MCAL ADC Module Software Design Document

Document Version : 100
Document Owner : Texas Instruments
Document Status : Published
Last Approval Date : Mar 15, 2022

- Revision History
- Terms and Abbreviations
- Introduction
  - Overview
  - Purpose and Scope
  - Module Overview
  - Requirements
    - Features Supported
    - Features Not Supported / NON Compliance
  - Assumptions
  - Constraints
  - Hardware and SW platforms
  - Dependencies
  - Stakeholders
  - References
- Design Description
  - Fundamental Operation
  - Different Input Values
    - Single-ended input
    - Differential input
  - Conversion Modes of ADC
    - One Shot Mode
    - Continuous Mode
  - Trigger source
    - Software API Call
    - Hardware Event

    - Assess Modes
        - Adc_GetStreamLastPointer
    - Priority Handling and Queuing Operations
    - Interrupt Service Routines
    - Conversion Mode Example
    - Directory Structure
    - Configurator
        - NON Standard configurable parameters
    - Debug Information
    - Error Classification
        - Development Errors
        - Error Detection
        - Error notification (DET)
        - Runtime Errors
        - Error notification (DEM)
- Implementation Details
    - Data structures and resources
    - Dynamic Behavior - Control Flow Diagram
    - Dynamic Behavior - Data Flow Diagram
    - Application Parameters
    - Safety Diagnostic Features
- Low Level Definitions
    - Driver API's
        - Adc_Init
        - Adc_SetupResultBuffer
        - Adc_DeInit

        - Adc_StartGroupConversion
        - Adc_StopGroupConversion
        - Adc_ReadGroup
        - Adc_EnableGroupNotification
        - Adc_DisableGroupNotification
        - Adc_GetGroupStatus
        - Adc_GetStreamLastPointer
        - Adc_GetVersionInfo
        - Adc_RegisterReadback
        - Adc_EnableInternalDiagnostic
        - Adc_DisableInternalDiagnostic
- Performance Objectives
    - Resource Consumption Objectives
    - Critical timing and Performance
    - ADC Conversion Mode
- Testing Guidelines
- Template Revision History

1 Revision History

Version	Date	Author	Document Status	Comments
0.1	09 Feb 2019	Sunil M S	Draft	First Version
0.2	18 Feb 2019	Sunil M S	In Review	Addressed Review Comments
0.3	19 Apr 2021	Harish S	In Review	SDD with Aspice Format
0.4	05 Aug 2021	Nikki S	In Review	Adding Design IDs
0.5	19 Aug 2021	Harish S	In Approval	Fixing Review Comments
1.0	07 Sep 2021	Nikki S	Published	Upload to Galileo
1.1	24 Jan 2022	Nikki S	In Review	JACINTOREQ-1870

v100	04 Mar 2022	Nikki S	Comola Workflow Added	Review Comments addressed.

2 Terms and Abbreviations

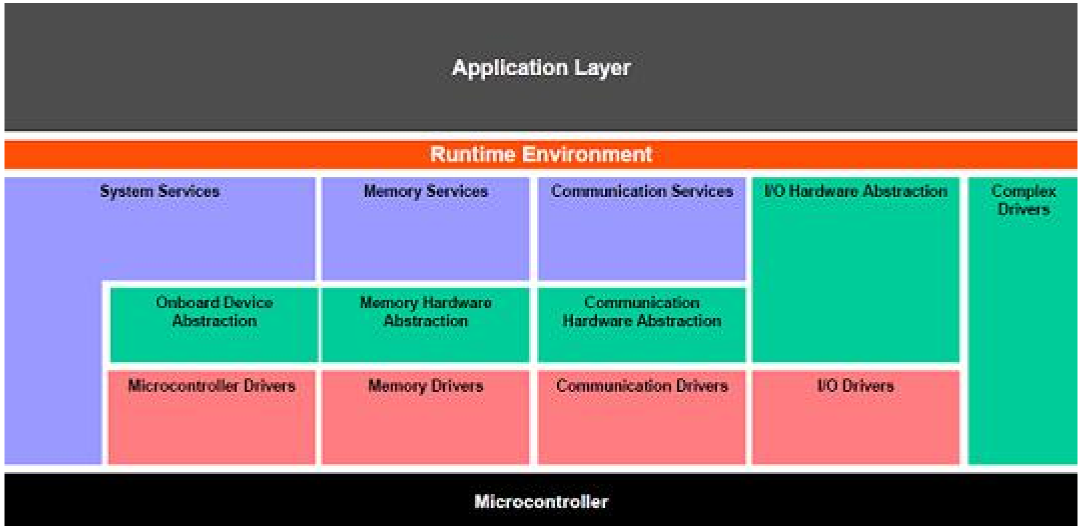
Abbreviation /Term	Meaning / Explanation
ADC	Analog-to-Digital Converter
AFE	Analog Front End Converter
API	Application Programming Interface
BSW	Basic Software
DAR	Decision Analysis and Resolution
DEM	Development Error Tracer
DET	Diagnostic Event Manager
FSM	Finite State Machine

Abbreviation /Term	Meaning / Explanation
HW	Hardware
I/O	Input Output
MCAL	Microcontroller Abstraction Layer
MSPS	Million Samples Per Second
RTE	Runtime Environmental
SAR	Successive-Approximation-Register
SBL	Serial Bootloader
SW	Software

3 Introduction

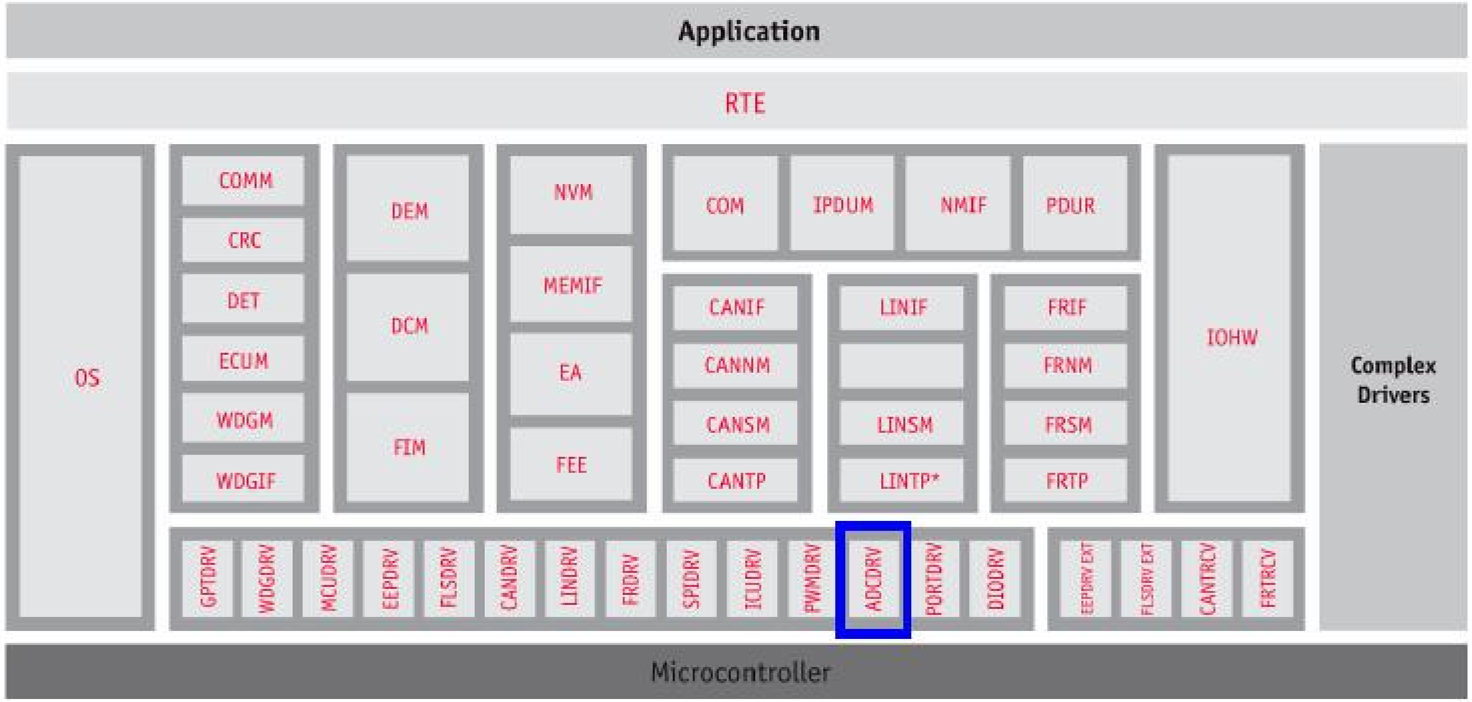
3.1 Overview

The figure below depicts the AUTOSAR layered architecture as 3 distinct layers, Application, Runtime Environment (RTE) and Basic Software (BSW). The BSW is further divided into 4 layers, Services, Electronic Control Unit Abstraction, MicroController Abstraction (MCAL) and Complex Drivers.



AUTOSAR Architecture

MCAL is the lowest abstraction layer of the Basic Software. It contains software modules that interact with the Microcontroller and its internal peripherals directly. Adc driver is part of the I/O Drivers (block, shown above). Below shows the position of the Adc driver in the AUTOSAR Architecture.



AUTOSAR Architecture – ADC MCAL

The ADC module initializes and controls the internal Analogue Digital Converter Unit(s) of the microcontroller. It provides services to start and stop a conversion respectively to enable and disable the trigger source for a conversion. Furthermore it provides services to enable and disable a notification mechanism and routines to query the status and result of a conversion.

The ADC module works on so called ADC Channel Groups, which are build from so called ADC Channels. An ADC Channel Group combines an analogue input pin (ADC Channel), the needed ADC circuitry itself and conversion result register into an entity that can be individually controlled and accessed via the ADC module.

3.2 Purpose and Scope

This document specifies the implementation of MCAL driver for the module ADC, using hardware IP "adc12_16ffc_10_rel.1.0.x". All requirements for implementing the driver are mapped in the document itself. Also, document holds the detailed information regarding the MACROs, structures and APIs for the implementation.

3.3 Module Overview

The analog-to-digital converter (ADC) module is a successive-approximation-register (SAR) general purpose analog-to-digital converter. Below listed are some of the key features provided. Please refer to the specific SoC User Manual for the detailed TRM.

- 4 MSPS rate at 60 MHz sample clock can be operated with either single-ended or differential input.
- Separate positive and negative ADC reference in case the maximum analog input level is smaller than the analog supply.
- Built-in functional safety self-tests
- Programmable Finite State Machine (FSM) sequencer that supports the following steps:
    - Software register bit for start of conversion
    - Optional start of conversion (SOC), hardware synchronized to external hardware event
    - Single conversion (one-shot)
    - Continuous conversions
    - Sequence through all input channels based on a mask

  - Programmable open delay before sampling each input
  - Programmable sampling delay for each input
  - Programmable averaging of input samples - 16, 8, 4, 2, or 1
  - Store data in either of two FIFO – 256-word 16-bit RAM
  - Support for servicing FIFOs via DMA or CPU
  - Programmable DMA request event (for each FIFO)
  - Dynamically enable or disable channel inputs during operation
  - Software register bit for end of conversion
- Support for the following interrupts and status, with masking:
  - Interrupt after a sequence of conversions (all non-masked input channels)
  - Interrupt for FIFO threshold levels
  - Interrupt if sampled data is out of a programmable range
  - Interrupt for FIFO overflow and underflow conditions
  - Status bit to indicate if ADC is busy converting

3.4 Requirements

The ADC driver shall implement as per requirements detailed in 4(BSW General Requirements / Coding guidelines),1(AUTOSAR 4.3.1). It’s recommended to refer1((AUTOSAR 4.3.1)for clarification.

Note: Please refer the Reference Section 3.10.

3.4.1 Features Supported

Below listed are some of the key features that are expected to be supported

- Initialization and de-initialization of internal analog-to-digital conversion unit
- Grouping of ADC channels to so called ADC channel groups
- Starting and stopping conversions of software triggered channel groups
- Streaming functionality (storage of multiple results)
- Accessing conversion results in two different ways: by value using Adc_ReadGroup() API and by reference using Adc_GetStreamLastPointer() API
- Getting information about the status of a group
- Queuing of conversion requests
- Priority levels and abort/restart and suspend/resume of channel groups
- The following combinations of modes are supported by the ADC driver
    - One-shot, Software Trigger, Single Access Mode
    - Continuous, Software Trigger, Circular Single Access Mode
    - Continuous, Software Trigger, Linear Single Access Mode (similar to one-shot mode)
    - Continuous, Software Trigger, Circular Streaming Access Mode
    - Continuous, Software Trigger, Linear Streaming Access Mode
- Supports additional configuration parameters, refer section (NON Standard configurable parameters) & (Adc_RegisterReadback)

3.4.2 Features Not Supported / NON Compliance

- Enabling and disabling hardware trigger of hardware triggered channel groups
- Management of hardware low-power states and the corresponding API
- All limit checking requirements
- The following combinations of modes are not supported by the ADC driver
    - All hardware trigger modes
    - One-shot, Software Trigger, Stream Access (Circular) Mode.
    - Optional ADC module specific clock prescale factor as hardware doesn’t support.

