| | |
|---|---|
| **Document Title** | Modeling Show Cases |
| **Document Owner** | AUTOSAR |
| **Document Responsibility** | AUTOSAR |
| **Document Identification No** | 789 |
| **Document Classification** | Auxiliary |

| | |
|---|---|
| **Document Status** | Final |
| **Part of AUTOSAR Standard** | Classic Platform |
| **Part of Standard Release** | 4.3.0 |

| **Document Change History** | | | |
|---|---|---|---|
| **Date** | **Release** | **Changed by** | **Description** |
| 2016-11-30 | 4.3.0 | AUTOSAR Release Management | Initial release |

**Advice for users**

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

# Table of Contents

# Bibliography

[1] Methodology
AUTOSAR_TR_Methodology

[2] Modeling Show Cases Examples
AUTOSAR_EXP_ModelingShowCases

[3] Software Component Template
AUTOSAR_TPS_SoftwareComponentTemplate

[4] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes

# 1 Introduction

The objective of this report is the illustration and execution of AUTOSAR modeling and the AUTOSAR methodology (see [1]) for selected show cases.

Each show case focuses on a few specific topics and gives an overview of their basic usage and their application in the field. Where appropriate, the show cases are based on real world applications of the AUTOSAR standard.

It contains

- explanatory background on the functional use case for which the specific part of the AUTOSAR modeling is applied.
- illustration of the AUTOSAR model content in form of interlinked tables
- explanation of the processing results of these AUTOSAR models (e.g. C code, A2L files, ...)
- snippets of the full-blown examples. The complete examples are provided in the archive `AUTOSAR_EXP_ModelingShowCases.zip` [2].

# 2 Overview

The report is organized in chapters according to the main focus of the contained show cases. Each chapter contains a topic specific overview and at least one show case. Each chapter is self contained and understandable without reading any other chapter.

The technical report on the AUTOSAR Methodology [1] deserves a special mentioning as accompanying document for going through the show cases.

In this first version of the technical report, the show cases are targeting the topic of measurement and calibration, involving the creation of A2L files based on AUTOSAR models. For these show cases, also the specification of the `SoftwareComponentTemplate` [3] is a good accompanying document.

# 3 Measurement and Calibration

Measurements and Calibration (short: MC) is a major step in the development of electronic control units (ECUs). Measurement and Calibration systems (MC systems), involving software tools (MC tools) as well as the hardware to access an ECU (not in focus here), enable the developer to measure variables and to adapt calibration parameters (or "characteristics") during the run-time of the ECU.

For instance, the following tasks are regularly done by "Measurement and Calibration"

- Adaptation to real hardware (e.g. inserting the electrical characteristics of a sensor)
- Calibration of controllers (e.g. adjusting the parameters of a closed loop controller)
- Tuning of ECU internal environment models (e.g. for "virtual sensors")
- Validation of ECU functions
- Tracking of development errors
- Collecting data for automated optimization of parameters

The "Introductory Show Case" (see 3.1), illustrates all basic artifacts on the way from a physical system that is to be controlled by an ECU until measuring and calibrating with a MC system.

As didactic simplification only a few data types were used, e.g. neither CURVEs[1] nor MAPs[2] were chosen, nor any ApplicationCompositeDataType.

However, those advanced topics, their modeling in AUTOSAR as well as their transfer to a MC tool, is of particular interest: they are regularly needed and used the field. Therefore, the "Advanced Show Case" in chapter 3.2 especially highlights these topics. This show case is directly derived from the real world modeling and structuring approach of a major Tier 1 in the powertrain domain. So it also illustrates "good practices" in the field for designing AUTOSAR systems which are to be measured and calibrated later on in their development.

[1] CURVEs are two dimensional functions defined via axis points and the corresponding function values. Interpolation or extrapolation is used to calculate function values that are not directly defined.

[2] MAPs are similar to CURVEs but three dimensional

## 3.1 Introductory Show Case

As introduction to measurement and calibration with AUTOSAR a simple, artificial closed-loop control system was chosen. This allows interesting feedback of the system when using a MC tool. At the same time, the model, the source code and generated files are still comprehensible.

A drawback is, that not all typical "real world" data types are featured, for instance. Such topics are covered in the "Advanced Show Case", chapter 3.2.

### 3.1.1 Physical System

This section contains a description of the physical system setup. It can safely be skipped, if only the AUTOSAR modeling itself is of interest to the reader.

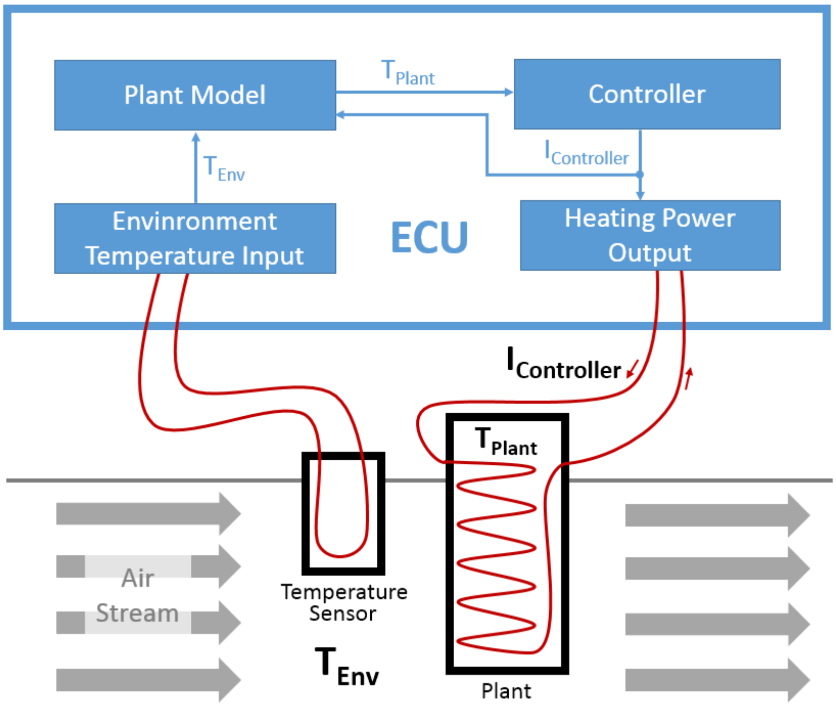


**Figure 3.1: Physical Overview**

Figure 3.1 shows the major physical values and entities of our system. The control task is the following: The plant is a sensor in an airstream that requires heating. The temperature $T_{Plant}$ is to be controlled by the ECU. However, there is no direct way to measure $T_{Plant}$.

For the estimation of $T_{Plant}$ the following properties of the system are used:

- $T_{Plant}$ depends on the temperature of the environment $T_{Env}$, i.e. the temperature of the air stream. $T_{Env}$ can be measured directly.

- $T_{Plant}$ depends on the current $I_{Controller}$ which is output by the ECU and controlled by the controller and therefore known.
- The plant itself acts as a thermal energy storage. So $T_{Plant}$ also depends on the heat quantity that is currently stored within the plant.
- All other influences on $T_{Plant}$ are considered to be insignificant. So they can safely be ignored for this control task.

An estimation of $T_{Plant}$ can be calculated by a plant model, which uses $T_{Env}$ and $I_{Controller}$ as inputs and has the stored heat quantity as internal state.

#### 3.1.1.1 Components Overview

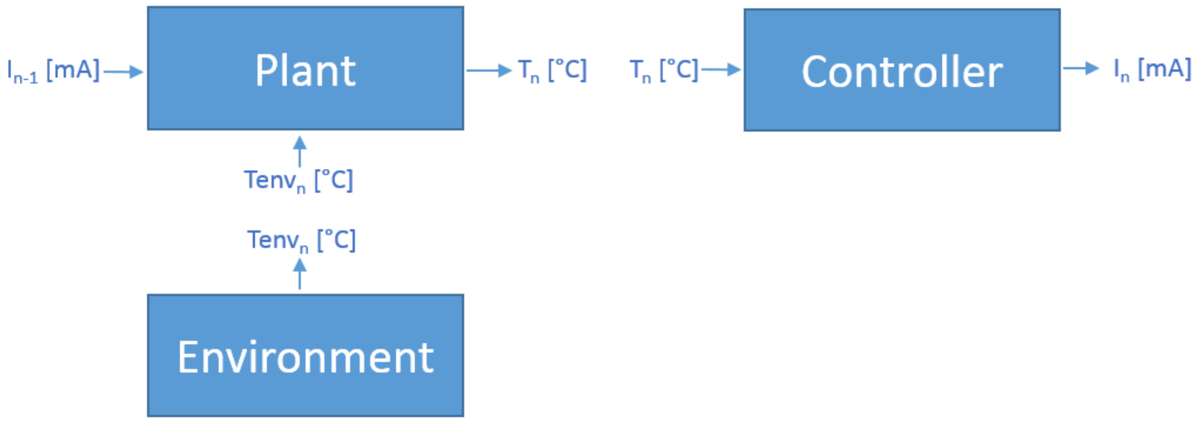


**Figure 3.2: Component Overview**

For this show case, the interaction with a real physical environment is completely left out, i.e. there is no `Heating Power Output` component and the profile of $T_{Env}$ is randomly generated inside the component `Environment`. This cuts off a lot of complexity from the example, and allows to run the software system on a PC without complex environment simulations.

For completeness: The plant model is calculated inside the component `Plant` and the controller inside the component `Controller`.

As typical for ECUs the calculations happen in a time-discrete manner, i.e. the calculations in the components are executed periodically at discrete in time steps. In the following, the index $n \in \{1, 2, ...\}$ denotes the current time step. The previous time step is denoted by the index $n - 1$. The index $0$ denotes the initialization value. This also means that time step $1$ is the first, that is actually calculated by the ECU.

Furthermore $\Delta t$ denotes the time in seconds, that elapsed between the calculation of the previous time step and the current time step. In case of time step $1$, $\Delta t$ denotes the time that elapsed between initialization of the system and time step $1$. For setting the actual value of $\Delta t$ the frequency bandwidth of the physical properties in the system has to be taken into account. Decreasing the value of $\Delta t$ usually increases the quality of the sampling of physical signals up to a certain point where the costs of further decreasing the value of $\Delta t$ outweighs the benefit gained in terms of signal quality.

#### 3.1.1.2 The Environment

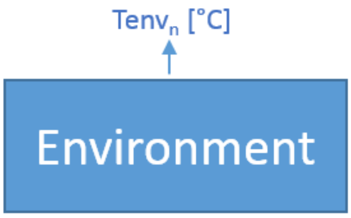


**Figure 3.3: Environment**

The modeling and implementation of the environment is not in the focus of this show case. The temperature $Tenv_n[°\mathtt{C}]$ is generated (pseudo) randomly. This is done in order to see the controller and the plant model "in action" during run-time of the system.

The generated profile is a random walk limited by an upper and a lower boundary, with saturation at these boundaries.

The random walk is configurable via $\mathtt{T_{LowLimit}}\,[°\mathtt{C}]$ and $\mathtt{T_{HighLimit}}\,[°\mathtt{C}]$, for the boundaries, and $\mathtt{T_{StepSize}}\,[\mathtt{K}]$, for the change of the temperature during one time step.

