Document Title	Modeling and Naming Aspects for Documentation, Measurement, and Calibration
Document Category	Technical report (TR)
Document UID	537
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	• Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• No content changes
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• No content changes
2021-11-25	R21-11	AUTOSAR Release Management	• No content changes
2020-11-30	R20-11	AUTOSAR Release Management	• No content changes
2019-11-28	R19-11	AUTOSAR Release Management	• No content changes • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Editorial changes

▽

△

2016-11-30	4.3.1	AUTOSAR Release Management	• Editorial changes
2015-07-31	4.3.0	AUTOSAR Release Management	• Editorial changes
2014-10-31	4.2.2	AUTOSAR Release Management	• Minor changes
2013-10-31	4.2.1	AUTOSAR Release Management	• P/L-List now also available as .arxml as part of MOD_AISpecification [1]
2013-03-15	4.1.3	AUTOSAR Administration	• Minor changes
2013-03-15	4.1.1	AUTOSAR Administration	• Documentation issues included • Additional data prototypes considered besides data in port and parameter interfaces • Example for flat map with arrays added • Refined name space concept in accordance to meta model (e.g. usage of symbolic names)
2011-12-22	4.0.3	AUTOSAR Administration	• 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 Scope 6

2 How to read this document 7
- 2.1 Conventions used . . . 7
- 2.2 Numbering of Rules . . . 8
- 2.3 Acronyms and abbreviations . . . 10

3 Related Documentation 11

4 Requirements 12

5 Traceability 14

6 Methodological Background 15
- 6.1 SwCalibrationAccess . . . 15
- 6.2 FlatMap for Measurement and Calibration Data . . . 16
- 6.3 AliasNameSet for Display Names . . . 21
- 6.4 Unique Symbolic Names for SW Component Types et. al. . . . 22
- 6.5 FlatMap for Unique Names of SW Component Prototypes . . . 23
- 6.6 Virtual Name Spaces . . . 23
- 6.7 References in Documentation . . . 24
- 6.8 Instance cp- and pb-paths . . . 29

7 Overall Modeling and Generation Rules 31
- 7.1 Name Part Length . . . 33

8 Data Prototypes 34

9 Data Prototypes in DataInterfaces 36
- 9.1 Increase Readability of Display Names . . . 38
- 9.2 VariableDataPrototypes in SenderReceiverInterfaces . . . 40
- 9.3 ParameterDataPrototypes in ParameterInterface . . . 41

10 Data Prototypes in ClientServerInterfaces 43
- 10.1 Data Prototypes within InternalBehavior . . . 43

11 Name Space 45

12 SwSystemconsts 48

13 PortPrototypeBlueprints 49

14 ComponentHierarchy 52

15 SwComponentPrototypes 56

16 Unique SW-Signal Display Names 57

A Appendix: Keywords Phys/Log (P/L-List) for Powertrain Domain 61

B Change history of AUTOSAR traceable items 62
B.1 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 62
B.1.1 Added Specification Items in R26-11 . . . 62
B.1.2 Changed Specification Items in R26-11 . . . 62
B.1.3 Deleted Specification Items in R26-11 . . . 62
B.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 62
B.2.1 Added Specification Items in R25-11 . . . 62
B.2.2 Changed Specification Items in R25-11 . . . 62
B.2.3 Deleted Specification Items in R25-11 . . . 62
B.3 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 63
B.3.1 Added Specification Items in R24-11 . . . 63
B.3.2 Changed Specification Items in R24-11 . . . 63
B.3.3 Deleted Specification Items in R24-11 . . . 63

1 Scope

This document is specific for the powertrain domain. For the powertrain domain efficient handling of calibration and measurement is of great importance.

This document gives some methodological background on calibration relevant issues of AUTOSAR independent of any application domain.

Examples are provided. Although they might resemble standardized application interfaces they are simply to be seen as examples[1].

The main focus of this document, however, is to give a proposal how to automatically generate display names[2] for measurement, calibration and diagnostic tools (MCD). If some assumptions and specific modeling rules are fulfilled then automatic generation leads to adequate display names. Of course, manual definition of display names is still possible and in cases the generated name is not suitable even necessary.

This document does not describe a complete mapping to MCD-2 MC (A2L, [2]). Its current focus is on variable and parameter prototypes of sender receiver and parameter interfaces as well as system constants. Local measurables and calibration parameters as well as other interface types are not (yet) considered.

This document does not contain rules for modeling and naming of AUTOSAR elements in general. This is already covered by [3].

The XML code which is shown in the document is compliant to the AUTOSAR xsd of Release 4.0.

The algorithms described in this document where applied to the display names part of [4].

[1] E.g. the paths are not correct: normally it is /AUTOSAR/AISpecification/Units ([1]) etc. but in this document for better readability the paths were shortened to /AUTOSAR/Units a.s.o..

[2] Please note when using A2L [2]: the display name as used in this document is not identical to the DISPLAY_IDENTIFIER for A2L but identical to the attribute Name itself.

2 How to read this document

2.1 Conventions used

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as follows.

Note that the requirement level of the document in which they are used modifies the force of these words.

- MUST: This word, or the adjective “LEGALLY REQUIRED”, means that the definition is an absolute requirement of the specification due to legal issues.
- MUST NOT: This phrase, or the phrase “MUST NOT”, means that the definition is an absolute prohibition of the specification due to legal issues.
- SHALL: This phrase, or the adjective “REQUIRED”, means that the definition is an absolute requirement of the specification.
- SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective “RECOMMENDED”, means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase “NOT RECOMMENDED”, means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective “OPTIONAL”, means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item.

An implementation, which does not include a particular option, SHALL be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, SHALL be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides).

2.2 Numbering of Rules

All rules or recommendations are identified by an ID. See [5] [TPS_STDT_00042] for name pattern of such identifiers and [5] Table 2.1 for the predefined names including the ones used in this document.

The ID starts with "TR_MCM" for the Modeling Rules for Measurement and Calibration followed by five digits ([TR_MCM_xxxxx]).

The ID starts with "TR_MCG" for the Generation Rules for Measurement and Calibration followed by five digits ([TR_MCG_xxxxx]).

The ID starts with "TR_MCA" for the Assumptions made for being able to use specific generation rules ([TR_MCA_xxxx]). They are introduced for easier stating which assumptions are true for your project (line) and then to decide which rules (TR_MCR) are applicable.

The ID starts with "TR_MCR" for the Requirements for Measurement and Calibration followed by five digits ([RS_MCR_xxxxx]).

Terms taken from the Meta model like e.g. AutosarDataPrototype are written in italic.

The following table gives an overview of renumbered or deleted rules:

New	Old	Comment
[RS_MCR_9{0}$_{0..n}$<number>]	[MCR<number>]	For avoiding same numbers for requirements and rules etc. [5]
[TR_MCM_7{0}$_{0..n}$<number>]	[MCM<number>]	For avoiding same numbers for modeling rules from generation rules etc. [5]
[TR_MCA_8{0}$_{0..n}$<number>]	[MCA<number>]	For avoiding same numbers for assumptions and rules etc. [5]
[TR_MCA_80789]	[MCA030b]	Renumbered according [TPS_STDT_00042] in [5]
[TR_MCG_00788]	[MCG005b]	Renumbered according [TPS_STDT_00042] in [5]
RS_MCR_90781	[MCR010a]	Renumbered according [TPS_STDT_00042] in [5]
RS_MCR_90782	[MCR010b]	Renumbered according [TPS_STDT_00042] in [5]
RS_MCR_90783	[MCR013a]	Renumbered according [TPS_STDT_00042] in [5]
RS_MCR_90784	[MCR013b]	Renumbered according [TPS_STDT_00042] in [5]
[TR_MCG_00787]	[MCG710]	Because same number as in [TR_MCA_80710]
changed	[TR_MCM_70030]	
changed	[TR_MCG_00035]	
changed	[TR_MCM_70022]	
changed	[TR_MCG_00770]	
–	TR_MCG_00712	Implicitly contained in other rules
–	TR_MCG_00715	Implicitly contained in other rules
–	TR_MCG_00400	Implicitly contained in rule [TR_MCG_00021]

For formulating the rules the used syntax follows the one in [5]. However, some additional placeholders were "invented" to be able to express the intended rules.

The additionally used placeholders are listed in the following:

- componentDescriptor

- componentHierarchy
- data
- dataInfo
- element
- operationInfo
- port
- portBlueprint
- portBlueprintDescriptor
- systemDescriptor
- swSignal

The name of a concrete identifiable is presented as (identifiable).name, e.g. Port-Prototype.name.

2.3 Acronyms and abbreviations

Abbreviation / Acronym	Description
AI	Application interfaces
component	components are the SwComponentPrototypes of a CompositionSwComponentType
cp-path	SwComponentPrototype-Path, see chapter 6.3
dataElement or data element	Element of a port interface of type AutosarDataPrototype, dataElements are VariableDataPrototypes of a PortInterface
data prototype	An AutosarDataPrototype like VariableDataPrototype or ParameterDataPrototype etc.
Descriptor	<X>Descriptor: Short form for "ShortName FlatInstanceDesriptor in a FlatMap of target element x of element type X"
Element	Element might be a prototype element like SwComponentPrototype, AutosarDataPrototype etc. or an ARElement
MCD	Measurement, Calibration and Diagnostic
name	Used as short form for ShortName if not mentioned otherwise
parameter	Element of a ParameterInterface of type ParameterDataPrototype, parameters are the ParameterDataPrototypes of a PortInterface
pb-path	PortPrototypeBlueprint-Path, see chapter 6.3
P/L-list	List with abbreviations for physical and logical types. See chapter A.
port	port prototype or port prototype blueprint - depending on context
SW	Software
SW-C	Software Component
virtual name space	In this document we use the term "virtual name space for {element} or {prototype}" to denote that the names of the corresponding elements are unique within the corresponding ARPackage, recursively. See chapter6.6.

Table 2.1: Acronyms and abbreviations used in the scope of this Document

3 Related Documentation

[1] XML Specification of Application Interfaces
AUTOSAR_CP_MOD_AISpecification

[2] ASAM MCD-2 MC (ASAP2 / A2L)
http://www.asam.net
ASAM_AE_MCD-2_MC_BS_V1-7-1.pdf

[3] SW-C and System Modeling Guide
AUTOSAR_CP_TR_SWCModelingGuide

[4] Explanation of Application Interfaces of the Powertrain Engine Domain
AUTOSAR_CP_EXP_AIPowertrain

[5] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[6] Software Component Template
AUTOSAR_CP_TPS_SoftwareComponentTemplate

[7] Specification of RTE Software
AUTOSAR_CP_SWS_RTE

[8] System Template
AUTOSAR_CP_TPS_SystemTemplate

[9] Generic Structure Template
AUTOSAR_FO_TPS_GenericStructureTemplate

4 Requirements

Here you find a listing of the requirements together with a short description.

RS_MCR_90000 For display names a global name space within one ECU is sometimes required by Measurement and Calibration Tools or Calibration Data Exchange Formats like for example A2L. Therefore this shall be supported.

RS_MCR_90001 For one calibration team at OEM or Supplier site: The display name of a software signal shall be stable and identical in all systems to be calibrated by the team independent of the project, the project configuration or the underlying product or product line software architecture.

RS_MCR_90002 There shall be a default derivable display name.

RS_MCR_90003 Automatic generation of display names shall be possible.

RS_MCR_90004 Automatic generation/derivation of model element names from display names should be possible.

RS_MCR_90005 It should be possible but the exception in practice to manually define a display name for a data prototype. See also requirements for automatic generation and default display names.

RS_MCR_90006 A display name should be kept as short as possible. It should not exceed 31 characters. In the best case a display name would not exceed 16 characters.

In case of multiple instances the name length can be extended to ensure uniqueness.

The current restrictions of MCD tools shall be considered.

RS_MCR_90008 In case of multiple instantiation it shall be ensured that instances with the same semantic meaning have the same display names in projects with different numbers of instances.

RS_MCR_90009 In case the MCD tools do not provide an easy-to-use mechanism for identification of maps, curves etc. the corresponding information should be part of the display name itself.

RS_MCR_90781 If elements belong together it should be possible to sort them alphabetically.

RS_MCR_90782 For elements belonging together it should be possible to sort subgroups according to their physical or logical meaning.

RS_MCR_90011 The naming convention shall consider a number of about 20000 calibration relevant names per project. Thus readability is an important requirement.

RS_MCR_90783 Modeling Rules or Recommendations for Measurement and Calibration shall not violate the general "shall"-rules for naming as defined in the Meta Model.

RS_MCR_90784 Modeling Rules or Recommendations for Measurement and Calibration shall not violate the general "shall"-rules for naming as defined in [3].

RS_MCR_90014 The display names should be useable also for legacy systems to implement global variables.