- Standard AUTOSAR ADC specification , categorizes few BSW General Requirements as non-requirements, please refer MCAL-3431 for details.

3.5 Assumptions

Below listed are assumed to valid for this design/implementation, exceptions and other deviations are listed for each explicitly. Care should be taken to ensure these assumptions are addressed.

1. The functional clock to the ADC module is expected to be ON before calling any ADC module API.
2. The ADC driver as such doesn’t perform any PRCM programming to get the functional clock.
3. The clock-source selection for ADC is not performed by the ADC driver, other entities such as SBL, MCAL module MCU shall perform the same.
4. Other entities such as SBL, MCAL module PORT module shall configure the port pins used by the ADC module.

Note that assumptions above are specified by AUTOSAR ADC specification.

3.6 Constraints

None

3.7 Hardware and SW platforms

Hardware Platform

- Refer to specified SoC User Manual to check if ADC module is supported.

Software Platforms

- Bare-Metal

3.8 Dependencies

SBL/GEL files shall be used to configure port pins and clocks as TI MCAL does not include MCU and PORT driver module.

SBL

- **FICLK:**Is interface clock required for internal operation of the peripheral. This is not expected to change and typically programmed by SBL,
- **SYS_CLK:** MCU_ADC system clock. Must be 6x times the MCU_ADC0_SMPL_CLK clock.
- **SMPL_CLK:** MCU_ADC sampling clock. It has 4 different clock sources.

Design Identifier	Description
MCAL-5700	SWS_Adc_00247 : Adc_Init PORT driver dependency
MCAL-5834	SWS_Adc_00248 : Adc_Init MCU driver dependency

3.9 Stakeholders

- Developers
- Test Engineers
- Customer Integrator

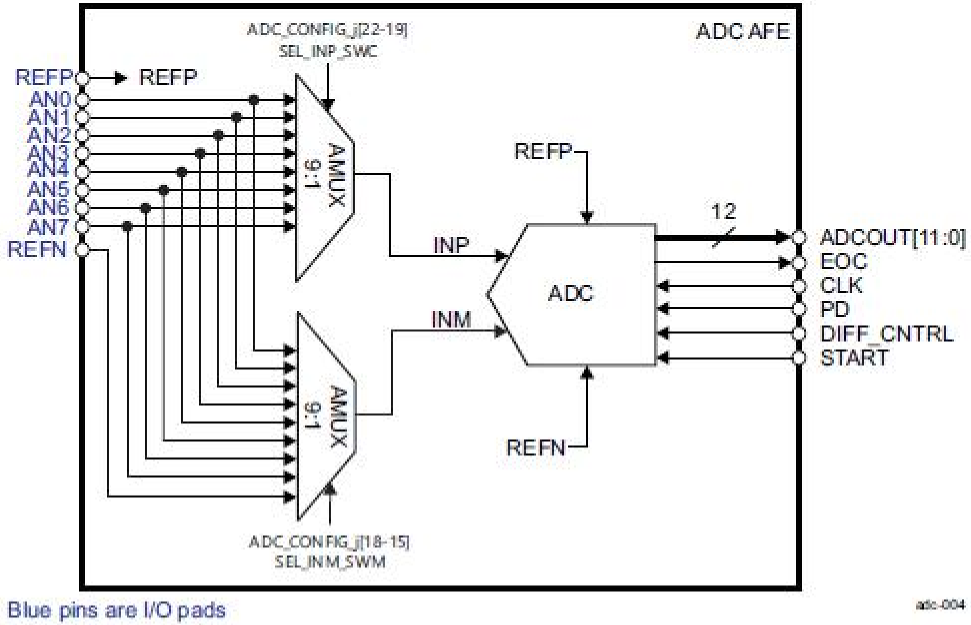
3.10 References

	Specification	Comment / Link
1	AUTOSAR 4.3.1	AUTOSAR Specification for ADC Driver.
2	BSW General Requirements / Coding guidelines	Autosar and Coding guidelines for the Mcal drivers.
3	Software Product Specification (SPS)	Product Functional requirements.
4	Software Architecture	Mcal Software Architecture.

4 Design Description

4.1 Fundamental Operation

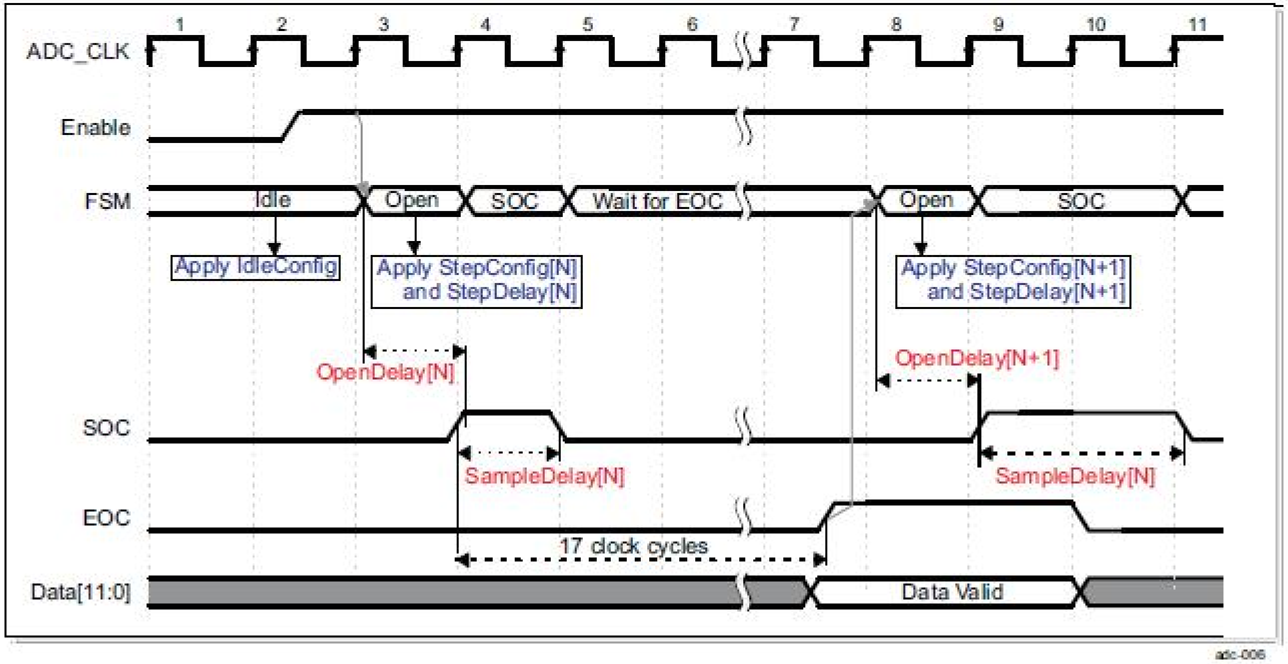
ADC is an 8 channel general purpose SAR ADC controller which supports 12 bit conversion samples from an analog front end converter (AFE).



ADC Functional Block Diagram

Before enabling the module, the user must first program the step configuration registers in order to configure a channel input to be sampled. There are 16 programmable step configuration registers which are used by the sequencer to control which switches to turn on or off (inputs to the AFE), which channel to sample, and which mode to use (HW triggered or SW enabled, one-shot or continuous, averaging, where to save the FIFO data, etc). The user can program the delay between driving the inputs to the AFE and the time to send the start of conversion signal. This delay is called open delay(and can also be programmed to zero). The user also has control of the sampling time (width of the start of conversion signal) to the AFE which is called the sample delay. Each channel input is configured independently via the Step Delay register.

The ADC sequencer is completely controlled by software and behaves accordingly to how the ADC_CONFIG_j registers are programmed. A step is the general term for sampling a input. It is defined by the programmer who decides which input values to send to the AFE as well as how and when to sample a input. If a step is configured as software (SW) controlled when the ADC is first enabled, the sequencer will then wait for a ADC_STEPENABLE register bit to turn on. After a step is enabled, the sequencer will start with the lowest step (1) and continue until step (16). If a step is not enabled, then sequencer will skip to the next step. If all steps are disabled, then the sequencer will remain in the IDLE state. An ENDOFSEQUENCE interrupt is generated after the last active step is completed before going back to the IDLE state. The ENDOFSEQUENCE interrupt does not mean data is in the FIFO.We should use FIFO interrupts.



Example Timing Diagram for Sequencer

Using the minimum delay values, the ADC can sample at 18 ADC clocks per sample. Once the ADC is enabled and assuming at least one step is active, the FSM will transition from the IDLE state and apply the first active ADC_CONFIG_j and ADC_DELAY_j register settings. It is possible for the OpenDelay value to be 0, and the FSM will immediately skip to the SampleDelay state. The AFE will begin sampling of the analog voltage on high level of the SOC signal. Voltage sampling duration is 4 clock cycles long. After the AFE is finished converting the input data (13 more cycles later), the end of conversion (EOC) signal is sent and the FSM will then apply the next active step. This process is repeated and continued (from step 1 to step 16) until the last active step is completed.

4.2 Different Input Values

ADC can be operated with either single-ended or differential input values.

4.2.1 Single-ended input

Single-ended inputs are generally sufficient for most applications. In single-ended applications, all signals are referenced to a common ground at the ADC.

4.2.2 Differential input

An ADC with fully-differential inputs digitizes the differential analog input voltage (REFP – REFN) over a span of full scale voltage. Fully-differential inputs offer wider dynamic range and better SNR performance over single-ended.

4.3 Conversion Modes of ADC

4.3.1 One Shot Mode

The ADC module shall support the conversion mode One-shot Conversion for all ADC Channel groups. One-shot conversion means that exactly one conversion is executed for each channel configured for the group being converted.

Design Identifier	Description
MCAL-5865	SWS_Adc_00380 : ADC Module conversion mode one-shot
MCAL-5833	SWS_Adc_00338 : ADC Additional software conversion request
MCAL-5805	SWS_Adc_00384 : ADC Module Environment check for conversion completion

4.3.2 Continuous Mode

The ADC module shall support the conversion mode Continuous Conversion for all ADC Channel groups with trigger source software. Continuous Conversion means that after the conversion has been completed, the conversion of the whole group is repeated. The conversions of the individual ADC channels within the group as well as the repetition of the whole group don't need any additional trigger events to be executed. Converting the individual channels within the group can be done sequentially or in parallel depending on hardware and/or software capabilities.

Design Identifier	Description
MCAL-5764	SWS_Adc_00381 : ADC Module conversion mode continuous conversion
MCAL-5833	SWS_Adc_00338 : ADC Additional software conversion request
MCAL-5805	SWS_Adc_00384 : ADC Module Environment check for conversion completion

4.4 Trigger source

4.4.1 Software API Call

The ADC module shall support the start condition Software API Call for all conversion modes. The trigger source Software API Call means that the conversion of an ADC Channel group is started/stopped with a service provided by the ADC module.

Design Identifier	Description
MCAL-5756	SWS_Adc_00356 : ADC Module start condition Software API Call

4.4.2 Hardware Event

The ADC module shall support the start condition Hardware Event for groups configured in One-Shot conversion mode. The trigger source Hardware Event means that the conversion of an ADC Channel group can be started by a hardware event, e.g. an expired timer or an edge detected on an input line.

Design Identifier	Description
MCAL-5760	SWS_Adc_00138 : ADC Multiple HW unit support

4.5 Assess Modes

4.5.1 Adc_GetStreamLastPointer

The ADC module shall support result access using the API function Adc_GetStreamLastPointer. Calling Adc_GetStreamLastPointer informs the user about the position of the group conversion results of the latest conversion round in the result buffer and about the number of valid conversion results in the result buffer. The result buffer is an external buffer provided from the application.

Design Identifier	Description
MCAL-5731	SWS_Adc_00382 : ADC Module support Adc_GetStreamLastPointer

The ADC module shall support result access using the API function Adc_ReadGroup. Calling Adc_ReadGroup copies the group conversion results of the latest conversion round to an application buffer which start address is specified as API parameter of Adc_ReadGroup.

Design Identifier	Description
MCAL-5710	SWS_Adc_00383 : ADC Module support Adc_ReadGroup

4.6 Priority Handling and Queuing Operations

Priority mechanism is implemented using a pure software function as hardware priority mechanism is not supported by the ADC module. This means only ADC_PRIORITY_NONE and ADC_PRIORITY_HW_SW options are supported ADC_PRIORITY_HW is not supported.

Priority mechanism can be statically enabled or disabled using the configuration macro ADC_PRIORITY_IMPLEMENTATION which can be changed during configuration step. When priority mechanism is enabled and when a high priority group is started when a lower priority group is in progress for the same ADC unit, the driver stops the current group and schedules the high priority group. Once the high priority group conversion is completed (either implicitly or explicitly) the driver will re-schedule the lower priority group. While restarting the group, the driver always starts from channel 0 of the group i.e. irrespective of whether the group parameter 'groupReplacement' is ADC_GROUP_REPL_ABORT_RESTART or ADC_GROUP_REPL_SUSPEND_RESUME the driver always does restart operation. Resume operation is not supported.