Assuming $rand_n\,[\text{-}] \in \{-1, 0, 1\}$ and $n \in \{1, 2, 3, ...\}$ then $Tenv_n$ is characterized by this equation (with $Tenv_0\,[°\mathtt{C}] \;=\; -273.15\,[°\mathtt{C}]$):

$$Tenv_n\,[°\mathtt{C}] \;=\; Tenv_{n-1}\,[°\mathtt{C}] + \mathtt{T_{StepSize}}\,[\mathtt{K}] \cdot rand_n\,[\text{-}]$$

if and only if $Tenv_n$ would be inside the boundaries, i.e.

$$\mathtt{T_{LowLimit}} \;<\; Tenv_n \;<\; \mathtt{T_{HighLimit}}$$

If $Tenv_n$ would be outside one of the boundaries, it is set to the value of that boundary.

#### 3.1.1.3 The Plant

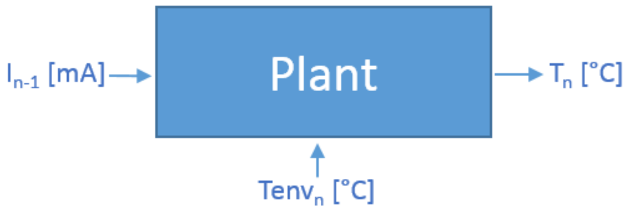


**Figure 3.4: Plant**

The plant is an electrically heated mass that is exposed to the air flow in the environment. The heat quantity $Qplant$ that is stored inside the plant is considered to always be directly proportional to the temperature $T$ with constant proportionality factor. Neither the mass of the plant, nor the specific heat capacity changes during the run-time of our system.

For simplicity, this proportionality factor is considered to be $1\,[\frac{\mathtt{J}}{\mathtt{K}}]$. For the calculations inside the `Plant` component, we are always using $[\mathtt{K}]$ as unit for temperatures, so the conversion from and to $[^\circ\mathtt{C}]$ only happens at the interface of the component.

With this, we have the following:

$$
\begin{aligned}
Qplant_n\,[\mathtt{J}] &= T_n\,[\mathtt{K}] \cdot 1\left[\frac{\mathtt{J}}{\mathtt{K}}\right] \\
T_n\,[\mathtt{K}] &= \frac{Qplant_n\,[\mathtt{J}]}{1\left[\frac{\mathtt{J}}{\mathtt{K}}\right]}
\end{aligned}
$$

This also means, that $Qplant_n\,[\mathtt{J}] = 0\,[\mathtt{J}]$ corresponds $T_n\,[\mathtt{K}] = 0\,[\mathtt{K}]$, i.e. absolute zero. So $Qplant_n\,[\mathtt{J}] \geq 0\,[\mathtt{J}]$ shall always be true.

In each time step, there are two heat flows: One from the electrical heater to the plant and one from the plant to the environment. A negative heat flow means that heat energy is flowing away from the plant. Respectively, a positive heat flow means that heat energy is stored in the plant.

The heat flow $Qheater_n\,[\mathtt{J}]$ from the electrical heater to the plant in one time step is considered to be proportional to the current $I_n\,[\mathtt{mA}]$ through the plant during this time step. The proportionality factor is $\mathtt{h}_{\mathtt{Heater}}\,[\frac{\mathtt{J}}{\mathtt{mA\,s}}]$. Of course, the plant can only be heated up by the electrical heater, i.e. a "negative" current $I_n$ would not cool down the plant, but causes the same heat up as $-I_n$. So we have

$$
Qheater_n\,[\mathtt{J}] = |\ I_n\ |\ [\mathtt{mA}] \cdot \mathtt{h}_{\mathtt{Heater}}\left[\frac{\mathtt{J}}{\mathtt{mA\,s}}\right] \cdot \Delta t\,[\mathtt{s}]
$$

The cool down of the plant can only happen via the second heat flow, i.e. the heat flow $Qenv_n\,[\mathtt{J}]$ from the plant to the environment. The flow in one time step is considered to be proportional to the difference between the temperature of the plant (calculated from the stored heat quantity during the last time step) and the temperature of the environment (received in this time step, but actually "measured" during the last time step). With the proportionality factor $\mathtt{h_{Env}}\,[\frac{\mathtt{J}}{\mathtt{K}}]$, we have:

$$Qenv_n\,[\mathtt{J}] \;=\; (Tenv_n\,[\mathtt{K}] - T_{n-1}\,[\mathtt{K}]) \cdot \mathtt{h_{Env}}\left[\frac{\mathtt{J}}{\mathtt{K}}\right] \cdot \Delta t\,[\mathtt{s}]$$

The heat quantity that was stored in the plant in last time step $Qplant_{n-1}$ is now modified by these two heat flows. This results in the stored heat quantity in the current time step. With $Qplant_0[\mathtt{J}] = 0[\mathtt{J}]$, we have

$$Qplant_n\,[\mathtt{J}] \;=\; Qplant_{n-1}\,[\mathtt{J}] + Qheater_n\,[\mathtt{J}] + Qenv_n\,[\mathtt{J}]$$

#### 3.1.1.4 The Controller


**Figure 3.5: Controller**

For the closed loop control an I controller (by and large) was chosen for component `Controller`. This means that the amplification of the input signal is proportional to the integral of the errors, i.e. the deviation between measured variable and setpoint. Because the controller cannot actively cool down the temperature of the plant, the output $I_n >= 0$ for all $n$.

Again, all temperatures are converted to and from $[^\circ\mathtt{C}]$ at the interface of the component. All internal calculation are done in $[\mathtt{K}]$.

The error during the current time step is the difference between $\mathtt{T_{SetPoint}}\,[\mathtt{K}]$ and the measured variable $T_n\,[\mathtt{K}]$:

$$e_n\,[\mathtt{K}] \;=\; \mathtt{T_{SetPoint}}\,[\mathtt{K}] - T_n\,[\mathtt{K}]$$

The integral part of the controller is calculated via summing up all errors from the previous steps. With $eSum_n\,[\mathtt{Ks}] = 0\,[\mathtt{Ks}]$ we have:

$$eSum_n\,[\mathtt{Ks}] \;=\; eSum_{n-1}\,[\mathtt{Ks}] + e_n\,[\mathtt{K}] \cdot \Delta t$$

A further design decision for the controller was, to limit the integral and to saturate at the limits. This has the benefit that it limits the current $I_n$ that is output by the controller. Furthermore, it enables the controller to react faster after long deviations.

The lower limit is $0\,[\mathtt{Ks}]$. So if $eSum_n$ would fall below zero in time step $n$, we set $eSum_n[\mathtt{Ks}] = 0\,[\mathtt{Ks}]$. The upper limit is $\mathtt{L}_{\mathtt{MaxESum}}\,[\mathtt{Ks}]$. If $eSum_n$ would exceed $L_{MaxESum}$ in time step $n$ we set $eSum_n[\mathtt{Ks}] = \mathtt{L}_{\mathtt{MaxESum}}\,[\mathtt{Ks}]$.

The integral state $eSum_n$ of the controller is then amplified by $\mathtt{k}\left[\frac{\mathtt{mA}}{\mathtt{Ks}}\right]$ to calculate the current $I_n\,[\mathtt{mA}]$, i.e. the output of the controller:

$$I_n\,[\mathtt{mA}] = eSum_n\,[\mathtt{Ks}] \cdot \mathtt{k}\left[\frac{\mathtt{mA}}{\mathtt{Ks}}\right]$$

So the limitations of the $eSum_n$ guarantee, that

$$0\,[\mathtt{mA}] \leq I_n\,[\mathtt{mA}] \leq \mathtt{L}_{\mathtt{MaxESum}}\,[\mathtt{Ks}] \cdot \mathtt{k}\left[\frac{\mathtt{mA}}{\mathtt{Ks}}\right]$$

### 3.1.2 AUTOSAR Modeling

This section gives a brief overview of the AUTOSAR modeling. More insight can be gained by browsing through the hyper-linked tables in section 3.1.7. These tables are generated from the AUTOSAR model of this show case. If this is still not sufficient, the complete model is available in .arxml format in AUTOSAR_EXP_ModelingShowCases.zip [2].

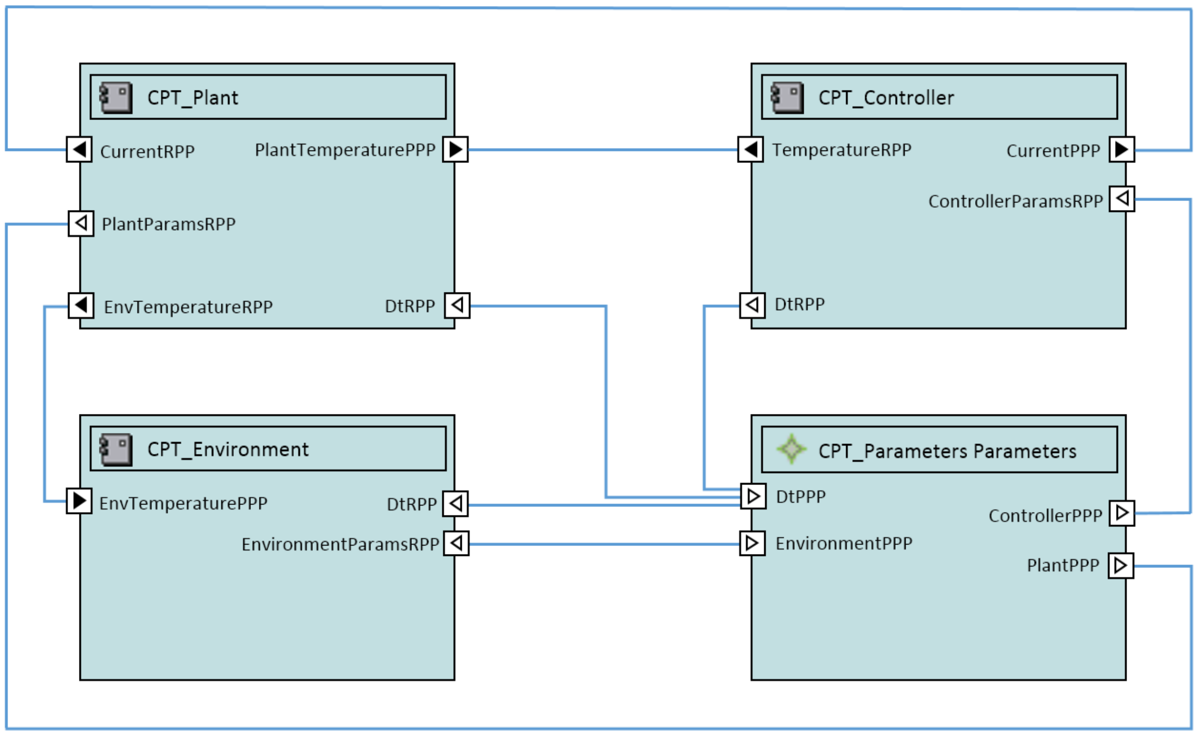


**Figure 3.6: Example Composition**

In this show case the components specified in section 3.1.1.1 are modeled as ApplicationSwComponentTypes.

- Environment
- Plant
- Controller

To keep the example simple, no SwcImplementations were modeled. For some tasks, like generation of a MemMap for an embedded controller, this would be needed.

The in- and outputs of the ApplicationSwComponentTypes are modeled as SenderReceiverInterface. The internal state is realized as implicitInterRunnableVariables. Besides the illustrative aspect, the rationale for this design decision was that the internal state is likely to be used by more than one runnable in ApplicationSwComponentTypes (at least "outside" of an introductory show case).

