Document Title	Requirements on ECU Configuration
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	85

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R20-11

Document Change History

Date	Release	Changed by	Description
2020-11-30	R20-11	AUTOSAR Release Management	• No content changes
2019-11-28	R19-11	AUTOSAR Release Management	• Changed Document Identification No to 85 • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• Editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Updated title of [RS_ECUC_00066].
2015-07-31	4.2.2	AUTOSAR Release Management	• Layout Update.
2014-10-31	4.2.1	AUTOSAR Release Management	• Updated [RS_ECUC_00008]. • Added [RS_ECUC_00085]. • Added [RS_ECUC_00086]. • Tracing update.
2013-10-31	4.1.2	AUTOSAR Release Management	• Layout Update.

2013-03-15	4.1.1	AUTOSAR Administration	• Layout Update.
2011-12-22	4.0.3	AUTOSAR Administration	• Updated [RS_ECUC_00083]. • Added detailed change history in chapter 5.
2010-02-02	3.1.4	AUTOSAR Administration	• Introduction of Variant Handling • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• "Advice for users" revised • "Release Notes" added • "Revision Information" added
2006-11-28	2.1	AUTOSAR Administration	• Added requirements [RS_ECUC_00076] • Legal disclaimer revised
2006-05-16	2.0	AUTOSAR Administration	• Initial release

Disclaimer

Table of Contents

References

[1] Methodology
AUTOSAR_TR_Methodology

[2] Standardization Template
AUTOSAR_TPS_StandardizationTemplate

[3] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral

[4] Requirements on Runtime Environment
AUTOSAR_SRS_RTE

[5] Glossary
AUTOSAR_TR_Glossary

[6] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate

[7] XML Schema Production Rules
AUTOSAR_TPS_XMLSchemaProductionRules

[8] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration

[9] Specification of ECU Configuration Parameters (XML)
AUTOSAR_MOD_ECUConfigurationParameters

[10] Requirements on Standardization Template
AUTOSAR_RS_StandardizationTemplate

[11] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture

[12] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate

[13] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate

[14] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping

1 Scope of this document

ECU Configuration is one activity performed during the development of an AUTOSAR ECU.

The input to the ECU Configuration is one portion of the System Configuration Description which is called ECU Extract of System Configuration. The activity of ECU Configuration is to provide configuration information for all the software within one ECU. This spans from AUTOSAR SW-Components over the RTE Configuration to the multitude of Basic Software Modules.

The Output of the ECU Configuration is the ECU Configuration Description which is used to actually generate and build the ECU Executable.

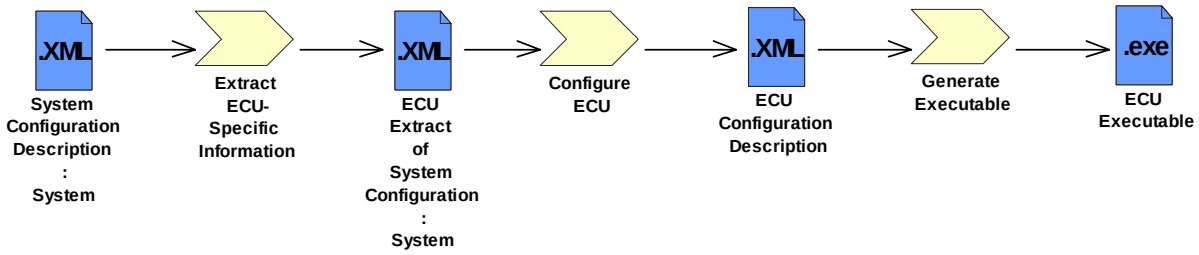

Figure 1.1: Overview AUTOSAR Methodology [1]

The main focus of this requirements document is the format of the ECU Configuration Description.

1.1 Document Conventions

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([2]).

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([2]).

2 Related Documentationn

2.1 Input Documents

The following input documents have been used in the development of these requirements:

- AUTOSAR General Requirements on Basic Software Modules [3]
- AUTOSAR Requirements on Runtime Environment [4]
- AUTOSAR Methodology [1]
- AUTOSAR Glossary [5]
- AUTOSAR Generic Structure Template [6]
- AUTOSAR XML Schema Production Rules [7]

2.2 Specification Documents

The requirements collected in this document will be satisfied by two specification documents:

- ECU Configuration Specification [8] This document provides the outline of the configuration methodology and the development guidelines for the ECU Configuration Parameters.
- ECU Configuration Parameters XML [9] This document contains the specification of the AUTOSAR standardized configuration parameters of BSW, RTE, SW-Components and ECU integration.

