Document Title	# Specification of ECU Resource Template
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	060
Document Classification	Standard

Document Version	2.2.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Description
2011-11-08	2.2.0	AUTOSAR Administration	• Added detailed change history (appendix C) • Added [constr_3500]
2010-07-29	2.1.0	AUTOSAR Administration	• Added Glossary appendix. • Updated category definitions to upper case.
2009-12-04	2.0.0	AUTOSAR Administration	• Reworked for Release 4.0.
2008-01-22	1.0.3	AUTOSAR Administration	• Correction of References
2007-12-05	1.0.2	AUTOSAR Administration	• Document meta information extended • Small layout adaptations made

2007-01-31	1.0.1	AUTOSAR Administration	• Legal disclaimer revised • Release Notes added • "'Advice for users"' revised • "'Revision Information"' added
2005-05-09	1.0.0	AUTOSAR Administration	Initial release

Table of Contents

References

[1] Requirements on ECU Resource Template
AUTOSAR_RS_ECUResourceTemplate.pdf

[2] Meta Model
AUTOSAR_MMOD_MetaModel.eap

[3] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate.pdf

[4] Model Persistence Rules for XML
AUTOSAR_TR_XMLPersistenceRules.pdf

[5] Generic Structure Template
AUTOSAR_TPS_GenericStructureTemplate.pdf

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

[7] 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.

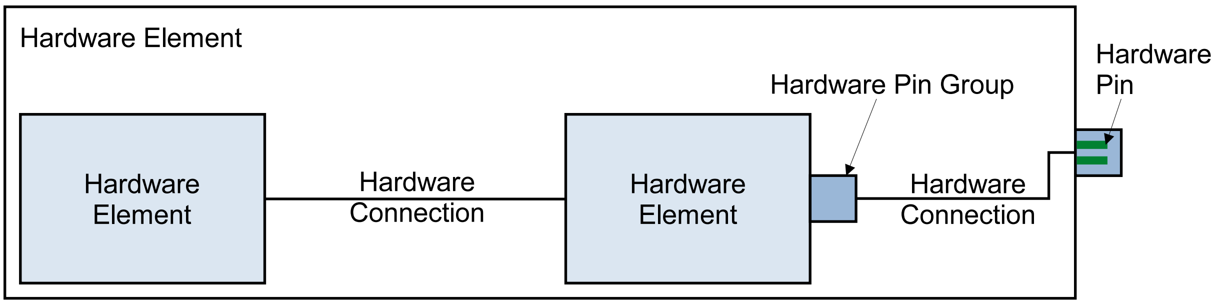

Figure 1.1: Overview of ECU Resource template

Modeling elements in the ECU Resource Template can be hierarchically organized. A particular ECU can be described as a hierarchical composition of 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 Requirements Traceability

Tracing of the Requirements on ECU Resource Template [1].

Requirement	Description	Satisfied by
ECUR0005 Support configuration of Basic Software	The ECU Resource template shall provide means to describe hardware properties which are supporting the configuration of the AUTOSAR Basic Software.	The relationships between upstream templates and ECU Configuration are described in the AUTOSAR Metamodel [2]. The configuration parameters in the M1 model contain a number of tagged values with the mapping information.
ECUR0003 Describe characteristic properties of specific hardware elements	The ECU Resource template shall provide means to describe the common and characteristic properties of hardware elements based on their kind.	The requirement is fulfilled by the defined categories and their attributes in chapter 3.
ECUR0004 Describe generic hardware	The ECU Resource template shall provide means to describe hardware elements of any kind.	A HW vendor can extend the categories of AUTOSAR. New categories can be defined. Attributes can be added to existing categories and new literals to existing enumerations.
ECUR0006 Describe connections between hardware elements	The ECU Resource template shall provide means to describe in an abstracted way how the individual hardware elements - in an ECU and on the outside of the ECU - are connected.	Hardware Connections can be described on several levels in the ECU Resource Template. These levels are described in chapter 2.5.
ECUR0014 Timing properties of hardware	The ECU Resource template shall provide means to describe the timing properties for hardware I/O, e.g. the latency introduced by a digital I/O hardware port.	A HW vendor can extend the categories of AUTOSAR. New categories can be defined. New timing attributes can be added to existing categories and new literals to existing enumerations.
ECUR0015 Describe variability of the hardware	It shall be possible to describe the variability the actual hardware provides.	The requirement is fulfilled by the AUTOSAR Variant Handling concept (chapter 2.7).
ECUR0017 Documentation Support	The ECU Resource template shall provide means to add documentation to the hardware elements.	The requirement is fulfilled by the AUTOSAR Documentation Support concept (chapter 2.8).