RS_MCR_90015 Different data should have different display names.

A2L e.g. does not explicitly forbid having the same display identifier for different data. However, to use this mechanism can lead to confusion and seems not to have any advantages.

RS_MCR_90016 One and the same software signal should only have exactly one display name in one calibration project.

5 Traceability

The following table references the requirements specified in chapter 4 and links to the fulfillments of these.

Requirement	Satisfied by
RS_MCR_90000	[TR_MCG_00790] [TR_MCG_00791] [TR_MCM_70002]
RS_MCR_90001	[TR_MCM_70040]
RS_MCR_90002	[TR_MCG_00005] [TR_MCG_00010] [TR_MCG_00015] [TR_MCG_00020] [TR_MCG_00021] [TR_MCG_00030] [TR_MCG_00033] [TR_MCG_00035] [TR_MCG_00040] [TR_MCG_00045] [TR_MCG_00070] [TR_MCG_00080] [TR_MCG_00090] [TR_MCG_00120] [TR_MCG_00310] [TR_MCG_00320] [TR_MCG_00330] [TR_MCG_00340] [TR_MCG_00350] [TR_MCG_00360] [TR_MCG_00510] [TR_MCG_00512] [TR_MCG_00520] [TR_MCG_00610] [TR_MCG_00705] [TR_MCG_00706] [TR_MCG_00713] [TR_MCG_00716] [TR_MCG_00740] [TR_MCG_00750] [TR_MCG_00752] [TR_MCG_00760] [TR_MCG_00770] [TR_MCG_00780] [TR_MCG_00785] [TR_MCG_00786] [TR_MCG_00787] [TR_MCG_00788] [TR_MCG_00790] [TR_MCG_00791]
RS_MCR_90003	[TR_MCG_00005] [TR_MCG_00010] [TR_MCG_00015] [TR_MCG_00020] [TR_MCG_00021] [TR_MCG_00030] [TR_MCG_00033] [TR_MCG_00035] [TR_MCG_00040] [TR_MCG_00045] [TR_MCG_00070] [TR_MCG_00080] [TR_MCG_00090] [TR_MCG_00120] [TR_MCG_00310] [TR_MCG_00320] [TR_MCG_00330] [TR_MCG_00340] [TR_MCG_00350] [TR_MCG_00360] [TR_MCG_00510] [TR_MCG_00512] [TR_MCG_00520] [TR_MCG_00610] [TR_MCG_00705] [TR_MCG_00706] [TR_MCG_00713] [TR_MCG_00716] [TR_MCG_00740] [TR_MCG_00750] [TR_MCG_00752] [TR_MCG_00760] [TR_MCG_00770] [TR_MCG_00780] [TR_MCG_00785] [TR_MCG_00786] [TR_MCG_00787] [TR_MCG_00788] [TR_MCG_00790] [TR_MCG_00791]
RS_MCR_90004	[TR_MCG_00785] [TR_MCG_00786]
RS_MCR_90005	[TR_MCG_00004]
RS_MCR_90006	[TR_MCA_80020] [TR_MCA_80030] [TR_MCA_80530] [TR_MCA_80710] [TR_MCA_80715] [TR_MCA_80720] [TR_MCA_80730] [TR_MCA_80789] [TR_MCG_00010] [TR_MCG_00015] [TR_MCG_00020] [TR_MCG_00030] [TR_MCG_00045] [TR_MCG_00120] [TR_MCG_00713] [TR_MCG_00740] [TR_MCG_00750] [TR_MCG_00752] [TR_MCG_00790] [TR_MCG_00791] [TR_MCM_70020] [TR_MCM_70022] [TR_MCM_70030] [TR_MCM_70040] [TR_MCM_70390]
RS_MCR_90008	[TR_MCA_80720] [TR_MCA_80730] [TR_MCG_00004]
RS_MCR_90009	[TR_MCG_00090] [TR_MCG_00310] [TR_MCG_00320] [TR_MCG_00330] [TR_MCG_00340] [TR_MCG_00350] [TR_MCG_00510] [TR_MCG_00520] [TR_MCG_00790] [TR_MCG_00791]
RS_MCR_90011	[TR_MCG_00010] [TR_MCG_00015] [TR_MCG_00020] [TR_MCG_00030] [TR_MCG_00045] [TR_MCG_00070] [TR_MCG_00090] [TR_MCG_00120] [TR_MCG_00310] [TR_MCG_00320] [TR_MCG_00330] [TR_MCG_00340] [TR_MCG_00350] [TR_MCG_00713] [TR_MCG_00740] [TR_MCG_00750] [TR_MCG_00752] [TR_MCM_70060]
RS_MCR_90014	[TR_MCG_00001] [TR_MCG_00790] [TR_MCG_00791]
RS_MCR_90015	[TR_MCA_80034] [TR_MCG_00005] [TR_MCG_00785] [TR_MCG_00786] [TR_MCG_00787] [TR_MCG_00788] [TR_MCG_00790] [TR_MCG_00791]
RS_MCR_90016	[TR_MCG_00785] [TR_MCG_00786] [TR_MCG_00787] [TR_MCG_00790] [TR_MCG_00791]
RS_MCR_90781	[TR_MCM_70070]
RS_MCR_90782	[TR_MCM_70050] [TR_MCM_70065]
RS_MCR_90783	[TR_MCM_70005]
RS_MCR_90784	[TR_MCM_70010] [TR_MCM_70060]

Table 5.1: Requirements Tracing

6 Methodological Background

The following chapters give background information on aspects of the AUTOSAR metal model and methodology. They can be skipped except for chapter 6.6 and 6.7 that introduce some new aspects that are important for understanding the following topics of the document.

6.1 SwCalibrationAccess

Display Names have to be provided for all instances of AutosarDataPrototypes for which the attribute SwCalibrationAccess is set via its SwDataDefProps to "readOnly" or "readWrite".

The information SwCalibrationAccess is mandatory for ApplicationDataTypes but can be overwritten. For example the SwCalibrationAccess can be overwritten in AutosarDataPrototypes or FlatInstanceDescriptor.swDataDefProps a.s.o. (see [constr_1015] in [6], [SWS_Rte_07196] in [7]).

In the current standardization of application interfaces ([1]) the information SwCalibrationAccess of ApplicationDataTypes is set to ReadOnly.

Example:

```
...
<VARIABLE-DATA-PROTOTYPE>
<SHORT-NAME>EngN</SHORT-NAME>
<TYPE-TREF DEST="APPLICATION-DATA-TYPE">/OEM1/ApplicationDataTypes/N1</TYPE
    -TREF>
<SW-DATA-DEF-PROPS>
  ...
  <SW-CALIBRATION-ACCESS>NOT-ACCESSIBLE</SW-CALIBRATION-ACCESS>
  ...
</SW-DATA-DEF-PROPS>
</VARIABLE-DATA-PROTOTYPE>
...
```

overwrites

```
...
<APPLICATION-PRIMITIVE-DATA-TYPE>
<SHORT-NAME>N1</SHORT-NAME>
<CATEGORY>VALUE</CATEGORY>
<SW-DATA-DEF-PROPS>
  <SW-DATA-DEF-PROPS-VARIANTS>
    <SW-DATA-DEF-PROPS-CONDITIONAL>
      ...
      <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
      ...
    </SW-DATA-DEF-PROPS-CONDITIONAL>
  </SW-DATA-DEF-PROPS-VARIANTS>
</SW-DATA-DEF-PROPS>
...
```

```
</APPLICATION-PRIMITIVE-DATA-TYPE>
```

6.2 FlatMap for Measurement and Calibration Data

Whereas SystemSignals serve for inter-ECU-communication, software signals are implicitly used within ECU-internal communication. There is no model element called software signal because they are represented by instance references (InstanceRef) to an AutosarDataPrototype. However, for calibration engineers there is the need to have unique names for software signals similar as there is the need to have unique names for inter-ECU-communication. These unique names for software signals are identical to the display names that are discussed in this document.

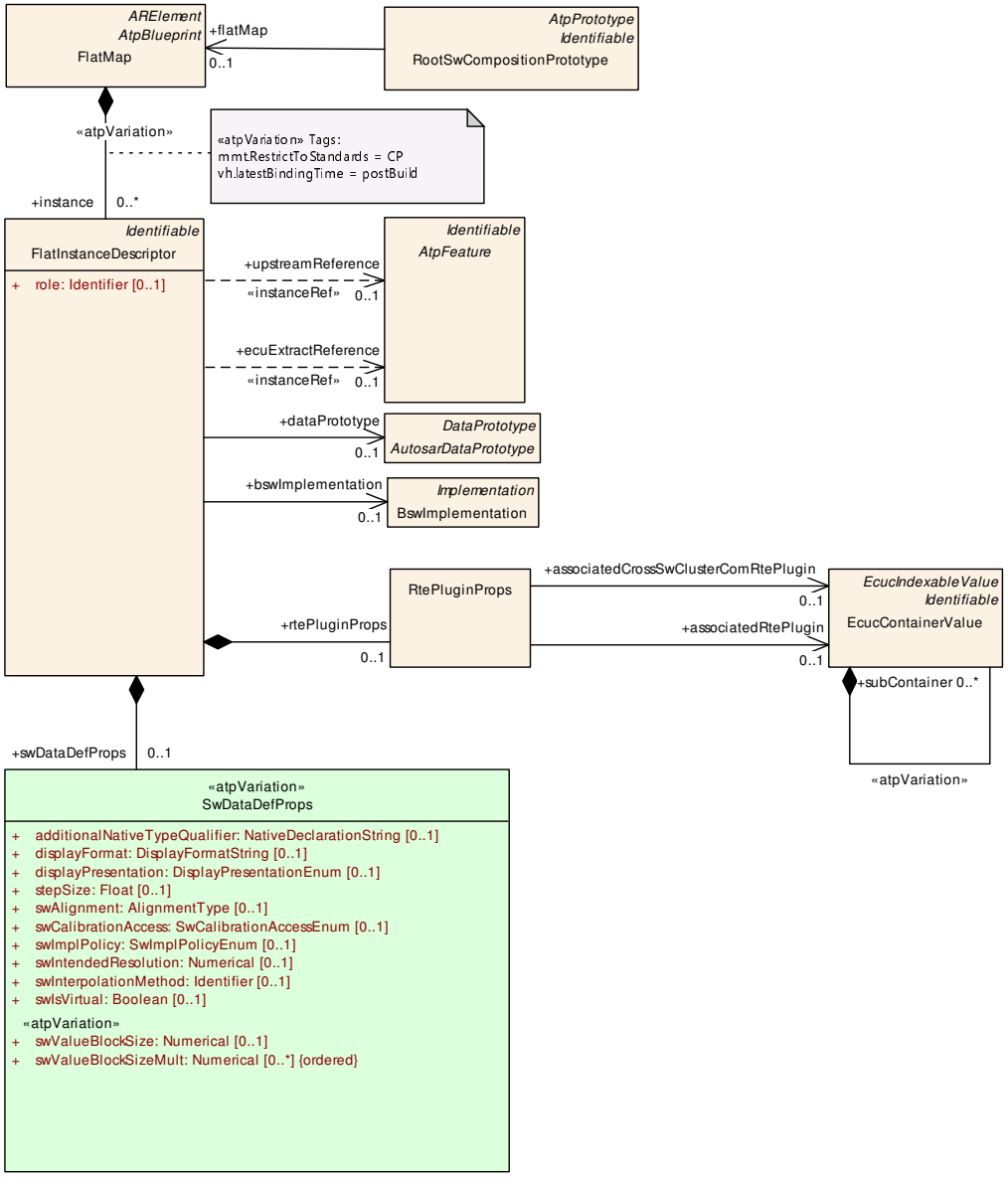


Figure 6.1: FlatMap

The AUTOSAR methodology provides means to specify unique names e.g. to be used to generate A2L-files out of it [2]. This is done via so-called FlatMaps (see Figure 6.1). The FlatMap consists of several FlatInstanceDescriptors. In our context a FlatInstanceDescriptor represents exactly one VariableDataPrototype or a ParameterDataPrototype and attaches a unique name (shortName) to it. This shortName can later be used as the display name of the software signal. In [8] the following mapping is recommended to A2L:

FlatInstanceDescriptor.shortName ->

MEASUREMENT Name for VariableDataPrototypes

CHARACTERISTIC Name for ParameterDataPrototypes

In [7] Chapter 4.2.9 *"Measurement and Calibration"*, it is described how the RTE deals with Measurement and Calibration.

[TR_MCM_70002] ⌈For display names a global name space within one ECU is modeled as a FlatMap.

Fulfills requirements: RS_MCR_90000⌋

An instance of a *FlatMap* containing the definition of display names for a set of software signals is an XML-artifact which can be represented by a single file. Depending on its scope, it is attached (via reference) to the top- level-composition of a system description or to the top-level-composition (*RootSwCompositionPrototype*) of an ECU-extract.

