| Document Title | Specification of ECU Resource Template |
|---|---|
| **Document Category** | Template specification (TPS) |
| **Document UID** | 60 |
| **Document Status** | published |
| **Part of AUTOSAR Standard** | Classic Platform |
| **Part of Standard Release** | R26-11 |

| Document Change History | | | |
|---|---|---|---|
| **Date** | **Release** | **Changed by** | **Description** |
| 2026-11-26 | R26-11 | AUTOSAR Release Management | • No content changes |
| 2025-11-27 | R25-11 | AUTOSAR Release Management | • Removed TTCan examples |
| 2024-11-27 | R24-11 | AUTOSAR Release Management | • No content changes |
| 2023-11-23 | R23-11 | AUTOSAR Release Management | • Editorial: wrap tables in specification items<br>• Extend HwPin attributes |
| 2022-11-24 | R22-11 | AUTOSAR Release Management | • Support for MACsec<br>• Support for firewall<br>• Modeled State Management<br>• Support for Service-oriented Vehicle Diagnostic<br>• minor corrections / clarifications / editorial changes |
| 2021-11-25 | R21-11 | AUTOSAR Release Management | • Removal of REST support<br>• Rework of signal-to-service conversion<br>• Rework of raw data stream configuration<br>• Rework of diagnostic mapping<br>• minor corrections / clarifications / editorial changes |

▽

△

| | | | |
|---|---|---|---|
| 2020-11-30 | R20-11 | AUTOSAR Release Management | • Remodeling of Phm contribution<br>• Reporting of Security Events<br>• Support for cryptographic Operations<br>• Remodeling of Diagnostic Mapping<br>• minor corrections / clarifications / editorial changes |
| 2019-11-28 | R19-11 | AUTOSAR Release Management | • Overhaul of Signal-to-Service Translation<br>• Support for Raw Data Streams<br>• Support for Vehicle Package<br>• Support for Service Versioning<br>• Changed Document Status from Final to published |
| 2019-03-29 | 19-03 | AUTOSAR Release Management | • Introduction of Diagnostic Port Interfaces<br>• Overhaul of Software Cluster and introduction of Software Package<br>• Support for Identity and Access Management<br>• Network Management Configuration |
| 2018-10-31 | 18-10 | AUTOSAR Release Management | • Finish introduction of CppImplementationDataType<br>• Support for optional elements in structures<br>• Rework configuration of adaptive platform modules |
| 2018-03-29 | 18-03 | AUTOSAR Release Management | • Time Synchronization<br>• DDS Deployment |

▽

△

| | | | |
|---|---|---|---|
| 2017-10-27 | 17-10 | AUTOSAR Release Management | • Optional elements in Service Interfaces<br>• Interaction with web services<br>• Secure Communication<br>• Support for interaction with crypto and persistency<br>• Signal-to-Service translation<br>• Support for E2E communication<br>• Platform Health Management<br>• Uploadable Software Package |
| 2017-03-31 | 17-03 | AUTOSAR Release Management | • Initial release |

**Disclaimer**

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

# Table of Contents

1 Introduction 9
1.1 Scope of the ECU Resource Template . . . 9
1.2 Overview ECU Resource Template . . . 10
1.3 Document Conventions . . . 10
2 General Hardware Description 13
2.1 Hardware Description Entity . . . 14
2.2 Hardware Type . . . 16
2.3 Hardware Element . . . 17
2.3.1 Multiple occurrence of Hardware Elements . . . 18
2.4 Hardware Pin and Pin Group . . . 19
2.5 Hardware Connection . . . 20
2.5.1 Scope of Connections . . . 22
2.6 Hardware Category Definition . . . 23
2.6.1 Vendor specific extensions of Hardware Category Definition . . . 26
2.7 Ecu Resource Variant Handling . . . 28
2.8 Documentation Support . . . 28
2.9 Infrastructural aspects . . . 28
3 Hardware Type Specific Description 30
3.1 HwElement categories . . . 30
3.1.1 Ecu . . . 30
3.1.2 Processing Unit . . . 31
3.1.3 Micro-Controller . . . 31
3.1.4 Memory . . . 31
3.1.5 Communication Controller . . . 31
3.1.6 Communication Transceiver . . . 32
3.1.7 Digital IO . . . 32
3.1.8 Analog IO . . . 32
3.1.9 Timer . . . 33
3.1.10 Watchdog . . . 33
3.1.11 SensorActuator . . . 33
3.2 HwPinGroup categories . . . 33
3.2.1 CommunicationPort . . . 33
3.3 HwPin categories . . . 34
A Examples 35
A.1 Hardware Element . . . 35
A.2 Hierarchy of Hardware Elements . . . 35
A.3 HwPinGroups and HwPins . . . 36
A.4 Hardware Element Connection . . . 37
A.5 Combined Example . . . 38

A.5.1 Micro-controller description . . . 39
A.5.2 Transceiver description . . . 40
A.5.3 Ecu description . . . 41
A.6 Attribute Definition . . . 42
A.7 Attribute Value Example . . . 43
B Glossary 45
C History of Constraints and Specification Items 48
C.1 Constraint and Specification Item History of this document according to AUTOSAR Release R4.0.2 . . . 48
C.2 Constraint and Specification Item History of this document according to AUTOSAR Release R4.0.3 . . . 48
C.2.1 Added Constraints in R4.0.3 . . . 48
C.3 Constraint and Specification Item History of this document according to AUTOSAR Release R4.1.1 . . . 48
C.3.1 Added Constraints in R4.1.1 . . . 48
C.3.2 Added Traceables in R4.1.1 . . . 48
C.4 Constraint and Specification Item History of this document according to AUTOSAR Release R4.1.2 . . . 50
C.5 Constraint and Specification Item History of this document according to AUTOSAR Release R4.1.3 . . . 50
C.6 Constraint and Specification Item History of this document according to AUTOSAR Release R4.2.1 . . . 50
C.7 Constraint and Specification Item History of this document according to AUTOSAR Release R4.2.2 . . . 50
C.8 Constraint and Specification Item History of this document according to AUTOSAR Release R4.3.0 . . . 51
C.9 Constraint and Specification Item History of this document according to AUTOSAR Release R4.3.1 . . . 51
C.10 Constraint and Specification Item History of this document according to AUTOSAR Release R4.4.0 . . . 51
C.11 Constraint and Specification Item History of this document according to AUTOSAR Release R19-11 . . . 51
C.12 Constraint and Specification Item History of this document according to AUTOSAR Release R20-11 . . . 51
C.13 Constraint and Specification Item History of this document according to AUTOSAR Release R21-11 . . . 51
C.13.1 Added Specification Items in R21-11 . . . 51
C.13.2 Changed Specification Items in R21-11 . . . 52
C.13.3 Deleted Specification Items in R21-11 . . . 52
C.13.4 Added Constraints in R21-11 . . . 52
C.13.5 Changed Constraints in R21-11 . . . 52
C.13.6 Deleted Constraints in R21-11 . . . 52

C.14 Constraint and Specification Item History of this document according to AUTOSAR Release R22-11 . . . 52
C.15 Constraint and Specification Item History of this document according to AUTOSAR Release R23-11 . . . 53
C.15.1 Added Specification Items in R23-11 . . . 53
C.15.2 Changed Specification Items in R23-11 . . . 53
C.15.3 Deleted Specification Items in R23-11 . . . 53
C.15.4 Added Constraints in R23-11 . . . 53
C.15.5 Changed Constraints in R23-11 . . . 53
C.15.6 Deleted Constraints in R23-11 . . . 53
C.16 Constraint and Specification Item History of this document according to AUTOSAR Release R24-11 . . . 54
C.16.1 Added Specification Items in R24-11 . . . 54
C.16.2 Changed Specification Items in R24-11 . . . 54
C.16.3 Deleted Specification Items in R24-11 . . . 54
C.16.4 Added Constraints in R24-11 . . . 54
C.16.5 Changed Constraints in R24-11 . . . 54
C.16.6 Deleted Constraints in R24-11 . . . 54
C.17 Constraint and Specification Item History of this document according to AUTOSAR Release R25-11 . . . 54
C.17.1 Added Specification Items in R25-11 . . . 54
C.17.2 Changed Specification Items in R25-11 . . . 55
C.17.3 Deleted Specification Items in R25-11 . . . 55
C.17.4 Added Constraints in R25-11 . . . 55
C.17.5 Changed Constraints in R25-11 . . . 55
C.17.6 Deleted Constraints in R25-11 . . . 55
C.18 Constraint and Specification Item History of this document according to AUTOSAR Release R26-11 . . . 55
C.18.1 Added Specification Items in R26-11 . . . 55
C.18.2 Changed Specification Items in R26-11 . . . 55
C.18.3 Deleted Specification Items in R26-11 . . . 55
C.18.4 Added Constraints in R26-11 . . . 56
C.18.5 Changed Constraints in R26-11 . . . 56
C.18.6 Deleted Constraints in R26-11 . . . 56
D Mentioned Class Tables 57

# References

[1] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[2] Generic Structure Template
AUTOSAR_FO_TPS_GenericStructureTemplate

[3] IEEE standard for radix-independent floating-point arithmetic
(ANSI/IEEE Std 854-1987)

[4] Software Component Template
AUTOSAR_CP_TPS_SoftwareComponentTemplate

[5] AUTOSAR XML Schema Production Rules
AUTOSAR_FO_TPS_XMLSchemaProductionRules

[6] Software Process Engineering Meta-Model Specification
http://www.omg.org/spec/SPEM/2.0/

# 1 Introduction

One of the most prominent goals of AUTOSAR is the standardization of descriptions relevant for automotive software applications. In this context, the description of underlying ECU hardware is one of the major topics to resolve.

This document contains a specification of the modeling elements required to describe the hardware to the necessary extent. One aspect of the ECU Resource Template is to provide the system design engineer with the necessary information to assist the system partitioning, e.g. available memory and communication means of dedicated ECUs. Another aspect of the ECU Resource Template is to support the ECU Configuration engineers and tools with information required for the configuration of the micro-controller and ECU abstraction layer residing on a particular ECU.

The focus of the ECU Resource Template is to describe an already engineered piece of hardware, its content and structure. It is not in the focus of the ECU Resource Template to support the design of electronics hardware itself. There are established tools and exchange formats to aid in the design of electronics hardware already available. But such tools may be able to export their design using the AUTOSAR ECU Resource Template format for later usage in AUTOSAR design tools.

Where applicable, please consult the glossary and the abbreviation list contained in this document. The general characteristics of the ECU Resource Description are introduced followed by a detailed description of the hardware components inside the ECU.

## 1.1 Scope of the ECU Resource Template

The scope of the ECU Resource Template is the description of ECUs by means of the following basic building blocks:

- Hardware Elements
- Hardware PinGroups and Hardware Pins
- Hardware Connections

The HW Elements are the main describing elements of an ECU, For example: Processing units, memory, peripherals and sensors/actuators. HW Elements have a unique name and can be identified within an ECU description. HW Elements do not necessarily have to be described on the level of an ECU. It is possible to describe HW Elements as parts of other HW Elements. By this means a hierarchical description of HW Elements can be created.

HW Elements provide HW PinGroups and HW Pins for being interconnected among each others. HW PinGroups allow a rough description of how certain groups of HW Pins are arranged. The detailed description can be done using the HW Pins.

HW Connections are used to describe connections on several levels:

- connections between HW Elements
- connections between HW PinGroups
- connections between HW Pins

The different levels of abstraction allow to define and gather the required information for the different use-cases of the ECU Resource Template. For a rough understanding how the HW Elements are arranged in the ECU the connections between HW Elements are sufficient. To actually know at which HW Pin a certain signal is provided the detailed HW Pin connections are required.

## 1.2 Overview ECU Resource Template

Figure 1.1 depicts the main elements of an ECU Resource description and their inter-relations.

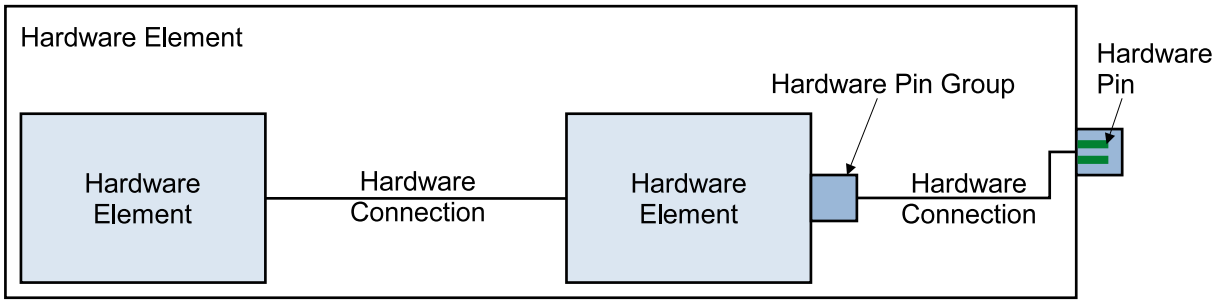



**Figure 1.1: Overview of ECU Resource template**

Modeling elements in the ECU Resource Template can be hierarchically organized. A particular ECU (the physical box containing the electronics) can be described as a hierarchical composition of one or more micro-controllers and ECU electronics. Each micro-controller is in turn composed of processing units, memory, peripherals and management units.

The same approach can be used to describe a particular ECU in combination with all sensors and actuators attached to the ECU.

The ECU Electronics is the hardware present on the ECU to guarantee the operation of the Processing Units (clock) as well as the conditioning of signals going out of the ECU or coming in (communication transceiver, amplifier, discrete electronics).

## 1.3 Document Conventions

Technical terms are typeset in mono spaced font, e.g. `PortPrototype`. As a general rule, plural forms of technical terms are created by adding “s” to the singular form, e.g.

PortPrototypes. By this means the document resembles terminology used in the AUTOSAR XML Schema.

This document contains constraints in textual form that are distinguished from the rest of the text by a unique numerical constraint ID, a headline, and the actual constraint text starting after the ⌈ character and terminated by the ⌋ character.

The purpose of these constraints is to literally constrain the interpretation of the AUTOSAR meta-model such that it is possible to detect violations of the standardized behavior implemented in an instance of the meta-model (i.e. on M1 level).

Makers of AUTOSAR tools are encouraged to add the numerical ID of a constraint that corresponds to an M1 modeling issue as part of the diagnostic message issued by the tool.

The attributes of the classes introduced in this document are listed in form of class tables. They have the form shown in the example of the top-level element AUTOSAR:

Please note that constraints are not supposed to be enforceable at any given time in an AUTOSAR workflow. During the development of a model, constraints may legitimately be violated because an incomplete model will obviously show inconsistencies.

However, at specific points in the workflow, constraints shall be enforced as a safeguard against misconfiguration.

The points in the workflow where constraints shall be enforced, sometimes also known as the “imposition time” of the constraint, are different for each model category, e.g. on the classic platform, the constraints defined for software-components are typically enforced prior to the generation of the RTE while the constraints against the definition of an Ecu extract shall be applied when the Ecu configuration for the Com stack is created.