Requirement	Description	Satisfied by
ECUR0018 Support hardware descriptions from several sources	The ECU Resource template shall provide means to combine the hardware descriptions from several sources.	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 and the nested hardware elements (chapter 2.3).
ECUR0007 Processing Unit specification	The ECU Resource template shall provide dedicated means to describe a processing unit. A processing unit shall be defined as the core of the micro controller / processor.	The requirement is fulfilled by the Processing Unit Category (chapter 3.1.2).
ECUR0008 Available memory	The ECU Resource template shall provide dedicated means to describe memory segments. This includes all possible memory kinds like RAM, ROM, EEPROM, Flash, etc.	The requirement is fulfilled by the Memory Category (chapter 3.1.4).
ECUR0009 Available communication means	The ECU Resource template shall provide dedicated means to describe communication hardware.	The requirement is fulfilled by the Hw Pin Group Categories. (chapter 3.2).
ECUR0010 Available IO HW-Peripherals	The ECU Resource template shall provide dedicated means to describe IO-HW-Peripherals.	The requirement is fulfilled by the Digital IO (chapter 3.1.7) and Analog IO (chapter 3.1.8) categories.
ECUR0016 IO-HW-Abstraction specification	The ECU Resource template shall provide the abstract connection information between the hardware sensor / actuator and the IO-HW peripheral using the IO-HW-Abstraction layer.	The requirement is fulfilled by the ECU Abstraction Software Component that is defined in the Software Component Template [3]. The ECU Abstraction is a special AtomicSwComponentType that resides between a software-component that wants to access ECU periphery and the Microcontroller Abstraction.
ECUR0011 Available sensors and actuators	The ECU Resource template shall provide dedicated means to describe sensors and actuators.	The requirement is fulfilled by the SensorActuator Category (chapter 3.1.11).
ECUR0012 Development according to the AUTOSAR Generic Structure Template document	The UML representation of the ECU Resource template SHALL be developed according to the AUTOSAR Generic Structure Template.	The requirement is fulfilled by the AUTOSAR development process.

Requirement	Description	Satisfied by
ECUR0013 Transformation of ECU Resource template modeling according to the AUTOSAR Model Persistence Rules for XML	The XML representation for the ECU Resource template shall be derived from its UML representation according to the AUTOSAR Model Persistence Rules for XML.	The requirement is fulfilled by the AUTOSAR XML Schema generation process. The document called Model Persistence Rules [4] for XML describes how XML is used and how the meta-model designed in the "Ecu Resource Template" should be translated by the "Schema Generator" (MDS) into XML-Schema (XSD) "Data Exchange Format".

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.