The mapping table (`FlatMap`) can be maintained manually or the display names may be automatically generated. Of course, also a mix of both approaches is possible. This mix is supported by *<<atpSplitable>>*.

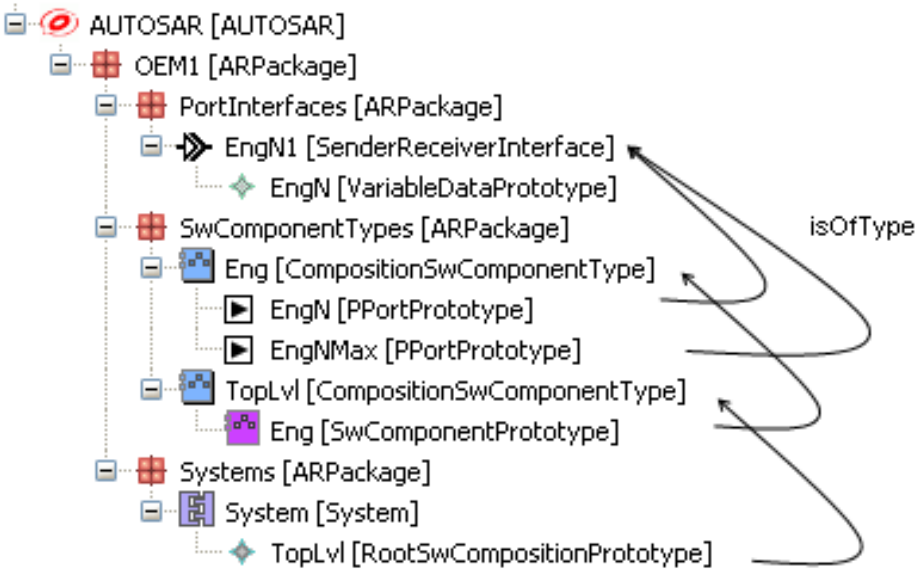


Figure 6.2: Example AUTOSAR System EngN

Example:

Figure 6.2 shows an example of an AUTOSAR system. In the following a FlatMap for the software signal with the display name "Eng_n" is described in arxml:

```
<AR-PACKAGE>
  <SHORT-NAME>OEM1</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>FlatMaps</SHORT-NAME>
      <ELEMENTS>
        <FLAT-MAP>
          <SHORT-NAME>OEMMap</SHORT-NAME>
          <INSTANCES>
            <FLAT-INSTANCE-DESCRIPTOR>
              <SHORT-NAME>Eng_n</SHORT-NAME>
              <UPSTREAM-REFERENCE-IREF>
                <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/
                    OEM1/Systems/System/TopLvl</CONTEXT-ELEMENT-REF>
                <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/OEM1/
                    SwComponentTypes/TopLvl/Eng</CONTEXT-ELEMENT-REF>
                <CONTEXT-ELEMENT-REF DEST="PORT-PROTOTYPE">/OEM1/
                    SwComponentTypes/Eng/EngN</CONTEXT-ELEMENT-REF>
                <TARGET-REF DEST="VARIABLE-DATA-PROTOTYPE">/OEM1/
                    PortInterfaces/EngN1/EngN</TARGET-REF>
              </UPSTREAM-REFERENCE-IREF>
            </FLAT-INSTANCE-DESCRIPTOR>
          </INSTANCES>
        </FLAT-MAP>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AR-PACKAGE>
...
```

For array types an index has to be added if only a single element out of the array is intended to be referenced. Example (no figure) for the first array element (INDEX: "0") of an array:

```
<FLAT-INSTANCE-DESCRIPTOR>
  <SHORT-NAME>Esc_vWhlInd_0</SHORT-NAME>
  <UPSTREAM-REFERENCE-IREF>
    <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/OEM1/Systems
        /System/TopLvl</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/OEM1/
        SwComponentTypes/TopLvl/Pt</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="PORT-PROTOTYPE">/OEM1/SwComponentTypes/Pt/
        EscVWhlInd</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/OEM1/
        PortInterfaces/WhlSpdCircuml1/WhlSpdCircuml</CONTEXT-ELEMENT-REF>
    <TARGET-REF DEST="APPLICATION-ARRAY-ELEMENT" INDEX="0">/OEM1/
        ApplicationDataTypes/WhlSpdCircumlPerWhl1</TARGET-REF>
  </UPSTREAM-REFERENCE-IREF>
</FLAT-INSTANCE-DESCRIPTOR>
```

When only standardizing PortPrototypeBlueprints and not the software components themselves the display names can only be standardized by using the reference to the PortPrototypeBlueprint. In this case the display names are themselves blueprints and are used as basis for deriving display names of port prototypes that were derived from the port prototype blueprint.

For multiple instantiated components the display names cannot be completely standardized because in general neither the number of instances nor the instance names are known in advance but only on ECU level.

Due to [constr_2528] in [5] an element referencing blueprints has to be a blueprint itself or a BlueprintMap a FlatMap defining display names for blueprints is also a blueprint (i.e. it must be element of an ARPackage of Category Blueprint).

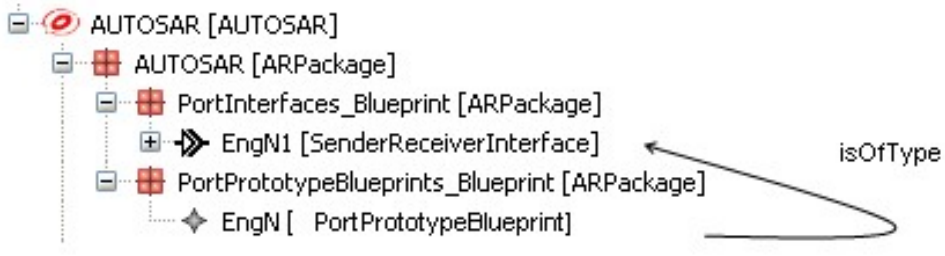


Figure 6.3: Example AUTOSAR Blueprints EngN

Example, see Figure 6.3:

Assume the port prototype blueprint "EngN" refers to port interface "EngN1". "EngN1" might be a blueprint itself. For a blueprint the NamePattern has to be added. This is done for the complete FlatMap but also for its single FlatInstanceDescriptors. In this example we know that there is no multiple instantiation for this

blueprint. Therefore the NamePattern is equal to the ShortName of the descriptor ({blueprintName}).

Additionally a BlueprintCondition is introduced to denote that e.g. not all standardized application interfaces are present in every system. Thus, it is allowed to derive a FlatMap that contains fewer elements than the blueprint FlatMap plus additional ones.

```
<AR-PACKAGE>
  <SHORT-NAME>AUTOSAR</SHORT-NAME>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>FlatMaps_Blueprint</SHORT-NAME>
      <CATEGORY>BLUEPRINT></CATEGORY>
      <ELEMENTS>
        <FLAT-MAP>
          <SHORT-NAME NAME-PATTERN="{anyName}">ARMap</SHORT-NAME>
          <INSTANCES>
            <FLAT-INSTANCE-DESCRIPTOR>
              <SHORT-NAME NAME-PATTERN="{blueprintName}">AR_Eng_n</SHORT-
                NAME>
              <LONG-NAME><L-4 L="EN">Actual Engine Speed</L-4></LONG-NAME>
```

```
            <UPSTREAM-REFERENCE-IREF>
              <CONTEXT-ELEMENT-REF DEST="PORT-PROTOTYPE-BLUEPRINT">/
                 AUTOSAR/PortPrototypeBlueprints_Blueprint/EngN</CONTEXT-
                 ELEMENT-REF>
              <TARGET-REF DEST="VARIABLE-DATA-PROTOTYPE">/AUTOSAR/
                 PortInterfaces_Blueprint/EngN1/EngN</TARGET-REF>
            </UPSTREAM-REFERENCE-IREF>
            <VARIATION-POINT>
              <SHORT-LABEL NAME-PATTERN="{blueprintName}">VP1</SHORT-
                 LABEL>
              <BLUEPRINT-CONDITION>
                <P><L-1 L="EN">used only if port prototypes are derived
                   from this blueprint</L-1></P>
                <P><L-1 L="EN">The condition swSyscond has to be
                   implemented in the derived element (Upcoming)</L-1></P
                   >
              </BLUEPRINT-CONDITION>
              <SW-SYSCOND>undefined</SW-SYSCOND>
            </VARIATION-POINT>
          </FLAT-INSTANCE-DESCRIPTOR>
        </INSTANCES>
      </FLAT-MAP>
    </ELEMENTS>
  </AR-PACKAGE>
  ...
```

The `FlatMap` presented earlier might so have been derived from this blueprint `FlatMap`, formally described as:

```
<AR-PACKAGE>
  <SHORT-NAME>BlueprintMappingSets</SHORT-NAME>
  <ELEMENTS>
    <BLUEPRINT-MAPPING-SET>
    <SHORT-NAME>OEM1BlueprintMapSet</SHORT-NAME>
    <BLUEPRINT-MAPS>
      <BLUEPRINT-MAPPING>
        <BLUEPRINT-REF DEST="FLATMAP">/AUTOSAR/Flatmaps_Blueprint/ARMap</
           BLUEPRINT-REF>
        <DERIVED-OBJECT-REF DEST="FLAT-MAP">/OEM1/Flatmaps/OEMMap</DERIVED-
           OBJECT-REF>
      </BLUEPRINT-MAPPING>
      <BLUEPRINT-MAPPING>
        <BLUEPRINT-REF DEST="PORT-PROTOTYPE-BLUEPRINT">/AUTOSAR/
           PortPrototypeBlueprints_Blueprint/EngN</BLUEPRINT-REF>
        <DERIVED-OBJECT-REF DEST="P-PORT-PROTOTYPE">/OEM1/SwComponentTypes/
           Eng/EngN</DERIVED-OBJECT-REF>
      </BLUEPRINT-MAPPING>
      <BLUEPRINT-MAPPING>
        <BLUEPRINT-REF DEST="SENDER-RECEIVER-INTERFACE">/AUTOSAR/
           PortInterfaces_Blueprint/EngN1</BLUEPRINT-REF>
        <DERIVED-OBJECT-REF DEST="SENDER-RECEIVER-INTERFACE">/OEM1/
           PortInterfaces/EngN1</DERIVED-OBJECT-REF>
      </BLUEPRINT-MAPPING>
    </BLUEPRINT-MAPPING-SET>
  </ELEMENTS>
</AR-PACKAGE>
```

6.3 AliasNameSet for Display Names

An AliasNameAssignment (see Figure 6.4) can be used to associate an alternative name to a flat instance descriptor or an Identifiable [8].

In [8] the following mapping is recommended to A2L [2]:

AliasNameAssignment.shortLabel ->

MEASUREMENT -> DISPLAY_IDENTIFIER for VariableDataPrototypes

CHARACTERISTIC -> DISPLAY_IDENTIFIER for ParameterDataPrototypes

but

SYSTEM_CONSTANT -> Name for SwSystemconsts

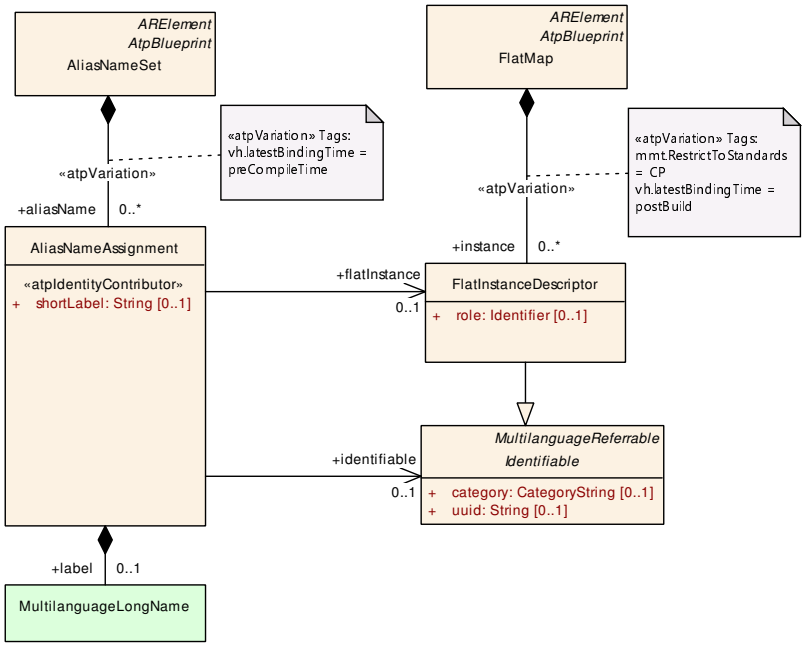


Figure 6.4: AliasNameAssignment

Example for system constants:

```
<ALIAS-NAME-ASSIGNMENT>
  <SHORT-LABEL>ENGNRCYL_SC</SHORT-LABEL>
  <IDENTIFIABLE-REF DEST="SW-SYSTEMCONST">/OEM1/SwSystemconsts/EngNrCyl</
     IDENTIFIABLE-REF>
</ALIAS-NAME-ASSIGNMENT>
```

Since FlatMaps do not allow referencing SwSystemconsts the alias mechanism is used to specify unique display names for SwSystemconsts.