For each document, possible imposition times of constraints are defined and the imposition times are typically mentioned in the constraint themselves to give a proper orientation for implementers of AUTOSAR authoring tools.

Let AUTOSAR be an example of a typical class table. The first rows in the table have the following meaning:

**Class**: The name of the class as defined in the UML model.

**Package**: The UML package the class is defined in. This is only listed to help locating the class in the overall meta model.

**Note**: The comment the modeler gave for the class (class note). Stereotypes and UML tags of the class are also denoted here.

**Base Classes**: If applicable, the list of direct base classes.

The headers in the table have the following meaning:

**Attribute**: The name of an attribute of the class. Note that AUTOSAR does not distinguish between class attributes and owned association ends.

**Type**: The type of an attribute of the class.

**Mul.**: The assigned multiplicity of the attribute, i.e. how many instances of the given data type are associated with the attribute.

**Kind**: Specifies, whether the attribute is aggregated in the class (`aggr` aggregation), an UML attribute in the class (`attr` primitive attribute), or just referenced by it (`ref` reference). Instance references are also indicated (`iref` instance reference) in this field.

**Note**: The comment the modeler gave for the class attribute (role note). Stereotypes and UML tags of the class are also denoted here.

Please note that the chapters that start with a letter instead of a numerical value represent the appendix of the document. The purpose of the appendix is to support the explanation of certain aspects of the document and does not represent binding conventions of the standard.

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see [1, Standardization Template].

Please note: By intent, TPS documents (and their traceable items) do not trace up to an RS (requirement item).

# 2 General Hardware Description

The ECU Resource Template utilizes the basic building blocks

- hardware elements
- hierarchies of hardware elements
- hardware pins
- hardware pin groups
- hardware connections

to describe the relevant aspects of the actual hardware. The ECU Resource Template allows however to choose the appropriate level of detail in the description of the hardware, depending on the use case. It also allows to describe arbitrary hardware and its connections.

**[TPS_ECUR_01015] Support of AUTOSAR Basic Software configuration** ⌈The primary goal of the ECU Resource Template is to support the configuration of the AUTOSAR Basic Software by providing information on the respective hardware and the how the hardware is connected to each other.⌋

In figure 2.1 the overview of the involved classes is shown.

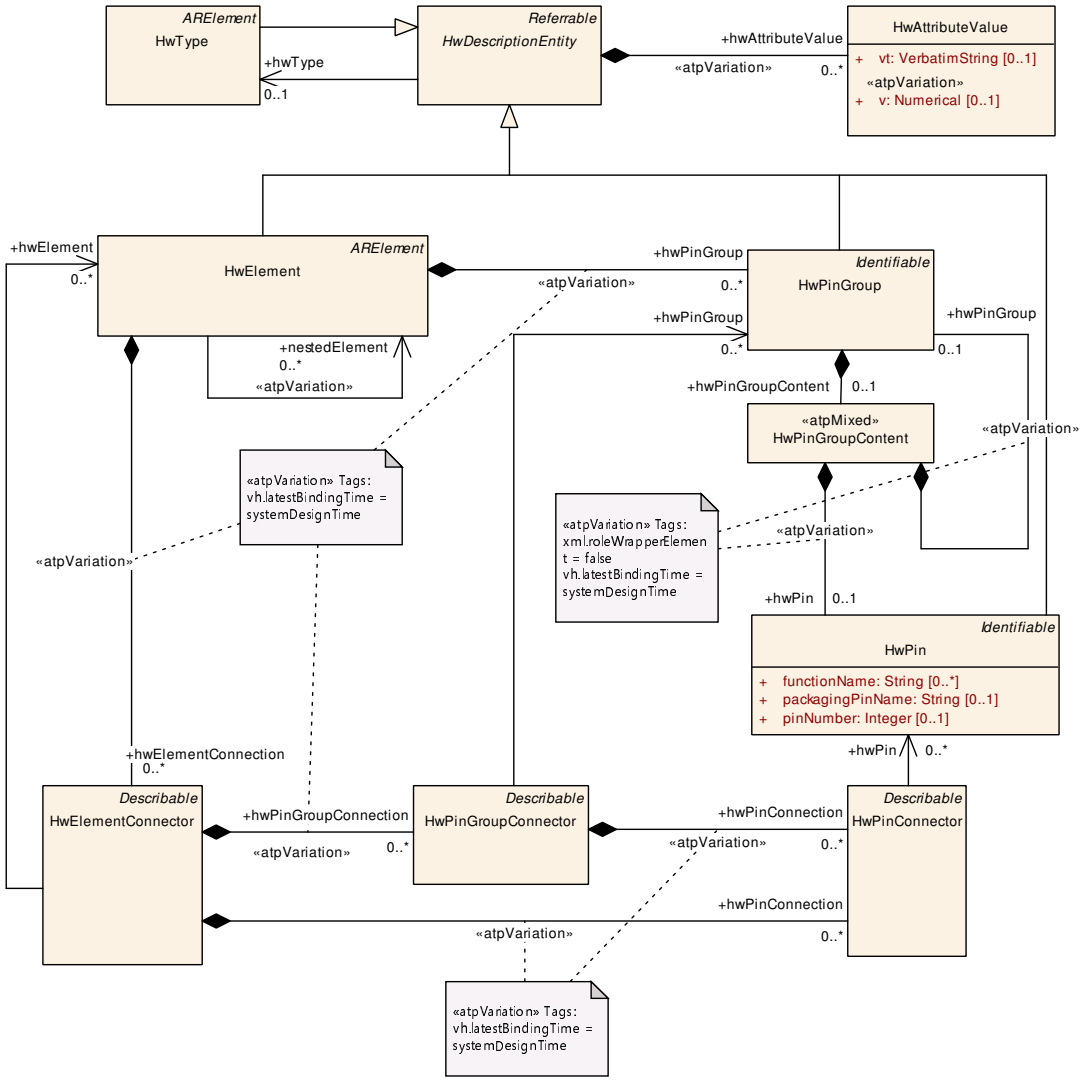



**Figure 2.1: Overview of ECU Resource template classes**

## 2.1 Hardware Description Entity

In order to allow flexibility of the ECU Resource Template with respect to the description of a multitude of hardware types the ECU Resource Template only provides the generic means to describe hardware elements and their connectivity. The description of specific attributes can be provided according to section 2.6.

**[TPS_ECUR_01002] Definition of Hardware Elements** ⌈The `HwDescriptionEntity` allows to provide a set of attribute values which are defined by one or more hardware categories.⌋

Please refer to chapter 3 for details on the actual applicable hardware categories and corresponding attributes.

The HwDescriptionEntity is able to specify for which hardware categories (see section 2.6) this HwDescriptionEntity is applicable. It is possible to define several references in the role hwCategory.

- **[TPS_ECUR_01000] Definition of HwCategory** ⌈It shall be possible to reference different kinds of HwCategory elements in order to describe different aspects of the hardware (e.g. a Can controller with an integrated Spi channel).⌋

- **[TPS_ECUR_01001] Extension of HwCategory** ⌈It shall be possible to extend the standardized HwCategory specification with additional attributes.⌋

  For more details see section Vendor specific extensions of Hardware Category Definition.

For a description of the hwType reference please refer to section 2.2.

Each HwDescriptionEntity may aggregate several HwAttributeValue elements.

| ***Class*** | ***HwDescriptionEntity*** (abstract) | | | |
|---|---|---|---|---|
| ***Note*** | This meta-class represents the ability to describe a hardware entity. | | | |
| ***Base*** | *ARObject*, *Referrable* | | | |
| ***Subclasses*** | HwElement, HwPin, HwPinGroup, HwType | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| hwAttribute Value | HwAttributeValue | * | aggr | This aggregation represents a particular hardware attribute value.<br>**Stereotypes:** atpVariation<br>**Tags:**<br>atp.Splitkey=hwAttributeValue, hwAttributeValue.variation Point.shortLabel<br>vh.latestBindingTime=systemDesignTime<br>xml.sequenceOffset=50 |
| hwCategory | HwCategory | * | ref | One of the associations representing one particular category of the hardware entity.<br>**Tags:**<br>atp.Splitkey=hwCategory<br>xml.sequenceOffset=30 |
| hwType | HwType | 0..1 | ref | This association is used to assign an optional HwType which contains the common attribute values for all occurences of this HwDescriptionEntity. Note that Hw Types can not be redefined and therefore shall not have a hwType reference.<br>**Tags:** atp.Splitkey=hwType |

**Table 2.1: HwDescriptionEntity**

**[TPS_ECUR_01014] Definition of HwAttributeValue** ⌈The HwAttributeValue is used to specify one value for a predefined attribute. The link of the attribute is defined with the reference to HwAttributeDef in the role hwAttributeDef which is subject to variant handling.⌋

The definition of attributes is described in section 2.6.

**[TPS_ECUR_01003] Values of hardware attributes** ⌈The actual value of a HwAttributeValue can be provided in one of two ways:

- **vt** - the value is specified in a textual representation.
- **v** - the value is specified in a numerical representation. The actual value can be subject to variant handling.

⌋

For more details see section Ecu Resource Variant Handling.

| *Class* | **HwAttributeValue** | | | |
|---|---|---|---|---|
| ***Note*** | This metaclass represents the ability to assign a hardware attribute value. Note that v and vt are mutually exclusive. | | | |
| ***Base*** | *ARObject* | | | |
| ***Aggregated by*** | *HwDescriptionEntity*.hwAttributeValue | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| annotation | Annotation | 0..1 | aggr | Optional annotation that can be added to each Hw AttributeValue.<br>**Tags:** atp.Splitkey=annotation |
| hwAttributeDef | HwAttributeDef | 0..1 | ref | This association represents the definition of the particular hardware attribute value.<br>**Tags:** atp.Splitkey=hwAttributeDef |
| v | Numerical | 0..1 | attr | This represents a numerical hardware attribute value.<br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=systemDesignTime |
| vt | VerbatimString | 0..1 | attr | This represents a textual hardware attribute value. |

**Table 2.2: HwAttributeValue**

**[constr_11001] Multiplicity of HwAttributeValue.hwAttributeDef** ⌈For each HwAttributeValue the reference in the role hwAttributeDef shall exist.⌋

## 2.2 Hardware Type

**[TPS_ECUR_01016] Definition of HwType** ⌈The HwType is used to gather attribute values for elements which can occur several times in an Ecu and will not change due to their multiple usage.⌋

For details on the multiple occurrence of hardware elements please refer to section 2.3.1.

A HwType is an ARElement which inherits from HwDescriptionEntity. The features of ARElement allow the hardware type to have a name and stand for its own inside some package. The features of HwDescriptionEntity allow the hardware type to describe hardware categories and attribute values (see section 2.1).

**[TPS_ECUR_01017] Attribute values defined in the HwType are applicable for all occurrences of this HwType** ⌈The attribute values defined in the HwType are appli-

cable for all occurrences of this HwType, although it is possible to override the value in the HwElement.⌋

For more details see section Hardware Element.

**[constr_3511] HwType shall not have a reference to another HwType** ⌈A HwType (being a HwDescriptionEntity) shall not have a reference to another HwType in the role hwType. The definition of HwTypes is not hierarchical.⌋

The HwType does not specify any structural features of the hardware. The description of hardware pin groups, hardware pins and hardware connections is only possible at the hardware element level.

| *Class* | **HwType** | | | |
|---|---|---|---|---|
| ***Note*** | This represents the ability to describe Hardware types on an abstract level. The particular types of hardware are distinguished by the category. This category determines the applicable attributes. The possible categories and attributes are defined in HwCategory.<br>**Tags:** atp.recommendedPackage=HwTypes | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *HwDescriptionEntity*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Aggregated by*** | ARPackage.element | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table 2.3: HwType**

## 2.3 Hardware Element

**[TPS_ECUR_01005] The HwElement describes one piece of hardware** ⌈The HwElement describes how one piece of hardware - as a building block - is contributing to the overall circuit describing the ECU. It can be used to describe any hardware, independent of their granularity and scale. So an ECU can be described as a whole, the connected sensors and actuators, the built-in micro-controller and communication transceiver. But also the processing cores and the memory segments inside the micro-controller can be described.⌋

**[TPS_ECUR_01018] HwElement is self contained** ⌈Each HwElement can be described in a self contained way because the HwElement is an ARElement.⌋

Each HwElement inherits from HwDescriptionEntity and is therefore capable to describe a set of attributes (see section 2.1 for details).

**[TPS_ECUR_01019] HwElement can refer to a HwType** ⌈Each HwElement can optionally refer to a HwType element in the role hwType. In the HwType the attribute values, which are common for all occurrences of the hardware type, are described. In case the HwElement provides an attribute value which is also provided in the referenced HwType the attribute value from the HwElement takes precedence.⌋

The features of the nestedElement reference are specified in section 2.3.1.

The HwElement can describe several HwPinGroup elements which are contained in the role hwPinGroup (for details on the HwPinGroup refer to section 2.4).

The hardware element can describe several HwElementConnector elements which are contained in the role hwElementConnection (for details on the HwElement-Connector refer to section 2.5).

| ***Class*** | **HwElement** | | | |
|---|---|---|---|---|
| ***Note*** | This represents the ability to describe Hardware Elements on an instance level. The particular types of hardware are distinguished by the category. This category determines the applicable attributes. The possible categories and attributes are defined in HwCategory.<br>**Tags:** atp.recommendedPackage=HwElements | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *HwDescriptionEntity*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Aggregated by*** | ARPackage.element | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| hwElement Connection | HwElementConnector | * | aggr | This represents one particular connection between two hardware elements.<br>**Stereotypes:** atpVariation<br>**Tags:**<br>atp.Splitkey=hwElementConnection, hwElement Connection.variationPoint.shortLabel<br>vh.latestBindingTime=systemDesignTime<br>xml.sequenceOffset=110 |
| hwPinGroup | HwPinGroup | * | aggr | This aggregation is used to describe the connection facilities of a hardware element. Note that hardware element has no pins but only pingroups.<br>**Stereotypes:** atpVariation<br>**Tags:**<br>atp.Splitkey=hwPinGroup, hwPinGroup.shortName, hw PinGroup.variationPoint.shortLabel<br>vh.latestBindingTime=systemDesignTime<br>xml.sequenceOffset=90 |
| nestedElement | HwElement | * | ref | This association is used to establish hierarchies of hw elements. Note that one particular HwElement can be target of this association only once. I.e. multiple instantiation of the same HwElement is not supported (at any hierarchy level).<br>**Stereotypes:** atpVariation<br>**Tags:**<br>atp.Splitkey=nestedElement, nestedElement.hwElement, nestedElement.variationPoint.shortLabel<br>vh.latestBindingTime=systemDesignTime<br>xml.sequenceOffset=70 |

**Table 2.4: HwElement**

### 2.3.1 Multiple occurrence of Hardware Elements

**[TPS_ECUR_01020] Hierarchy of hardware** ⌈The hierarchy of hardware is described via referencing the contained hardware elements with the role nestedElement. The containment hierarchy of hardware elements is not represented as a hierarchical structure in the XML description but as linked list.⌋

This modeling allows the usage of different ARXML files for the description of the container hardware element and the nested hardware elements. E.g. the CPU is described by a Semiconductor-Vendor, the project specific usage of such a CPU is described by the ECU vendor.