For variables that should just be available for measurement in a MC tool, arTypedPerInstanceMemorys are used. For this use case, no synchronization of access to the variable needs to be implemented, so the way with the least overhead was chosen.

All parameters in the specification of the components were put in a fourth SwComponentType, in the ParameterSwComponentType "Parameters".

A distinct ParameterInterface was defined for the parameters of each of the three ApplicationSwComponentTypes. The respective PPortPrototypes of the ParameterSwComponentType hold the initValue for each ParameterDataPrototype in the PortPrototypes. Each value is specified in a ValueSpecification aggregated by a ParameterProvideComSpec.

The component types are instantiated in the CompositionSwComponentType "Composition".

This Composition is the type of the rootSoftwareComposition of the ECU_Extract. This also implies that all SwComponentPrototypes of Composition are mapped to one EcuInstance.

Some information on the FlatMap can be found in section 3.1.3.1.

### 3.1.3 RTE Generation, Measurement and Calibration

The McSupport file is an interface between the RTE generator and the A2L generator. A RTE generator provides a McDataInstance for each calibrateable or measurable object. From logical view the generation of McSupport could be seen in two steps:

1. Provide unique names for all parameters, measurements, component prototypes which are instantiated one or multiple times. This is done by the used AUTOSAR Authoring Tool.

2. Generate the McSupport itself. This is usually done by the RTE generator. A2L supports only one global namespace, while AUTOSAR defines own namespaces within each ARPackage. This means, that on the one hand unique names are needed for all objects which are to be accessible during measurement and calibration (parameters, measurements, component prototypes). But on the other hand, unique names are needed for all other things that will appear in A2L, e.g. CompuMethods, Units. For them the RTE generator will create unique names.

AUTOSAR specifies additionally an AliasNameSet to override names which is not used here.

See AUTOSAR_EXP_ModelingShowCases.zip [2] for the generated Rte_McSupportData.arxml file.

#### 3.1.3.1 FlatMap

In this show case, the FlatMap gives unique names to the

- dataElements
- implicitInterRunnableVariables
- arTypedPerInstanceMemorys

The RTE-Generator uses this information for the generation of the McSupport file as well as for generation of the .c and .h files.

The FlatMap consists of a FlatInstanceDescriptor for each instance of these VariableDataPrototypes.

The flat map for this use case can be found in AUTOSAR_EXP_ModelingShowCases.zip [2].

#### 3.1.3.2 ECU Documentation, Measurement and Calibration

When developing an ECU one usual requirement is, that objects described in A2L can be easily found in the documentation of the ECU. This is a challenge since documentation is on the level of SwComponentTypes while A2L is defined on the level of a System of category "ECU_EXTRACT".

- The names of SwComponentPrototypes are potentially different to the names of SwComponentTypes
- The names of McDataInstances are potentially different to the names of DataPrototypes

The challenge gets bigger, if types are instantiated multiple times. This issue needs to be solved by proper architecture, modeling conventions and clever generation of the FlatMap.

In this show case, this topic is only slightly touched by instantiating TemperatureSRIF two times, for the interface transporting $T_{Env}$ as well as for the interface transporting $T_{Plant}$.

It is demonstrated that the FlatMap can be used to solve the issue. However, we manually crafted our FlatMap, which is usually not possible in the field. FlatMaps are usually automatically generated by customizable, "clever", not standardized tools.

### 3.1.4 A2L File

With the information in the McSupport file an A2L file is generated. However, for this generation the memory addresses for the variables and characteristics are needed. They are usually extracted from the map file that is output by the linker of the ECU executable. The exact process as well as the tool for the A2L file generation is not standardized.

An example A2L file is provided in AUTOSAR_EXP_ModelingShowCases.zip.

### 3.1.5 Implementation in C

The implementation in C is a straight forward realization of the physical specification within the AUTOSAR modeling (see section 3.1.1.1 and 3.1.2). Therefore, the listings are presented without further explanation besides the comments in the source code.

A remark on the (pseudo) random numbers generated in line 22 of `Environment.c` (Listing 3.1): The numbers don't have good "pseudo randomness" properties but are sufficient for this show case, nevertheless. This way of generation was only chosen, because it fits in one line of C code without introducing a dependency to a library.

**Listing 3.1: Environment.c**

```
1  #include "Rte_Environment.h"
2
3  #define envRE_START_SEC_CODE
4  #include "Environment_MemMap.h"
5
6  FUNC (void, Environment_CODE) envRE_func (void)
7  {
8     /* read parameters for simulation of the temperature profile   */
9     float32 lLowLimit    = Rte_Prm_EnvParamsRPP_env_TLowLimit();
10    float32 lStepSize    = Rte_Prm_EnvParamsRPP_env_TStepSize();
11
12    /* retrieve internal state                                      */
13    uint32  lSeed      = Rte_IrvIRead_envRE_Seed();
14    float32 lTEnv      = Rte_IrvIRead_envRE_TEnv();
15    float32 direction = (float32)(lSeed % 3) - 1.0;
16
17    /* calc high limit with parameter, store for measurement       */
18    *Rte_Pim_THighLimit()
19         = lLowLimit + Rte_Prm_EnvParamsRPP_env_THighLimitDistance();
20
21    /*  update state for pseudo random number generation           */
22    lSeed = (8253729 * lSeed + 2396403);
23
24    /* calculate environment temperature                            */
25    lTEnv += lStepSize * direction;
26
27    /* saturating environment temperature at the bounds            */
28    if( lTEnv < lLowLimit)  { lTEnv = lLowLimit; }
29    if( lTEnv > *Rte_Pim_THighLimit())
30                            { lTEnv = *Rte_Pim_THighLimit(); }
31
32    /* Store internal state                                         */
33    Rte_IrvIWrite_envRE_Seed(lSeed);
34    Rte_IrvIWrite_envRE_TEnv(lTEnv);
35
36    /* write output                                                 */
37    Rte_IWrite_envRE_EnvTemperaturePPP_T(lTEnv);
38 }
39 #define envRE_STOP_SEC_CODE
40 #include "Environment_MemMap.h"
```

**Listing 3.2: Plant.c**

```c
#include "Rte_Plant.h"

#define plantRE_START_SEC_CODE
#include "Plant_MemMap.h"

FUNC (void, Plant_CODE) plantRE_func (void)
{
   /* read input                                                  */
   float32 lTenv     = Rte_IRead_plantRE_EnvTemperatureRPP_T();
   float32 lI        = Rte_IRead_plantRE_CurrentRPP_I();

   /* retrieve internal state                                     */
   float32 lQPlant   = Rte_IrvIRead_plantRE_QPlant();

   /* read parameters                                             */
   float32 lDt       = Rte_Prm_DtRPP_Dt();
   float32 lEFactor = Rte_Prm_PlantParamsRPP_plnt_EnvFactor();
   float32 lHFactor = Rte_Prm_PlantParamsRPP_plnt_HeaterFactor();

   /* heat capacity of 1 assumed                                  */
   float32 lTPlant      = lQPlant;

    /* calculate heat flows, store in PIM to make them measurable */
   *Rte_Pim_QEnv()      = (lTenv - lTPlant) * lEFactor * lDt;
   *Rte_Pim_QHeater()   = lI * lHFactor * lDt;

   /* update heat quantity in plant                               */
   lQPlant   = lQPlant + *Rte_Pim_QHeater() + *Rte_Pim_QEnv();

   /* limit heat quantity to absolute zero                        */
   lQPlant   = lQPlant < 0 ? 0 : lQPlant;

   /* heat capacity of 1 assumed                                  */
   lTPlant   = lQPlant;

   /* store internal state of plant: stored heat quantity         */
   Rte_IrvIWrite_plantRE_QPlant(lQPlant);

   /* Write output of plant: temerature of plant                  */
   Rte_IWrite_plantRE_PlantTemperaturePPP_T(lTPlant);
}
#define plantRE_STOP_SEC_CODE
#include "Plant_MemMap.h"
```

**Listing 3.3: Controller.c**

```
#include "Rte_Controller.h"

#define ControllerRE_START_SEC_CODE
#include "Controller_MemMap.h"

FUNC (void, Controller_CODE) controllerRE_func (void)
{
   /* read input, define output variable                          */
   float32 lT        = Rte_IRead_ControllerRE_TemperatureRPP_T();
   float32 lI;

   /* retrieve internal state: Sum of errors until last time step */
   float32 lESum     = Rte_IrvIRead_ControllerRE_ESum();

   /* read parameters                                             */
   float32 lDt       = Rte_Prm_DtRPP_Dt();
   float32 lSetPoint = Rte_Prm_ControllerParamsRPP_ctrl_SetPoint();
   float32 lK        = Rte_Prm_ControllerParamsRPP_ctrl_K();
   float32 lMaxESum  = Rte_Prm_ControllerParamsRPP_ctrl_MaxESum();

    /* store current error in PIM to make it measurable           */
   *Rte_Pim_E() = lSetPoint - lT;

    /* update eSum                                                */
   lESum += *Rte_Pim_E() * lDt;

   /* limit   eSum                                                */
   if(lESum > lMaxESum) { lESum = lMaxESum; }
   if(lESum < 0)        { lESum = 0;        }

   /* Controller equation: Calculation of manipulated variable    */
   lI = lESum * lK;

   /* Store internal state                                        */
   Rte_IrvIWrite_ControllerRE_ESum(lESum);

   /* Write output of controller                                  */
   Rte_IWrite_ControllerRE_CurrentPPP_I(lI);
}
#define ControllerRE_STOP_SEC_CODE
#include "Controller_MemMap.h"
```

### 3.1.6 A walk with T_Plant through the Show Case

This section revisits the complete show case, but focuses on one physical value: $T_{Plant}$. It visits all artifacts and highlights all places that relate to $T_{Plant}$ to illustrate the dependencies between all artifacts.

#### 3.1.6.1 Physical System

Our journey begins at the physical system, where the value of the physical system outside of the ECU is identified with a software value inside the ECU.

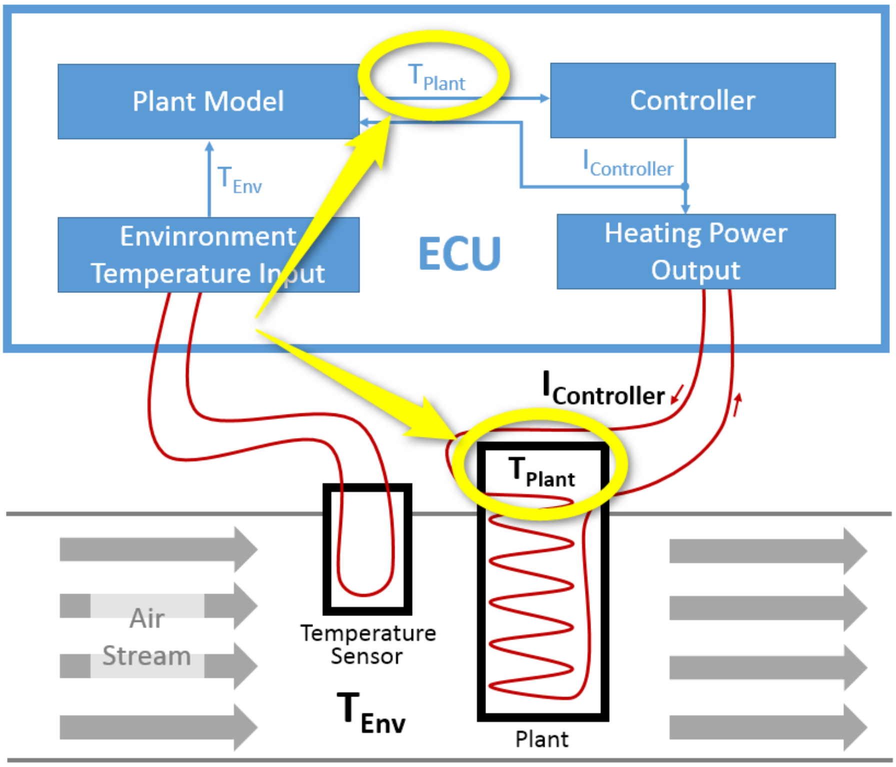


**Figure 3.7: Physical Overview**

##### 3.1.6.1.1 Components

It was located at the interface between two architectural components, sent by the Plant and received by the Controller. Furthermore a sequencing was introduced[3], i.e. in one time step the Plant is calculated before the Controller.