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

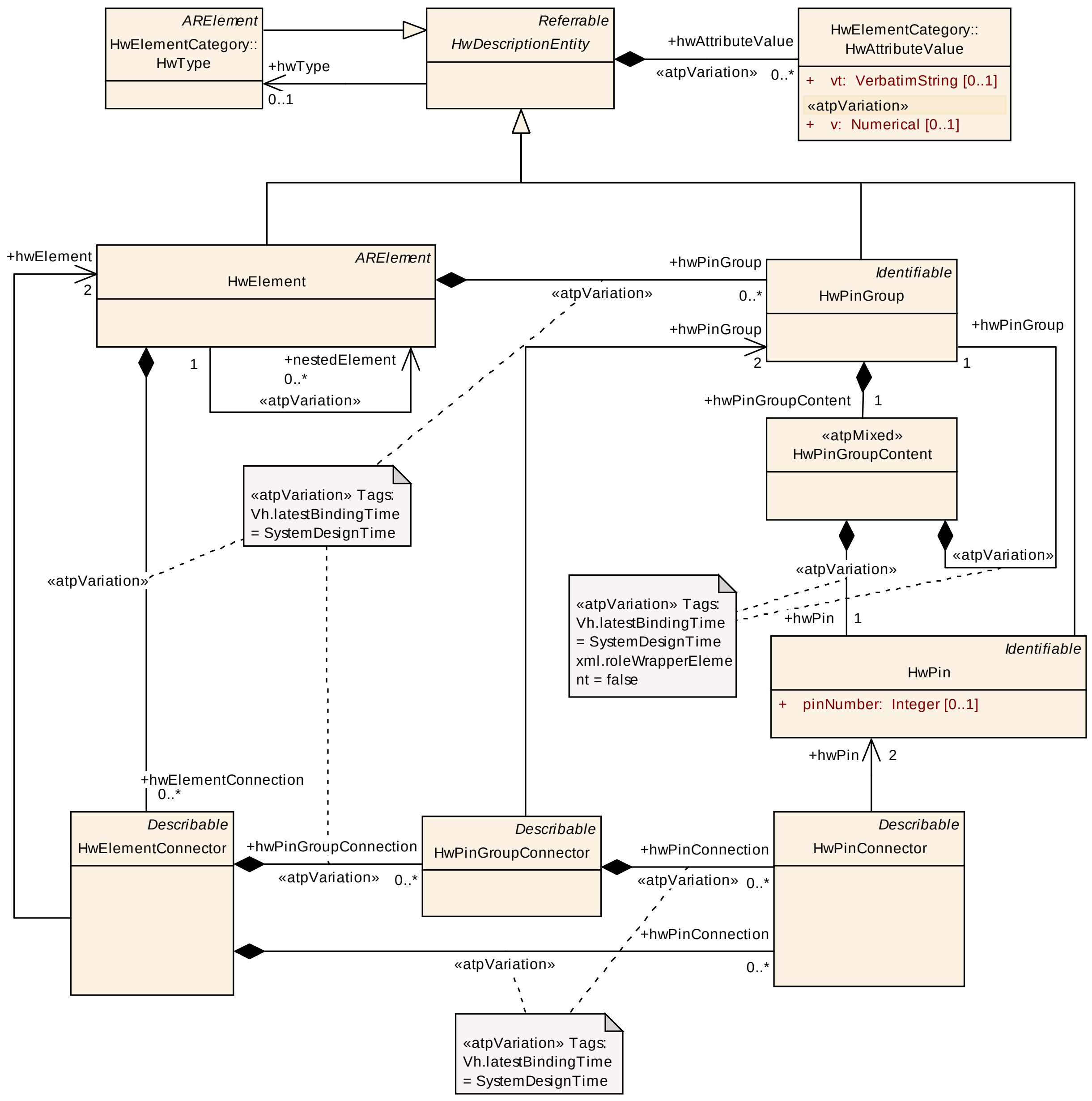

Figure 2.1: Overview of ECU Resource template classes

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.

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`.

- It shall be possible 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).
- It shall be possible to extend the standardized `HwCategory` specification with additional attributes (see section 2.6.1).

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

Each `HwDescriptionEntity` may aggregate several `HwAttributeValue` elements.

Class	**HwDescriptionEntity (abstract)**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
Note	This meta-class represents the ability to describe a hardware entity.			
Base	ARObject,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
hwAttributeValue	HwAttributeValue	*	aggr	This aggregation represents a particular hardware attribute value. **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=SystemDesignTime xml.sequenceOffset=50
hwCategory	HwCategory	*	ref	One of the associations representing one particular category of the hardware entity. **Tags:** 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 HwTypes can not be redefined and therefore must not have a hwType reference.

Table 2.1: HwDescriptionEntity

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`. The definition of attributes is described in section 2.6.

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 (see also section 2.7).

Using the role `annotation` additional documentation can be provided in the `Annotation` element to the `HwAttributeValue`.