**[constr_3512] No support of multiple instantiation** ⌈An essential constraint is that each HwElement can only be target of one nestedElement reference. This means that there is no concept of multiple instantiation of hardware elements. If the same hardware element shall be used several times (using the nestedElement reference) each occurrence has to have its own description. This is also true for nested elements of the referenced nested element.⌋

Thus the hardware element and all its structural features (hardware pin groups, hardware pins and hardware connections) need to be cloned. There is however the possibility to reference the same HwType from several HwElement clones.

## 2.4 Hardware Pin and Pin Group

The HwPinGroup allows to describe dedicated channels of connectivity for hardware elements. It can be used to describe grouped hardware ports like ADC and DIO. It can structure the port information hierarchically. At the detailed level it can be used to describe individual hardware pins.

Each HwPinGroup is Identifiable. A HwPinGroup can only exist inside a HwElement or another HwPinGroup.

Each HwPinGroup inherits from HwDescriptionEntity and is therefore capable to describe a set of attributes (see section 2.1 for details).

The content of the HwPinGroup is aggregated in the role hwPinGroupContent.

| ***Class*** | **HwPinGroup** | | | |
|---|---|---|---|---|
| ***Note*** | This meta-class represents the ability to describe groups of pins which are used to connect hardware elements. This group acts as a bundle of pins. Thereby they allow to describe high level connections. Pin groups can even be nested. | | | |
| ***Base*** | *ARObject*, *HwDescriptionEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Aggregated by*** | HwElement.hwPinGroup, HwPinGroupContent.hwPinGroup | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| hwPinGroup Content | HwPinGroupContent | 0..1 | aggr | This aggregation describes the contained pins/pin groups.<br>**Tags:** atp.Splitkey=hwPinGroupContent |

**Table 2.5: HwPinGroup**

The HwPinGroupContent can contain HwPinGroup and HwPin. The HwPinGroupContent is defined as «atpMixed» (see Generic Structure Template [2]). The elements contained in the HwPinGroupContent (HwPinGroup and HwPin) can occur in an arbitrary order and multiple times. This allows to describe the ordered occurrence of pins and pin groups within pin groups. One major use-case is to describe physical connectors and plugs with chambers and pins.

| *Class* | «atpMixed» **HwPinGroupContent** | | | |
|---|---|---|---|---|
| ***Note*** | This meta-class specifies a mixture of hwPins and hwPinGroups. | | | |
| ***Base*** | *ARObject* | | | |
| ***Aggregated by*** | HwPinGroup.hwPinGroupContent | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| hwPin | HwPin | 0..1 | aggr | This aggregation represents a hardware pin in a hardware pin group.<br>**Stereotypes:** atpVariation<br>**Tags:**<br>vh.latestBindingTime=systemDesignTime<br>xml.roleWrapperElement=false |
| hwPinGroup | HwPinGroup | 0..1 | aggr | This aggregation represents a nested hardware pin group.<br>**Stereotypes:** atpVariation<br>**Tags:**<br>vh.latestBindingTime=systemDesignTime<br>xml.roleWrapperElement=false |

**Table 2.6: HwPinGroupContent**

Each `HwPin` is `Identifiable`. A `HwPin` can only exist inside a `HwPinGroupContent` and therefore indirectly in a `HwPinGroup`.

Each `HwPin` inherits from `HwDescriptionEntity` and is therefore capable to describe a set of attributes (see section 2.1 for details).

| *Class* | **HwPin** | | | |
|---|---|---|---|---|
| ***Note*** | This meta-class represents the possibility to describe a hardware pin. | | | |
| ***Base*** | *ARObject*, *HwDescriptionEntity*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| ***Aggregated by*** | HwPinGroupContent.hwPin | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| functionName | String | * | attr | This attribute describes the function of the pin (e.g. CLK for Clock). |
| packagingPin Name | String | 0..1 | attr | This attribute contains the name of the pin according to the packaging of the hardware element (e.g. A03). |
| pinNumber | Integer | 0..1 | attr | This attribute contains the physical pin number. |

**Table 2.7: HwPin**

**[TPS_ECUR_01039] Semantics of `HwPin` attributes** ⌈The attributes `packagingPinName` and `functionName` shall be used to describe the semantics of the `HwPin`. The `shortName` of the `HwPin` does not carry any defined standardized semantics.⌋

Note: For new descriptions of `HwPin`s, usage of `packagingPinName` is preferred over `pinNumber`. The `pinNumber` can additionally be used to specify a numerical ordering of the pins.

## 2.5 Hardware Connection

Connections can be described on several levels in the ECU Resource Template. This allows the expression of details on the needed level of abstraction.

**[TPS_ECUR_01006] Connections between HwElements** ⌈The HwElement-Connector allows to describe the connection between two HwElements. It is not meant to describe the actual technical connectivity between the two hardware elements. It is used to describe the general connectivity between the hardware elements.⌋

| *Class* | **HwElementConnector** | | | |
|---|---|---|---|---|
| *Note* | This meta-class represents the ability to connect two hardware elements. The details of the connection can be refined by hwPinGroupConnection. | | | |
| *Base* | *ARObject*, *Describable* | | | |
| *Aggregated by* | HwElement.hwElementConnection | | | |
| *Attribute* | *Type* | *Mult.* | *Kind* | *Note* |
| hwElement | HwElement | * | ref | This association connects two hardware elements.<br>**Tags:** atp.Splitkey=hwElement |
| hwPin Connection | HwPinConnector | * | aggr | This represents one particular connection between two hardware pins. This connection shall be used if pin-to-pin-connection is to be described but no description of the connection between the hierarchical composition of HwPinGroups (using HwPinGroupConnector) is required.<br>**Stereotypes:** atpVariation<br>**Tags:**<br>atp.Splitkey=hwPinConnection, hwPin Connection.variationPoint.shortLabel<br>vh.latestBindingTime=systemDesignTime<br>xml.sequenceOffset=60 |
| hwPinGroup Connection | HwPinGroupConnector | * | aggr | This represents one particular connection between two hardware pin groups.<br>**Stereotypes:** atpVariation<br>**Tags:**<br>atp.Splitkey=hwPinGroupConnection, hwPinGroup Connection.variationPoint.shortLabel<br>vh.latestBindingTime=systemDesignTime<br>xml.sequenceOffset=50 |

**Table 2.8: HwElementConnector**

**[constr_11002] Multiplicity of HwElementConnector.hwElement** ⌈For each HwElementConnector there shall exist exactly 2 references in the role hwElement.⌋

The HwPinGroupConnector allows to describe the connection between two HwPin-Groups.

| *Class* | **HwPinGroupConnector** | | | |
|---|---|---|---|---|
| *Note* | This meta-class represents the ability to connect two pin groups. | | | |
| *Base* | *ARObject*, *Describable* | | | |
| *Aggregated by* | HwElementConnector.hwPinGroupConnection | | | |
| *Attribute* | *Type* | *Mult.* | *Kind* | *Note* |

▽

△

| *Class* | **HwPinGroupConnector** | | | |
|---|---|---|---|---|
| hwPin Connection | HwPinConnector | * | aggr | This represents one particular connection between two hardware pins. The connected pins shall match the connection provided by the parent hwPinGroup Connection.<br>**Stereotypes:** atpVariation<br>**Tags:** atp.Splitkey=hwPinConnection, hwPin Connection.variationPoint.shortLabel vh.latestBindingTime=systemDesignTime |
| hwPinGroup | HwPinGroup | * | ref | This association connects two hardware pin groups.<br>**Tags:** atp.Splitkey=hwPinGroup |

**Table 2.9: HwPinGroupConnector**

**[constr_11003] Multiplicity of `HwPinGroupConnector.hwPinGroup`** ⌈For each `HwPinGroupConnector` there shall exist exactly 2 references in the role `hwPinGroup`.⌋

The `HwPinConnector` allows to describe the connection between two `HwPin`s.

| *Class* | **HwPinConnector** | | | |
|---|---|---|---|---|
| ***Note*** | This meta-class represents the ability to connect two pins. | | | |
| ***Base*** | *ARObject*, *Describable* | | | |
| ***Aggregated by*** | HwElementConnector.hwPinConnection, HwPinGroupConnector.hwPinConnection | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| hwPin | HwPin | * | ref | This association connects two hardware pins.<br>**Tags:** atp.Splitkey=hwPin |

**Table 2.10: HwPinConnector**

**[constr_11004] Multiplicity of `HwPinConnector.hwPin`** ⌈For each `HwPinConnector` there shall exist exactly 2 references in the role `hwPin`.⌋

### 2.5.1 Scope of Connections

The hardware connections are part of a hardware element and connect the two artifacts via references to the description of the artifacts. In principle such references can refer to any hardware element and its features in the input information. But the scope of connections is restricted based on the containing hardware element of the hardware connection.

**[constr_3513] Scope of connections** ⌈Each hardware connection shall only connect features which both are in the hierarchical scope of the hardware element. The hierarchical scope encloses

- all features belonging to the hardware element containing the connection

- all features belonging to hardware elements which are referenced directly and indirectly in the `nestedElement` relation from the hardware element containing connection.

⌋

Especially it is allowed to specify connections in hardware elements which are in deeper hierarchical level and also connections which cross hierarchical levels.

In the example from figure A.1 the following connections are allowed:

- connections specified in the scope of hardware element "MyEcu"
  - all the shown connections can be specified on this level
  - even the connections inside another hierarchical hardware element (e.g. between "Pu1" and "Can") can be specified on this level
  - even the connections crossing hierarchical levels (e.g. between "Can" and "Trcv") can be specified on this level
- connections specified in the scope of hardware element "MicroController"
  - only the connections inside the hardware element "MicroController" (e.g. between "Pu1" and "Can") can be specified.

## 2.6 Hardware Category Definition

The definition of dedicated hardware types allows a flexible usage of the ECU Resource Template. Since the definition of hardware types and the applicable attributes is specified as an AUTOSAR XML file itself it can be updated and extended without the needs to update the AUTOSAR XML-Schema.

In figure 2.2 the relationship between the definition and the description of hardware is shown.

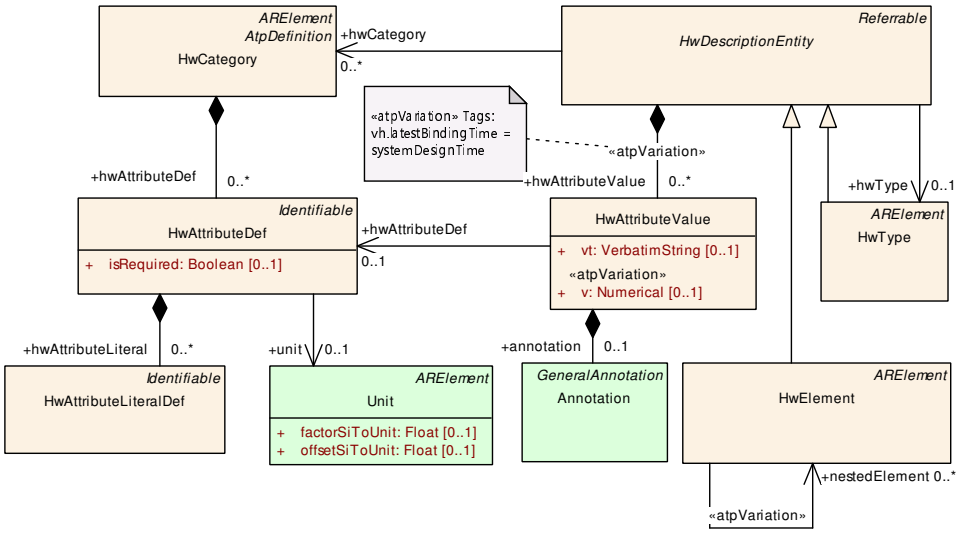



**Figure 2.2: Definition of hardware categories**

The element HwCategory specifies what type of hardware is defined. This can be a for example a memory segment, a processing unit, a communication transceiver etc.

The HwCategory is later referenced from the HwDescriptionEntity in the role hwCategory to describe what type of hardware is described. Possible values for the shortName of the HwCategory element are defined in [TPS_ECUR_01041] and [TPS_ECUR_01045].

The HwCategory may contain several HwAttributeDef elements.

| ***Class*** | **HwCategory** | | | |
|---|---|---|---|---|
| ***Note*** | This metaclass represents the ability to declare hardware categories and its particular attributes.<br>**Tags:** atp.recommendedPackage=HwCategorys | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpDefinition*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Aggregated by*** | ARPackage.element | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| hwAttributeDef | HwAttributeDef | * | aggr | This aggregation describes particular hardware attribute definition.<br>**Tags:** atp.Splitkey=hwAttributeDef, hwAttributeDef.shortName |

**Table 2.11: HwCategory**

The HwAttributeDef specifies one attribute which is applicable for the HwCategory.

The name of the attribute is defined in the shortName.

The type of the attribute is specified by the category. Applicable values for the category of HwAttributeDef are defined in [TPS_ECUR_01040].

**[constr_3500] category of HwAttributeDef shall not be extended** ⌈In contrast to the general rule that category can be extended by user-specific values it is **not al-**

**lowed** to extend the meaning of the attribute `category` of meta-class `HwAttributeDef`⌋

**[TPS_ECUR_01040] Standardized categories for `HwAttributeDef`** ⌈

| Category | Description |
|---|---|
| BOOLEAN | Defines a boolean attribute.<br>The values of a boolean attribute can be provided in<br>• textual format 'true' / 'false' (using the `vt` element of `HwAttributeValue`<br>• numerical format '1' (true) / '0' (false) (using the `v` element of `HwAttributeValue` |
| INTEGER | Defines an integer attribute.<br>The values of an integer attribute can be a signed / unsigned whole number. The value has to fit in a signed / unsigned 64-bit number space. |
| FLOAT | Defines a float attribute.<br>The value of a float attribute is represented as an IEEE double-precision 64-bit floating point of the IEEE 754-1985 standard [3]. |
| ENUMERATION | Defines an enumeration attribute. The possible enumeration literals are defined with the element `vt`.<br>The value of an enumeration attribute is provided as text in the `vt` element of `HwAttributeValue`. |
| STRING | Defines a string attribute.<br>The value of a string attribute is provided as text in the `vt` element of `HwAttributeValue`. |

⌋

The element `isRequired` specifies whether the attribute is mandatory for the defined `HwCategory`.

