Document Title	Specification of ADC Driver
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	2.1.1
Document Status	Final
Part of Release	2.1
Revision	0014

Document Change History

Date	Version	Changed by	Change Description
24.01.2007	2.1.1	AUTOSAR Administration	• "Advice for users" revised • "Revision Information" added
23.11.2006	2.1.0	AUTOSAR Administration	• Removed the "On Demand" functionality. Related services not available anymore. • Removed the "Gated Continuous" conversion mode. Related services not available anymore. • Removed the distinction between internal and external hardware trigger. • Introduced a priority mechanism for channel groups for allowing channel groups with higher priority to interrupt ongoing conversions (can cover also the "On demand" functionality). • Reworked the "Streaming Access Mode". A dedicated data structure for the returned values of a conversion is now clearly defined. • Conversion values access now allowed only through channel groups (no single channel value available. Related service not available anymore).
27.03.2006	2.0.0	AUTOSAR Administration	Document structure adapted to common Release 2.0 SWS Template.
30.06.2005	1.0.0	AUTOSAR Administration	Initial Release.

Page left intentionally blank

Disclaimer

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).
Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration of the AUTOSAR Basic Software module ADC Driver.

The ADC Driver 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 Driver shall work on so called ADC Channels. An ADC channel combines an analogue input pin, the needed ADC circuitry itself and a conversion result register into an entity that can be individually controlled and accessed via the ADC Driver.

2 Acronyms and abbreviations

Abbreviation / Acronym:	*Description:*
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ADC	Analogue Digital Converter
MCU	Microcontroller Unit
API	Application Programming Interface
HW	Hardware
SW	Software

Table 1: Acronyms and abbreviations used in this document

3 Related documentation

3.1 Input documents

[1] General Requirements on Basic Software Modules,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_General.pdf

[2] General Requirements on SPAL,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_SPAL_General.pdf

[3] Specification of Standard Types,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_StandardTypes.pdf

[4] List of Basic Software Modules,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_BasicSoftwareModules.pdf

[5] Specification of Diagnostics Event Manager,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_DEM.pdf

[6] Specification of Development Error Tracer,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_DET.pdf

[7] Requirements on ADC Driver,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SRS_ADC_Driver.pdf

[8] Specification of ECU Configuration,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_ECU_Configuration.pdf

[9] Layered Software Architecture,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_LayeredSoftwareArchitecture.pdf

[10] Specification of ECU State Manager,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_ECU_StateManager.pdf

[11] Specification of I/O Hardware Abstraction,
https://svn.autosar.org/repos/10Releases/
AUTOSAR_SWS_IOHW_Abstraction.pdf

3.2 Related standards and norms

[12] HIS Specification I/O Drivers
http://www.automotive-his.de/download/
API_IODriver_2_1_3.pdf

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

Module DET
In development mode the Development Error Tracer (DET) will be called whenever a development error is encountered by this module.

Module DEM
Production errors will be reported to the Diagnostic Event Manager (DEM).

Module MCU Driver
The Microcontroller Unit Driver (MCU Driver) is primarily responsible for initializing and controlling the chip's internal clock sources and clock prescalers. The clock frequency may affect:

- Trigger frequency.
- Conversion time.

Module PORT driver
Port pins used by the ADC Driver shall be configured using the PORT module. Both analogue input pins and external trigger pins have to be considered.

5.1 File structure

5.1.1 Code file structure

ADC240: The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files named:

- <Module Prefix>_Lcfg.c – for link time configurable parameters and
- <Module Prefix>_PBcfg.c – for post build time configurable parameters.

These files shall contain all link time and post-build time configurable parameters.

5.1.2 Header file structure

ADC267: The file include structure shall be as follows.

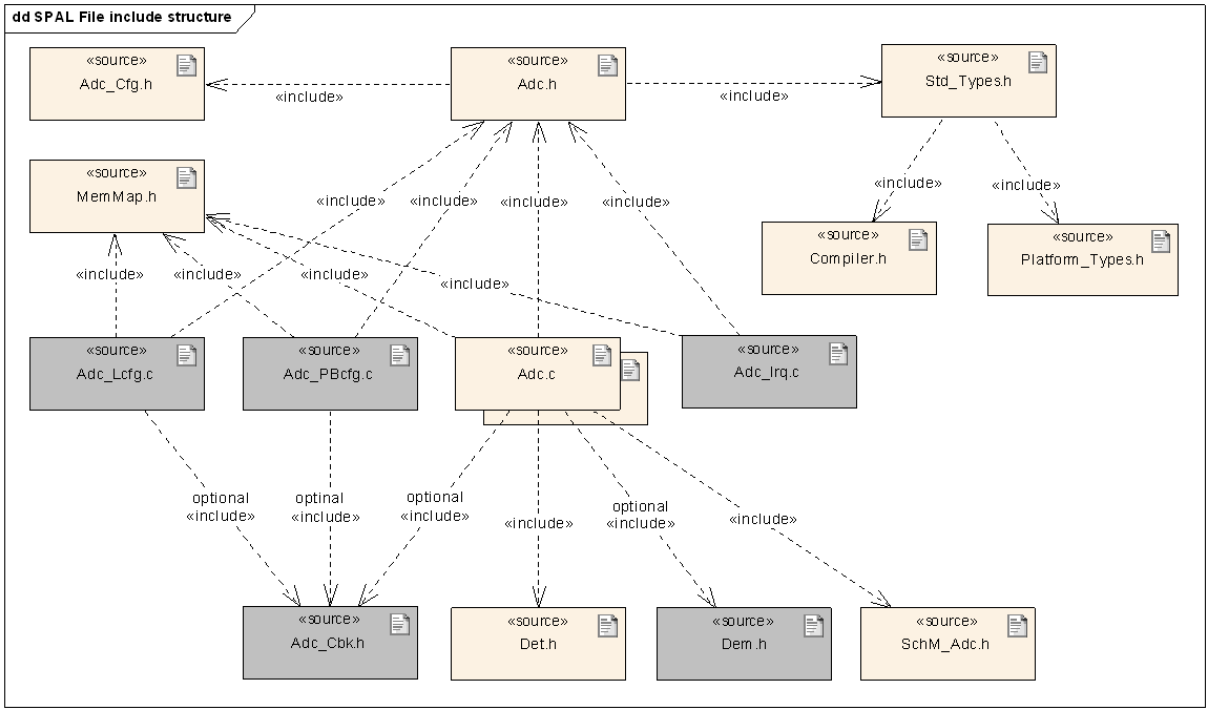

Figure 1: ADC Driver file include structure

ADC239: The module shall include the Dem.h file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

6 Requirements traceability

Document: General Requirements on Basic Software Modules

Requirements	***Satisfied***
[BSW00344] Reference to link-time configuration	Not applicable. (No link time configuration parameters defined for this module).
[BSW00404] Reference to post build time configuration	**ADC028**
[BSW00405] Reference to multiple configuration sets	**ADC054**, **ADC242**
[BSW00345] Pre-compile-time configuration	**ADC027**, **ADC275**
[BSW159] Tool-based configuration	Both static and runtime configuration parameters are located outside the source code of the module. This is the prerequisite for automatic configuration.
[BSW167] Static configuration checking	Not applicable. (Requirement on configuration tool).
[BSW171] Configurability of optional functionality	**ADC120**, **ADC121**, **ADC228**, **ADC237**, **ADC259**, **ADC260**, **ADC265**, **ADC266**
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable. (No reconfiguration and not a SWC)
[BSW00380] Separate C-File for configuration parameters	**ADC240**
[BSW00419] Separate C-Files for pre-compile time configuration parameters	**ADC240**
[BSW00381] Separate configuration header file for pre-compile time parameters	**ADC267**
[BSW00412] Separate H-File for configuration parameters	**ADC267**
[BSW00383] List dependencies of configuration files	**ADC267**
[BSW00384] List dependencies to other modules	See chapter 5.
[BSW00387] Specify the configuration class of call-back function	Not applicable. (This module does not provide any callback routines).
[BSW00388] Introduce containers	**ADC027**, **ADC028**, **ADC242**, **ADC268**
[BSW00389] Containers shall have names	**ADC027**, **ADC028**, **ADC242**, **ADC268**
[BSW00390] Parameter content shall be unique within the module	**ADC027**, **ADC028**, **ADC242**, **ADC268**
[BSW00391] Parameter shall have unique names	**ADC027**, **ADC028**, **ADC242**, **ADC268**
[BSW00392] Parameters shall have a type	**ADC027**, **ADC028**, **ADC242**, **ADC268**
[BSW00393] Parameters shall have a range	**ADC027**, **ADC028**, **ADC242**, **ADC268**
[BSW00394] Specify the scope of the parameters	**ADC027**, **ADC028**, **ADC242**, **ADC268**
[BSW00395] List the required parameters (per parameter)	**ADC027**, **ADC028**, **ADC242**, **ADC268**
[BSW00396] Configuration classes	**ADC027**, **ADC028**, **ADC242**, **ADC268**
[BSW00397] Pre-compile-time parameters	**ADC027**, **ADC242**, **ADC268**
[BSW00398] Link-time parameters	Not applicable. (No link time configuration parameters defined for this module).
[BSW00399] Loadable Post-build time parameters	**ADC028**

Requirements	*Satisfied*
[BSW00400] Selectable Post-build time parameters	ADC028
[BSW00402] Published information	ADC030
[BSW00375] Notification of wake-up reason	Not applicable. (This module does not provide any wake-up reason).
[BSW101] Initialization interface	ADC054
[BSW00416] Sequence of Initialization	Not applicable. (SW Integration requirement).
[BSW00406] Check module initialization	ADC068, ADC107
[BSW168] Diagnostic Interface of SW components	Not applicable (This module does not support a special diagnostic interface).
[BSW00407] Function to read out published parameters	ADC236, ADC237
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable. (driver has no AUTOSAR interfaces).
[BSW00424] BSW main processing function task allocation	Not applicable (This module does not provide a schedulable main function).
[BSW00425] Trigger conditions for schedulable objects	Not applicable. (Requirement on implementation, not on specification).
[BSW00426] Exclusive areas in BSW modules	Not applicable. (Requirement on implementation, not on specification).
[BSW00427] ISR description for BSW modules	Not applicable. (Requirement on implementation, not on specification).
[BSW00428] Execution order dependencies of main processing functions	Not applicable. (Requirement on implementation, not on specification).
[BSW00429] Restricted BSW OS functionality access	Not applicable. (Requirement on implementation, not on specification).
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable. (Requirement on implementation, not on specification).
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable. (This module does not provide a schedulable main function).
[BSW00433] Calling of main processing functions	Not applicable. (This is a general requirement).
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable. (This is a special requirement for the BSW scheduler).
[BSW00336] Shutdown interface	ADC111
[BSW00337] Classification of errors	ADC065, ADC069, ADC229, ADC230
[BSW00338] Detection and Reporting of development errors	ADC233, ADC234, ADC067
[BSW00369] Do not return development error codes via API	ADC233, ADC234, ADC067
[BSW00339] Reporting of production relevant error and exceptions	ADC068, ADC069, ADC235, ADC239
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable. (Module is a BSW).
[BSW00323] API parameter checking	ADC065, ADC125, ADC126, ADC128, ADC129, ADC130, ADC131, ADC152, ADC225, ADC241, ADC269

Requirements	*Satisfied*
[BSW004] Version check	ADC030, ADC124
[BSW00409] Header files for production code error IDs	ADC239
[BSW00385] List possible error notifications	ADC065, ADC069
[BSW00386] Configuration for detecting an error	ADC068, ADC069, ADC107, ADC112, ADC125, ADC126, ADC128, ADC129, ADC130, ADC131, ADC133, ADC136, ADC137, ADC152, ADC154, ADC164, ADC165, ADC166, ADC225, ADC233, ADC241, ADC269, ADC218
[BSW161] Microcontroller abstraction	Not applicable. (Architectural AUTOSAR concept is the basis for this driver).
[BSW162] ECU layout abstraction	Not applicable. (Architectural AUTOSAR concept is the basis for this driver).
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable. (Architectural AUTOSAR concept is the basis for this driver).
[BSW00415] User dependent include files	ADC267
[BSW164] Implementation of interrupt service routines	Not applicable. (ADC driver is a part of microcontroller abstraction layer).
[BSW00325] Runtime of interrupt service routines	Not applicable. (Requirement on implementation, not on specification).
[BSW00326] Transition from ISRs to OS tasks	Not applicable. (Requirement on implementation, not on specification).
[BSW00342] Usage of source code and object code	Not applicable. (Requirement on implementation, not on specification).
[BSW00343] Specification and configuration of time	Not applicable. (Requirement on implementation, not on specification).
[BSW160] Human-readable configuration data	Not applicable. (Requirement on implementation, not on specification).
[BSW007] HIS MISRA C	Not applicable. (Requirement on implementation, not on specification).
[BSW00300] Module naming convention	ADC267
[BSW00413] Accessing instances of BSW modules	Not applicable (requirement on implementation, not on specification)
[BSW00347] Naming separation of different instances of BSW drivers	Not applicable. (Requirement on implementation, not on specification).
[BSW00305] Self-defined data types naming convention	Chapter 8.2.
[BSW00307] Global variables naming convention	Not applicable. (Requirement on implementation, not on specification).
[BSW00310] API naming convention	Chapter 8.3.
[BSW00373] Main processing function naming convention	Not applicable. (Requirement on implementation, not on specification).
[BSW00327] Error values naming convention	ADC065