2.3 Abbreviations

Abbreviation	*Meaning*
BSW	Basic Software
BSWMD	Basic Software Module Description
ECUC	ECU Configuration
SW-C	Software Component

Table 2.1: Abbrevisations

3 Requirements Tracing

The following table references the requirements specified in [10] and links to the fulfillments of these.

Requirement	Description	Satisfied by
[RS_BRF_01024]	AUTOSAR shall provide naming rules for public symbols	[RS_ECUC_00086]
[RS_BRF_01028]	AUTOSAR shall provide naming conventions for symbols in its documentation	[RS_ECUC_00086]
[RS_BRF_01120]	AUTOSAR shall support re-flashing of configured BSW data	[RS_ECUC_00008] [RS_ECUC_00085]
[RS_BRF_01136]	AUTOSAR shall support variants of configured BSW data resolved after system start-up	[RS_ECUC_00078] [RS_ECUC_00079] [RS_ECUC_00080] [RS_ECUC_00082] [RS_ECUC_00083] [RS_ECUC_00084]
[SRS_BSW_00159]	All modules of the AUTOSAR Basic Software shall support a tool based configuration	[RS_ECUC_00049]
[SRS_BSW_00167]	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	[RS_ECUC_00050]
[SRS_BSW_00344]	BSW Modules shall support link-time configuration	[RS_ECUC_00048]
[SRS_BSW_00345]	BSW Modules shall support pre-compile configuration	[RS_ECUC_00047]

Table 3.1: RequirementsTracing

4 Requirements on ECU Configuration

4.1 Requirements on the Template

The requirements in this section all concern how the ECU Configuration Template shall be defined.

[RS_ECUC_00032] ECU Configuration Description shall be the root for the whole configuration information of an ECU ⌈

Type:	valid
Description:	The ECU Configuration Template shall become the backbone for the integration of software on a particular ECU. Any necessary configuration information shall be aggregated or referenced in this template.
Rationale:	Integration of software means basically the resolution of interdependencies among particular parts of the software. The resolution process can only be performed properly if all relevant information is accessible. Note that this requirement does not imply that all relevant information is copied into the template. References to elements in other templates may be included to avoid redundant elements in the model.
Use Case:	The ECU Configuration Template will include references to SW Components described using the SW component part of the meta model. It will also allow to specify the sequencing of runnables within task, something which is not specified by any other template.
Dependencies:	Consequence: ECU Configuration tools (editors and generators) need to be able to read parts of other templates as well (like System Template, SW Component Template, ECU Resource Template).
Supporting Material:	–

⌋()

[RS_ECUC_00072] Support for referencing from dependent containers ⌈

Type:	valid
Description:	It shall be possible to define references from parameter container to other parameter definitions in the ECUC Parameter definition and to elements in other AUTOSAR templates for standardized AUTOSAR Configuration parameters.
Rationale:	If there is a reference between two containers then it is assumed that there is a dependency from the referencing container to the referenced container.
Use Case:	Examples for such dependencies are: • PortPin references the driver that uses this Pin • The BitPosition of a COM Signal in an IPdu is defined by the SignalPosition in the SystemTemplate

▽

△

Dependencies:	–
Supporting Material:	–

⌋()

[RS_ECUC_00050] Specify ECU Configuration Parameter Definition ⌈

Type:	valid
Description:	Definition of ECU Configuration Parameters and their constraints such as configuration class, value range, multiplicities have to be captured.
Rationale:	Constraints on configuration parameters are important to be identified and verified. This also includes validity of the actual parameters themselves, like ranges and predefined values.
Use Case:	• Clock frequency needs to be > 0; • 0 <= Data_Length <= 8
Dependencies:	–
Supporting Material:	–

⌋(*SRS_BSW_00167*)

[RS_ECUC_00055] Support standardization of mandatory and optional configuration parameters ⌈

Type:	valid
Description:	In the standardization of ECU Configuration Parameter Definitions it shall be possible to define if a parameter is mandatory or optional. The definition of optional and mandatory shall be as follows: A mandatory parameter must be implemented by all implementations of that module. It must always be present in a completed ECU configuration description. An optional parameter may be omitted by an implementation.
Rationale:	Parameters which are not applicable for every implementation can anyhow be standardized if it is clarified that not every implementation needs to support that parameter. Note that for a concrete implementation a parameter is either supported and must then be present in the ECU configuration or not supported and must then not be present. Thus for a concrete implementation no optionality exists anymore, only in the standardized version of the parameter definition. Note that an implementation may still choose to fix the value for a mandatory parameter (e.g. an object code delivery fixes the values of pre-compile parameters). Anyhow, the value selected must then be stated in the ECU Configuration Description [8].

