating mode used is USER1.		
Default value	ADC_MCAL_SUPERVISOR		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcInitDeInitApiMode		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.25 AdcSafetyEnable

Table 52 Specification for AdcSafetyEnable

Name	`AdcSafetyEnable`		
Description	The parameter determines whether to enable or disable the safety check and reporting. The detection of safety related errors is enabled by default to ensure that safety issues are addressed during the product lifecycle.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	TRUE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL

(table continues...)

Table 52 (continued) Specification for AdcSafetyEnable	
Dependency	-
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.5.26 AdcSleepMode

Table 53 Specification for AdcSleepMode			
Name	`AdcSleepMode`		
Description	The parameter determines whether the ADC driver accepts or rejects the sleep mode request from the SCU IP. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_SLEEP_MODE_ACCEPT: Sleep Mode request from SCU is accepted. ADC_SLEEP_MODE_REJECT: Sleep Mode request from SCU is rejected.		
Default value	ADC_SLEEP_MODE_ACCEPT		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.27 AdcStartupCalibApi

Table 54 Specification for AdcStartupCalibApi			
Name	`AdcStartupCalibApi`		
Description	The parameter adds or removes the Adc_GetStartupCalStatus() and Adc_TriggerStartupCal() APIs from the code. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 54 (continued) Specification for AdcStartupCalibApi

Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.28 AdcSupplyVoltage

Table 55 Specification for AdcSupplyVoltage

Name	`AdcSupplyVoltage`		
Description	The parameter adjusts the analog circuitry to the supply voltage. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_VOLTAGE_3P3V: Fixed 3.3V power supply is connected. ADC_VOLTAGE_5V: Fixed 5V power supply is connected. ADC_VOLTAGE_CONTROLLED_BY_SUPPLY: Automatic control: Voltage range is controlled by the power supply.		
Default value	ADC_VOLTAGE_CONTROLLED_BY_SUPPLY		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.29 AdcSyncConvEnable

Table 56 Specification for AdcSyncConvEnable

Name	`AdcSyncConvEnable`		
Description	The parameter enables or disables the synchronous conversions across the ADC hardware groups. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef

(table continues...)

Table 56 (continued) Specification for AdcSyncConvEnable

Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcResultHandlingImplementation		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.30 AdcTriggerOneConversionEnable

Table 57 Specification for AdcTriggerOneConversionEnable

Name	`AdcTriggerOneConversionEnable`		
Description	The parameter enables the execution of one dummy conversion for each configured hardware unit before start-up calibration is triggered in the Adc_TriggerStartupCal() API. (Refer Errata ADC_TC.083) The default value of this parameter is set to FALSE to minimize the executable code size. *Note: An error message will be raised if this parameter is set to TRUE while the AdcStartupCalibApi parameter is set to FALSE.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcStartupCalibApi		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.5.31 AdcVersionInfoApi

Table 58 Specification for AdcVersionInfoApi

Name	`AdcVersionInfoApi`

(table continues...)

Table 58 (continued) Specification for AdcVersionInfoApi

Description	The parameter adds or removes the Adc_GetVersionInfo() API from the code. When set to TRUE, the API is available at runtime. The default value of this parameter is set to FALSE to minimize the executable code size.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6 Container: AdcGlobalInputClass

Container defines the parameters for global input classes.

The multiplicity of this parameter is added based on the number of global classes supported by the hardware.

Note: This parameter writes into common SFRs of all the kernels and can be implemented as post-build. Hence, it is represented inside the AdcConfigSet container.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.6.1 AdcChConvMode

Table 59 Specification for AdcChConvMode

Name	AdcChConvMode		
Description	The parameter defines the noise reduction level for standard conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_NOISE_REDUCTION_STEPS_0: Standard Conversion ADC_NOISE_REDUCTION_STEPS_1: One additional conversion step for Noise reduction ADC_NOISE_REDUCTION_STEPS_3: Three additional conversion steps for Noise reduction ADC_NOISE_REDUCTION_STEPS_7: Seven additional conversion steps for Noise reduction		
Default value	ADC_NOISE_REDUCTION_STEPS_0		

(table continues...)

Table 59 (continued) Specification for AdcChConvMode

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.2 AdcChPreChargeClkCycles

Table 60 Specification for AdcChPreChargeClkCycles

Name	`AdcChPreChargeClkCycles`		
Description	The parameter defines the number of analog input precharge clock cycles for standard conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_INPUT_PRECHARGE_CYCLES_0: No precharge ADC_INPUT_PRECHARGE_CYCLES_16: Precharge for sixteen clock cycles ADC_INPUT_PRECHARGE_CYCLES_32: Precharge for thirty two clock cycles ADC_INPUT_PRECHARGE_CYCLES_8: Precharge for eight clock cycles		
Default value	ADC_INPUT_PRECHARGE_CYCLES_0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.3 AdcChSESPSEnable

Table 61 Specification for AdcChSESPSEnable

Name	`AdcChSESPSEnable`
Description	The parameter defines whether the Spread Early Sample Point for the standard conversions is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.

(table continues...)

Table 61 (continued) Specification for AdcChSESPSEnable

Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.4 AdcChSampleTime

Table 62 Specification for AdcChSampleTime

Name	`AdcChSampleTime`		
Description	The parameter defines the number of clock cycles to be added to the minimum sample phase of two sample clock cycles. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 31		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.5 AdcEmuxChConvMode

Table 63 Specification for AdcEmuxChConvMode

Name	`AdcEmuxChConvMode`
Description	The parameter defines the noise reduction level for the EMUX conversions. The default value of this parameter is set to the reset value of the corresponding SFR.

(table continues...)

Table 63 (continued) Specification for AdcEmuxChConvMode

Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_NOISE_REDUCTION_STEPS_0: Standard Conversion ADC_NOISE_REDUCTION_STEPS_1: One additional conversion step for Noise reduction ADC_NOISE_REDUCTION_STEPS_3: Three additional conversion steps for Noise reduction ADC_NOISE_REDUCTION_STEPS_7: Seven additional conversion steps for Noise reduction		
Default value	ADC_NOISE_REDUCTION_STEPS_0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcEmuxEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.6 AdcEmuxChPreChargeClkCycles

Table 64 Specification for AdcEmuxChPreChargeClkCycles

Name	`AdcEmuxChPreChargeClkCycles`		
Description	The parameter defines the number of analog input precharge clock cycles for the EMUX conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_INPUT_PRECHARGE_CYCLES_0: No precharge ADC_INPUT_PRECHARGE_CYCLES_16: Precharge for sixteen clock cycles ADC_INPUT_PRECHARGE_CYCLES_32: Precharge for thirty two clock cycles ADC_INPUT_PRECHARGE_CYCLES_8: Precharge for eight clock cycles		
Default value	ADC_INPUT_PRECHARGE_CYCLES_0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcEmuxEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.7 AdcEmuxChSESPSEnable

Table 65 Specification for AdcEmuxChSESPSEnable

Name	AdcEmuxChSESPSEnable		
Description	The parameter defines whether the Spread Early Sample Point for the EMUX conversions is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcEmuxEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.6.8 AdcEmuxChSampleTime

Table 66 Specification for AdcEmuxChSampleTime

Name	AdcEmuxChSampleTime		
Description	The parameter defines the number of clock cycles to be added to the minimum sample phase of two sample clock cycles for the EMUX conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 31		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcEmuxEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7 Container: AdcGroup

Container contains the group configuration (parameters). The short-name of all AdcGroups must be unique across the ADC hardware groups.

That a minimum of one channel exists for a group is controlled through the lower multiplicity of AdcGroupDefinition parameter. The upper multiplicity of this container is added as 32 per ADC hardware unit, which is more than sufficient for application use case, given that the number of channels in a sequence is eight for primary hardware groups and sixteen for secondary hardware groups supported as per the EVADC hardware unit.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.7.1 AdcChannel0Alias

Table 67 Specification for AdcChannel0Alias

Name	`AdcChannel0Alias`		
Description	The parameter configures the alias for Channel0. The hardware supports maximum of eight channels (0 to 7) for primary hardware groups and sixteen channels (0 to 15) for secondary hardware groups. The channel numbers upto 31 can be configured for diagnostics use cases. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: A value of 0 here, indicates that aliasing for Channel0 is disabled.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 31		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.2 AdcChannel1Alias

Table 68 Specification for AdcChannel1Alias

Name	`AdcChannel1Alias`		
Description	The parameter configures the alias for Channel1. The hardware supports maximum of eight channels (0 to 7) for primary hardware groups and sixteen channels (0 to 15) for secondary hardware groups. The channel numbers upto 31 can be configured for diagnostics use cases. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: A value of 1 here indicates that aliasing for Channel1 is disabled.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef

(table continues...)

Table 68 (continued) Specification for AdcChannel1Alias			
Range	0 - 31		
Default value	1		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.3 AdcEmuxChGroup

Table 69 Specification for AdcEmuxChGroup			
Name	`AdcEmuxChGroup`		
Description	The parameter determines whether the group is a EMUX feature support group or not The default value of this parameter is set to FALSE to minimize the executable code size. *Note: AdcGroup enabled with EMUX feature should contain only one analog channel for the EMUX interface.*		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcEmuxGroupInterface1, AdcEmuxGroupInterface0, AdcEmuxEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.4 AdcEmuxStartSelection

Table 70 Specification for AdcEmuxStartSelection	
Name	`AdcEmuxStartSelection`

(table continues...)

Table 70 (continued) Specification for AdcEmuxStartSelection

Description	The parameter configures the start selection channel of the EMUX. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: A value of start selection channel should be always equal to the number of maximum EMUX channels configured.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 7		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcEmuxChGroup		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.5 AdcGroupAccessMode

Table 71 Specification for AdcGroupAccessMode

Name	`AdcGroupAccessMode`		
Description	The parameter determines whether the group is a single access group or a streaming access group. *Note: ADC_ACCESS_MODE_STREAMING is not allowed for One-Shot software triggered groups (AdcGroupTriggSrc = ADC_TRIGG_SRC_SW and AdcGroupConvMode = ADC_CONV_MODE_ONESHOT).* *Note: The AdcGroupAccessMode parameter is editable only when the AdcResultHandlingImplementation parameter is ADC_INTERRUPT_MODE_RESULT_HANDLING.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_ACCESS_MODE_SINGLE: Single value access mode ADC_ACCESS_MODE_STREAMING: Streaming access mode		
Default value	ADC_ACCESS_MODE_SINGLE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcGroupTriggSrc, AdcResultHandlingImplementation, AdcGroupConversionMode		

(table continues...)

Table 71 (continued) Specification for AdcGroupAccessMode	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.7.6 AdcGroupConversionMode

Table 72 Specification for AdcGroupConversionMode

Name	`AdcGroupConversionMode`		
Description	The parameter determines whether a group is converting in single-shot mode or continuous conversion mode.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_CONV_MODE_CONTINUOUS: Conversions of an ADC channel group are performed continuously after a software API call (start). The conversions themselves are running automatically (no additional software or hardware trigger needed). ADC_CONV_MODE_ONESHOT: The conversion of an ADC channel group is performed once after a trigger.		
Default value	ADC_CONV_MODE_ONESHOT		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.7 AdcGroupDefinition

Table 73 Specification for AdcGroupDefinition

Name	`AdcGroupDefinition`

(table continues...)

Table 73 (continued) Specification for AdcGroupDefinition

Description	The parameter defines the assignment of AdcChannel to an AdcGroup. Therefore, multiple nodes of AdcGroupDefinition builds a list of channels for the AdcGroup. The maximum number of channels in a sequence is eight for primary hardware groups and sixteen for secondary hardware groups. The upper multiplicity is added based on the queue depth as per EVADC hardware support. *Note: If the referred channels do not belong to the same AdcHwUnitId, an error message is raised.* *Note: If this container contains a channel with the AdcChannelLimitCheck parameter set to TRUE, the number of channels is restricted to 1.* *Note: If the same analog channel is referred to more than once in the same group definition, an error message is raised.* *Note: If the AdcEmuxChGroup parameter is set to TRUE, then the number of channels in this parameter are restricted to 1.*		
Multiplicity	1..16	**Type**	EcucReferenceDef
Range	Reference to Node: AdcChannel		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcEmuxChGroup, AdcChannelLimitCheck, AdcHwUnitId		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.8 AdcGroupEcucPartitionRef

Table 74 Specification for AdcGroupEcucPartitionRef

Name	`AdcGroupEcucPartitionRef`		
Description	The parameter maps an ADC channel group to zero or multiple ECUC partitions to limit the access to this channel group. The ECUC partitions referenced are a subset of the ECUC partitions where the ADC driver is mapped to. *Note: The parameter support is added only for AUTOSAR schema compliance. This parameter is not used in code generation logic, hence this parameter is made editable false.*		
Multiplicity	0..*	**Type**	EcucReferenceDef
Range	Reference to Node: EcucPartition		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE

(table continues...)

Table 74 (continued) Specification for AdcGroupEcucPartitionRef

Value configuration class	Pre-Compile	**Multiplicity configuration class**	Pre-Compile
Origin	AUTOSAR_ECUC	**Scope**	ECU
Dependency	-		
Autosar Version	Applicable for Autosar version 4.4.0.		

1.3.1.7.9 AdcGroupId

Table 75 Specification for AdcGroupId

Name	`AdcGroupId`		
Description	The parameter defines the numeric ID of the ADC channel group. The maximum number of groups are limited to 32 in each ADC hardware unit. The numeric ID of the group is auto-generated by the configuration tool depending on the hardware unit. The symbolic name of this parameter is to be used while calling the APIs. The symbolic name of each AdcChannel group is a macro defined with the numeric ID. Group ID per hardware unit: 0 to 31 for HWUNIT_ADC0 32 to 63 for HWUNIT_ADC1 64 to 95 for HWUNIT_ADC2 96 to 127 for HWUNIT_ADC3 128 to 159 for HWUNIT_ADC4 160 to 191 for HWUNIT_ADC5 192 to 223 for HWUNIT_ADC6 224 to 255 for HWUNIT_ADC7 256 to 287 for HWUNIT_ADC8 288 to 319 for HWUNIT_ADC9 320 to 351 for HWUNIT_ADC10 352 to 383 for HWUNIT_ADC11 Here, the bits 0 to 4 determine the Group ID and the bits 5 to 8 determine the ADC hardware unit. The lowest group ID for a HW group is selected as the default value.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 383		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-

(table continues...)

Table 75 (continued) Specification for AdcGroupId

Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcHwUnitId		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.10 AdcGroupPriority

Table 76 Specification for AdcGroupPriority

Name	`AdcGroupPriority`		
Description	The parameter configures the priority level of the AdcGroup. The default value of this parameter is set to the lowest priority. Here, 0 is the lowest priority and 255 is the highest. *Note: This node exists only when AdcPriorityImplementation is not set to ADC_PRIORITY_NONE.*		
Multiplicity	0..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcPriorityImplementation		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.11 AdcGroupReplacement

Table 77 Specification for AdcGroupReplacement

Name	`AdcGroupReplacement`		
Description	The parameter provides the group replacement mechanism on AdcChannel group level if a group conversion is interrupted by a group which has a higher priority. This parameter is not editable since only the abort-restart mechanism is supported by the ADC driver.		
Multiplicity	0..1	**Type**	EcucEnumerationParamDef
Range	ADC_GROUP_REPL_ABORT_RESTART: Abort or restart mechanism is used on the group level if a group is interrupted by a group with a higher priority. The complete conversion round of the interrupted group (all group channels) is restarted after the higher priority group conversion is finished. If the group is configured in streaming access mode, only the results of the interrupted conversion round are discarded. Results of the previous conversion rounds that are already written to the result buffer are not affected.		

(table continues...)

Table 77 (continued) Specification for AdcGroupReplacement

Default value	ADC_GROUP_REPL_ABORT_RESTART		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.12 AdcGroupTriggSrc

Table 78 Specification for AdcGroupTriggSrc

Name	`AdcGroupTriggSrc`		
Description	The parameter defines the type of source event that starts a group conversion. Group conversion is started by a software API call for groups configured as ADC_TRIGG_SRC_SW and by a hardware event for groups configured as ADC_TRIGG_SRC_HW. *Note: The AdcGroupTriggSrc parameter value is allowed to be ADC_TRIGG_SRC_SW only when the AdcEnableStartStopGroupApi parameter is set to TRUE.* *Note: The AdcGroupTriggSrc parameter value is allowed to be ADC_TRIGG_SRC_HW only when the AdcHwTriggerApi parameter is set to TRUE.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_TRIGG_SRC_HW: Hardware triggered (GTM/ERU/CCU6) group. ADC_TRIGG_SRC_SW: Software triggered group.		
Default value	ADC_TRIGG_SRC_SW		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcHwTriggerApi, AdcEnableStartStopGroupApi		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.13 AdcHwExtGateSelect

Table 79 Specification for AdcHwExtGateSelect

Name	`AdcHwExtGateSelect`

(table continues...)