Requirements	*Satisfied*
[BSW00335] Status values naming convention	ADC221, ADC222, ADC224
[BSW00350] Development error detection keyword	ADC027, ADC233
[BSW00408] Configuration parameter naming convention	Chapter 10.2.
[BSW00410] Compiler switches shall have defined values	Chapter 10.2.
[BSW00411] Get version info keyword	ADC237
[BSW00346] Basic set of module files	ADC267
[BSW158] Separation of configuration from implementation	ADC027, ADC028, ADC242, ADC267
[BSW00314] Separation of interrupt frames and service routines	ADC267
[BSW00370] Separation of call-back interface from API	ADC267, Chapter 8.4.
[BSW00348] Standard type header	ADC267, Chapter 8.1.1
[BSW00353] Platform specific type header	ADC267, Chapter 8.1.1.
[BSW00361] Compiler specific language extension header	ADC267
[BSW00301] Limit imported information	Not applicable. (Requirement on implementation, not on specification).
[BSW00302] Limit exported information	Not applicable. (Requirement on implementation, not on specification).
[BSW00328] Avoid duplication of code	Not applicable. (Requirement on implementation, not on specification).
[BSW00312] Shared code shall be reentrant	Not applicable. (Requirement on implementation, not on specification).
[BSW006] Platform independency	Not applicable. (Requirement on implementation, not on specification).
[BSW00357] Standard API return type	Not applicable. (Type not used in this module).
[BSW00377] Module specific API return types	Chapter 8.3.10.
[BSW00304] AUTOSAR integer data types	Chapter 8.2, Chapter 0.
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable. (No integer data types redefined in this specification).
[BSW00378] AUTOSAR boolean type	Chapter 10.2.
[BSW00306] Avoid direct use of compiler and platform specific keywords	Not applicable. (Requirement on implementation, not on specification).
[BSW00308] Definition of global data	Not applicable. (Requirement on implementation, not on specification).
[BSW00309] Global data with read-only constraint	Chapter 8.3.1.
[BSW00371] Do not pass function pointers via API	Not applicable. (Requirement on implementation, not on specification).
[BSW00358] Return type of init() functions	Chapter 8.3.1.
[BSW00414] Parameter of init function	Chapter 8.3.1, ADC275
[BSW00376] Return type and parameters of main processing functions	Not applicable. (This module does not provide a schedulable main function).
[BSW00359] Return type of call-back functions	ADC270
[BSW00360] Parameters of call-back functions	ADC270
[BSW00329] Avoidance of generic interfaces	Not applicable. (No generic interface in this module. See chapter 8.3).
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable. (Requirement on implementation,

Requirements	Satisfied
	not on specification).
[BSW00331] Separation of error and status values	ADC065, ADC269
[BSW009] Module User Documentation	Not applicable. (Requirement for documentation not for module specification).
[BSW00401] Documentation of multiple instances of configuration parameters	Chapter 10.2
[BSW172] Compatibility and documentation of scheduling strategy	Chapter 8.3
[BSW010] Memory resource documentation	Not applicable. (Requirement on implementation, not on specification).
[BSW00333] Documentation of call-back function context	Chapter 8.6.3, ADC153
[BSW00374] Module vendor identification	ADC030
[BSW00379] Module identification	ADC030
[BSW003] Version identification	ADC030
[BSW00318] Format of module version numbers	ADC030
[BSW00321] Enumeration of module version numbers	ADC030
[BSW00341] Microcontroller compatibility documentation	Not applicable. (Requirement on implementation, not on specification).
[BSW00334] Provision of XML file	Not applicable. (Requirement on implementation, not on specification).
[BSW00435] Module header file structure for the basic software scheduler	ADC267
[BSW00436] Module header file structure for the basic software memory mapping	ADC267

Document: General Requirements on SPAL

Requirements	Satisfied by
[BSW12263] Object code compatible configuration concept	ADC028, ADC268
[BSW12056] Configuration of notification mechanisms	ADC079, ADC080, ADC084, ADC085,
[BSW12267] Configuration of wake-up sources	Not applicable. (This module does not provide any wake-up reason).
[BSW12057] Driver module initialization	ADC054
[BSW12125] Initialization of hardware resources	ADC056
[BSW12163] Driver module deinitialization	ADC110, ADC111
[BSW12461] Responsibility for register initialization	ADC054, ADC147ADC246, ADC247, ADC248, ADC249, ADC250
[BSW12462] Provide settings for register initialization	Chapter 10.2.
[BSW12463] Combine and forward settings for register initialization	Not applicable. (Applies only for configuration tool).
[BSW12068] MCAL initialization sequence	Not applicable. (This is a general software integration requirement).
[BSW12069] Wake-up notification of ECU State Manager	Not applicable. (This module does not provide any wake-up reason).
[BSW157] Notification mechanisms of drivers and handlers	ADC057, ADC058, ADC082, ADC083, ADC104

Requirements	***Satisfied by***
[BSW12169] Control of operation mode	Not applicable. (The module does not support different modes).
[BSW12063] Raw value mode	**ADC113**
[BSW12075] Use of application buffers	Not applicable. (No applications buffers used).
[BSW12129] Resetting of interrupt flags	**ADC078**
[BSW12064] Change of operation mode during running operation	Not applicable. (The module does not support different modes).
[BSW12448] Behavior after development error detection	**ADC065**, **ADC107**, **ADC112**, **ADC125**, **ADC126**, **ADC128**, **ADC129**, **ADC130**, **ADC131**, **ADC133**, **ADC136**, **ADC137**, **ADC152**, **ADC154**, **ADC164**, **ADC165**, **ADC166**, **ADC225**, **ADC241**, **ADC269**
[BSW12067] Setting of wake-up conditions	Not applicable. (This module does not provide any wake-up reason).
Non Functional Requirements	***Satisfied by***
[BSW12077] Non-blocking implementation	Not applicable. (Requirement on implementation, not on specification).
[BSW12078] Runtime and memory efficiency	Not applicable. (Requirement on implementation, not on specification).
[BSW12092] Access to drivers	Not applicable. (Requirement on implementation, not on specification).
[BSW12265] Configuration data shall be kept constant	Not applicable. (Requirement on implementation, not on specification).
[BSW12264]Specification of configuration items	Chapter 10.2.
Requirements (module specific)	***Satisfied by***
[BSW12307] ADC channel configuration	**ADC011**, **ADC019**, **ADC290**, **ADC023**, **ADC089**, **ADC099**, **ADC268**, **ADC087**, **ADC088**
[BSW12447] ADC channel group configuration	**ADC001**, **ADC002**, **ADC014**, **ADC094**, **ADC095**, **ADC099**, **ADC100**, **ADC101**, **ADC104**, **ADC105**, **ADC251**, **ADC254**, **ADC287**, **ADC280**, **ADC090**, **ADC291**, **ADC292**, **ADC277**, **ADC098**, **ADC091**
[BSW12817] Configuration of group trigger source	**ADC094**, **ADC095**, ADC119**ADC146**, **ADC279**, **ADC253**, **ADC283**
[BSW12818] Assignment of an ADC channel to multiple ADC channle groups	**ADC092**
[BSW12821] Buffer configuration for stream conversion mode	**ADC291**
[BSW12820] ADC priority for channel groups	**ADC288**, **ADC289**, **ADC287**
[BSW12280] ADC channel group results access mode	**ADC140**, **ADC252**, **ADC291**, **ADC292**, **ADC317**
[BSW12283] Mask out information bits	**ADC122**
[BSW12819] ADC channel group read service	**ADC275**, **ADC113**, **ADC122**, **ADC141**, **ADC291**, **ADC292**, **ADC318**

Requirements	*Satisfied by*
[BSW12822] ADC uniform result structure	ADC291, ADC320
[BSW12317] ADC channel group notification function	ADC081, ADC104, ADC155, ADC156, ADC157
[BSW12291] ADC channel group status service	ADC220, ADC221, ADC222, ADC224, ADC226, ADC255, ADC219
[BSW12318] Enable / disable notification functions	ADC057, ADC058, ADC077, ADC156, ADC157
[BSW12364] Start and stop conversion of an ADC channel group	ADC061, ADC072, ADC145, ADC146, ADC157, ADC253, ADC255, ADC060, ADC276
[BSW12292] Handling of signed values	ADC113, ADC214
[BSW12288] ADC stream buffer handling	ADC291, ADC292
[BSW12802] Identify most recent sample and number of available samples	ADC213ADC214, ADC215, ADC216, ADC217, ADC219
[BSW12823] Enable / Disable Hardware Triggers	ADC114, ADC144, ADC273, ADC281, ADC116, ADC282
[BSW12824] Right-aligned results.	ADC113
[BSW12825] Structure of result buffer for stream conversion mode.	ADC319

7 Functional specification

7.1 General behavior

7.1.1 Background & Rationale

The table below shows a list of possible desired functionalities of an ADC user and in which way they are provided by the ADC Driver. Furthermore the table also depicts a possible realization and the mapping of these functionalities to the capabilities of a commercial microcontroller (C16x).

Desired Functionality	*ADC Driver Function*	*Example: C16x Derivate Wording*
Just one conversion result of a single channel.	Software triggered one-shot conversion where the converted group consists of exactly one channel.	Fixed channel, single conversion, software trigger.
Cyclic conversion of a single channel.	Hardware triggered one-shot conversion where the converted group consists of exactly one channel.	Fixed channel, single conversion, hardware trigger.
Repeated conversion of a single channel.	Continuous conversion where the converted group consists of exactly one channel.	Fixed channel, continuous conversion.
Just one conversion result of each channel within a group.	Software triggered one-shot conversion where the converted group consists of more than one channel.	Auto scan, single conversion, software trigger.
Cyclic conversion of each channel within a group.	Hardware triggered one-shot conversion where the converted group consists of more than one channel.	Auto scan, single conversion, hardware trigger.
Repeated conversion of each channel within a group.	Continuous conversion where the converted group consists of more than one channel.	Auto scan, continuous conversion.

Table 2: Different possibilities of One-shot and Continuous conversions

7.1.2 Requirements

ADC090: The ADC Driver shall allow grouping of one or more ADC channels into so called ADC Channel groups.

ADC091: An ADC Channel Group shall contain at least one ADC Channel.

ADC092: The ADC Driver shall allow the assignment of an ADC channel to more than one group.

ADC277: All channels contained in one ADC Channel group shall belong to the same ADC HW Unit.

ADC251: The ADC Driver shall support the following conversion modes for all ADC Channel groups:

- One-shot conversion: Exactly one conversion is executed for each channel configured for the group being converted.
- Continuous conversion[1]: After it 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.

ADC253: The following start conditions or trigger sources shall be supported:
- Software API Call: The conversion of an ADC Channel group shall be started/stopped with a service provided by the ADC Driver. This start condition shall be available for all conversion modes.
- Hardware Event: 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. This trigger source shall be available only for groups configured in One-Shot conversion mode.

ADC279: The ADC Driver shall allow configuring exactly one trigger source for each ADC Channel group.

ADC252: The ADC driver shall support the following result access modes:
- Single Value Result Access Mode: The ADC Driver shall return the latest conversion result available for the requested ADC Channel.
- Streaming Access Mode: Access to a buffer through a pointer.

ADC140: The ADC Driver shall guarantee the consistency of the returned result value for each completed conversion.

Note: It is the responsibility of the module's user to ensure that a conversion has been completed for the requested channel before requesting the conversion result. If no conversion has been completed for the requested channel (e.g. because the conversion of the ADC Channel group has been stopped by the user) the value returned by the ADC Driver may be arbitrary.

ADC288: The ADC Driver shall allow the configuration of a priority level for each channel group. This implies a prioritization mechanism, implemented in SW, or where available supported by the HW.

Note: The priority mechanism is not related with anything else (HW/SW trigger, conversion mode, etc.) The result is that the higher priority conversion request is always converted no matter of which nature it is.

ADC310: The ADC Driver priority mechanism shall allow aborting/suspending and restarting/resuming of conversions.

ADC311: The ADC Driver priority mechanism shall allow the queuing of requests for different groups.

[1] On some microcontroller also called „auto-scan mode".

Note: Higher priority groups can abort/suspend lower priority groups. In this case the priority handler should put the interrupted conversion in the queue and this conversion will be resumed later, transparently to the user.

ADC312: In the ADC Driver priority mechanism the lowest priority shall be 0.

ADC289: The ADC driver priority mechanism shall allow the configuration of 256 priority levels (0..255). If not fully supported by the hardware a prioritization mechanism implemented in SW is needed.

ADC314: As an optimization, the user of the ADC Driver can restrict the range of available priorities to those that are already provided by the underlying hardware.

ADC315: To further optimize the resource usage of the ADC Driver the user should be able to statically disable the priority mechanism as a whole. This means every ADC Channel Group gets the same priority, there is no suspending/aborting of ongoing conversions, a queuing mechanism might not be needed since only one conversion is handled at a time (first come first served).

ADC309: Normally for each channel group there is only one conversion request at the same time (i.e. the group is not requested for a successive conversion before the previous one has been completed). Nevertheless, there could be some constellation in which this is not true (e.g. the group has a low priority and the time between two consecutive conversions is shorter that the time needed for converting all groups with higher priority). This can be managed for example by implementing the queue mechanism with a set of counters, one for each channel group/priority level, tracing the times a group has been requested for conversion (i.e. put into the queue). When the counter is 0, no conversion is performed for the related group, when it is >0, the given number of conversions is performed.