▽

△

Use Case:	The parameter ACTIVATE_PULLUP in the PORT Module is only needed if the hardware supports it. Thus an implementation of the PORT module may omit the parameter if the target HW does not support pull ups.
Dependencies:	–
Supporting Material:	–

⌋()

[RS_ECUC_00070] Support mandatory and optional containers ⌈

Type:	valid
Description:	It shall be possible to group configuration parameters into a hierarchy of containers. It shall be possible to define if a container is mandatory or optional. The definition of optional and mandatory shall be as follows: A mandatory container defined in a module must be implemented by all implementations of that module. It must always be present in a completed ECU Configuration description. All parameters and sub-containers of that container must also be present unless they are optional. An optional container may be omitted by an implementation. If an optional container is omitted, all parameters and sub-containers defined within that container are omitted as well, independent of whether they are defined optional or mandatory.
Rationale:	Containers which are not applicable for every implementation can anyhow be standardized if it is clarified that not every implementation needs to support that container.
Use Case:	An ADC module may define a set of parameters for each ADC channel for the microprocessor. If there is no ADC channel present on a specific microcontroller, none of these parameters is useful. If a channel is defined, then all mandatory parameters for that channel should be filled in. So it is better to define an optional container that includes mandatory parameters than defining several optional parameters (which may then be omitted independently of each other).
Dependencies:	–
Supporting Material:	–

⌋()

[RS_ECUC_00043] Duplication free description ⌈

Type:	valid
Description:	ECU Configuration description shall contain each configuration information only once, even if it is required to configure several modules.

▽

△

Rationale:	This helps to avoid inconsistency in the description and keeps the description compact. Note that it is still allowed to include derived information: If configuration parameter C may be in principle calculated from configuration items A and B, it is still possible to include C in the template.
Use Case:	Definition of tasks used in the ECU may be relevant for OS configuration and RTE configuration. But it should only be defined once in the ECU configuration template.
Dependencies:	–
Supporting Material:	–

⌋()

[RS_ECUC_00002] Support of vendor-specific ECU Configuration Parameters ⌈

Type:	valid
Description:	ECU Configuration shall provide means for vendor specific information in addition to the standard configuration parameters, which are defined in the SWS of the BSW module.
Rationale:	It has to be assured that really all ECU information can be stored in ECU Configuration description. So specific items can be passed to the generation tools.
Use Case:	A special attribute and some tool settings for NVRAM Manager have to be defined in the ECU Configuration Parameter Definition and the actual values stored in the ECU Configuration Description.
Dependencies:	[RS_ECUC_00018] is requiring extension mechanism to support the evolution of the standard.
Supporting Material:	–

⌋()

[RS_ECUC_00046] Support definition of configuration class ⌈

Type:	valid
Description:	The ECU Configuration parameter definition shall support the definition of the configuration class of configuration parameters.
Rationale:	The BSW SWS allows several configuration classes: pre-compile time link time ▽

▽

△

	△ post-build time. The standardization of configuration parameters does not necessarily fix the configuration class of parameters. Instead it may define different configuration class implementation variants of a module. An actual BSW module implementation does fix the configuration class for each parameter. This information has to be stored in the ECU Configuration Parameter definition.
Use Case:	If the parameter XXX_DEV_ERROR_DETECT is specified to be only pre-compile configurable then the parameter value has to be identical in all configuration sets of this BSW module instance.
Dependencies:	–
Supporting Material:	–

⌋()

[RS_ECUC_00012] One description mechanism for different configuration classes ⌈

Type:	valid
Description:	There shall be only one configuration description mechanism for all different kinds of configuration classes. So regardless if "pre-compile-time" or "link-time" or "post-build-time" configuration is described the description format shall be the same.
Rationale:	The description of the configuration shall be independent from the BSW implementations. Additional information might be needed if the parameter class changes in the implementation, e.g. memory location for post-build-time parameters.
Use Case:	If one BSW module is configured to be "pre-compile-time" configuration - and this module is exchanged against a "link-time" configuration one - the ECU Configuration description shall not change due to this exchange.
Dependencies:	–
Supporting Material:	–

⌋()

[RS_ECUC_00049] ECU Configuration description shall be tool process-able ⌈

Type:	valid
Description:	The ECU Configuration Descriptions shall be read- and writable by ECU Configuration tools and generators.
Rationale:	The configuration of an ECU shall be supported by tools.