Table 79 (continued) Specification for AdcHwExtGateSelect

Description	The parameter configures the hardware gating source for the group.The range of values is dependent on the device. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: The AdcHwExtGateSelect parameter is editable for hardware triggered groups (AdcGroupTriggSrc = ADC_TRIG_SRC_HW)* *Note: The AdcHwExtGateSelect parameter shall be ADC_GATE_NONE for software triggered groups (AdcGroupTriggSrc = ADC_TRIG_SRC_SW)* *Note: The AdcHwExtGateSelect parameter shall be CCU6 when trigger is through a GATE input (AdcHwExtTrigSelect = ADC_TRIG_15_GxREQTRP_GxREQGTySEL).*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_GATE_NONE: Hardware gate signal is not selected ADC_GATE_[x]_[y]: Hardware gate signal selected. [x] and [y] depends on the device		
Default value	ADC_GATE_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcGroupTriggSrc, AdcHwExtTrigSelect		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.14 AdcHwExtTrigSelect

Table 80 Specification for AdcHwExtTrigSelect

Name	`AdcHwExtTrigSelect`		
Description	The parameter configures the hardware trigger source for the group.The range of values is dependent on the device. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: The AdcHwExtTrigSelect parameter is editable for hardware triggered groups (AdcGroupTriggSrc = ADC_TRIG_SRC_HW).* *Note: The AdcHwExtTrigSelect parameter shall be ADC_TRIG_NONE for software triggered groups (AdcGroupTriggSrc = ADC_TRIG_SRC_SW).* *Note: The AdcHwExtTrigSelect parameter shall not be ADC_TRIG_NONE for hardware triggered groups (AdcGroupTriggSrc = ADC_TRIG_SRC_HW).*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 80 (continued) Specification for AdcHwExtTrigSelect			
Range	ADC_TRIG_NONE: Hardware trigger signal is not selected. ADC_TRIG_[x]_[y]: Hardware trigger signal is selected. [x] and [y] depends on the device.		
Default value	ADC_TRIG_NONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcGroupTriggSrc		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.15 AdcHwGateSignal

Table 81 Specification for AdcHwGateSignal			
Name	`AdcHwGateSignal`		
Description	The parameter configures the hardware gating signal level. *Note: The AdcHwGateSignal parameter is editable for hardware triggered groups (AdcGroupTriggSrc = ADC_TRIG_SRC_HW).* *Note: The AdcHwGateSignal parameter is editable when the AdcHwExtGateSelect parameter is not equal to ADC_GATE_NONE and AdcHwExtTrigSelect is not equal to ADC_TRIG_15_GxREQTRP_GxREQGTySEL.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_GATE_LVL_HIGH: Gate is enabled when a rising edge is detected and the signal remains high. ADC_GATE_LVL_LOW: Gate is enabled when a falling edge is detected and the signal remains low.		
Default value	ADC_GATE_LVL_HIGH		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcGroupTriggSrc, AdcHwExtTrigSelect, AdcHwExtGateSelect		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.16 AdcHwTrigSignal

Table 82 Specification for AdcHwTrigSignal

Name	AdcHwTrigSignal		
Description	The parameter determines the edge of the external request signal on which the driver starts the conversion sequence. *Note: The AdcHwTrigSignal parameter is editable and is applicable when AdcGroupTriggSrc is configured as ADC_TRIGG_SRC_HW.* *Note: The AdcHwTrigSignal parameter shall be ADC_HW_TRIG_RISING_EDGE,when the AdcHwExtTrigSelect parameter is GTM and GTM TOM or ATOM timer configuration for trigger signal is inside the ADC driver. The driver supports all the types of HW Trigger Signal(rising,falling and both) when the GTM TOM or ATOM configuration for trigger signal is outside the ADC driver.* *Note: The AdcHwTrigSignal parameter shall be ADC_HW_TRIG_RISING_EDGE, when AdcHwExtTrigSelect is through GATE_PIN and AdcHwExtGateSelect is GTM or CCU6 (Timer).*		
Multiplicity	0..1	**Type**	EcucEnumerationParamDef
Range	ADC_HW_TRIG_BOTH_EDGES: Group is triggered on both the edges of the trigger signal. ADC_HW_TRIG_FALLING_EDGE: Group is triggered on the falling edge of the trigger signal ADC_HW_TRIG_RISING_EDGE: Group is triggered on rising edge of the trigger signal		
Default value	ADC_HW_TRIG_RISING_EDGE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcHwExtGateSelect, AdcHwExtTrigSelect, AdcGroupTriggSrc		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.17 AdcHwTrigTimer

Table 83 Specification for AdcHwTrigTimer

Name	AdcHwTrigTimer		
Description	Parameter specifies the reload value of the ADC module embedded timer (REQTM). For cyclic triggering of the ADC, GTM, or CCU6 trigger is used. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: The maximum value of this parameter is specified based on the maximum value supported by the EVADC hardware trigger timer.*		
Multiplicity	0..1	**Type**	EcucIntegerParamDef
Range	0 - 1023		
Default value	0		

(table continues...)

Table 83 (continued) Specification for AdcHwTrigTimer

Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.18 AdcNotification

Table 84 Specification for AdcNotification

Name	`AdcNotification`		
Description	The parameter is used by the ADC driver to invoke the user-defined function whenever the conversion of all the channels of a group is completed. The parameter can be configured as a name or an address (numeric value) of the notification function. If configured as NULL_PTR, notification is not issued by the ADC driver. *Note: The parameter is editable only when AdcGrpNotifCapability is set to TRUE.* *Note: By default, the notification parameter will be NULL_PTR , to remove the dependency from the user defined functions.* *Note: The ADC driver does not validate the configured function name or address for correctness and hence the responsibility falls on the user.*		
Multiplicity	0..1	**Type**	EcucFunctionNameDef
Range	String		
Default value	NULL_PTR		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcGrpNotifCapability		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.19 AdcResRegDefinition

Table 85 Specification for AdcResRegDefinition

Name	`AdcResRegDefinition`

(table continues...)

Table 85 (continued) Specification for AdcResRegDefinition

Description	Parameter configures the result register in which to store the ADC conversion results. Value of the parameter should be in ascending order. Therefore, the first configured channel should have the lowest result register index. So, a default value set to 0 . Possible set of result registers configured for a group having five channels may be as: 4, 5, 6, 7, 8 or 0, 1, 2, 3, 4 or 11, 12 ,13, 14, 15. *Note: If the AdcResRegDefinition parameter is not configured in contiguous-ascending order, an error message is raised.* *Note: The number of nodes should be the same as the number of nodes configured for the corresponding AdcGroupDefinition container.* *Note: An error message will be raised if the AdcResRegDefinition parameter is not configured to 0 when the AdcEmuxChGroup is set to TRUE.*		
Multiplicity	1..16	**Type**	EcucIntegerParamDef
Range	0 - 15		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	IFX	**Scope**	LOCAL
Dependency	AdcEmuxChGroup, AdcGroupDefinition		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.20 AdcStreamingBufferMode

Table 86 Specification for AdcStreamingBufferMode

Name	`AdcStreamingBufferMode`

(table continues...)

Table 86 (continued) Specification for AdcStreamingBufferMode

Description	The parameter configures the streaming buffer as a linear buffer or as a circular buffer. *Note: The AdcStreamingBufferMode parameter cannot be ADC_STREAM_BUFFER_CIRCULAR for software triggered One-Shot conversion (AdcGroupTriggSrc = 'ADC_TRIGG_SRC_SW' and AdcGroupConversionMode = 'ADC_CONV_MODE_ONESHOT' and AdcGroupAccessMode = 'ADC_ACCESS_MODE_SINGLE' and AdcResultHandlingImplementation = 'ADC_INTERRUPT_MODE_RESULT_HANDLING')* *Note: The AdcStreamingBufferMode parameter cannot be ADC_STREAM_BUFFER_LINEAR for software triggered continuous single access groups (AdcGroupTriggSrc = 'ADC_TRIGG_SRC_SW', AdcGroupConversionMode = 'ADC_CONV_MODE_CONTINUOUS' and AdcGroupAccessMode = 'ADC_ACCESS_MODE_SINGLE' and AdcResultHandlingImplementation = 'ADC_INTERRUPT_MODE_RESULT_HANDLING')* *Note: The AdcStreamingBufferMode parameter cannot be ADC_STREAM_BUFFER_LINEAR for hardware triggered single access groups (AdcGroupTriggSrc = 'ADC_TRIGG_SRC_HW' and AdcGroupAccessMode = 'ADC_ACCESS_MODE_SINGLE' and AdcResultHandlingImplementation = 'ADC_INTERRUPT_MODE_RESULT_HANDLING')* *Note: The AdcStreamingBufferMode parameter is editable only when the AdcResultHandlingImplementation parameter is ADC_INTERRUPT_MODE_RESULT_HANDLING.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_STREAM_BUFFER_CIRCULAR: The ADC driver continues the conversion even if the stream buffer is full (number of samples reached) by wrapping around the stream buffer itself. ADC_STREAM_BUFFER_LINEAR: The ADC driver stops the conversion as soon as the stream buffer is full (number of samples reached).		
Default value	ADC_STREAM_BUFFER_LINEAR		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcGroupAccessMode, AdcGroupTriggSrc, AdcResultHandlingImplementation, AdcGroupConversionMode		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.7.21 AdcStreamingNumSamples

Table 87 Specification for AdcStreamingNumSamples

Name	`AdcStreamingNumSamples`

(table continues...)

Table 87 (continued) Specification for AdcStreamingNumSamples

Description	The parameter configures the number of ADC values to be acquired per channel in streaming access mode. *Note: The AdcStreamingNumSamples parameter should be greater than 1 for Streaming access groups (AdcGroupAccessMode = ADC_ACCESS_MODE_STREAMING and AdcResultHandlingImplementation = ADC_INTERRUPT_MODE_RESULT_HANDLING)* *Note: The AdcStreamingNumSamples parameter should be equal to 1 for single access groups (AdcGroupAccessMode = ADC_ACCESS_MODE_SINGLE and AdcResultHandlingImplementation = ADC_INTERRUPT_MODE_RESULT_HANDLING)* *Note: The AdcStreamingNumSamples parameter is editable only when the AdcResultHandlingImplementation parameter is ADC_INTERRUPT_MODE_RESULT_HANDLING.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - 255		
Default value	1		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	AdcGroupAccessMode, AdcResultHandlingImplementation		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8 Container: AdcHwUnit

Container consists of the configuration parameter and containers for ADC hardware group.

The upper multiplicity of this parameter is added based on the total number of EVADC hardware units supported by the hardware.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.8.1 AdcAnalogClockSyncDelay

Table 88 Specification for AdcAnalogClockSyncDelay

Name	`AdcAnalogClockSyncDelay`		
Description	The parameter configures analog clock synchronization delay.The delay should be less than or equal to AdcPrescale-1(DIVA). The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 7		
Default value	0		

(table continues...)

Table 88 (continued) Specification for AdcAnalogClockSyncDelay

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcPrescale		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.2 AdcCalibrationSampleTime

Table 89 Specification for AdcCalibrationSampleTime

Name	`AdcCalibrationSampleTime`		
Description	The parameter configures the calibration sample time. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_CAL_TIME_2_TIMES_TADCI: Calibration sample time = 2 x tADCI ADC_CAL_TIME_4_TIMES_TADCI: Calibration sample time = 4 x tADCI ADC_CAL_TIME_6_TIMES_TADCI: Calibration sample time = 6 x tADCI ADC_CAL_TIME_8_TIMES_TADCI: Calibration sample time = 8 x tADCI		
Default value	ADC_CAL_TIME_2_TIMES_TADCI		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.3 AdcClockSource

Table 90 Specification for AdcClockSource

Name	`AdcClockSource`
Description	The parameter defines the ADC module specific clock source. The clock to ADC module is controlled through AdcSystemClock. Therefore, this parameter is unused. The default value of this parameter is set to the reset value of the corresponding SFR.

(table continues...)

Table 90 (continued) Specification for AdcClockSource

Multiplicity	0..1	**Type**	EcucEnumerationParamDef
Range	ADC_SYSTEM_CLK: System clock for the EVADC.		
Default value	ADC_SYSTEM_CLK		
Post-build variant value	TRUE	**Post-build variant multiplicity**	TRUE
Value configuration class	Post-Build	**Multiplicity configuration class**	Post-Build
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.4 AdcHwUnitId

Table 91 Specification for AdcHwUnitId

Name	`AdcHwUnitId`		
Description	The parameter is used to arrange the configuration parameters in the configuration structure according to ADC hardware unit ID. This parameter should have a unique value across AdcHwUnit container nodes for same config set, that is, a hardware unit can be configured only once in a config set. The lowest index ADC hardware group available is selected as the default value for this parameter.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_HWUNIT[x]: [x] = list of available Hardware units.		
Default value	ADC_HWUNIT[x]		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.5 AdcIdlePrechargeEnable

Table 92 Specification for AdcIdlePrechargeEnable

Name	`AdcIdlePrechargeEnable`

(table continues...)

Table 92 (continued) Specification for AdcIdlePrechargeEnable

Description	The parameter determines whether precharging of the sampling capacitor is enabled or not. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.6 AdcInputBufferEnable

Table 93 Specification for AdcInputBufferEnable

Name	`AdcInputBufferEnable`		
Description	The parameter determines whether input buffering is enabled or not. If enabled, buffering time is selected through AIPC(AdcChPreChargeClkCycles). The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.7 AdcMSBDoubleClkEnable

Table 94 Specification for AdcMSBDoubleClkEnable

Name	AdcMSBDoubleClkEnable		
Description	The parameter configures the number of clock cycles for MSB conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.8 AdcPostCalibrationDisable

Table 95 Specification for AdcPostCalibrationDisable

Name	AdcPostCalibrationDisable		
Description	The parameter configures whether post-calibration is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.9 AdcPrechargeReference

Table 96 Specification for AdcPrechargeReference

Name	AdcPrechargeReference		
Description	The parameter determines whether VDDM/VSSM is used for precharging or not. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_VDD_VSM_NOT_USED: VAREF/VAGND for the conversion. ADC_VDD_VSM_USED: VDDM/VSSM for precharging and VAREF/VAGND for the final adjustment during a conversion.		
Default value	ADC_VDD_VSM_USED		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.10 AdcPrescale

Table 97 Specification for AdcPrescale

Name	AdcPrescale		
Description	The parameter configures the divider for generating the analog internal clock (fADCDI) from fADC. The lower multiplicity of this parameter is 1, because the divider for generating analog frequency is mandatory for performing the conversion. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	2 - 32		
Default value	4		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.11 AdcReferencePrechargePhases

Table 98 Specification for AdcReferencePrechargePhases

Name	AdcReferencePrechargePhases		
Description	The parameter configures the number of precharge clock phases for the reference input. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_PRECHARGE_PHASE_1: Precharge the reference input for one clock phase ADC_PRECHARGE_PHASE_2: Precharge the reference input for two clock phases		
Default value	ADC_PRECHARGE_PHASE_1		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.12 AdcRequestSource0ConvMode

Table 99 Specification for AdcRequestSource0ConvMode

Name	AdcRequestSource0ConvMode		
Description	The parameter configures the conversion start mode for a request source. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: The configuration of the conversion start mode for request sources is editable and applicable only when AdcPriorityImplementation is not ADC_PRIORITY_NONE.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_CANCEL_INJECT_REPEAT_MODE: Request source can cancel the ongoing conversion of other sources. ADC_WAIT_FOR_START_MODE: Request source waits for the completion of the current conversion from other source.		
Default value	ADC_WAIT_FOR_START_MODE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL

(table continues...)