**[TPS_ECUR_01031] Definition of attribute unit** ⌈Optionally the attribute definition can have a reference to a `Unit` element which specifies in which unit the value of this attribute shall be specified.⌋

For details on the `Unit` specification please refer to the Software Component Template [4].

| Class | HwAttributeDef | | | |
|---|---|---|---|---|
| Note | This metaclass represents the ability to define a particular hardware attribute.<br>The category of this element defines the type of the attributeValue. If the category is Enumeration the hw AttributeEnumerationLiterals specify the available literals. | | | |
| Base | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| Aggregated by | HwCategory.hwAttributeDef | | | |
| **Attribute** | **Type** | **Mult.** | **Kind** | **Note** |
| hwAttribute Literal | HwAttributeLiteralDef | * | aggr | The available EnumerationLiterals of the Enumeration definition. Only applicable if the category of the Hw AttributeDef equals Enumeration.<br>**Tags:** atp.Splitkey=hwAttributeLiteral, hwAttribute Literal.shortName |
| isRequired | Boolean | 0..1 | attr | This attribute specifies if the defined attribute value is required to be provided. |
| unit | Unit | 0..1 | ref | This association specifies the physical unit of the defined hardware attribute. This is optional due to the fact that there are textual attributes.<br>**Tags:** atp.Splitkey=unit |

**Table 2.12: HwAttributeDef**

**[constr_11005] Multiplicity of `HwAttributeDef.isRequired`** ⌈For each `HwAttributeDef` the attribute `isRequired` shall exist.⌋

In case the `category` of the `HwAttributeDef` is set to `Enumeration` the applicable enumeration literals are specified with the element `HwAttributeLiteralDef`.

| Class | HwAttributeLiteralDef | | | |
|---|---|---|---|---|
| Note | One available EnumerationLiteral of the Enumeration definition. Only applicable if the category of the Hw AttributeDef equals Enumeration. | | | |
| Base | *ARObject*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| Aggregated by | HwAttributeDef.hwAttributeLiteral | | | |
| **Attribute** | **Type** | **Mult.** | **Kind** | **Note** |
| – | – | – | – | – |

**Table 2.13: HwAttributeLiteralDef**

In example A.8 the definition of some attributes for the `MemorySegment` `HwCategory` are described.

### 2.6.1 Vendor specific extensions of Hardware Category Definition

In order to allow the description of arbitrary hardware and their relationships the ECU Resource Template allows the extension of the definition of hardware categories and and hardware attributes. When extending the ECU Resource Description for vendor specific usage the following rules shall be respected:

- **[TPS_ECUR_01021] Definition of new hardware categories** ⌈New `HwCategories` for `HwElement` and `HwPinGroup` and `HwPin` can be defined if they are different from the already defined categories. This definition shall be in a pack-

age which is not the AUTOSAR package. A HwDescriptionEntity shall then reference the extended HwCategory.⌋

For more details on the already defined categories see chapter Hardware Type Specific Description.

- **[TPS_ECUR_01022] Extension of existing hardware categories** ⌈An existing HwCategory can be extended with new attribute definitions. The extension is via defining a HwCategory of the same name as the standardized one in a different package than AUTOSAR. A HwDescriptionEntity shall then reference the standardized and the extended HwCategory.⌋

  For more details on the already existing hardware categories see chapter Hardware Type Specific Description.

- **[TPS_ECUR_01023] No redefinition of hardware attributes** ⌈An extension of the standardized HwCategory shall not define the same hardware attributes as already defined in the standardized HwCategory.⌋

- **[TPS_ECUR_01024] Extension of enumeration** ⌈An existing enumeration attribute can be extended with new enumeration literals.⌋

  For more details on existing enumerations attributes see chapter Hardware Type Specific Description.

- **[TPS_ECUR_01025] No removal od existing enumeration literals** ⌈Enumeration literals shall not be removed from the specified enumeration attributes.⌋

  For more details on existing enumerations attributes see chapter Hardware Type Specific Description.

- **[TPS_ECUR_01026] No change of category** ⌈The category (type) of specified attributes shall not be changed.⌋

  For more details see chapter Hardware Type Specific Description.

- **[TPS_ECUR_01027] No change of `isRequired` value** ⌈The value of the isRequired element shall not be changed for specified attributes.⌋

  For more details see chapter Hardware Type Specific Description.

- **[TPS_ECUR_01028] No change of `Unit` value** ⌈The value of the Unit element shall not be changed for specified attributes.⌋

  For more details see chapter Hardware Type Specific Description.

## 2.7 Ecu Resource Variant Handling

For details on the AUTOSAR variant handling support please refer to the *AUTOSAR Generic Structure Template* [2]. The structure is shown in figure 2.1.

**[TPS_ECUR_01029] Support for variant handling** ⌈In the description of a hardware element the following relationships are subject to variant handling:

- `nestedElement`
- `hwPinGroup`
- `hwElementConnection`

⌋

The existence of a `HwPinGroup` can be variant via the aggregation role `hwPinGroup` from `HwElement`. So different alternatives of `HwPinGroup` can be specified. The content of the `HwPinGroup` can as well be variant via the roles `hwPinGroup` and `hwPin` from the `HwPinGroupContent`.

The existence of a `HwElementConnector` can be variant via the aggregation role `hwElementConnection` from `HwElement`. The existence of individual `HwPinGroupConnector`s and `HwPinConnector`s in several roles is as well subject to variability.

For the description of attribute values the existence of the `HwAttributeValue` and the actual `v` element are subject to variability (see also figure 2.2).

## 2.8 Documentation Support

AUTOSAR provides support for integrated and well structured documentation. More details about the AUTOSAR Documentation Support concept can be found in the AUTOSAR Generic Structure Template [2].

**[TPS_ECUR_01030] Documentation support** ⌈An optional documentation block can be applied to any `Identifiable` and `Describable` element in an Ecu Resource Description. This type of documentation is typically used to capture a short introduction about the role of an element or respectively how it is built.⌋

## 2.9 Infrastructural aspects

**[TPS_ECUR_01032] Modeling of ECU Resource metamodel** ⌈The modeling of the ECU Configuration Value and ECU Configuration Parameter Definition metamodels is done according to the Generic Structure Template [2].⌋

**[TPS_ECUR_01033] Transformation of the ECU Resource metamodel to schema definition** ⌈The transformation of the ECU Resource metamodel to schema definitions is done according to the XML Schema Production Rules [5].⌋

# 3 Hardware Type Specific Description

Chapter 2 introduced the general building blocks which are provided to describe hardware elements and their relationships. But in order to use the information from the ECU Resource Description to aid the configuration of an ECU there is need to describe dedicated attributes of specific hardware elements (e.g. memory size).

The following sections deal with the special elements that are necessary to specify a partly or complete engineered ECU with the ECU Resource Template.

## 3.1 HwElement categories

**[TPS_ECUR_01041] Standardized `HwCategories` for `HwElement`** ⌈

| HwCategory | Description |
|---|---|
| Ecu | Describes an Ecu |
| ProcessingUnit | Describes a micro-controller core |
| MicroController | Describes a micro-controller |
| MemorySegment | Describes a memory segment |
| CommunicationController | Describes a communication controller |
| CommunicationTransceiver | Describes a communication transceiver |
| Digital | Describes a digital IO peripheral |
| Analog | Describes an analog IO peripheral |
| Timer | Describes a timer peripheral |
| Watchdog | Describes a watchdog peripheral |
| SensorActuator | Describes sensors and actuators |

⌋

### 3.1.1 Ecu

**[TPS_ECUR_01034] `HwCategory` of an Ecu** ⌈The `HwCategory` of an ECU is defined as `Ecu`.⌋

Currently no special attributes are defined for the ECU.

There exists an inconsistency between the System Template and the ECU Resource Template concerning the usage of the term “Ecu”. In the System Template “Ecu” is used to determine one instance of an AUTOSAR Stack (e.g. like in ECUInstance). In the Ecu Resource Template ’Ecu" is used to describe the physical box (HardwareElement of `HwCategory` Ecu) containing the electronics which may contain several processing units with several AUTOSAR Stack instances running.

### 3.1.2 Processing Unit

The processing unit describes one core of a micro-controller.

**[TPS_ECUR_01007] HwCategory of a processing unit** ⌈The HwCategory of a processing unit is defined as ProcessingUnit.⌋

Currently no special attributes are defined for the processing unit.

### 3.1.3 Micro-Controller

The micro-controller describes one piece of hardware as delivered by the manufacturer of the micro-controller hardware. Typically the micro-controller contains one or several processing units, memory segments and peripherals.

**[TPS_ECUR_01035] HwCategory of a micro-controller** ⌈The HwCategory of a micro-controller is defined as MicroController.⌋

Currently no special attributes are defined for the micro-controller.

Example A.1 shows a simple description of a high-level view on a micro-controller.

### 3.1.4 Memory

**[TPS_ECUR_01042] HwCategory of a memory segment** ⌈The HwCategory of a memory segment is defined as MemorySegment.⌋

**[TPS_ECUR_01008] Standardized attributes for the HwCategory MemorySegment** ⌈

| Attribute | Required | Unit | Description |
|---|---|---|---|
| memorySize | true | INTEGER | Specifies the size of the memory segment in bytes. |
| memoryType | true | ENUMERATION | Specifies the type of memory:<br>• RAM<br>• ROM<br>• EEPROM<br>• Flash |

⌋

### 3.1.5 Communication Controller

**[TPS_ECUR_01009] HwCategory of a communication controller** ⌈The HwCategory of a communication controller is CommunicationController.⌋

**[TPS_ECUR_01043] Standardized attributes for the `HwCategory` `CommunicationController`** ⌈

| Attribute | Required | Unit | Description |
|---|---|---|---|
| communication-ControllerType | true | ENUMERATION | Specifies the type of communication controller:<br>• CAN<br>• LIN<br>• FlexRay<br>• Ethernet<br>• Spi |

⌋

### 3.1.6 Communication Transceiver

**[TPS_ECUR_01010] `HwCategory` of a communication transceiver** ⌈The `HwCategory` of a communication transceiver is defined as `CommunicationTransceiver`.⌋

**[TPS_ECUR_01044] Standardized attributes for the `HwCategory` `CommunicationTransceiver`** ⌈

| Attribute | Required | Unit | Description |
|---|---|---|---|
| supportsDisabling | false | BOOLEAN | Specifies whether the transceiver can be disabled |
| supportsWakeUp | false | BOOLEAN | Specifies whether the transceiver can indicate a wake-up situation on the bus |

⌋

### 3.1.7 Digital IO

**[TPS_ECUR_01011] `HwCategory` of a digital IO** ⌈The `HwCategory` of a digital IO hardware element is defined as `Digital`.⌋

Currently no special attributes are defined for the digital IO.

### 3.1.8 Analog IO

**[TPS_ECUR_01036] `HwCategory` of an analog IO** ⌈The `HwCategory` of an analog IO hardware element is defined as `Analog`.⌋

Currently no special attributes are defined for the analog IO.

### 3.1.9 Timer

**[TPS_ECUR_01037] `HwCategory` of a timer** ⌈The `HwCategory` of a timer is defined as `Timer`.⌋

Currently no special attributes are defined for the timer.

### 3.1.10 Watchdog

**[TPS_ECUR_01038] `HwCategory` of a watchdog** ⌈The `HwCategory` of a watchdog is defined as `Watchdog`.⌋

Currently no special attributes are defined for the watchdog.

### 3.1.11 SensorActuator

**[TPS_ECUR_01012] `HwCategory` of a sensor/actuator** ⌈The `HwCategory` of a sensor/actuator is defined as `SensorActuator`.⌋

Currently no special attributes are defined for the sensor/actuator.

## 3.2 HwPinGroup categories

**[TPS_ECUR_01045] Standardized `HwCategories` for `HwPinGroup`** ⌈

| HwCategory | Description |
|---|---|
| CommunicationPort | Describes a communication connector |

⌋

### 3.2.1 CommunicationPort

**[TPS_ECUR_01013] `HwCategory` of a Communication Port** ⌈The `HwCategory` of a Communication Port is defined as `CommunicationPort`.⌋

**[TPS_ECUR_01046] Standardized attributes for the HwCategory CommunicationPort** ⌈

| Attribute | Required | Unit | Description |
|---|---|---|---|
| communication-PortType | true | ENUMERATION | Specifies the type of communication port:<br>• CAN<br>• LIN<br>• FlexRay<br>• Ethernet<br>• Spi |

⌋

## 3.3 HwPin categories

There are no dedicated HwCategories specified for HwPin.

# A Examples

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

This chapter contains a collection of selected examples that reflect concepts described in different chapters of this document.

## A.1 Hardware Element

Example A.1 shows a simple description of a high-level view on a micro-controller.

```
<AR-PACKAGE>
  <SHORT-NAME>VendorA_Example1</SHORT-NAME>
  <ELEMENTS>
    <HW-ELEMENT>
      <SHORT-NAME>SomeMicroController</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/EcuR/MicroController</
            HW-CATEGORY-REF>
      </HW-CATEGORY-REFS>
    </HW-ELEMENT>
  </ELEMENTS>
</AR-PACKAGE>
```

**Listing A.1: Example for `HwElement`**

## A.2 Hierarchy of Hardware Elements

Example A.2 shows the hierarchical description of a processing unit in a micro-controller.

```
<AR-PACKAGE>
  <SHORT-NAME>VendorA_Example2</SHORT-NAME>
  <ELEMENTS>
    <HW-ELEMENT>
      <SHORT-NAME>SomeMicroController</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/EcuR/MicroController</
            HW-CATEGORY-REF>
      </HW-CATEGORY-REFS>
      <NESTED-ELEMENTS>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA_Example2/
              ProcessingUnit0</HW-ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
      </NESTED-ELEMENTS>
    </HW-ELEMENT>
    <HW-ELEMENT>
      <SHORT-NAME>ProcessingUnit0</SHORT-NAME>
      <HW-CATEGORY-REFS>
```

```
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/EcuR/ProcessingUnit</
            HW-CATEGORY-REF>
        </HW-CATEGORY-REFS>
      </HW-ELEMENT>
    </ELEMENTS>
</AR-PACKAGE>
```

**Listing A.2: Example for hierarchy of HwElements**

## A.3 HwPinGroups and HwPins

Example A.3 shows the description of pin groups and pins of the micro-controller.

```
<AR-PACKAGE>
  <SHORT-NAME>VendorA_Example3</SHORT-NAME>
  <ELEMENTS>
    <HW-ELEMENT>
      <SHORT-NAME>SomeMicroController</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/EcuR/MicroController</
            HW-CATEGORY-REF>
      </HW-CATEGORY-REFS>
      <HW-PIN-GROUPS>
        <HW-PIN-GROUP>
          <SHORT-NAME>Adc</SHORT-NAME>
          <HW-PIN-GROUP-CONTENT>
            <HW-PIN-GROUP>
              <SHORT-NAME>AdcPortA</SHORT-NAME>
            </HW-PIN-GROUP>
            <HW-PIN-GROUP>
              <SHORT-NAME>AdcPortB</SHORT-NAME>
              <HW-PIN-GROUP-CONTENT>
                <HW-PIN>
                  <SHORT-NAME>AdcB01</SHORT-NAME>
                </HW-PIN>
                <HW-PIN>
                  <SHORT-NAME>AdcB02</SHORT-NAME>
                </HW-PIN>
              </HW-PIN-GROUP-CONTENT>
            </HW-PIN-GROUP>
          </HW-PIN-GROUP-CONTENT>
        </HW-PIN-GROUP>
      </HW-PIN-GROUPS>
    </HW-ELEMENT>
  </ELEMENTS>
</AR-PACKAGE>
```