7.2 Conversion processing and interaction

7.2.1 Background & Rationale

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

- **Example 1**: **Group X** containing channels [CH0, CH1, CH2, CH3, 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**: **Group X** containing channels [CH0, CH1, CH2, CH3, CH4] is configured in One-Shot conversion mode. After finishing the scan the notification (if enabled) is called.
- **Example 3**: **Group X** 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**: **Group X** containing channel [CH4] is configured in One-Shot conversion mode. After finishing the scan the notification (if enabled) is called.

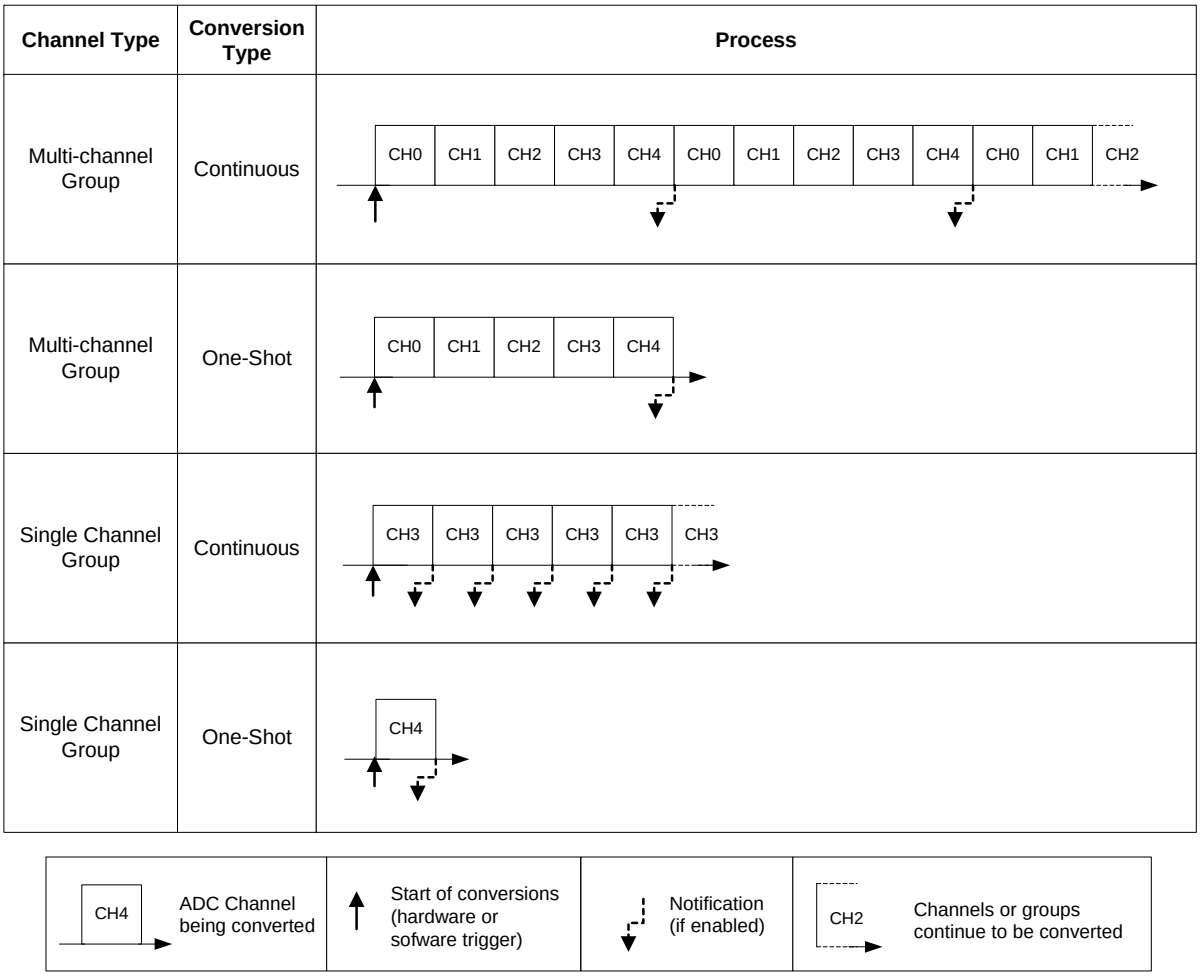

Figure 2: Conversion Mode behavior examples

7.2.2 Requirements

ADC280: The ADC Driver shall convert only one ADC Channel group per ADC HW Unit at a time. Concurrent conversion of different (even exclusive) ADC Channel groups on the same ADC HW Unit shall not be supported by the ADC Driver.

Note: Concurrent conversion of ADC Channel groups on different ADC HW Units may be possible, depending on the capabilities of the hardware. Also concurrent conversion of individual channels within one channel group may be possible if supported by the hardware.

ADC254: If a channel shall be used in different conversion modes (e.g. continuous conversion mode during normal operation and one-shot conversion mode for a special conversion at a dedicated point in time), this channel shall be assigned to different groups configured with the respective conversion modes.

ADC255: In order to change the conversion mode for a channel shared between two groups the ADC user has to stop the conversion of the first group containing the specified channel and then start the conversion of the second group containing the specified channel.

7.3 Version check

7.3.1 Background & Rationale

The integration of incompatible files shall be avoided. Minimum implementation is the version check of the header file inside the .c file (version numbers of .c and .h files shall be identical).

7.3.2 Requirements

ADC124:For included header files
`ADC_AR_MAJOR_VERSION`
`ADC_AR_MINOR_VERSION`
shall be identical.
For the module internal .c and .h files
`ADC_SW_MAJOR_VERSION`
`ADC_SW_MINOR_VERSION`
`ADC_AR_MAJOR_VERSION`
`ADC_AR_MINOR_VERSION`
`ADC_AR_PATCH_VERSION`
shall be identical.

7.4 Error classification

ADC230: Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file Dem_IntErrId.h and included via Dem.h.

ADC229: Development error values are of type uint8.

ADC065: The following errors shall be detectable by the ADC Driver depending on its configuration (development / production mode).

Type of error	*Relevance*	*Related error code*	*Value [hex]*
Adc_Init has not been called prior to another function call (see **ADC154**, **ADC294**, **ADC295**, **ADC296**, **ADC297**, **ADC298**, **ADC299**, **ADC300**, **ADC301**, **ADC302**).	Development	ADC_E_UNINIT	0x0A
Adc_StartGroupConversion was called while another conversion is already running (see **ADC276**). Adc_DeInit was called while a conversion is still ongoing (see **ADC112**).	Development	ADC_E_BUSY	0x0B
Adc_StopGroupConversion was called while no conversion was running (see **ADC241**).	Development	ADC_E_IDLE	0x0C
Invalid group ID requested (see **ADC125**, **ADC126**, **ADC152**, **ADC128**, **ADC129**, **ADC130**, **ADC131**, **ADC225**, **ADC218**, **ADC304**).	Development	ADC_E_PARAM_GROUP	0x15
Adc_EnableHardwareTrigger or Adc_DisableHardwareTrigger called on a group with conversion mode configured as continuous (see **ADC281**, **ADC282**).	Development	ADC_E_WRONG_CONV_MODE	0x16
Adc_StartGroupConversion or Adc_StopGroupConversion called on a group with trigger source configured as hardware (see **ADC133**, **ADC164**). Adc_EnableHardwareTrigger or Adc_DisableHardwareTrigger called on a group with trigger source configured as software API (see **ADC136**, **ADC137**).	Development	ADC_E_WRONG_TRIGG_SRC	0x17
Enable/disable notification function for a group whose configuration set has no notification available (see **ADC165**, **ADC166**).	Development	ADC_E_NOTIF_CAPABILITY	0x18
--	Production	--	Assigned by DEM

Table 3:Error classification

ADC069: Additional errors that are detected because of specific implementation and/or specific hardware properties shall be added in the ADC device specific implementation specification. The classification and enumeration shall be compatible to the errors listed above.

7.5 Error detection

ADC233: The detection of development errors is configurable (ON/OFF) at pre-compile time. The switch ADC_DEV_ERROR_DETECT (see chapter 10.2) shall activate or deactivate the detection of all development errors.

ADC234: If the switch ADC_DEV_ERROR_DETECT is enabled, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.4 and chapter 8.3.

ADC235: The detection of production code errors cannot be switched off.

ADC269: If development error detection is enabled for the ADC Driver, the following API parameter checking shall be performed according to the respective functions (see table below). The error shall be reported to the Development Error Tracer. For description and values of the error codes refer to chapter 7.4.

Function	***Criteria of detection***	***Related error code***
Adc_Init	ADC driver and hardware already initilaized.	ADC_E_UNINIT
Adc_DeInit	Function called prior to initialization. Function called while conversion is running.	ADC_E_UNINIT ADC_E_BUSY
Adc_StartGroupConversion	Function called prior to initialization. Function called while a conversion is already running. Function called with non existing group. Function called for a group configured for hardware trigger source.	ADC_E_UNINIT ADC_E_BUSY ADC_E_PARAM_GROUP ADC_E_WRONG_TRIGG_SRC
Adc_StopGroupConversion	Function called prior to initialization. Function called while no conversion is running any more. Function called with non existing group. Function called for a group configured for hardware trigger source.	ADC_E_UNINIT ADC_E_IDLE ADC_E_PARAM_GROUP ADC_E_WRONG_TRIGG_SRC

Adc_ValueReadGroup	Function called prior to initialization. Function called with non existing group.	ADC_E_UNINIT ADC_E_PARAM_GROUP
Adc_EnableHardwareTrigger	Function called prior to initialization. Function called with non existing group. Function called for a group configured for software API trigger source. Function called for a group configured for Continuous conversion mode.	ADC_E_UNINIT ADC_E_PARAM_GROUP ADC_E_WRONG_TRIGG_SRC ADC_E_WRONG_CONV_MODE
Adc_DisableHardwareTrigger	Function called prior to initialization. Function called with non existing group. Function called for a group configured for software API trigger source. Function called for a group configured for Continuous conversion mode. Function called for a non enabled group.	ADC_E_UNINIT ADC_E_PARAM_GROUP ADC_E_WRONG_TRIGG_SRC ADC_E_WRONG_CONV_MODE ADC_E_PARAM_GROUP
Adc_EnableGroupNotification	Function called prior to initialization. Function called with non existing group.	ADC_E_UNINIT ADC_E_PARAM_GROUP
Adc_DisableGroupNotification	Function called prior to initialization. Function called with non existing group.	ADC_E_UNINIT ADC_E_PARAM_GROUP
Adc_GetGroupStatus	Function called prior to initialization. Function called with non existing group.	ADC_E_UNINIT ADC_E_PARAM_GROUP
Adc_GetStreamLastPointer	Function called prior to initialization. Function called with non existing group.	ADC_E_UNINIT ADC_E_PARAM_GROUP
Adc_GetVersionInfo		

Table 4:Error detection

7.6 Error notification

ADC067: Detected development errors shall be reported to the *Det_ReportError* service of the Development Error Tracer (DET) if the pre-processor switch ADC_DEV_ERROR_DETECT is set (see chapter 10)

ADC068: Production errors shall be reported to the Diagnostic Event Manager

8 API specification

8.1 Imported types

8.1.1 Standard types

- Std_Types.h
  - Std_VersionInfoType

8.2 Type definitions

8.2.1 Adc_ConfigType

Type :	`Structure`
Range :	Implementation specific configuration data structure. See chapter 10.2 for configurable parameters.
Description:	Data structure containing the set of configuration parameters required for initializing the ADC Driver and ADC HW Unit(s).

8.2.2 Adc_ChannelType

Type:	`uint8`
Range	0..255
Description:	Numeric ID of an ADC channel.

8.2.3 Adc_GroupType

Type:	`uint8`
Range	0..255
Description:	Numeric ID of an ADC channel group.

8.2.4 Adc_ValueGroupType

Type:	`uint8..uint32, sint8..sint32`
Range	Implementation specific.
Description:	Used for reading the converted values of a channel group (raw, without further scaling, right aligned). The result values shall be stored in an integer buffer, i.e. and array of integers. The following rules shall apply for the driver implementation: • **ADC318**: In single value access mode the result buffer shall have as many elements as channels belonging to the group. In this way each buffer element corresponds to a channel, in the order the channel are defined in the group. • **ADC319**: In streaming access mode the result buffer shall have `m*n`

	elements, where n is the number of channels belonging to the group, m the number of samples acquired per channel. In this way the first m elements belong to the first channel in the group, the second m elements to the second channel and so on. • **ADC320**: The dimension (in number of bits) of each buffer element (of type integer) shall be uniform, tailored on he largest (in number of bits) channel belonging to the group. *Note: The information about number of channels belonging to the group and number of samples acquired per channel can be derived from the group configuration data.*

8.2.5 Adc_ClockSourceType

Type:	uint8..uint32
Range	The range of this type is µC specific and has to be described by the supplier.
Description:	Type of clock input for the conversion unit to select different clock sources, if provided by hardware. (This is not an API type).

8.2.6 Adc_PrescaleType

Type:	uint8..uint32
Range	The range of this type is µC specific and has to be described by the supplier.
Description:	Type of clock prescaler factor. (This is not an API type).

8.2.7 Adc_ConversionTimeType

Type:	uint8..uint32
Range	The range of this type is µC specific and has to be described by the supplier.
Description:	Type of conversion time, i.e. the time during which the sampled analogue value is converted into digital representation. (This is not an API type).