Aliases or DISPLAY_IDENTIFIER are not supported for SwSystemconsts, neither in AUTOSAR nor in ASAM MCD-2MC [2].

AliasNameAssignments are also needed if r- and p-ports both have an entry in the FlatMap but shalll have the same name in the generated A2L-file. Within the FlatMap

the names of the r- and p-ports have to be different because of the name space the FlatMap defines. In general the ShortName of the FlatInstanceDescriptor of the p-port is used as alias name for the r-ports the p-port is connected with.

Additionally this mechanism is used if no unique display names can be generated but only a part of a display name that is to be extended for uniqueness. This might be the case for PortPrototypeBlueprints.

Example for blueprinted alias of an already existing flat instance descriptor that was generated automatically e.g. and now is intended to be "overwritten":

```
<ALIAS-NAME-ASSIGNMENT>
  <SHORT-LABEL NAME-PATTERN="{blueprintName}">Esc_vWhlIndFrntLe</SHORT-
     LABEL>
  <FLAT-INSTANCE-REF DEST="FLAT-INSTANCE-DESCRIPTOR">/AUTOSAR/Flatmaps/
     ARMap/Esc_vWhlInd_0</FLAT-INSTANCE-REF>
  ...
</ALIAS-NAME-ASSIGNMENT>
```

This document does not contain rules for display names of Units. Nevertheless here some hints w.r.t. units. The units are standardized in [1], resp. For Units the meta model provides an explicit DisplayName Tag [6].

We have ([8] and [6])

AliasNameAssignment.shortLabel ->

UNIT -> Name for Units

Unit.DisplayName

UNIT -> Display for Units

Note that the usage of the AliasNameSet is not defined within AUTOSAR. It is intended to be used by the A2L - generator which is not standardized by AUTOSAR. Nevertheless [8] gives some recommendations how to make the mapping.

6.4 Unique Symbolic Names for SW Component Types et. al.

In some cases the RTE requires unique names although elements are contained in different name spaces to avoid name clashes in c code. For these cases the concept of symbol properties (SymbolProps) as special case of ImplementationProps was introduced [6].

Symbol properties are only available for atomic software component types because compositions are flattened and not visible in c code.

```
<APPLICATION-SW-COMPONENT-TYPE>
  <SHORT-NAME>TMdl</SHORT-NAME>
  <SYMBOL-PROPS>
    <SHORT-NAME>AR_TMdl</SHORT-NAME>
    <SYMBOL>AR_TMdl</SYMBOL>
  </SYMBOL-PROPS>
```

```
</APPLICATION-SW-COMPONENT-TYPE>
```

Symbol properties are also available for other elements like for example RunnableEntity and ImplementationDataType.

6.5 FlatMap for Unique Names of SW Component Prototypes

Within the ECU extract of a system the FlatMap entries are needed to specify unique names for component prototypes [8].

Example:

The example specifies a unique name for the component prototype "Eng". Please note that the example from Figure 2 was extended by an additional component hierarchy "Pt" to make example more real.

```
<FLAT-MAP>
  <SHORT-NAME>ARMap</SHORT-NAME>
  <INSTANCES>
    <FLAT-INSTANCE-DESCRIPTOR>
      <SHORT-NAME>Eng</SHORT-NAME>
      <UPSTREAM-REFERENCE-IREF>
        <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/OEM1/
            Systems/System/TopLvl</CONTEXT-ELEMENT-REF>
        <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/OEM1/
            SwComponentTypes/TopLvl/Pt</CONTEXT-ELEMENT-REF>
        <TARGET-REF DEST="SW-COMPONENT-PROTOTYPE">/OEM1/SwComponentTypes/Pt
            /Eng</TARGET-REF>
      </UPSTREAM-REFERENCE-IREF>
    </FLAT-INSTANCE-DESCRIPTOR>
  </INSTANCES>
</FLAT-MAP>
```

6.6 Virtual Name Spaces

In this document we use the term "virtual name space for {element}" to denote that the name of all elements of kind {element} are unique within a specific ARPackage. This means that for all ARPackages being subpackages of this package as well as other elements of the meta model like SwComponentTypes that define their own name spaces according to the meta model this property of being able to define its own name space is ignored.

It also should be taken into account that there are restrictions to the AUTOSAR meta model specified for the RTE [7], i.e. the meta model allows defining more name spaces than the actual implementations of the RTE would support. E.g. [SWS_Rte_07190] requires unique names for SwComponentTypes regardless of the ARPackage hierarchy. However, there is the possibility to add a symbolic name in a later stage of development via SymbolProps (see chapter 6.4).

AUTOSAR allows defining so called global elements [9]. Global elements can be referenced by using this reference base. It means that the ShortNames of these elements are unique within the current package (if the base is the current package) or assumed to be unique in the referenced ARPackage. However, their usage is restricted ([constr_2538]) and shall not be misused for defining virtual name spaces.

Thus, AUTOSAR does not provide the possibility to explicitly specify virtual name spaces.

As we will later see however we will need abbreviations for paths to packages. This can be realized via AliasNameAssignments.

Example:

```
<ALIAS-NAME-SET>
  <SHORT-NAME>NameSpaceAbbr</SHORT-NAME>
  <ALIAS-NAMES>
    <ALIAS-NAME-ASSIGNMENT>
      <SHORT-LABEL>AR_Types</SHORT-LABEL>
      <IDENTIFIABLE-REF DEST="AR-PACKAGE">/AUTOSAR/SwComponentTypes</
          IDENTIFIABLE-REF>
    </ALIAS-NAME-ASSIGNMENT>
  </ALIAS-NAMES>
</ALIAS-NAME-SET>
```

6.7 References in Documentation

There are several possibilities for references within textual documentation. In this context the two most important ones are:

- Usage of Xref
- Usage of Tt

Both are used within paragraphs (<P>).

Example:

```
<P>
  <L-1 L="EN">This is a direct reference to a system constant
    <XREF SHOW-RESOURCE-SHORT-NAME="SHOW-SHORT-NAME">
      <REFERRABLE-REF DEST="SW-SYSTEMCONST">/OEM1/SwSystemconsts/DIESEL</
          REFERRABLE-REF>
    </XREF>.
  </L-1>
</P>
```

This would e.g. be printed like this:

"This is a direct reference to a system constant ’DIESEL’."

Since instance references are not supported within paragraphs one possibility would be to first define a FlatInstanceDescriptor and to reference this element.

Example:

```
<P>
  <L-1 L="EN">This is a reference to a sw signal
    <XREF SHOW-RESOURCE-SHORT-NAME="SHOW-SHORT-NAME">
      <REFERRABLE-REF DEST="FLAT-INSTANCE-DESCRIPTOR">/AUTOSAR/
        AISpecification/Flatmaps/PowertrainFlatmap/Eng_n</REFERRABLE-REF>
    </XREF>.
  </L-1>
</P>
```

This would e.g. be printed like this:

"This is a reference to a sw signal 'Eng_n'."

If there is no flat map then also DocumentContext information can be used for unique names. A DocumentContext is very similar to a FlatInstanceDescriptor.

Example:

```
<DOCUMENTATION-CONTEXT>
  <SHORT-NAME>AR_Eng_n</SHORT-NAME>
  <FEATURE-IREF>
    <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/OEM1/Systems
      /System/TopLvl</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/
      AUTOSAR_AISpecification/SwComponentTypes/TopLvl/Pt</CONTEXT-ELEMENT-
      REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/
      AUTOSAR_AISpecification/SwComponentTypes/Pt/Eng</CONTEXT-ELEMENT-REF
      >
    <CONTEXT-ELEMENT-REF DEST="PORT-PROTOTYPE">/AUTOSAR_AISpecification/
      SwComponentTypes/Eng/EngN</CONTEXT-ELEMENT-REF>
  </FEATURE-IREF>
</DOCUMENTATION-CONTEXT>

<P>
  <L-1 L="EN">This is a reference to a sw signal
    <XREF SHOW-RESOURCE-SHORT-NAME="SHOW-SHORT-NAME">
      <REFERRABLE-REF DEST="DOCUMENTATION-CONTEXT">/OEM1/Documentations/
        Eng_Docu/AR_Eng_n</REFERRABLE-REF>
    </XREF>.
  </L-1>
</P>
```

This would e.g. be printed like this:

"This is a reference to a sw signal 'AR_Eng_n'."

In some cases also the usage of technical terms, i.e. Tt, is interesting. This even can be an alternative to XREF in general if some prerequisites are fulfilled.

Example:

```
<P>
  <L-1 L="EN">This is a reference to system constant
    <TT TYPE="SwSystemconsts">DIESEL</TT>
```

```
    via a technical term.
  </L-1>
</P>
```

This would e.g. be printed like this:

"This is a reference to system constant DIESEL via a technical term."

In the technical terms overview chapter there would be a heading "SwSystemconsts" and one entry "DIESEL" with reference to the pages this technical term is used:

"Technical Terms - SwSystemconsts DIESEL 23, 50, 90"

Concrete example from Powertrain Explanation Document [4]: Given the corresponding xml-specifications the table could have been automatically derived:

DisplayName w/o Name Space. With Name Space add AR_ before name, e.g. AR_Abs_flgActv	*Shortname of Port / additional information if needed*	*Longname of Displayname (= PortPrototype-Blueprint name extended in case of multiple data prototypes or arrays)*
Pt_nClu	PtNClu	Powertrain: Actual Speed at Clutch
Pt_nEngSp	PtNEngSp	Powertrain: Engine Speed Setpoint
Pt_ratTqDistbnReqFrntLe	PtRatTqDistbnReq	Powertrain: Requested Percental Distribution of Torque to Wheels - Front Left Wheel
Pt_ratTqDistbnReqFrntRi	PtRatTqDistbnReq	Powertrain: Requested Percental Distribution of Torque to Wheels - Front Right Wheel