**Listing A.3: Example for HwPinGroup and HwPin**

## A.4 Hardware Element Connection

Example A.4 shows the description of the internal structure of a micro-controller in order to define which memory segments are accessible from which processing unit (core).

```
<AR-PACKAGE>
  <SHORT-NAME>VendorA_Example4</SHORT-NAME>
  <ELEMENTS>
    <HW-ELEMENT>
      <SHORT-NAME>SomeMicroController</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/EcuR/MicroController</
            HW-CATEGORY-REF>
        </HW-CATEGORY-REFS>
      <NESTED-ELEMENTS>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA_Example4/Core0</HW-
              ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA_Example4/Core1</HW-
              ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA_Example4/Mem01</HW-
              ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
        <!-- ... -->
      </NESTED-ELEMENTS>
      <HW-ELEMENT-CONNECTIONS>
        <HW-ELEMENT-CONNECTOR>
          <HW-ELEMENT-REFS>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA_Example4/Core0</HW-
                ELEMENT-REF>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA_Example4/Mem01</HW-
                ELEMENT-REF>
          </HW-ELEMENT-REFS>
        </HW-ELEMENT-CONNECTOR>
        <HW-ELEMENT-CONNECTOR>
          <HW-ELEMENT-REFS>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA_Example4/Core0</HW-
                ELEMENT-REF>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA_Example4/Mem02</HW-
                ELEMENT-REF>
          </HW-ELEMENT-REFS>
        </HW-ELEMENT-CONNECTOR>
        <!-- .. -->
      </HW-ELEMENT-CONNECTIONS>
    </HW-ELEMENT>
    <HW-ELEMENT>
      <SHORT-NAME>Core0</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/EcuR/ProcessingUnit</
            HW-CATEGORY-REF>
      </HW-CATEGORY-REFS>
```

```
    </HW-ELEMENT>
    <HW-ELEMENT>
      <SHORT-NAME>Core1</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/EcuR/ProcessingUnit</
            HW-CATEGORY-REF>
      </HW-CATEGORY-REFS>
    </HW-ELEMENT>
    <HW-ELEMENT>
      <SHORT-NAME>Mem01</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/EcuR/MemorySegment</HW
            -CATEGORY-REF>
      </HW-CATEGORY-REFS>
    </HW-ELEMENT>
    <HW-ELEMENT>
      <SHORT-NAME>Mem02</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/EcuR/MemorySegment</HW
            -CATEGORY-REF>
      </HW-CATEGORY-REFS>
    </HW-ELEMENT>
    <HW-ELEMENT>
      <SHORT-NAME>Mem03</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/EcuR/MemorySegment</HW
            -CATEGORY-REF>
      </HW-CATEGORY-REFS>
    </HW-ELEMENT>
  </ELEMENTS>
</AR-PACKAGE>
```

**Listing A.4: Example for `HwElementConnector`**

## A.5 Combined Example

In this example section several mechanisms are utilized to describe an Ecu and some of its electronics attributes. The overview is shown in figure A.1. The individual sections describe the different abstraction layers.

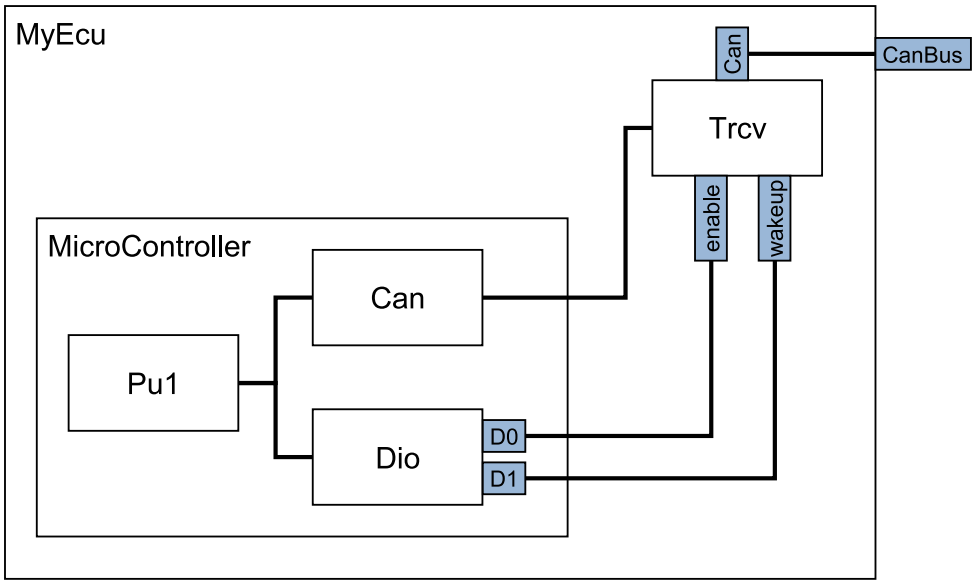



**Figure A.1: Example of Ecu description**

### A.5.1 Micro-controller description

The micro-controller consists of the processing unit, a Can controller and a Dio module. The processing unit is defined to have access to both of the peripherals.

The Dio module defines two `HwPinGroup`s in order to support more detailed connection description.

The whole micro-controller is defined in an own `ARPackage` so it can be used in several projects.

```
<AR-PACKAGE>
  <SHORT-NAME>CpuVendor</SHORT-NAME>
  <ELEMENTS>
    <HW-ELEMENT>
      <SHORT-NAME>MicroController</SHORT-NAME>
      <NESTED-ELEMENTS>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/CpuVendor/Pu1</HW-ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/CpuVendor/Can</HW-ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/CpuVendor/Dio</HW-ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
      </NESTED-ELEMENTS>
      <HW-ELEMENT-CONNECTIONS>
        <HW-ELEMENT-CONNECTOR>
          <HW-ELEMENT-REFS>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/CpuVendor/Pu1</HW-ELEMENT-
              REF>
```

```
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/CpuVendor/Can</HW-ELEMENT-
                REF>
          </HW-ELEMENT-REFS>
        </HW-ELEMENT-CONNECTOR>
        <HW-ELEMENT-CONNECTOR>
          <HW-ELEMENT-REFS>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/CpuVendor/Pu1</HW-ELEMENT-
                REF>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/CpuVendor/Dio</HW-ELEMENT-
                REF>
          </HW-ELEMENT-REFS>
        </HW-ELEMENT-CONNECTOR>
      </HW-ELEMENT-CONNECTIONS>
    </HW-ELEMENT>
    <HW-ELEMENT>
      <SHORT-NAME>Pu1</SHORT-NAME>
    </HW-ELEMENT>
    <HW-ELEMENT>
      <SHORT-NAME>Can</SHORT-NAME>
    </HW-ELEMENT>
    <HW-ELEMENT>
      <SHORT-NAME>Dio</SHORT-NAME>
      <HW-PIN-GROUPS>
        <HW-PIN-GROUP>
          <SHORT-NAME>D0</SHORT-NAME>
        </HW-PIN-GROUP>
        <HW-PIN-GROUP>
          <SHORT-NAME>D1</SHORT-NAME>
        </HW-PIN-GROUP>
      </HW-PIN-GROUPS>
    </HW-ELEMENT>
  </ELEMENTS>
</AR-PACKAGE>
```

**Listing A.5: Example for Ecu dscription**

### A.5.2 Transceiver description

The transceiver module is defined as a `HwElement` which provides three `HwPinGroup`s to describe its connectivity.

The transceiver module is defined in an own `ARPackage` so it can be used in several projects.

```
<AR-PACKAGE>
  <SHORT-NAME>TransceiverVendor</SHORT-NAME>
  <ELEMENTS>
    <HW-ELEMENT>
      <SHORT-NAME>Trcv</SHORT-NAME>
      <HW-PIN-GROUPS>
        <HW-PIN-GROUP>
          <SHORT-NAME>enable</SHORT-NAME>
        </HW-PIN-GROUP>
        <HW-PIN-GROUP>
```

```
        <SHORT-NAME>wakeup</SHORT-NAME>
      </HW-PIN-GROUP>
      <HW-PIN-GROUP>
        <SHORT-NAME>CanBus</SHORT-NAME>
      </HW-PIN-GROUP>
    </HW-PIN-GROUPS>
  </HW-ELEMENT>
 </ELEMENTS>
</AR-PACKAGE>
```

**Listing A.6: Example for Transceiver description**

### A.5.3 Ecu description

The Ecu contains the micro-controller and the transceiver.

The Ecu defines one HwPinGroup to represent the CanBus communication to the outside of the Ecu.

The Ecu defines the detailed connectivity inside.

```
<AR-PACKAGE>
  <SHORT-NAME>EcuVendor</SHORT-NAME>
  <ELEMENTS>
    <HW-ELEMENT>
      <SHORT-NAME>MyEcu</SHORT-NAME>
      <NESTED-ELEMENTS>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/CpuVendor/MicroController</HW-
              ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/TransceiverVendor/Trcv</HW-
              ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
      </NESTED-ELEMENTS>
      <HW-PIN-GROUPS>
        <HW-PIN-GROUP>
          <SHORT-NAME>CanBus</SHORT-NAME>
        </HW-PIN-GROUP>
      </HW-PIN-GROUPS>
      <HW-ELEMENT-CONNECTIONS>
        <HW-ELEMENT-CONNECTOR>
          <HW-ELEMENT-REFS>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/CpuVendor/Can</HW-ELEMENT-
                REF>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/TransceiverVendor/Trcv</HW-
                ELEMENT-REF>
          </HW-ELEMENT-REFS>
        </HW-ELEMENT-CONNECTOR>
        <HW-ELEMENT-CONNECTOR>
          <HW-ELEMENT-REFS>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/CpuVendor/Dio</HW-ELEMENT-
                REF>
```

```
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/TransceiverVendor/Trcv</HW-
                ELEMENT-REF>
          </HW-ELEMENT-REFS>
          <HW-PIN-GROUP-CONNECTIONS>
            <HW-PIN-GROUP-CONNECTOR>
              <HW-PIN-GROUP-REFS>
                <HW-PIN-GROUP-REF DEST="HW-PIN-GROUP">/CpuVendor/Dio/D0</HW
                    -PIN-GROUP-REF>
                <HW-PIN-GROUP-REF DEST="HW-PIN-GROUP">/TransceiverVendor/
                    Trcv/enable</HW-PIN-GROUP-REF>
              </HW-PIN-GROUP-REFS>
            </HW-PIN-GROUP-CONNECTOR>
            <HW-PIN-GROUP-CONNECTOR>
              <HW-PIN-GROUP-REFS>
                <HW-PIN-GROUP-REF DEST="HW-PIN-GROUP">/CpuVendor/Dio/D1</HW
                    -PIN-GROUP-REF>
                <HW-PIN-GROUP-REF DEST="HW-PIN-GROUP">/TransceiverVendor/
                    Trcv/wakeup</HW-PIN-GROUP-REF>
              </HW-PIN-GROUP-REFS>
            </HW-PIN-GROUP-CONNECTOR>
          </HW-PIN-GROUP-CONNECTIONS>
        </HW-ELEMENT-CONNECTOR>
        <HW-ELEMENT-CONNECTOR>
          <HW-ELEMENT-REFS>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/TransceiverVendor/Trcv</HW-
                ELEMENT-REF>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/EcuVendor/MyEcu</HW-ELEMENT-
                REF>
          </HW-ELEMENT-REFS>
          <HW-PIN-GROUP-CONNECTIONS>
            <HW-PIN-GROUP-CONNECTOR>
              <HW-PIN-GROUP-REFS>
                <HW-PIN-GROUP-REF DEST="HW-PIN-GROUP">/TransceiverVendor/
                    Trcv/CanBus</HW-PIN-GROUP-REF>
                <HW-PIN-GROUP-REF DEST="HW-PIN-GROUP">/EcuVendor/MyEcu/
                    CanBus</HW-PIN-GROUP-REF>
              </HW-PIN-GROUP-REFS>
            </HW-PIN-GROUP-CONNECTOR>
          </HW-PIN-GROUP-CONNECTIONS>
        </HW-ELEMENT-CONNECTOR>
      </HW-ELEMENT-CONNECTIONS>
    </HW-ELEMENT>
  </ELEMENTS>
</AR-PACKAGE>
```

**Listing A.7: Example for Ecu description**

## A.6 Attribute Definition

Example A.8 shows how a category and associated attribute definitions are described in the ECU Resource Template.

```
    <AR-PACKAGE>
      <SHORT-NAME>AUTOSAR</SHORT-NAME>
```

```
<AR-PACKAGES>
  <AR-PACKAGE>
    <SHORT-NAME>EcuR</SHORT-NAME>
    <ELEMENTS>
      <HW-CATEGORY>
        <SHORT-NAME>MemorySegment</SHORT-NAME>
        <HW-ATTRIBUTE-DEFS>
          <HW-ATTRIBUTE-DEF>
            <SHORT-NAME>memorySize</SHORT-NAME>
            <DESC>
              <L-2 L="EN">Specifies the size of the memory segment in
                  bytes.</L-2>
            </DESC>
            <CATEGORY>INTEGER</CATEGORY>
            <IS-REQUIRED>true</IS-REQUIRED>
          </HW-ATTRIBUTE-DEF>
          <HW-ATTRIBUTE-DEF>
            <SHORT-NAME>memoryType</SHORT-NAME>
            <DESC>
              <L-2 L="EN">Specifies the type of memory: RAM, ROM,
                 EEPROM, Flash.</L-2>
            </DESC>
            <CATEGORY>ENUMERATION</CATEGORY>
              <HW-ATTRIBUTE-LITERALS>
                <HW-ATTRIBUTE-LITERAL-DEF>
                  <SHORT-NAME>RAM</SHORT-NAME>
                </HW-ATTRIBUTE-LITERAL-DEF>
                <HW-ATTRIBUTE-LITERAL-DEF>
                  <SHORT-NAME>ROM</SHORT-NAME>
                </HW-ATTRIBUTE-LITERAL-DEF>
                <HW-ATTRIBUTE-LITERAL-DEF>
                  <SHORT-NAME>FLASH</SHORT-NAME>
                </HW-ATTRIBUTE-LITERAL-DEF>
                <HW-ATTRIBUTE-LITERAL-DEF>
                  <SHORT-NAME>EEPROM</SHORT-NAME>
                </HW-ATTRIBUTE-LITERAL-DEF>
              </HW-ATTRIBUTE-LITERALS>
              <IS-REQUIRED>true</IS-REQUIRED>
          </HW-ATTRIBUTE-DEF>
        </HW-ATTRIBUTE-DEFS>
      </HW-CATEGORY>
    </ELEMENTS>
  </AR-PACKAGE>
</AR-PACKAGES>
</AR-PACKAGE>
```

**Listing A.8: Example for attribute definition**

## A.7 Attribute Value Example

Example A.9 shows the description of attributes which have been defined using the ECU Resource Template (see example A.8).

```
<AR-PACKAGE>
```

```
  <SHORT-NAME>VendorA_Example5</SHORT-NAME>
  <ELEMENTS>
    <HW-ELEMENT>
      <SHORT-NAME>MemorySeg001</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/EcuR/MemorySegment</HW
            -CATEGORY-REF>
      </HW-CATEGORY-REFS>
      <HW-ATTRIBUTE-VALUES>
        <HW-ATTRIBUTE-VALUE>
          <HW-ATTRIBUTE-DEF-REF DEST="HW-ATTRIBUTE-DEF">/AUTOSAR/EcuR/
              MemorySegment/memoryType</HW-ATTRIBUTE-DEF-REF>
          <VT>RAM</VT>
        </HW-ATTRIBUTE-VALUE>
        <HW-ATTRIBUTE-VALUE>
          <HW-ATTRIBUTE-DEF-REF DEST="HW-ATTRIBUTE-DEF">/AUTOSAR/EcuR/
              MemorySegment/memorySize</HW-ATTRIBUTE-DEF-REF>
          <V>1024</V>
        </HW-ATTRIBUTE-VALUE>
      </HW-ATTRIBUTE-VALUES>
    </HW-ELEMENT>
  </ELEMENTS>
</AR-PACKAGE>
```

**Listing A.9: Example for attribute value**

# B Glossary

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

**Artifact** This is a Work Product Definition that provides a description and definition for tangible work product types. Artifacts may be composed of other artifacts ([6]).