Table 99	(continued) Specification for AdcRequestSource0ConvMode
Dependency	AdcPriorityImplementation
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.8.13 AdcRequestSource1ConvMode

Table 100	Specification for AdcRequestSource1ConvMode		
Name	`AdcRequestSource1ConvMode`		
Description	The parameter configures the conversion start mode for a request source. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: The configuration of the conversion start mode for request sources is editable and applicable only when AdcPriorityImplementation is not ADC_PRIORITY_NONE.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_CANCEL_INJECT_REPEAT_MODE: Request source can cancel the ongoing conversion of other sources. ADC_WAIT_FOR_START_MODE: Request source waits for the completion of current conversion from other source.		
Default value	ADC_WAIT_FOR_START_MODE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcPriorityImplementation		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.14 AdcRequestSource2ConvMode

Table 101	Specification for AdcRequestSource2ConvMode		
Name	`AdcRequestSource2ConvMode`		
Description	The parameter configures the conversion start mode for a request source. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: The configuration of the conversion start mode for request sources is editable and applicable only when AdcPriorityImplementation is not ADC_PRIORITY_NONE.*		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 101 (continued) Specification for AdcRequestSource2ConvMode			
Range	ADC_CANCEL_INJECT_REPEAT_MODE: Request source can cancel the ongoing conversion of other sources. ADC_WAIT_FOR_START_MODE: Request source waits for the completion of current conversion from other source.		
Default value	ADC_WAIT_FOR_START_MODE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcPriorityImplementation		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.15 AdcSampleSyncEnable

Table 102 Specification for AdcSampleSyncEnable			
Name	`AdcSampleSyncEnable`		
Description	The parameter determines whether the ADC Sample Time Synchronization is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcSyncClockDisable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.8.16 AdcSyncConvMode

Table 103 Specification for AdcSyncConvMode	
Name	`AdcSyncConvMode`

(table continues...)

Table 103 (continued) Specification for AdcSyncConvMode

Description	The parameter configures whether the hardware groups operate in standalone or synchronous mode. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_STAND_ALONE: Hardware group operates in stand-alone mode. ADC_SYNC_MASTER: Hardware group operates in master mode. ADC_SYNC_SLAVE: Hardware group operates in slave mode.		
Default value	ADC_STAND_ALONE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcHwUnitId, AdcEmuxGroupInterface1, AdcEmuxGroupInterface0, AdcEmuxEnable, AdcSyncConvEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9 Container: AdcHwUnitInputClass

Container defines the parameters for hardware unit input classes.

The multiplicity of this parameter is added based on the number of hardware unit classes supported by the hardware.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.9.1 AdcChConvMode

Table 104 Specification for AdcChConvMode

Name	AdcChConvMode		
Description	The parameter configures the noise reduction level for standard conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_NOISE_REDUCTION_STEPS_0: Standard conversion ADC_NOISE_REDUCTION_STEPS_1: One additional conversion step for noise reduction ADC_NOISE_REDUCTION_STEPS_3: Three additional conversion steps for noise reduction ADC_NOISE_REDUCTION_STEPS_7: Seven additional conversion steps for noise reduction		
Default value	ADC_NOISE_REDUCTION_STEPS_0		

(table continues...)

Table 104 (continued) Specification for AdcChConvMode

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.2 AdcChPreChargeClkCycles

Table 105 Specification for AdcChPreChargeClkCycles

Name	`AdcChPreChargeClkCycles`		
Description	The parameter configures the number of analog input precharge clock cycles for standard conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_INPUT_PRECHARGE_CYCLES_0: No precharge ADC_INPUT_PRECHARGE_CYCLES_16: Precharge for sixteen clock cycles ADC_INPUT_PRECHARGE_CYCLES_32: Precharge for thirty two clock cycles ADC_INPUT_PRECHARGE_CYCLES_8: Precharge for eight clock cycles		
Default value	ADC_INPUT_PRECHARGE_CYCLES_0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.3 AdcChSESPSEnable

Table 106 Specification for AdcChSESPSEnable

Name	`AdcChSESPSEnable`
Description	The parameter configures whether the Spread Early Sample Point for the standard conversions is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.

(table continues...)

Table 106 (continued) Specification for AdcChSESPSEnable

Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.4 AdcChSampleTime

Table 107 Specification for AdcChSampleTime

Name	`AdcChSampleTime`		
Description	The parameter configures the number of clock cycles to be added to the minimum sample phase of two sample clock cycles. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 31		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.5 AdcEmuxChConvMode

Table 108 Specification for AdcEmuxChConvMode

Name	`AdcEmuxChConvMode`
Description	The parameter configures the noise reduction level for the EMUX conversions. The default value of this parameter is set to the reset value of the corresponding SFR.

(table continues...)

Table 108 (continued) Specification for AdcEmuxChConvMode

Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_NOISE_REDUCTION_STEPS_0: Standard conversion ADC_NOISE_REDUCTION_STEPS_1: One additional conversion step for noise reduction ADC_NOISE_REDUCTION_STEPS_3: Three additional conversion steps for noise reduction ADC_NOISE_REDUCTION_STEPS_7: Seven additional conversion steps for noise reduction		
Default value	ADC_NOISE_REDUCTION_STEPS_0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcHwUnitId, AdcEmuxGroupInterface0, AdcEmuxGroupInterface1, AdcEmuxEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.6 AdcEmuxChPreChargeClkCycles

Table 109 Specification for AdcEmuxChPreChargeClkCycles

Name	`AdcEmuxChPreChargeClkCycles`		
Description	The parameter configures the number of analog input precharge clock cycles for the EMUX conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ADC_INPUT_PRECHARGE_CYCLES_0: No precharge ADC_INPUT_PRECHARGE_CYCLES_16: Precharge for sixteen clock cycles ADC_INPUT_PRECHARGE_CYCLES_32: Precharge for thirty two clock cycles ADC_INPUT_PRECHARGE_CYCLES_8: Precharge for eight clock cycles		
Default value	ADC_INPUT_PRECHARGE_CYCLES_0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcHwUnitId, AdcEmuxGroupInterface1, AdcEmuxGroupInterface0, AdcEmuxEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.7 AdcEmuxChSESPSEnable

Table 110 Specification for AdcEmuxChSESPSEnable

Name	AdcEmuxChSESPSEnable		
Description	The parameter configures whether the Spread Early Sample Point for the EMUX conversions is enabled or disabled. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcHwUnitId, AdcEmuxGroupInterface0, AdcEmuxGroupInterface1, AdcEmuxEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.9.8 AdcEmuxChSampleTime

Table 111 Specification for AdcEmuxChSampleTime

Name	AdcEmuxChSampleTime		
Description	The parameter configures the number of clock cycles to be added to the minimum sample phase of two sample clock cycles for the EMUX conversions. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 31		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcHwUnitId, AdcEmuxGroupInterface1, AdcEmuxGroupInterface0, AdcEmuxEnable		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10 Container: AdcPowerStateConfig

Container contains the configuration parameters related to the power states supported.

The upper multiplicity of this container is added based on the total number of power modes supported by the EVADC hardware unit.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.10.1 AdcPowerState

Table 112 Specification for AdcPowerState

Name	`AdcPowerState`		
Description	The parameter configures the availability of the different power state supported by the EVADC hardware unit. The available power modes are as follows: 0 - Normal operation (permanently on) 1 - Fast standby mode 2 - Slow standby mode 3 - Analog converter off *Note: Each power state must be selected only once across all AdcPowerStateConfig instances. An error is generated if the same value is selected twice. The normal operation (full power) is selected as the default value.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 3		
Default value	0		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.10.2 AdcPowerStateReadyCbkRef

Table 113 Specification for AdcPowerStateReadyCbkRef

Name	`AdcPowerStateReadyCbkRef`		
Description	The parameter determines the callback function for the power mode. *Note: Since power state transitions are always synchronous, this parameter is disabled in tresos.*		
Multiplicity	1..1	**Type**	EcucFunctionNameDef

(table continues...)

Table 113 (continued) Specification for AdcPowerStateReadyCbkRef

Range	String		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.11 Container: AdcPublishedInformation

Container contains the published information of the ADC driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.11.1 AdcChannelValueSigned

Table 114 Specification for AdcChannelValueSigned

Name	`AdcChannelValueSigned`		
Description	The parameter provides the information whether the result value of the ADC driver has sign information (TRUE) or not (FALSE).		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.11.2 AdcGroupFirstChannelFixed

Table 115 Specification for AdcGroupFirstChannelFixed

Name	`AdcGroupFirstChannelFixed`		
Description	The parameter provides the 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 hardware unit.		
Multiplicity	1..1	**Type**	EcucBooleanParamDef
Range	TRUE FALSE		
Default value	FALSE		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.11.3 AdcMaxChannelResolution

Table 116 Specification for AdcMaxChannelResolution

Name	`AdcMaxChannelResolution`		
Description	The parameter provides the information of the maximum channel resolution in bits.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	1 - 63		
Default value	12		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	AUTOSAR_ECUC	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12 Container: CommonPublishedInformation

Container contains the common published information of the ADC driver.

Post-Build Variant Multiplicity: -

Multiplicity Configuration Class: -

1.3.1.12.1 ArMajorVersion

Table 117 Specification for ArMajorVersion

Name	ArMajorVersion		
Description	The parameter provides the major version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	4		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.2 ArMinorVersion

Table 118 Specification for ArMinorVersion

Name	ArMinorVersion		
Description	The parameter provides the minor version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per the selected Autosar version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.3 ArPatchVersion

Table 119 Specification for ArPatchVersion

Name	ArPatchVersion		
Description	The parameter provides the patch version of the AUTOSAR specification.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef

(table continues...)

Table 119 (continued) Specification for ArPatchVersion

Range	0 - 255		
Default value	As per the selected Autosar version		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.4 ModuleId

Table 120 Specification for ModuleId

Name	`ModuleId`		
Description	The parameter provides the Module Id of the ADC driver.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	123		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.5 Release

Table 121 Specification for Release

Name	`Release`		
Description	The parameter specifies the derivate for which the configuration project is created.		
Multiplicity	1..1	**Type**	EcucStringParamDef
Range	String		
Default value	As per hardware derivative		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-

(table continues...)

Table 121 (continued) Specification for Release

Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.6 SwMajorVersion

Table 122 Specification for SwMajorVersion

Name	`SwMajorVersion`		
Description	The parameter specifies the major version of the driver software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.7 SwMinorVersion

Table 123 Specification for SwMinorVersion

Name	`SwMinorVersion`		
Description	The parameter specifies the minor version of the driver software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		

(table continues...)

Table 123 (continued) Specification for SwMinorVersion

Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.12.8 SwPatchVersion

Table 124 Specification for SwPatchVersion

Name	SwPatchVersion		
Description	The parameter specifies the patch version of the driver software.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 255		
Default value	As per Driver		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.12.9 VendorId

Table 125 Specification for VendorId

Name	VendorId		
Description	The parameter specifies the vendor ID for Infineon.		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65535		
Default value	17		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Published-Information	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13 Container: EruGatingConfig

Container contains the ERU configuration for the gating signal.

Note: The EruGatingConfig container is available only for the hardware triggered groups and when the AdcHwExtGateSelect parameter is an ERU signal.

Note: If the group is configured as HW_ERU_GATE and the EruGatingConfig container is missing, an error message is raised.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

1.3.1.13.1 AdcEruErsInputPin

Table 126 Specification for AdcEruErsInputPin

Name	`AdcEruErsInputPin`		
Description	The parameter determines the input pin for the selected ERS.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ERS_REQ_[x]: Range of literals varies based on target device.		
Default value	ERS_REQ_[x]		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.13.2 AdcEruErsRef

Table 127 Specification for AdcEruErsRef

Name	`AdcEruErsRef`		
Description	The parameter is a reference to the ERU container in the MCU. It lists all the ERU-ERS channels available. If the ERU-ERS channel referred to, is not marked as to be used by ADC driver in the MCU driver, an error message is raised.		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: McuEruChannelInputLineConf		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcHwUnitId		

(table continues...)

Table 127 (continued) Specification for AdcEruErsRef

Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.1.13.3 AdcEruOguRef

Table 128 Specification for AdcEruOguRef

Name	`AdcEruOguRef`		
Description	The parameter is a reference to the ERU container in the MCU. It lists all the ERU-ERS channels available. If the ERU-OGU channel referred to is not marked as to be used by the ADC driver in MCU driver, an error message is raised.		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: McuEruChannelOutputUnitConf		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcHwUnitId		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.14 Container: EruTriggerConfig

Container contains the ERU configuration for the trigger signal.

Note: The EruTriggerConfig container is available only for hardware triggered groups and when the AdcHwExtTrigSelect parameter is an ERU signal.

Note: If the group is configured as HW_ERU_TRIG and the EruTriggerConfig container is missing, an error message is raised.

Post-Build Variant Multiplicity: TRUE

Multiplicity Configuration Class: Post-Build

1.3.1.14.1 AdcEruErsInputPin

Table 129 Specification for AdcEruErsInputPin

Name	`AdcEruErsInputPin`		
Description	The parameter determines the input pin for the selected ERS.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	ERS_REQ_[x]: Range of literals varies based on target device.		
Default value	ERS_REQ_[x]		

(table continues...)

Table 129 (continued) Specification for AdcEruErsInputPin

Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.14.2 AdcEruErsRef

Table 130 Specification for AdcEruErsRef

Name	`AdcEruErsRef`		
Description	The parameter is a reference to the ERU container in the MCU. It lists all the ERU-ERS channels available. If the ERU-ERS channel referred to, is not marked as to be used by the ADC driver in the MCU driver, an error message is raised.		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: McuEruChannelInputLineConf		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcHwUnitId		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.14.3 AdcEruOguRef

Table 131 Specification for AdcEruOguRef

Name	`AdcEruOguRef`		
Description	The parameter is a reference to the ERU container in the MCU. It lists all the ERU-ERS channels available. If the ERU-OGU channel referred to, is not marked as to be used by the ADC driver in the MCU driver, an error message is raised.		
Multiplicity	1..1	**Type**	EcucReferenceDef
Range	Reference to Node: McuEruChannelOutputUnitConf		
Default value	NULL		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

(table continues...)

Table 131 (continued) Specification for AdcEruOguRef

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	AdcHwUnitId		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.15 Container: GtmGatingTimerConfig

The GtmGatingTimerConfig container is available only for hardware triggered groups and when AdcHwExtgateSelect is a GTM signal. If the container is not configured for such a group, then the initialization of the GTM channel intended to be used for gate signal must be done outside of the ADC driver.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.15.1 GtmTimerCM0Ticks

Table 132 Specification for GtmTimerCM0Ticks

Name	`GtmTimerCM0Ticks`		
Description	The parameter specifies the period compare value (CM0) for the TOM/ATOM channel in ticks. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: For TOM channel, the upper limit of this parameter is restricted to 65535.* *Note: This parameter is editable only when the GtmTimerTimePeriod parameter is equal to 0 (which implies that the user has not entered the period directly in micro seconds).* *Note: An error message will be raised if value entered in this parameter is not in the specified range.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	GtmTimerTimePeriod, GtmTimerUsed		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.15.2 GtmTimerClockSelect

Table 133 Specification for GtmTimerClockSelect

Name	`GtmTimerClockSelect`		
Description	The parameter decides the clock source for the GTM timer. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef
Range	GTM_CONFIGURABLE_CLOCK_0: Configurable Clock 0 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_1: Configurable Clock 1 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_2: Configurable Clock 2 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_3: Configurable Clock 3 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_4: Configurable Clock 4 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_5: Configurable Clock 5 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_6: Configurable Clock 6 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_7: Configurable Clock 7 will be supplied to the Channel GTM_FIXED_CLOCK_0: Fixed Clock 0 will be supplied to the Channel GTM_FIXED_CLOCK_1: Fixed Clock 1 will be supplied to the Channel GTM_FIXED_CLOCK_2: Fixed Clock 2 will be supplied to the Channel GTM_FIXED_CLOCK_2: Fixed Clock 2 will be supplied to the Channel GTM_FIXED_CLOCK_3: Fixed Clock 3 will be supplied to the Channel GTM_FIXED_CLOCK_4: Fixed Clock 4 will be supplied to the Channel		
Default value	GTM_FIXED_CLOCK_0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	GtmTimerUsed		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.15.3 GtmTimerTimePeriod

Table 134 Specification for GtmTimerTimePeriod

Name	`GtmTimerTimePeriod`

(table continues...)