Class	**HwAttributeValue**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
Note	This metaclass represents the ability to assign a hardware attribute value. Note that v and vt are mutually exclusive.			
Base	ARObject			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
annotation	Annotation	0..1	aggr	Optional annotation that can be added to each HwAttributeValue.
hwAttribute Def	HwAttributeDef	1	ref	This association represents the definition of the particular hardware attribute value.
v	Numerical	0..1	attr	This represents a numerical hardware attribute value. **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=SystemDesignTime
vt	VerbatimString	0..1	attr	This represents a textual hardware attribute value.

Table 2.2: HwAttributeValue

2.2 Hardware Type

The hardware type 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).

The attribute values defined in the `HwType` are applicable for all occurrences of this `HwType`, although it is possible to override the value in the `HwElement` (see section 2.3).

A `HwType` (being a `HwDescriptionEntity`) shall not have a reference to another `HwType` in the role `hwType`. The definition of `HwType`s 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**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
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. **Tags:** atp.recommendedPackage=HwTypes			
Base	ARElement,ARObject,CollectableElement,HwDescription Entity,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 2.3: HwType

2.3 Hardware Element

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.

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).

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 hardware element 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**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
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. **Tags:** atp.recommendedPackage=HwElements			
Base	ARElement,ARObject,CollectableElement,HwDescriptionEntity,Identifiable,MultilanguageReferrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
hwElementConnection	HwElementConnector	*	aggr	This represents one particular connection between two hardware elements. **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=SystemDesignTime 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. **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=SystemDesignTime 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). **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=SystemDesignTime xml.sequenceOffset=70

Table 2.4: HwElement

2.3.1 Multiple occurrence of Hardware Elements

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.

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**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
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			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
hwPinGroupContent	HwPinGroupContent	1	aggr	This aggregation describes the contained pins/pin groups.

Table 2.5: HwPinGroup

The HwPinGroupContent can contain HwPinGroup and HwPin. The HwPinGroupContent is defined as «atpMixed» (see Generic Structure Template [5]). 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**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
Note	This meta-class specifies a mixture of hwPins and hwPinGroups.			
Base	ARObject			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
hwPin	HwPin	1	aggr	This aggregation represents a hardware pin in a hardware pin group. **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=SystemDesignTime xml.roleWrapperElement=false
hwPinGrou p	HwPinGroup	1	aggr	This aggregation represents a nested hardware pin group. **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=SystemDesignTime 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**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
Note	This meta-class represents the possibility to describe a hardware pin.			
Base	ARObject,HwDescriptionEntity,Identifiable,MultilanguageReferrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
pinNumber	Integer	0..1	attr	This attribute contains the physical pin number.

Table 2.7: HwPin

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.

The `HwElementConnector` 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**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
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			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
hwElement	HwElement	2	ref	This association connects two hardware elements.
hwPinConnection	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. **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=SystemDesignTime xml.sequenceOffset=60
hwPinGroupConnection	HwPinGroupConnector	*	aggr	This represents one particular connection between two hardware pin groups. **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=SystemDesignTime xml.sequenceOffset=50

Table 2.8: HwElementConnector

The `HwPinGroupConnector` allows to describe the connection between two `HwPinGroup`s.

Class	**HwPinGroupConnector**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
Note	This meta-class represents the ability to connect two pin groups.			
Base	ARObject,Describable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
hwPinConnection	HwPinConnector	*	aggr	This represents one particular connection between two hardware pins. The connected pins must match the connection provided by the parent hwPinGroupConnection. **Stereotypes:** atpVariation **Tags:** Vh.latestBindingTime=SystemDesignTime
hwPinGroup	HwPinGroup	2	ref	This association connects two hardware pin groups.

Table 2.9: HwPinGroupConnector

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

Class	**HwPinConnector**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate			
Note	This meta-class represents the ability to connect two pins.			
Base	ARObject,Describable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
hwPin	HwPin	2	ref	This association connects two hardware pins.

Table 2.10: HwPinConnector

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.

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.