At a high level, an artifact is represented as a single conceptual file.

**AUTOSAR Tool** This is a software tool which supports one or more tasks defined as AUTOSAR tasks in the methodology. Depending on the supported tasks, an AUTOSAR tool can act as an authoring tool, a converter tool, a processor tool or as a combination of those (see separate definitions).

**AUTOSAR Authoring Tool** An AUTOSAR Tool used to create and modify AUTOSAR XML Descriptions. Example: System Description Editor.

**AUTOSAR Converter Tool** An AUTOSAR Tool used to create AUTOSAR XML files by converting information from other AUTOSAR XML files. Example: ECU Flattener

**AUTOSAR Definition** This is the definition of parameters which can have values. One could say that the parameter values are Instances of the definitions. But in the meta model hierarchy of AUTOSAR, definitions are also instances of the meta model and therefore considered as a description. Examples for AUTOSAR definitions are: `PortPrototype`, `PostBuildVariantCriterion`, `SwSystemconst`.

**AUTOSAR XML Description** In AUTOSAR this means “filled Template”. In fact an AUTOSAR XML description is the XML representation of an AUTOSAR model.

The AUTOSAR XML description can consist of several files. Each individual file represents an AUTOSAR partial model and shall validate successfully against the AUTOSAR XML schema.

**AUTOSAR Meta-Model** This is an UML2.0 model that defines the language for describing AUTOSAR systems. The AUTOSAR meta-model is an UML representation of the AUTOSAR templates. UML2.0 class diagrams are used to describe the attributes and their interrelationships. Stereotypes, UML tags and OCL expressions (object constraint language) are used for defining specific semantics and constraints.

**AUTOSAR Meta-Model Tool** The AUTOSAR Meta-Model Tool is the tool that generates different views (class tables, list of constraints, diagrams, XML Schema etc.) on the AUTOSAR meta-model.

**AUTOSAR Model** This is a representation of an AUTOSAR product. The AUTOSAR model represents aspects suitable to the intended use according to the AUTOSAR methodology.

Strictly speaking, this is an instance of the AUTOSAR meta-model. The information contained in the AUTOSAR model can be anything that is representable according to the AUTOSAR meta-model.

**AUTOSAR Partial Model** In AUTOSAR, the possible partitioning of models is marked in the meta-model by ≪atpSplitable≫. One partial model is represented in an AUTOSAR XML description by one file. The partial model does not need to fulfill all semantic constraints applicable to an AUTOSAR model.

**AUTOSAR Processor Tool** An AUTOSAR Tool used to create non-AUTOSAR files by processing information from AUTOSAR XML files. Example: RTE Generator

**AUTOSAR Specification Element** An AUTOSAR Specification Element is a named element that is part of an AUTOSAR specification. Examples: requirement, constraint, specification item, class or attribute in the meta model, methodology, deliverable, methodology activity, model element, bsw module etc.

**AUTOSAR Template** The term "Template" is used in AUTOSAR to describe the format different kinds of descriptions. The term template comes from the idea, that AUTOSAR defines a kind of form which shall be filled out in order to describe a model. The filled form is then called the description.

In fact the AUTOSAR templates are now defined as a meta-model.

**AUTOSAR Validation Tool** A specialized AUTOSAR Tool which is able to check an AUTOSAR model against the rules defined by a profile.

**AUTOSAR XML Schema** This is a W3C XML schema that defines the language for exchanging AUTOSAR models. This Schema is derived from the AUTOSAR meta-model. The AUTOSAR XML Schema defines the AUTOSAR data exchange format.

**Blueprint** This is a model from which other models can be derived by copy and refinement. Note that in contrast to meta model resp. types, this process is *not* an instantiation.

**Instance** Generally this is a particular exemplar of a model or of a type.

**Life Cycle** Life Cycle is the course of development/evolutionary stages of a model element during its life time.

**Meta-Model** This defines the building blocks of a model. In that sense, a Meta-Model represents the language for building models.

**Meta-Data** This includes pertinent information about data, including information about the authorship, versioning, access-rights, timestamps etc.

**Model** A Model is an simplified representation of reality. The model represents the aspects suitable for an intended purpose.

**Partial Model** This is a part of a model which is intended to be persisted in one particular artifact.

**Pattern in GST** This is an approach to simplify the definition of the meta model by applying a model transformation. This transformation creates an enhanced model out of an annotated model.

**Property** A property is a structural feature of an object. As an example a “connector” has the properties “receive port” and “send port”.

Properties are made variant by the ≪atpVariation≫.

**Prototype** This is the implementation of a role of a type within the definition of another type. In other words a type may contain Prototypes that in turn are typed by “Types”. Each one of these prototypes becomes an instance when this type is instantiated.

**Type** A type provides features that can appear in various roles of this type.

**Value** This is a particular value assigned to a “Definition”.

**Variability** Variability of a system is its quality to describe a set of variants. These variants are characterized by variant specific property settings and / or selections. As an example, such a system property selection manifests itself in a particular “receive port” for a connection.

This is implemented using the ≪atpVariation≫.

**Variant** A system variant is a concrete realization of a system, so that all its properties have been set respectively selected. The software system has no variability anymore with respect to the binding time.

This is implemented using EvaluatedVariantSet.

**Variation Binding** A variant is the result of a variation binding process that resolves the variability of the system by assigning particular values/selections to all the system’s properties.

This is implemented by VariationPoint.

**Variation Binding Time** The variation binding time determines the step in the methodology at which the variability given by a set of variable properties is resolved.

This is implemented by vh.latestBindingTime at the related properties.

**Variation Definition Time** The variation definition time determines the step in the methodology at which the variation points are defined.

**Variation Point** A variation point indicates that a property is subject to variation. Furthermore, it is associated with a condition and a binding time which define the system context for the selection / setting of a concrete variant.

This is implemented by VariationPoint.

# C History of Constraints and Specification Items

The content of this appendix chapter is *informative* in nature and shall **not** be considered as *normative* content.

This chapter provides the change history of traceable items in this document. The lists also include traceable items that have been removed from the document in a later version. These items do not appear as hyperlinks in the document.

## C.1 Constraint and Specification Item History of this document according to AUTOSAR Release R4.0.2

No changes.

## C.2 Constraint and Specification Item History of this document according to AUTOSAR Release R4.0.3

### C.2.1 Added Constraints in R4.0.3

| Number | Heading |
|---|---|
| [constr_3500] | `category` of `HwAttributeDef` shall not be extended |

**Table C.1: Added Constraints in R4.0.3**

## C.3 Constraint and Specification Item History of this document according to AUTOSAR Release R4.1.1

### C.3.1 Added Constraints in R4.1.1

| Number | Heading |
|---|---|
| [constr_3511] | `HwType` shall not have a reference to another `HwType` |
| [constr_3512] | No support of multiple instantiation |
| [constr_3513] | Scope of connections |

**Table C.2: Added Constraints in R4.1.1**

### C.3.2 Added Traceables in R4.1.1

| *SWS Item* | *Rationale* |
|---|---|
| [TPS_ECUR_01000] | Definition of `HwCategory` |

| | |
|---|---|
| [TPS_ECUR_01001] | Extension of `HwCategory` |
| [TPS_ECUR_01002] | Definition of Hardware Elements |
| [TPS_ECUR_01003] | Values of hardware attributes |
| [TPS_ECUR_01005] | The `HwElement` describes one piece of hardware |
| [TPS_ECUR_01006] | Connections between `HwElement`s |
| [TPS_ECUR_01007] | The category of a processing unit is defined as `ProcessingUnit`. |
| [TPS_ECUR_01008] | The special attributes which are applicable for the category `MemorySegment` hardware elements are defined in [TPS_ECUR_01008]. |
| [TPS_ECUR_01009] | The category of a communication controller is `CommunicationController`. |
| [TPS_ECUR_01010] | The category of a communication transceiver is defined as `CommunicationTransceiver`. |
| [TPS_ECUR_01011] | The category of a digital IO hardware element is defined as `Digital`. |
| [TPS_ECUR_01012] | The category of a sensor/actuator is defined as `SensorActuator`. |
| [TPS_ECUR_01013] | The category of a Communication Port is defined as `CommunicationPort`. |
| [TPS_ECUR_01014] | Definition of `HwAttributeValue` |
| [TPS_ECUR_01015] | Support of AUTOSAR Basic Software configuration |
| [TPS_ECUR_01016] | Definition of `HwType` |
| [TPS_ECUR_01017] | Attribute values defined in the `HwType` are applicable for all occurrences of this `HwType` |
| [TPS_ECUR_01018] | `HwElement` is self contained |
| [TPS_ECUR_01019] | `HwElement` can refer to a `HwType` |
| [TPS_ECUR_01020] | Hierarchy of hardware |
| [TPS_ECUR_01021] | Definition of new hardware categories |
| [TPS_ECUR_01022] | Extension of existing hardware categories |
| [TPS_ECUR_01023] | No redefinition of hardware attributes |
| [TPS_ECUR_01024] | Extension of enumeration |
| [TPS_ECUR_01025] | No removal od existing enumeration literals |
| [TPS_ECUR_01026] | No change of category |
| [TPS_ECUR_01027] | No change of `isRequired` value |
| [TPS_ECUR_01028] | No change of `Unit` value |

| | |
|---|---|
| [TPS_ECUR_01029] | Support for variant handling |
| [TPS_ECUR_01030] | Documentation support |
| [TPS_ECUR_01031] | Definition of attribute unit |
| [TPS_ECUR_01032] | Modeling of ECU Resource metamodel |
| [TPS_ECUR_01033] | Transformation of the ECU Resource metamodel to schema definition |
| [TPS_ECUR_01034] | Category of an Ecu |
| [TPS_ECUR_01035] | Category of a micro-controller |
| [TPS_ECUR_01036] | Category of an analog IO |
| [TPS_ECUR_01037] | Category of a timer |
| [TPS_ECUR_01038] | Category of a watchdog |

**Table C.3: Added Traceables in R4.1.1**

## C.4 Constraint and Specification Item History of this document according to AUTOSAR Release R4.1.2

No changes.

## C.5 Constraint and Specification Item History of this document according to AUTOSAR Release R4.1.3

No changes.

## C.6 Constraint and Specification Item History of this document according to AUTOSAR Release R4.2.1

No changes.

## C.7 Constraint and Specification Item History of this document according to AUTOSAR Release R4.2.2

No changes.

## C.8 Constraint and Specification Item History of this document according to AUTOSAR Release R4.3.0

No changes.

## C.9 Constraint and Specification Item History of this document according to AUTOSAR Release R4.3.1

No changes.

## C.10 Constraint and Specification Item History of this document according to AUTOSAR Release R4.4.0

No changes.

## C.11 Constraint and Specification Item History of this document according to AUTOSAR Release R19-11

No changes.

## C.12 Constraint and Specification Item History of this document according to AUTOSAR Release R20-11

No changes.

## C.13 Constraint and Specification Item History of this document according to AUTOSAR Release R21-11

### C.13.1 Added Specification Items in R21-11

none

### C.13.2 Changed Specification Items in R21-11

| Number | Heading |
|---|---|
| [TPS_ECUR_01014] | Definition of HwAttributeValue |
| [TPS_ECUR_01021] | Definition of new hardware categories |
| [TPS_ECUR_01022] | Extension of existing hardware categories |

**Table C.4: Changed Specification Items in R21-11**

### C.13.3 Deleted Specification Items in R21-11

none

### C.13.4 Added Constraints in R21-11

| Number | Heading |
|---|---|
| [constr_11001] | Multiplicity of HwAttributeValue.hwAttributeDef |
| [constr_11002] | Multiplicity of HwElementConnector.hwElement |
| [constr_11003] | Multiplicity of HwPinGroupConnector.hwPinGroup |
| [constr_11004] | Multiplicity of HwPinConnector.hwPin |
| [constr_11005] | Multiplicity of HwAttributeDef.isRequired |

**Table C.5: Added Constraints in R21-11**

### C.13.5 Changed Constraints in R21-11

none

### C.13.6 Deleted Constraints in R21-11

none

## C.14 Constraint and Specification Item History of this document according to AUTOSAR Release R22-11

No changes.