Table 134 (continued) Specification for GtmTimerTimePeriod

Description	The parameter defines the time period of GTM timer in micro seconds. A nominal time-interval is configured as the default value for time-based trigger. *Note: If the calculated number of GTM ticks are greater than 0xFFFF for TOM channels or greater than 0xFFFFFF for ATOM channels an error message is raised.* *Note: This parameter is editable only when the GtmTimerCM0Ticks parameter is equal to 0 (user has not entered the period match ticks directly).* *Note: An error message will be raised if value entered in this parameter is not in the specified range.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65536000		
Default value	20000		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	GtmTimerCM0Ticks		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.15.4 GtmTimerUsed

Table 135 Specification for GtmTimerUsed

Name	`GtmTimerUsed`		
Description	The parameter defines the GTM timer channel used. *Note: If the timer channel referred to, is not marked as to be used by the ADC driver in the MCU driver an error message is raised.*		
Multiplicity	1..1	**Type**	EcucChoiceReference Def
Range	Reference to Node: McuGtmAtomChannelAllocationConf, McuGtmTomChannelAllocationConf		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.16 Container: GtmTriggerTimerConfig

The GtmTriggerTimerConfig container is available only for hardware triggered groups and when the AdcHwExtTrigSelect parameter is a GTM signal. If the container is not configured for such a group, then the initialization of the GTM channel intended to be used for trigger signal must be done outside of the ADC driver.

Post-Build Variant Multiplicity: FALSE

Multiplicity Configuration Class: Pre-Compile

1.3.1.16.1 GtmTimerCM0Ticks

Table 136 Specification for GtmTimerCM0Ticks

Name	`GtmTimerCM0Ticks`		
Description	The parameter specifies the period compare value (CM0) for the TOM/ATOM channel in ticks. The default value of this parameter is set to the reset value of the corresponding SFR. *Note: For TOM channel, the upper limit of this parameter is restricted to 65535.* *Note: This parameter is editable only when the GtmTimerTimePeriod parameter is equal to 0 (which implies that the user has not entered the period directly in micro seconds).* *Note: An error message will be raised if value entered in this parameter is not in the specified range.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 16777215		
Default value	0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	GtmTimerTimePeriod, GtmTimerUsed		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.16.2 GtmTimerClockSelect

Table 137 Specification for GtmTimerClockSelect

Name	`GtmTimerClockSelect`		
Description	The parameter decides the clock source for the GTM timer. The default value of this parameter is set to the reset value of the corresponding SFR.		
Multiplicity	1..1	**Type**	EcucEnumerationParamDef

(table continues...)

Table 137 (continued) Specification for GtmTimerClockSelect

Range	GTM_CONFIGURABLE_CLOCK_0: Configurable Clock 0 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_1: Configurable Clock 1 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_2: Configurable Clock 2 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_3: Configurable Clock 3 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_4: Configurable Clock 4 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_5: Configurable Clock 5 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_6: Configurable Clock 6 will be supplied to the Channel GTM_CONFIGURABLE_CLOCK_7: Configurable Clock 7 will be supplied to the Channel. GTM_FIXED_CLOCK_0: Fixed Clock 0 will be supplied to the Channel GTM_FIXED_CLOCK_1: Fixed Clock 1 will be supplied to the Channel GTM_FIXED_CLOCK_2: Fixed Clock 2 will be supplied to the Channel GTM_FIXED_CLOCK_3: Fixed Clock 3 will be supplied to the Channel GTM_FIXED_CLOCK_4: Fixed Clock 4 will be supplied to the Channel		
Default value	GTM_FIXED_CLOCK_0		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-
Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	GtmTimerUsed		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.16.3 GtmTimerTimePeriod

Table 138 Specification for GtmTimerTimePeriod

Name	`GtmTimerTimePeriod`		
Description	The parameter defines the time period of GTM timer in micro seconds. A nominal time-interval is configured as the default value for time-based trigger. *Note: If the calculated number of GTM ticks are greater than 0xFFFF for TOM channels or greater than 0xFFFFFF for ATOM channels an error message is raised.* *Note: This parameter is editable only when the GtmTimerCM0Ticks parameter is equal to 0 (user has not entered the period match ticks directly).* *Note: An error message will be raised if value entered in this parameter is not in the specified range.*		
Multiplicity	1..1	**Type**	EcucIntegerParamDef
Range	0 - 65536000		
Default value	20000		
Post-build variant value	TRUE	**Post-build variant multiplicity**	-

(table continues...)

Table 138 (continued) Specification for GtmTimerTimePeriod

Value configuration class	Post-Build	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	GtmTimerCM0Ticks		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.1.16.4 GtmTimerUsed

Table 139 Specification for GtmTimerUsed

Name	`GtmTimerUsed`		
Description	The parameter defines the GTM timer channel used. *Note: If the timer channel referred to, is not marked as to be used by the ADC driver in the MCU driver an error message is raised.*		
Multiplicity	1..1	**Type**	EcucChoiceReference Def
Range	Reference to Node: McuGtmAtomChannelAllocationConf, McuGtmTomChannelAllocationConf		
Default value	NULL		
Post-build variant value	FALSE	**Post-build variant multiplicity**	-
Value configuration class	Pre-Compile	**Multiplicity configuration class**	-
Origin	IFX	**Scope**	LOCAL
Dependency	-		
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.		

1.3.2 Functions - Type definitions

This section lists all the data types of the ADC driver.

1.3.2.1 Adc_ChannelRangeSelectType

Table 140 Specification for Adc_ChannelRangeSelectType

Syntax	`Adc_ChannelRangeSelectType`	
Type	Enumeration	
File	`Adc.h`	
Range	0 - ADC_RANGE_UNDER_LOW	Range below low limit - low limit value included

(table continues...)

Table 140 (continued) Specification for Adc_ChannelRangeSelectType

	1 - ADC_RANGE_BETWEEN	Range between low limit and high limit - high limit value included
	2 - ADC_RANGE_OVER_HIGH	Range above high limit
	3 - ADC_RANGE_ALWAYS	Complete range - independent of channel limit settings
	4 - ADC_RANGE_NOT_UNDER_LOW	Range above low limit
	5 - ADC_RANGE_NOT_BETWEEN	Range above high limit or below low limit - low limit value included
	6 - ADC_RANGE_NOT_OVER_HIGH	Range below high limit - high limit value included
Description	Defines the type for which conversion values are taken into account related to the borders defined with the AdcChannelLowLimit and AdcChannelHighLimit parameter.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.2 Adc_ChannelType

Table 141 Specification for Adc_ChannelType

Syntax	`Adc_ChannelType`	
Type	uint8	
File	`Adc.h`	
Range	0 - 255	
Description	Defines the type for numeric ID of an ADC channel. The Adc_ChannelType is used for both logical and physical channel ID. The maximum number of logical channels per hardware group is 255 (ID 0 to ID 254, here ID 255 is used as invalid channel ID) and the maximum number of physical channels per hardware group is 16 (ID 0 to ID 15). Hence, the data width is selected based on the maximum channel ID.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.3 Adc_ConfigType

Table 142 Specification for Adc_ConfigType

Syntax	`Adc_ConfigType`	
Type	Structure	
File	`Adc.h`	
Range	--	The elements of the data structure are specific to the microcontroller.

(table continues...)

Table 142 (continued) Specification for Adc_ConfigType

Description	Defines the data structure containing the set of configuration parameters required for initializing the ADC driver and ADC hardware unit(s).
Source	AUTOSAR
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.2.4 Adc_ConversionTimeType

Table 143 Specification for Adc_ConversionTimeType

Syntax	`Adc_ConversionTimeType`	
Type	uint8	
File	`Adc.h`	
Range	--	Unused
Description	Defines the type for conversion time during which the sampled analog value is converted into its digital representation. Since this type is not used by the driver, the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.5 Adc_GroupAccessModeType

Table 144 Specification for Adc_GroupAccessModeType

Syntax	`Adc_GroupAccessModeType`	
Type	uint8	
File	`Adc.h`	
Range	0 - ADC_ACCESS_MODE_SINGLE	Single value access mode.
	1 - ADC_ACCESS_MODE_STREAMING	Streaming value access mode.
Description	Defines the type for configuring the access mode to group results. To reduce memory usage, the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.6 Adc_GroupConvModeType

Table 145 Specification for Adc_GroupConvModeType

Syntax	`Adc_GroupConvModeType`
Type	uint8
File	`Adc.h`

(table continues...)

Table 145 (continued) Specification for Adc_GroupConvModeType

Range	0 - ADC_CONV_MODE_ONESHOT	Exactly one conversion of each channel in an ADC channel group is performed after the configured trigger event. In case of group trigger source software, a One Shot conversion, once started, can be stopped by a software API call. In case of group trigger source hardware, a One Shot conversion, once started, can be stopped by disabling the trigger event.
	1 - ADC_CONV_MODE_CONTINUOUS	Repeated conversions of each ADC channel in an ADC channel group are performed. Continuous conversion mode is only available for group trigger source software. A Continuous conversion, once started, can be stopped by a software API call.
Description	Defines the type for configuring the conversion mode for an ADC channel group. To reduce memory usage, the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.7 Adc_GroupDefType

Table 146 Specification for Adc_GroupDefType

Syntax	`Adc_GroupDefType`	
Type	Structure	
File	`Adc.h`	
Range	--	The elements of the data structure are specific to the microcontroller.
Description	Defines the type for assignment of channels to a channel group (this is not an API type).	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.8 Adc_GroupPriorityType

Table 147 Specification for Adc_GroupPriorityType

Syntax	`Adc_GroupPriorityType`
Type	uint8
File	`Adc.h`

(table continues...)

Table 147 (continued) Specification for Adc_GroupPriorityType

Range	0..255	
Description	Defines the type for configuring the priority level of the group. The lowest priority is 0. Data width is specified based on the maximum priority level supported by AUTOSAR.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.9 Adc_GroupReplacementType

Table 148 Specification for Adc_GroupReplacementType

Syntax	`Adc_GroupReplacementType`	
Type	uint8	
File	`Adc.h`	
Range	0 - ADC_GROUP_REPL_ABORT_RESTART	Abort/Restart mechanism is used on group level if a group is interrupted by a higher priority group. The complete conversion round of the interrupted group (all group channels) is restarted after the higher priority group conversion is finished. If the group is configured in streaming access mode only the results of the interrupted conversion round are discarded. Results of the previous conversion rounds which are already written to the result buffer are not affected.
	1 - ADC_GROUP_REPL_SUSPEND_RESUME	Suspend/Resume mechanism is used on group level if a group is interrupted by a higher priority group. The conversion round of the interrupted group is completed after the higher priority group conversion is finished. Results of the previous conversion rounds which are already written to the result buffer are not affected.
Description	Defines the type for replacement mechanism used on AdcChannel group level if a group conversion is interrupted by a group which has a higher priority. To reduce memory usage, the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.10 Adc_GroupType

Table 149 Specification for Adc_GroupType

Syntax	`Adc_GroupType`	
Type	uint16	
File	`Adc.h`	
Range	0 - 383	The range of this type is microcontroller specific and has to be described by the supplier.
Description	Defines the type for numeric ID of an ADC channel group. The lower five bits represent the 'AdcChannel group ID', while the bits 5 to 8 represent the 'ADC hardware group' or 'ADC hardware unit'. The ADC driver supports a maximum of 32 channel groups per EVADC hardware group. Hence, the data width is selected based on the maximum channel group ID possible across all hardware groups.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.11 Adc_HwTriggerSignalType

Table 150 Specification for Adc_HwTriggerSignalType

Syntax	`Adc_HwTriggerSignalType`	
Type	uint8	
File	`Adc.h`	
Range	0 - ADC_HW_TRIG_RISING_EDGE	React on the rising edge of the hardware trigger signal
	1 - ADC_HW_TRIG_FALLING_EDGE	React on the falling edge of the hardware trigger signal
	2 - ADC_HW_TRIG_BOTH_EDGES	React on both edges of the hardware trigger signal
Description	Defines the type for configuring on which edge of the hardware trigger signal the driver should react. To reduce memory usage, the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.12 Adc_HwTriggerTimerType

Table 151 Specification for Adc_HwTriggerTimerType

Syntax	`Adc_HwTriggerTimerType`
Type	uint16
File	`Adc.h`

(table continues...)

Table 151 (continued) Specification for Adc_HwTriggerTimerType

Range	0 - 1023	
Description	Defines the type for reload value of the ADC module embedded timer. Data width is specified based on the range supported by the EVADC hardware trigger timer.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.13 Adc_NotifyFnPtrType

Table 152 Specification for Adc_NotifyFnPtrType

Syntax	`Adc_NotifyFnPtrType`
Type	Pointer to a function of type void Function_Name (void)
File	`Adc.h`
Description	Defines the function pointer type for call back functions (on group conversion completion).
Source	IFX
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.2.14 Adc_PowerStateRequestResultType

Table 153 Specification for Adc_PowerStateRequestResultType

Syntax	`Adc_PowerStateRequestResultType`	
Type	Enumeration	
File	`Adc.h`	
Range	0 - ADC_SERVICE_ACCEPTED	Power state change executed.
	1 - ADC_NOT_INIT	ADC Module not initialized.
	2 - ADC_SEQUENCE_ERROR	Wrong API call sequence.
	3 - ADC_HW_FAILURE	The hardware module has a failure which prevents it to enter the required power state or the read power state from hardware SFR corresponds to invalid range.
	4 - ADC_POWER_STATE_NOT_SUPP	ADC Module does not support the requested power state.
	5 - ADC_TRANS_NOT_POSSIBLE	ADC Module cannot transition directly from the current power state to the requested power state or the hardware peripheral is still busy.
Description	Defines the type for result of the requests related to power state transitions.	
Source	AUTOSAR	

(table continues...)

Table 153 (continued) Specification for Adc_PowerStateRequestResultType

Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.2.15 Adc_PowerStateType

Table 154 Specification for Adc_PowerStateType

Syntax	`Adc_PowerStateType`	
Type	uint8	
File	`Adc.h`	
Range	0 - ADC_FULL_POWER	Full Power
	1 - FAST_STANDBY_MODE	Fast standby mode
	2 - SLOW_STANDBY_MODE	Slow standby mode
	3 - ADC_OFF	Converter is off
Description	Defines the type for power state currently active or set as target power state. To reduce memory usage the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.16 Adc_PrescaleType

Table 155 Specification for Adc_PrescaleType

Syntax	`Adc_PrescaleType`	
Type	uint8	
File	`Adc.h`	
Range	2 - 32	Range of values for prescaler
Description	Defines the type for clock prescaler used for analog clock divider (DIVA). Data width is specified based on range supported by the hardware.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.17 Adc_PriorityImplementationType

Table 156 Specification for Adc_PriorityImplementationType

Syntax	`Adc_PriorityImplementationType`	
Type	uint8	
File	`Adc.h`	
Range	0 - ADC_PRIORITY_NONE	Priority mechanism is not available

(table continues...)

Table 156 (continued) Specification for Adc_PriorityImplementationType

	1 - ADC_PRIORITY_HW	Hardware priority mechanism is available only
	2 - ADC_PRIORITY_HW_SW	Hardware and software priority mechanism is available
Description	Defines the type for configuring the prioritization mechanism. To reduce memory usage, the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.18 Adc_ResolutionType

Table 157 Specification for Adc_ResolutionType

Syntax	`Adc_ResolutionType`	
Type	uint8	
File	`Adc.h`	
Range	--	Unused
Description	Defines the type for channel resolution in number of bits. Since this type is not used by the driver, smallest data type supported by the TriCore is selected for type definition.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.19 Adc_ResultAlignmentType

Table 158 Specification for Adc_ResultAlignmentType

Syntax	`Adc_ResultAlignmentType`	
Type	uint8	
File	`Adc.h`	
Range	0 - ADC_ALIGN_RIGHT	Results are right-aligned
	1 - ADC_ALIGN_LEFT	Results are left-aligned
Description	Defines the type for alignment of ADC raw results in ADC result buffer (left/right alignment). To reduce memory usage, the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.20 Adc_SamplingTimeType

Table 159 Specification for Adc_SamplingTimeType

Syntax	`Adc_SamplingTimeType`

(table continues...)