8.2.8 Adc_SamplingTimeType

Type:	uint8..uint32
Range	The range of this type is µC specific and has to be described by the supplier.
Description:	Type of sampling time, i.e. the time during which the value is sampled, (in clock-cycles). (This is not an API type).

8.2.9 Adc_VoltageSourceType

Type:	`uint8..uint32, sint8..sint32`
Range	The range of this type is µC specific and has to be described by the supplier.
Description:	Type of reference voltage source. (This is not an API type).

8.2.10 Adc_ResolutionType

Type:	`uint8`
Range	The range of this type is µC specific and has to be described by the supplier.
Description:	Type of channel resolution in number of bits. (This is not an API type).

8.2.11 Adc_StatusType

Type:	`Enumeration`	
Range	`ADC_IDLE`	• The conversion of the specified group has not been started. • No result is available.
	`ADC_BUSY`	• The conversion of the specified group has been started and is still going on. • So far no result is available.
	`ADC_COMPLETED`	• The (first) conversion of the specified group has been finished (further conversions might still be going on). • A (first) result is available.
Description:	Current status of the conversion of the requested ADC Channel group (see chapter 8.3.10).	

8.2.12 Adc_TriggerSourceType

Type:	`Enumeration`	
Range	`ADC_TRIGG_SRC_SW`	Group is triggered by a software API call.
	`ADC_TRIGG_SRC_HW`	Group is triggered by a hardware event.
Description:	Type for configuring the trigger source for an ADC Channel group.	

8.2.13 Adc_GroupConvModeType

Type:	`Enumeration`	
Range	`ADC_CONV_MODE_ONESHOT`	Exactly one conversion of each channel in an ADC channel group is performed after the configured trigger event. A started One-Shot conversion can be stopped by a software API call.
	`ADC_CONV_MODE_CONTINUOUS`	Repeated conversions of each ADC Channel in an ADC Channel group are performed as long as the trigger event is active. A started Continuous conversion can be stopped by a software API call or by disabling the trigger event.
Description:	Type for configuring the conversion mode of an ADC Channel group.	

8.2.14 Adc_GroupPriorityType

Type:	`uint8`
Range	`0..255`
Description:	Priority level of the channel. Lowest priority is 0.

8.2.15 Adc_GroupDefType

Type:	`Structure`
Range	Implementation specific configuration data structure.
Description:	Type of assignment of channels to a channel group (this not an API type).

8.2.16 Adc_PointerBufferType

Type:	`uint8..uint32`
Range	The range of this type is µC specific and has to be described by the supplier.
Description:	Type of ADC user data buffer elements.

8.2.17 Adc_StreamNumSampleType

Type:	`uint8..uint32`
Range	The range of this type is µC specific and has to be described by the supplier.
Description:	Type for the number of samples of a streaming conversion buffer.

8.2.18 Adc_HwUnitType

Type:	`uint8`
Range	0..255
Description:	Numeric ID of an ADC Hw Unit.

8.2.19 Adc_StreamBufferModeType

Type:	`Enumeration`	
Range	`ADC_STREAM_BUFFER_LINEAR`	The ADC Driver stops the conversion as soon as the stream buffer is full (number of samples reached).
	`ADC_STREAM_BUFFER_CIRCULAR`	The ADC Driver continue the conversion even if the stream buffer is full (number of samples reached) by wrapping around the stream buffer itself.
Description:	Type for configuring the streaming access mode buffer type.	

8.2.20 Adc_GroupAccessModeType

Type:	`Enumeration`	
Range	`ADC_ACCESS_MODE_SINGLE`	Single value access mode.
	`ADC_ACCESS_MODE_STREAMING`	Streaming access mode.
Description:	Type for configuring the access mode to group conversion results.	

8.2.21 Adc_HwTriggerSignalType

Type:	`Enumeration`	
Range	`ADC_HW_TRIG_RISING_EDGE`	React on the rising edge of the hardware trigger signal (only if supported by the ADC hardware).
	`ADC_HW_TRIG_FALLING_EDGE`	React on the falling edge of the hardware trigger signal (only if supported by the ADC hardware).
	`ADC_HW_TRIG_BOTH_EDGES`	React on both edges of the hardware trigger signal (only if supported by the ADC hardware).
Description:	Type for configuring on which edge of the hardware trigger signal the driver should reach, i.e. start the conversion (only if supported by the ADC hardware).	

8.2.22 Adc_HwTriggerTimerType

Type:	`uint8..uint32`
Range	The range of this type is µC specific and has to be described by the supplier.
Description:	Type for the reload value of the ADC module embedded timer (only if supported by the ADC hardware).

8.3 Function definitions

8.3.1 Adc_Init

Service name:	Adc_Init	
Syntax :	`void Adc_Init ( const Adc_ConfigType *ConfigPtr )`	
Service ID [hex] :	0x00	
Sync/Async :	Synchronous	
Reentrancy :	Non re-entrant	
Parameters (in):	`ConfigPtr`	Pointer to configuration set.
Parameters (out):	None	
Return value:	None	
Description:	**ADC054**: This service shall initialize the ADC hardware unit and driver according to the configuration set referenced by `ConfigPtr`. **ADC056**:This function shall only initialize the configured resources. Resources that are not contained in the configuration file shall not be touched. The following rules regarding initialization of controller registers shall apply to this driver implementations: ▪ **ADC246**: If the hardware allows for only one usage of the register, the driver module implementing that functionality is responsible for initializing the register. ▪ **ADC247**: If the register can affect several hardware modules and if it is an I/O register, it shall be initialized by the PORT driver. ▪ **ADC248**: If the register can affect several hardware modules and if it is not an I/O register, it shall be initialized by the MCU driver. ▪ **ADC249**: One-time writable registers that require initialization directly after reset shall be initialized by the startup code. ▪ **ADC250**: All other registers shall be initialized by the startup code. **ADC077**: By default, notifications and hardware trigger capability shall be disabled after module initialization (if statically configured as active). **ADC107**: If development error detection is enabled the routine shall check that the ADC driver and hardware are not yet initialized and return error ADC_E_UNINIT (see Table 3) if they are.	
Caveats:	--	
Configuration:	See chapter 10.2.	

8.3.2 Adc_DeInit

Service name:	Adc_DeInit	
Syntax:	`void Adc_DeInit ( void )`	
Service ID [hex]:	0x01	
Sync/Async :	Synchronous	
Reentrancy:	Non re-entrant	
Parameters (in):	None	
Parameters (out):	None	
Return value:	None	
Description:	**ADC110**: This service shall return all ADC HW Units to a state comparable to their power on reset state. Values of registers which are not writeable are excluded. It's the responsibility of the hardware design that this state does not lead to undefined activities in the µC. **ADC111**: The service shall disable all used interrupts and notifications. **ADC112**: If development error detection is enabled and `Adc_DeInit` is called, while the ADC HW Unit is converting, the driver shall report `ADC_E_BUSY` (see Table 3) and the service will be left without any action. **ADC154**: If development error detection is enabled for the ADC Driver, calling any other ADC function except `ADC_Init` after `ADC_DeInit` being called will cause a development error `ADC_E_UNINIT` (see Table 3). The called service shall be left without any action.	
Caveats:	This service shall not be called while a conversion is ongoing.	
Configuration:	**ADC228**: This function shall be pre compile time configurable `On/Off` by the configuration parameter: `ADC_DEINIT_API` (see chapter 10.2).	

8.3.3 Adc_StartGroupConversion

Service name:	Adc_StartGroupConversion	
Syntax :	`void Adc_StartGroupConversion ( Adc_GroupType Group )`	
Service ID [hex]:	0x02	
Sync/Async :	Asynchronous	
Reentrancy:	re-entrant	
Parameters (in):	`Group`	Numeric ID of requested ADC Channel group.
Parameters (out):	None	
Return value:	None	
Description:	**ADC061**: This service shall start the conversion of all channels of the requested ADC Channel group. Depending on the group configuration single-shot or continuous conversion is started. **ADC060**: When configured and enabled, the group notification function shall be called, whenever a conversion of all channels of the requested group is completed. **ADC156**: Starting a group conversion will **NOT** automatically enable the notification mechanism for that group (this has to be done by a separate API call). **ADC125**: If development error detection is enabled for the ADC Driver, calling this function with a non-existing channel group ID will cause a development error `ADC_E_PARAM_GROUP` (see Table 3). The service shall then be left without any action. **ADC133**: If development error detection is enabled for the ADC Driver, calling this function on a group with trigger source configured as hardware will cause a development error `ADC_E_WRONG_TRIGG_SRC` (see Table 3). The service shall then be left without any action. **ADC276:** If development error detection is enabled for the ADC Driver, calling this function while another conversion is currently ongoing or a HW trigger is already enabled (priority mechanism disabled) or the conversion request queue is full (priority mechanism enabled) will cause a development error `ADC_E_BUSY` (see Table 3). The service shall then be left without any action. *Note: The priority mechanism is not related with anything else (HW/SW trigger, conversion mode, etc.) The result is that the higher priority conversion request is always converted no matter of which nature it is.* **ADC294**: If development error detection is enabled and `Adc_StartGroupConversion` is called prior to initializing the driver, the development error code `ADC_E_UNINIT` will be reported.	
Caveats:	**ADC146**: This service can only be used for groups configured in software trigger mode `ADC_GROUP_CONV_MODE` (see chapter 10.2).	
Configuration:	**ADC259** : This function shall be pre-compile time configurable `On/Off` by the configuration parameter `ADC_ENABLE_START_STOP_GROUP_API` (see chapter 10.2).	

8.3.4 Adc_StopGroupConversion

Service name:	Adc_StopGroupConversion	
Syntax :	`void Adc_StopGroupConversion ( Adc_GroupType Group )`	
Service ID [hex]:	0x03	
Sync/Async :	Synchronous	
Reentrancy:	re-entrant	
Parameters (in):	`Group`	Numeric ID of requested ADC Channel group.
Parameters (out):	None	
Return value:	None	
Description:	**ADC072**: This service shall stop the conversion of the requested ADC Channel group. **ADC303**: It should be possible to call this service both in single-shot and in continuous conversion mode. **ADC155**: Stopping a group conversion shall automatically disable group notification for the requested group. **ADC126**: If development error detection is enabled for the ADC Driver, calling this function with a non-existing group ID will cause a development error `ADC_E_PARAM_GROUP` (see Table 3). The service shall then be left without any action. **ADC164**: If development error detection is enabled for the ADC Driver, calling this function on a group with trigger source configured as hardware will cause a development error `ADC_E_WRONG_TRIGG_SRC` (see Table 3). The service shall then be left without any action. **ADC241**: If development error detection is enabled for the ADC Driver, calling this function while no conversion is currently ongoing will cause a development error `ADC_E_IDLE` (see Table 3). The service shall then be left without any action. **ADC295**: If development error detection is enabled and `Adc_StoptGroupConversion` is called prior to initializing the driver, the development error code `ADC_E_UNINIT` will be reported.	
Caveats:	**ADC283**: This service can only be used for groups configured in software trigger mode `ADC_GROUP_CONV_MODE` (see chapter 10.2).	
Configuration:	**ADC260** : This function shall be pre compile time configurable `On/Off` by the configuration parameter `ADC_ENABLE_START_STOP_GROUP_API` (see also chapter 10.2).	

8.3.5 Adc_ValueReadGroup

Service name:	Adc_ValueReadGroup	
Syntax:	`Adc_ValueGroupType *Adc_ValueReadGroup ( Adc_GroupType Group )`	
Service ID [hex]:	0x04	
Sync/Async :	Synchronous	
Reentrancy:	Re-entrant	
Parameters (in):	`Group`	Numeric ID of requested ADC channel group.
Parameters (out):	None	
Return value:	`Adc_ValueGroupType*`	Pointer to the latest conversion results available for the requested channel group.
Description:	**ADC075**: Service for reading the latest available conversion results of the requested channel group. **ADC113**: This service shall return the raw converted values without further scaling. The returned values shall be right-aligned. **ADC122**: If applicable, this service shall mask out all information or diagnostic bits provided by the conversion but not belonging to the conversion results themselves. **ADC152**: If development error detection is enabled for the ADC Driver, calling this function with a non-existing channel group ID will cause a development error `ADC_E_PARAM_GROUP` (see Table 3). **ADC141**: The read function shall provide atomic access by the use of either atomic instruction or the masking of interrupts using OS interrupt masking functions. **ADC296**: If development error detection is enabled and `Adc_ValueReadGroup` is called prior to initializing the driver, the development error code `ADC_E_UNINIT` will be reported.	
Caveats:	None	
Configuration:	None	

8.3.6 Adc_EnableHardwareTrigger

Service name:	Adc_EnableHardwareTrigger	
Syntax:	`void Adc_EnableHardwareTrigger ( Adc_GroupType Group )`	
Service ID [hex]:	0x05	
Sync/Async :	Synchronous	
Reentrancy:	re-entrant	
Parameters (in):	`Group`	Numeric ID of requested ADC Channel group.
Parameters (out):	None	
Return value:	None	
Description:	**ADC114**: This service shall enable the hardware trigger for the requested ADC Channel group. **ADC144**: The conversion of the requested group shall be executed, whenever a trigger event occurs. **ADC273**: The ADC Driver shall handle only one group conversion request per HW unit at the same time. It's up to the user to guarantee no concurrent conversion on the same HW unit (happening of different hardware triggers at the same time). **ADC306**: If development error detection is enabled for the ADC Driver, calling this function while another conversion is currently ongoing or a HW trigger is already enabled will cause a development error `ADC_E_BUSY` (see Table 3). The service shall then be left without any action. **ADC128**: If development error detection is enabled for the ADC Driver, calling this function with an invalid channel group ID will cause a development error `ADC_E_PARAM_GROUP` (see Table 3). The service shall then be left without any action. **ADC136**: If development error detection is enabled for the ADC Driver, calling this function for a group configured for software API trigger mode will cause a development error `ADC_E_WRONG_TRIGG_SRC` (see Table 3). The service shall then be left without any action. **ADC281:** If development error detection is enabled for the ADC Driver, calling this function for a group configured for continuous conversion mode will cause a development error `ADC_E_WRONG_CONV_MODE` (see Table 3). The service shall then be left without any action. **ADC297**: If development error detection is enabled and `Adc_EnableHardwareTrigger` is called prior to initializing the driver, the development error code `ADC_E_UNINIT` will be reported.	
Caveats:	**ADC120**: This service can only be used for groups configured in hardware trigger mode `ADC_GROUP_TRIGG_SRC` (see chapter 10.2).	
Configuration:	**ADC265**: This function shall be pre-compile time configurable `On/Off` by the configuration parameter `ADC_HW_TRIGGER_API` (see chapter 10.2).	