# C.15 Constraint and Specification Item History of this document according to AUTOSAR Release R23-11

## C.15.1 Added Specification Items in R23-11

| Number | Heading |
|---|---|
| [TPS_ECUR_01039] | Semantics of `HwPin` attributes |
| [TPS_ECUR_01040] | Standardized categories for `HwAttributeDef` |
| [TPS_ECUR_01041] | Standardized `HwCategories` for `HwElement` |
| [TPS_ECUR_01042] | `HwCategory` of a memory segment |
| [TPS_ECUR_01043] | Standardized attributes for the `HwCategory CommunicationController` |
| [TPS_ECUR_01044] | Standardized attributes for the `HwCategory CommunicationTransceiver` |
| [TPS_ECUR_01045] | Standardized `HwCategories` for `HwPinGroup` |
| [TPS_ECUR_01046] | Standardized attributes for the `HwCategory CommunicationPort` |

**Table C.6: Added Specification Items in R23-11**

## C.15.2 Changed Specification Items in R23-11

| Number | Heading |
|---|---|
| [TPS_ECUR_01008] | Standardized attributes for the `HwCategory MemorySegment` |

**Table C.7: Changed Specification Items in R23-11**

## C.15.3 Deleted Specification Items in R23-11

none

## C.15.4 Added Constraints in R23-11

none

## C.15.5 Changed Constraints in R23-11

none

## C.15.6 Deleted Constraints in R23-11

none

## C.16 Constraint and Specification Item History of this document according to AUTOSAR Release R24-11

### C.16.1 Added Specification Items in R24-11

none

### C.16.2 Changed Specification Items in R24-11

none

### C.16.3 Deleted Specification Items in R24-11

none

### C.16.4 Added Constraints in R24-11

none

### C.16.5 Changed Constraints in R24-11

none

### C.16.6 Deleted Constraints in R24-11

none

## C.17 Constraint and Specification Item History of this document according to AUTOSAR Release R25-11

### C.17.1 Added Specification Items in R25-11

none

### C.17.2 Changed Specification Items in R25-11

| Number | Heading |
|---|---|
| [TPS_ECUR_01043] | Standardized attributes for the `HwCategory CommunicationController` |
| [TPS_ECUR_01046] | Standardized attributes for the `HwCategory CommunicationPort` |

**Table C.8: Changed Specification Items in R25-11**

### C.17.3 Deleted Specification Items in R25-11

none

### C.17.4 Added Constraints in R25-11

none

### C.17.5 Changed Constraints in R25-11

none

### C.17.6 Deleted Constraints in R25-11

none

## C.18 Constraint and Specification Item History of this document according to AUTOSAR Release R26-11

### C.18.1 Added Specification Items in R26-11

none

### C.18.2 Changed Specification Items in R26-11

none

### C.18.3 Deleted Specification Items in R26-11

none

### C.18.4 Added Constraints in R26-11

none

### C.18.5 Changed Constraints in R26-11

none

### C.18.6 Deleted Constraints in R26-11

none

# D Mentioned Class Tables

This chapter contains the remaining set of meta-class tables which are not shown directly in the main body of this document.

| **Class** | ***AUTOSARElement*** (abstract) | | | |
|---|---|---|---|---|
| **Note** | An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course). | | | |
| **Base** | *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| **Subclasses** | *AbstractMessageToSecureChannelMapping*, AclObjectSet, AclOperation, AclPermission, AclRole, AliasNameSet, ApplicabilityInfoSet, ApplicationPartition, *AutosarDataType*, *BaseType*, BlueprintMappingSet, BswEntryRelationshipSet, BswModuleDescription, BswModuleEntry, BuildActionManifest, CalibrationParameterValueSet, ClientIdDefinitionSet, Collection, CompuMethod, ConsistencyNeedsBlueprintSet, ConstantSpecification, ConstantSpecificationMappingSet, CpSoftwareCluster, CpSoftwareClusterBinaryManifestDescriptor, CpSoftwareClusterMappingSet, CpSoftwareClusterResourcePool, CryptoEllipticCurveProps, CryptoServiceCertificate, CryptoServiceKey, CryptoServicePrimitive, CryptoServiceQueue, CryptoSignatureScheme, DataConstr, DataTransformationSet, DataTypeMappingSet, DdsCpConfig, *DiagnosticCommonElement*, DiagnosticConnection, DiagnosticContributionSet, DltArgumentPropsSet, DltContext, DltEcu, Documentation, E2EProfileCompatibilityProps, EcucDefinitionCollection, EcucDestinationUriDefSet, EcucModuleConfigurationValues, EcucModuleDef, EcucValueCollection, EthIpProps, EthTcpIpIcmpProps, EthTcpIpProps, EvaluatedVariantSet, FMFeature, FMFeatureMap, FMFeatureModel, FMFeatureSelectionSet, FirewallRule, FlatMap, GeneralPurposeConnection, HwCategory, HwElement, HwType, *IEEE1722TpConnection*, IPSecConfigProps, IPv6ExtHeaderFilterSet, *IdsCommonElement*, IdsDesign, *Implementation*, ImpositionTimeDefinitionGroup, InterpolationRoutineMappingSet, J1939ControllerApplication, KeywordSet, LifeCycleInfoSet, LifeCycleStateDefinitionGroup, LogAndTraceMessageCollectionSet, MacSecGlobalKayProps, MacSecParticipantSet, McFunction, McGroup, ModeDeclarationGroup, ModeDeclarationMappingSet, OsTaskProxy, PhysicalDimension, PhysicalDimensionMappingSet, *PortInterface*, PortInterfaceMappingSet, PortPrototypeBlueprint, PostBuildVariantCriterion, PostBuildVariantCriterionValueSet, PredefinedVariant, RapidPrototypingScenario, SdgDef, *SecureComProps*, *SecurityConnectivityAssociation*, SignalServiceTranslationPropsSet, SomeipSdClientEventGroupTimingConfig, SomeipSdClientServiceInstanceConfig, SomeipSdServerEventGroupTimingConfig, SomeipSdServerServiceInstanceConfig, SwAddrMethod, SwAxisType, SwComponentMappingConstraints, *SwComponentType*, SwRecordLayout, SwSystemconst, SwSystemconstantValueSet, SwcBswMapping, System, SystemComSpecDefinitionSet, SystemSignal, SystemSignalGroup, TDCpSoftwareClusterMappingSet, TcpOptionFilterSet, *TimingExtension*, TlsConnectionGroup, TlvDataIdDefinitionSet, TransformationPropsSet, Unit, UnitGroup, *UploadablePackageElement*, ViewMapSet | | | |
| **Aggregated by** | ARPackage.element | | | |
| **Attribute** | **Type** | **Mult.** | **Kind** | **Note** |
| – | – | – | – | – |

**Table D.1: AUTOSARElement**

| **Class** | **ARPackage** | | | |
|---|---|---|---|---|
| **Note** | AUTOSAR package, allowing to create top level packages to structure the contained AUTOSARElements.<br>ARPackages are open sets. This means that in a file based description system multiple files can be used to partially describe the contents of a package.<br>This is an extended version of MSR's SW-SYSTEM. | | | |
| **Base** | *ARObject*, *AtpBlueprint*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Referrable* | | | |
| **Aggregated by** | ARPackage.arPackage, AUTOSAR.arPackage | | | |
| **Attribute** | **Type** | **Mult.** | **Kind** | **Note** |

▽

△

| *Class* | **ARPackage** | | | |
|---|---|---|---|---|
| arPackage | ARPackage | * | aggr | This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy.<br>**Stereotypes:** atpVariation<br>**Tags:**<br>atp.Splitkey=arPackage, arPackage.shortName, ar Package.variationPoint.shortLabel<br>vh.latestBindingTime=blueprintDerivationTime<br>xml.sequenceOffset=30 |
| element | PackageableElement | * | aggr | Elements that are part of this package<br>**Stereotypes:** atpVariation<br>**Tags:**<br>atp.Splitkey=element, element.shortName, element.variationPoint.shortLabel<br>vh.latestBindingTime=systemDesignTime<br>xml.sequenceOffset=20 |
| referenceBase | ReferenceBase | * | aggr | This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references.<br>**Tags:**<br>atp.Splitkey=referenceBase, referenceBase.shortLabel<br>xml.sequenceOffset=10 |

**Table D.2: ARPackage**

| *Class* | **AUTOSAR** | | | |
|---|---|---|---|---|
| *Note* | Root element of an AUTOSAR description, also the root element in corresponding XML documents.<br>**Tags:** xml.globalElement=true | | | |
| *Base* | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| adminData | AdminData | 0..1 | aggr | This represents the administrative data of an Autosar file.<br>**Tags:**<br>atp.Splitkey=adminData<br>xml.sequenceOffset=10 |
| arPackage | ARPackage | * | aggr | This is the top level package in an AUTOSAR model.<br>**Stereotypes:** atpVariation<br>**Tags:**<br>atp.Splitkey=arPackage, arPackage.shortName, ar Package.variationPoint.shortLabel<br>vh.latestBindingTime=blueprintDerivationTime<br>xml.sequenceOffset=30 |
| fileInfo Comment | FileInfoComment | 0..1 | aggr | This represents a possibility to provide a structured comment in an AUTOSAR file.<br>**Stereotypes:** atpStructuredComment<br>**Tags:**<br>atp.Splitkey=fileInfoComment<br>xml.roleElement=true<br>xml.sequenceOffset=-10<br>xml.typeElement=false |

▽

△

| *Class* | **AUTOSAR** | | | |
|---|---|---|---|---|
| introduction | DocumentationBlock | 0..1 | aggr | This represents an introduction on the Autosar file. It is intended for example to represent disclaimers and legal notes.<br>**Tags:**<br>atp.Splitkey=introduction<br>xml.sequenceOffset=20 |

**Table D.3: AUTOSAR**

| *Class* | ***Describable*** (abstract) | | | |
|---|---|---|---|---|
| ***Note*** | This meta-class represents the ability to add a descriptive documentation to non identifiable elements. | | | |
| ***Base*** | *ARObject* | | | |
| ***Subclasses*** | CyclicTiming, EventControlledTiming, FunctionGroupErrorMapping, HwElementConnector, HwPin Connector, HwPinGroupConnector, IPduTiming, Ipv4DhcpServerConfiguration, Ipv6DhcpServer Configuration, PncMapping, *TransformationComSpecProps*, *TransformationDescription*, *Transformation ISignalProps* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| adminData | AdminData | 0..1 | aggr | This represents the administrative data for the describable object.<br>**Tags:**<br>atp.Splitkey=adminData<br>xml.sequenceOffset=-20 |
| category | CategoryString | 0..1 | attr | The category is a keyword that specializes the semantics of the Describable. It affects the expected existence of attributes and the applicability of constraints.<br>**Tags:** xml.sequenceOffset=-50 |
| desc | MultiLanguageOverview Paragraph | 0..1 | aggr | This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question.<br>More elaborate documentation, (in particular how the object is built or used) should go to "introduction".<br>**Tags:**<br>atp.Splitkey=desc<br>xml.sequenceOffset=-60 |
| introduction | DocumentationBlock | 0..1 | aggr | This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock.<br>**Tags:**<br>atp.Splitkey=introduction<br>xml.sequenceOffset=-30 |

**Table D.4: Describable**

| *Class* | **EvaluatedVariantSet** | | | |
|---|---|---|---|---|
| ***Note*** | This meta class represents the ability to express if a set of ARElements is able to support one or more particular variants.<br>In other words, for a given set of evaluatedElements this meta class represents a table of evaluated variants, where each PredefinedVariant represents one column. In this column each descendant sw SystemconstantValue resp. postbuildVariantCriterionValue represents one entry.<br>In a graphical representation each swSystemconstantValueSet / postBuildVariantCriterionValueSet could be used as an intermediate headline in the table column.<br>If the approvalStatus is "APPROVED" it expresses that the collection of CollectableElements is known be valid for the given evaluatedVariants.<br>Note that the EvaluatedVariantSet is a CollectableElement. This allows to establish a hierarchy of EvaluatedVariantSets.<br>**Tags:** atp.recommendedPackage=EvaluatedVariantSets | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Aggregated by*** | ARPackage.element | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| approvalStatus | NameToken | 0..1 | attr | Defines the approval status of a predefined variant. Two values are predefined: "APPROVED" and "REJECTED":<br>• Approved variants are known to work.<br>• Rejected variants are known NOT to work.<br>Further values can be approved on a per-company basis; within AUTOSAR only "APPROVED" and "REJECTED" should be recognized. |
| evaluated Element | CollectableElement | * | ref | This represents a particular element which is evaluated in context of the EvaluatedVariants. The approvalStatus applies to this element (and all of its descendants). In other words, the referenced elements are those that were considered when the predefined variant was evaluated.<br>**Tags:** atp.Splitkey=evaluatedElement |
| evaluated Variant | PredefinedVariant | * | ref | This metaclass represents one particular variant which was evaluated. LowerMultiplicity is set to 0 to support a stepwise approach.<br>**Tags:** atp.Splitkey=evaluatedVariant |

**Table D.5: EvaluatedVariantSet**

| *Class* | ***Identifiable*** (abstract) |
|---|---|
| ***Note*** | Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables. |
| ***Base*** | *ARObject*, *MultilanguageReferrable*, *Referrable* |