Table 159 (continued) Specification for Adc_SamplingTimeType

Type	uint8	
File	`Adc.h`	
Range	--	Unused
Description	Defines the type for sampling time during which the value is sampled, (in clock-cycles). Since this type is not used by the driver, smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.21 Adc_StartupCalibStatusType

Table 160 Specification for Adc_StartupCalibStatusType

Syntax	`Adc_StartupCalibStatusType`	
Type	uint8	
File	`Adc.h`	
Range	0 - ADC_STARTUP_CALIB_NOT_TRIGGERED	Startup Calibration not triggered
	1 - ADC_STARTUP_CALIB_ONGOING	Startup Calibration is ongoing
	2 - ADC_STARTUP_CALIB_OVER	Startup calibration over
Description	Defines the type for startup calibration status of the ADC driver. Data width is specified based on range of calibration status supported by the hardware. To reduce memory usage, the smallest data type supported by the TriCore is selected for type definition.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.22 Adc_StatusType

Table 161 Specification for Adc_StatusType

Syntax	`Adc_StatusType`	
Type	uint8	
File	`Adc.h`	
Range	0 - ADC_IDLE	The conversion of the specified group has not been started
	1 - ADC_BUSY	The conversion of the specified group has been started and is still going on and so far no result is available.
	2 - ADC_COMPLETED	A conversion round (which is not the final one) of the specified group has been finished. At least a result is available for all channels of the group.

(table continues...)

Table 161 (continued) Specification for Adc_StatusType

	3 - ADC_STREAM_COMPLETED	The result buffer is completely filled. For each channel of the selected group the number of samples to be acquired is available
Description	Defines the type for reporting the current status of the conversion of requested channel group. To reduce memory usage, the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.23 Adc_StreamBufferModeType

Table 162 Specification for Adc_StreamBufferModeType

Syntax	`Adc_StreamBufferModeType`	
Type	uint8	
File	`Adc.h`	
Range	0 - ADC_STREAM_BUFFER_LINEAR	The ADC Driver stops the conversion as soon as the stream buffer is full (number of samples reached).
	1 - ADC_STREAM_BUFFER_CIRCULAR	The ADC Driver continues the conversion even if the stream buffer is full (number of samples reached) by wrapping around the stream buffer itself.
Description	Defines the type for configuring the streaming access mode buffer type. To reduce memory usage, the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.24 Adc_StreamNumSampleType

Table 163 Specification for Adc_StreamNumSampleType

Syntax	`Adc_StreamNumSampleType`	
Type	uint8	
File	`Adc.h`	
Range	0 - 255	
Description	Defines the type for configuring the number of group conversions in streaming access mode.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.25 Adc_SyncConvModeType

Table 164 Specification for Adc_SyncConvModeType

Syntax	`Adc_SyncConvModeType`	
Type	uint8	
File	`Adc.h`	
Range	0 - ADC_SYNC_CONV_MODE_NONE	The Sync conversion mode of the specified group is none
	1 - ADC_SYNC_CONV_MODE_MASTER	The Sync conversion mode of the specified group is Master
	2 - ADC_SYNC_CONV_MODE_SLAVE	The Sync conversion mode of the specified group is Slave
Description	Defines the type for Sync conversion mode of the ADC driver. To reduce memory usage, the smallest data type supported by the TriCore is selected for type definition.	
Source	IFX	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.26 Adc_TriggerSourceType

Table 165 Specification for Adc_TriggerSourceType

Syntax	`Adc_TriggerSourceType`	
Type	uint8	
File	`Adc.h`	
Range	0 - ADC_TRIGG_SRC_SW	Group is triggered by a software API call.
	1 - ADC_TRIGG_SRC_HW	Group is triggered by a hardware event.
Description	Defines the type for configuring the trigger source for an ADC channel group. To reduce memory usage, the smallest data type supported by the TriCore is selected for type definition.	
Source	AUTOSAR	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.2.27 Adc_ValueGroupType

Table 166 Specification for Adc_ValueGroupType

Syntax	`Adc_ValueGroupType`	
Type	uint16	
File	`Adc.h`	
Range	0x0 - 0xFFF	
Description	Defines the type for reading the converted value of channel group (raw, without further scaling, alignment according to pre-compile switch ADC_RESULT_ALIGNMENT). Data width is specified based on maximum resolution supported by the EVADC hardware.	

(table continues...)

Table 166 (continued) Specification for Adc_ValueGroupType

Source	AUTOSAR
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3 Functions - APIs

This section lists all the APIs of the ADC driver.

1.3.3.1 Adc_Init

Table 167 Specification for `Adc_Init` API

Syntax	`void Adc_Init ( const Adc_ConfigType * const ConfigPtr )`	
Service ID	0	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different CPU core	
Parameters (in)	ConfigPtr	Pointer to configuration set.
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The API initializes the ADC hardware groups as per the configuration pointer passed and sets all the global variables to their initialized state. The SFRs of the configured ADC hardware unit are first reset to default values and then initialized as per the configuration. It also disables the notifications and hardware trigger capability. Groups are set to default ADC_IDLE state. This API must be invoked from all the cores using the ADC driver, as each call initializes only the SFRs and global variables of the hardware groups used by the invoking core. The SFRs and global variables common to all cores are initialized by the MCAL's master core. The ADC initialization status is set at the end of the Initialization function execution. *Note: If development error detection for the ADC module is enabled and if null pointer is passed as a parameter to the API by the master core or the slave core invokes the API with configuration pointer other than the one passed by the master core then the Adc_Init API reports the ADC_E_PARAM_CONFIG error for the Autosar version 4.2.2 and the ADC_E_PARAM_POINTER error for the Autosar version 4.4.0.*	
Source	AUTOSAR	
Error handling	ADC_E_ALREADY_INITIALIZED, ADC_E_PARAM_CONFIG, ADC_E_CLC_FAILURE, ADC_E_MASTER_CORE_UNINIT, ADC_E_CORE_NOT_CONFIGURED, ADC_E_PARAM_POINTER	

(table continues...)

Table 167 (continued) Specification for `Adc_Init` API

Configuration dependencies	-
User hints	1. ADC driver does not perform a NULL_PTR check on ConfigPtr, when DET is off. 2. Starup calibration must be triggered by the user once the initialization from all cores is completed. 3. Mcu_Init() and Port_Init() should be called before calling this API. 4. Interrupts should be in a disabled before calling this API.
SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), EVADC_CLC(rw), EVADC_EMUXSEL(w), EVADC_GLOBCFG(w), EVADC_GLOB_BOUND(w), EVADC_GLOB_EFLAG(w), EVADC_GLOB_EVNP(w), EVADC_GLOB_ICLASS(w), EVADC_GLOB_RCR(w), EVADC_GLOB_RES(w), EVADC_GLOB_TE(w), EVADC_GLOB_TF(w), EVADC_G_ALIAS(w), EVADC_G_ANCFG(w), EVADC_G_ARBCFG(w), EVADC_G_ARBPR(w), EVADC_G_BOUND(w), EVADC_G_CEFCLR(w), EVADC_G_CEVNP0(w), EVADC_G_CEVNP1(w), EVADC_G_CHCTR(w), EVADC_G_EMUXCS(rw), EVADC_G_EMUXCTR(w), EVADC_G_ICLASS(w), EVADC_G_Q_QCTRL(w), EVADC_G_Q_QMR(w), EVADC_G_Q_REQTM(w), EVADC_G_RCR(w), EVADC_G_REFCLR(w), EVADC_G_REVNP0(w), EVADC_G_REVNP1(w), EVADC_G_SEFCLR(rw), EVADC_G_SEVNP(w), EVADC_G_SYNCTR(w), EVADC_G_TRCTR(w), EVADC_G_VFR(w), GTM_ATOM_AGC_ENDIS_CTRL(rw), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_FUPD_CTRL(rw), GTM_ATOM_AGC_GLB_CTRL(w), GTM_ATOM_AGC_OUTEN_CTRL(rw), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_EICR(w), SCU_IGCR(w), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.2 Adc_DeInit

Table 168 Specification for `Adc_DeInit` API

Syntax	`void Adc_DeInit ( void )`
Service ID	0x01
Sync/Async	Synchronous

(table continues...)

Table 168 (continued) Specification for `Adc_DeInit` API

Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different CPU core	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The API resets all SFRs of the EVADC configured during initialization to their reset states including SFRs for enabling service requests. API must be invoked from all cores using the ADC driver, as each call resets only the SFRs and global variables of the hardware groups used by the calling core. The SFRs and global variables common to all cores are reset by the MCAL's master core. This API is only available when AdcDeInitApi is configured as TRUE. *Note: SFRs of the hardware groups, not configured during Adc_Init, are not deinitialized by this API. The state of an ADC driver is set to UNINIT at the beginning of the deinitialization function.*	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_BUSY, ADC_E_SLAVE_CORE_INIT	
Configuration dependencies	AdcDeInitApi	
User hints	1. Resetting of the SRC register for various SRNs, should be handled by the OS/Application. 2. The ADC module's environment must not call the function Adc_DeInit while any group is not in state ADC_IDLE.	

(table continues...)

Table 168 (continued) Specification for `Adc_DeInit` API

SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), EVADC_CLC(w), EVADC_EMUXSEL(w), EVADC_GLOBCFG(w), EVADC_GLOB_BOUND(w), EVADC_GLOB_EFLAG(w), EVADC_GLOB_EVNP(w), EVADC_GLOB_ICLASS(w), EVADC_GLOB_RCR(w), EVADC_GLOB_RES(w), EVADC_GLOB_TE(w), EVADC_GLOB_TF(w), EVADC_G_ALIAS(w), EVADC_G_ANCFG(w), EVADC_G_ARBCFG(w), EVADC_G_ARBPR(w), EVADC_G_BOUND(w), EVADC_G_CEFCLR(w), EVADC_G_CEVNP0(w), EVADC_G_CEVNP1(w), EVADC_G_CHCTR(w), EVADC_G_EMUXCS(rw), EVADC_G_EMUXCTR(w), EVADC_G_ICLASS(w), EVADC_G_Q_QCTRL(w), EVADC_G_Q_QMR(w), EVADC_G_Q_REQTM(w), EVADC_G_RCR(w), EVADC_G_REFCLR(w), EVADC_G_REVNP0(w), EVADC_G_REVNP1(w), EVADC_G_SEFCLR(rw), EVADC_G_SEVNP(w), EVADC_G_SYNCTR(w), EVADC_G_TRCTR(w), EVADC_G_VFR(w), GTM_ATOM_AGC_ENDIS_CTRL(rw), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_FUPD_CTRL(rw), GTM_ATOM_AGC_GLB_CTRL(w), GTM_ATOM_AGC_OUTEN_CTRL(rw), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_EICON0(rw), SCU_EICR(w), SCU_IGCR(w), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.3 Adc_SetupResultBuffer

Table 169 Specification for `Adc_SetupResultBuffer` API

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

Service ID	0x0c	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	Group	Numeric ID of requested ADC channel group
	DataBufferPtr	Pointer to the start of result data buffer

(table continues...)

Table 169 (continued) Specification for `Adc_SetupResultBuffer` API

Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Result buffer pointer initialized correctly E_NOT_OK: Operation failed or development error occurred
Description	The API sets up the start address of group specific result buffers, where the conversion results will be stored. The application has to ensure that the application buffer, where the DataBufferPtr points to, can hold all the conversion results of the specified group. *Note: This API is not available when the AdcResultHandlingImplementation parameter is set to ADC_DMA_MODE_RESULT_HANDLING. In this scenario the start of the application result buffer is provided through the destination address of the DMA channel.* *Note: The environment of the ADC module shall ensure that the application buffer, whose address is passed as parameter in the Adc_SetupResultBuffer API, has the size large enough to hold all group channel conversion results and if streaming access is selected, it shall be large enough to hold these results multiple times as specified with streaming sample parameter.*	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_BUSY, ADC_E_PARAM_POINTER, ADC_E_PARAM_GROUP, ADC_E_CORE_GROUP_MISMATCH	
Configuration dependencies	AdcResultHandlingImplementation	
User hints	1. ADC module's environment must ensure that no group conversions are started without prior initialization of the according result buffer pointer to point to a valid result buffer. 2. ADC Driver does not work as expected if NULL_PTR is passed via DataBufferPtr when DET is off.	
SFR accessed	CPU_CORE_ID(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.4 Adc_EnableGroupNotification

Table 170 Specification for `Adc_EnableGroupNotification` API

Syntax	`void  Adc_EnableGroupNotification`
	`(`
	`  const Adc_GroupType Group`
	`)`

Service ID	0x07
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info

(table continues...)

Table 170 (continued) Specification for `Adc_EnableGroupNotification` API

Re-entrancy	Reentrant for different channel groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The API enables the notification mechanism for the requested ADC channel group. *Note: This API is only available when the AdcGrpNotifCapability parameter is configured as TRUE.*	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_PARAM_GROUP, ADC_E_NOTIF_CAPABILITY, ADC_E_CORE_GROUP_MISMATCH	
Configuration dependencies	AdcGrpNotifCapability	
User hints	None	
SFR accessed	CPU_CORE_ID(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.5 Adc_DisableGroupNotification

Table 171 Specification for `Adc_DisableGroupNotification` API

Syntax	`void Adc_DisableGroupNotification ( const Adc_GroupType Group )`	
Service ID	0x08	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different channel groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group
Parameters (out)	-	-

(table continues...)

Table 171 (continued) Specification for `Adc_DisableGroupNotification` API

Parameters (in - out)	-	-
Return	void	-
Description	The API disables the notification mechanism for the requested ADC channel group. *Note: This API is only available when the AdcGrpNotifCapability parameter is configured as TRUE.*	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_PARAM_GROUP, ADC_E_NOTIF_CAPABILITY, ADC_E_CORE_GROUP_MISMATCH	
Configuration dependencies	AdcGrpNotifCapability	
User hints	None	
SFR accessed	CPU_CORE_ID(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.6 Adc_StartGroupConversion

Table 172 Specification for `Adc_StartGroupConversion` API

Syntax	`void Adc_StartGroupConversion ( const Adc_GroupType Group )`	
Service ID	0x02	
Sync/Async	Asynchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for channel groups executing on different ADC hardware groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The API starts the conversion of all the channels of the requested AdcChannel group. *Note: This API is available only when the AdcEnableStartStopGroupApi parameter is configured as TRUE.*	

(table continues...)

Table 172 (continued) Specification for `Adc_StartGroupConversion` API

Source	AUTOSAR
Error handling	ADC_E_UNINIT, ADC_E_BUSY, ADC_E_PARAM_GROUP, ADC_E_WRONG_TRIGG_SRC, ADC_E_BUFFER_UNINIT, ADC_E_CORE_GROUP_MISMATCH, ADC_E_CONVERTER_OFF, ADC_SE_CALIB_ONGOING
Configuration dependencies	AdcEnableStartStopGroupApi
User hints	1. ADC Driver does not work as expected if wrong group is passed when DET is off 2. The ADC module's environment should call the function Adc_StartGroupConversion for groups configured with software trigger source only.
SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), EVADC_GLOB_TE(rw), EVADC_G_ALIAS(w), EVADC_G_ARBCFG(r), EVADC_G_ARBPR(rw), EVADC_G_BOUND(w), EVADC_G_CEFCLR(w), EVADC_G_CHCTR(rw), EVADC_G_EMUXCS(w), EVADC_G_EMUXCTR(w), EVADC_G_Q_QCTRL(w), EVADC_G_Q_QINR(w), EVADC_G_Q_QMR(w), EVADC_G_Q_REQTM(w), EVADC_G_RCR(rw), EVADC_G_REFCLR(w), EVADC_G_SEFCLR(w), EVADC_G_VFR(w), GTM_ATOM_AGC_ENDIS_CTRL(rw), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_FUPD_CTRL(rw), GTM_ATOM_AGC_GLB_CTRL(w), GTM_ATOM_AGC_OUTEN_CTRL(rw), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_CM0(w), GTM_ATOM_CH_CM1(w), GTM_ATOM_CH_CN0(w), GTM_ATOM_CH_CTRL(w), GTM_ATOM_CH_IRQ_EN(w), GTM_ATOM_CH_IRQ_MODE(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_ATOM_CH_SR0(w), GTM_ATOM_CH_SR1(w), GTM_TOM_CH_CM0(w), GTM_TOM_CH_CM1(w), GTM_TOM_CH_CN0(w), GTM_TOM_CH_CTRL(w), GTM_TOM_CH_IRQ_EN(w), GTM_TOM_CH_IRQ_MODE(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_SR0(w), GTM_TOM_CH_SR1(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_EICR(w), SCU_IGCR(w), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SRC_VADC_G_SR(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.7 Adc_StopGroupConversion

Table 173 Specification for `Adc_StopGroupConversion` API

Syntax	`void Adc_StopGroupConversion`
	`(`
	`  const Adc_GroupType Group`
	`)`

Service ID	0x03

(table continues...)