[3] Please note that this sequencing is a design decision. As there is also a data flow from the Plant to the Controller one could also argue for another calculation sequence.

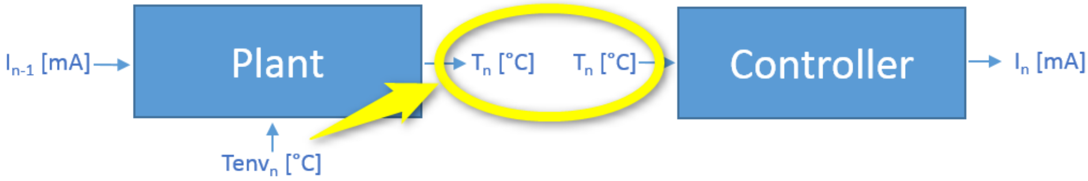


**Figure 3.8: Component Overview**

#### 3.1.6.1.2 Equations

The functional behavior is defined by the equations for the `Plant`

$$Qplant_n\,[\mathtt{J}] = T_n\,[\mathtt{K}]\;1\left[\frac{\mathtt{J}}{\mathtt{K}}\right]$$

$$T_n\,[\mathtt{K}] = \frac{Qplant_n\,[\mathtt{J}]}{1\left[\frac{\mathtt{J}}{\mathtt{K}}\right]}$$

**Figure 3.9: Dependency between $Q_{Plant}$ and $T_{Plant}$**

$$Qenv_n\,[\mathtt{J}] = (Tenv_n\,[\mathtt{K}] - T_{n-1}\,[\mathtt{K}]) \cdot \mathtt{h_{Env}}\left[\frac{\mathtt{J}}{\mathtt{K}}\right] \cdot \Delta t\,[\mathtt{s}]$$

**Figure 3.10: Heat flow from the `Plant` to the `Environment`**

$T_{Plant}$ is also used by the physical equations in the component `Controller`:

$$e_n\,[\mathtt{K}] = \mathtt{T_{SetPoint}}\,[\mathtt{K}] - T_n\,[\mathtt{K}]$$

**Figure 3.11: Calculation of the control error in the `Controller`**

Furthermore, calculations inside the components are done in Kelvin $[\mathtt{K}]$. The conversion from and to $[^\circ\mathtt{C}]$ happens at the interface level.

### 3.1.6.2 AUTOSAR Modeling

This architecture, i.e. the layout of the physical system, is modeled in AUTOSAR. The functional behavior defined by the equations will be implemented in C Code later on.

#### 3.1.6.2.1 Physical Dimension and Unit

A `PhysicalDimension` is defined: $T_{Plant}$ is a temperature.

| Common PhysicalDimension attributes | |
|---|---|
| shortName | Temperature |
| currentExp | 0 |
| lengthExp | 0 |
| luminousIntensity-Exp | 0 |
| massExp | 0 |
| molarAmountExp | 0 |
| temperatureExp | 1 |
| timeExp | 0 |

**Table 3.1: PhysicalDimension Temparature**

The corresponding ARXML description is:

**Listing 3.4: Physical Dimension of Temperature**

```
<PHYSICAL-DIMENSION>
  <SHORT-NAME>Temperature</SHORT-NAME>
  <LENGTH-EXP>0</LENGTH-EXP>
  <MASS-EXP>0</MASS-EXP>
  <TIME-EXP>0</TIME-EXP>
  <CURRENT-EXP>0</CURRENT-EXP>
  <TEMPERATURE-EXP>1</TEMPERATURE-EXP>
  <MOLAR-AMOUNT-EXP>0</MOLAR-AMOUNT-EXP>
  <LUMINOUS-INTENSITY-EXP>0</LUMINOUS-INTENSITY-EXP>
</PHYSICAL-DIMENSION>
```

$T_{Plant}$ shall have the Unit DegreeCelsius:

| Common Unit attributes | |
|---|---|
| shortName | DegreeCelsius |
| displayName | °C |
| offsetSiToUnit | -273.15 |
| factorSiToUnit | 1.0 |
| physicalDimension | Temparature |

**Table 3.2: Unit DegreeCelsius**

The corresponding ARXML description is:

**Listing 3.5: Unit Degree Celsius**

```
<UNIT>
  <SHORT-NAME>DegreeCelsius</SHORT-NAME>
  <DISPLAY-NAME>°C</DISPLAY-NAME>
  <FACTOR-SI-TO-UNIT>1.0</FACTOR-SI-TO-UNIT>
  <OFFSET-SI-TO-UNIT>-273.15</OFFSET-SI-TO-UNIT>
  <PHYSICAL-DIMENSION-REF DEST="PHYSICAL-DIMENSION">
    /McInt/PhysicalDimensions/Temparature
  </PHYSICAL-DIMENSION-REF>
```

```
</UNIT>
```

The following is presented for completeness, although not directly needed for $T_{Plant}$. It is possible to link more than one unit to a physical dimension. So in the model, there is also a definition for the unit Kelvin:

| Common Unit attributes | |
|---|---|
| shortName | Kelvin |
| displayName | K |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 1.0 |
| physicalDimension | Temperature |

**Table 3.3: Unit Kelvin**

The corresponding ARXML Code is:

**Listing 3.6: Unit Kelvin**

```
<UNIT>
  <SHORT-NAME>Kelvin</SHORT-NAME>
  <DISPLAY-NAME>K</DISPLAY-NAME>
  <FACTOR-SI-TO-UNIT>1.0</FACTOR-SI-TO-UNIT>
  <OFFSET-SI-TO-UNIT>0.0</OFFSET-SI-TO-UNIT>
  <PHYSICAL-DIMENSION-REF DEST="PHYSICAL-DIMENSION">
    /McInt/PhysicalDimensions/Temparature
  </PHYSICAL-DIMENSION-REF>
</UNIT>
```

#### 3.1.6.2.2 Application Data Type

A new ApplicationDataType is defined for temperatures in degree Celsius:

| Common ApplicationDataType attributes | | | |
|---|---|---|---|
| shortName | Temperature_C | | |
| category | VALUE | | |
| desc | Type for a temperature in [°C] | | |
| swCalibrationAccess | readOnly | | |
| unit | DegreeCelsius | | |
| Range | | | |
| Conversion | | | |
| category | LINEAR | | |
| direction | compuInternalToPhys | | |
| desc | lowerLimit | upperLimit | compuNumerator / compuDenominator |
| - | - | - | $Phys = \frac{-273.15 + 1 * Internal}{1}$ |

**Table 3.4: ApplicationDataType Temperature_C**

The corresponding ARXML Code is split between the definition of the ApplicationDataType:

**Listing 3.7: Datatype**

```
<APPLICATION-PRIMITIVE-DATA-TYPE>
  <SHORT-NAME>Temperature_C</SHORT-NAME>
  <DESC>
  <L-2 L="EN">Type for a temperature in [°C]</L-2>
  </DESC>
  <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>
    <COMPU-METHOD-REF DEST="COMPU-METHOD">
      /McInt/CompuMethods/Temperature_C
    </COMPU-METHOD-REF>
    <UNIT-REF DEST="UNIT">/McInt/Units/DegreeCelsius</UNIT-REF>
    </SW-DATA-DEF-PROPS-CONDITIONAL>
  </SW-DATA-DEF-PROPS-VARIANTS>
  </SW-DATA-DEF-PROPS>
</APPLICATION-PRIMITIVE-DATA-TYPE>
```

and the CompuMethod, which is referenced by the ApplicationDataType:

**Listing 3.8: Conversion**

```
<COMPU-METHOD>
  <SHORT-NAME>Temperature_C</SHORT-NAME>
  <DESC>
  <L-2 L="EN">Conversion from [°C] to [K]</L-2>
  </DESC>
  <CATEGORY>LINEAR</CATEGORY>
```

```
    <DISPLAY-FORMAT>%.1f</DISPLAY-FORMAT>
    <UNIT-REF DEST="UNIT">/McInt/Units/DegreeCelsius</UNIT-REF>
    <COMPU-INTERNAL-TO-PHYS>
    <COMPU-SCALES>
      <COMPU-SCALE>
      <COMPU-RATIONAL-COEFFS>
        <COMPU-NUMERATOR>
        <V>-273.15</V>
        <V>1</V>
        </COMPU-NUMERATOR>
        <COMPU-DENOMINATOR>
        <V>1</V>
        </COMPU-DENOMINATOR>
      </COMPU-RATIONAL-COEFFS>
      </COMPU-SCALE>
    </COMPU-SCALES>
    </COMPU-INTERNAL-TO-PHYS>
  </COMPU-METHOD>
```

This ApplicationDataType is mapped to the ImplementationDataType float32. The DataTypeMappingSet that contains this DataTypeMap is referenced inside the SwcInternalBehaviors of the ApplicationSwComponentTypes presented later on.