▽

△

Use Case:	ECU Configuration will have to have tool support.
Dependencies:	–
Supporting Material:	–

⌋(*SRS_BSW_00159*)

[RS_ECUC_00065] Development according to the AUTOSAR Generic Structure Template document ⌈

Type:	valid
Description:	The ECU Configuration Description shall be developed according to the AUTOSAR Generic Structure Template.
Rationale:	The experience and tools already available for the AUTOSAR Modeling shall be reused.
Use Case:	The template for the ECU Configuration is similar to other templates already done within AUTOSAR.
Dependencies:	–
Supporting Material:	AUTOSAR Generic Structure Template [6]

⌋()

[RS_ECUC_00066] Transformation of ECUC model according to the AUTOSAR XML Schema Production Rules ⌈

Type:	valid
Description:	The ECU Configuration Template schema shall be derived using the model transformation described in the AUTOSAR XML Schema Production Rules.
Rationale:	The experience and tools already available for the AUTOSAR Modeling from other templates shall be reused.
Use Case:	The template for the ECU Configuration is similar to other templates already done within AUTOSAR.
Dependencies:	–
Supporting Material:	AUTOSAR XML Schema Production Rules [7].

⌋()

[RS_ECUC_00018] Extension handling ⌈

Type:	valid
Description:	The ECU Configuration must allow for later extensions to support the evolution of the standard.
Rationale:	It may be possible, that extensions/additional aspects must be handled in the ECU Configuration and ECU resources which are currently not part of the standard. But at some point in time these extensions shall become part of the standards next version.
Use Case:	If e.g. new type of ECU resources become available, it must be possible to easily incorporate this into the AUTOSAR architecture without changing the standard at once, since changes to the standard may take some time.
Dependencies:	[RS_ECUC_00002] is specifying vendor-specific extensions that will never be part of the standard.
Supporting Material:	–

⌋()

[RS_ECUC_00074] Support Sequential ECU Configuration ⌈

Type:	valid
Description:	It shall be possible to edit the different parts of the ECU Configuration Description separately.
Rationale:	The ECU Configuration Description contains the configuration of several modules whose configurations influence each other. So the dependencies have to be resolved sequentially and iteratively.
Use Case:	The RTE configuration editor can allocate an OS Task but still the OS configuration editor is able to change the properties of this OS Task.
Dependencies:	[RS_ECUC_00025] implies that the tools can be used iteratively as well.
Supporting Material:	–

⌋()

[RS_ECUC_00025] Compatible with iterative design ⌈

Type:	valid
Description:	Support iterative design as mandated by AUTOSAR methodology.
Rationale:	It is rare that development of any product is finished without any design changes. A design change invariably requires iteration of some portion of the design stages. AUTOSAR tools will be used iteratively and this includes ECU configuration.
Use Case:	Product design change after configuration of an ECU which requires a new ECU configuration to be developed.

▽

△

Dependencies:	This is closely related with the usage of the System Constraint/Configuration Description. [RS_ECUC_00074]
Supporting Material:	–

⌋()

[RS_ECUC_00078] Variable existence of container on value side ⌈

Type:	valid
Description:	The existence of a container and its substructure shall be variable in the ECU Configuration Parameter Description.
Rationale:	Containers hold most of the structure of the configuration description. By enabling variability on container many use-cases can be supported.
Use Case:	Variable existence of an OSTask.
Dependencies:	–
Supporting Material:	–

⌋(*RS_BRF_01136*)

[RS_ECUC_00079] Variable existence of value ⌈

Type:	valid
Description:	The existence of a parameter or reference shall be variable in the ECU Configuration Parameter Description.
Rationale:	In case a parameter or reference is optional the existence of the value can be variable.
Use Case:	Specifying the choice between several alternative parameters.
Dependencies:	–
Supporting Material:	–

⌋(*RS_BRF_01136*)

[RS_ECUC_00080] Variable value ⌈

Type:	valid
Description:	The value of a parameter or reference shall be variable in the ECU Configuration Parameter Description.
Rationale:	Calculating the value of a parameter based on variant selectors.

▽

△

Use Case:	Configuring multiple baud rates of a bus based on variation.
Dependencies:	–
Supporting Material:	–

⌋(*RS_BRF_01136*)

[RS_ECUC_00082] Variable lower and upper multiplicity in ECU Configuration Parameter definition ⌈

Type:	valid
Description:	The definition of the lower and upper multiplicity in the ECU Configuration Parameter definition shall be variable.
Rationale:	Through the lower and upper multiplicity the existence of elements in the ECU Configuration Parameter description is controlled. Making the bounds variable allows for freedom in the definition.
Use Case:	Make the decision whether parameters are mandatory or optional variant.
Dependencies:	–
Supporting Material:	–

⌋(*RS_BRF_01136*)

[RS_ECUC_00083] Variable default value in ECU Configuration Parameter definition ⌈