Table 173 (continued) Specification for `Adc_StopGroupConversion` API

Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for channel groups executing on different ADC hardware groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The API stops the conversion of the requested ADC channel group. It also disables the group notification and sets the group to IDLE state. *Note: This API is available only when the AdcEnableStartStopGroupApi parameter is configured as TRUE.*	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_IDLE, ADC_E_PARAM_GROUP, ADC_E_WRONG_TRIGG_SRC, ADC_E_CONV_STOP_TIME_FAILURE, ADC_E_CORE_GROUP_MISMATCH	
Configuration dependencies	AdcEnableStartStopGroupApi	
User hints	ADC module's environment should call the function Adc_StopGroupConversion for groups configured with trigger source software only.	
SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), EVADC_GLOB_TE(rw), EVADC_G_ALIAS(w), EVADC_G_ARBCFG(r), EVADC_G_ARBPR(rw), EVADC_G_BOUND(w), EVADC_G_CEFCLR(w), EVADC_G_CHCTR(rw), EVADC_G_EMUXCS(w), EVADC_G_EMUXCTR(w), EVADC_G_Q_QCTRL(w), EVADC_G_Q_QINR(w), EVADC_G_Q_QMR(w), EVADC_G_Q_REQTM(w), EVADC_G_RCR(rw), EVADC_G_REFCLR(w), EVADC_G_SEFCLR(w), EVADC_G_VFR(w), GTM_ATOM_AGC_ENDIS_CTRL(rw), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_FUPD_CTRL(rw), GTM_ATOM_AGC_GLB_CTRL(w), GTM_ATOM_AGC_OUTEN_CTRL(rw), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_EICR(w), SCU_IGCR(w), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SRC_VADC_G_SR(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	

(table continues...)

Table 173 (continued) Specification for `Adc_StopGroupConversion` API	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.8 Adc_EnableHardwareTrigger

Table 174 Specification for `Adc_EnableHardwareTrigger` API		
Syntax	`void Adc_EnableHardwareTrigger ( const Adc_GroupType Group )`	
Service ID	0x05	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for channel groups executing on different ADC hardware groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	The API enables the hardware trigger or gate for the requested ADC channel group. *Note: This API is available only when the AdcHwTriggerApi parameter is configured as TRUE.*	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_PARAM_GROUP, ADC_E_WRONG_CONV_MODE, ADC_E_WRONG_TRIGG_SRC, ADC_E_BUFFER_UNINIT, ADC_E_BUSY, ADC_E_CORE_GROUP_MISMATCH, ADC_E_CONVERTER_OFF, ADC_SE_CALIB_ONGOING	
Configuration dependencies	AdcHwTriggerApi	
User hints	1. ADC module’s environment should call the function Adc_EnableHardwareTrigger for groups configured in hardware trigger mode only. 2. ADC module's environment must guarantee that no concurrent conversions take place on the same hardware unit (happening of different hardware triggers at the same time).	

(table continues...)

Table 174 (continued) Specification for `Adc_EnableHardwareTrigger` API

SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), EVADC_GLOB_TE(rw), EVADC_G_ALIAS(w), EVADC_G_ARBCFG(r), EVADC_G_ARBPR(rw), EVADC_G_BOUND(w), EVADC_G_CEFCLR(w), EVADC_G_CHCTR(rw), EVADC_G_EMUXCS(w), EVADC_G_EMUXCTR(w), EVADC_G_Q_QCTRL(w), EVADC_G_Q_QINR(w), EVADC_G_Q_QMR(w), EVADC_G_Q_REQTM(w), EVADC_G_RCR(rw), EVADC_G_REFCLR(w), EVADC_G_SEFCLR(w), EVADC_G_VFR(w), GTM_ATOM_AGC_ENDIS_CTRL(rw), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_FUPD_CTRL(rw), GTM_ATOM_AGC_GLB_CTRL(w), GTM_ATOM_AGC_OUTEN_CTRL(rw), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_CM0(w), GTM_ATOM_CH_CM1(w), GTM_ATOM_CH_CN0(w), GTM_ATOM_CH_CTRL(w), GTM_ATOM_CH_IRQ_EN(w), GTM_ATOM_CH_IRQ_MODE(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_ATOM_CH_SR0(w), GTM_ATOM_CH_SR1(w), GTM_TOM_CH_CM0(w), GTM_TOM_CH_CM1(w), GTM_TOM_CH_CN0(w), GTM_TOM_CH_CTRL(w), GTM_TOM_CH_IRQ_EN(w), GTM_TOM_CH_IRQ_MODE(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_SR0(w), GTM_TOM_CH_SR1(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_EICR(w), SCU_IGCR(w), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SRC_VADC_G_SR(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.9 Adc_DisableHardwareTrigger

Table 175 Specification for `Adc_DisableHardwareTrigger` API

Syntax	`void Adc_DisableHardwareTrigger ( const Adc_GroupType Group )`	
Service ID	0x06	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for channel groups executing on different ADC hardware groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group
Parameters (out)	-	-

(table continues...)

Table 175 (continued) Specification for `Adc_DisableHardwareTrigger` API

Parameters (in - out)	-	-
Return	void	-
Description	The API stops the ongoing conversion and disables the hardware trigger for the requested ADC channel group. It also disables the group notification and sets the group to IDLE state. *Note: This API is available only when the AdcHwTriggerApi parameter is configured as TRUE.*	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_PARAM_GROUP, ADC_E_WRONG_CONV_MODE, ADC_E_WRONG_TRIGG_SRC, ADC_E_CONV_STOP_TIME_FAILURE, ADC_E_IDLE, ADC_E_CORE_GROUP_MISMATCH	
Configuration dependencies	AdcHwTriggerApi	
User hints	ADC module's environment should call the function Adc_DisableHardwareTrigger for groups configured in hardware trigger mode only.	
SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), EVADC_GLOB_TE(rw), EVADC_G_ALIAS(w), EVADC_G_ARBCFG(r), EVADC_G_ARBPR(rw), EVADC_G_BOUND(w), EVADC_G_CEFCLR(w), EVADC_G_CHCTR(rw), EVADC_G_EMUXCS(w), EVADC_G_EMUXCTR(w), EVADC_G_Q_QCTRL(w), EVADC_G_Q_QINR(w), EVADC_G_Q_QMR(w), EVADC_G_Q_REQTM(w), EVADC_G_RCR(rw), EVADC_G_REFCLR(w), EVADC_G_SEFCLR(w), EVADC_G_VFR(w), GTM_ATOM_AGC_ENDIS_CTRL(rw), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_FUPD_CTRL(rw), GTM_ATOM_AGC_GLB_CTRL(w), GTM_ATOM_AGC_OUTEN_CTRL(rw), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_EICR(w), SCU_IGCR(w), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SRC_VADC_G_SR(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.10 Adc_GetGroupStatus

Table 176 Specification for `Adc_GetGroupStatus` API

Syntax	`Adc_StatusType Adc_GetGroupStatus ( const Adc_GroupType Group )`	
Service ID	0x09	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different channel groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Adc_StatusType	Conversion status for the requested group.
Description	The API returns the status of the requested ADC channel group.	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_PARAM_GROUP, ADC_E_CORE_GROUP_MISMATCH	
Configuration dependencies	-	
User hints	None	
SFR accessed	CPU_CORE_ID(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.11 Adc_ReadGroup

Table 177 Specification for `Adc_ReadGroup` API

Syntax	`Std_ReturnType Adc_ReadGroup ( const Adc_GroupType Group, Adc_ValueGroupType * const DataBufferPtr )`
Service ID	0x04
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info

(table continues...)

Table 177 (continued) Specification for `Adc_ReadGroup` API

Re-entrancy	Reentrant for different AdcChannel Groups	
Parameters (in)	Group	Numeric ID of requested ADC channel group.
Parameters (out)	-	-
Parameters (in - out)	DataBufferPtr	ADC results of all channels of the selected group are stored in the data buffer addressed with the pointer.
Return	Std_ReturnType	E_OK: Results are available and written to the data buffer. E_NOT_OK: No results are available or development error occurred
Description	The API reads the group conversion result of the last completed conversion round of the requested group and stores the channel conversion value starting from the DataBufferPtr address. The channel conversion results are stored in ascending channel order. *Note: This API is available only when the AdcReadGroupApi parameter is configured as TRUE.* *Note: This API is not available when the AdcResultHandlingImplementation parameter is set to ADC_DMA_MODE_RESULT_HANDLING.*	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_IDLE, ADC_E_PARAM_GROUP, ADC_E_CORE_GROUP_MISMATCH, ADC_E_PARAM_POINTER	
Configuration dependencies	AdcReadGroupApi	
User hints	ADC module's environment must ensure that a conversion has been completed for the requested group before requesting the conversion result.	
SFR accessed	CPU_CORE_ID(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.12 Adc_GetStreamLastPointer

Table 178 Specification for `Adc_GetStreamLastPointer` API

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

Service ID	0x0b
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info

(table continues...)

Table 178 (continued) Specification for `Adc_GetStreamLastPointer` API

Re-entrancy	Reentrant for different AdcChannel Groups	
Parameters (in)	Group	Numeric ID of requested ADC Channel group.
Parameters (out)	PtrToSamplePtr	Pointer to result buffer pointer.
Parameters (in - out)	-	-
Return	Adc_StreamNumSampleType	Number of valid samples per channel.
Description	The API returns the number of valid samples per channel, stored in the result buffer. It also updates the PtrToSamplePtr with the address (within the group result buffer) of the latest result sample of the first channel. With the pointer and the return value, all valid group conversion results can be accessed. *Note: This API is not available when AdcResultHandlingImplementation parameter is set to ADC_DMA_MODE_RESULT_HANDLING.*	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_PARAM_GROUP, ADC_E_CORE_GROUP_MISMATCH, ADC_E_IDLE, ADC_E_PARAM_POINTER	
Configuration dependencies	AdcResultHandlingImplementation	
User hints	None	
SFR accessed	CPU_CORE_ID(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.13 Adc_GetCurrentPowerState

Table 179 Specification for `Adc_GetCurrentPowerState` API

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

Service ID	0x11
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info
Re-entrancy	Reentrant

(table continues...)

Table 179 (continued) Specification for `Adc_GetCurrentPowerState` API

Parameters (in)	-	-
Parameters (out)	CurrentPowerState Result	The current power mode of the ADC HW Unit is returned in this parameter If the API returns E_OK: ADC_SERVICE_ACCEPTED: Current power mode was returned. If the API returns E_NOT_OK: ADC_NOT_INIT: ADC Module not initialized. ADC_HW_FAILURE: Current power state read from the SFR corresponds to invalid range.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Power mode could be provided E_NOT_OK: Power mode could not be provided
Description	The API returns the current power state of the ADC hardware groups assigned to the calling core. *Note: This API is available only when the AdcLowPowerStatesSupport parameter is configured as TRUE.*	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_PARAM_POINTER, ADC_SE_POWER_STATE_INVALID	
Configuration dependencies	AdcLowPowerStatesSupport	
User hints	None	
SFR accessed	CPU_CORE_ID(r), EVADC_G_ARBCFG(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.14 Adc_GetTargetPowerState

Table 180 Specification for `Adc_GetTargetPowerState` API

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

Service ID	0x12
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info

(table continues...)

Table 180 (continued) Specification for `Adc_GetTargetPowerState` API

Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	TargetPowerState Result	The Target power mode of the ADC HW Unit is returned in this parameter If the API returns E_OK: ADC_SERVICE_ACCEPTED:Target power mode was returned. If the API returns E_NOT_OK: ADC_NOT_INIT: ADC Module not initialized. ADC_HW_FAILURE: Current power state read from the SFR corresponds to invalid range.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Power mode could be provided E_NOT_OK: Power mode could not be provided
Description	The API returns the power state successfully prepared by the Adc_PreparePowerState API. If the power state is not prepared, then the current power state of hardware units assigned to the invoking core is returned. *Note: This API is available only when the AdcLowPowerStatesSupport parameter is configured as TRUE.*	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_PARAM_POINTER, ADC_SE_POWER_STATE_INVALID	
Configuration dependencies	AdcLowPowerStatesSupport	
User hints	None	
SFR accessed	CPU_CORE_ID(r), EVADC_G_ARBCFG(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.15 Adc_PreparePowerState

Table 181 Specification for `Adc_PreparePowerState` API

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

Service ID	0x13

(table continues...)

Table 181 (continued) Specification for `Adc_PreparePowerState` API

Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different CPU cores	
Parameters (in)	PowerState	The target power state intended to be attained
Parameters (out)	Result	If the API returns E_OK: ADC_SERVICE_ACCEPTED: ADC Module power state preparation was started. ADC_SEQUENCE_ERROR: Current power state of all hardware units is same as the target power state If the API returns E_NOT_OK: ADC_NOT_INIT: ADC Module not initialized. ADC_POWER_STATE_NOT_SUPP: ADC Module does not support the requested power state. ADC_HW_FAILURE: Current power state read from the SFR corresponds to invalid range.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Preparation process started or Current power state of all the hardware units is same as the target power state E_NOT_OK: Service is rejected
Description	The API starts the needed process for the ADC hardware units assigned to the invoking core to enter the target power state. If the current power state of the hardware units assigned to the invoking core is same as the target power state, then the API does not perform the intended action. The API must be invoked from all the cores using the ADC driver, as each call prepares the power state only for the ADC hardware groups used by the calling core. *Note: This API is available only when the AdcLowPowerStatesSupport parameter is configured as TRUE.*	
Source	AUTOSAR	
Error handling	ADC_E_UNINIT, ADC_E_POWER_STATE_NOT_SUPPORTED, ADC_E_PARAM_POINTER, ADC_SE_POWER_STATE_INVALID	
Configuration dependencies	AdcLowPowerStatesSupport	
User hints	None	
SFR accessed	CPU_CORE_ID(r), EVADC_G_ARBCFG(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.16 Adc_SetPowerState

Table 182 Specification for `Adc_SetPowerState` API

Syntax	`Std_ReturnType Adc_SetPowerState ( Adc_PowerStateRequestResultType * const Result )`	
Service ID	0x10	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different CPU cores	
Parameters (in)	-	-
Parameters (out)	Result	If the API returns E_OK: ADC_SERVICE_ACCEPTED: Power state change executed. If the API returns E_NOT_OK: ADC_NOT_INIT: ADC Module not initialized. ADC_SEQUENCE_ERROR: wrong API call sequence. ADC_TRANS_NOT_POSSIBLE: ADC channel groups are not in state IDLE or notifications enabled.
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Power Mode changed E_NOT_OK: Service is rejected
Description	The API sets the already prepared power states for all the ADC hardware units assigned to the invoking core. The API must be invoked from all the cores using the ADC driver, as each call sets the power state only for the ADC hardware groups used by the calling core. *Note: When the converter is set to 'full power mode', then the conversion is started immediately by the hardware after a request (no additional time delay). This is the default mode after modules initialization.* *Note: When the converter is set to 'Fast standby mode' or 'Slow standby mode' and no conversion is requested, then it saves power. When a conversion is triggered then the converter wakes up automatically, but it needs certain wake-up time before conversions can be performed. There is no power saving while the conversions are being performed during this mode. During this phase the driver reports the current power state as 'Fast standby mode' or 'Slow standby mode'.* *In this mode, when the conversion is no longer active, then the converter saves power and a new conversion request will trigger the wake-up cycle again (time delay for wake-up).* *Note: When the converter is set to 'off mode', it enters power saving mode and no further conversions are possible until a power mode transition is made explicitly to come out of 'off mode'.* *Note: This API is available only when the AdcLowPowerStatesSupport parameter is configured as TRUE.*	
Source	AUTOSAR	

(table continues...)