This driver also supports queuing mechanism to queue multiple requests to the driver for the same ADC unit. Queuing mechanism can be statically enabled or disabled using the configuration macro ADC_ENABLE_QUEUING which can be changed during the configuration step. When priority mechanism is enabled and when queue is enabled, the driver processes the requests on a first come first serve basis.When queuing is disabled, the driver will raise a development error. When any group is started when the hardware unit is busy converting another group,it queues that group and returns without any operation.

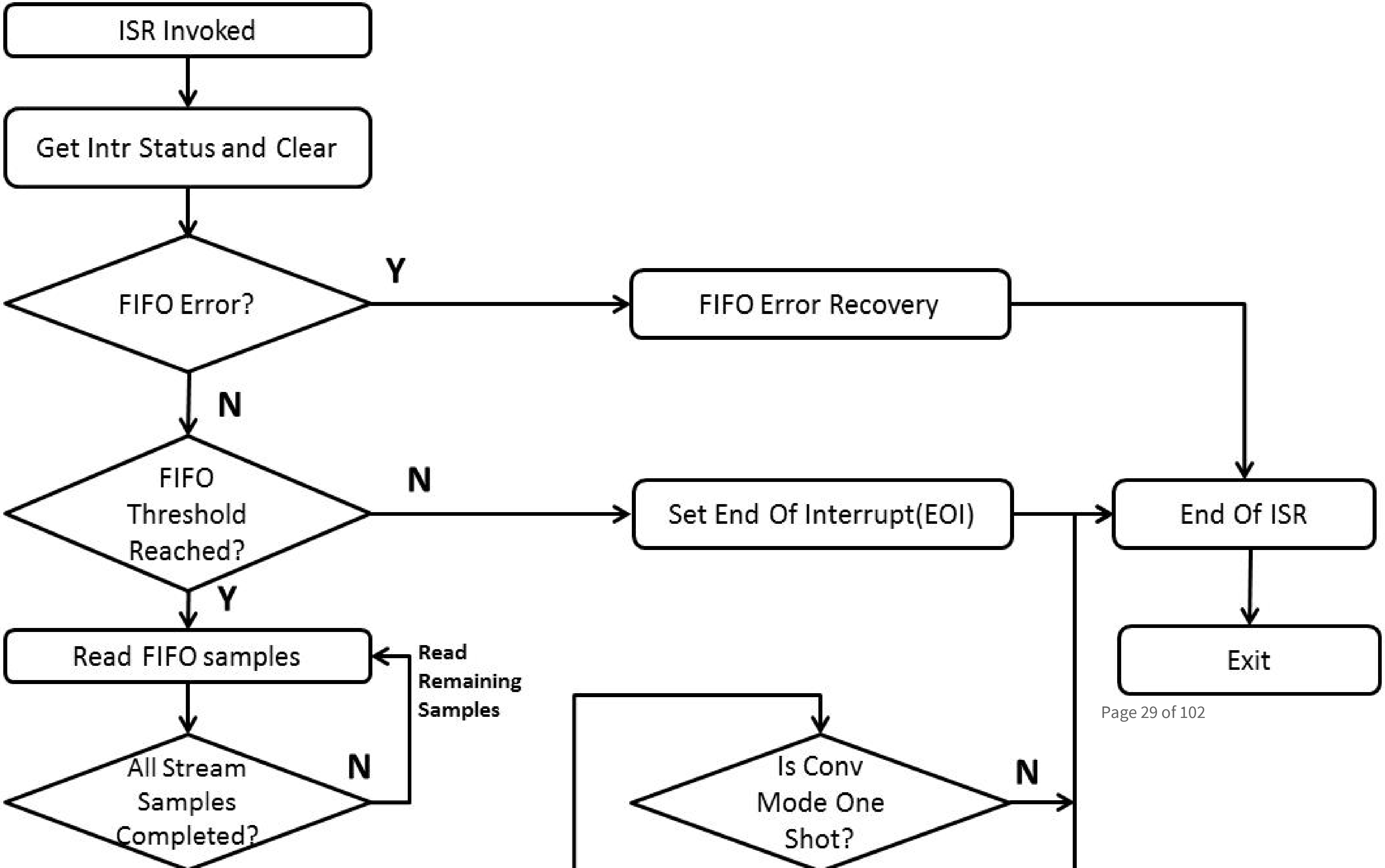
Design Identifier	Description
MCAL-5882	SWS_Adc_00315 : ADC Priority Mechanism static configuration
MCAL-5837	SWS_Adc_00332 : ADC Priority Mechanism queuing of requested groups on priority
MCAL-5814	SWS_Adc_00288 : ADC Channel group priority level
MCAL-5807	SWS_Adc_00522 : Adc_PriorityImplementationType
MCAL-5791	SWS_Adc_00417 : ADC Priority Mechanism same level
MCAL-5768	SWS_Adc_00289 : ADC Priority Mechanism levels

Design Identifier	Description
MCAL-5752	SWS_Adc_00311 : ADC Priority Mechanism queuing of requested groups
MCAL-5733	SWS_Adc_00340 : ADC Priority Mechanism ADC_PRIORITY_HW_SW
MCAL-5712	SWS_Adc_00516 : Adc_GroupPriorityType
MCAL-5701	SWS_Adc_00312 : ADC Priority Mechanism lowest priority
MCAL-5692	ECUC_Adc_00287 : AdcGroupPriority
MCAL-5643	ECUC_Adc_00393 : AdcPriorityImplementation
MCAL-5837	SWS_Adc_00332 : ADC Priority Mechanism queuing of requested groups on priority
MCAL-5752	SWS_Adc_00311 : ADC Priority Mechanism queuing of requested groups
MCAL-5722	SWS_Adc_00335 : ADC Queuing Mechanism queuing of requested groups
MCAL-5714	SWS_Adc_00333 : ADC Queuing Mechanism
MCAL-5678	ECUC_Adc_00391 : AdcEnableQueuing

Design Identifier	Description
MCAL-5831	SWS_Adc_00345 : ADC Channel group Suspend/Resume

4.7 Interrupt Service Routines

For each of the configured hardware units, one interrupt service routine has to be mapped. The Integrator has to map the interrtable risk of physical injury or of dupt service routines to the interrupt sources of the respective ADC unit interrupt. The supported ISR’s are part of the Adc_Irq.h file.


ISR Invoked
Get Intr Status and Clear
FIFO Error?
Y
FIFO Error Recovery
N
FIFO Threshold Reached?
N
Set End Of Interrupt(EOI)
End Of ISR
Y
Read FIFO samples
Read Remaining Samples
Exit
All Stream Samples Completed?
N
Is Conv Mode One Shot?
N

ISR Flow Chart

Design ID	Description
MCAL-5840	SWS_Adc_00416 : IoHwAb_AdcNotification Notification mechanism reenabled
MCAL-5801	SWS_Adc_00084 : IoHwAb_AdcNotification Notification call-back configuration
MCAL-5795	SWS_Adc_00060 : ADC Group Notification function
MCAL-5789	SWS_Adc_00413 : ADC Api's Reentrancy
MCAL-5757	SWS_Adc_00430 : ADC Interrupted Channel group configuration
MCAL-5753	SWS_Adc_00080 : IoHwAb_AdcNotification Notification call-back NULL_PTR
MCAL-5747	SWS_Adc_00310 : ADC Channel group Abort/Restart
MCAL-5739	SWS_Adc_00415 : ADC module's integrity check off
MCAL-5708	SWS_Adc_00414 : ADC module's integrity check on

4.8 Conversion Mode Example

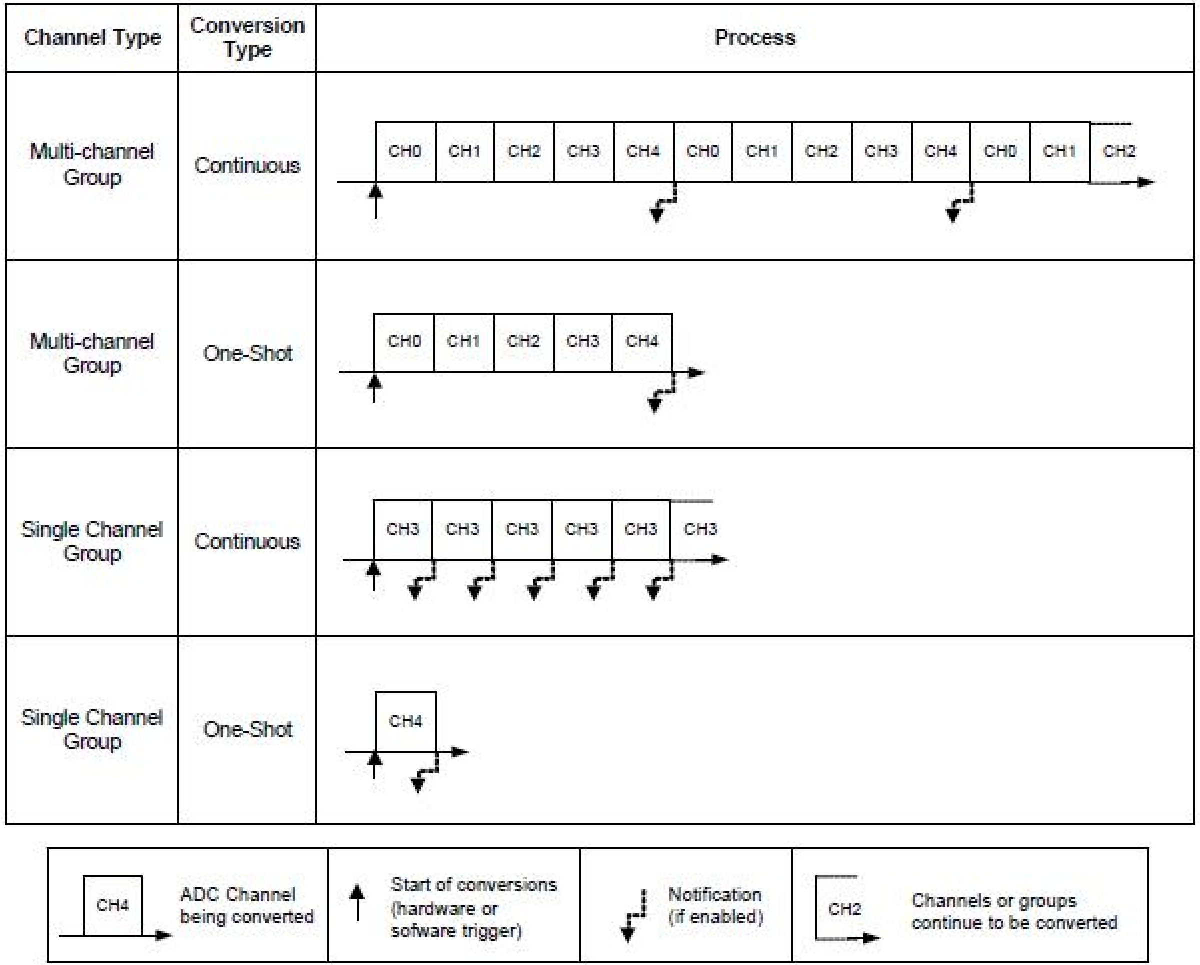
The following examples specify the order of channel conversion depending on group and conversion type:

Example 1: Channel group containing channels [CH0, CH1, CH2, CH3, and CH4] is configured in Continuous conversion mode. After finishing each scan, the notification (if enabled) is called. Then a new scan is started automatically.

Example 2: Channel group containing channels [CH0, CH1, CH2, CH3, and CH4] is configured in One-Shot conversion mode. After finishing the scan the notification (if enabled) is called.

Example 3: Channel group containing channel [CH3] is configured in Continuous conversion mode. After finishing each scan the notification (if enabled) is called. Then a new scan is started automatically.

Example 4: Channel group containing channel [CH4] is configured in One-Shot conversion mode. After finishing the scan the notification (if enabled) is called.