Type:	valid
Description:	The default value in the ECU Configuration Parameter definition shall be variable. This variation is not supported for "enumeration parameter definition"
Rationale:	The default value is a first value which is entered for parameters. To allow meaningful default values these shall be variant in order to adjust them.
Use Case:	Make the value of the default variant.
Dependencies:	–
Supporting Material:	–

⌋(*RS_BRF_01136*)

[RS_ECUC_00084] Variable min and max ranges in ECU Configuration Parameter definition ⌈

Type:	valid
Description:	The min and max ranges of ECU Configuration Parameters shall be variable in the ECU Configuration Parameter definition.
Rationale:	Relationships between variation and the min/max values are quite often in the definition of the ECU Configuration Parameter definition.
Use Case:	The max value for the NvRamBlockId can be either 255 or 65535, depending whether a 8- or 16-bit selection has been chosen.
Dependencies:	–
Supporting Material:	–

⌋(*RS_BRF_01136*)

[RS_ECUC_00086] The TPS_ECUConfiguration shall provide naming conventions for public symbols. ⌈

Type:	valid
Description:	The TPS_ECUConfiguration shall provide naming conventions for public symbols. This especially includes requirement ids, module abbreviations, meta data and configuration symbols used in the document of a release.
Rationale:	Avoid ambiguities and name clashes inside the specification. Provide a consistent uniform presentation of meta data to the reader of the specification. Allow automatic processing of specification elements.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋(*RS_BRF_01024*, *RS_BRF_01028*)

4.2 Requirements from the clients of the ECUC

[RS_ECUC_00047] Pre-compile time configuration of BSW ⌈

Type:	valid
Description:	The configuration of parameters of BSW at pre-compile time shall be possible for configuration parameters that are defined to be pre-compile time configurable.
Rationale:	Some AUTOSAR BSW module's configuration parameters are defined to be pre-compile time configurable for efficiency reasons. The ECU Configuration needs to support those parameters.

▽

△

Use Case:	Support of pre-compile time configuration parameters.
Dependencies:	–
Supporting Material:	AUTOSAR Glossary [5]

⌋(*SRS_BSW_00345*)

[RS_ECUC_00048] Link time configuration of BSW ⌈

Type:	valid
Description:	The configuration of parameters of BSW during link time shall be possible for configuration parameters that are defined to be link time configurable.
Rationale:	When a BSW module is delivered as object code it can only be configured at link time or post-build time.
Use Case:	Support of link time configuration parameters.
Dependencies:	–
Supporting Material:	AUTOSAR Glossary [5]

⌋(*SRS_BSW_00344*)

[RS_ECUC_00008] Post-build time configuration of BSW ⌈

Type:	valid
Description:	It shall be possible to reconfigure configuration parameters post-build time that are defined to be post-build configurable.
Rationale:	This will allow post-build time configuration of the BSW Components in an ECU to adapt to changes in the surrounding system. The effect on the overall system must be taken into account when defining a parameter as post-build time configurable.
Use Case:	The FMC development and after sales methodology heavily depends on the possibility to reconfigure the CAN and LIN communication post-build time. The main configuration items are mapping of signals into frames, frame priority and frame timing.
Dependencies:	–
Supporting Material:	AUTOSAR Glossary [5]

⌋(*RS_BRF_01120*)

[RS_ECUC_00085] Handling different configuration variants at post-build time ⌈

Type:	valid
Description:	It shall be possible to have variation points in the ECU configuration which are bound at post-build time.
Rationale:	This will allow the existence of several ECU configuration variants in one ECU.
Use Case:	The same ECU software can be used for multiple ECUs which share the same application (e.g. left and right door modules) and the correct configuration for each one of them is chosen at runtime.
Dependencies:	–
Supporting Material:	AUTOSAR Glossary [5]

⌋(*RS_BRF_01120*)

[RS_ECUC_00039] Support configuration of BSW ⌈

Type:	valid
Description:	ECU Configuration SHALL support all configuration parameters defined in the BSW SWS documents. The list of all supported modules can be found in the Layered Software Architecture document [11].
Rationale:	The BSW has to be configured out of the ECU Configuration description.
Use Case:	• BSW SWS define a list of configuration parameters. These parameters need to be reflected in the ECU Configuration template. • The OS SWS will identify configuration parameters from existing formats (such as OIL) and place them in the SWS. In that case only the content of OIL is used in the ECUC, not the actual format itself.
Dependencies:	–
Supporting Material:	Related released specifications of BSW modules [11].