**Listing 3.9: Type Mapping**

```
<DATA-TYPE-MAPPING-SET>
  <SHORT-NAME>DataTypeMappingSet</SHORT-NAME>
  <DATA-TYPE-MAPS>
  <DATA-TYPE-MAP>
    <APPLICATION-DATA-TYPE-REF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">
       /McInt/ApplicationDataTypes/Temperature_C
    </APPLICATION-DATA-TYPE-REF>
    <IMPLEMENTATION-DATA-TYPE-REF DEST="IMPLEMENTATION-DATA-TYPE">
       /AUTOSAR_PlatformTypes/ImplementationDataTypes/float32
    </IMPLEMENTATION-DATA-TYPE-REF>
  </DATA-TYPE-MAP>
  ...
  </DATA-TYPE-MAPS>
</DATA-TYPE-MAPPING-SET>
```

For completeness, also the ARXML containing the definition of float32 is inserted here:

**Listing 3.10: Implementation Type and Base Type**

```
<AR-PACKAGE>
  <SHORT-NAME>AUTOSAR_PlatformTypes</SHORT-NAME>
  <AR-PACKAGES>
  <AR-PACKAGE>
    <SHORT-NAME>ImplementationDataTypes</SHORT-NAME>
    <ELEMENTS>
    <IMPLEMENTATION-DATA-TYPE>
      <SHORT-NAME>float32</SHORT-NAME>
      <CATEGORY>VALUE</CATEGORY>
      <SW-DATA-DEF-PROPS>
```

```
          <SW-DATA-DEF-PROPS-VARIANTS>
            <SW-DATA-DEF-PROPS-CONDITIONAL>
            <BASE-TYPE-REF DEST="SW-BASE-TYPE">/AUTOSAR_PlatformTypes/
                SwBaseTypes/float32</BASE-TYPE-REF>
            </SW-DATA-DEF-PROPS-CONDITIONAL>
          </SW-DATA-DEF-PROPS-VARIANTS>
          </SW-DATA-DEF-PROPS>
        </IMPLEMENTATION-DATA-TYPE>
        ...
        </ELEMENTS>
      </AR-PACKAGE>
      <AR-PACKAGE>
        <SHORT-NAME>SwBaseTypes</SHORT-NAME>
        <ELEMENTS>
        <SW-BASE-TYPE>
          <SHORT-NAME>float32</SHORT-NAME>
          <CATEGORY>FIXED_LENGTH</CATEGORY>
          <BASE-TYPE-SIZE>32</BASE-TYPE-SIZE>
          <BASE-TYPE-ENCODING>IEEE754</BASE-TYPE-ENCODING>
        </SW-BASE-TYPE>
        ...
        </ELEMENTS>
      </AR-PACKAGE>
      ...
    </AR-PACKAGE>
```

#### 3.1.6.2.3 Port Interface

The Temperature_C is used to define the SenderReceiverInterface which is used to type the "transport" of a temperature in degree Celsius between SwComponentTypes. Please note that in the show case, this PortInterface is not only used to type the "transport" of $T_{Plant}$, but also to type the "transport" of $T_{Env}$.

| Common SenderReceiverInterface attributes | |
|---|---|
| shortName | TemperatureSRIF |
| desc | Interface type for transferring temperatures in [°C] |
| properties of the dataElementss | |
| properties of VariableDataPrototype | |
| shortName | T |
| type | Temperature_C |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | VAR |

Table 3.5: SenderReceiverInterface TemperatureSRIF

In ARXML:

Listing 3.11: Port Interface

```
<SENDER-RECEIVER-INTERFACE>
  <SHORT-NAME>TemperatureSRIF</SHORT-NAME>
  <DESC>
  <L-2 L="EN">Interface type for transferring temperatures in [°C]</L-2
      >
  </DESC>
  <IS-SERVICE>false</IS-SERVICE>
  <DATA-ELEMENTS>
  <VARIABLE-DATA-PROTOTYPE>
    <SHORT-NAME>T</SHORT-NAME>
    <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
      <SW-ADDR-METHOD-REF DEST="SW-ADDR-METHOD">/McInt/SwAddrMethods/
          VAR</SW-ADDR-METHOD-REF>
      <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
      <SW-IMPL-POLICY>STANDARD</SW-IMPL-POLICY>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
    </SW-DATA-DEF-PROPS>
    <TYPE-TREF DEST="APPLICATION-PRIMITIVE-DATA-TYPE">/McInt/
        ApplicationDataTypes/Temperature_C</TYPE-TREF>
  </VARIABLE-DATA-PROTOTYPE>
  </DATA-ELEMENTS>
</SENDER-RECEIVER-INTERFACE>
```

For completeness, also the referenced SwAddrMethod is described here:

| Common SwAddrMethod attributes | |
|---|---|
| shortName | VAR |
| desc | Memory section for variables |
| sectionType | var |
| memoryAllocationKeywordPolicy | addrMethodShortName |
| sectionInitializationPolicy | - |
| option | safetyQM |

**Table 3.6: SwAddrMethod VAR**

In ARXML:

**Listing 3.12: Software Address Method**

```
<SW-ADDR-METHOD>
  <SHORT-NAME>VAR</SHORT-NAME>
  <DESC>
  <L-2 L="EN">Memory section for variables</L-2>
  </DESC>
  <OPTIONS>
  <OPTION>safetyQM</OPTION>
  </OPTIONS>
</SW-ADDR-METHOD>
```

#### 3.1.6.2.4 Software Components

The two ApplicationSwComponentTypes Controller and Plant are using $T_{Plant}$.

In Plant a PPortPrototype, typed by TemperatureSRIF, is defined for sending out $T_{Plant}$.

Furthermore dataWriteAccess is granted to the single RunnableEntity in this ApplicationSwComponentType. You also see the symbol, i.e. the name of the implementing C function, as well as the TimingEvent that triggers the execution of the RunnableEntity. These two are of further interest for tying together the system.

| Common **ApplicationSwComponentType** attributes | |
|---|---|
| **shortName** | Plant |
| **properties of the ports** | |
| **properties of PPortPrototype** | |
| **shortName** | PlantTemperaturePPP |
| **desc** | Port for sending out the estimated temperature of the plant |
| **providedInterface** | TemperatureSRIF |
| **[ ... ]** | |
| **internalBehavior** | PlantInternalBehavior |
| **[ ... ]** | |
| **properties of the runnables** | |
| **properties of RunnableEntity** | |
| **shortName** | plantRE |
| **symbol** | plantRE_func |
| **properties of the events** | |
| **properties of TimingEvent** | |
| **shortName** | plant100ms |
| **startOnEvent** | plantRE |
| **period** | 0.1 |

**Table 3.7: ApplicationSwComponentType Plant**

In ARXML:

**Listing 3.13: Plant**

```
<APPLICATION-SW-COMPONENT-TYPE>
  <SHORT-NAME>Plant</SHORT-NAME>
  <PORTS>
  <P-PORT-PROTOTYPE>
    <SHORT-NAME>PlantTemperaturePPP</SHORT-NAME>
    <DESC>
    <L-2 L="EN">Port for sending out the estimated temperature of the
        plant</L-2>
```

```
    </DESC>
    <PROVIDED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">
     /McInt/PortInterfaces/TemperatureSRIF
    </PROVIDED-INTERFACE-TREF>
  </P-PORT-PROTOTYPE>
  ...
  </PORTS>
  <INTERNAL-BEHAVIORS>
  <SWC-INTERNAL-BEHAVIOR>
    <SHORT-NAME>PlantInternalBehavior</SHORT-NAME>
    <DATA-TYPE-MAPPING-REFS>
    <DATA-TYPE-MAPPING-REF DEST="DATA-TYPE-MAPPING-SET">
      /McInt/DataTypeMappings/DataTypeMappingSet
    </DATA-TYPE-MAPPING-REF>
    </DATA-TYPE-MAPPING-REFS>
    <EVENTS>
    <TIMING-EVENT>
      <SHORT-NAME>plant100ms</SHORT-NAME>
      <START-ON-EVENT-REF DEST="RUNNABLE-ENTITY">
       /McInt/SwComponents/Plant/PlantInternalBehavior/plantRE
      </START-ON-EVENT-REF>
      <PERIOD>0.1</PERIOD>
    </TIMING-EVENT>
    </EVENTS>
    ...
    <RUNNABLES>
    <RUNNABLE-ENTITY>
      <SHORT-NAME>plantRE</SHORT-NAME>
      <DATA-WRITE-ACCESSS>
      <VARIABLE-ACCESS>
        <SHORT-NAME>DWA_PlantTemperature</SHORT-NAME>
        <ACCESSED-VARIABLE>
        <AUTOSAR-VARIABLE-IREF>
          <PORT-PROTOTYPE-REF DEST="P-PORT-PROTOTYPE">
             /McInt/SwComponents/Plant/PlantTemperaturePPP
          </PORT-PROTOTYPE-REF>
          <TARGET-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">
            /McInt/PortInterfaces/TemperatureSRIF/T
          </TARGET-DATA-PROTOTYPE-REF>
        </AUTOSAR-VARIABLE-IREF>
        </ACCESSED-VARIABLE>
      </VARIABLE-ACCESS>
      </DATA-WRITE-ACCESSS>
      ...
    </RUNNABLE-ENTITY>
    </RUNNABLES>
  </SWC-INTERNAL-BEHAVIOR>
  </INTERNAL-BEHAVIORS>
</APPLICATION-SW-COMPONENT-TYPE>
```

In `Controller` a `RPortPrototype`, typed by `TemperatureSRIF`, is defined for receiving $T_{Plant}$.

Furthermore `dataReadAccess` is granted to the single `RunnableEntity` in this `ApplicationSwComponentType`. You also see the `symbol`, i.e. the name of the implementing C function, as well as the TimingEvent that triggers the execution of the RunnableEntity. These two are of further interest for tying together the system.

| Common ApplicationSwComponentType attributes | |
|---|---|
| shortName | Controller |
| **properties of the ports** | |
| **properties of RPortPrototype** | |
| shortName | TemperatureRPP |
| desc | Port to receive the temperature of the plant |
| requiredInterface | TemperatureSRIF |
| **[ ... ]** | |
| internalBehavior | ControllerInternalBehavior |
| **[ ... ]** | |
| **properties of the runnables** | |
| **properties of RunnableEntity** | |
| shortName | ControllerRE |
| symbol | controllerRE_func |
| **properties of the events** | |
| **properties of TimingEvent** | |
| shortName | controller100ms |
| startOnEvent | ControllerRE |
| period | 0.1 |

**Table 3.8: ApplicationSwComponentType Controller**

In ARXML:

**Listing 3.14: Controller**

```
<APPLICATION-SW-COMPONENT-TYPE>
  <SHORT-NAME>Controller</SHORT-NAME>
  <PORTS>
  <R-PORT-PROTOTYPE>
    <SHORT-NAME>TemperatureRPP</SHORT-NAME>
    <DESC>
    <L-2 L="EN">Port to receive the temperature of the plant</L-2>
    </DESC>
    <REQUIRED-INTERFACE-TREF DEST="SENDER-RECEIVER-INTERFACE">
      /McInt/PortInterfaces/TemperatureSRIF
     </REQUIRED-INTERFACE-TREF>
  </R-PORT-PROTOTYPE>
  ...
  </PORTS>
  <INTERNAL-BEHAVIORS>
  <SWC-INTERNAL-BEHAVIOR>
    <SHORT-NAME>ControllerInternalBehavior</SHORT-NAME>
    <DATA-TYPE-MAPPING-REFS>
    <DATA-TYPE-MAPPING-REF DEST="DATA-TYPE-MAPPING-SET">
```

```
        /McInt/DataTypeMappings/DataTypeMappingSet
      </DATA-TYPE-MAPPING-REF>
      </DATA-TYPE-MAPPING-REFS>
      <EVENTS>
      <TIMING-EVENT>
        <SHORT-NAME>controller100ms</SHORT-NAME>
        <START-ON-EVENT-REF DEST="RUNNABLE-ENTITY">
         /McInt/SwComponents/Controller/ControllerInternalBehavior/
            ControllerRE
        </START-ON-EVENT-REF>
        <PERIOD>0.1</PERIOD>
      </TIMING-EVENT>
      </EVENTS>
      ...
      <RUNNABLES>
      <RUNNABLE-ENTITY>
        <SHORT-NAME>ControllerRE</SHORT-NAME>
        <DATA-READ-ACCESSS>
        <VARIABLE-ACCESS>
          <SHORT-NAME>DRA_temperature</SHORT-NAME>
          <ACCESSED-VARIABLE>
          <AUTOSAR-VARIABLE-IREF>
            <PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">
             /McInt/SwComponents/Controller/TemperatureRPP
            </PORT-PROTOTYPE-REF>
            <TARGET-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-PROTOTYPE">
             /McInt/PortInterfaces/TemperatureSRIF/T
            </TARGET-DATA-PROTOTYPE-REF>
          </AUTOSAR-VARIABLE-IREF>
          </ACCESSED-VARIABLE>
        </VARIABLE-ACCESS>
        </DATA-READ-ACCESSS>
        ...
      </RUNNABLE-ENTITY>
      </RUNNABLES>
    </SWC-INTERNAL-BEHAVIOR>
    </INTERNAL-BEHAVIORS>
  </APPLICATION-SW-COMPONENT-TYPE>
```