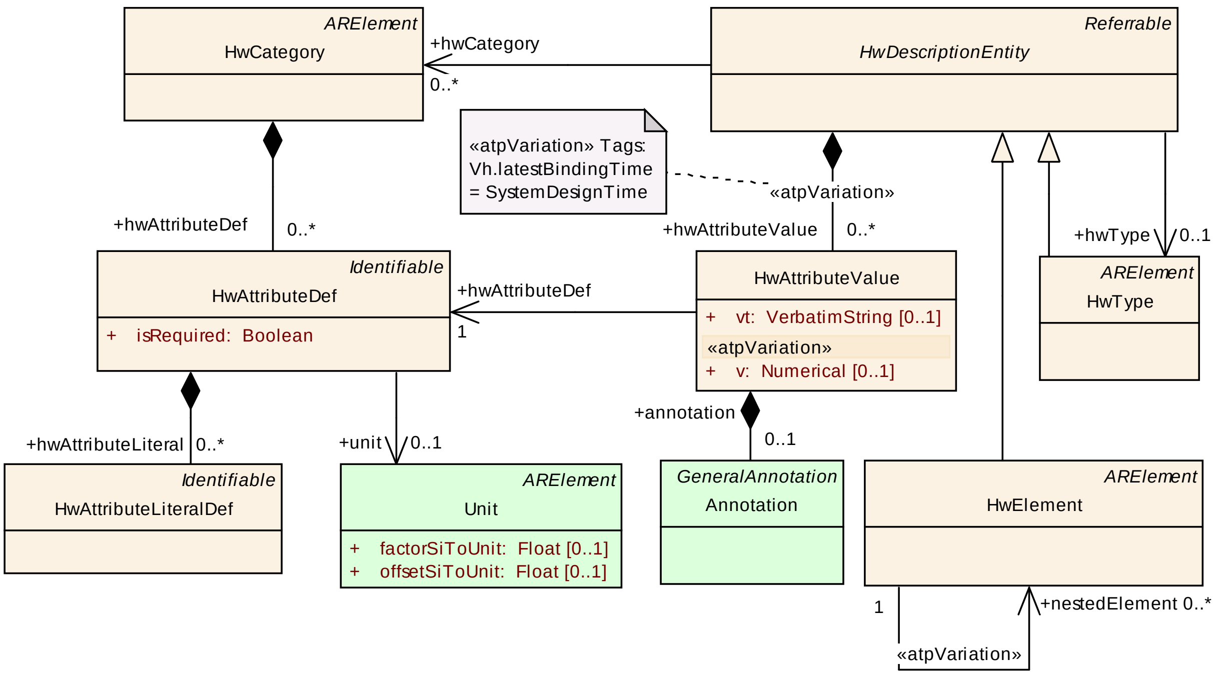

Figure 2.2: Definition of hardware categories

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 table 3.1 and table 3.5.

The `HwCategory` may contain several `HwAttributeDef` elements.

Class	**HwCategory**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
Note	This metaclass represents the ability to declare hardware categories and its particular attributes. **Tags:** atp.recommendedPackage=HwCategorys			
Base	ARElement,ARObject,CollectableElement,Identifiable,Multilanguage Referrable,PackageableElement,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***

Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
hwAttribute Def	HwAttributeDef	*	aggr	This aggregation describes particular hardware attribute definition.

Table 2.11: HwCategory

The `HwAttributeDef` specifies one attribute which is applicable for the `HwCate-gory`.

The name of the attribute is defined in the `shortName`.

The type of the attribute is specified by the `category`. Applicable values for the `cat-egory` of `HwAttributeDef` are defined in table 2.12.

[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 allowed** to extend the meaning of the attribute `category` of meta-class `HwAttribut-eDef` ⌋

Category	**Description**
`BOOLEAN`	Defines a boolean attribute. The values of a boolean attribute can be provided in • textual format 'true' / 'false' (using the `vt` element of `HwAttributeValue` • numerical format '1' (true) / '0' (false) (using the `v` element of `HwAttributeValue`
`INTEGER`	Defines an integer attribute. 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. The value of a float attribute is represented as an IEEE double-precision 64-bit floating point of the IEEE 754-1985 standard [6].
`ENUMERATION`	Defines an enumeration attribute. The possible enumeration literals are defined with the element `hwAttributeLiteral`. The value of an enumeration attribute is provided as text in the `vt` element of `HwAttributeValue`.
`STRING`	Defines a string attribute. The value of a string attribute is provided as text in the `vt` element of `HwAttributeValue`.