▽

△

| **Class** | **Identifiable** (abstract) | | | |
|---|---|---|---|---|
| **Subclasses** | ARPackage, *AbstractDoIpLogicAddressProps*, *AbstractEvent*, *AbstractImplementationDataTypeElement*, *AbstractSecurityEventFilter*, *AbstractSecurityIdsmInstanceFilter*, *AbstractServiceInstance*, AppOsTask ProxyToEcuTaskProxyMapping, ApplicationEndpoint, ApplicationError, ApplicationPartitionToEcuPartition Mapping, AppliedStandard, AsynchronousServerCallResultPoint, *AtpBlueprint*, *AtpClassifier*, *Atp Feature*, AutosarOperationArgumentInstance, AutosarVariableInstance, *BinaryManifestAddressable Object*, BinaryManifestItemDefinition, *BinaryManifestResource*, BinaryManifestResourceDefinition, Block State, BswInternalTriggeringPoint, BswModuleDependency, *BuildActionEntity*, BuildActionEnvironment, CanSecSecureChannel, CanTpAddress, CanTpChannel, CanTpNode, Chapter, ClientIdDefinition, Client ServerOperation, Code, *CollectableElement*, ComManagementMapping, *CommConnectorPort*, *CommunicationConnector*, *CommunicationController*, Compiler, ConsistencyNeeds, ConsumedEvent Group, *CouplingElementAbstractDetails*, CouplingPort, *CouplingPortAbstractShaper*, *CouplingPort StructuralElement*, *CpSoftwareClusterResource*, CpSoftwareClusterResourceToApplicationPartition Mapping, CpSoftwareClusterToApplicationPartitionMapping, CpSoftwareClusterToEcuInstanceMapping, CpSoftwareClusterToResourceMapping, *CryptoServiceMapping*, CyclicHandlingComDataToOsTask ProxyMapping, DataPrototypeGroup, DataPrototypeTransformationPropsIdent, DataTransformation, *Dds AbstractServiceInstanceElementCp*, DdsCpDomain, DdsCpPartition, DdsCpQosProfile, DdsCpTopic, DependencyOnArtifact, *DiagEventDebounceAlgorithm*, DiagnosticAuthTransmitCertificateEvaluation, DiagnosticConnectedIndicator, DiagnosticDataElement, DiagnosticDebounceAlgorithmProps, Diagnostic ExtendedDataRecordElement, DiagnosticFunctionInhibitSource, DiagnosticParameterElement, *DiagnosticRoutineSubfunction*, DltApplication, DltArgument, DltArgumentProps, DltLogChannel, Dlt Message, DoIpInterface, DoIpLogicAddress, DoIpRoutingActivation, ECUMapping, *EOCExecutableEntity RefAbstract*, EcuPartition, EcuPartitionToCoreMapping, EcucContainerValue, *EcucDefinitionElement*, EcucDestinationUriDef, EcucEnumerationLiteralDef, EcucQuery, EcucValidationCondition, Ethernet WakeupSleepOnDatalineConfig, EventHandler, ExclusiveArea, *ExecutableEntity*, *ExecutionTime*, FMAttributeDef, FMFeatureMapAssertion, FMFeatureMapCondition, FMFeatureMapElement, FMFeature Relation, FMFeatureRestriction, FMFeatureSelection, FlatInstanceDescriptor, FlexrayTpConnection Control, FlexrayTpNode, FlexrayTpPduPool, *FrameTriggering*, GlobalTimeGateway, *GlobalTimeMaster*, *GlobalTimeSlave*, *HeapUsage*, HwAttributeDef, HwAttributeLiteralDef, HwPin, HwPinGroup, *IEEE1722 TpAcfBus*, *IEEE1722TpAcfBusPart*, IPSecRule, IPv6ExtHeaderFilterList, ISignalToIPduMapping, ISignal Triggering, *IdentCaption*, ImpositionTime, InternalTriggeringPoint, J1939Node, J1939SharedAddress Cluster, J1939TpNode, Keyword, LifeCycleState, LinScheduleTable, LinTpNode, Linker, MacAddress VlanMembership, MacMulticastGroup, MacSecKayParticipant, McDataInstance, MemorySection, Mode Declaration, ModeDeclarationMapping, ModeSwitchPoint, ModeSwitchSenderComSpecProps, Network Endpoint, *NmCluster*, NmEcu, *NmNode*, NvBlockDescriptor, *PackageableElement*, ParameterAccess, PduActivationRoutingGroup, PduToFrameMapping, PduTriggering, PerInstanceMemory, *Physical Channel*, PortElementToCommunicationResourceMapping, PortGroup, *PortInterfaceMapping*, Queued ReceiverComSpecProps, ResourceConsumption, RootSwCompositionPrototype, RptComponent, Rpt Container, RptExecutableEntity, RptExecutableEntityEvent, RptExecutionContext, RptProfile, RptService Point, RteEventInCompositionSeparation, RteEventInCompositionToOsTaskProxyMapping, RteEventIn SystemSeparation, RteEventInSystemToOsTaskProxyMapping, RunnableEntityGroup, *SdgAttribute*, Sdg Class, SecOcJobRequirement, SecureCommunicationAuthenticationProps, SecureCommunication FreshnessProps, SecurityEventContextDataElement, SecurityEventContextProps, *ServerCallPoint*, ServerComSpecProps, *ServiceNeeds*, SignalServiceTranslationElementProps, SignalServiceTranslation EventProps, SignalServiceTranslationProps, SocketAddress, SomeipTpChannel, *StackUsage*, Static SocketConnection, StructuredReq, SwGenericAxisParamType, SwServiceArg, SwcServiceDependency, SwcToApplicationPartitionMapping, SwcToEcuMapping, SwcToImplMapping, SwitchAsynchronousTraffic ShaperGroupEntry, SwitchAtsInstanceEntry, SwitchFlowMeteringEntry, SwitchStreamFilterActionDest PortModification, SwitchStreamFilterEntry, SwitchStreamFilterRule, SwitchStreamGateEntry, Switch StreamIdentification, SystemMapping, SystemSignalGroupToCommunicationResourceMapping, System SignalToCommunicationResourceMapping, TDCpSoftwareClusterMapping, TDCpSoftwareCluster ResourceMapping, TcpOptionFilterList, *TimingClock*, TimingClockSyncAccuracy, TimingCondition, *TimingConstraint*, *TimingDescription*, TimingExtensionResource, TimingModeInstance, TlsCryptoCipher Suite, TlsCryptoCipherSuiteProps, Topic1, TpAddress, TraceableTable, TraceableText, *TracedFailure*, TransformationISignalPropsIdent, *TransformationProps*, TransformationTechnology, Trigger, Variable Access, VariationPointProxy, ViewMap, VlanConfig, WaitPoint | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| adminData | AdminData | 0..1 | aggr | This represents the administrative data for the identifiable object.<br>**Tags:**<br>atp.Splitkey=adminData<br>xml.sequenceOffset=-40 |

▽

△

| ***Class*** | ***Identifiable*** (abstract) | | | |
|---|---|---|---|---|
| annotation | Annotation | * | aggr | Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes.<br>**Tags:**<br>atp.Splitkey=annotation<br>xml.sequenceOffset=-25 |
| category | CategoryString | 0..1 | attr | The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints.<br>**Tags:** xml.sequenceOffset=-50 |
| desc | MultiLanguageOverview Paragraph | 0..1 | aggr | This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question.<br>More elaborate documentation, (in particular how the object is built or used) should go to "introduction".<br>**Tags:**<br>atp.Splitkey=desc<br>xml.sequenceOffset=-60 |
| introduction | DocumentationBlock | 0..1 | aggr | This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock.<br>**Tags:**<br>atp.Splitkey=introduction<br>xml.sequenceOffset=-30 |
| uuid | String | 0..1 | attr | The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp.<br>**Tags:** xml.attribute=true |

**Table D.6: Identifiable**

| ***Class*** | ***PortPrototype*** (abstract) |
|---|---|
| ***Note*** | Base class for the ports of an AUTOSAR software component.<br>The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports. |
| ***Base*** | *ARObject*, *AtpFeature*, *AtpPrototype*, *Identifiable*, *MultilanguageReferrable*, *Referrable* |

▽

△

| **Class** | ***PortPrototype*** (abstract) | | | |
|---|---|---|---|---|
| **Subclasses** | *AbstractProvidedPortPrototype*, *AbstractRequiredPortPrototype* | | | |
| **Aggregated by** | *AtpClassifier*.atpFeature, *SwComponentType*.port | | | |
| **Attribute** | **Type** | **Mult.** | **Kind** | **Note** |
| clientServer Annotation | ClientServerAnnotation | * | aggr | Annotation of this `PortPrototype` with respect to client/ server communication.<br>**Tags:** atp.Splitkey=clientServerAnnotation |
| delegatedPort Annotation | DelegatedPort Annotation | 0..1 | aggr | Annotations on this delegated port.<br>**Tags:** atp.Splitkey=delegatedPortAnnotation |
| ioHwAbstraction Server Annotation | IoHwAbstractionServer Annotation | * | aggr | Annotations on this IO Hardware Abstraction port.<br>**Tags:** atp.Splitkey=ioHwAbstractionServerAnnotation |
| modePort Annotation | ModePortAnnotation | * | aggr | Annotations on this mode port.<br>**Tags:** atp.Splitkey=modePortAnnotation |
| nvDataPort Annotation | NvDataPortAnnotation | * | aggr | Annotations on this non voilatile data port.<br>**Tags:** atp.Splitkey=nvDataPortAnnotation |
| parameterPort Annotation | ParameterPort Annotation | * | aggr | Annotations on this parameter port.<br>**Tags:** atp.Splitkey=parameterPortAnnotation |
| senderReceiver Annotation | SenderReceiver Annotation | * | aggr | Collection of annotations of this ports sender/receiver communication.<br>**Tags:** atp.Splitkey=senderReceiverAnnotation |
| triggerPort Annotation | TriggerPortAnnotation | * | aggr | Annotations on this trigger port.<br>**Tags:** atp.Splitkey=triggerPortAnnotation |

**Table D.7: PortPrototype**

| **Class** | **PostBuildVariantCriterion** | | | |
|---|---|---|---|---|
| **Note** | This class specifies one particular PostBuildVariantSelector.<br>**Tags:** atp.recommendedPackage=PostBuildVariantCriterions | | | |
| **Base** | *ARElement*, *ARObject*, *AtpDefinition*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| **Aggregated by** | ARPackage.element | | | |
| **Attribute** | **Type** | **Mult.** | **Kind** | **Note** |
| compuMethod | CompuMethod | 0..1 | ref | The compuMethod specifies the possible values for the variant criterion serving as an enumerator.<br>**Tags:** atp.Splitkey=compuMethod |

**Table D.8: PostBuildVariantCriterion**

| **Class** | ***Referrable*** (abstract) | | | |
|---|---|---|---|---|
| **Note** | Instances of this class can be referred to by their identifier (while adhering to namespace borders). | | | |
| **Base** | *ARObject* | | | |
| **Subclasses** | *AtpDefinition*, BswDistinguishedPartition, *BswModuleCallPoint*, BswModuleClientServerEntry, Bsw VariableAccess, CouplingPortTrafficClassAssignment, *DiagnosticEnvModeElement*, EthernetPriority Regeneration, ExclusiveAreaNestingOrder, *HwDescriptionEntity*, *ImplementationProps*, LinSlaveConfig Ident, ModeTransition, *MultilanguageReferrable*, PncMappingIdent, *SingleLanguageReferrable*, SoCon IPduIdentifier, TpConnectionIdent | | | |
| **Attribute** | **Type** | **Mult.** | **Kind** | **Note** |

▽

△

| *Class* | ***Referrable*** (abstract) | | | |
|---|---|---|---|---|
| shortName | Identifier | 1 | attr | This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference.<br>**Stereotypes:** atpIdentityContributor<br>**Tags:**<br>xml.enforceMinMultiplicity=true<br>xml.sequenceOffset=-100 |
| shortName Fragment | ShortNameFragment | * | aggr | This specifies how the Referrable.shortName is composed of several shortNameFragments.<br>**Tags:**<br>atp.Splitkey=shortNameFragment<br>xml.sequenceOffset=-90 |

**Table D.9: Referrable**

| *Class* | **SwSystemconst** | | | |
|---|---|---|---|---|
| ***Note*** | This element defines a system constant which serves an input to select a particular variation point. In particular a system constant serves as an operand of the binding function (swSyscond) in a Variation point.<br>Note that the binding process can only happen if a value was assigned to to the referenced system constants.<br>**Tags:** atp.recommendedPackage=SwSystemconsts | | | |
| ***Base*** | *ARElement*, *ARObject*, *AtpDefinition*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *PackageableElement*, *Referrable* | | | |
| ***Aggregated by*** | ARPackage.element | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| swDataDef Props | SwDataDefProps | 0..1 | aggr | This denotes the data definition properties of the system constant. This supports to express the limits and optionally a conversion within the internal to physical values by a compu method.<br>**Tags:**<br>atp.Splitkey=swDataDefProps<br>xml.sequenceOffset=40 |

**Table D.10: SwSystemconst**

| *Class* | **Unit** | | | |
|---|---|---|---|---|
| ***Note*** | This is a physical measurement unit. All units that might be defined should stem from SI units. In order to convert one unit into another factor and offset are defined.<br>For the calculation from SI-unit to the defined unit the factor (factorSiToUnit ) and the offset (offsetSiTo Unit ) are applied as follows:<br>x [{unit}] := y * {siUnit} * factorSiToUnit [{unit}/{siUnit}] + offsetSiToUnit [{unit}]<br>For the calculation from a unit to SI-unit the reciprocal of the factor (factorSiToUnit ) and the negation of the offset (offsetSiToUnit ) are applied.<br>y [{siUnit}] := (x*{unit} - offsetSiToUnit [{unit}]) / factorSiToUnit [{unit}/{siUnit}]<br>**Tags:** atp.recommendedPackage=Units | | | |
| ***Base*** | *ARElement*, *ARObject*, *CollectableElement*, *Identifiable*, *MultilanguageReferrable*, *Packageable Element*, *Referrable* | | | |
| ***Aggregated by*** | ARPackage.element | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |

▽

△

| *Class* | **Unit** | | | |
|---|---|---|---|---|
| displayName | SingleLanguageUnit Names | 0..1 | aggr | This specifies how the unit shall be displayed in documents or in user interfaces of tools.The displayName corresponds to the Unit.Display in an ASAM MCD-2MC file.<br>**Tags:**<br>atp.Splitkey=displayName<br>xml.sequenceOffset=20 |
| factorSiToUnit | Float | 0..1 | attr | This is the factor for the conversion from SI Units to units.<br>The inverse is used for conversion from units to SI Units.<br>**Tags:** xml.sequenceOffset=30 |
| offsetSiToUnit | Float | 0..1 | attr | This is the offset for the conversion from and to siUnits.<br>**Tags:** xml.sequenceOffset=40 |
| physical Dimension | PhysicalDimension | 0..1 | ref | This association represents the physical dimension to which the unit belongs to. Note that only values with units of the same physical dimensions might be converted.<br>**Tags:**<br>atp.Splitkey=physicalDimension<br>xml.sequenceOffset=50 |

**Table D.11: Unit**

| *Class* | **VariationPoint** | | | |
|---|---|---|---|---|
| ***Note*** | This meta-class represents the ability to express a "structural variation point". The container of the variation point is part of the selected variant if swSyscond evaluates to true and each postBuildVariant Criterion is fulfilled. | | | |
| ***Base*** | *ARObject* | | | |
| ***Attribute*** | ***Type*** | ***Mult.*** | ***Kind*** | ***Note*** |
| blueprint Condition | DocumentationBlock | 0..1 | aggr | This represents a description that documents how the variation point shall be resolved when deriving objects from the blueprint.<br>Note that variationPoints are not allowed within a blueprintCondition.<br>**Tags:** xml.sequenceOffset=28 |
| desc | MultiLanguageOverview Paragraph | 0..1 | aggr | This allows to describe shortly the purpose of the variation point.<br>**Tags:** xml.sequenceOffset=20 |
| formalBlueprint Generator | BlueprintGenerator | 0..1 | aggr | This represents a description that documents how the variation point shall be resolved when deriving objects from the blueprint by using ARMQL.<br>Note that variationPoints are not allowed within a formal BlueprintGenerator.<br>**Tags:**<br>atp.Status=draft<br>xml.sequenceOffset=30 |
| postBuildVariant Condition | PostBuildVariant Condition | * | aggr | This is the set of post build variant conditions which all shall be fulfilled in order to (postbuild) bind the variation point.<br>**Tags:** xml.sequenceOffset=40 |

▽

△

| *Class* | **VariationPoint** | | | |
|---|---|---|---|---|
| sdg | Sdg | 0..1 | aggr | An optional special data group is attached to every variation point. These data can be used by external software systems to attach application specific data. For example, a variant management system might add an identifier, an URL or a specific classifier.<br>**Tags:** xml.sequenceOffset=50 |
| shortLabel | Identifier | 0..1 | attr | This provides a name to the particular variation point to support the RTE generator. It is necessary for supporting splitable aggregations and if binding time is later than codeGenerationTime, as well as some RTE conditions. It needs to be unique with in the enclosing Identifiables with the same ShortName.<br>**Stereotypes:** atpIdentityContributor<br>**Tags:** xml.sequenceOffset=10 |
| swSyscond | ConditionByFormula | 0..1 | aggr | This condition acts as Binding Function for the Variation Point. Note that the multiplicity is 0..1 in order to support pure postBuild variants.<br>**Tags:** xml.sequenceOffset=30 |

**Table D.12: VariationPoint**