The two ApplicationSwComponentTypes are then used to type SwComponent-Prototypes in the Composition. The PortPrototypes of the SwComponent-Prototypes are connected by an AssemblySwConnector:

| Common CompositionSwComponentType attributes | |
|---|---|
| shortName | Composition |
| properties of the components | |
| properties of SwComponentPrototype | |
| shortName | CPT_Controller |
| type | Controller |
| properties of SwComponentPrototype | |
| shortName | CPT_Plant |
| type | Plant |
| [ ... ] | |

**Table 3.9: CompositionSwComponentType Composition**

In ARXML:

**Listing 3.15: Composision**

```xml
<COMPOSITION-SW-COMPONENT-TYPE>
  <SHORT-NAME>Composition</SHORT-NAME>
  <COMPONENTS>
  <SW-COMPONENT-PROTOTYPE>
    <SHORT-NAME>CPT_Controller</SHORT-NAME>
    <TYPE-TREF DEST="APPLICATION-SW-COMPONENT-TYPE">/McInt/SwComponents
       /Controller</TYPE-TREF>
  </SW-COMPONENT-PROTOTYPE>
    <SHORT-NAME>CPT_Plant</SHORT-NAME>
    <TYPE-TREF DEST="APPLICATION-SW-COMPONENT-TYPE">/McInt/SwComponents
       /Plant</TYPE-TREF>
  </SW-COMPONENT-PROTOTYPE>
  ...
  </COMPONENTS>
  <CONNECTORS>
  <ASSEMBLY-SW-CONNECTOR>
    <SHORT-NAME>
       ASC_CPT_Plant_TemperaturePPP_CPT_Controller_TemperatureRPP</
       SHORT-NAME>
    <PROVIDER-IREF>
    <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE">/McInt/
       SwComponents/Composition/CPT_Plant</CONTEXT-COMPONENT-REF>
    <TARGET-P-PORT-REF DEST="P-PORT-PROTOTYPE">/McInt/SwComponents/
       Plant/PlantTemperaturePPP</TARGET-P-PORT-REF>
    </PROVIDER-IREF>
    <REQUESTER-IREF>
    <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE">/McInt/
       SwComponents/Composition/CPT_Controller</CONTEXT-COMPONENT-REF>
    <TARGET-R-PORT-REF DEST="R-PORT-PROTOTYPE">/McInt/SwComponents/
       Controller/TemperatureRPP</TARGET-R-PORT-REF>
    </REQUESTER-IREF>
  </ASSEMBLY-SW-CONNECTOR>
  ...
  </CONNECTORS>
```

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

#### 3.1.6.3 System

In the ECU_Extract, i.e. a System with category ECU_EXTRACT, the Composition is used to type the rootSoftwareComposition. All SwComponentPrototypes in Composition are mapped to the single EcuInstance in this show case.

**Listing 3.16: System and EcuInstance**

```
<ECU-INSTANCE>
  <SHORT-NAME>EcuInstance</SHORT-NAME>
</ECU-INSTANCE>
<SYSTEM>
  <SHORT-NAME>EcuExtract</SHORT-NAME>
  <CATEGORY>ECU_EXTRACT</CATEGORY>
  <MAPPINGS>
  <SYSTEM-MAPPING>
    <SHORT-NAME>SystemMapping</SHORT-NAME>
    <SW-MAPPINGS>
    <SWC-TO-ECU-MAPPING>
      <SHORT-NAME>SwcToEcuMapping</SHORT-NAME>
      <COMPONENT-IREFS>
      <COMPONENT-IREF>
        <CONTEXT-COMPOSITION-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">
          /McInt/System/EcuExtract/RootSwCompositionPrototype
        </CONTEXT-COMPOSITION-REF>
        <TARGET-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE">
          /McInt/SwComponents/Composition/CPT_Controller
         </TARGET-COMPONENT-REF>
      </COMPONENT-IREF>
      <COMPONENT-IREF>
        <CONTEXT-COMPOSITION-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">
         /McInt/System/EcuExtract/RootSwCompositionPrototype
        </CONTEXT-COMPOSITION-REF>
        <TARGET-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE">
         /McInt/SwComponents/Composition/CPT_Plant
        </TARGET-COMPONENT-REF>
      </COMPONENT-IREF>
      ...
      </COMPONENT-IREFS>
      <ECU-INSTANCE-REF DEST="ECU-INSTANCE">/McInt/System/EcuInstance</
         ECU-INSTANCE-REF>
    </SWC-TO-ECU-MAPPING>
    </SW-MAPPINGS>
  </SYSTEM-MAPPING>
  </MAPPINGS>
  <ROOT-SOFTWARE-COMPOSITIONS>
  <ROOT-SW-COMPOSITION-PROTOTYPE>
    <SHORT-NAME>RootSwCompositionPrototype</SHORT-NAME>
    <FLAT-MAP-REF DEST="FLAT-MAP">/McInt/System/FlatMap</FLAT-MAP-REF>
    <SOFTWARE-COMPOSITION-TREF DEST="COMPOSITION-SW-COMPONENT-TYPE">
       /McInt/SwComponents/Composition
    </SOFTWARE-COMPOSITION-TREF>
```

```
    </ROOT-SW-COMPOSITION-PROTOTYPE>
  </ROOT-SOFTWARE-COMPOSITIONS>
</SYSTEM>
```

The FlatMap that is referenced in the ECU_Extract, gives the name TPlant to a dataElement (see ecuExtractReference below). The name TPlant is later on displayed in the MC Tool.

**Listing 3.17: FlatMap**

```
<FLAT-MAP>
  <SHORT-NAME>FlatMap</SHORT-NAME>
  <INSTANCES>
  <FLAT-INSTANCE-DESCRIPTOR>
    <SHORT-NAME>TPlant</SHORT-NAME>
    <ECU-EXTRACT-REFERENCE-IREF>
    <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/McInt/
       System/EcuExtract/RootSwCompositionPrototype</CONTEXT-ELEMENT-
       REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/McInt/
       SwComponents/Composition/CPT_Plant</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="P-PORT-PROTOTYPE">/McInt/SwComponents/
       Plant/PlantTemperaturePPP</CONTEXT-ELEMENT-REF>
    <TARGET-REF DEST="VARIABLE-DATA-PROTOTYPE">/McInt/PortInterfaces/
       TemperatureSRIF/T</TARGET-REF>
    </ECU-EXTRACT-REFERENCE-IREF>
  </FLAT-INSTANCE-DESCRIPTOR>
  ...
  </INSTANCES>
</FLAT-MAP>
```

#### 3.1.6.4 ECU Configuration

There are further things that need to be defined before the RTE and the OS can be generated. For instance, the order in which the RTEEvents for the RunnableEntitys are invoked and the assignment to an OsTask. This is done via EcucModuleConfigurationValues. The interesting parts of the RTE configuration are:

**Listing 3.18: RTE Config**

```
...
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>controller100ms</SHORT-NAME>
  <DEFINITION-REF ...>.../RteEventToTaskMapping</DEFINITION-REF>
  <PARAMETER-VALUES>
  <ECUC-NUMERICAL-PARAM-VALUE>
    <DEFINITION-REF ...>.../RtePositionInTask</DEFINITION-REF>
    <VALUE>3</VALUE>
  </ECUC-NUMERICAL-PARAM-VALUE>
  ...
  </PARAMETER-VALUES>
  <REFERENCE-VALUES>
  <ECUC-REFERENCE-VALUE>
```

```
    <DEFINITION-REF ...>.../RteMappedToTaskRef</DEFINITION-REF>
    <VALUE-REF ...>.../OS/OS_CFG/task_100ms</VALUE-REF>
  </ECUC-REFERENCE-VALUE>
  <ECUC-REFERENCE-VALUE>
    <DEFINITION-REF ...>.../RteEventRef</DEFINITION-REF>
    <VALUE-REF DEST="TIMING-EVENT">.../controller100ms</VALUE-REF>
  </ECUC-REFERENCE-VALUE>
  </REFERENCE-VALUES>
</ECUC-CONTAINER-VALUE>
...
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>plant100ms</SHORT-NAME>
  <DEFINITION-REF ...>.../RteEventToTaskMapping</DEFINITION-REF>
  <PARAMETER-VALUES>
  <ECUC-NUMERICAL-PARAM-VALUE>
    <DEFINITION-REF ...>.../RtePositionInTask</DEFINITION-REF>
    <VALUE>2</VALUE>
  </ECUC-NUMERICAL-PARAM-VALUE>
  ...
  </PARAMETER-VALUES>
  <REFERENCE-VALUES>
  <ECUC-REFERENCE-VALUE>
    <DEFINITION-REF ...>.../RteMappedToTaskRef</DEFINITION-REF>
    <VALUE-REF ...>.../OS/OS_CFG/task_100ms</VALUE-REF>
  </ECUC-REFERENCE-VALUE>
  <ECUC-REFERENCE-VALUE>
    <DEFINITION-REF ...>.../RteEventRef</DEFINITION-REF>
    <VALUE-REF DEST="TIMING-EVENT">.../plant100ms</VALUE-REF>
  </ECUC-REFERENCE-VALUE>
  </REFERENCE-VALUES>
</ECUC-CONTAINER-VALUE>
...
```

This part of the `OS` configuration defines the name of the `OSTask`, that we see later on in the generated C code:

**Listing 3.19: OsConfig**

```
...
<AR-PACKAGE>
  <SHORT-NAME>OS</SHORT-NAME>
  <ELEMENTS>
  <ECUC-MODULE-CONFIGURATION-VALUES>
    <SHORT-NAME>OS_CFG</SHORT-NAME>
    <DEFINITION-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/EcucDefs/Os</
       DEFINITION-REF>
    <CONTAINERS>
      ...
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>task_100ms</SHORT-NAME>
      <DEFINITION-REF ...>.../OsTask</DEFINITION-REF>
      <PARAMETER-VALUES>
      ...
    </ECUC-CONTAINER-VALUE>
    ...
```

These configurations are tied to the ECU_Extract by an EcucValueCollection:

**Listing 3.20: EcuC Value Collection**

```
<ECUC-VALUE-COLLECTION>
  <SHORT-NAME>EcucValueCollection</SHORT-NAME>
  <ECU-EXTRACT-REF DEST="SYSTEM">/McInt/System/EcuExtract</ECU-EXTRACT-
      REF>
  <ECUC-VALUES>
  <ECUC-MODULE-CONFIGURATION-VALUES-REF-CONDITIONAL>
    <ECUC-MODULE-CONFIGURATION-VALUES-REF DEST="ECUC-MODULE-
        CONFIGURATION-VALUES">/McInt/RTE/RTE_CFG</ECUC-MODULE-
        CONFIGURATION-VALUES-REF>
  </ECUC-MODULE-CONFIGURATION-VALUES-REF-CONDITIONAL>
  <ECUC-MODULE-CONFIGURATION-VALUES-REF-CONDITIONAL>
    <ECUC-MODULE-CONFIGURATION-VALUES-REF DEST="ECUC-MODULE-
        CONFIGURATION-VALUES">/McInt/OS/OS_CFG</ECUC-MODULE-
        CONFIGURATION-VALUES-REF>
  </ECUC-MODULE-CONFIGURATION-VALUES-REF-CONDITIONAL>
  </ECUC-VALUES>
</ECUC-VALUE-COLLECTION>
```

This completes the presentation of the AUTOSAR modeling in our walk through.