⌋()

[RS_ECUC_00015] Configuration of multiple instances of BSW modules ⌈

Type:	valid
Description:	It shall be possible to describe the configuration of multiple instances of one BSW module type on one ECU - if applicable.
Rationale:	Some BSW module types can be present multiple times within one ECU, using different implementations, e.g. FLASH, EEP and watchdog drivers. For some BSW modules it is not possible to have multiple instances like the OS, NVRAM-Manager, etc.
Use Case:	When two different external EEPROM chips from different vendors are present on an ECU, there is a need for two different BSW drivers to cope with these.
Dependencies:	–

▽

△

Supporting Material:	–

⌋()

[RS_ECUC_00021] Select AUTOSAR SW Component and BSW Module implementation ⌈

Type:	valid
Description:	Allow selection of a specific SW Module implementation if more than one is available.
Rationale:	Multiple implementations (code) of modules may be available for incorporation into a system configuration. A specific selection may be required.
Use Case:	• Multiple CAN drivers from different vendors. • Multiple driver versions from a single vendor. • Different implementations of an AUTOSAR SW Component.
Dependencies:	–
Supporting Material:	–

⌋()

[RS_ECUC_00068] Mapping of software sections to memory ⌈

Type:	valid
Description:	Parts of software (code, data) shall be map-able to specific memory areas.
Rationale:	Each software consists of several sections which are defined when developing/compiling the software. Those sections need to be placed in different memory areas on the ECU.
Use Case:	• If data shall be post-build configurable it needs to be placed in non-volatile memory. The location the data is placed is needed for tools to be able to change the data later on. • Certain SW variables have to be placed in a NOINIT memory area which will not be initialized during ECU start-up.
Dependencies:	Software needs to publish the memory sections that have been used while developing/compiling the software. This needs to be part of the BSWMD [12] and the SWC-T [13].
Supporting Material:	Specification of Memory Mapping [14]

⌋()

[RS_ECUC_00040] Support configuration of RTE ⌈

Type:	valid
Description:	The generation and configuration of the RTE shall be supported.
Rationale:	The RTE has to be generated and configured on the basis of the ECUC's information.
Use Case:	Analyze all configuration requirements of RTE SWS document.
Dependencies:	–
Supporting Material:	–

⌋()

[RS_ECUC_00076] Support the configuration of which AUTOSAR Services are available on a specific ECU ⌈

Type:	valid
Description:	AUTOSAR defines several services which may be integrated into an ECU. It shall be possible to define which AUTOSAR Services are present on a specific ECU.
Rationale:	There may be ECUs which do not need all AUTOSAR Services, so a subset may be defined.
Use Case:	If the ECU does not use any NvRam there is no need for a NvRam Manager on this ECU.
Dependencies:	–
Supporting Material:	–

⌋()

4.3 Requirements from Software Components

[RS_ECUC_00041] Support AUTOSAR SW-Component integration ⌈

Type:	valid
Description:	The integration of AUTOSAR SW Components shall be supported.
Rationale:	AUTOSAR SW Components shall be instantiated on an ECU. The relevant information has to be provided to enable their integration and execution on that ECU.
Use Case:	Usage of AUTOSAR SW Components.
Dependencies:	–
Supporting Material:	–

⌋()

[RS_ECUC_00073] Support Service Configuration of AUTOSAR SW Components ⌈

Type:	valid
Description:	ECU Configuration shall support configuration of the interface between the AUTOSAR SW Components and the AUTOSAR Services.
Rationale:	During the configuration of an ECU, specific values for AUTOSAR SW Components service access need to be configured.
Use Case:	The SW Component requires NVRAM Blocks with symbolic IDs. The NVRAM Manager configuration is assigning specific ID values and needs to configure those values in the SW Component.
Dependencies:	–
Supporting Material:	–

⌋()

[RS_ECUC_00016] Execution order of runnable entities within one OS task ⌈

Type:	valid
Description:	It shall be possible to describe the execution order of runnable entities with OS tasks across SW-Component borders.
Rationale:	The SW-C template can only describe constraints on the execution order of runnable entities within one SW-Component. But it has to be possible to describe execution order between SW Component's runnable entities mapped to a specific OS task to establish control flow algorithms where the control flow is passed from one SW-C to the next in a defined order.
Use Case:	Three periodically executed runnable entities are mapped to one OS task. It has to be defined in which order those runnable entities are to be executed when the OS task is activated.
Dependencies:	–
Supporting Material:	–

⌋()

4.4 Requirements on the Configuration Parameter Definition

[RS_ECUC_00071] Support for Generic Configuration Editor ⌈

Type:	valid
Description:	ECU Parameter definition shall be defined in a tool processable format. The parameter definitions shall contain enough information to support generic configuration editors.
Rationale:	A Generic Configuration Editor may read in parameter definitions for standardized and HW-vendor specific parameters. It may then display all defined parameters and allows to fill a ECU Configuration Description with all defined parameters.
Use Case:	Reduce the number of configuration editors required to configure an ECU. Simple BSW modules should be configurable with generic configuration editors. This also reduces the effort of BSW vendors who do not need to write specific configuration editors for simple BSW modules.
Dependencies:	–
Supporting Material:	–

⌋()

4.5 Process requirements

These requirements provide information what the goal for the document is going to be. They are only informative in the sense that they are used to communicate the actions taken and will be removed at a later stage.