8.3.7 Adc_DisableHardwareTrigger

Service name:	Adc_DisableHardwareTrigger	
Syntax:	`void Adc_DisableHardwareTrigger ( Adc_GroupType Group )`	
Service ID [hex]:	0x06	
Sync/Async :	Synchronous	
Reentrancy:	re-entrant	
Parameters (in):	`Group`	Numeric ID of requested ADC Channel group.
Parameters (out):	None	
Return value:	None	
Description:	**ADC116**: This service shall disable the hardware trigger for the requested ADC Channel group. **ADC145**: If applicable, an ongoing conversion shall be aborted. **ADC157**: If enabled, this function shall also disable the notification mechanism for the requested group. **ADC129**: If development error detection is enabled for the ADC Driver, calling this function with a non-existing channel group ID will cause a development error `ADC_E_PARAM_GROUP` (see Table 3). The service shall then be left without any action. **ADC137**: If development error detection is enabled for the ADC Driver, calling this function for a group configured for software API trigger mode will cause a development error `ADC_E_WRONG_TRIGG_SRC` (see Table 3). The service shall then be left without any action. **ADC282:** If development error detection is enabled for the ADC Driver, calling this function for a group configured for continuous conversion mode will cause a development error `ADC_E_WRONG_CONV_MODE` (see Table 3). The service shall then be left without any action. **ADC304**: If development error detection is enabled for the ADC Driver, calling this function with a non enabled group will cause a development error `ADC_E_PARAM_GROUP` (see Table 3). The service shall then be left without any action. **ADC298**: If development error detection is enabled and `Adc_DisableHardwareTrigger` is called prior to initializing the driver, the development error code `ADC_E_UNINIT` will be reported.	
Caveats:	**ADC121**: This service can only be used for groups configured in hardware trigger mode `ADC_GROUP_TRIGG_SRC` (see chapter 10.2).	
Configuration:	**ADC266:** This function shall be pre-compile time configurable `On/Off` by the configuration parameter `ADC_HW_TRIGGER_API` (see chapter 10.2).	

8.3.8 Adc_EnableGroupNotification

Service name:	Adc_EnableGroupNotification	
Syntax :	`void Adc_EnableGroupNotification ( Adc_GroupType Group )`	
Service ID [hex]:	0x07	
Sync/Async :	Synchronous	
Reentrancy:	re-entrant	
Parameters (in):	`Group`	Numeric ID of requested ADC Channel group.
Parameters (out):	None	
Return value:	None	
Description:	**ADC057**: This service shall enable the notification mechanism for the requested ADC Channel group. **ADC130**: If development error detection is enabled for the ADC Driver, calling this function with a non-existing channel group ID will cause a development error `ADC_E_PARAM_GROUP` (see Table 3). The service shall then be left without any action. **ADC165**: If development error detection is enabled for the ADC Driver, calling this function when the group notification function pointer is NULL will cause a development error `ADC_E_NOTIF_CAPABILITY` (see Table 3). The service shall then be left without any action. **ADC299**: If development error detection is enabled and `Adc_EnableGroupNotification` is called prior to initializing the driver, the development error code `ADC_E_UNINIT` will be reported.	
Caveats:	None	
Configuration:	**ADC100:** This function shall be pre-compile time configurable `On/Off` by the configuration parameter `ADC_GRP_NOTIF_CAPABILITY` (see chapter 10.2).	

8.3.9 Adc_DisableGroupNotification

Service name:	Adc_DisableGroupNotification	
Syntax:	`void Adc_DisableGroupNotification ( Adc_GroupType Group )`	
Service ID [hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	re-entrant	
Parameters (in):	`Group`	Numeric ID of requested ADC Channel group.
Parameters (out):	None	
Return value:	None	
Description:	**ADC058**: This service shall disable the notification mechanism for the requested ADC Channel group. **ADC131**: If development error detection is enabled for the ADC Driver, calling this function with a non-existing channel group ID will cause a development error `ADC_E_PARAM_GROUP` (see Table 3). The service shall then be left without any action. **ADC166**: If development error detection is enabled for the ADC Driver, calling this function when the group notification function pointer is NULL will cause a development error `ADC_E_NOTIF_CAPABILITY` (see Table 3). The service shall then be left without any action. **ADC300**: If development error detection is enabled and `Adc_DisableGroupNotification` is called prior to initializing the driver, the development error code `ADC_E_UNINIT` will be reported.	
Caveats:	None	
Configuration:	**ADC101:** This function shall be pre-compile time configurable `On/Off` by the configuration parameter `ADC_GRP_NOTIF_CAPABILITY` (see chapter 10.2).	

8.3.10 Adc_GetGroupStatus

Service name:	Adc_GetGroupStatus	
Syntax:	`Adc_StatusType Adc_GetGroupStatus ( Adc_GroupType Group )`	
Service ID [hex]:	0x09	
Sync/Async :	Synchronous	
Reentrancy:	Re-entrant	
Parameters (in):	`Group`	Numeric ID of requested ADC Channel group.
Parameters (out):	None	
Return value:	`Adc_StatusType`	Conversion status for the requested group.
Description:	**ADC220**: This service shall return the conversion status of the requested ADC Channel group. **ADC221**: This service shall return `ADC_IDLE` if it is called before the conversion of the requested group has been started. **ADC222**: This service shall return `ADC_BUSY`: ▪ If it is called while the first conversion of the requested group is still ongoing (continuous conversion mode). ▪ Once trigger is enabled for group with HW trigger source. ▪ Once `Adc_StartGroupConversion` is called for group with SW trigger source **ADC224**: This service shall return `ADC_COMPLETED`: ▪ If it is called after the first conversion of the requested group has been finished in continuous conversion mode. ▪ Once conversion is completed in one shot mode. **ADC225**: If development error detection is enabled for the ADC Driver, calling this function with a non-existing group ID will cause a development error `ADC_E_PARAM_GROUP` (seeTable 3). The service shall then be left without any action. **ADC226**: : This function shall provide atomic access to the status data by the use of either atomic instructions or the masking of interrupts using OS interrupt masking functions. **ADC305**: To guarantee consistent retuned values, it is assumed that ADC group conversion is always started (or enabled in case of HW group) successfully by SW before status polling begins. **ADC307**: After `Adc_Init` all groups are in `ADC_IDLE` state. **ADC308**: Whenever `Adc_StopGroupConversion` or `Adc_DisableHardwareTrigger` is called successfully, the group status shall be set to `ADC_IDLE`. **ADC301**: If development error detection is enabled and `Adc_GetGroupStatus` is called prior to initializing the driver, the development error code `ADC_E_UNINIT` will be reported.	
Caveats:	None	
Configuration:	None	

8.3.11 Adc_GetStreamLastPointer

Service name:	Adc_GetStreamLastPointer	
Syntax:	`Adc_StreamNumSampleType Adc_GetStreamLastPointer ( Adc_GroupType Group Adc_PointerBufferType *SamplePtr )`	
Service ID [hex]:	0x0b	
Sync/Async :	Synchronous	
Reentrancy:	Re-entrant	
Parameters (in):	`Group`	Selected numeric group ID.
Parameters (out):	`SamplePtr`	Pointer to the last acquired value.
Return value:	`Adc_StreamNumSampleType`	Number of valid samples per channel.
Description:	**ADC214**: This service shall return the pointer to the last converted value. All values stored in the buffer are left without further scaling. **ADC215**: Asking for a group buffer pointer prior to starting the conversion of the group, will return a default value null. **ADC216**: Asking for a group buffer pointer while a conversion of the group is in progress will return null (data are available only at the end of conversion). **ADC217**: The buffer pointer is overwritten when the next conversion result on this group is available. **ADC218**: If development error detection is enabled for the ADC Driver, calling this function with a non-existing group ID will cause a development error (`ADC_E_PARAM_GROUP`, seeTable 3) and the desired functionality shall be left without any action. **ADC219**: It is up to the user to guarantee the consistency of the data that has been read by checking the return value `Adc_GetGroupStatus`. **ADC302**: If development error detection is enabled and `Adc_GetStreamLastPointer` is called prior to initializing the driver, the development error code `ADC_E_UNINIT` will be reported.	
Caveats:	None	
Configuration:	None	

8.3.12 Adc_GetVersionInfo

Service name:	Adc_GetVersionInfo
Syntax:	`void Adc_GetVersionInfo ( Std_VersionInfoType *versioninfo )`
Service ID [hex]:	0x0a
Sync/Async :	Synchronous
Reentrancy:	non reentrant
Parameters (in):	`none`
Parameters (out):	`versioninfo` Pointer to where to store the version information of this module.
Return value:	none
Description:	**ADC236**: This service returns the version information of this module. The version information includes: - Module Id. - Vendor Id. - Vendor specific version numbers (**BSW00407**). Hint: If source code for caller and called of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.
Caveats:	--
Configuration:	**ADC237**: This function shall be pre-compile time configurable `On/Off` by the configuration parameter `ADC_VERSION_INFO_API (see chapter 10.2).`

8.4 Call-back Notifications

Since the ADC Driver is a module on the lowest architectural layer it doesn't provide any call-back functions for lower layer modules.

8.5 Scheduled functions

None

8.6 Expected Interfaces

In this chapter all interfaces required from other modules are listed.

8.6.1 Mandatory Interfaces

This chapter defines all interfaces which are required to fulfill a core functionality of the module.

API function	*Module*	*Description*	*Configuration parameter (description see chapter 10)*
Dem_ReportErrorStatus	Dem	Production error status.	Implementation specific.

8.6.2 Optional Interfaces

This chapter defines all interfaces which are required to fulfill an optional functionality of the module.

API function	*Module*	*Description*	*Configuration parameter (description see chapter 10)*
Det_ReportError	Det	Development error notification.	ADC_DEV_ERROR_DETECT

8.6.3 Configurable interfaces

In this chapter all interfaces are listed where the target function could be configured. The target function is usually a call-back function. The names of this kind of interfaces are not fixed because they are configurable.

ADC078: The ISR´s, providing the “conversion completed events”, shall be responsible for resetting the interrupt flags (if needed by hardware) and calling the associated notification function.

ADC079: The notification functions shall be configurable as function pointers within the initialization data structure `Adc_ConfigType`.

ADC080: If for a notification function the NULL pointer is configured, no call-back shall be executed.

ADC081: For each group an own notification function shall be provided.

ADC153: The notification functions are running in interrupt context. It’s the responsibility of the user to keep the code of these functions reasonably short.

ADC270: The notification functions shall have no parameters and no return value.

Service name:	IoHwAb_Adc_Notification_<GroupID>	

Syntax:	`void IoHwAb_Adc_Notification_<GroupID>`
	`(`
	`  void`
	`)`	

Reentrancy:	Re-entrancy of this API call depends on the user's code.	
Parameters (in):	None	
Parameters (out):	None	
Return value:	None	
Description:	**ADC082**: Prototype for the notification call-back function. It has to be implemented by the user. **ADC104**: The ADC Driver shall support an individual notification per ADC Channel group (if capability is configured) that is called whenever the conversion for all channels of that group is completed. **ADC083**: When the notification mechanism is disabled, no notification shall be sent. When the notifications are re-enabled again, the user will not be notified of events that occurred while notifications have been disabled. **ADC084**: To avoid parameter values and to improve runtime efficiency, for every group, a particular notification call-back has to be declared for all notifications, not configured as NULL pointer. **ADC085**: The call-back notifications shall be configurable as pointers to user defined functions within the configuration structure.	
Caveats:	None	
Configuration:	For all available channel groups, call-back functions have to be declared during the configuration phase of the module.	