#### 3.1.6.5 RTE Generation

In the following, some snippets of the generated RTE are presented. However, they are examples only and may differ if different RTE generators are used.

Among other things, the OsTask is generated as defined in the ECU configuration above:

**Listing 3.21: Rte.c**

```
...
#define RTE_START_SEC_VAR
#include "MemMap.h" /*lint !e537 permit multiple inclusion */
...
VAR(float32, RTE_DATA) TPlant;
...
#define RTE_STOP_SEC_VAR
#include "MemMap.h" /*lint !e537 permit multiple inclusion */
...
TASK(task_100ms)
{
   ...
   Rte_ImplicitBufs.isa_1._task_100ms.sbuf1.value = TPlant;
   ...
   plantRE_func();
   ...
   controllerRE_func();
   ...
   TPlant = Rte_ImplicitBufs.isa_1._task_100ms.sbuf1.value;
   ...
} /* task_100ms */
```

```
...
```

Also a MACRO to write $T_{Plant}$ in the Plant

**Listing 3.22: Rte_Plant.h**

```
...
#define Rte_IRead_plantRE_EnvTemperatureRPP_T() ((CONST(float32,
    RTE_DATA)) Rte_ImplicitBufs.isa_1._task_100ms.sbuf0.value )
...
```

and to read $T_{Plant}$ in the Controller

**Listing 3.23: Rte_Controller.h**

```
...
#define Rte_IRead_ControllerRE_TemperatureRPP_T() ((CONST(float32,
    RTE_DATA)) Rte_ImplicitBufs.isa_1._task_100ms.sbuf1.value )
...
```

was generated. Furthermore, the McSupport file is generated as an interface between the "AUTOSAR world" and the "A2L world". As the reader can see, this is a compilation of necessary data from the AUTOSAR model presented before:

**Listing 3.24: McSupportData**

```
...
<AR-PACKAGE>
   <SHORT-NAME>BswImplementations</SHORT-NAME>
   <ELEMENTS>
    <BSW-IMPLEMENTATION>
     <SHORT-NAME>Rte</SHORT-NAME>
     <MC-SUPPORT>
      ...
      <MC-VARIABLE-INSTANCES>
        <MC-DATA-INSTANCE>
         <SHORT-NAME>TPlant</SHORT-NAME>
         <DESC>
           <L-2 L="EN">Type for a temperature in [°C]</L-2>
         </DESC>
         <CATEGORY>VALUE</CATEGORY>
         <FLAT-MAP-ENTRY-REF DEST="FLAT-INSTANCE-DESCRIPTOR">/McInt/
             System/FlatMap/TPlant</FLAT-MAP-ENTRY-REF>
         <RESULTING-PROPERTIES>
           <SW-DATA-DEF-PROPS-VARIANTS>
           <SW-DATA-DEF-PROPS-CONDITIONAL>
             <BASE-TYPE-REF BASE="Rte_MCSD_SwBaseTypes" DEST="SW-BASE-
                 TYPE">float32</BASE-TYPE-REF>
             <SW-CALIBRATION-ACCESS>READ-ONLY</SW-CALIBRATION-ACCESS>
             <COMPU-METHOD-REF BASE="Rte_MCSD_CompuMethods" DEST="COMPU
                 -METHOD">McInt_CompuMethods_Temperature_C</COMPU-
                 METHOD-REF>
             <DISPLAY-FORMAT>%.1f</DISPLAY-FORMAT>
             <UNIT-REF BASE="Rte_MCSD_Units" DEST="UNIT">
                 McInt_Units_DegreeCelsius</UNIT-REF>
           </SW-DATA-DEF-PROPS-CONDITIONAL>
           </SW-DATA-DEF-PROPS-VARIANTS>
```

```
          </RESULTING-PROPERTIES>
          <SYMBOL>TPlant</SYMBOL>
          </MC-DATA-INSTANCE>
         ...
       </MC-VARIABLE-INSTANCES>
    </ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
    <SHORT-NAME>Units</SHORT-NAME>
    <ELEMENTS>
     <UNIT>
     <SHORT-NAME>McInt_Units_DegreeCelsius</SHORT-NAME>
     <DISPLAY-NAME>°C</DISPLAY-NAME>
     <FACTOR-SI-TO-UNIT>1.0</FACTOR-SI-TO-UNIT>
     <OFFSET-SI-TO-UNIT>-273.15</OFFSET-SI-TO-UNIT>
     <PHYSICAL-DIMENSION-REF BASE="Rte_MCSD_PhysicalDimensions" DEST="
         PHYSICAL-DIMENSION">McInt_PhysicalDimensions_Temparature</
         PHYSICAL-DIMENSION-REF>
     </UNIT>
     ...
    </ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
    <SHORT-NAME>CompuMethods</SHORT-NAME>
    <ELEMENTS>
     <COMPU-METHOD>
     <SHORT-NAME>McInt_CompuMethods_Temperature_C</SHORT-NAME>
     <DESC>
       <L-2 L="EN">Conversion from [°C] at an interface to [K] for
           internal computations</L-2>
     </DESC>
     <CATEGORY>LINEAR</CATEGORY>
     <DISPLAY-FORMAT>%f</DISPLAY-FORMAT>
     <UNIT-REF BASE="Rte_MCSD_Units" DEST="UNIT">
         McInt_Units_DegreeCelsius</UNIT-REF>
     <COMPU-INTERNAL-TO-PHYS>
       <COMPU-SCALES>
       <COMPU-SCALE>
         <COMPU-RATIONAL-COEFFS>
         <COMPU-NUMERATOR>
           <V>-273.15</V>
           <V>1</V>
         </COMPU-NUMERATOR>
         <COMPU-DENOMINATOR>
           <V>1</V>
         </COMPU-DENOMINATOR>
         </COMPU-RATIONAL-COEFFS>
       </COMPU-SCALE>
       </COMPU-SCALES>
     </COMPU-INTERNAL-TO-PHYS>
     </COMPU-METHOD>
     ...
    </ELEMENTS>
</AR-PACKAGE>
<AR-PACKAGE>
    <SHORT-NAME>PhysicalDimensions</SHORT-NAME>
```

```
      <ELEMENTS>
       <PHYSICAL-DIMENSION>
       <SHORT-NAME>McInt_PhysicalDimensions_Temparature</SHORT-NAME>
       <LENGTH-EXP>0</LENGTH-EXP>
       <MASS-EXP>0</MASS-EXP>
       <TIME-EXP>0</TIME-EXP>
       <CURRENT-EXP>0</CURRENT-EXP>
       <TEMPERATURE-EXP>1</TEMPERATURE-EXP>
       <MOLAR-AMOUNT-EXP>0</MOLAR-AMOUNT-EXP>
       <LUMINOUS-INTENSITY-EXP>0</LUMINOUS-INTENSITY-EXP>
       </PHYSICAL-DIMENSION>
       ...
   </AR-PACKAGE>
   <AR-PACKAGE>
      <SHORT-NAME>SwBaseTypes</SHORT-NAME>
      <ELEMENTS>
      <SW-BASE-TYPE>
      <SHORT-NAME>float32</SHORT-NAME>
      <CATEGORY>FIXED_LENGTH</CATEGORY>
      <BASE-TYPE-SIZE>32</BASE-TYPE-SIZE>
      <BASE-TYPE-ENCODING>IEEE754</BASE-TYPE-ENCODING>
      </SW-BASE-TYPE>
      ...
      </ELEMENTS>
   </AR-PACKAGE>
   ...
```

#### 3.1.6.6 Implementation in C

The implementation in C-Code is a direct implementation of the physical equations. The `Plant` uses the `MACRO`, generated by the `RTE` generator, to write $T_{Plant}$:

**Listing 3.25: Plant**

```c
#include "Rte_Plant.h"
...
FUNC (void, Plant_CODE) plantRE_func (void)
{
   ...
   /* heat capacity of 1 assumed                                 */
   float32 lTPlant     = lQPlant;
    /* calculate heat flows, store in PIM to make them measurable */
   *Rte_Pim_QEnv()     = (lTenv - lTPlant) * lEFactor * lDt;
   ...
   /* heat capacity of 1 assumed                                 */
   lTPlant   = lQPlant;
   ...
   /* Write output of plant: temerature of plant                 */
   Rte_IWrite_plantRE_PlantTemperaturePPP_T(lTPlant);
}
```

The `Controller` uses the `MACRO`, generated by the `RTE` generator, to read $T_{Plant}$:

**Listing 3.26: Controller**

```
#include "Rte_Controller.h"
...
FUNC (void, Controller_CODE) controllerRE_func (void)
{
    /* read input, define output variable                          */
    float32 lT         = Rte_IRead_ControllerRE_TemperatureRPP_T();
    ...

     /* store current error in PIM to make it measurable           */
    *Rte_Pim_E() = lSetPoint - lT;
    ...
}
```

#### 3.1.6.7 A2L File

Using the `McSupport` file and the map file from the linker, an example `A2L` file was generated for this show case. The snippet below is an example only and could differ if a different `A2L` file generator is used:

**Listing 3.27: A2L File**

```
...
/begin MEASUREMENT TPlant
       "TPlant"
       FLOAT32_IEEE
       McInt_CompuMethods_Temperature_C
       0
       0
       -1E+32
       1E+32
       DISPLAY_IDENTIFIER  "TPlant"
       ECU_ADDRESS 0xe000001c
       FORMAT  "%.1f"
       PHYS_UNIT "°C"
/end MEASUREMENT
...
/begin UNIT McInt_PhysicalDimensions_Temparature
       "McInt_PhysicalDimensions_Temparature"
       "McInt_PhysicalDimensions_Temparature"
       EXTENDED_SI
       SI_EXPONENTS 0 0 0 0 1 0 0
/end UNIT
/begin UNIT McInt_Units_DegreeCelsius
       "McInt_Units_DegreeCelsius"
       "°C"
       DERIVED
       REF_UNIT McInt_PhysicalDimensions_Temparature
       UNIT_CONVERSION 1 -273.15
/end UNIT
/begin COMPU_METHOD McInt_CompuMethods_Temperature_C
       "McInt_CompuMethods_Temperature_C"
       LINEAR
       "%f"
       "°C"
```

```
        COEFFS_LINEAR 1 -273.15
        REF_UNIT McInt_Units_DegreeCelsius
/end COMPU_METHOD
...
```

#### 3.1.6.8 Measurement and Calibration Tool

The `A2L` file is then used by a `MC` tool to measure $T_{Plant}$. Of course, in addition to the `A2L` file a suitable `ECU` access[4] must be available, to actually do measurement and calibration with the AUTOSAR system of this show case. However, the `ECU` access is not presented because this is not in the focus of this show case.