Channel Type	Conversion Type	Process
Multi-channel Group	Continuous	CH0 CH1 CH2 CH3 CH4 CH0 CH1 CH2 CH3 CH4 CH0 CH1 CH2
Multi-channel Group	One-Shot	CH0 CH1 CH2 CH3 CH4
Single Channel Group	Continuous	CH3 CH3 CH3 CH3 CH3 CH3
Single Channel Group	One-Shot	CH4

CH4 ADC Channel being converted	Start of conversions (hardware or sofware trigger)	Notification (if enabled)	CH2 Channels or groups continue to be converted

Conversion Mode examples

4.9 Directory Structure

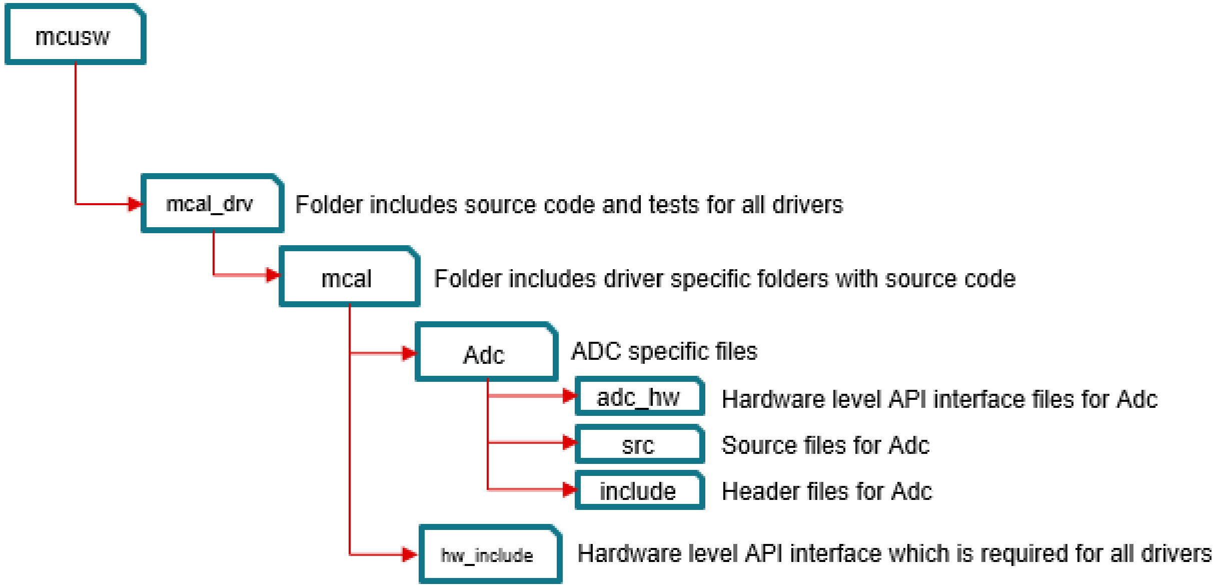
The directory structure is as depicted in figures below, the source files can be categorized under “Driver Implementation” and “Example Application”

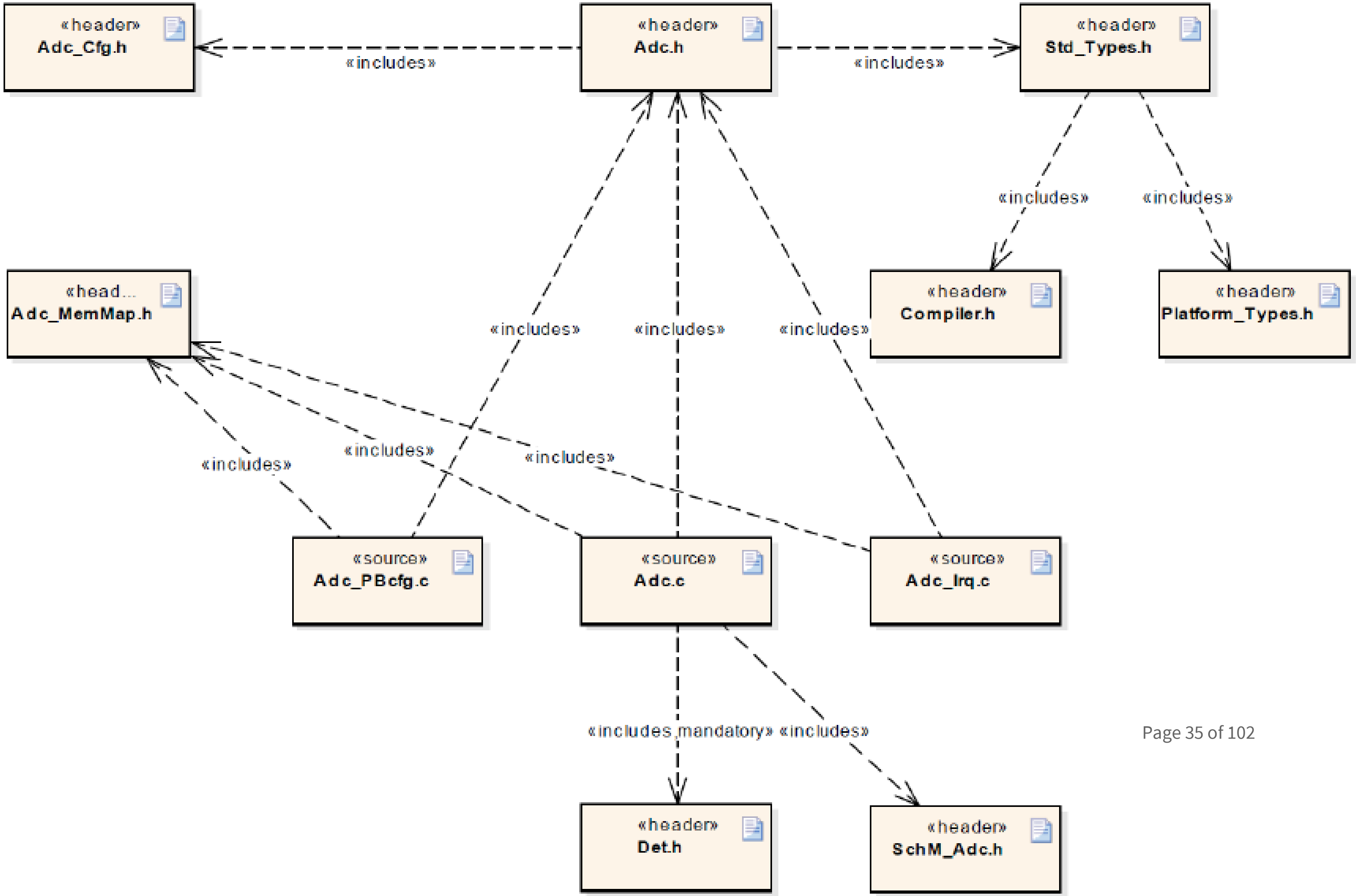
Driver Implemented by

Adc.h, Adc_Dbg.h and Adc_Irq.h: Shall implement the interface provided by the driver

Adc.c, Adc_Priv.c, Adc_Irq.c, Adc_Utils.c and Adc_Priv.h: Shall implement the driver functionality

hw_adc.h, lldr_adcss.h and lld_hw_adc.h: Shall include the SOC specific register definitions.


mcusw
mcal_drv
Folder includes source code and tests for all drivers
mcal
Folder includes driver specific folders with source code
Adc
ADC specific files
adc_hw
Hardware level API interface files for Adc
src
Source files for Adc
include
Header files for Adc
hw_include
Hardware level API interface which is required for all drivers

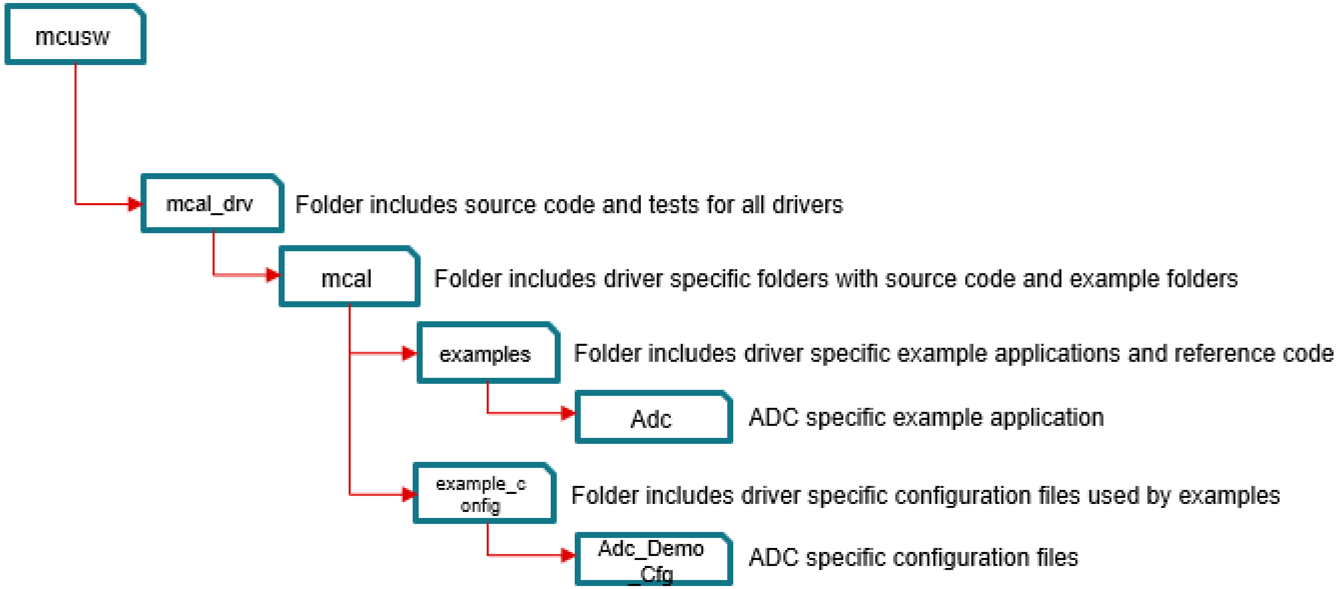

«header»
Adc_Cfg.h
«header»
Adc.h
«header»
Std_Types.h
«includes»
«includes»
«includes»
«includes»
«head...
Adc_MemMap.h
«header»
Compiler.h
«header»
Platform_Types.h
«includes»
«includes»
«includes»
«includes»
«includes»
«includes»
«source»
Adc_PBcfg.c
«source»
Adc.c
«source»
Adc_Irq.c
«includes mandatory»
«includes»
«header»
Det.h
«header»
SchM_Adc.h

Example Application

Adc_Cfg.h and Adc_Cfg.c: Shall implement the generated configuration for pre-compile variant

Adc_PBcfg.c: Shall implement the generated configuration for post-build variant

AdcApp.c and AdcApp.h: Shall implement the example application that demonstrates the use of the driver


mcusw
mal_drv
Folder includes source code and tests for all drivers
mcal
Folder includes driver specific folders with source code and example folders
examples
Folder includes driver specific example applications and reference code
Adc
ADC specific example application
example_config
Folder includes driver specific configuration files used by examples
Adc_Demo_Cfg
ADC specific configuration files

4.10 Configurator

The AUTOSAR ADC Driver Specification details mandatory parameters that shall be configurable via the configurator.

Design ID	Description
MCAL-5890	SWS_Adc_00363 : Variant POST BUILD Support
MCAL-5698	ECUC_Adc_00027 :
MCAL-5697	ECUC_Adc_00410 : AdcChannelValueSigned
MCAL-5696	ECUC_Adc_00028 : AdcGroup
MCAL-5694	ECUC_Adc_00405 : AdcDevErrorDetect
MCAL-5692	ECUC_Adc_00287 : AdcGroupPriority
MCAL-5691	ECUC_Adc_00409 : AdcVersionInfoApi
MCAL-5689	ECUC_Adc_00406 : AdcEnableStartStopGroupApi
MCAL-5688	ECUC_Adc_00398 : AdcGroupId

Design ID	Description
MCAL-5685	ECUC_Adc_00030 : AdcPublishedInformation
MCAL-5684	ECUC_Adc_00412 : AdcMaxChannelResolution
MCAL-5683	ECUC_Adc_00105 : AdcGrpNotifCapability
MCAL-5680	ECUC_Adc_00011 : AdcChannelConvTime
MCAL-5679	ECUC_Adc_00389 : AdcHwUnitId
MCAL-5678	ECUC_Adc_00391 : AdcEnableQueuing
MCAL-5677	ECUC_Adc_00392 : AdcChannelId
MCAL-5676	ECUC_Adc_00023 : AdcChannelRefVoltsrcLow
MCAL-5674	ECUC_Adc_00268 : AdcChannel
MCAL-5673	ECUC_Adc_00402 : AdcNotification
MCAL-5672	ECUC_Adc_00390 : AdcConfigSet

Design ID	Description
MCAL-5665	ECUC_Adc_00317 : AdcGroupAccessMode
MCAL-5664	ECUC_Adc_00089 : AdcChannelRefVoltsrcHigh
MCAL-5661	ECUC_Adc_00292 : AdcStreamingNumSamples
MCAL-5660	ECUC_Adc_00316 : AdcStreamingBufferMode
MCAL-5659	ECUC_Adc_00404 : AdcDeInitApi
MCAL-5658	ECUC_Adc_00399 : AdcGroupTriggSrc
MCAL-5657	ECUC_Adc_00435 : AdcGroupReplacement
MCAL-5655	ECUC_Adc_00014 : AdcGroupDefinition
MCAL-5654	ECUC_Adc_00242 : AdcHwUnit
MCAL-5653	ECUC_Adc_00444 : AdcResultAlignment
MCAL-5651	ECUC_Adc_00290 : AdcChannelSampTime