9 Sequence diagrams

9.1 Initialization of the ADC Driver

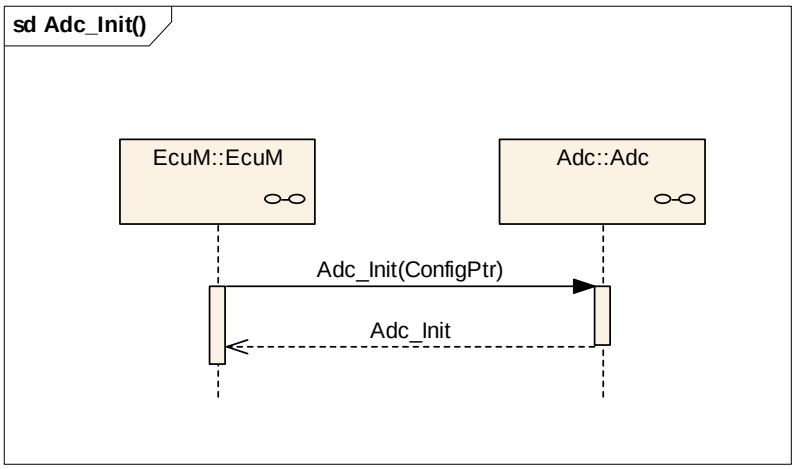

Figure 3: Initialization of the ADC Driver

9.2 De-Initialization of the ADC Driver

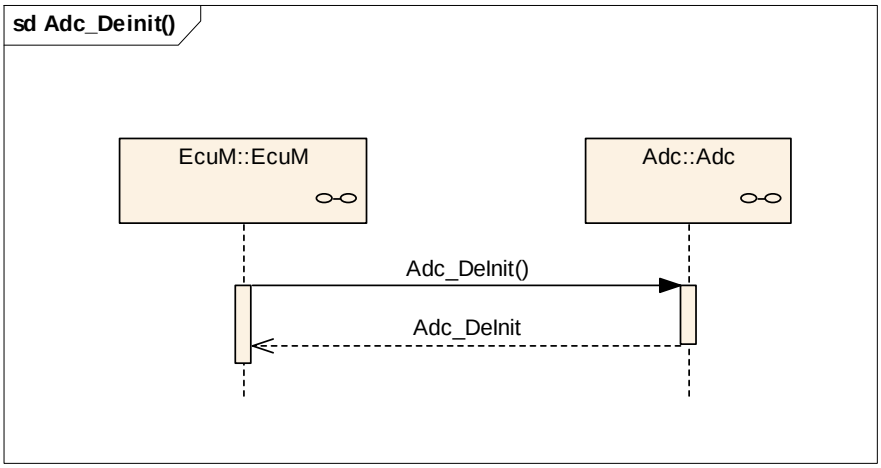

Figure 4: De-Initialization of the ADC Driver

9.3 Software triggered One-Shot conversion without notification

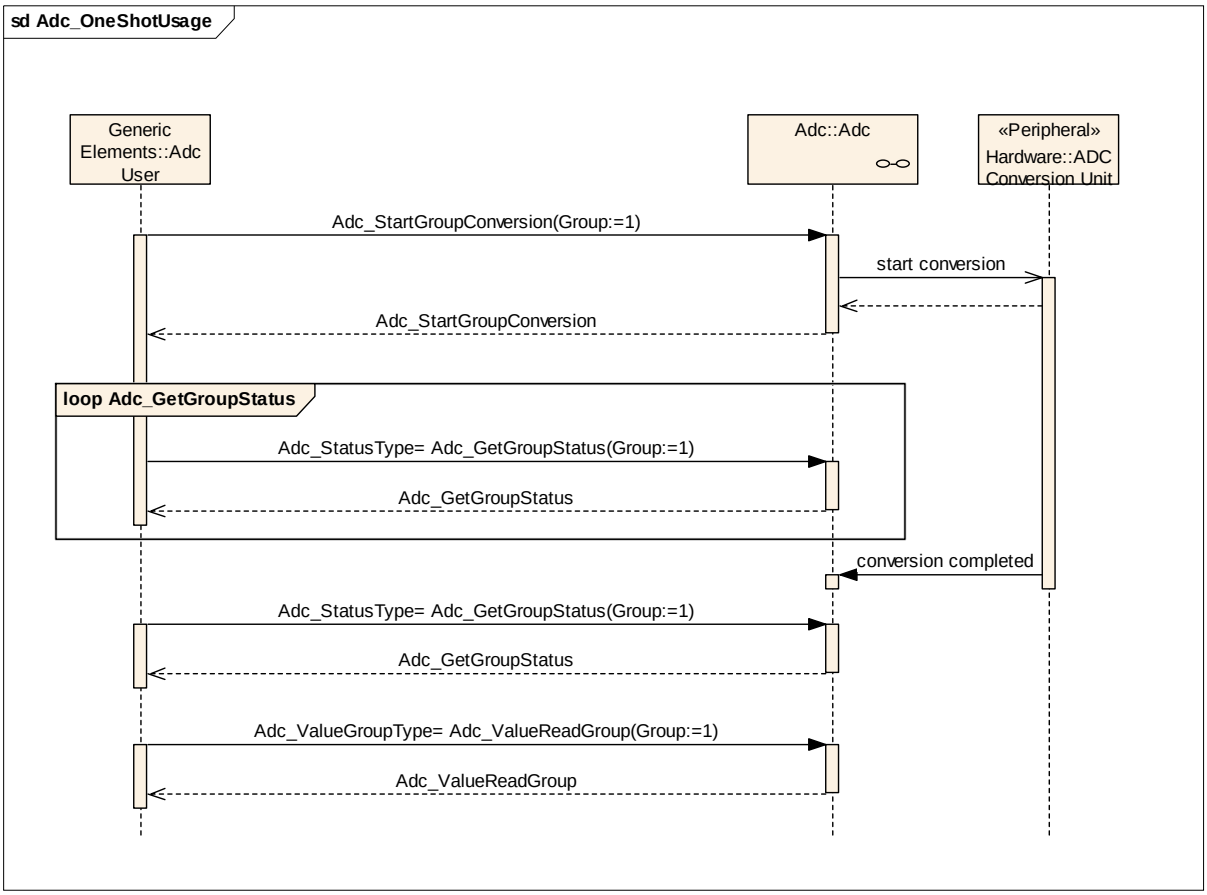

Figure 5: Software triggered One-Shot conversion of without notification

9.4 Software triggered continuous conversion with notification

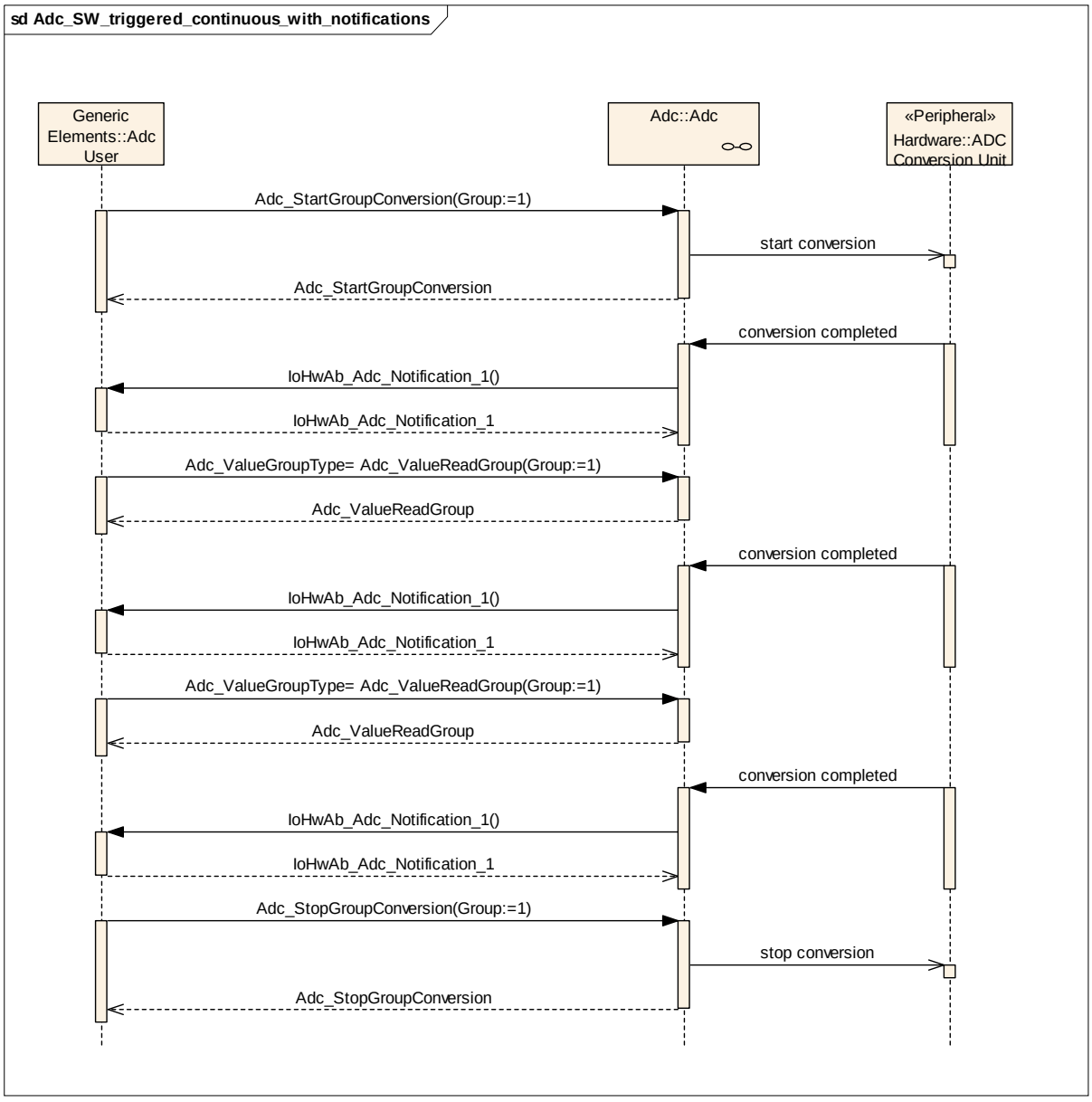

Figure 6: Software triggered continuous conversion with notification

9.5 Hardware triggered One-Shot conversion with notification

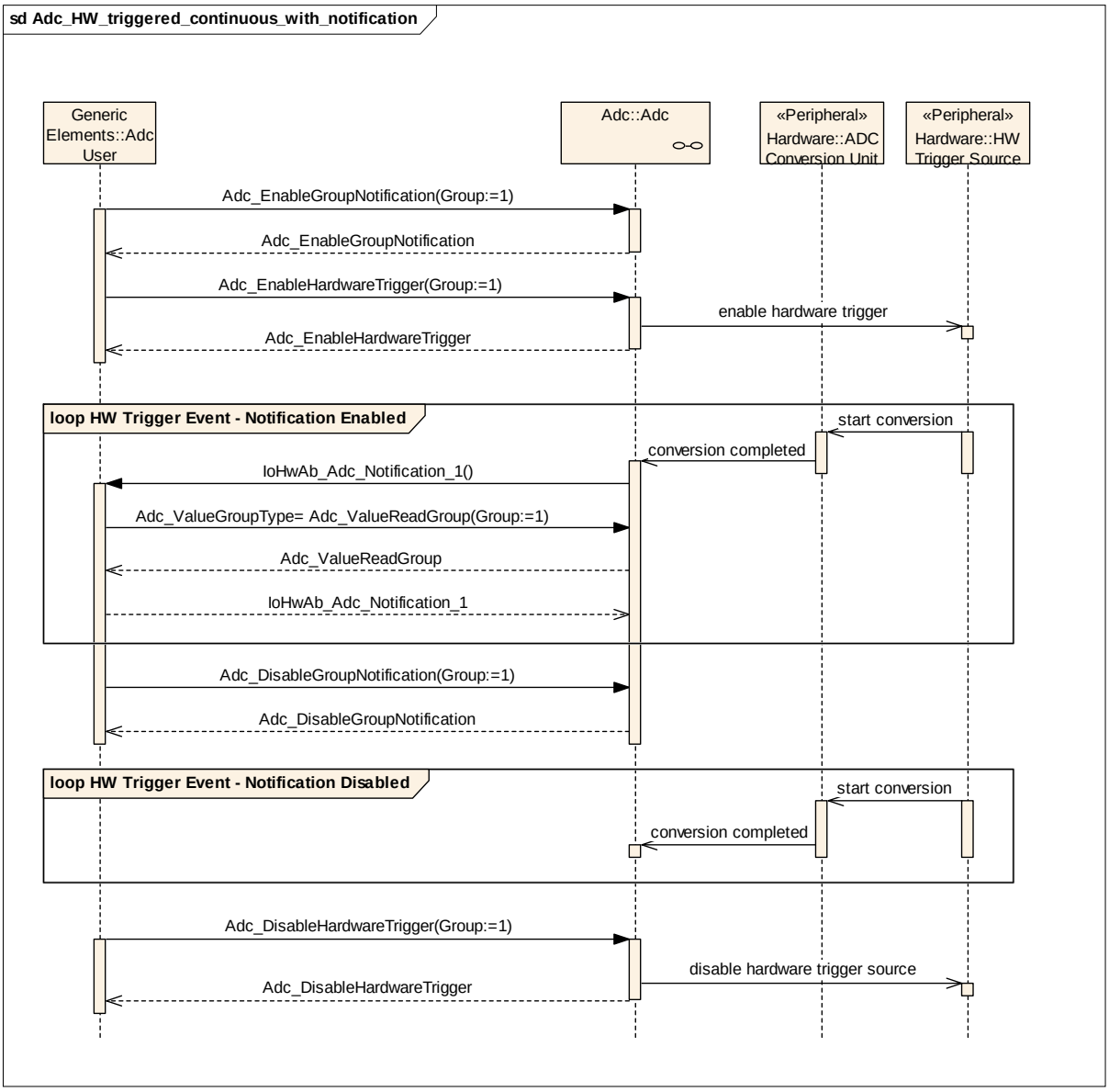

Figure 7: Hardware triggered One-Shot conversion with notification

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module ADC Driver.