Table 182 (continued) Specification for `Adc_SetPowerState` API

Error handling	ADC_E_UNINIT, ADC_E_NOT_DISENGAGED, ADC_E_PERIPHERAL_NOT_PREPARED, ADC_E_PARAM_POINTER
Configuration dependencies	AdcLowPowerStatesSupport
User hints	None
SFR accessed	CPU_CORE_ID(r), EVADC_G_ARBCFG(w) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.3.17 Adc_GetVersionInfo

Table 183 Specification for `Adc_GetVersionInfo` API

Syntax	`void Adc_GetVersionInfo ( Std_VersionInfoType * const versioninfo )`	
Service ID	0x0a	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	versioninfo	Pointer to where to store the version information of the ADC driver.
Parameters (in - out)	-	-
Return	void	-
Description	The API returns the version information of the ADC driver. *Note: This API is available only when the AdcVersionInfoApi parameter is configured as TRUE.*	
Source	AUTOSAR	
Error handling	ADC_E_PARAM_POINTER	
Configuration dependencies	AdcVersionInfoApi	
User hints	-	
SFR accessed	-	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.18 Adc_TriggerStartupCal

Table 184 Specification for `Adc_TriggerStartupCal` API

Syntax	`Std_ReturnType Adc_TriggerStartupCal ( void )`	
Service ID	0x31	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Non Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Start-up calibration is triggered E_NOT_OK: Start-up calibration is not triggered
Description	The API triggers the start-up calibration. The API should be invoked only once. It can be invoked from any of the cores. However, before the API is invoked it should be ensured that initialization sequence of all the cores is over. *Note: This API is available only when the AdcStartupCalibApi parameter is configured as TRUE.*	
Source	IFX	
Error handling	ADC_E_UNINIT	
Configuration dependencies	AdcStartupCalibApi	
User hints	API should be triggered after the ADC initialization is completed in all the Cores.	
SFR accessed	EVADC_GLOBCFG(rw), EVADC_G_CEFCLR(w), EVADC_G_CHCTR(rw), EVADC_G_Q_QCTRL(w), EVADC_G_Q_QINR(w), EVADC_G_Q_QMR(w), EVADC_G_RCR(w), EVADC_G_REFCLR(w), EVADC_G_SEFCLR(w), EVADC_G_VFR(w) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.19 Adc_GetStartupCalStatus

Table 185 Specification for `Adc_GetStartupCalStatus` API

Syntax	`Adc_StartupCalibStatusType Adc_GetStartupCalStatus ( void )`	
Service ID	0x30	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant	
Parameters (in)	-	-
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Adc_StartupCalibStatusType	ADC_STARTUP_CALIB_NOT_TRIGGERED: Startup Calibration not triggered or DET occurred ADC_STARTUP_CALIB_ONGOING: Startup Calibration is ongoing. ADC_STARTUP_CALIB_OVER: Startup calibration over
Description	The API returns the status of the start-up calibration for all the ADC hardware groups assigned to invoking core. *Note: This API is available only when the AdcStartupCalibApi parameter is configured as TRUE.*	
Source	IFX	
Error handling	ADC_E_UNINIT	
Configuration dependencies	AdcStartupCalibApi	
User hints	None	
SFR accessed	CPU_CORE_ID(r), EVADC_G_ARBCFG(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.3.20 Adc_InitCheck

Table 186 Specification for `Adc_InitCheck` API

Syntax	`Std_ReturnType Adc_InitCheck ( const Adc_ConfigType * const ConfigPtr )`

(table continues...)

Table 186 (continued) Specification for `Adc_InitCheck` API

Service ID	0x32	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different CPU core	
Parameters (in)	ConfigPtr	Pointer to ADC configuration Set
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	Std_ReturnType	E_OK: Initialization check passed E_NOT_OK: In Case of - Driver is not initialized - Global Variables or SFR is not set as expected - Invalid input parameter
Description	The API checks whether the initialization was performed correctly or not. The check for correct initialization is only done in the context of the invoking core. *Note: The API is available only when the AdcInitCheckApi parameter is configured as TRUE.* *Note: Init check should be performed in the following sequence:* *1. Call Adc_Init from a core.* *2. Call Adc_InitCheck from the same core.*	
Source	IFX	
Error handling	-	
Configuration dependencies	AdcSafetyEnable,AdcInitCheckApi	
User hints	The ADC module's environment must ensure that Adc_InitCheck should be performed in the following sequence: 1. Call Adc_Init from a core. 2. Call Adc_InitCheck from the same core.	

(table continues...)

Table 186 (continued) Specification for `Adc_InitCheck` API

SFR accessed	CPU_CORE_ID(r), EVADC_CLC(r), EVADC_EMUXSEL(r), EVADC_GLOBCFG(r), EVADC_GLOB_BOUND(r), EVADC_GLOB_EFLAG(r), EVADC_GLOB_EVNP(r), EVADC_GLOB_ICLASS(r), EVADC_GLOB_RCR(r), EVADC_GLOB_RES(r), EVADC_GLOB_TE(r), EVADC_GLOB_TF(r), EVADC_G_ALIAS(r), EVADC_G_ANCFG(r), EVADC_G_ARBCFG(r), EVADC_G_ARBPR(r), EVADC_G_BOUND(r), EVADC_G_CEFCLR(r), EVADC_G_CEFLAG(r), EVADC_G_CEVNP0(r), EVADC_G_CEVNP1(r), EVADC_G_CHCTR(r), EVADC_G_EMUXCS(r), EVADC_G_EMUXCTR(r), EVADC_G_ICLASS(r), EVADC_G_Q_QCTRL(r), EVADC_G_Q_QMR(r), EVADC_G_Q_QSR(r), EVADC_G_Q_REQTM(r), EVADC_G_RCR(r), EVADC_G_REFCLR(r), EVADC_G_REFLAG(r), EVADC_G_RES(r), EVADC_G_REVNP0(r), EVADC_G_REVNP1(r), EVADC_G_SEFCLR(r), EVADC_G_SEFLAG(r), EVADC_G_SEVNP(r), EVADC_G_SYNCTR(r), EVADC_G_TRCTR(r), EVADC_G_VFR(r), GTM_ATOM_AGC_ENDIS_STAT(r), GTM_ATOM_CH_CM0(r), GTM_ATOM_CH_CM1(r), GTM_ATOM_CH_CN0(r), GTM_ATOM_CH_CTRL(r), GTM_ATOM_CH_IRQ_EN(r), GTM_ATOM_CH_IRQ_MODE(r), GTM_ATOM_CH_SR0(r), GTM_ATOM_CH_SR1(r), GTM_TOM_CH_CM0(r), GTM_TOM_CH_CM1(r), GTM_TOM_CH_CN0(r), GTM_TOM_CH_CTRL(r), GTM_TOM_CH_IRQ_EN(r), GTM_TOM_CH_IRQ_MODE(r), GTM_TOM_CH_SR0(r), GTM_TOM_CH_SR1(r), GTM_TOM_TGC0_ENDIS_STAT(r), GTM_TOM_TGC1_ENDIS_STAT(r), SCU_EICR(r), SCU_IGCR(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.4 Notifications and Callbacks

The ADC driver does not provide any notifications or callbacks.

1.3.5 Scheduled functions

The ADC driver does not provide any scheduled functions.

1.3.6 Interrupt service routines

This section lists all the interrupt handlers of the ADC driver.

1.3.6.1 Adc_ChEventInterruptHandler

Table 187 Specification for `Adc_ChEventInterruptHandler` API

Syntax	`void Adc_ChEventInterruptHandler ( const uint32 KernelId )`
Service ID	0x36
Sync/Async	Synchronous
Safety Level	Refer to the release notes for the safety related info

(table continues...)

Table 187 (continued) Specification for `Adc_ChEventInterruptHandler` API

Re-entrancy	Reentrant for different ADC hardware groups	
Parameters (in)	KernelId	Hardware group ID
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Handles the interrupts from a channel event for the passed ADC KernelId. Channel events are triggered when the conversion results are in range (as configured) for a limit checking group. Application will be notified if a notification is configured (in tresos) and enabled. *Note: This API is available only when AdcLimitCheckApi is configured as TRUE.*	
Source	IFX	
Error handling	ADC_E_CONV_STOP_TIME_FAILURE, ADC_SE_INT_PLAUSIBILITY, ADC_SE_PARAM_KERNEL	
Configuration dependencies	AdcEnableLimitCheck	
User hints	User must call this interrupt handler from the ISR of GxSRN3 of each kernel and pass the kernel ID as the parameter.	
SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), EVADC_GLOB_TE(rw), EVADC_G_ALIAS(w), EVADC_G_ARBCFG(r), EVADC_G_ARBPR(rw), EVADC_G_BOUND(w), EVADC_G_CEFCLR(w), EVADC_G_CEFLAG(r), EVADC_G_CHCTR(rw), EVADC_G_EMUXCS(w), EVADC_G_EMUXCTR(w), EVADC_G_Q_QCTRL(w), EVADC_G_Q_QINR(w), EVADC_G_Q_QMR(w), EVADC_G_Q_REQTM(w), EVADC_G_RCR(rw), EVADC_G_REFCLR(w), EVADC_G_RES(r), EVADC_G_SEFCLR(w), EVADC_G_VFR(w), GTM_ATOM_AGC_ENDIS_CTRL(rw), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_FUPD_CTRL(rw), GTM_ATOM_AGC_GLB_CTRL(w), GTM_ATOM_AGC_OUTEN_CTRL(rw), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_EICR(w), SCU_IGCR(w), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SRC_VADC_G_SR(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*	
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.	

1.3.6.2 Adc_RS0EventInterruptHandler

Table 188 Specification for `Adc_RS0EventInterruptHandler` API

Syntax	`void Adc_RS0EventInterruptHandler ( const uint32 KernelId )`	
Service ID	0x33	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different ADC hardware groups	
Parameters (in)	KernelId	Hardware group ID
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Handles the interrupts from Request Source 0 event for the passed ADC KernelId. Request Source 0 event is triggered when all channels of an AdcGroup installed on it, completes one round of conversion. Application will be notified if a notification is configured (in tresos) and enabled.	
Source	IFX	
Error handling	ADC_E_CONV_STOP_TIME_FAILURE, ADC_SE_INT_PLAUSIBILITY, ADC_SE_PARAM_KERNEL	
Configuration dependencies	-	
User hints	User must call this interrupt handler from the ISR of GxSRN0 of each kernel and pass the kernel ID as the parameter.	

(table continues...)

Table 188 (continued) Specification for `Adc_RS0EventInterruptHandler` API

SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), EVADC_GLOB_TE(rw), EVADC_G_ALIAS(w), EVADC_G_ARBCFG(r), EVADC_G_ARBPR(rw), EVADC_G_BOUND(w), EVADC_G_CEFCLR(w), EVADC_G_CHCTR(rw), EVADC_G_EMUXCS(w), EVADC_G_EMUXCTR(w), EVADC_G_Q_QCTRL(w), EVADC_G_Q_QINR(w), EVADC_G_Q_QMR(w), EVADC_G_Q_REQTM(w), EVADC_G_RCR(rw), EVADC_G_REFCLR(w), EVADC_G_RES(r), EVADC_G_SEFCLR(w), EVADC_G_SEFLAG(r), EVADC_G_VFR(w), GTM_ATOM_AGC_ENDIS_CTRL(rw), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_FUPD_CTRL(rw), GTM_ATOM_AGC_GLB_CTRL(w), GTM_ATOM_AGC_OUTEN_CTRL(rw), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_EICR(w), SCU_IGCR(w), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SRC_VADC_G_SR(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.6.3 Adc_RS1EventInterruptHandler

Table 189 Specification for `Adc_RS1EventInterruptHandler` API

Syntax	`void Adc_RS1EventInterruptHandler ( const uint32 KernelId )`	
Service ID	0x34	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different ADC hardware groups	
Parameters (in)	KernelId	Hardware group ID
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-

(table continues...)

Table 189 (continued) Specification for `Adc_RS1EventInterruptHandler` API

Description	Handles the interrupts from Request Source 1 event for the passed ADC KernelId. Request Source 1 event is triggered when all channels of an AdcGroup installed on it, completes one round of conversion. Application will be notified if a notification is configured (in tresos) and enabled. *Note: This function is available only if the AdcPriorityImplementation parameter is not equal to ADC_PRIORITY_NONE.*
Source	IFX
Error handling	ADC_E_CONV_STOP_TIME_FAILURE, ADC_SE_INT_PLAUSIBILITY, ADC_SE_PARAM_KERNEL
Configuration dependencies	AdcPriorityImplementation
User hints	User must call this interrupt handler from the ISR of GxSRN1 of each kernel and pass the kernel ID as the parameter.
SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), EVADC_GLOB_TE(rw), EVADC_G_ALIAS(w), EVADC_G_ARBCFG(r), EVADC_G_ARBPR(rw), EVADC_G_BOUND(w), EVADC_G_CEFCLR(w), EVADC_G_CHCTR(rw), EVADC_G_EMUXCS(w), EVADC_G_EMUXCTR(w), EVADC_G_Q_QCTRL(w), EVADC_G_Q_QINR(w), EVADC_G_Q_QMR(w), EVADC_G_Q_REQTM(w), EVADC_G_RCR(rw), EVADC_G_REFCLR(w), EVADC_G_RES(r), EVADC_G_SEFCLR(w), EVADC_G_SEFLAG(r), EVADC_G_VFR(w), GTM_ATOM_AGC_ENDIS_CTRL(rw), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_FUPD_CTRL(rw), GTM_ATOM_AGC_GLB_CTRL(w), GTM_ATOM_AGC_OUTEN_CTRL(rw), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_EICR(w), SCU_IGCR(w), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SRC_VADC_G_SR(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.6.4 Adc_RS2EventInterruptHandler

Table 190 Specification for `Adc_RS2EventInterruptHandler` API

Syntax	`void  Adc_RS2EventInterruptHandler`
	`(`
	`  const uint32 KernelId`
	`)`

(table continues...)

Table 190 (continued) Specification for `Adc_RS2EventInterruptHandler` API

Service ID	0x35	
Sync/Async	Synchronous	
Safety Level	Refer to the release notes for the safety related info	
Re-entrancy	Reentrant for different ADC hardware groups	
Parameters (in)	KernelId	Hardware group ID
Parameters (out)	-	-
Parameters (in - out)	-	-
Return	void	-
Description	Handles the interrupts from Request Source 2 event for the passed ADC KernelId. Request Source 2 event is triggered when all channels of an AdcGroup installed on it, completes one round of conversion. Application will be notified if a notification is configured (in tresos) and enabled. *Note: This function is available only if the AdcPriorityImplementation parameter is not equal to ADC_PRIORITY_NONE.*	
Source	IFX	
Error handling	ADC_E_CONV_STOP_TIME_FAILURE, ADC_SE_INT_PLAUSIBILITY, ADC_SE_PARAM_KERNEL	
Configuration dependencies	AdcPriorityImplementation	
User hints	User must call this interrupt handler from the ISR of GxSRN2 of each kernel and pass the kernel ID as the parameter.	

(table continues...)