Table 2.12: Hardware Attribute Categories

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

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 [3].

Class	**HwAttributeDef**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
Note	This metaclass represents the ability to define a particular hardware attribute. The category of this element defines the type of the attributeValue. If the category is Enumeration the hwAttributeEnumerationLiterals specify the available literals.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
hwAttribute Literal	HwAttributeLiter alDef	*	aggr	The available EnumerationLiterals of the Enumeration definition. Only applicable if the category of the HwAttributeDef equals Enumeration.
isRequired	Boolean	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.

Table 2.13: HwAttributeDef

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

Class	**HwAttributeLiteralDef**			
Package	M2::AUTOSARTemplates::EcuResourceTemplate::HwElementCategory			
Note	One available EnumerationLiteral of the Enumeration definition. Only applicable if the category of the HwAttributeDef equals Enumeration.			
Base	ARObject,Identifiable,MultilanguageReferrable,Referrable			
Attribute	***Datatype***	***Mul.***	***Kind***	***Note***
–	–	–	–	–

Table 2.14: HwAttributeLiteralDef

In example A.8 the definition of some attributes for the `MemorySegment` category 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:

- New hardware categories for `HwElement` and `HwPinGroup` and `HwPin` can be defined if they are different from the categories defined in section 3. This definition shall be in a package which is not the `AUTOSAR` package. A `HwDescriptionEntity` shall then reference the extended hardware category.

- An existing hardware category from section 3 can be extended with new attribute definitions. The extension is via defining a hardware category of the same name as the standardized one in a different package than AUTOSAR. A HwDescriptionEntity shall then reference the standardized and the extended hardware category.
- An extension of the standardized hardware category shall not define the same hardware attributes as already defined in the standardized hardware category.
- An existing enumeration attribute from section 3 can be extended with new enumeration literals.
- Enumeration literals shall not be removed from the specified enumeration attributes in section 3.
- The category (type) of specified attributes in section 3 shall not be changed.
- The value of the isRequired element shall not be changed for specified attributes in section 3.
- The value of the unit element shall not be changed for specified attributes in section 3.

2.7 Ecu Resource Variant Handling

For details on the AUTOSAR variant handling support please refer to the *AUTOSAR Generic Structure Template* [5]. The structure is shown in figure 2.1.

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 HwPinGroupConnecors and HwPinConnectors 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 [5]. 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.

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

An overview of the applicable categories for `HwElement` is shown in table 3.1.

Category	Description
`Ecu`	Describes an Ecu (see section 3.1.1).
`ProcessingUnit`	Describes a micro-controller core (see section 3.1.2).
`MicroController`	Describes a micro-controller (see section 3.1.3).
`MemorySegment`	Describes a memory segment (see section 3.1.4).
`CommunicationController`	Describes a communication controller (see section 3.1.5).
`CommunicationTransceiver`	Describes a communication transceiver (see section 3.1.6).
`Digital`	Describes a digital IO peripheral (see section 3.1.7).
`Analog`	Describes an analog IO peripheral (see section 3.1.8).
`Timer`	Describes a timer peripheral (see section 3.1.9).
`Watchdog`	Describes a watchdog peripheral (see section 3.1.10).
`SensorActuator`	Describes sensors and actuators (see section 3.1.11).

Table 3.1: Hardware Element Categories

3.1.1 Ecu

The category of an ECU is defined as `Ecu`.

Currently no special attributes are defined for the ECU.

3.1.2 Processing Unit

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

The category 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.

The category 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

The special attributes which are applicable for the category `MemorySegment` hardware elements are defined in table 3.2.

Attribute	Required	Type / Unit	Description
memorySize	true	INTEGER	Specifies the size of the memory segment in bytes.
memoryType	true	ENUMERATION	Specifies the type of memory: • RAM • ROM • EEPROM • Flash

Table 3.2: `MemorySegment` Hardware Element Attributes

3.1.5 Communication Controller

The category of a communication controller is `CommunicationController`.