Design ID	Description
MCAL-5649	ECUC_Adc_00394 : AdcReadGroupApi
MCAL-5647	ECUC_Adc_00087 : AdcClockSource
MCAL-5643	ECUC_Adc_00393 : AdcPriorityImplementation
MCAL-5642	ECUC_Adc_00397 : AdcGroupConversionMode
MCAL-5637	ECUC_Adc_00411 : AdcGroupFirstChannelFixed
MCAL-5636	ECUC_Adc_00019 : AdcChannelResolution

4.10.1 NON Standard configurable parameters

Following lists this design's specific configurable parameters

Parameter	Usage comment	Category

AdcChannelOpenDelay	Number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion	Mandatory
AdcChannelSampleDelay	Number of ADC clock cycles to hold SOC high.The actual delay is +1 clock cycle of this value	Mandatory
AdcChannelRangeCheckEnable	Option to enable range check per channel.Currently this is not supported by the driver	Optional
AdcChannelAveragingMode	Option for averaging the sampled data.ADC allows user to program the number of samplings to average	Mandatory
AdcGroupLog	Enables/Disables ADC Group logging.Useful in debugging	Optional
AdcGroupLogMaxLen	Maximum length of Group log buffer that can be used	Optional
AdcFifoErrLog	Enables/Disables ADC Fifo error logging.Useful in debugging	Optional

AdcFifoErrLogMaxLen	Max length of Fifo error log buffer that can be used	Optional
AdcMaxGroupCount	Maximum group count across all hwunits configured	Mandatory
AdcMaxHwUnitCount	Maximum HW unit count - This should match the sum of the units of ISR	Mandatory
AdcHwUnitActive	Enables/Disables HW UNIT	Mandatory
AdcEnableRegisterReadbackApi	Enable API to readback ADC critical registers	Mandatory
AdcTypeofInterruptFunction	Type of ISR function CAT1 : interrupt void func(void) CAT2 : ISR(func)	Mandatory
AdcAfePowerUpWaitTicks	Software must wait minimum 4us after a AFE(Analog Front End).(Each tick 31.25us(32 K Counter), Required 4us(1/8th of 31.25us) = ~1U(apprx)	Mandatory

AdcOsCounterRef	This parameter contains a reference to the OsCounter, which is used by the ADC driver.	Mandatory
AdcDefaultOSCounterId	Default Os Counter Id if node reference to OsCounter ref AdcOsCounterRef is not set	Mandatory
AdcTimeoutDuration	ADC timeout - used in ADC AFE busy wait and FSM busy wait.Each tick is 31.25us (for 32K Counter). Wait for 100ms is 0xC80	Mandatory
AdcMaxRange	Maximum value of range for ADC sampled data. This is of type published information and not editable	Optional
AdcMinRange	Minimum value of range for ADC sampled data. This is of type published information and not editable	Optional
AdcMaxNumChannels	Number of MCAL channels - in terms of ADC HW, this represents the number of hardware steps.This is a fixed value as per the ADC module and can't be changed. This is of type published information and not editable	Optional

AdcMaxHwChannelId	Max HW Channel Id - This parameter defines the max value for assignment of the channel to the physical ADC hardware channel. This is of type published information and not editable	Optiona l
AdcMinHwChannelId	Min HW Channel Id - This parameter defines the min value for assignment of the channel to the physical ADC hardware channel. This is of type published information and not editable	Optiona l
AdcMaxOpenDelay	Maximum value of open delay. This is of type published information and not editable	Optiona l
AdcMinOpenDelay	Minimum value of open delay. This is of type published information and not editable	Optiona l
AdcMaxSampleDelay	Maximum value of sample delay. This is of type published information and not editable	Optiona l
AdcMinSampleDelay	Minimum value of sample delay. This is of type published information and not editable	Optiona l

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

Design Identifier	Description
MCAL-5681	AdcMinRange
MCAL-5667	AdcMaxRange
MCAL-5635	AdcTypeofInterruptFunction
MCAL-5650	AdcMaxHwChannelId
MCAL-5640	AdcMinHwChannelId
MCAL-5695	AdcMaxNumChannels
MCAL-5693	AdcGroupLogMaxLen

Design Identifier	Description
MCAL-5690	AdcMaxGroupCount
MCAL-5687	AdcDemEventParameterRefs
MCAL-5686	AdcDeviceVariant
MCAL-5675	AdcTimeoutDuration
MCAL-5671	AdcEnableRegisterReadbackApi
MCAL-5670	AdcChannelRangeCheckEnable
MCAL-5669	AdcMinSampleDelay
MCAL-5668	AdcMinOpenDelay
MCAL-5666	AdcMaxSampleDelay
MCAL-5663	AdcChannelOpenDelay
MCAL-5656	AdcMaxOpenDelay

Design Identifier	Description
MCAL-5652	AdcDefaultOSCounterId
MCAL-5650	AdcMaxHwChannelId
MCAL-5648	AdcFifoErrLogMaxLen
MCAL-5646	AdcGroupLog
MCAL-5645	AdcChannelSampleDelay
MCAL-5644	AdcAfePowerUpWaitTicks
MCAL-5641	AdcOsCounterRef
MCAL-5640	AdcMinHwChannelId
MCAL-5639	AdcChannelAveragingMode
MCAL-5638	AdcFifoErrLog
MCAL-5635	AdcTypeofInterruptFunction

4.11 Debug Information

The current ongoing group conversion of ADC is logged and shall be visible to applications. Also in case of FIFO error, groupId and corresponding hw unit is also logged. This is achieved by Adc_Dbg.h, as depicted in (Directory Structure).

4.12 Error Classification

Errors are classified in two categories, development error and runtime / production error.

Design Identifier	Description
MCAL-5877	SWS_Adc_00131 : Adc_DisableGroupNotification DET ADC_E_PARAM_GROUP
MCAL-5875	Adc_Init DET Error : ADC_E_PARAM_CONFIG
MCAL-5872	SWS_Adc_00351 : Adc_StartGroupConversion DET ADC_E_BUSY
MCAL-5863	SWS_Adc_00165 : Adc_EnableGroupNotification DET ADC_E_NOTIF_CAPABILITY
MCAL-5862	SWS_Adc_00302 : Adc_GetStreamLastPointer DET ADC_E_UNINIT
MCAL-5859	SWS_Adc_00107 : Adc_Init DET ADC_E_ALREADY_INITIALIZED
MCAL-5851	SWS_Adc_00296 : Adc_ReadGroup DET ADC_E_UNINIT

Design Identifier	Description
MCAL-5850	SWS_Adc_00126 : Adc_StopGroupConversion DET ADC_E_PARAM_GROUP
MCAL-5846	SWS_Adc_00424 : Adc_StartGroupConversion DET ADC_E_BUFFER_UNINIT
MCAL-5842	SWS_Adc_00241 : Adc_StopGroupConversion DET ADC_E_IDLE
MCAL-5841	SWS_Adc_00112 : Adc_DeInit DET ADC_E_BUSY
MCAL-5830	SWS_Adc_00458 : Adc_GetVersionInfo DET ADC_E_PARAM_POINTER
MCAL-5827	SWS_Adc_00434 : Adc_SetupResultBuffer DET ADC_E_UNINIT
MCAL-5825	SWS_Adc_00125 : Adc_StartGroupConversion DET ADC_E_PARAM_GROUP
MCAL-5824	SWS_Adc_00152 : Adc_ReadGroup DET ADC_E_PARAM_GROUP
MCAL-5819	SWS_Adc_00346 : Adc_StartGroupConversion DET ADC_E_BUSY
MCAL-5818	SWS_Adc_00299 : Adc_EnableGroupNotification DET ADC_E_UNINIT
MCAL-5813	SWS_Adc_00166 : Adc_DisableGroupNotification DET ADC_E_NOTIF_CAPABILITY

Design Identifier	Description
MCAL-5810	SWS_Adc_00301 : Adc_GetGroupStatus DET ADC_E_UNINIT
MCAL-5809	SWS_Adc_00133 : Adc_StartGroupConversion DET ADC_E_WRONG_TRIGG_SRC
MCAL-5796	SWS_Adc_00388 : Adc_ReadGroup DET ADC_E_IDLE
MCAL-5786	SWS_Adc_00423 : Adc_SetupResultBuffer DET ADC_E_PARAM_GROUP
MCAL-5785	SWS_Adc_00295 : Adc_StopGroupConversion DET ADC_E_UNINIT
MCAL-5784	SWS_Adc_00154 : Adc_DeInit DET ADC_E_UNINIT
MCAL-5783	SWS_Adc_00215 : Adc_GetStreamLastPointer DET ADC_E_IDLE
MCAL-5777	SWS_Adc_00164 : Adc_StopGroupConversion DET ADC_E_WRONG_TRIGG_SRC
MCAL-5776	SWS_Adc_00130 : Adc_EnableGroupNotification DET ADC_E_PARAM_GROUP
MCAL-5773	SWS_Adc_00457 : Adc_SetupResultBuffer DET ADC_E_PARAM_POINTER
MCAL-5759	SWS_Adc_00426 : Adc_StartGroupConversion DET ADC_E_BUSY

Design Identifier	Description
MCAL-5758	SWS_Adc_00433 : Adc_SetupResultBuffer DET ADC_E_BUSY
MCAL-5755	SWS_Adc_00344 : Adc_Init DET ADC_E_PARAM_CONFIG Variant PC
MCAL-5751	SWS_Adc_00348 : Adc_StartGroupConversion DET ADC_E_BUSY
MCAL-5743	SWS_Adc_00225 : Adc_GetGroupStatus DET ADC_E_PARAM_GROUP
MCAL-5737	SWS_Adc_00300 : Adc_DisableGroupNotification DET ADC_E_UNINIT
MCAL-5725	SWS_Adc_00294 : Adc_StartGroupConversion DET ADC_E_UNINIT
MCAL-5720	SWS_Adc_00428 : Adc_StartGroupConversion DET ADC_E_BUSY
MCAL-5717	SWS_Adc_00343 : Adc_Init DET ADC_E_PARAM_CONFIG Variant PB
MCAL-5713	SWS_Adc_00427 : Adc_StartGroupConversion DET ADC_E_BUSY
MCAL-5707	SWS_Adc_00218 : Adc_GetStreamLastPointer DET ADC_E_PARAM_GROUP
MCAL-5687	AdcDemEventParameterRefs

4.12.1 Development Errors

Type of Error	Related Error code	Value (Hex)
Adc_Init has not been called prior to another function call	ADC_E_UNINIT	0x0A
Adc_StartGroupConversion/Adc_EnableHardwareTrigger/Adc_DeInit was called while a conversion is still ongoing	ADC_E_BUSY	0x0B
Adc_StopGroupConversion was called while no conversion was running	ADC_E_IDLE	0x0C
Adc_Init has been called while ADC is already initialized	ADC_E_ALREADY_INITIALIZED	0x0D
Adc_Init has been called with incorrect configuration parameter	ADC_E_PARAM_CONFIG	0x0E
Adc_SetupResultBuffer or Adc_GetVersionInfo called with invalid data buffer pointer, NULL_PTR passed	ADC_E_PARAM_POINTER	0x14
Invalid group ID requested	ADC_E_PARAM_GROUP	0x15

Adc_EnableHardwareTrigger or Adc_DisableHardwareTrigger called on a group with conversion mode configured as continuous	ADC_E_WRONG_CONV_MODE	0x16
Adc_StartGroupConversion or Adc_StopGroupConversion called on a group with trigger source configured as hardware or Adc_EnableHardwareTrigger or Adc_DisableHardwareTrigger called on a group with trigger source configured as software API	ADC_E_WRONG_TRIGG_SRC	0x17
Enable/disable notification function for a group whose configuration set has no notification available	ADC_E_NOTIF_CAPABILITY	0x18
Conversion started and result buffer pointer is not initialized	ADC_E_BUFFER_UNINIT	0x19
One or more ADC group/channel not in IDLE state	ADC_E_NOT_DISENGAGED	0x1A
Unsupported power state request	ADC_E_POWER_STATE_NOT_SUPPORTED	0x1B
Requested power state can not be reached directly	ADC_E_TRANSITION_NOT_POSSIBLE	0x1C
ADC not prepared for target power state	ADC_E_PERIPHERAL_NOT_PREPARED	0x1D

4.12.2 **Error Detection**

The detection of development errors is configurable (ON / OFF) at pre-compile time. The switch AdcDevErrorDetect will activate or deactivate the detection of all development errors.

4.12.3 **Error notification (DET)**

All detected development errors are reported to Det_ReportError service of the Development Error Tracer (DET). All runtime errors are reported to Det_ReportRuntimeError service of the Development Error Tracer (DET). If a callout has been configured then this callout shall be called.

4.12.4 **Runtime Errors**

The following runtime/production errors shall be detectable by Adc driver.

Type of Error	**Related Error code**	**Value (Hex)**
Reference to the DemEventParameter which shall be issued when the error Timeout on blocking API call occurs	ADC_E_HARDWARE_ERROR	Defined By Integrator

4.12.5 **Error notification (DEM)**

All detected run time errors shall be reported to Dem_ReportErrorStatus () service of the Diagnostic Event Manager (DEM).