Figure 6.5

In arxml:

```
<DOCUMENTATION-CONTENT>
  <TABLE>
    <TGROUP COLS="3">
      <COLSPEC COLNUM="1" COLNAME="col1" COLWIDTH="1.00*" />
      <COLSPEC COLNUM="2" COLNAME="col2" COLWIDTH="1.00*" />
      <COLSPEC COLNUM="3" COLNAME="col3" COLWIDTH="1.00*" />
      <THEAD>
        <ROW>
          <ENTRY COLNAME="col1">
            <P><L-1 L="EN">
              <TT TYPE="ARGlossary">DisplayName</TT> w/o Name Space.
                With Name Space add AR_ before name, e.g. AR_Abs_flgActv
            </L-1></P>
          </ENTRY>
          <ENTRY COLNAME="col2">
            <P><L-1 L="EN">
```

```
          <TT TYPE="ARMetaModelElement">ShortName</TT> of Port
            / additional information if needed
        </L-1></P>
      </ENTRY>
      <ENTRY COLNAME="col3">
        <P><L-1 L="EN">
          <TT TYPE="ARMetaModelElement">LongName</TT> of
          <TT TYPE="ARGlossary">DisplayName</TT> (= name
          extended in case of multiple data prototypes or arrays)
        </L-1></P>
      </ENTRY>
    </ROW>
  </THEAD>
  <TBODY>
    <ROW>
      <ENTRY COLNAME="col1">
        <P><L-1 L="EN">
          <XREF SHOW-RESOURCE-SHORT-NAME="SHOW-SHORT-NAME">
            <REFERRABLE-REF DEST="FLAT-INSTANCE-DESCRIPTOR">/AUTOSAR/
              AISpecification/Flatmaps/PowertrainFlatmap/Pt_nClu</
              REFERRABLE-REF>
          </XREF>
        </L-1></P>
      </ENTRY>
      <ENTRY COLNAME="col2">
        <P><L-1 L="EN">
          <XREF SHOW-RESOURCE-SHORT-NAME="SHOW-SHORT-NAME">
            <REFERRABLE-REF DEST="PORT-PROTOTYPE-BLUERPRINT">/AUTOSAR/
              AISpecification/PortPrototypeBlueprints/PtNClu</
              REFERRABLE-REF>
          </XREF>
        </L-1></P>
      </ENTRY>
      <ENTRY COLNAME="col3">
        <P><L-1 L="EN">
          <XREF SHOW-RESOURCE-LONG-NAME="SHOW-LONG-NAME">
            <REFERRABLE-REF DEST="FLAT-INSTANCE-DESCRIPTOR">/AUTOSAR/
              AISpecification/Flatmaps/PowertrainFlatmap/Pt_nClu</
              REFERRABLE-REF>
          </XREF>
        </L-1></P>
      </ENTRY>
    </ROW>
    <ROW>
      <ENTRY COLNAME="col1">
        <P><L-1 L="EN">
          <XREF SHOW-RESOURCE-SHORT-NAME="SHOW-SHORT-NAME">
            <REFERRABLE-REF DEST="FLAT-INSTANCE-DESCRIPTOR">/AUTOSAR/
              AISpecification/Flatmaps/PowertrainFlatmap/Pt_nEngSp</
              REFERRABLE-REF>
          </XREF>
        </L-1></P>
      </ENTRY>
      <ENTRY COLNAME="col2">
        <P><L-1 L="EN">
          <XREF SHOW-RESOURCE-SHORT-NAME="SHOW-SHORT-NAME">
```

```
          <REFERRABLE-REF DEST="PORT-PROTOTYPE-BLUERPRINT">/AUTOSAR/
             AISpecification/PortPrototypeBlueprints/PtNEngSp</
             REFERRABLE-REF>
        </XREF>
      </L-1></P>
    </ENTRY>
    <ENTRY COLNAME="col3">
      <P><L-1 L="EN">
        <XREF SHOW-RESOURCE-LONG-NAME="SHOW-LONG-NAME">
          <REFERRABLE-REF DEST="FLAT-INSTANCE-DESCRIPTOR">/AUTOSAR/
             AISpecification/Flatmaps/PowertrainFlatmap/Pt_nEngSp</
             REFERRABLE-REF>
        </XREF>
      </L-1></P>
    </ENTRY>
  </ROW>
  <ROW>
    <ENTRY COLNAME="col1">
      <P><L-1 L="EN">
        <XREF SHOW-RESOURCE-SHORT-NAME="SHOW-SHORT-NAME">
          <REFERRABLE-REF DEST="FLAT-INSTANCE-DESCRIPTOR">/AUTOSAR/
             AISpecification/Flatmaps/PowertrainFlatmap/
             Pt_ratTqDistbnReqFrntLe</REFERRABLE-REF>
        </XREF>
      </L-1></P>
    </ENTRY>
    <ENTRY COLNAME="col2">
      <P><L-1 L="EN">
        <XREF SHOW-RESOURCE-SHORT-NAME="SHOW-SHORT-NAME">
          <REFERRABLE-REF DEST="PORT-PROTOTYPE-BLUERPRINT">/AUTOSAR/
             AISpecification/PortPrototypeBlueprints/
             PtRatTqDistbnReqFrnt</REFERRABLE-REF>
        </XREF>
      </L-1></P>
    </ENTRY>
    <ENTRY COLNAME="col3">
      <P><L-1 L="EN">
        <XREF SHOW-RESOURCE-LONG-NAME="SHOW-LONG-NAME">
          <REFERRABLE-REF DEST="FLAT-INSTANCE-DESCRIPTOR">/AUTOSAR/
             AISpecification/Flatmaps/PowertrainFlatmap/
             Pt_ratTqDistbnReqFrntLe</REFERRABLE-REF>
        </XREF>
      </L-1></P>
    </ENTRY>
  </ROW>
  <ROW>
    <ENTRY COLNAME="col1">
      <P><L-1 L="EN">
        <XREF SHOW-RESOURCE-SHORT-NAME="SHOW-SHORT-NAME">
          <REFERRABLE-REF DEST="FLAT-INSTANCE-DESCRIPTOR">/AUTOSAR/
             AISpecification/Flatmaps/PowertrainFlatmap/
             Pt_ratTqDistbnReqFrntRi</REFERRABLE-REF>
        </XREF>
      </L-1></P>
    </ENTRY>
    <ENTRY COLNAME="col2">
```

```
            <P><L-1 L="EN">
              <XREF SHOW-RESOURCE-SHORT-NAME="SHOW-SHORT-NAME">
                <REFERRABLE-REF DEST="PORT-PROTOTYPE-BLUERPRINT">/AUTOSAR/
                   AISpecification/PortPrototypeBlueprints/
                   PtRatTqDistbnReqFrnt</REFERRABLE-REF>
              </XREF>
            </L-1></P>
          </ENTRY>
          <ENTRY COLNAME="col3">
            <P><L-1 L="EN">
              <XREF SHOW-RESOURCE-LONG-NAME="SHOW-LONG-NAME">
                <REFERRABLE-REF DEST="FLAT-INSTANCE-DESCRIPTOR">/AUTOSAR/
                   AISpecification/Flatmaps/PowertrainFlatmap/
                   Pt_ratTqDistbnReqFrntRi</REFERRABLE-REF>
              </XREF>
            </L-1></P>
          </ENTRY>
        </ROW>
      </TBODY>
    </TGROUP>
  </TABLE>
</DOCUMENTATION-CONTENT>
```

6.8 Instance cp- and pb-paths

In this chapter new terms are introduced that allow a shortened description of the flat instance descriptors. In the context of calibration and measurement the data prototypes are of relevance.

Definition. The instance `cp-path` (SwComponentPrototype-Path) is the instance path to the software component prototype + one of its ports prototypes + one of the data prototypes (recursively) of the port interface of the port prototype:

{instance path to sw component prototype}/{port prototype}/{data prototype})$_{1..n}$

e.g.

```
/OEM1/Systems/System/TopLvl/Pt/Eng/EngN/EngN
```

i.e. in Backus Naur Form:

```
/({ARPackage of System}/)1..n
{System.name}/
{RootSwCompositionPrototype.name}/
({SwComponentPrototype - part of referenced SwComponentType
of RootSwCompositionPrototype or parent SwComponentPrototype
resp.}/)0..n
{PortPrototype of referenced SwComponentType}
(/{AutosarDataPrototype of referenced PortInterface or within
parent data prototype})1..n
```

Parent sw component prototype means the SwComponentPrototype referring to the SwComponentType the SwComponentPrototype is part of. E.g. in the cp-path

/OEM1/Systems/System/TopLvl/Pt/Eng/EngN/EngN

"Pt" is the parent sw component prototype of "Eng" (see example later in document).

Definition. The instance pb-path (PortPrototypeBlueprint- Path) is the instance path to the port prototype blueprint + one of the data prototypes of the port interface of the port prototype blueprint:

{instance path to port prototype blueprint)/({data prototype})$_{1..n}$

e.g.

/AUTOSAR/PortPrototypeBlueprints_Blueprint/EngN/EngN

i.e. in Backus Naur Form:

/({ARPackage of PortPrototypeBlueprint}/)$_{1..n}$
{PortPrototypeBlueprint}
(/{AutosarDataPrototype of referenced PortInterface or within parent data prototype})$_{1..n}$

7 Overall Modeling and Generation Rules

[TR_MCM_70005] ⌈The model and the generated model elements shall be compliant to the Meta Model.

Fulfills requirements: RS_MCR_90783⌋

The meta model parts relevant for this document are documented in [8], [6], [5], and [9].

[TR_MCM_70010] ⌈The model shall be compliant to the Shall-Rules defined in [3, AUTOSAR_CP_TR_SWCModelingGuide]. Recommendations and optional rules, however, may be violated.

Fulfills requirements: RS_MCR_90784⌋

[TR_MCG_00001] ⌈The generated display names shall be conformant to the C programming language.

Fulfills requirements: RS_MCR_90014⌋

[TR_MCG_00004] ⌈If a manually defined display name is available then it shall not be overwritten by a generated one.

Fulfills requirements: RS_MCR_90005, RS_MCR_90008⌋

For explanation of "shall" and "recommendation/should" please see chapter 2.1.

There are two separate display name schemas, one for prototypes and one for AREle-ments.

[TR_MCG_00785] ⌈For ARElements we have the following name parts:

```
element : {nameSpace}_{data}
```

with

```
nameSpace : {ARPackage.name}(_{ARPackage.name})0..n
data : {ARElement.name}(_{typeId})
```

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90004, RS_MCR_90015, RS_MCR_90016⌋

The {nameSpace} corresponds to the package hierarchy since *ARPackages* define name spaces. The {typeId} corresponds to additional information, e.g. for calibration parameters.

Elements discussed in the following are *PortPrototypeBlueprints*, *SwSystemconsts*, *Units* and *Systems*.

[TR_MCG_00790] ⌈For ARElements that have a symbolic name (SymbolProps) we have the following name parts:

`element : {ARElement.symbol}(_{typeId})`

Fulfills requirements: RS_MCR_90000, RS_MCR_90002, RS_MCR_90003, RS_MCR_90006, RS_MCR_90009, RS_MCR_90014, RS_MCR_90015, RS_MCR_90016,⌋

Now the display name schema for prototypes is presented:

[TR_MCG_00786] ⌈For prototypes we have the following name parts:

`prototype : {nameSpace}_{componentHierarchy}_{data}`

with

`nameSpace : ({ARPackage.name}_)`$_{1..n}${`System.name}`
`componentHierarchy : ({SwComponentPrototype.name}_)`$_{0..n}$

or

`componentHierarchy : {RootSwCompositionPrototype.name}`

The {data} differs depending on the kind of the prototype.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90004, RS_MCR_90015, RS_MCR_90016,⌋

The {nameSpace} corresponds to path to the relevant system. The {componentHierarchy} considers the *SwComponentPrototypes* hierarchy including multiple instantiation. The *RootSwCompositionPrototype* can typically be ignored in the {componentHierarchy} because there is exactly one within a system. Only if the port directly belongs to the root component then its name is relevant.

The {data} refers to the port and one of the data prototypes of its port interface plus an optional suffix. The details on {data} see next chapters.

[TR_MCG_00791] ⌈For prototypes that have a symbolic name (SymbolProps) we have the following name parts:

`prototype : {Prototype.symbol}`

Fulfills requirements: RS_MCR_90000, RS_MCR_90002, RS_MCR_90003, RS_MCR_90006, RS_MCR_90009, RS_MCR_90014, RS_MCR_90015, RS_MCR_90016⌋

Prototypes discussed in the following are *SwComponentPrototypes* and *AutosarDataPrototypes* of a *PortInterface* of a complex data type like e.g. an array. Note: *PortPrototypeBlueprints* are elements, no prototypes. *PortPrototype* names themselves are only relevant as part of display names but not for themselves in the context of measurement and calibration.

In the following we will have a closer look at every name part and will discuss how to shorten it or discuss in which cases the name part can be omitted completely.

7.1 Name Part Length

A display name of a sw signal should not exceed 31 characters. Therefore we recommend fulfilling the following rules. We are aware that this will not always be possible.

[TR_MCM_70022] ⌈The {nameSpace} of a sw signal should not exceed 3 characters.

Fulfills requirements: RS_MCR_90006⌋

[TR_MCM_70020] ⌈It is recommended to further restrict the length of model element names relevant for generating the display names. *AutosarDataPrototype* names and *PortPrototype* names should not exceed 23 characters.

Fulfills requirements: RS_MCR_90006⌋

[TR_MCM_70390] ⌈The suffixes added to distinguish different type identifiers ({typeId}) of *AutosarDataPrototypes* should not exceed 3 characters.

Fulfills requirements: RS_MCR_90006⌋

8 Data Prototypes

In this chapter we deal with the {dataInfo} name part that will be needed for {data}.

[TR_MCG_00021] ⌈{dataInfo} refers to one *AutosarDataPrototype*.

In case of primitive types we have:

`dataInfo : {AutosarDataPrototype.name}`

In case of complex types we have:

`dataInfo : {AutosarDataPrototype.name} (_{parentData})`$_{0..n}$`_{AutosarDataPrototype.name}`

with

`parentData : ({AutosarDataPrototype.name}|{index})`

{index} is needed in case of *ArrayElements*.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

Example for Records:

Assume we have the port prototype LockgCenSts referencing *PortInterface* LockgCenSts with data element LockgCenSts with the following *ApplicationRecordDataType*:

ShortName: LockgCenSts1
recordElement 1: LockgSt
recordElement 2: TrigSrc

Then we would have

dataInfo : LockgCenSts_LockgSt

or

dataInfo : LockgCenSts_TrigSrc

depending which of the two sw signals you are interested in.

Since *ArrayElements* do not have names only the index number can be used for automatic generation. So in general it is better to define display names manually in this case.

Arrays are handled in section 6.2; they will need the {index} name part. Note that the {parentData} above may be an *ArrayElement*.

Example for Arrays:

Display name of the single elements of the 4-element array

EscVWhlInd

could be generated by using [TR_MCG_00021]

Esc_vWhlInd_0, Esc_vWhlInd_1, Esc_vWhlInd_2 and Esc_vWhlInd_3

or - manually extending the generated display name of the array itself -

Esc_vWhlIndFrntLe, Esc_vWhlIndFrntRi,
Esc_vWhlIndReLe and Esc_vWhlIndReRi

for front left wheel etc.