Attribute	Required	Type / Unit	Description
communication Controller Type	true	ENUMERATION	Specifies the type of communication controller: • CAN • TTCAN • LIN • FlexRay • Ethernet • Spi

Table 3.3: `CommunicationController` Hardware Element Attributes

3.1.6 Communication Transceiver

The category of a communication transceiver is defined as `CommunicationTransceiver`.

Attribute	Required	Type / Unit	Description
supports Disabling	false	BOOLEAN	Specifies whether the transceiver can be disabled.
supports WakeUp	false	BOOLEAN	Specifies whether the transceiver can indicate a wake-up situation on the bus.

Table 3.4: `CommunicationTransceiver` Hardware Element Attributes

3.1.7 Digital IO

The category 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

The category of an analog IO hardware element is defined as `Analog`.

Currently no special attributes are defined for the analog IO.

3.1.9 Timer

The category of a timer is defined as `Timer`.

Currently no special attributes are defined for the timer.

3.1.10 Watchdog

The category of a watchdog is defined as `Watchdog`.

Currently no special attributes are defined for the watchdog.

3.1.11 SensorActuator

The category of a sensor/actuator is defined as `SensorActuator`.

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

3.2 HwPinGroup categories

An overview of the applicable categories for `HwPinGroup` is shown in table 3.5.

Category	Description
`CommunicationPort`	Describes a communication connector (see section 3.2.1).

Table 3.5: Hardware Pin Group Categories

3.2.1 CommunicationPort

The category of a Communication Port is defined as `CommunicationPort`.

Attribute	Required	Type / Unit	Description
communication PortType	true	ENUMERATION	Specifies the type of communication port: • CAN • TTCAN • LIN • FlexRay • Ethernet • Spi

Table 3.6: `CommunicationPort` Hardware Element Attributes

3.3 HwPin categories

There are no dedicated categories specified for `HwPin`.

A Examples

A.1 Hardware Element

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

Example A.1

```
<AR-PACKAGE>
  <SHORT-NAME>VendorA</SHORT-NAME>
  <ELEMENTS>
    <HW-ELEMENT>
      <SHORT-NAME>MicroController_0815</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/MicroController</HW-
           CATEGORY-REF>
      </HW-CATEGORY-REFS>
    </HW-ELEMENT>
  </ELEMENTS>
</AR-PACKAGE>
```

A.2 Hierarchy of Hardware Elements

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

Example A.2

```
<AR-PACKAGE>
  <SHORT-NAME>VendorA</SHORT-NAME>
  <ELEMENTS>
    <HW-ELEMENT>
      <SHORT-NAME>MicroController_0815</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/MicroController</HW-
           CATEGORY-REF>
      </HW-CATEGORY-REFS>
      <NESTED-ELEMENTS>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA/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/ProcessingUnit</HW-
           CATEGORY-REF>
      </HW-CATEGORY-REFS>
    </HW-ELEMENT>
  </ELEMENTS>
</AR-PACKAGE>
```

A.3 HwPinGroups and HwPins

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

Example A.3

```
<AR-PACKAGE>
  <SHORT-NAME>VendorA</SHORT-NAME>
  <ELEMENTS>
    <HW-ELEMENT>
      <SHORT-NAME>MicroController_0815</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/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>
```

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).

Example A.4

```
<AR-PACKAGE>
  <SHORT-NAME>VendorA</SHORT-NAME>
  <ELEMENTS>
    <HW-ELEMENT>
      <SHORT-NAME>MicroController_0815</SHORT-NAME>
      <HW-CATEGORY-REFS>
        <HW-CATEGORY-REF DEST="HW-CATEGORY">/AUTOSAR/MicroController</HW-
            CATEGORY-REF>
        </HW-CATEGORY-REFS>
      <NESTED-ELEMENTS>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA/Core0</HW-ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA/Core1</HW-ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA/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/Core0</HW-ELEMENT-
                REF>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA/Mem01</HW-ELEMENT-
                REF>
          </HW-ELEMENT-REFS>
        </HW-ELEMENT-CONNECTOR>
        <HW-ELEMENT-CONNECTOR>
          <HW-ELEMENT-REFS>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA/Core0</HW-ELEMENT-
                REF>
            <HW-ELEMENT-REF DEST="HW-ELEMENT">/VendorA/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/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/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/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/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/MemorySegment</HW-
           CATEGORY-REF>
      </HW-CATEGORY-REFS>
    </HW-ELEMENT>
  </ELEMENTS>
</AR-PACKAGE>
```

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.