5 Implementation Details

5.1 Data structures and resources

The sections below lists some of key data structures that shall be implemented and used in driver implementation

Maximum number of groups

Type	Identifier	Comments
uint32	ADC_MAX_GROUP	Maximum group across all hwunit.

Maximum number of hw unit

Type	Identifier	Comments
uint32	ADC_MAX_HW_UNIT	Sum of all active hw units configured.

Maximum number of hw channels

Type	Identifier	Comments
uint32	ADC_NUM_CHANNEL	Number of MCAL channels - in terms of ADC HW, this represents the number of hardware steps.

Adc_ConfigType

Data structure containing the set of configuration parameters required for initializing the ADC Driver and ADC HW Unit(s), please refer section 8.2.1 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_ChannelType

Numeric ID of an ADC channel, refer section 8.2.2 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_GroupType

Numeric ID of an ADC channel group, refer section 8.2.3 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_ValueGroupType

Type for reading the converted values of a channel group (raw, without further scaling, alignment according precompile switch ADC_RESULT_ALIGNMENT). Refer section 8.2.4 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_PrescaleType

Type of clock prescaler factor. Refer section 8.2.5 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_ConversionTimeType

Type of conversion time, i.e. the time during which the sampled analogue value is converted into digital representation. Refer section 8.2.6 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_SamplingTimeType

Type of sampling time, i.e. the time during which the value is sampled, (in clock-cycles). Refer section 8.2.7 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_ResolutionType

Type of channel resolution in number of bits. Refer section 8.2.8 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_StatusType

Current status of the conversion of the requested ADC Channel group. Refer section 8.2.9 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_TriggerSourceType

Type for configuring the trigger source for an ADC Channel group. Refer section 8.2.10 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_GroupConvModeType

Type for configuring the conversion mode of an ADC Channel group. Refer section 8.2.11 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_GroupPriorityType

Priority level of the channel. Lowest priority is 0. Refer section 8.2.12 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_GroupDefType

Type for assignment of channels to a channel group (this is not an API type). Refer section 8.2.13 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_StreamNumSampleType

Type for configuring the number of group conversions in streaming access mode (in single access mode, parameter is 1). Refer section 8.2.14 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_StreamBufferModeType

Type for configuring the streaming access mode buffer type. Refer section 8.2.15 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_GroupAccessModeType

Type for configuring the access mode to group conversion results. Refer section 8.2.16 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_HwTriggerSignalType

Type for configuring on which edge of the hardware trigger signal the driver should react, i.e. start the conversion (only if supported by the ADC hardware). Refer section 8.2.17 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_HwTriggerTimerType

Type for the reload value of the ADC module embedded timer (only if supported by the ADC hardware). Refer section 8.2.18 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_PriorityImplementationType

Type for configuring the prioritization mechanism. Refer section 8.2.19 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_GroupReplacementType

Replacement mechanism, which is used on ADC group level, if a group conversion is interrupted by a group which has a higher priority. Refer section 8.2.20 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_ChannelRangeSelectType

In case of active limit checking: defines which conversion values are taken into account related to the boardes defineed with AdcChannelLowLimit and AdcChannelHighLimit. Refer section 8.2.21 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_ResultAlignmentType

Type for alignment of ADC raw results in ADC result buffer (left/right alignment). Refer section 8.2.22 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_PowerStateType

Power state currently active or set as target power state. Refer section 8.2.23 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_PowerStateRequestResultType

Result of the requests related to power state transitions. Refer section 8.2.24 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Adc_RegisterReadbackType

Name	Type	Range	Comments
adcRev	uint32	0 to 0xFFFFFFFF	H/W version identifier, will not change for a given SoC
adcCtrl	uint32	0 to 0xFFFFFFFF	Controls various parameters of the spi controller state
adcMisc	uint32	0 to 0xFFFFFFFF	Internal Calibration

Adc_ChannelConfigType

Structure containing parameters for ADC MCAL channel configuration. In terms of ADC hardware, this represents the step configuration. There are ADC_NUM_CHANNEL steps in the ADC hardware and each step could be mapped to an actual hardware input channel.

Type	Variable Name	Comments
uint32	hwChannelId	The hardware channel number from which input is given.Valid values: ADC_MIN_HW_CHANNEL_ID to ADC_MAX_HW_CHANNEL_ID
uint32	openDelay	Number of ADC clock cycles to wait after applying the step configuration registers and before sending the start of ADC conversion.Valid values: ADC_MIN_OPEN_DELAY to ADC_MAX_OPEN_DELAY
uint32	sampleDelay	Number of ADC clock cycles to hold SOC high. The actual delay is +1 of this value.Valid values: ADC_MIN_SAMPLE_DELAY to ADC_MAX_SAMPLE_DELAY.
uint32	rangeCheckEnable	Option to enable range check per channel.Enabled if it is TRUE.Disabled if it is FALSE.Note: Since there are no standard MCAL API for this feature, this is not supported by the ADC driver. So set this to FALSE.

Adc_AveragingMode	averagingMode	Option for averaging the sampled data.

Adc_GroupConfigType

ADC Group configuration structure.

Type	Variable Name	Comments
Adc_GroupType	groupId	Group ID - This should be same as that of the index in to the groupCfg[] array of Adc_ConfigType structure.
Adc_GroupPriorityType	groupPriority	Group priority.
Adc_HWUnitType	hwUnitId	HWUnit associated with this group.
Adc_GroupEndNotifyType	Adc_GroupEndNotification	Group end notification callback fxn pointer.
Adc_StreamNumSampleType	streamNumSamples	Contains how many samples fit into result buffer.
Adc_ResolutionType	resolution	Group resolution - This is not configurable and should be set to ADC_DEF_CHANNEL_RESOLUTION

Adc_GroupConvModeType	convMode	Operation mode of the group
Adc_TriggerSourceType	triggSrc	Determines the trigger source (hw or sw trigger). Note: Only SW trigger is supported
Adc_GroupAccessModeType	accessMode	Defines the type of the groups buffer
Adc_StreamBufferModeType	streamBufMode	Use linear or circular stream buffer
Adc_HwTriggerSignalType	hwTrigSignal	Use rising or falling edge for event pin trigger. Note: Since HW trigger is not supported, this parameter is not used
Adc_HwTriggerTimerType	hwTrigTimer	Hardware trigger event. Note: Since HW trigger is not supported, this parameter is not used. Set it to 0.
Adc_GroupReplacementType	groupReplacement	Group replacement logic when priority mechanism is ON - i.e prioritySupport is not ADC_PRIORITY_NONE.

uint32	highRange	Sampled ADC data is compared to this value. If the sampled data is greater than this value, then interrupt is generated.Since there are no standard MCAL API for this feature, this is not supported by the ADC driver.
uint32	lowRange	Sampled ADC data is compared to this value. If the sampled data is lesser than this value, then interrupt is generated.Since there are no standard MCAL API for this feature, this is not supported by the ADC driver.
uint32	numChannels	Number of channels in this group
Adc_ChannelConfigType	channelConfig[ADC_NUM_CHANNEL]	Channel (HW step) configuration. numChannels elements should be initialized.

Adc_HwUnitConfigType

ADC Hardware unit configuration structure.

Type	Variable Name	Comments
Adc_HWUnitType	hwUnitId	ADC HW unit to use

Adc_ConfigType

Used to define all channels specific parameters, shall be supplied to Adc_Init () function. Values of these are expected to be populated by configurator.

Type	Variable Name	Comments
uint8	maxGroup	Maximum number of group should not be more than ADC_MAX_GROUP.
uint8	maxHwUnit	Maximum number of HW unit should not be more than ADC_MAX_HW_UNIT.
Adc_GroupConfigType	groupCfg[ADC_MAX_GROUP]	Group configurations.
Adc_HwUnitConfigType	hwUnitCfg[ADC_MAX_HW_UNIT]	HW Unit configurations.

Global Variables

This design expects that implementation will require to use following global variables.

Variable	Type	Description	Default Value
Adc_DrvIsInit	uint32	ADC driver init status	FALSE

Adc_DrvObj	Adc_DriverObjType	ADC driver object, local to the implementation and scope shall NOT be limited to Adc.c	Un defined
Adc_GroupLogObj	Adc_GroupLogType	ADC group log object, local to the driver implementation and scope shall be limited to Adc.c	Un defined
Adc_FifoErrLogObj	Adc_FifoErrLogType	ADC FIFO error log object, local to the driver implementation and scope shall be limited to Adc.c	Un defined

Design Identifier	Description
MCAL-5888	SWS_Adc_00318 : Adc_ValueGroupType Single value access mode
MCAL-5881	SWS_Adc_00508 : Adc_ValueGroupType
MCAL-5871	SWS_Adc_00520 : Adc_HwTriggerSignalType
MCAL-5858	SWS_Adc_00521 : Adc_HwTriggerTimerType
MCAL-5856	SWS_Adc_00320 : Adc_ValueGroupType Dimension of each buffer element
MCAL-5855	SWS_Adc_00511 : Adc_SamplingTimeType

Design Identifier	Description
MCAL-5852	SWS_Adc_00506 : Adc_ChannelType
MCAL-5843	SWS_Adc_00525 : Adc_ResultAlignmentType
MCAL-5836	SWS_Adc_00514 : Adc_TriggerSourceType
MCAL-5829	SWS_Adc_00526 : Adc_PowerStateType
MCAL-5826	SWS_Adc_00518 : Adc_StreamNumSampleType
MCAL-5822	SWS_Adc_00510 : Adc_ConversionTimeType
MCAL-5817	SWS_Adc_00517 : Adc_GroupDefType
MCAL-5811	SWS_Adc_00513 : Adc_StatusTypeType
MCAL-5807	SWS_Adc_00522 : Adc_PriorityImplementationType
MCAL-5800	SWS_Adc_00512 : Adc_ResolutionType
MCAL-5769	SWS_Adc_00528 : Adc_GroupAccessModeType

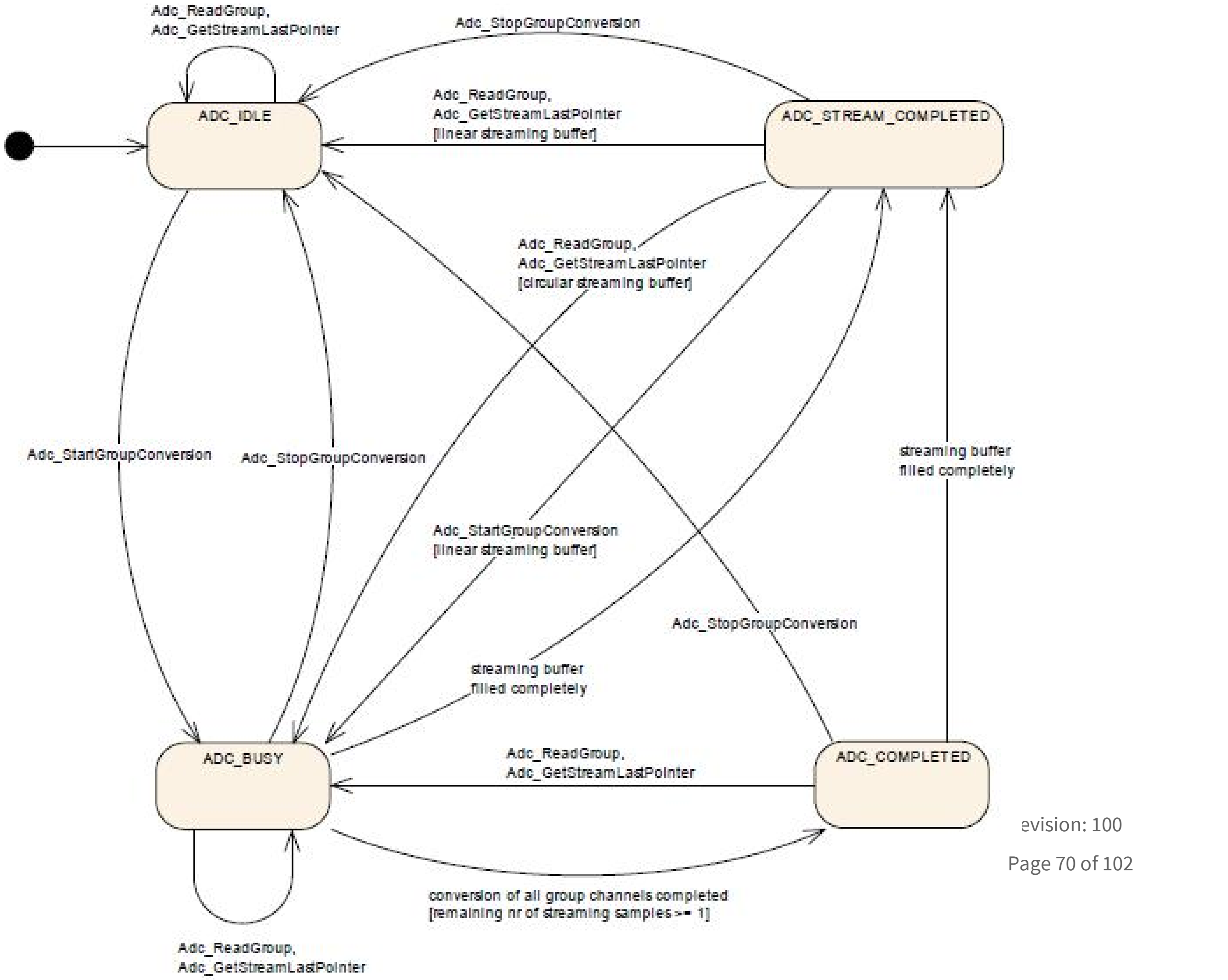
Design Identifier	Description
MCAL-5761	SWS_Adc_00319 : Adc_ValueGroupType Streaming access mode
MCAL-5742	SWS_Adc_00519 : Adc_StreamBufferModeType
MCAL-5735	SWS_Adc_00515 : Adc_GroupConvModeType
MCAL-5732	SWS_Adc_00523 : Adc_GroupReplacementType
MCAL-5723	SWS_Adc_00364 : ADC Module Imported Types
MCAL-5716	SWS_Adc_00505 : Adc_ConfigType
MCAL-5712	SWS_Adc_00516 : Adc_GroupPriorityType
MCAL-5709	SWS_Adc_00509 : Adc_PrescaleType
MCAL-5705	SWS_Adc_00527 : Adc_PowerStateRequestResultType
MCAL-5703	SWS_Adc_00524 : Adc_ChannelRangeSelectType
MCAL-5702	SWS_Adc_00507 : Adc_GroupType

Design Identifier	Description
MCAL-5635	AdcTypeofInterruptFunction

5.2 Dynamic Behavior - Control Flow Diagram

States

Driver group status will be in one of the following states. ADC_IDLE, ADC_BUSY, ADC_COMPLETED, ADC_STREAM_COMPLETED. A variable shall be maintained per ADC channel group to track and maintain the state. The diagram below shows transitions of states and it’s associated service APIs.