Chapter 10.3 specifies published information of the module ADC Driver.

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture
- AUTOSAR ECU Configuration Specification
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.1.3 Specification template for configuration parameters

The following tables consist of three sections:
- the general section
- the configuration parameter section
- the section of included/referenced containers

Pre-compile time - specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* – the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* – the configuration parameter shall be of configuration class *Post Build* and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

10.2 Configuration and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

10.2.1 Variants

Variant PC: This variant is limited to pre-compile configuration parameters only.

Variant PB: This variant allows a mix of pre-compile time and post-build multiple selectable configuration parameters.

ADC275: The initialization function of this module shall always have a pointer as a parameter, even though for Variant PC no configuration set shall be given. Instead a NULL pointer shall be passed to the initialization function.

10.2.2 AdcDriverGeneralConfiguration

SWS Item	**ADC027**
Container Name	AdcDriverGeneralConfiguration
Description	This container contains the general configuration (parameters) of the ADC Driver software module.
Configuration Parameters	

Name	ADC_DEV_ERROR_DETECT		
Description	Switches the Development Error Detection and Notification ON or OFF.		
Type	#define		
Unit	--		
Range	STD_ON	Enabled	
	STD_OFF	Disabled	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	ADC_VERSION_INFO_API		
Description	Adds / removes the service Adc_GetVersionInfo() from the code.		
Type	#define		
Unit	--		
Range	STD_ON	Adc_GetVersionInfo() can be used.	
	STD_OFF	Adc_GetVersionInfo() can not be used.	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	ADC_DEINIT_API		
Description	Adds / removes the service Adc_DeInit() from the code.		
Type	#define		
Unit	--		
Range	STD_ON	Adc_DeInit() can be used.	
	STD_OFF	Adc_DeInit() can not be used.	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	ADC_ENABLE_START_STOP_GROUP_API		
Description	Adds / removes the services Adc_StartGroupConversion() and Adc_StopGroupConversion() from the code.		
Type	#define		
Unit	--		
Range	STD_ON	Adc_StartGroupConversion ()and Adc_StopGroupConversion() can be used.	
	STD_OFF	Adc_StartGroupConversion ()and Adc_StopGroupConversion() cannot be used.	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	ADC_HW_TRIGGER_API		
Description	Adds / removes the services Adc_EnableHardwareTrigger() and Adc_DisableHardwareTrigger() from the code.		
Type	#define		
Unit	--		
Range	STD_ON	Adc_EnableHardwareTrigger() and Adc_DisableHardwareTrigger() can be used.	
	STD_OFF	Adc_EnableHardwareTrigger() and Adc_DisableHardwareTrigger() cannot be used.	
Configuration Class	***Pre-compile***	X	all Variants.
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	ADC_GRP_NOTIF_CAPABILITY		
Description	**ADC105**: Determines, if the group notification mechanism (the functions to enable and disable the notifications) is available at runtime.		
Type	#define		
Unit	--		
Range	STD_ON	Enabled	
	STD_OFF	Disabled	
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Name	ADC_GRP_PRIORITY_IMP_LEVEL
Description	Determines the number of available priority levels (255 means a full support of the priority mechanism support, 0 means that the priority mechanism is not available at all, any other value means a partial support of the priority mechanism).

Type	`#define`		
Unit	`uint8`		
Range	`0..255`		
Configuration Class	***Pre-compile***	X	all Variants
	Link time	--	--
	Post Build	--	--
Scope	Module		
Dependency	None		

Included Containers			
Container Name	***Multiplicity***	***Scope***	***Dependency***
AdcHWUnitConfiguration	1..*		

10.2.3 AdcHWUnitConfiguration

SWS Item	**ADC242**
Container Name	AdcHWUnitConfiguration
Description	This container contains the Driver configuration (parameters) depending on grouping of channels The organization of this data structure could contain dependencies to the microcontroller so this is left up to the implementer and its location is left up to the configuration. **ADC138**: The ADC Driver shall support one or several ADC HW units of the same type. The selection of ADC HW unit shall be done by this configuration container.
Configuration Parameters	

Name	`ADC_HWUNIT_ID`		
Description	Numeric ID of the HW Unit. This symbolic name allows accessing Hw Unit data.		
Type	`Adc_HwUnitType`		
Unit	`uint8`		
Range	`0..255`		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Name	`ADC_CLK_SRC`		
Description	**ADC087**: The ADC module specific clock input for the conversion unit can be configured to select different clock sources if provided by hardware.		
Type	`Adc_ClockSourceType`		
Unit	`--`		
Range	`--`		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Name	ADC_PRESCALE		
Description	**ADC088**: Optional ADC module specific clock prescale factor, if supported by hardware.		
Type	Adc_PrescaleType		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Included Containers			
Container Name	***Multiplicity***	***Scope***	***Dependency***
AdcGroupConfiguration	1..*	--	--

10.2.4 AdcGroupConfiguration

SWS Item	**ADC028**
Container Name	AdcGroupConfiguration
Description	This container contains the Group configuration (parameters).
Configuration Parameters	

Name	ADC_GROUP_ID		
Description	Numeric ID of the group. This parameter is the symbolic name to be used on the API. This symbolic name allows accessing Channel Group data.		
Type	Adc_GroupType		
Unit	uint8		
Range	0..255		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Name	ADC_GROUP_PRIORITY		
Description	**ADC287 :** Priority level of the group.		
Type	Adc_GroupPriorityType		
Unit	--		
Range	0..255		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	ADC_GRP_PRIORITY_IMP_LEVEL		

Name	ADC_GROUP_DEFINITION		
Description	**ADC014**: Assignment of channels to a channel group. **ADC098**: All channels of a group share the same group configuration (channels can have different channel specific configurations).		
Type	Adc_GroupDefType		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Name	ADC_GROUP_TRIGG_SRC		
Description	Type of source event that starts a group conversion.		
Type	Adc_TriggerSrcType		
Unit	--		
Range	ADC_TRIGG_SRC_SW	**ADC094**: Group is triggered by a software API call.	
	ADC_TRIGG_SRC_HW	**ADC095**: Group is triggered by a hardware event.	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	ADC_GROUP_CONV_MODE: Trigger source HW is not available for continuous conversion mode.		

Name	ADC_GROUP_CONV_MODE		
Description	Type of conversion mode supported by the driver.		
Type	Adc_GroupConvModeType		
Unit	--		
Range	ADC_CONV_MODE_ONESHOT	**ADC001**: The conversion of an ADC channel group is performed once after a trigger.	
	ADC_CONV_MODE_CONTINUOUS	**ADC002**: Conversions of an ADC channel group are performed continuously after a software API call (start). The conversions itself are running automatically (no additional software or hardware trigger needed).	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	ADC_GROUP_TRIGG_SRC: Continuous conversion mode only available for software triggered groups.		

Name	ADC_GROUP_ACCESS_MODE		
Description	**ADC317**: Type of access mode to group conversion results.		
Type	Adc_GroupAccessModeType		
Unit	--		
Range	ADC_ACCESS_MODE_SINGLE	Single value access mode.	
	ADC_ACCESS_MODE_STREAMING	Streaming access mode.	
Configuration Class	***Pre-compile***	X	Variant PC

	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Name	ADC_STREAMING_BUFFER_POINTER		
Description	**ADC291**: Pointer to data buffer (destination for conversion results). Note: in streaming access mode the Adc_ValueGroupType buffer is made of m*n elements, where n is the number of channels belonging to the group, m the number of samples acquired per channel (i.e. ADC_STREAMING_NUM_SAMPLES). ADC_STREAMING_BUFFER_POINTER points to one value of this buffer.		
Type	Adc_PointerBufferType		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	ADC_GROUP_ACCESS_MODE: Valid only for streaming access mode.		

Name	ADC_STREAMING_NUM_SAMPLES		
Description	**ADC292**: Number of ADC values to be acquired per channel in streaming access mode. Note: in normal access mode this parameter assumes value 1, since only one sample per channel is processed.		
Type	Adc_StreamNumSampleType		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	ADC_GROUP_ACCESS_MODE: Valid only for streaming access mode. In normal access mode this parameter assumes value 1, since only one sample per channel is processed.		

Name	ADC_STREAMING_BUFFER_MODE		
Description	**ADC316**: Configure streaming buffer as "linear buffer" (i.e. the ADC Driver stops the conversion as soon as the stream buffer is full) or as "ring buffer" (wraps around if the end of the stream buffer is reached).		
Type	Adc_StreamBufferModeType		
Unit	--		
Range	ADC_STREAM_BUFFER_LINEAR	The ADC Driver stops the conversion as soon as the stream buffer is full (number of samples reached).	
	ADC_STREAM_BUFFER_CIRCULAR	The ADC Driver continue the conversion even if the stream buffer is full (number of samples reached) by wrapping around the stream buffer itself.	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	ADC_GROUP_ACCESS_MODE: Valid only for streaming access mode.		

Name	ADC_HW_TRIG_SIGNAL		
Description	Configures on which edge of the hardware trigger signal the driver should reach, i.e. start the conversion (only if supported by the ADC hardware).		
Type	Adc_HwTriggerSignalType		
Unit	--		
Range	ADC_HW_TRIG_RISING_EDGE	React on the rising edge of the hardware trigger signal (only if supported by the ADC hardware).	
	ADC_HW_TRIG_FALLING_EDGE	React on the falling edge of the hardware trigger signal (only if supported by the ADC hardware).	
	ADC_HW_TRIG_BOTH_EDGES	React on both edges of the hardware trigger signal (only if supported by the ADC hardware).	
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	ADC_TRIGG_SRC_HW: Valid only if the group is configured to be triggered by a hardware event.		

Name	ADC_HW_TRIG_TIMER		
Description	Reload value of the ADC module embedded timer (only if supported by the ADC hardware).		
Type	Adc_HwTriggerTimerType		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	ADC_TRIGG_SRC_HW: Valid only if the group is configured to be triggered by a hardware event.		

Name	ADC_NOTIFICATION		
Description	Call-back function for each group.		
Type	--		
Unit	--		
Range	Function pointer		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	This parameter is only relevant, if notification capability is configured available by ADC_GRP_NOTIF_CAPABILITY.		

Included Containers			
Container Name	***Multiplicity***	***Scope***	***Dependency***
AdcChannelConfiguration	1..*	--	--

10.2.5 AdcChannelConfiguration

SWS Item	**ADC268**
Container Name	AdcChannelConfiguration
Description	This container contains the channel configuration (parameters) depending on the hardware capability. The organization of this data structure could contain dependencies to the microcontroller so this is left up to the implementer and its location is left up to the configuration.
Configuration Parameters	

Name	ADC_CHANNEL_ID		
Description	Numeric ID of the channel. This parameter is the symbolic name to be used on the API. This symbolic name allows accessing Channel data.		
Type	Adc_ChannelType		
Unit	--		
Range	0..255		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Name	ADC_CHANNEL_CONV_TIME		
Description	**ADC011**: Configuration of conversion time, i.e. the time during which the analogue value is converted into digital representation, (in clock cycles) for each channel, if supported by hardware.		
Type	Adc_ConversionTimeType		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Name	ADC_CHANNEL_SAMP_TIME		
Description	**ADC290**: : Configuration of sampling time, i.e. the time during which the value is sampled, (in clock cycles) for each channel, if supported by hardware.		
Type	Adc_SamplingTimeType		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Name	ADC_CHANNEL_RESOLUTION		
Description	**ADC019**: Channel resolution in bits.		
Type	Adc_ResolutionType		
Unit	--		
Range	0..255		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	ADC_MAX_CHANNEL_RESOLUTION: The actual resolution has to be less or equal than the maximum resolution.		

Name	ADC_CHANNEL_REF_VOLTSRC_LOW		
Description	**ADC023**: Lower reference voltage source for each channel.		
Type	Adc_VoltageSourceType		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Name	ADC_CHANNEL_REF_VOLTSRC_HIGH		
Description	**ADC089**: Upper reference voltage source for each channel.		
Type	Adc_VoltageSourceType		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	X	Variant PC
	Link time	--	--
	Post Build	M	Variant PB
Scope	Module		
Dependency	None		

Included Containers			
Container Name	***Multiplicity***	***Scope***	***Dependency***
None	--	--	--

10.3 Published information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

SWS Item	ADC030	
Information elements		
Information element name	**Type / Range**	**Information element description**
ADC_VENDOR_ID	uint16 / --	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list.
ADC_MODULE_ID	uint8 / --	Module ID of this module from Module List.
ADC_AR_MAJOR_VERSION	uint8 / --	Major version number of AUTOSAR specification where the appropriate implementation is based on.
ADC_AR_MINOR_VERSION	uint8 / --	Minor version number of AUTOSAR specification where the appropriate implementation is based on.
ADC_AR_PATCH_VERSION	uint8 / --	Patch level version number of AUTOSAR specification where the appropriate implementation is based on.
ADC_SW_MAJOR_VERSION	uint8 / --	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
ADC_SW_MINOR_VERSION	uint8 / --	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
ADC_SW_PATCH_VERSION	uint8 / --	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.
ADC_MAX_CHANNEL_RESOLUTION	uint8 / --	Maximum channel resolution in bits. (Does not specify accuracy).
ADC_CHANNEL_VALUESIGNED	Boolean / TRUE / FALSE	Information whether the result value of the ADC driver has sign information or not. If the result shall be interpreted as signed value it shall apply to “C”-language rules.
ADC_GROUP_FIRST_CHANNEL_FIXED	Boolean / TRUE / FALSE	Information whether the first channel of an ADC Channel group can be configured (FALSE) or is fixed (TRUE) to a value determined by the ADC HW Unit.