9 Data Prototypes in DataInterfaces

In this chapter we deal with the {data} name part of data interfaces. The general schema is (see [TR_MCG_00786]):

`data : {port}_{dataInfo}_{typeId}`

[TR_MCG_00005] ⌈For {data} of data prototypes within data interfaces we have the following name parts:

`data : {port}_{dataInfo}(_{typeId})`$_{0..1}$

{port} may refer to *PortPrototype* or *PortPrototypeBlueprint*, i.e.

`port : (PortPrototype.name)`

or

`port : {PortPrototypeBlueprint.name}`

{dataInfo} refers to one data prototype in the role of a *dataElement*, a parameter or an *nvData* within the data interface of the port.

{typeId} depends on the type of the final data prototype.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90015⌋

Example:

Assume we have the port prototype LockgCenSts referencing *PortInterface* LockgCenSts with data element LockgCenSts with the following *ApplicationRecordDataType*:

ShortName: LockgCenSts1
recordElement 1: LockgSt
recordElement 2: TrigSrc

Then we would have

data : LockgCenSts_LockgCenSts_LockgSt

or

data : LockgCenSts_LockgCenSts_TrigSrc

depending which of the two sw signals you are interested in.

The {data} name part is often quite long, contains a lot of redundancy and does not always fulfill all the requirements. Depending on specific additional rules and assumptions this name can be shortened.

These rules and assumptions are documented in the following.

Only one of the following four rules [TR_MCG_00010], [TR_MCG_00015], [TR_MCG_00020] and [TR_MCG_00030] can be applied.

[TR_MCG_00010] ⌈If the data element name is identical to the port name then the port prototype or port prototype blueprint name can be ignored within the display name of the element. We have

`data : {dataInfo}(_{typeId})`$_{0..1}$

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90006, RS_MCR_90011⌋

[TR_MCG_00015] ⌈If the data element name is identical to "Val" (for "Value") then the data prototype name can be ignored within the display name.

In case of primitive types we have:

`data : {port}(_{typeId})`$_{0..1}$

In case of complex types we have:

`data : {port}_{dataInfo}(_{typeId})`$_{0..1}$

with

`dataInfo : ({parentData}_)`$_{0..n}${AutosarDataPrototype.name}

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90006, RS_MCR_90011⌋

[TR_MCG_00020] ⌈If the *PortInterface* contains exactly one data element then the data element name can be ignored within the display name.

In case of primitive types we have:

`data : {port}(_{typeId})`$_{0..1}$

In case of complex types we have:

`data : {port}_{dataInfo}(_{typeId})`$_{0..1}$

with

`dataInfo : ({parentData}_)`$_{0..n}${AutosarDataPrototype.name}

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90006, RS_MCR_90011⌋

[TR_MCG_00030] ⌈If

1. the PortInterface contains more than one data element and
2. the PortPrototype or PortPrototypeBlueprint name is a substring of the data element name

then the port name can be ignored in the display name. We have

`data : {dataInfo}(_{typeId})`$_{0..1}$

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90006, RS_MCR_90011⌋

Example 1:

Take the following examples for {port}_{data} from Figure 6.2 (with primitive data type):

EngN_EngN
EngNMax_EngN

and the following example:

EngN_Val

And finally the following example assuming the port interface has two data prototypes EngNMax and EngNMin:

EngN_EngNMax

Rules [TR_MCG_00010], [TR_MCG_00015], [TR_MCG_00020] and [TR_MCG_00030] then would lead to

EngN or EngN
EngNMax
EngN
EngNMax

Example 2:

See above, our record example LockgCenSts would then lead to ([TR_MCG_00010]):

```
data :  LockgCenSts_LockgSt
```

or

```
data :  LockgCenSts_TrigSrc
```

depending which of the two sw signals you are interested in.

Similar rules can be added for data prototypes within complex data types.

9.1 Increase Readability of Display Names

To further increase the readability of display names the following modeling rules are recommended.

Our examples from the previous chapter were very short, e.g.:

EngN

However, the generated name could also look like this:

TrsmCluStTar

and may even be much longer.

In this case it would improve readability to point out the mean/source as well as the physical or logical meaning of the AutosarDataPrototype and add an underscore before it:

TrsmClu_StTar

To even improve readability more the physical or logical types should be written in small letters:

TrsmClu_stTar

To be able to do so additional generating and modeling rules are necessary:

[TR_MCM_70050] ⌈There shall be a small and manageable list of physical and logical types for generating display names. This list is called the P/L-list.

Fulfills requirements: RS_MCR_90782⌋

In chapter A a proposal for such a P/L-List for the powertrain domain is presented.

[TR_MCM_70060] ⌈The P/L-List is a subset of the keyword abbreviations from [1, AUTOSAR_CP_MOD_AISpecification].

Fulfills requirements: RS_MCR_90011, RS_MCR_90784⌋

However, not all physical types or actions of [1, AUTOSAR_CP_MOD_AISpecification] are elements of the P/L-list; it would be too large. Also other keywords not belonging to the semantic field "Physical Type/Action" might be part of the P/L-list. For example no logical types are contained in the field "Physical type/Action" in the modeling guideline but these are very useful for distinguishing control flow and functional software signals.

[TR_MCM_70065] ⌈Assuming the P/L-list is not empty every *AutosarDataPrototype* name should contain at least one keyword from the P/L-list.

Fulfills requirements: RS_MCR_90782⌋

The P/L-list can be empty. Then there is no effect on display name generation and modeling.

[TR_MCG_00070] ⌈For generating the display name only the first occurrence of a keyword out of the P/L-list within the port or data element name is relevant: an underscore is added before and the keyword is written in small letters. If there are both, port prototype name and data prototype name then the underscore between them is deleted.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90011⌋

Example:

Assume we have {port}_{*AutosarDataPrototype*.name} : EngN_Max. Then rule [TR_MCG_00070] leads to

Eng_nMax.

[TR_MCG_00080] ⌈If a name starts with a keyword out of the P/L-list then no "_" is added before the keyword from the P/L-list.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

Reason: Within the *FlatMap Identifiables* are used for the display name. For *Identifiables* it is not allowed to start the name with "_" or to contain "_" ([9]).

[TR_MCM_70070] ⌈A *PortPrototype* or *PortPrototypeBlueprint* name should not start with a keyword out of the P/L-list. I.e. it is recommended to first add a keyword of *Classification* "Mean-Environment-Device" or other semantic information to it.

Fulfills requirements: RS_MCR_90781⌋

The rules above lead to the following schema in syntax of name pattern language of [5]:

```
display Name :  ({part1)_}0..1({physLog})0..1{part2}(_{typeId})0..1
```

with

```
part1 :  (({keyword})1..n ({index})0..n)1..n
physLog :  {keyword from P/L-List, written in small letters}
part2 :  ({keyword} | {index} | '_')0..n
typeId :  { 'MP' | 'SC' | 'C' | 'T' | 'M' | 'CA' | 'Ax' }
```

The kinds of suffix allowed are described in the following.

9.2 VariableDataPrototypes in SenderReceiverInterfaces

To distinguish between variables that are needed for inter-runnable exchange and variables that are only defined for measurement purposes, the following rule is defined:

[TR_MCG_00090] ⌈Add a suffix "_MP" for the display name of VariableDataPrototypes that are measurement points. We have

```
data :  {port}_{dataInfo }_MP
```

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90009, RS_MCR_90011⌋

Whether a data prototype is a measurement point or not is specified within *SwImplPolicy* of *SwDataDefProps*.

Example:

Assume the following software signal to be defined only for measurement purposes:

TrsmClu_stTar

With rule [TR_MCG_00090] we then have

TrsmClu_stTar_MP

9.3 ParameterDataPrototypes in ParameterInterface

The type of a parameter is characterized by the category of its data type, e.g. MAP, CURVE [6]. An *AutosarDataPrototype* is in the role of a parameter if it is an *ApplicationCompositeElementDataPrototype* being part of a *ParameterDataPrototype* or it is a *ParameterDataPrototype* itself.

It increases readability if it can be seen at one glance whether it is a variable or a parameter. Thus the following rules are added to distinguish parameters from variables:

[TR_MCG_00310] ⌈Add a suffix "_C" for "constant" for the display name of a *AutosarDataPrototype* in the role of a parameter being a single data point (Category of *ApplicationDataType* equal to VALUE, STRING or BOOLEAN). We have

`data : {port}_{dataInfo}_C`

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90009, RS_MCR_90011⌋

[TR_MCG_00320] ⌈Add a suffix "_T" for "table" or "curve" for the display name of a *AutosarDataPrototype* in the role of a parameter being a set of data points with one axis (dependency) (Category of *ApplicationDataType* equal to CURVE). We have

`data : {port}_{dataInfo}_T`

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90009, RS_MCR_90011⌋

[TR_MCG_00330] ⌈Add a suffix "_M" for "map", i.e. a *AutosarDataPrototype* in the role of a parameter with a two dimensional set of data points with two axes (dependencies) (Category of *ApplicationDataType* equal to MAP). We have

`data : {port}_{dataInfo}_M`

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90009, RS_MCR_90011⌋

[TR_MCG_00340] ⌈Add a suffix "_CA" for "array of calibration parameters" for the display name of a *AutosarDataPrototype* in the role of a parameter of type ApplicationArrayDataType with type of *ApplicationArrayElements* of Category CURVE or VAL_BLK. We have

`data : {port}_{dataInfo}_CA`

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90009, RS_MCR_90011⌋

[TR_MCG_00350] ⌈Add a suffix "_Ax" for "axis" for the display name of a *AutosarDataPrototype* in the role of a parameter with axis elements (Category COM_AXIS and RES_AXIS). We have

```
data :  {port}_{dataInfo}_Ax
```

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90009, RS_MCR_90011⌋

For the other categories of *ParameterDataPrototypes* additional suffixes should be added.

Example:

CtryNrCod

would lead to (assuming CtryNrCod is a parameter of Category VALUE, "Nr" is part of the P/L-list and applying rule [TR_MCG_00310]):

Ctry_nrCod_C

10 Data Prototypes in ClientServerInterfaces

In this chapter we deal with the {data} name part of client server interfaces. The general schema is:

```
data :  {port}_{operationInfo}_{typeId}
```

[TR_MCG_00360] ⌈For {data} of data prototypes within client server interfaces we have the following name parts:

```
data :  {port}_{operationInfo}(_{typeId})0..1
```

{port} may refer to *PortPrototype* or *PortPrototypeBlueprint*, i.e.

```
port :  (PortPrototype.name)
```

or

```
port :  {PortPrototypeBlueprint.name}
```

For {operationInfo} we have

```
operationInfo:  {ClientServerOperation.name}_{dataInfo}
```

with {dataInfo} being the data prototype information of one of the arguments of the operation (*ArgumentDataPrototype*).

{typeId} depends on the type of the final data prototype.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

10.1 Data Prototypes within InternalBehavior

In this chapter we deal with the {data} name part of *AutosarDataPrototypes* within *InternalBehavior*.

For *InterhalBehavior* we have:

- *constantMemory*
- *staticMemory*

For *SwcInternalBehavior* we have:

- *arTypedPerInstanceMemory*
- *explicitInterRunnableVariable*
- *implicitInterRunnableVariable*
- *perInstanceParameter*
- *sharedParameter*

For *BswInternalBehavior* we have:

- *perInstanceParameter*

The general schema is:

`data : {swcInternalBehavior}_{dataInfo}_{typeId}`

It is assumed that the internal behavior is not reused within different *SwComponentTypes* or *BswModuleDescriptions*, i.e. it is assumed that there is at most one *SwComponentType* per *SwcInternalBehavior* and one *BswModuleDescription* per *BswBehavior*.

SwcInternalBehavior defines the name space for internal data prototypes similar to the port prototype for the requested or provided data prototypes. Therefore it also has to be considered. Only atomic components do have an internal behavior.

[TR_MCG_00033] ⌈For {data} of data prototypes within *InternalBehavior* we have the following name parts:

$\texttt{data : \{swcInternalBehavior\}_\{dataInfo\}(_\{typeId\})}_{0..1}$

with

`swcInternalBehavior : {{SwcInternalBehavior.name} | Int}`

"Int" is the abbreviation for "Internal". No *PortPrototype* of the component shall have a name identical to "Int". Otherwise the name of the *SwcInternalBehavior* shall be used to make name unique.

{dataInfo} refers to one *AutosarDataPrototype* in the role of an *arTypedPerInstanceMemory*, an *explicitInterRunnableVariable*, an *implicitInterRunnableVariable*, a *perInstanceParameter*, a *sharedParameter*, a *constantMemory* or a *staticMemory*.