Adc_ReadGroup,
Adc_GetStreamLastPointer
Adc_StopGroupConversion
ADC_IDLE
Adc_ReadGroup,
Adc_GetStreamLastPointer
[linear streaming buffer]
ADC_STREAM_COMPLETED
Adc_ReadGroup,
Adc_GetStreamLastPointer
[circular streaming buffer]
Adc_StartGroupConversion
Adc_StopGroupConversion
streaming buffer
filled completely
Adc_StartGroupConversion
[linear streaming buffer]
Adc_StopGroupConversion
streaming buffer
filled completely
ADC_BUSY
Adc_ReadGroup,
Adc_GetStreamLastPointer
ADC_COMPLETED
conversion of all group channels completed
[remaining nr of streaming samples >= 1]
Adc_ReadGroup,
Adc_GetStreamLastPointer

Driver States SW Trigger, Streaming Access : Sourced from AUTOSAR Spec

variable shall be maintained per ADC channel group to track and maintain the state. The diagram below shows transitions of states and it’s associated service APIs.

5.3 Dynamic Behavior - Data Flow Diagram

Not Applicable

5.4 Application Parameters

Adc_Init

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
CfgPtr	Pointer to configuration set in Variant PB	0-0xFFFFFFFF	-	-	N.A

Adc_SetupResultBuffer

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Group	Numeric ID of requested ADC channel group	0-8	-	-	N.A

DataBufferPtr	pointer to result data buffer	0-0xFFFFFFFF	-	-	N.A

Adc_GetGroupStatus

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Group	Numeric ID of requested ADC channel group	0-8	-	-	N.A

Adc_GetStreamLastPointer

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Group	Numeric ID of requested ADC channel group	0-8	-	-	N.A
PtrToSamplePtr	Out Parameter: Pointer to result buffer pointer	0-0xFFFFFFFF	-	-	N.A

Adc_StartGroupConversion

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant

Group	Numeric ID of requested ADC channel group	0-8	-	-	N.A

Adc_StopGroupConversion

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Group	Numeric ID of requested ADC channel group	0-8	-	-	N.A

Adc_ReadGroup

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Group	Numeric ID of requested ADC channel group	0-8	-	0	N.A
DataBufferPtr	ADC results of all channels of the selected group are stored in the data buffer addressed with the pointer	0-0xFFFFFFFF	-	NULL	N.A

Adc_EnableGroupNotification

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Group	Numeric ID of requested ADC channel group	0-8	-	-	N.A

Adc_DisableGroupNotification

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
Group	Numeric ID of requested ADC channel group	0-8	-	-	N.A

Adc_GetVersionInfo

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
versioninfo	Out Parameter: Pointer to where to store the version information of this module	-	-	-	N.A

Adc_RegisterReadback

Parameter	Description	Possible Value ranges	Unit of Value	Default Value	Variant
HWUnit	ADC Hardware microcontroller peripheral unit ID	-	-	-	N.A
RegRbPtr	Inout parameters: Pointer to where to store the readback values.	0-0xFFFFFFFF	-	-	N.A

5.5 Safety Diagnostic Features

ADC1A - Converter Self-Test

ADC supports a "self-test" mode where the input is tested against known reference voltages with an expected output. A full scale value is expected when the internal reference is REFP (Vref), and half-scale when connected to VMID (Vref).

The ADC MCAL driver provides the API - **Adc_EnableInternalDiagnostic()** and **Adc_DisableInternalDiagnostic()** to to configure the ADC's converter self-test mode for detecting shorts and open circuits at the inputs.

ADC1B - Converter Calibration Test

ADC supports a "self-test" mode where the input is tested against known reference voltages with an expected output. A full scale value is expected when the internal reference is REFP (Vref), and half-scale when connected to VMID (Vref).

The ADC MCAL driver provides the API - **Adc_EnableInternalDiagnostic()** and **Adc_DisableInternalDiagnostic()** to to configure the ADC's converter self-test mode for detecting drift in ADC measurments by performing conversions on known references.

ADC4 - Software Readback of Written Configuration / ADC5 - Periodic Software Readback of Static Configuration Registers

Software Readback of Written Configuration ensures that the configuration register are written with the expected value. Periodic readback of configuration registers can provide a diagnostic for inadvertent writes to these registers.

The ADC MCAL driver provides the API - **Adc_RegisterReadback** to readback static and written configuration registers to implement this diagnostic feature.

6 Low Level Definitions

6.1 Driver API's

For the standard API's please refer 8.3 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1. Sections below highlight other design considerations for the implementation.

6.1.1 Adc_Init

Refer section 8.3.1 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-5884	SWS_Adc_00250 : Adc_Init Startup code dependency
MCAL-5875	Adc_Init DET Error : ADC_E_PARAM_CONFIG
MCAL-5859	SWS_Adc_00107 : Adc_Init DET ADC_E_ALREADY_INITIALIZED
MCAL-5854	SWS_Adc_00077 : Adc_Init Disable notifications
MCAL-5834	SWS_Adc_00248 : Adc_Init MCU driver dependency

Design Identifier	Description
MCAL-5812	SWS_Adc_00249 : Adc_Init Initialize one time writable register
MCAL-5808	SWS_Adc_00307 : Adc_Init Set Group status
MCAL-5797	SWS_Adc_00246 : Adc_Init Initialize only one usage of register
MCAL-5781	SWS_Adc_00342 : Adc_Init Variant PC
MCAL-5767	SWS_Adc_00054 : Adc_Init Variant PB
MCAL-5755	SWS_Adc_00344 : Adc_Init DET ADC_E_PARAM_CONFIG Variant PC
MCAL-5736	SWS_Adc_00056 : Adc_Init Resource configuration
MCAL-5717	SWS_Adc_00343 : Adc_Init DET ADC_E_PARAM_CONFIG Variant PB
MCAL-5700	SWS_Adc_00247 : Adc_Init PORT driver dependency

6.1.2 **Adc_SetupResultBuffer**

Refer section 8.3.2 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-5848	SWS_Adc_00420 : Adc_SetupResultBuffer Initialize result buffer pointer
MCAL-5827	SWS_Adc_00434 : Adc_SetupResultBuffer DET ADC_E_UNINIT
MCAL-5786	SWS_Adc_00423 : Adc_SetupResultBuffer DET ADC_E_PARAM_GROUP
MCAL-5773	SWS_Adc_00457 : Adc_SetupResultBuffer DET ADC_E_PARAM_POINTER
MCAL-5758	SWS_Adc_00433 : Adc_SetupResultBuffer DET ADC_E_BUSY
MCAL-5754	SWS_Adc_00421 : Adc_SetupResultBuffer Initialize result buffer without prior initialization
MCAL-5721	SWS_Adc_00422 : Adc_SetupResultBuffer result buffer size

6.1.3 **Adc_DeInit**

Refer section 8.3.3 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-5847	SWS_Adc_00111 : Adc_DeInit Disable all interrupts/notifications
MCAL-5841	SWS_Adc_00112 : Adc_DeInit DET ADC_E_BUSY
MCAL-5838	SWS_Adc_00228 : Adc_DeInit Precompile Switch AdcDeInitApi
MCAL-5792	SWS_Adc_00110 : Adc_DeInit DeInitialize HW Units
MCAL-5784	SWS_Adc_00154 : Adc_DeInit DET ADC_E_UNINIT
MCAL-5748	SWS_Adc_00358 : Adc_DeInit Group state not idle

6.1.4 Adc_StartGroupConversion

Refer section 8.3.4 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-5891	SWS_Adc_00061 : Adc_StartGroupConversion Conversion mode

Design Identifier	Description
MCAL-5872	SWS_Adc_00351 : Adc_StartGroupConversion DET ADC_E_BUSY
MCAL-5869	SWS_Adc_00259 : Adc_StartGroupConversion Precompile Switch AdcEnableStartStopGroupApi
MCAL-5846	SWS_Adc_00424 : Adc_StartGroupConversion DET ADC_E_BUFFER_UNINIT
MCAL-5825	SWS_Adc_00125 : Adc_StartGroupConversion DET ADC_E_PARAM_GROUP
MCAL-5819	SWS_Adc_00346 : Adc_StartGroupConversion DET ADC_E_BUSY
MCAL-5809	SWS_Adc_00133 : Adc_StartGroupConversion DET ADC_E_WRONG_TRIGG_SRC
MCAL-5790	SWS_Adc_00146 : Adc_StartGroupConversion Trigger source
MCAL-5787	SWS_Adc_00156 : Adc_StartGroupConversion Notification mechanism
MCAL-5759	SWS_Adc_00426 : Adc_StartGroupConversion DET ADC_E_BUSY
MCAL-5751	SWS_Adc_00348 : Adc_StartGroupConversion DET ADC_E_BUSY
MCAL-5738	SWS_Adc_00431 : Adc_StartGroupConversion Initialize result buffer pointer

Design Identifier	Description
MCAL-5725	SWS_Adc_00294 : Adc_StartGroupConversion DET ADC_E_UNINIT
MCAL-5720	SWS_Adc_00428 : Adc_StartGroupConversion DET ADC_E_BUSY
MCAL-5713	SWS_Adc_00427 : Adc_StartGroupConversion DET ADC_E_BUSY

6.1.5 Adc_StopGroupConversion

Refer section 8.3.5 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-5873	SWS_Adc_00386 : Adc_StopGroupConversion Continuous mode stop ongoing conversion
MCAL-5870	SWS_Adc_00437 : Adc_StopGroupConversion One shot mode start/restart request
MCAL-5867	SWS_Adc_00438 : Adc_StopGroupConversion Continuous mode start/restart request
MCAL-5850	SWS_Adc_00126 : Adc_StopGroupConversion DET ADC_E_PARAM_GROUP

Design Identifier	Description
MCAL-5842	SWS_Adc_00241 : Adc_StopGroupConversion DET ADC_E_IDLE
MCAL-5835	SWS_Adc_00283 : Adc_StopGroupConversion Trigger source
MCAL-5794	SWS_Adc_00260 : Adc_StopGroupConversion Precompile Switch AdcEnableStartStopGroupApi
MCAL-5785	SWS_Adc_00295 : Adc_StopGroupConversion DET ADC_E_UNINIT
MCAL-5777	SWS_Adc_00164 : Adc_StopGroupConversion DET ADC_E_WRONG_TRIGG_SRC
MCAL-5763	SWS_Adc_00385 : Adc_StopGroupConversion Conversion mode
MCAL-5746	SWS_Adc_00360 : Adc_StopGroupConversion Set group status
MCAL-5718	SWS_Adc_00155 : Adc_StopGroupConversion Notification mechanism