10.4 Configuration of symbolic names

ADC099: The symbolic names of ADC channels and ADC channel groups for use by the upper layer shall be defined by the configurator. They are to be defined in the modules configuration header file.

11 Changes to Release 1

11.1 Deleted SWS Items

SWS Item	*Rationale*
ADC029	According new template

11.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
ADC070	ADC233, ADC234, ADC235	Required for new SWS template.
ADC148	ADC246, ADC247, ADC248, ADC249, ADC250	New formulation for one-time writeable registers.
ADC076	ADC244	Standardization of default value.

11.3 Changed SWS Items

SWS Item	*Rationale*
ADC001	Added stop functionality.
ADC030	Take in account the new template for published information and BSW003.
ADC086, ADC107, ADC112, ADC115, ADC118, ADC125, ADC126, ADC128, ADC129, ADC130, ADC131, ADC132, ADC133, ADC134, ADC135, ADC136, ADC137, ADC152, ADC154	Modified requirements to align statement for DET.
ADC112	Clarified behavior of Adc_Deinit function when call occurs while conversion is running.
ADC108	Stop conversion depending by HW capability.
ADC087	According to the new template.
ADC090	Removed “Basic” definition.
ADC091	Removed “Basic” definition.
ADC092	Removed “Basic” definition.
ADC093	Removed “Basic” definition.
ADC027	According to the new template.
ADC028	According to the new template.
ADC138	New Driver structure and functionalities.
ADC139	Split in different requirement IDs (see ADC253).
ADC140	New Driver structure and functionalities.
ADC067	According to the new template.
ADC150	Standardization of start-up code meaning.
ADC077	New Driver structure and functionalities.
ADC110	New formulation for shared registers.
ADC111	Clarification of functional perimeter of the ADC_DeInit function.
ADC127	New Driver structure and functionalities.

ADC124	New Driver structure and functionalities.
ADC113	Standardization of result value.
ADC114	New Driver structure and functionalities.
ADC144	New Driver structure and functionalities.
ADC142	New Driver structure and functionalities (Use of Adc_EnableHardwareTrigger to re-arm an internal timer).
ADC116	New Driver structure and functionalities.
ADC145	New Driver structure and functionalities.
ADC023	#4619 (Rfc#4198).
ADC089	#4619 (Rfc#4198).
ADC095	Notification capability corrected in hardware capability.
ADC096	Notification capability corrected in hardware capability.
ADC098	#4200: clarified definition of group configuration.

11.4 Added SWS Items

SWS Item	*Rationale*
ADC155	Added to substitute wrong duplicated ID in SWS ADC 1.0 (**ADC124**).
ADC156	Added to substitute wrong duplicated ID in SWS ADC 1.0 (**ADC125**).
ADC157	Added to substitute wrong duplicated ID in SWS ADC 1.0 (**ADC126**).
ADC158	Added to substitute wrong duplicated ID in SWS ADC 1.0 (**ADC128**).
ADC159	Added to substitute wrong duplicated ID in SWS ADC 1.0 (**ADC129**).
ADC160	Added to substitute wrong duplicated ID in SWS ADC 1.0 (**ADC130**).
ADC161	Added to substitute wrong duplicated ID in SWS ADC 1.0 (**ADC131**).
ADC162	Added to substitute wrong duplicated ID in SWS ADC 1.0 (**ADC132**).
ADC163	Added to substitute wrong duplicated ID in SWS ADC 1.0 (**ADC138**).
ADC164	#4191.
ADC165	#4194.
ADC166	#4194.
ADC167	New Driver structure and functionalities.
ADC168	#4192.
ADC169	New Driver structure and functionalities.
ADC174	New Driver structure and functionalities.
ADC175	New DET error.
ADC176	New DET error.
ADC177	New Driver structure and functionalities.
ADC179	New Driver structure and functionalities.
ADC180	New Driver structure and functionalities.
ADC181	New Driver structure and functionalities.
ADC182	New Driver structure and functionalities.
ADC183	New Driver structure and functionalities.
ADC184	New DET error.
ADC185	New DET error.
ADC186	New Driver structure and functionalities.
ADC187	New Driver structure and functionalities.
ADC188	New Driver structure and functionalities.
ADC189	New Driver structure and functionalities.
ADC190	New Driver structure and functionalities.
ADC191	New Driver structure and functionalities.
ADC192	New Driver structure and functionalities.
ADC193	New DET error.
ADC194	New Driver structure and functionalities.
ADC195	New Driver structure and functionalities.
ADC196	New Driver structure and functionalities.
ADC197	New Driver structure and functionalities.
ADC198	New Driver structure and functionalities.

ADC199	New Driver structure and functionalities.
ADC200	New Driver structure and functionalities.
ADC201	New Driver structure and functionalities.
ADC202	New Driver structure and functionalities.
ADC203	New Driver structure and functionalities.
ADC204	New Driver structure and functionalities.
ADC205	New Driver structure and functionalities.
ADC206	New Driver structure and functionalities.
ADC207	New Driver structure and functionalities.
ADC208	New DET error.
ADC209	New DET error.
ADC210	New Driver structure and functionalities.
ADC211	New Driver structure and functionalities.
ADC212	New DET error.
ADC213	New Driver structure and functionalities.
ADC214	New Driver structure and functionalities.
ADC215	New Driver structure and functionalities.
ADC216	New Driver structure and functionalities.
ADC217	New Driver structure and functionalities.
ADC218	New DET error.
ADC219	New Driver structure and functionalities.
ADC220	New Driver structure and functionalities.
ADC221	New Driver structure and functionalities.
ADC222	New Driver structure and functionalities.
ADC223	New Driver structure and functionalities.
ADC224	New Driver structure and functionalities.
ADC225	New DET error.
ADC226	New Driver structure and functionalities.
ADC228	Required for new SWS template (API pre compile time configurable On/Off).
ADC229, ADC230, ADC233, ADC234, ADC235, ADC236, ADC237, ADC238, ADC239, ADC240	Required for new SWS template.
ADC241	Missed DET error.
ADC242	Required for new SWS template.
ADC243	Configuration check.
ADC244	Missing requirements: description already present but not tagged as requirement IDs.
ADC246, ADC247, ADC248, ADC249, ADC250	Modified according **BSW12461**.
ADC251, ADC252, ADC253, ADC254, ADC255, ADC256, ADC257, ADC258,	Missing requirements: description already present but not tagged as requirement IDs.
ADC259, ADC260, ADC261, ADC262, ADC263, ADC264, ADC265, ADC266	Required for new SWS template (API pre compile time configurable On/Off).
ADC267, ADC268, ADC269, ADC270, ADC271	Missing requirements: description already present but not tagged as requirement IDs.
ADC272	Constrains on Adc_EnableHardwareTrigger function.
ADC273	Constrains on Adc_EnableHardwareTrigger function.
ADC274	Handling of multiple HW trigger for gated conversion.

ADC275	Added (SPAL decision, 41st meeting, minutes day2, issue 5).

12 Changes to Release 2.0.0

12.1 Deleted SWS Items

SWS Item	*Rationale*
ADC093	Superfluous specification element.
ADC163	Superfluous specification element.
ADC139	Superfluous specification element.
ADC167	On demand conversion not supported.
ADC256	On demand conversion not supported.
ADC257	Gated continuous conversion not supported.
ADC177	Gated continuous conversion not supported.
ADC132	NULL-pointer check removed from Adc_Init function.
ADC168	Check should be done offline by configuration tool (too much overhead).
ADC243	On demand conversion not supported.
ADC086	Check should be done offline by configuration tool (too much overhead).
ADC127	No concurrency supported, no re-entrance capabilities needed.
ADC158	No concurrency supported, no re-entrance capabilities needed.
ADC106	Superfluous specification element.
ADC244	“out of range” can’t be supported by an ADC Driver.
ADC142	Superfluous and confusing.
ADC272	No concurrency supported, no check needed.
ADC159	No concurrency supported, no re-entrance capabilities needed.
ADC143	Superfluous specification element.
ADC134	Superfluous check (functionality shall be configured out).
ADC161	No concurrency supported, no re-entrance capabilities needed.
ADC135	Superfluous check (functionality shall be configured out).
ADC162	No concurrency supported, no re-entrance capabilities needed.
ADC223	Adc_GroupStatusType reworked, specification element now obsolete.
ADC119	Superfluous (after reformulation of the item).
ADC096	No distinction between internal and external HW trigger source.
ADC195	On demand conversion not supported.
ADC196	Gated continuous conversion not supported.
ADC109	Integrated with ADC110 as requested in bugzilla issue #11903.

12.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
ADC115	**ADC281**	Similar content, different formulation.
ADC118	**ADC282**	Similar content, different formulation.

12.3 Changed SWS Items

SWS Item	*Rationale*
ADC090	Split into two requirements (added **ADC279** for second part).
ADC251	On demand and gated continuous conversion deleted, descriptions for one-shot and continuous conversion reformulated.
ADC252	Streaming result access mode deleted.
ADC253	Reformulated (conversion trigger sources).
ADC140	Reformulated (data consistency).
ADC254	Reformulated.

ADC255	Reformulated (no concurrency).
ADC124	Reformulated according to BSW004 (SRS General).
ADC065	Deleted second part (contained in ADC233).
ADC108	Reformulated (in the hope to make it more clear).
ADC241	Error code changed to make it distinguishable from other errors.
ADC120	Reformulated (trigger source).
ADC145	Reformulated (for all supported conversion modes).
ADC157	Reformulated to adapt to ADC155.
ADC121	Reformulated (trigger source).
ADC275	Reformulated /after BSW00414 had been reformulated).
ADC014, ADC098	Group definition configuration changed.

12.4 Added SWS Items

SWS Item	*Rationale*
ADC276	Added for stripped down SWS version.
ADC277	Added for stripped down SWS version.
ADC278	Added for stripped down SWS version.
ADC279	Added for stripped down SWS version.
ADC280	Added for stripped down SWS version.
ADC283	Added (analogue to ADC146).
ADC284, ADC285	Added (replacement for former group definition configuration).
ADC286	Added for stripped down SWS version.

13 Changes to Release 2.0.1

13.1 Deleted SWS Items

SWS Item	*Rationale*
ADC278	A corresponding requirement doesn't exist.
ADC108	Superfluous specification element.
ADC284	No more needed: changed the way to assign channels are to groups.
ADC285	No more needed: changed the way to assign channels are to groups.
ADC109	Integrated with ADC110 as requested in bugzilla issue #11903.
ADC107	Substituted by a correspondent requirement in each other API function.
ADC150, ADC151, ADC147	Redundant to **ADC246**, **ADC247**, **ADC248**, **ADC249**, **ADC250**
ADC213	The relevant information is given in ADC214.

13.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*

13.3 Changed SWS Items

SWS Item	*Rationale*
ADC252	Streaming result access mode added.
ADC075	Adapted to channel group concept.
ADC113	Adapted to channel group concept.
ADC122	Adapted to channel group concept.
ADC027	Added parameter `ADC_GRP_PRIORITY_INP_LEVEL`.
ADC011	Slightly reformulated.
ADC061	Slightly reformulated.
ADC276	Slightly reformulated to cover some Bugzilla entries..
ADC222	Slightly reformulated to cover some Bugzilla entries.
ADC224	Slightly reformulated to cover some Bugzilla entries.

13.4 Added SWS Items

SWS Item	*Rationale*
ADC288	Added for channel group priority mechanism.
ADC289	Added for channel group priority mechanism.
ADC214	Added for new service Adc_GetStreamLastPointer.
ADC215	Added for new service Adc_GetStreamLastPointer.
ADC216	Added for new service Adc_GetStreamLastPointer.
ADC217	Added for new service Adc_GetStreamLastPointer.
ADC218	Added for new service Adc_GetStreamLastPointer.
ADC219	Added for new service Adc_GetStreamLastPointer.
ADC287	Added for channel group priority mechanism.
ADC291	Added for streaming access mode.

ADC292	Added for streaming access mode.
ADC290	Added sampling time parameter.
ADC154, ADC294, ADC295, ADC296, ADC297, ADC298, ADC299, ADC300, ADC301, ADC302	Substitute removed requirement ADC107
ADC303	Added to API Adc_StopGroupConversion.
ADC304	Added to API Adc_DisableHardwareTrigger.
ADC305	Added to API GetGroupStatus
ADC306	Added to API Adc_EnableHardwareTrigger to cover some Bugzilla entries.
ADC307	Added to API Adc_GetGroupStatus to cover some Bugzilla entries.
ADC308	Added to API Adc_GetGroupStatus to cover some Bugzilla entries.
ADC309	Added for channel group priority mechanism.
ADC310	Added for channel group priority mechanism.
ADC311	Added for channel group priority mechanism.
ADC312	Added for channel group priority mechanism.
ADC314	Added for channel group priority mechanism.
ADC315	Added for channel group priority mechanism.
ADC316	Added for streaming access mode.
ADC317	Added for streaming access mode.
ADC318	Added for specifying the return value structure.
ADC319	Added for specifying the return value structure.
ADC320	Added for specifying the return value structure.