[RS_ECUC_00029] Identify mechanisms not criteria ⌈

Type:	valid
Description:	The WP products will only identify the methods and mechanisms that may be used to configure ECUs but will not identify any engineering trade-offs (e.g. between performance and size) that must be considered by configuration engineers.
Rationale:	The mechanisms are application independent so all potential impacts cannot be assessed. Application specific trade-offs must be considered by engineers utilizing the mechanism for an application build.
Use Case:	The configuration parameter definition will only define valid ranges, but not provide algorithms for finding the best setting.
Dependencies:	This should be part of the WP scope definition chapter of some deliverable.
Supporting Material:	–

⌋()

[RS_ECUC_00030] Clarify configuration terminology ⌈

Type:	valid
Description:	Ensure that all configuration terminology introduced into AUTOSAR is adequately explained, by glossary entries or other documentation.
Rationale:	Terms such as "pre-compile configuration", "link time configuration" and "post-build time configuration" mean different things depending on the point in development that they are applied. The work package must ensure that any terminology used is adequately clarified for each context in which it may appear.
Use Case:	Powertrain Applications Calibration Engineers interpret the meaning of "post-build time configuration" differently to Powertrain Software Engineers.
Dependencies:	–
Supporting Material:	–

⌋()

4.6 External Requirements

These are requirements that the ECU Configuration Description is not able to satisfy by its own. These requirements are placed on other AUTOSAR work products.

[RS_ECUC_00057] Memory needs of BSW modules ⌈

Type:	valid
Description:	BSW modules shall provide information on the memory needs for the module. This is also true when the memory need is dependent on the actual configuration.
Rationale:	The amount of memory required typically depends on the configuration of the module and is finally determined at latest after configuration.
Use Case:	Post build data for COM stack varies in size, depending on the number of frames sent and received, code size (ROM) may vary for generated code. Data structures in RAM and EEPROM may vary as well.
Dependencies:	This is a requirement on the BSW Module Description, even though the actual value might depend on the configuration.
Supporting Material:	–

⌋()

[RS_ECUC_00036] Identify un-initialized resources. ⌈

Type:	valid
Description:	ECU Configuration shall identify unused resources present in an ECU.
Rationale:	Resources not used by any BSW module (standard AUTOSAR or Complex Device Driver) are not initialized by ECU software. This may affect the robustness of an ECU and cause unexpected interrupts or result in spurious EMC issues. (Note: This is a requirement on the tool - not on the template)
Use Case:	When unused (thus un-initialized) resources are identified a user may add appropriate drivers to the ECU which at least initialize these resources (e.g. explicitly disable unused interrupts or I/O ports).
Dependencies:	–
Supporting Material:	This would have to be checked by the tool-chain and/or process.

⌋()

[RS_ECUC_00056] Identify conflicting usage of micro controller registers ⌈

Type:	valid
Description:	ECU Configuration tool shall identify micro controller registers which are configured and/or accessed by more than one BSW module in an inconsistent way.
Rationale:	BSW modules (AUTOSAR drivers, complex drivers etc) may directly access micro controller registers. When BSW modules, possibly even from different vendors, are integrated into a single ECU then these modules may try to configure the same micro controller registers differently. Such conflicts must be identified. (Note: This is a requirement on the tool - not on the ECUC template)
Dependencies:	–
Use Case:	4 LSB of a register are used by the GPT Driver, the 4 MSB of the same register are used by the ICU. If both modules write the entire register (8 bit) then they overwrite each other's configuration setting. Once a conflict is identified, different BSW module implementations or different BSW configuration may be used to resolve the problem.
Supporting Material:	[RS_BSWMD_00009] [12]

⌋()

[RS_ECUC_00062] Configuration option to initialize unused memory ⌈

Type:	valid
Description:	The AUTOSAR build environment shall provide a configuration option to initialize unused memory to a default value.