6.1.6 **Adc_ReadGroup**

Refer section 8.3.6 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-5868	SWS_Adc_00330 : Adc_ReadGroup State transition ADC_STREAM_COMPLETED->ADC_IDLE
MCAL-5866	SWS_Adc_00329 : Adc_ReadGroup State transition ADC_STREAM_COMPLETED->ADC_BUSY
MCAL-5851	SWS_Adc_00296 : Adc_ReadGroup DET ADC_E_UNINIT
MCAL-5828	SWS_Adc_00503 : ADC Reentrancy Adc_ReadGroup and Adc_GetGroupStatus
MCAL-5824	SWS_Adc_00152 : Adc_ReadGroup DET ADC_E_PARAM_GROUP
MCAL-5803	SWS_Adc_00122 : Adc_ReadGroup Diagnostic information
MCAL-5796	SWS_Adc_00388 : Adc_ReadGroup DET ADC_E_IDLE
MCAL-5766	SWS_Adc_00113 : Adc_ReadGroup Configuration parameter ADC_RESULT_ALIGNMENT
MCAL-5745	SWS_Adc_00331 : Adc_ReadGroup State transition ADC_COMPLETED->ADC_BUSY
MCAL-5727	SWS_Adc_00359 : Adc_ReadGroup Precompile Switch AdcReadGroupApi
MCAL-5715	SWS_Adc_00075 : Adc_ReadGroup Latest available result

6.1.7 Adc_EnableGroupNotification

Refer section 8.3.9 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-5863	SWS_Adc_00165 : Adc_EnableGroupNotification DET ADC_E_NOTIF_CAPABILITY
MCAL-5818	SWS_Adc_00299 : Adc_EnableGroupNotification DET ADC_E_UNINIT
MCAL-5776	SWS_Adc_00130 : Adc_EnableGroupNotification DET ADC_E_PARAM_GROUP
MCAL-5775	SWS_Adc_00057 : Adc_EnableGroupNotification Notification mechanism
MCAL-5729	SWS_Adc_00100 : Adc_EnableGroupNotification Precompile Switch AdcGrpNotifCapability

6.1.8 Adc_DisableGroupNotification

Refer section 8.3.10 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-5878	SWS_Adc_00058 : Adc_DisableGroupNotification Notification mechanism

Design Identifier	Description
MCAL-5877	SWS_Adc_00131 : Adc_DisableGroupNotification DET ADC_E_PARAM_GROUP
MCAL-5813	SWS_Adc_00166 : Adc_DisableGroupNotification DET ADC_E_NOTIF_CAPABILITY
MCAL-5772	SWS_Adc_00101 : Adc_DisableGroupNotification Precompile Switch AdcGrpNotifCapability
MCAL-5737	SWS_Adc_00300 : Adc_DisableGroupNotification DET ADC_E_UNINIT

6.1.9 Adc_GetGroupStatus

Refer section 8.3.11 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-5892	SWS_Adc_00221 : Adc_GetGroupStatus ADC_IDLE
MCAL-5887	SWS_Adc_00305 : Adc_GetGroupStatus Consistency check
MCAL-5860	SWS_Adc_00325 : Adc_GetGroupStatus ADC_STREAM_COMPLETED

Design Identifier	Description
MCAL-5857	SWS_Adc_00224 : Adc_GetGroupStatus ADC_COMPLETED
MCAL-5828	SWS_Adc_00503 : ADC Reentrancy Adc_ReadGroup and Adc_GetGroupStatus
MCAL-5810	SWS_Adc_00301 : Adc_GetGroupStatus DET ADC_E_UNINIT
MCAL-5802	SWS_Adc_00222 : Adc_GetGroupStatus ADC_BUSY
MCAL-5778	SWS_Adc_00220 : Adc_GetGroupStatus Conversion status
MCAL-5743	SWS_Adc_00225 : Adc_GetGroupStatus DET ADC_E_PARAM_GROUP
MCAL-5741	SWS_Adc_00436 : Adc_GetGroupStatus Status Aborted/suspended group
MCAL-5704	SWS_Adc_00226 : Adc_GetGroupStatus Atomic access

6.1.10 **Adc_GetStreamLastPointer**

Refer section 8.3.12 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-5862	SWS_Adc_00302 : Adc_GetStreamLastPointer DET ADC_E_UNINIT
MCAL-5853	SWS_Adc_00326 : Adc_GetStreamLastPointer State transition ADC_STREAM_COMPLETED->ADC_BUSY
MCAL-5806	SWS_Adc_00219 : Adc_GetStreamLastPointer Consistency check
MCAL-5804	SWS_Adc_00327 : Adc_GetStreamLastPointer State transition ADC_STREAM_COMPLETED->ADC_IDLE
MCAL-5783	SWS_Adc_00215 : Adc_GetStreamLastPointer DET ADC_E_IDLE
MCAL-5780	SWS_Adc_00216 : Adc_GetStreamLastPointer Initialize result buffer pointer NULL
MCAL-5731	SWS_Adc_00382 : ADC Module support Adc_GetStreamLastPointer
MCAL-5730	SWS_Adc_00214 : Adc_GetStreamLastPointer Initialize result buffer pointer
MCAL-5728	SWS_Adc_00328 : Adc_GetStreamLastPointer State transition ADC_COMPLETED->ADC_IDLE
MCAL-5719	SWS_Adc_00418 : Adc_GetStreamLastPointer Configuration parameter ADC_RESULT_ALIGNMENT
MCAL-5711	SWS_Adc_00387 : Adc_GetStreamLastPointer Return Valid samples

Design Identifier	Description
MCAL-5707	SWS_Adc_00218 : Adc_GetStreamLastPointer DET ADC_E_PARAM_GROUP

6.1.11 Adc_GetVersionInfo

Refer section 8.3.13 of the *ADC Driver* AutoSar Specification as listed in Reference 1 - AUTOSAR 4.3.1.

Design Identifier	Description
MCAL-5830	SWS_Adc_00458 : Adc_GetVersionInfo DET ADC_E_PARAM_POINTER
MCAL-5691	ECUC_Adc_00409 : AdcVersionInfoApi

6.1.12 Adc_RegisterReadback

As noted from previous implementation, the adc configuration registers could be potentially corrupted by other entities (s/w or h/w). One of the recommended detection methods would be to periodically read-back the configuration and confirm configuration is consistent. The service API defined below shall be implemented to enable this detection. Constraint: Should be called only after module initialization.

	Description	Comments

Service Name	Adc_RegisterReadback	Can be potentially turned OFF
Syntax	Std_ReturnType Adc_RegisterReadback(Adc_HWUnitType HWUnit, **Adc_RegisterReadbackType** *RegRbPtr)	**Adc_RegisterReadbackType** defines the type, that holds critical values, refer below
Sync / Async	Sync	
Reentrancy	Non Reentrant	
Parameter in	HWUnit	ADC Hardware microcontroller peripheral unit ID
Parameters out	RegRbPtr	A pointer of type **Adc_RegisterReadbackType**, which holds the read back values

Return Value	Standard return type	E_OK or E_NOT_OK in case of Adc not initialized or NULL buffer pointer

The critical register listed is a recommendation and implementation shall determine appropriate registers.

This service could potentially be turned OFF in the configurator.

Design Identifier	Description
MCAL-5864	ADC: Safety Diagnostics: ADC4: Software readback of written configuration
MCAL-5771	ADC: Safety Diagnostics: ADC5: Periodic software readback of static configuration registers
MCAL-5671	AdcEnableRegisterReadbackApi

6.1.13 Adc_EnableInternalDiagnostic

The safety diagnostic implementation of ADC1A - Converted Self-Test and ADC1B- Converter Calibration Test can be done in Functional Internal Diagnostic Debug Mode. This API enables the Functional Internal Diagnostic Debug mode and selects a known voltage source(REFP or VMID) that connects to the AFE. Constraint: Should be called only after module initialization.

This service could potentially be turned OFF in the configurator.

	Description	Comments
Service Name	Adc_EnableInternalDiagnostic	Can be potentially turned OFF
Syntax	Std_ReturnType Adc_EnableInternalDiagnostic(Adc_GroupType Group, Adc_RefSelType RefSelect)	
Sync / Async	Sync	
Reentrancy	Non Reentrant	
Parameter in	Group RefSelect	ADC Channel Group Reference Select for functional internal diagnostic debug mode

Parameters out	None	
Return Value	Standard return type	E_OK or E_NOT_OK in case of Adc not initialized or invalid group or conversion in progress

Design Identifier	Description
MCAL-5793	ADC: Safety Diagnostics: ADC1A - Converter Self-Test
MCAL-5726	ADC: Safety Diagnostics: ADC1B: Converter Calibration Test

6.1.14 Adc_DisableInternalDiagnostic

This API disables the Functional Internal Diagnostic Debug mode. Constraint: Should be called only after module initialization.

This service could potentially be turned OFF in the configurator.

	Description	Comments

Service Name	Adc_DisableInternalDiagnostic	Can be potentially turned OFF
Syntax	Std_ReturnType Adc_DisableInternalDiagnostic(Adc_GroupType Group)	
Sync / Async	Sync	
Reentrancy	Non Reentrant	
Parameter in	Group	ADC Channel Group
Parameters out	None	
Return Value	Standard return type	E_OK or E_NOT_OK in case of Adc not initialized or invalid group or conversion in progress

Design Identifier	Description
MCAL-5793	ADC: Safety Diagnostics: ADC1A - Converter Self-Test
MCAL-5726	ADC: Safety Diagnostics: ADC1B: Converter Calibration Test

7 Performance Objectives

7.1 Resource Consumption Objectives

ROM - Program(KB)	ROM - Data(KB)	RAM - Program(KB)	RAM - Data(KB)	EEPROM (KB)	% CPU Utilization
30	NA	4	NA	NA	NA

7.2 Critical timing and Performance

Not Applicable

8 Decision Analysis & Resolution (DAR)

Sections below list some of the important design decisions and rational behind those decision.

8.1 ADC Conversion Mode

The ADC conversion can be achieved by DMA or through interrupt (CPU) mode. The mode chosen can have system wide effect and important to choose the right method.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
1	Minimal restrictions on the system and guarantee ADC output value within threshold.	**DMA Mode:** The ADC module generates DMA events to the system PDMA as soon as the FIFO threshold is reached. This can be used by the PDMA to read the FIFO content. **Advantages:** • CPU loading is low and constant, irrespective of the ADC conversion clock. • Less probability of FIFO overflow in continuous mode as the DMA copy happens without CPU intervention. **Disadvantages:** • Complexity involved in designing the PDMA parameters for different modes of conversion like Linear, Circular. • Cache coherency needs to be taken care. This will result in Cache module dependency in driver or in the AUTOSAR stack.	Interrupt (CPU) Mode	In case of ADAS use case, the ADC module is used for voltage monitoring. In this case the ADC conversion is triggered by AUTOSAR stack periodically at regular interval. But this regular interval is in 10's or 100's of ms period and rather than at ADC clock rate. So for this case, single shot mode is preferred instead of continuous conversion. Hence the CPU loading is low and there is no chance of FIFO overflow in case of single shot mode. Thus in all respect (complexity, efficiency), CPU mode is sufficient for the ADAS use case. So it is decided to go with interrupt mode.	• CPU loading increases with increasing ADC conversion rate. • High probability of FIFO overflow in continuous mode as the CPU is involved in reading the FIFO.

No.	Decision Criteria	Alternatives	Selected alternative	Rationale	Trade-offs
		• Need of a common DMA complex driver with resource management as the PDMA is at system level and is common across SOC.			
		Interrupt (CPU) Mode : The ADC module generates interrupt events to the core as soon as the FIFO threshold is reached. This can be used by the ISR to read the FIFO content. **Advantages:** • Simple implementation in case of different conversion modes. • No cache coherency is needed and no dependency on cache APIs. **Disadvantages:** • CPU loading increases with increasing ADC conversion rate. • High probability of FIFO overflow in continuous mode as the CPU is involved in reading the FIFO.			

9 Testing Guidelines

The sections below identify some of the aspects of design that would require emphasis during testing of this design implementation

- **Differential Input Test**
    - Test cases shall perform test to check differential input values.
- **State Transitions**
    - Test cases shall exercise all state transitions as detailed in section (States)
    - Ensure non supported API's in a given state, returns valid error code
- **Mode**
    - Test cases shall ensure, adc operable in all supported modes
- **Performance**
    - Test cases shall ensure, adc performance measurement CPU utilization for a given configuration.
- **Stream Access**
    - Test cases shall ensure, adc operable in all stream access mode types
- **Concurrency**
    - Test cases shall ensure, multiple adc instances can be operated concurrently
- **ADC Input Values**
    - Test cases perform equivalence class test on different input range values.
- **Group Logging**
    - Test cases shall ensure, adc operable with group/FIFO error logging on.

10 Template Revision History

Author Name	Description	Version	Date
Yaniv Machani	Initial version	0.1	03 Oct 2018
Yaniv Machani	Updated to include EP views	0.4	02 Nov 2018
Yaniv Weizman	Restructuring and editing to further meet the A-SPICE and EP requirements	0.5	27 Dec 2018
Yaniv Weizman	Adding link to Architecture review template	0.6	22 Oct 2019
Yaniv Weizman	Adding requirement type column for requirements table (Functional/Non-Functional). Adding DAR table	0.65	13 Nov 2019

Author Name	Description	Version	Date
Yaniv Weizman	Adding tables for Testing guidelines	0.7	18 Nov 2019
Krishna	Updated based on ASPICE requirements	0.8	20 Aug 2020
Krishna	Updated based on the feedback from Jon N	0.9	09 Oct 2020
Krishna	Updated the traceability scheme	1.0	17 Dec 2020