{typeId} depends on the type of the final data prototype.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

[TR_MCA_80034] ⌈No *PortPrototype* of a component shall have a name identical to "Int". "Int" is reserved as abbreviation of the internal behavior.

Fulfills requirements: RS_MCR_90015⌋

11 Name Space

In this chapter we deal with the name space identifier {nameSpace}. The general schema for the name space for *ARElements* ({element}) is (see [TR_MCG_00785])

`nameSpace : ({ARPackage.name}/)`$_{1..n}$

The general schema of the name space for prototypes ({prototype}) is (see [TR_MCG_00786]):

`nameSpace : ({ARPackage.name}/)`$_{1..n}$`_{System.name}`

Details on the name space concept of AUTOSAR can be found in [9]. Virtual name spaces are defined in chapter 6.6.

Every element is part of an *ARPackage*.

Some prototype elements like *SwComponentPrototypes* are element of a different name space, for *SwComponentPrototypes* it is the *SwComponentType*. For those the {componentHierarchy} is more important than the *ARPackage* the component type is part of. But for *ARElements* like *PortPrototypeBlueprints*, *Units*, *System* etc. the {nameSpace} is very important.

In the following some rules are given how to deal with the {nameSpace}.

[TR_MCM_70030] ⌈An arbitrary abbreviation for the {nameSpace} may be defined for the elements within an *ARPackage*.

Fulfills requirements: RS_MCR_90006⌋

[TR_MCM_70040] ⌈For elements within the predefined *ARPackage* "AUTOSAR" the abbreviation "AR" should be used as part of the {nameSpace}.

Fulfills requirements: RS_MCR_90001, RS_MCR_90006⌋

[TR_MCG_00035] ⌈Package paths within generated display names should be shortened by using defined abbreviations.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

As explained in chapter 6.6 *AliasNameAssignments* can be used to formally specify such abbreviations for paths.

Example:

Assume we define the following abbreviations for the paths

```
AUTOSAR/SwComponentTypes -> AR_Types
```

Then rule [TR_MCG_00035] would lead to

`nameSpace : AR_Types`

instead of AUTOSAR_SwComponentTypes

Please note too that the short label was chosen as "AR_Types" and not "Types" or "hugo": This is because [TR_MCM_70040] to have a name space for AUTOSAR display names still should be fulfilled.

[TR_MCG_00120] ⌈If an *ARPackage* belongs to a virtual name space for the element class then the package name can be ignored in the display name of this element. Only the name of the *ARPackage* defining the virtual name space and the {nameSpace} of this *ARPackage* itself are to be considered.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90006, RS_MCR_90011⌋

Example:

In

`/AUTOSAR/AISpecification/PortPrototypesBlueprints_Blueprint/`

/AUTOSAR/ currently defines a virtual name space for all port prototypes blueprints. Therefore with [TR_MCG_00120] the display name of a blueprint can be shortened to

`element : {portBlueprint} : AUTOSAR_{data}`

With [TR_MCG_00035] and [TR_MCM_70040] the name can further be shortened to

`AR_{data}`

If we could not ensure uniqueness for /AUTOSAR/ then we could only say that /AUTOSAR/AISpecification defines a virtual name space for all port prototypes blueprints. Therefore with [TR_MCG_00120] the display name of a blueprint can only be shortened to

`portBlueprint : AUTOSAR_AISpecification_{data}`

AUTOSAR is the {nameSpace} of *ARPackage AISpecification*.

[TR_MCA_80710] ⌈Assumption: There is only one *System* relevant for display name generation.

Fulfills requirements: RS_MCR_90006⌋

This assumption is typically fulfilled because there is no need to have display names consistent over different systems: a *FlatMap* always covers only one system or one ECU but in different variations.

[TR_MCG_00713] ⌈If there is only one system the {nameSpace}, i.e. the names of the *ARPackages* as well as the system name, can be ignored in the display name of a prototype element. We have

`prototype : {componentHierarchy}_{data}`

Reason: the starting point is the *RootSwCompositionPrototype* for the System.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90006, RS_MCR_90011⌋

In rule [TR_MCG_00713] assumption [TR_MCA_80710] is assumed to be fulfilled: there is only one system.

If there is a *FlatMap* defining a unique name for a *System* also this name can be used for {nameSpace}.

[TR_MCG_00770] ⌈The display names specified in a *FlatMap* for *System* can be used for specifying the {nameSpace} name part of the display name of a sw signal. General rule:

```
nameSpace :  {systemDescriptor}
```

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

12 SwSystemconsts

Even constants values already evaluated at compile time can be visible ("*readOnly*") within a MCD system. For system constants (element : {systemconst}) the general schema is defined as follows (see [TR_MCG_00785]):

```
systemconst :  {nameSpace}_{SwSystemconst.name}_{typeId}
```

[TR_MCG_00510] ⌈Add a suffix "_SC" for "software configuration" for the display name of a *SwSystemconst*. We have

```
systemconst :  {nameSpace}_{data}
```

with

```
data :  {SwSystemconst.name}_SC
```

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90009⌋

[TR_MCG_00520] ⌈For a display name of a *SwSystemconst* use only capital letters.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90009⌋

Please note: [TR_MCG_00070] does not hold for system constants. This is because system constants typically do not contain physical or logical information but represent a feature or functionality.

[TR_MCA_80530] ⌈Assumption: There is exactly one virtual name space sw for system constants.

Fulfills requirements: RS_MCR_90006⌋

[TR_MCG_00512] ⌈If there is only one system and exactly one virtual name space for *SwSystemconsts* we have

```
systemconst :  {SwSystemconst.name}_SC
```

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

In rule [TR_MCG_00512] assumption [TR_MCA_80530] and [TR_MCA_80710] are assumed to be fulfilled.

Example:

```
/OEM1/SwSystemconsts/EngNrCyl
```

would lead to (applying [TR_MCG_00510], [TR_MCG_00520]; assuming uniqueness of names ([TR_MCA_80530]); even assuming "Nr" is part of the P/L-list):

```
ENGNRCYL_SC
```

13 PortPrototypeBlueprints

In this chapter we deal with unique names for port prototype blueprints. Since *PortPrototypeBlueprints* are *ARElements* ({element} : {portBlueprint}) the general schema is (see [TR_MCG_00785]):

`portBlueprint : {nameSpace}_{data}`

with

`nameSpace : ({ARPackage.name}_)`$_{1..n}$
`data : {port}_{dataInfo}`

{typeId} is not relevant for port prototype blueprints because they cannot be measured or calibrated themselves but may serve only as part of display names of sw signals.

[TR_MCG_00788] ⌈For port prototype blueprints the following basic display name schema

`portBlueprint : {nameSpace}_ {data}`

is used for generating the display name of a single data element.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90015⌋

The corresponding path of a port prototype blueprint is

`({ARPackage.name}/)`$_{1..n}${`PortPrototypeBlueprint.name}`

The port interface names are ignored.

For the data prototype EngN in example in Figure 6.3 the complete instance pb-path would be

`AUTOSAR/PortPrototypeBlueprints_Blueprint/EngN/EngN`

and the generated display name according to [TR_MCG_00788] without any shortening would be

`AUTOSAR_PortPrototypeBlueprints_Blueprint_EngN_EngN`

This name is quite long, contains a lot of redundancy and does not fulfill all the requirements. Depending on specific additional rules and assumptions this name can be shortened. The rules for the port and data prototypes can be found in chapter 8. The rules for shortening the name space identifier can be found in chapter 11. The overall rules can be found in chapter 7.

In the following we will make some assumptions about virtual name spaces. If these assumptions are fulfilled in a system then the display names can be further shortened.

Virtual name spaces are defined in chapter 6.6.

Example (continue Figure 6.2 and Figure 6.3 in chapter 6.2):

Assume the *ARPackages* "AUTOSAR" and "OEM1" define a virtual name space for port prototype blueprints. Within this virtual name space the port prototype blueprint names are unique. Given this additional information the generated display name for data prototype "EngN" will be ([TR_MCM_70040], [TR_MCG_00120]):

`AR_EngN`

If there is only one virtual name space ([TR_MCA_80030]) covering the packages "AUTOSAR" and "OEM1" the name can even be reduced to

`EngN`

In the following we will express these assumptions and rules formally:

Port prototype blueprint names are unique within a given package, no additional assumption w.r.t. uniqueness of port prototype blueprint names within a given *ARPackage* is needed.

For being able to apply [TR_MCG_00120] (see chapter 11) the following assumption should be fulfilled for port prototype blueprints:

[TR_MCA_80020] ⌈Assumption: Each Top-Level *ARPackages* defines a virtual name space for port prototype blueprints.

Fulfills requirements: RS_MCR_90006⌋

The following rule is subsumed in rule [TR_MCG_00120] and thus does not define anything new. For better understandability it is stated explicitly.

[TR_MCG_00040] ⌈Only packages explicitly defining virtual name spaces for port prototype blueprints within a product or product family shall be considered in the display name.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

Example:

With rule [TR_MCG_00040] and [TR_MCM_70040] and assumption [TR_MCA_80020]

`AR_PortPrototypeBlueprints_Blueprint_EngNMax`

could be shortened to

`AR_EngNMax`

[TR_MCA_80030] ⌈Assumption: There is exactly one virtual name space for port prototype blueprints.

Fulfills requirements: RS_MCR_90006⌋

[TR_MCG_00045] ⌈If there is only one virtual name space for port prototype blueprints the {nameSpace}, i.e. all *ARPackage* names, can be completely ignored in the display name. We have

`portBlueprint : {data}`

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90006, RS_MCR_90011⌋

In [TR_MCG_00045] assumption [TR_MCA_80030] is assumed to be fulfilled.

Example:

With rule [TR_MCG_00045] and assumption [TR_MCA_80030]

`AR_EngNMax`

could be further shortened to

`EngNMax`

14 ComponentHierarchy

In this chapter we deal with the component hierarchy {componentHierarchy}. The {componentHierarchy} is relevant for prototypes ({prototype}). The general schema is:

`componentHierarchy : {RootSwCompositionPrototype.name} (_{SwComponentPrototype.name})`$_{0..n}$

Component hierarchies are relevant for sw signals because they are always instantiated in the context of a *SwComponentPrototype*. They are not relevant for port prototype blueprints and other first class elements of an *ARPackage*.

[TR_MCG_00705] ⌈The {componentHierarchy} is defined as follows:

`componentHierarchy : {RootSwCompositionPrototype.name} (_{SwComponentPrototype.name})`$_{0..n}$

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

In rule [TR_MCG_00706] assumption [TR_MCA_80710] (see chapter 11) is assumed to be fulfilled: there is only one system.

[TR_MCG_00706] ⌈If there is only one System relevant then the {componentHierarchy} is defined as follows:

`componentHierarchy : {SwComponentPrototype.name} (_{SwComponentPrototype.name})`$_{0..n}$

or

`componentHierarchy : {RootSwCompositionPrototype.name}`

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

For being able to apply [TR_MCG_00120] (see chapter 11) one of the following assumptions should be fulfilled for sw component prototypes:

Similar to [TR_MCA_80020] we have:

[TR_MCA_80715] ⌈Assumption: Sw component prototype names are unique within a given *ARPackage*, i.e. this *ARPackage* defines a virtual name space for *SwComponentPrototypes*.

Fulfills requirements: RS_MCR_90006⌋

[TR_MCA_80720] ⌈Assumption: Each Top-Level *ARPackage* defines a separate virtual name space for sw component prototypes.

Fulfills requirements: RS_MCR_90006, RS_MCR_90008⌋

[TR_MCG_00740] ⌈If an *ARPackage* defines a virtual name space for *SwComponentPrototypes* then the parent sw component prototype names of the sw component pro-

totypes defined within this package can be ignored in the display name of the sw signal or the sw component prototype, resp. - except for the first one and the ones that have the same *SwComponentType*, i.e. that are multiple instances of the same *SwComponentType*.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90006, RS_MCR_90011⌋

In rule [TR_MCG_00740] assumption [TR_MCA_80715] is assumed to be fulfilled.

The first sw component prototype name is necessary because it cannot be excluded that there are sw component prototypes with the same name in different packages.

[TR_MCA_80730] ⌈Assumption: There is exactly one virtual name space for sw component prototypes.

Fulfills requirements: RS_MCR_90006, RS_MCR_90008⌋

[TR_MCG_00750] ⌈If there is only one virtual name space for all sw component prototypes then the parent component names can be ignored in the display name except the ones that have the same *SwComponentType*, i.e. that are multiple instances of the same *SwComponentType*.

RS_MCR_90002, RS_MCR_90003, RS_MCR_90006, RS_MCR_90011⌋

In [TR_MCG_00750] assumption [TR_MCA_80730] is assumed to be fulfilled.

Generation rule [TR_MCG_00750] shortens the display name. However, strange names might be the result. So they should be checked and if necessary changed manually.