Table 190 (continued) Specification for `Adc_RS2EventInterruptHandler` API

SFR accessed	CPU_COMPAT(w), CPU_CORE_ID(r), CPU_SYSCON(w), CPU_TPS_EXTIM_CLASS_EN(w), CPU_TPS_EXTIM_ENTRY_LVAL(w), CPU_TPS_EXTIM_EXIT_LVAL(w), EVADC_GLOB_TE(rw), EVADC_G_ALIAS(w), EVADC_G_ARBCFG(r), EVADC_G_ARBPR(rw), EVADC_G_BOUND(w), EVADC_G_CEFCLR(w), EVADC_G_CHCTR(rw), EVADC_G_EMUXCS(w), EVADC_G_EMUXCTR(w), EVADC_G_Q_QCTRL(w), EVADC_G_Q_QINR(w), EVADC_G_Q_QMR(w), EVADC_G_Q_REQTM(w), EVADC_G_RCR(rw), EVADC_G_REFCLR(w), EVADC_G_RES(r), EVADC_G_SEFCLR(w), EVADC_G_SEFLAG(r), EVADC_G_VFR(w), GTM_ATOM_AGC_ENDIS_CTRL(rw), GTM_ATOM_AGC_ENDIS_STAT(w), GTM_ATOM_AGC_FUPD_CTRL(rw), GTM_ATOM_AGC_GLB_CTRL(w), GTM_ATOM_AGC_OUTEN_CTRL(rw), GTM_ATOM_AGC_OUTEN_STAT(w), GTM_ATOM_CH_IRQ_NOTIFY(w), GTM_TOM_CH_IRQ_NOTIFY(w), GTM_TOM_TGC0_ENDIS_CTRL(rw), GTM_TOM_TGC0_ENDIS_STAT(w), GTM_TOM_TGC0_FUPD_CTRL(rw), GTM_TOM_TGC0_GLB_CTRL(w), GTM_TOM_TGC0_OUTEN_CTRL(rw), GTM_TOM_TGC0_OUTEN_STAT(w), GTM_TOM_TGC1_ENDIS_CTRL(rw), GTM_TOM_TGC1_ENDIS_STAT(w), GTM_TOM_TGC1_FUPD_CTRL(rw), GTM_TOM_TGC1_GLB_CTRL(w), GTM_TOM_TGC1_OUTEN_CTRL(rw), GTM_TOM_TGC1_OUTEN_STAT(w), SCU_CCUCON0(r), SCU_EICR(w), SCU_IGCR(w), SCU_OSCCON(r), SCU_SEICON0(rw), SCU_SYSPLLCON0(r), SCU_SYSPLLCON1(r), SRC_VADC_G_SR(w), STM_TIM0(r) *Note : The list includes all the SFRs accessed in the context of the API. It lists the SFRs accessed by the driver and called interfaces from other drivers. During runtime, the SFRs accessed from this list may vary based on configuration and execution context.*
Autosar Version	Applicable for Autosar versions 4.2.2 and 4.4.0.

1.3.7 Callout

The driver does not support any callout functions.

1.3.8 Errors Handling

This section describes the various errors reported by the ADC driver.

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
ADC_E_UNINIT: Error code is reported if initialization for the current core or the required cores has not been done before invoking the API.	AUTOSAR	0x0A	DET_SAFETY	0x0A	DET_SAFETY
ADC_E_BUSY: Error code is reported if the operation intended by the API cannot be executed because the requested AdcChannel group or hardware resources used by the AdcChannel group are currently busy.	AUTOSAR	0x0B	DET_SAFETY	0x0B	RUNTIME

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
ADC_E_IDLE: Error code is reported if the operation intended by the API cannot be executed because the requested AdcChannel group is currently in the IDLE state.	AUTOSAR	0x0C	DET_SAFETY	0x0C	RUNTIME
ADC_E_ALREADY_INITIALIZED: Error code is reported if initialization is requested from a core, which is already in the initialized state.	AUTOSAR	0x0D	DET_SAFETY	0x0D	DET_SAFETY
ADC_E_PARAM_CONFIG: Error code is reported if the API is invoked with invalid configuration pointer.	AUTOSAR	0x0E	DET_SAFETY	NA	NA
ADC_E_PARAM_POINTER: Error code is reported if the API is invoked with null pointer as a parameter. *Note: For applicability of this DET to Adc_Init API, refer the note in the description of the API.*	AUTOSAR	0x14	DET_SAFETY	0x14	DET_SAFETY
ADC_E_PARAM_GROUP: Error code is reported if the API is invoked with an invalid group ID.	AUTOSAR	0x15	DET_SAFETY	0x15	DET_SAFETY
ADC_E_WRONG_CONV_MODE: Error code is reported if the Adc_EnableHardwareTrigger or Adc_DisableHardwareTrigger API is called for a group with the conversion mode configured as continuous.	AUTOSAR	0x16	DET_SAFETY	0x16	DET_SAFETY
ADC_E_WRONG_TRIGG_SRC: Error code is reported if an AdcChannel group with trigger source as software is invoked through a hardware trigger API or an AdcChannel group with trigger source as hardware is invoked through a software trigger API.	AUTOSAR	0x17	DET_SAFETY	0x17	DET_SAFETY

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
ADC_E_NOTIF_CAPABILITY: Error code is reported if enable or disable notification functions is invoked with an AdcChannel group whose configuration set has no notification or NULL_PTR configured as notification.	AUTOSAR	0x18	DET_SAFETY	0x18	DET_SAFETY
ADC_E_BUFFER_UNINIT: Error code is reported if a conversion start request is placed while the result buffer pointer is not initialized.	AUTOSAR	0x19	DET_SAFETY	0x19	DET_SAFETY
ADC_E_NOT_DISENGAGED: Error code is reported if the API is invoked while one or more AdcChannel group is not in IDLE state or has notification enabled.	AUTOSAR	0x1A	DET_SAFETY	0x1A	RUNTIME
ADC_E_POWER_STATE_NOT_SUPPORTED: Error code is reported if the API is invoked with an unsupported power state.	AUTOSAR	0x1B	DET_SAFETY	0x1B	DET_SAFETY
ADC_E_PERIPHERAL_NOT_PREPARED: Error code is reported if the ADC is not prepared for the target power state.	AUTOSAR	0x1D	DET_SAFETY	0x1D	DET_SAFETY
ADC_E_CONVERTER_OFF: Error code is reported if start conversion or enable hardware trigger is invoked while the converter is in OFF state.	IFX	0x30	DET_SAFETY	0x30	DET_SAFETY
ADC_E_CORE_NOT_CONFIGURED: Error code is reported if the API is invoked from a core which has no ADC hardware group allocated.	IFX	0x64	DET_SAFETY	0x64	DET_SAFETY
ADC_E_CORE_GROUP_MISMATCH: Error code is reported if the API is invoked for an ADC hardware group that is not configured to be used by the current core.	IFX	0x65	DET_SAFETY	0x65	DET_SAFETY

Error Name: Description	Source	Error ID (AS422)	Type (AS422)	Error ID (AS440)	Type (AS440)
ADC_E_MASTER_CORE_UNINIT : Error code is reported if a slave core initialization is invoked prior to initialization of the master core.	IFX	0x66	DET_SAFETY	0x66	DET_SAFETY
ADC_E_SLAVE_CORE_INIT: Error code is reported if de-initialization from master core is invoked while any of the slave cores is still in the initialized state.	IFX	0x67	DET_SAFETY	0x67	DET_SAFETY
ADC_SE_CALIB_ONGOING: Error code is reported if start conversion or enable hardware trigger is invoked while start-up calibration is ongoing.	IFX	0xC8	SAFETY	0xC8	SAFETY
ADC_SE_INT_PLAUSIBILITY: Error code is reported if an unintended interrupt is triggering the ISR.	IFX	0xC9	SAFETY	0xC9	SAFETY
ADC_SE_POWER_STATE_INVALID: Error code is reported if power state read from the hardware is invalid (not configured).	IFX	0xCA	SAFETY	0xCA	SAFETY
ADC_SE_PARAM_KERNEL: Error code is reported if the kernel ID passed is not configured or it is beyond the available range in the hardware.	IFX	0xCB	SAFETY	0xCB	SAFETY
ADC_E_CLC_FAILURE: Error code is reported when enabling of CLC (module clock) fails.	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error
ADC_E_CONV_STOP_TIME_FAILURE: Error code is reported when an ongoing conversion cannot be stopped due to unknown hardware failure.	IFX	Assigned by DEM	Production Error	Assigned by DEM	Production Error

1.3.9 Deviations and limitations

This section describes the deviations and limitations of the ADC driver.

1.3.9.1 Deviations

This section describes the deviations of the ADC driver.

1.3.9.1.1 Software specification deviations

This section describes the deviations from software specification.

Table 191 Known deviations

Reference	Deviation
Access of SRC registers	The ADC driver updates the SRC registers of EVADC (MODULE_SRC.VADC.G[x].SRy) to clear the pending interrupt requests. Refer to the Key architectural consideration section for information on how to update this shared resource simultaneously by the application and the MCAL driver.
Address and Data CRC in DMA mode	Since the Data CRC and Address CRC features of DMA are not used for ADC driver, the user shall ensure that, while using the DMA mode a plausibility check of the conversion result is performed either by redundancy or by other means.
Safety error for unintended service request	Refer to the section Reporting of unintended service requests.
For all requirements related to Production/Runtime errors	Reporting of Production error: Dem_ReportErrorStatus is done through Mcal_Wrapper_Dem_ReportErrorStatus interface for AUTOSAR 4.2.2 and Dem_SetEventStatus is done through Mcal_Wrapper_Dem_SetEventStatus interface for AUTOSAR 4.4.0. Reporting of Runtime error: Det_ReportRuntimeError is done through Mcal_Wrapper_Det_ReportRuntimeError interface. This is applicable for only AUTOSAR 4.4.0. All production and runtime related datatypes and modified interfaces inclusion shall be done via Mcal_Wrapper.h

1.3.9.1.2 AMDC Violations

The ADC driver does not have any AMDC violations.

1.3.9.1.3 VSMD Violations

This section describes the violations reported by the EB VSMD checker tool with respect to AUTOSAR.

Table 192 Violations reported by VSMD checker tool for TpsEcuc_06051_ASR41

Rule ID:	TpsEcuc_06051_ASR41
VSMD Node(s):	/AURIX2G/EcucDefs/Adc/AdcConfigSet/AdcHwUnit/AdcChannel/AdcChannelId /AURIX2G/EcucDefs/Adc/AdcConfigSet/AdcHwUnit/AdcHwUnitId

(table continues...)

Table 192 (continued) Violations reported by VSMD checker tool for TpsEcuc_06051_ASR41

Description:	The implementationConfigClass of an EcucParameterDef or EcucAbstractReferenceDef in VSMD shall be the same or higher (where PreCompile configuration class is considered to be the lowest and PostBuild the highest) as in StMD with respect to the selected subset defined by the actually implemented supportedConfigVariant.
Additional Information:	The value assigned to AdcHwUnitId or AdcChannelId may change only at pre-compile time. Hence, the implementationConfigClass of AdcHwUnitId and AdcChannelId is pre-compile instead of post-build.

Table 193 Violations reported by VSMD checker tool for TpsEcuc_08032

Rule ID:	TpsEcuc_08032
VSMD Node(s):	/AURIX2G/EcucDefs/Adc/AdcConfigSet/AdcHwUnit/AdcChannel/AdcChannelId /AURIX2G/EcucDefs/Adc/AdcConfigSet/AdcHwUnit/AdcHwUnitId
Description:	If the EcucModuleDef.postBuildVariantSupport is set to true and the postBuildVariantValue for an EcucParameterDef or an EcucAbstractReferenceDef in this EcucModuleDef in the StMD is set to true, the corresponding VSMD shall also set it to true.
Additional Information:	The value assigned to AdcHwUnitId or AdcChannelId may change only at pre-compile time. Hence, the postBuildVariantValue of AdcHwUnitId and AdcChannelId is set to false.

Table 194 Violations reported by VSMD checker tool for TpsEcuc_08033

Rule ID:	TpsEcuc_08033
VSMD Node(s):	/AURIX2G/EcucDefs/Adc/AdcConfigSet/AdcHwUnit/AdcPrescale
Description:	If the EcucModuleDef.postBuildVariantSupport is set to true and the postBuildVariantMultiplicity for an EcucParameterDef or an EcucAbstractReferenceDef in this EcucModuleDef in the StMD is set to true, the corresponding VSMD shall also set it to true.
Additional Information:	The divider for generating analog frequency is mandatory for performing the conversions, hence both lower and upper multiplicity of AdcPrescale parameter is changed to 1.

Table 195 Violations reported by VSMD checker tool for TpsEcuc_08038

Rule ID:	TpsEcuc_08038

(table continues...)

Table 195 (continued) Violations reported by VSMD checker tool for TpsEcuc_08038

VSMD Node(s):	/AURIX2G/EcucDefs/Adc/AdcConfigSet/AdcHwUnit/AdcChannel/AdcChannelId /AURIX2G/EcucDefs/Adc/AdcConfigSet/AdcHwUnit/AdcHwUnitId
Description:	If the valueConfigClass attribute for an EcucParameterDef or an EcucAbstractReferenceDef is defined in the StMD,valueConfigClass.configClass for each valueConfigClass.configVariant in the VSMD shall be the same or higher as in the StMDwith respect to the selected subset defined by the actually implemented supportedConfigVariant of the corresponding EcucModuleDef.
Additional Information:	The value assigned to AdcHwUnitId or AdcChannelId may change only at pre-compile time. Hence, the valueConfigClass.configClass of AdcHwUnitId and AdcChannelId is pre-compile instead of post-build.

1.3.9.2 Limitations

This section describes the limitations of the ADC driver.

Table 196 Known limitations

Reference	Limitation
Input Class for channels in synchronous conversion	The user must configure global input classes (for CHCTR.ICLSEL through the AdcInputClassSelection configuration parameter) for the master and slave channels so that the conversion properties and timings for the master and slave channels are the same. If the global input class is not used, the properties of the kernel-specific input classes used for the master and slave channels must be identical.
Order of conversion for groups with same priority in HW-SW priority mode	The order of conversion for group with same priority in the hardware-software priority is not first come first serve always. The order is determined by hardware arbiter runtime. Among the groups with the same priority the one installed on RS0 is always executed first followed by the one on RS1 and then RS2.
Input parameter to `Adc_InitCheck()` API is unsued	Input parameter to `Adc_InitCheck()` API is unused, instead parameter used during `Adc_Init()` API is used for InitCheck evaluation.

(table continues...)

Table 196 (continued) Known limitations

DEM: ADC_E_CONV_STOP_TIME_FAILURE	The driver continues normal operation after reporting the Production error failure for ADC_E_CONV_STOP_TIME_FAILURE. The user is expected to increase the timeout in the configuration parameter AdcMaxChConvTimeCount to avoid the Production error failure. The scenario occurs when an AdcChannel Group is stopped and there are other pending channels in the groups in the queue and the timeout is not configured correctly.
The diagnostic feature is not supported with the hardware Master/Slave feature	ADC Converter Diagnostic feature cannot be used with synchronous conversion of Master/Slave configuration across different ADC hardware Units. The alias feature configuration support is provided only at the group level.
The diagnostic feature is not supported with DMA feature	ADC Converter Diagnostic requires the usage of the ADC Hw Priority mechanism and the DMA feature is supported only when the ADC HW Priority is set to none. Therefore, ADC Converted Diagnostic feature cannot be used with DMA feature enabled.

Revision history

Table 197 Revision History

Date	Version	Description
2023-06-16	5.0	Document is released.
2023-06-14	4.1	- In section 1.1.4.1, DEM module has been removed and is replaced with Mcal_Wrapper module. - In section 1.1.4.1, Runtime information is removed from DET module. - DEM has been modified to Production error where applicable. - Updated Figure 1, DEM Module is removed and Mcal_Wrapper Module is added. - Updated section 1.1.3.1 to include Mcal_Wrapper.h and removed Dem.h. - Assumption of Use added for "InitCheck Sequence" and "ConfigPtr passed to InitCheck" - Updated the Description of AdcDemEventParameterRefs container. - E_NOT_OK Description updated for Adc_InitCheck API in section 1.3.3.20 - ASIL Level has been updated to Safety level in Section 1.3.3 and 1.3.6 - Updated the section 1.3.9.1.1: Software Specification Deviations for Autosar requirements. Updated Reference from "DEM header file" to "For all requirements related to Production/Runtime errors". Updated Description of "DEM header file" to add Mcal_Wrapper Module Information. - Limitation is added for Diagnostic feature is not supported with DMA feature
2022-08-19	4.0	Document is released.
2022-08-16	3.1	Limitation is added for Diagnostic feature is not supported with the hardware Master/Slave feature
2021-11-08	3.0	Document is released.
2021-11-03	2.1	- Config variant attribute table information is removed and added this information in 'Configuration interfaces' section - Alias feature is added in section 'Key architectural considerations'.
2020-11-10	2.0	Document is released.
2020-11-09	1.1	- Redundant AoUs deleted. - SFR access information for APIs updated.
2020-08-13	1.0	Document is released.
2020-08-12	0.1	- Initial Draft - The ADC driver chapter moved from MC-ISAR_TC3xx_UM_Basic to this document - EMUX feature related updates in Key architectural considerations and Configuration interfaces section - Diagnostic features related updates related updates in Key architectural considerations and Configuration interfaces section - VSMD Violations justification table - Deviation added for DEM Header file - Limitation added for ADC_E_CONV_STOP_TIME_FAILURE DEM

Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2023-06-16

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-ocr1484806431059

Important notice

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

Warnings

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.