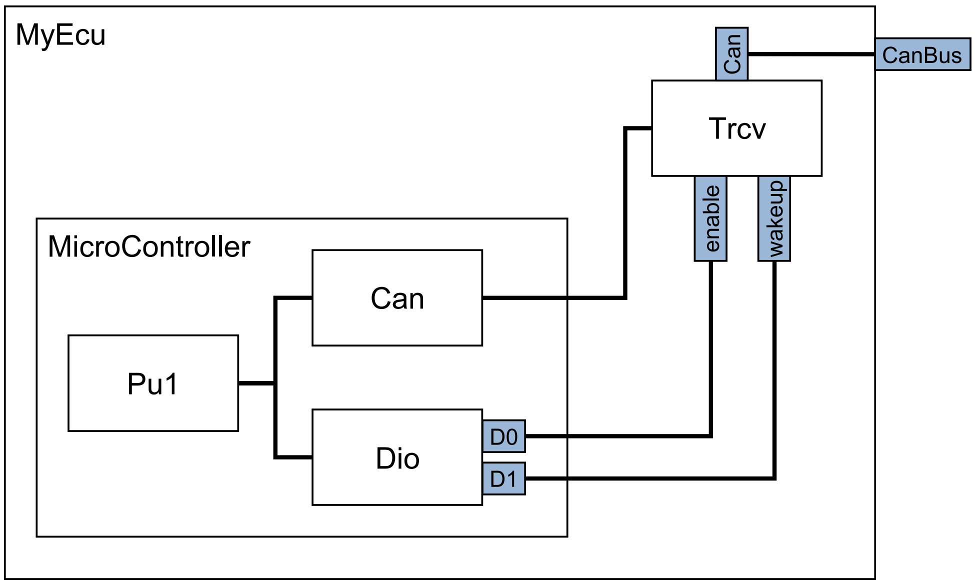

Figure A.1: Example of Ecu description

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.

Example A.5

```
<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">Pu1</HW-ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF
            DEST="HW-ELEMENT">Can</HW-ELEMENT-REF>
        </HW-ELEMENT-REF-CONDITIONAL>
        <HW-ELEMENT-REF-CONDITIONAL>
          <HW-ELEMENT-REF
            DEST="HW-ELEMENT">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">Pu1</HW-ELEMENT-REF>
            <HW-ELEMENT-REF
              DEST="HW-ELEMENT">Can</HW-ELEMENT-REF>
          </HW-ELEMENT-REFS>
        </HW-ELEMENT-CONNECTOR>
        <HW-ELEMENT-CONNECTOR>
          <HW-ELEMENT-REFS>
            <HW-ELEMENT-REF
              DEST="HW-ELEMENT">Pu1</HW-ELEMENT-REF>
            <HW-ELEMENT-REF
              DEST="HW-ELEMENT">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>
```

A.5.2 Transceiver description

The transceiver module is defined as a HwElement which provides three HwPin-Groups to describe its connectivity.

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

Example A.6

```
<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>
```

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.

Example A.7

```
<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/Can</HW-PIN-
                     GROUP-REF>
                <HW-PIN-GROUP-REF
                  DEST="HW-PIN-GROUP">/EcuVendor/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>
```

A.6 Attribute Definition

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

Example A.8

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

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).

Example A.9

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

B Glossary

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 ([7]).

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: `EcucParameterDef`, `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 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 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 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.

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 crates 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 implementing 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 Change History

C.1 Change History between AUTOSAR R4.0.1 against R3.1.5

The document and the MetaModel have been revised completely.

C.2 Change History between AUTOSAR R4.0.2 against R4.0.1

No changes to specification items.

C.3 Change History between AUTOSAR R4.0.3 against R4.0.2

C.3.1 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