Figure 14.1: Example with Multiple Instantiation - Coordinator Wheels

Example (see Figure 14.1):

For *SwComponentType* Wheel we have two instances: WheelLeft and WheelRight.

For Coordinator we have the following paths:

```
/OEM1/Systems/System/TopLvl/Pt/Coordinator
```

and

```
/OEM1/Systems/System/TopLvl/Veh/Coordinator
/OEM1/Systems/System/TopLvl/Veh/Body/WheelLeft/Coordinator
/OEM1/Systems/System/TopLvl/Veh/Body/WheelRight/Coordinator
```

We have only one system, so [TR_MCA_80710] is fulfilled and [TR_MCG_00713] can be applied. Every top-level package defines a virtual name space for *SwComponent-Prototypes*, i.e. [TR_MCA_80720] is fulfilled. However, there is no global virtual name space for sw component prototypes, i.e. [TR_MCA_80730] is not fulfilled .

Assume we want to create display names for all sw component prototypes with name "Coordinator".

With rule [TR_MCG_00706] we would have the following display names for sw component prototype hierarchies ({componentHierarchy}):

```
Pt_Coordinator
Veh_Coordinator
Veh_Body_WheelLeft_Coordinator
```

and

```
Veh_Body_WheelRight_Coordinator
```

With rule [TR_MCG_00740] we have:

```
Pt_Coordinator
Veh_Coordinator
Veh_WheelLeft_Coordinator
```

and

```
Veh_WheelRight_Coordinator
```

15 SwComponentPrototypes

In this chapter we deal with the display name of a single *SwComponentPrototype*. The general schema for sw component prototypes (prototype : {component}) is

```
component :
  {nameSpace}_{componentHierarchy}_{data}
```

{data} corresponds to the name of the sw component prototype or root composition. I.e. the last component prototype is not part of the {componentHierarchy}. We have

```
component :
  {nameSpace}_{componentHierarchy}_{SwComponentPrototype.name}
```

or

```
component :
  {nameSpace}_{RootSwCompositionPrototype.name}
```

Component hierarchies are relevant for sw signals because they are always instantiated in the context of a *SwComponentPrototype*. They are not relevant for port prototype blueprints and other first class elements of an *ARPackage* ({element}).

[TR_MCG_00610] ⌈For sw component prototypes the following basic display name schema

```
component :
  {nameSpace}_{componentHierarchy}_{SwComponentPrototype.name}
```

is used for generating the display names.

In case the *SwComponentPrototype* corresponds to the *RootSwCompositionPrototype* we have

```
component :  {nameSpace}_{RootSwCompositionPrototype.name}
```

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

The corresponding path of a component is

$$(\{\text{ARPackage.name}\}/)_{1..n}\{\text{System.name}\}/$$
$$\{\text{RootSwCompositionPrototype.name}\}/$$
$$(\{\text{SwComponentPrototype.name}\}/)_{0..n}$$
$$(\{\text{SwComponentPrototype.name}\})_{0..1}$$

16 Unique SW-Signal Display Names

The general schema for sw signals looks like this ((prototype) : {swSignal}):

`swSignal : {nameSpace}_{componentHierarchy}_{data}`

with

`data : {port}_{dataInfo}_{typeId}`

[TR_MCG_00787] is derived from [TR_MCG_00001]:

[TR_MCG_00787] ⌈For multiple instantiation and in case several systems have to be considered the following display name schema for a sw signal

`swSignal : {nameSpace}_{componentHierarchy}_{data}`

is used for generating the display names of sw signals.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90015, RS_MCR_90016⌋

The instance cp-path of a sw signal is

$(\{ARPackage.name\}/)_{1..n}\{System.name\}/$
$\{RootSwCompositionPrototype.name\}/$
$(\{SwComponentPrototype.name\}/)_{0..n}\{PortPrototype.name\}/$
$(\{AutosarDataPrototype.name\})_{1..n}$

Example: (see Figure 16.1 ExtrLi)

Several systems to be considered.

SwComponentType: ExtrLiAdpr

SwComponentPrototypes referencing Type ExtrLiAdpr: ExtrLiAdprFrntLe and ExtrLiAdprFrntRi

Given the instance `cp-path`

`OEM1/Systems/System/TopLvl/ExtrLiAdprFrntRi/ActvnOfIndcFrntRi/Cmd`

this would lead to the following display name with rule [TR_MCG_00787]:

`OEM1_Systems_System_ ExtrLiAdprFrntRi_ActvnOfIndcFrntRi_Cmd`

The following rule is derived from [TR_MCG_00713]. Additionally assumption [TR_MCA_80710] is assumed to be fulfilled: there is only one system..

[TR_MCG_00716] ⌈For multiple instantiation for a single system the following basic display name schema

`swSignal : {componentHierarchy}_{data}`

is used for generating the display name of a sw signal

The {nameSpace}, the sw component type names, the package names and port interface names are ignored.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

In rule [TR_MCG_00716] assumption [: there is only one system.] is assumed to be fulfilled.

Example: (see Figure 16.1 ExtrLi)

Assumption: only one System

SwComponentType: ExtrLiAdpr

SwComponentPrototypes referencing Type ExtrLiAdpr: ExtrLiAdprFrntLe and ExtrLiAdprFrntRi

Given the instance cp-path

`OEM1/Systems/System/TopLvl/ExtrLiAdprFrntRi/ActvnOfIndcFrntRi/Cmd`

this would lead to the following display name applying [TR_MCG_00716]:

`ExtrLiAdprFrntRi_ActvnOfIndcFrntRi_Cmd`

And after applying [TR_MCG_00020], chapter 11, we would have

`ExtrLiAdprFrntRi_ActvnOfIndcFrntRi`

Since there has to be a FlatMap for the ECU-Extract specifying unique names for component prototypes the rules in chapter 16 can be applied to generate these unique component prototype names. Rule [TR_MCG_00760] can substitute rule [TR_MCG_00787].

[TR_MCG_00760] ⌈The unique component prototype names specified in a *FlatMap* can be used for specifying part of the display name of a sw signal. General Rule:

`swSignal : {componentDescriptor}_{data}`

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

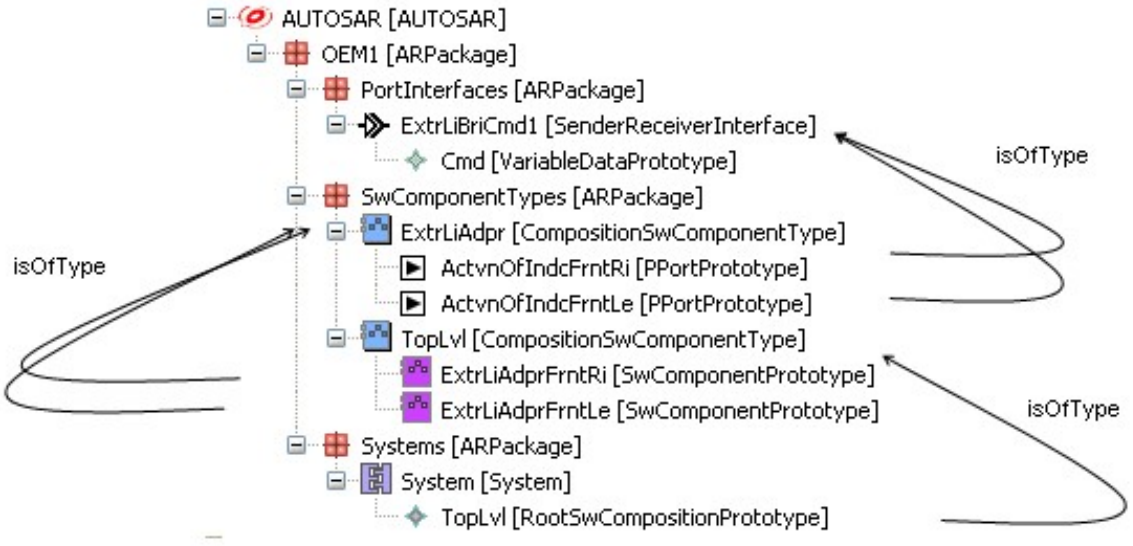


Figure 16.1: Example ExtrLi with Multiple Instantiation

Example (Figure 16.1 ExtrLi):

Assume we have the following FlatMap defining names for sw component prototypes:

```
<FLAT-MAP>
  <SHORT-NAME>ARMap</SHORT-NAME>
  <INSTANCES>
    <FLAT-INSTANCE-DESCRIPTOR>
      <SHORT-NAME>I1</SHORT-NAME>
      <UPSTREAM-REFERENCE-IREF>
        <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/OEM1/
            Systems/System/TopLvl</CONTEXT-ELEMENT-REF>
        <TARGET-REF DEST="SW-COMPONENT-PROTOTYPE">/OEM1/SwComponentTypes/
            TopLvl/ExtrLiAdprFrntRi</TARGET-REF>
      </UPSTREAM-REFERENCE-IREF>
    </FLAT-INSTANCE-DESCRIPTOR>
  </INSTANCES>
</FLAT-MAP>
```

Instead of

`TopLvl_ExtrLiAdprFrntRi_ActvnOfIndcFrntRi_Cmd`

with rule [TR_MCG_00760] and [TR_MCG_00020] and I1 being the unique short name defined in the FlatMap for TopLvl_ExtrLiAdprFrntRi we have:

`I1_ActvnOfIndcFrntRi`

[TR_MCA_80789] ⌈Assumption: There is exactly one virtual name space for port prototypes.

Fulfills requirements: RS_MCR_90006⌋

Similar to [TR_MCG_00045] we have the following rule:

[TR_MCG_00752] ⌈If there is only one virtual name space for port prototypes and the sw component types are only instantiated once then the component prototype names (the {componentHierarchy}) and the {nameSpace} can be ignored in the display name of a sw signal. We simply have:

```
swSignal :  {data}
```

Fulfills requirements: RS_MCR_90002, RS_MCR_90003, RS_MCR_90006, RS_MCR_90011⌋

If the port prototypes are based on port prototype blueprints the following rule holds:

[TR_MCG_00780] ⌈The display names specified in a *FlatMap* for the data elements within port prototype blueprints can be used for specifying the display name of a sw signal derived from this blueprint:

```
swSignal :  {portBlueprintDescriptor}
```

It has to be ensured that the display names of signals not derived from blueprints or for signals no flatmap with display names is available are different from those already normalized in the flatmaps for the blueprints.

Fulfills requirements: RS_MCR_90002, RS_MCR_90003⌋

The {componentHierarchy} can be ignored because there is no multiple instantiation. The {nameSpace} can be ignored because it is already contained in the blueprint descriptor name.

A Appendix: Keywords Phys/Log (P/L-List) for Powertrain Domain

The following table defines a P/L-list for the Powertrain Domain and can be used as input for the rules described in chapter 9.1.

The keyword abbreviations are defined as a view of the standardized keyword set of AUTOSAR called "PL_List".

The corresponding .arxml file containing the P/L-List is found as part of [1] and is called "AUTOSAR_MOD_AISpecification_Collection_AIMC_Keyword_Blueprint.arxml".

Keyword Abbreviation	Description
A	acceleration
Adr	address
Ag	angle
Ar	area
Cntr	counter
Fac	factor
Flg	flag, bit, boolean, binary signal
Frq	frequency
I	electric current
Idx	index
N	(rotational) speed
Nr	number
P	pressure
Posn	position
Pwr	power
R	resistance
Rat	ratio, duty cycle
St / Sts	state, status
T	temperature
Ti	time, duration
Tq	torque
U	voltage
V	velocity
Vol	volume
W	work

Table A.1: Keywords Phys/Log (P/L-List) for Powertrain Domain

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R26-11

B.1.1 Added Specification Items in R26-11

none

B.1.2 Changed Specification Items in R26-11

none

B.1.3 Deleted Specification Items in R26-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R25-11

B.2.1 Added Specification Items in R25-11

none

B.2.2 Changed Specification Items in R25-11

none

B.2.3 Deleted Specification Items in R25-11

none

B.3 Traceable item history of this document according to AUTOSAR Release R24-11

B.3.1 Added Specification Items in R24-11

none

B.3.2 Changed Specification Items in R24-11

none

B.3.3 Deleted Specification Items in R24-11

Number	Heading
[RS_MCR_90000]	
[RS_MCR_90001]	
[RS_MCR_90002]	
[RS_MCR_90003]	
[RS_MCR_90004]	
[RS_MCR_90005]	
[RS_MCR_90006]	
[RS_MCR_90008]	
[RS_MCR_90009]	
[RS_MCR_90011]	
[RS_MCR_90014]	
[RS_MCR_90015]	
[RS_MCR_90016]	
[RS_MCR_90781]	
[RS_MCR_90782]	
[RS_MCR_90783]	
[RS_MCR_90784]	

Table B.1: Deleted Specification Items in R24-11