▽

△

Rationale:	Faulty ECU software may access memory locations which are normally not used. Faulty ECU software may even attempt to execute code in memory locations which are not used. Unused RAM contains random data which can cause non-deterministic behavior in case of an erroneous read-access or execution. Initialization of unused memory improves reliability. (Note: This is a requirement on the build tool/environment - not on the template)
Use Case:	Fill unused memory (especially ROM/FLASH) with HALT instructions to increase robustness against unwanted code execution. Fill RAM with default value (0 or HALT instruction) to reduce threat of non-deterministic behavior in case of improper pointer operations.
Dependencies:	–
Supporting Material:	–

⌋()

5 Change History

5.1 Change History for AUTOSAR R4.0.1 against R3.1.5

5.1.1 Added Traceables in R4.0.1

Id	Heading
[RS_ECUC_00078]	Variable existence of container
[RS_ECUC_00079]	Variable existence of value
[RS_ECUC_00080]	Variable value
[RS_ECUC_00082]	Variable lower and upper multiplicity in ECU Configuration Parameter definition
[RS_ECUC_00083]	Variable default value in ECU Configuration Parameter definition
[RS_ECUC_00084]	Variable min and max ranges in ECU Configuration Parameter definition

Table 5.1: Added SRS Items in R4.0.1

5.1.2 Changed Traceables in R4.0.1

Id	Heading
[RS_ECUC_00046]	Support definition of configuration class
[RS_ECUC_00065]	Development according to the AUTOSAR Generic Structure Template document
[RS_ECUC_00066]	Transformation of ECUC model according to the AUTOSAR Model Persistence Rules for XML
[RS_ECUC_00072]	Support for referencing from dependent containers

Table 5.2: Changed SRS Items in R4.0.1

5.1.3 Deleted Traceables in R4.0.1

Id	Heading
[ECUC_0075]	Support exactly one to be configured micro-controller per ECU

Table 5.3: Removed SRS Items in R4.0.1

5.2 Change History for AUTOSAR R4.0.3 against R4.0.1

5.2.1 Added Traceables in 4.2.1

none

5.2.2 Changed Traceables in R4.0.3

Id	Heading
[RS_ECUC_00083]	Excluded "enumeration parameter definition"

Table 5.4: Changed SRS Items in R4.0.3

5.2.3 Deleted Traceables in 4.2.1

none

5.3 Change History for AUTOSAR R4.1.1 against R4.0.3

No changes.

5.4 Change History for AUTOSAR R4.1.2 against R4.1.1

No changes.

5.5 Change History for AUTOSAR R4.1.3 against R4.1.2

No changes.

5.6 Change History for AUTOSAR R4.2.1 against R4.1.3

5.6.1 Added Traceables in 4.2.1

Id	Heading
[RS_ECUC_00085]	Handling different configuration variants at post-build time
[RS_ECUC_00086]	The TPS_ECUConfiguration shall provide naming conventions for public symbols.

Table 5.5: Added Traceables in 4.2.1

5.6.2 Changed Traceables in 4.2.1

Id	Heading
[RS_ECUC_00008]	Post-build time configuration of BSW
[RS_ECUC_00078]	Variable existence of container on value side
[RS_ECUC_00079]	Variable existence of value
[RS_ECUC_00080]	Variable value
[RS_ECUC_00082]	Variable lower and upper multiplicity in ECU Configuration Parameter definition
[RS_ECUC_00083]	Variable default value in ECU Configuration Parameter definition
[RS_ECUC_00084]	Variable min and max ranges in ECU Configuration Parameter definition

Table 5.6: Changed Traceables in 4.2.1

5.6.3 Deleted Traceables in 4.2.1

none

5.7 Change History for AUTOSAR R4.2.2 against R4.2.1

No changes.

5.8 Change History for AUTOSAR R4.3.0 against R4.2.2

5.8.1 Added Traceables in 4.3.0

none

5.8.2 Changed Traceables in 4.3.0

Id	Heading
[RS_ECUC_00066]	Transformation of ECUC model according to the AUTOSAR XML Schema Production Rules

Table 5.7: Changed Traceables in 4.3.0

5.8.3 Deleted Traceables in 4.3.0

none

5.9 Change History for AUTOSAR R4.3.1 against R4.3.0

5.9.1 Added Traceables in 4.3.1

none

5.9.2 Changed Traceables in 4.3.1

none

5.9.3 Deleted Traceables in 4.3.1

none

5.10 Change History for AUTOSAR R4.4.0 against R4.3.1

5.10.1 Added Traceables in 4.4.0

none

5.10.2 Changed Traceables in 4.4.0

none

5.10.3 Deleted Traceables in 4.4.0

none

5.11 Change History for AUTOSAR R19-11 against R4.4.0

5.11.1 Added Traceables in 19-11

none

5.11.2 Changed Traceables in 19-11

none

5.11.3 Deleted Traceables in 19-11

none

5.12 Change History for AUTOSAR R20-11 against R19-11

5.12.1 Added Traceables in R20-11

none

5.12.2 Changed Traceables in R20-11

none

5.12.3 Deleted Traceables in R20-11

none