Below is a typical screen shot from a `MC` tool during an actual measurement and calibration task. You can see $T_{Plant}$ measured and displayed in degree Celsius.

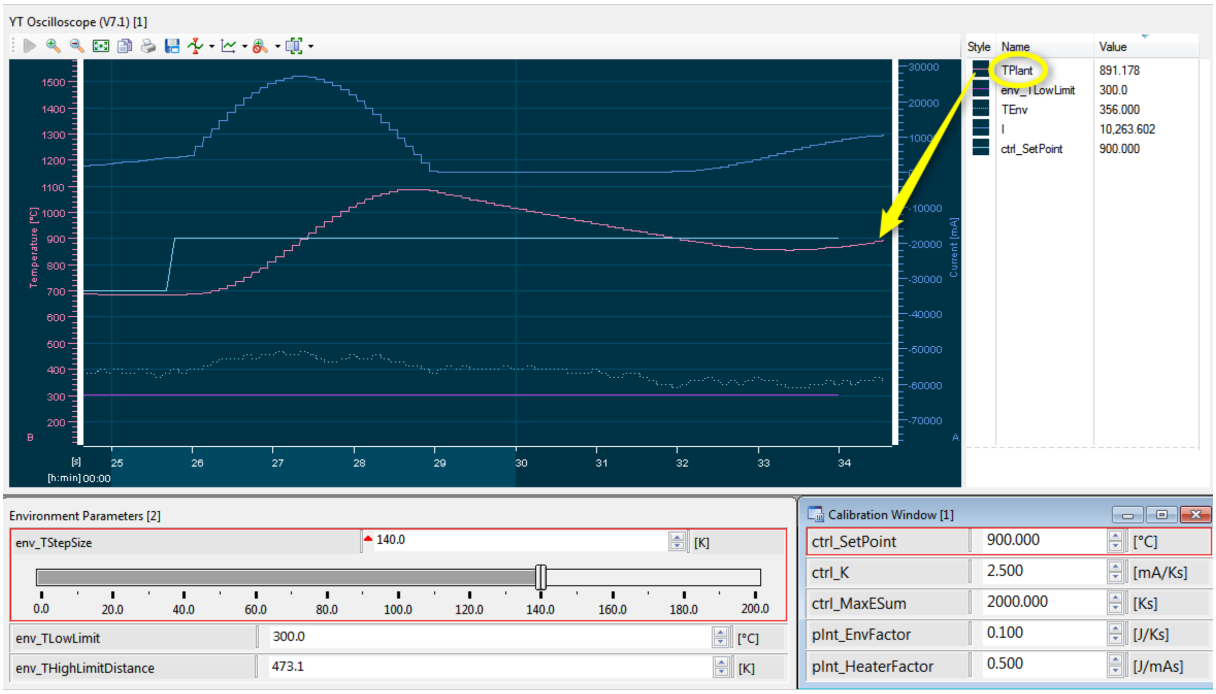


**Figure 3.12: Screenshot of a MC Tool**

[4] for instance, a measurement and calibration service like `XCP` or a hardware access to the memory of the micro controller

### 3.1.7 Show cases in the Example

#### 3.1.7.1 CompositionSwComponentTypes

| | |
|---|---|
| Common CompositionSwComponentType attributes | |
| shortName | Composition |
| properties of the components | |
| properties of SwComponentPrototype | |
| shortName | CPT_Controller |
| type | Controller |
| properties of SwComponentPrototype | |
| shortName | CPT_Parameters |
| type | Parameters |
| properties of SwComponentPrototype | |
| shortName | CPT_Plant |
| type | Plant |
| properties of SwComponentPrototype | |
| shortName | CPT_Environment |
| type | Environment |

**Table 3.10: CompositionSwComponentType Composition**

#### 3.1.7.2 ParameterSwComponentTypes

| Common ParameterSwComponentType attributes | |
|---|---|
| shortName | Parameters |
| desc | Type for providing the parameters to the ApplicationSwCompoments |
| **properties of the ports** | |
| **properties of PPortPrototype** | |
| shortName | ControllerPPP |
| desc | Port for providing the parameters for the controller |
| providedInterface | ControllerPIF |
| **properties of PPortPrototype** | |
| shortName | PlantPPP |
| desc | Port for providing the parameters for the plant |
| providedInterface | PlantPIF |
| **properties of PPortPrototype** | |
| shortName | EnvironmentPPP |
| desc | Port for providing the parameters for the environment |
| providedInterface | EnvironmentPIF |
| **properties of PPortPrototype** | |
| shortName | DtPPP |
| desc | Time of one time step |
| providedInterface | DtPIF |

**Table 3.11: ParameterSwComponentType Parameters**

#### 3.1.7.3 ApplicationSwComponentTypes

| Common ApplicationSwComponentType attributes | |
|---|---|
| shortName | Controller |
| **properties of the ports** | |
| **properties of RPortPrototype** | |
| shortName | TemperatureRPP |
| desc | Port to receive the temperature of the plant |
| requiredInterface | TemperatureSRIF |
| **properties of PPortPrototype** | |
| shortName | CurrentPPP |
| desc | Port for sending out the current output by this controller |
| providedInterface | CurrentSRIF |
| **properties of RPortPrototype** | |
| shortName | ControllerParamsRPP |
| desc | Port to get the parameters for the controller |
| requiredInterface | ControllerPIF |
| **properties of RPortPrototype** | |
| shortName | DtRPP |
| desc | Port to get delta t, i.e. time of one time step |
| requiredInterface | DtPIF |
| internalBehavior | ControllerInternalBehavior |

**Table 3.12: ApplicationSwComponentType Controller**

| Common SwcInternalBehavior attributes | |
|---|---|
| shortName | ControllerInternalBehavior |
| **properties of implicitInterRunnableVariables / explicitInterRunnableVariables** | |
| **properties of VariableDataPrototype** | |
| shortName | ESum |
| desc | Internal state of the controller: the sum of control errors |
| type | ESum |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | CODE |
| **properties of the arTypedPerInstaceMemorys** | |
| **properties of VariableDataPrototype** | |
| shortName | E |
| desc | Measurement point for the control error, the deviation between set point and acutal temperature of the plant, in the current time step |
| type | Temperature_K |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | CODE |
| **properties of the runnables** | |
| **properties of RunnableEntity** | |
| shortName | ControllerRE |
| symbol | controllerRE_func |
| **properties of the events** | |
| **properties of TimingEvent** | |
| shortName | controller100ms |
| startOnEvent | ControllerRE |
| period | 0.1 |

**Table 3.13: SwcInternalBehavior ControllerInternalBehavior**

| Common ApplicationSwComponentType attributes | |
|---|---|
| shortName | Plant |
| **properties of the ports** | |
| **properties of RPortPrototype** | |
| shortName | CurrentRPP |
| desc | Port to receive the current from the controller |
| requiredInterface | CurrentSRIF |
| **properties of PPortPrototype** | |
| shortName | PlantTemperaturePPP |
| desc | Port for sending out the estimated temperature of the plant |
| providedInterface | TemperatureSRIF |
| **properties of RPortPrototype** | |
| shortName | PlantParamsRPP |
| desc | Port to get the parameters for the plant |
| requiredInterface | PlantPIF |
| **properties of RPortPrototype** | |
| shortName | EnvTemperatureRPP |
| desc | Port to receive the tempertature of the environment |
| requiredInterface | TemperatureSRIF |
| **properties of RPortPrototype** | |
| shortName | DtRPP |
| desc | Port to get delta t, i.e. time of one time step |
| requiredInterface | DtPIF |
| internalBehavior | PlantInternalBehavior |

**Table 3.14: ApplicationSwComponentType Plant**

| Common SwcInternalBehavior attributes | |
|---|---|
| shortName | PlantInternalBehavior |
| properties of implicitInterRunnableVariables / explicitInterRunnableVariables | |
| properties of VariableDataPrototype | |
| shortName | QPlant |
| desc | Internal state of the plant: the stored eneregy quantity in the current time step |
| type | Energy |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | CODE |
| properties of the arTypedPerInstaceMemorys | |
| properties of VariableDataPrototype | |
| shortName | QHeater |
| desc | Measurement point for heat flow between the electrical heater and the plant in the current time step. |
| type | Energy |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | VAR |
| properties of VariableDataPrototype | |
| shortName | QEnv |
| desc | Measurement point for heat flow between the plant and the environment in the current time step. |
| type | Energy |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | CODE |
| properties of the runnables | |
| properties of RunnableEntity | |
| shortName | plantRE |
| symbol | plantRE_func |
| properties of the events | |
| properties of TimingEvent | |
| shortName | plant100ms |
| startOnEvent | plantRE |
| period | 0.1 |

**Table 3.15: SwcInternalBehavior PlantInternalBehavior**

| Common ApplicationSwComponentType attributes | |
|---|---|
| shortName | Environment |
| properties of the ports | |
| properties of PPortPrototype | |
| shortName | EnvTemperaturePPP |
| desc | Port to send out the temperature of the environment |
| providedInterface | TemperatureSRIF |
| properties of RPortPrototype | |
| shortName | EnvParamsRPP |
| desc | Port to get the parameters for the environment |
| requiredInterface | EnvironmentPIF |
| properties of RPortPrototype | |
| shortName | DtRPP |
| desc | Port to get delta t, i.e. time of one time step |
| requiredInterface | DtPIF |
| internalBehavior | EnvironmentInternalBehavior |

**Table 3.16: ApplicationSwComponentType Environment**

| Common SwcInternalBehavior attributes | |
|---|---|
| shortName | EnvironmentInternalBehavior |
| properties of implicitInterRunnableVariables / explicitInterRunnableVariables | |
| properties of VariableDataPrototype | |
| shortName | Seed |
| desc | Internal state of the environment: the current seed for the (pseudo) random number generation |
| type | uint32 |
| swImplPolicy | standard |
| swCalibrationAccess | notAccessible |
| swAddrMethod | CODE |
| properties of VariableDataPrototype | |
| shortName | TEnv |
| desc | Internal state of the environment: the temperture of the environment |
| type | Temperature_C |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | CODE |
| properties of the arTypedPerInstaceMemorys | |
| properties of VariableDataPrototype | |
| shortName | THighLimit |
| desc | Measurement point for the upper limit of the generated temperature profile |
| type | Temperature_C |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | CODE |
| properties of the runnables | |
| properties of RunnableEntity | |
| shortName | envRE |
| symbol | envRE_func |
| properties of the events | |
| properties of TimingEvent | |
| shortName | env100ms |
| startOnEvent | envRE |
| period | 0.1 |

**Table 3.17: SwcInternalBehavior EnvironmentInternalBehavior**

#### 3.1.7.4 ParameterInterfaces

| Common ParameterInterface attributes | |
|---|---|
| shortName | ControllerPIF |
| desc | Interface with all parameters for the controller |
| **properties of the parameters** | |
| **properties of ParameterDataPrototype** | |
| shortName | ctrl_SetPoint |
| desc | Set point for the temperature of the plant |
| type | Temperature_C |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CALIB |
| **properties of ParameterDataPrototype** | |
| shortName | ctrl_K |
| desc | Amplification factor for the I-controller |
| type | Amplification |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CALIB |
| **properties of ParameterDataPrototype** | |
| shortName | ctrl_MaxESum |
| desc | Upper limit of the integal part of the I-controller |
| type | ESum |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CALIB |

**Table 3.18: ParameterInterface ControllerPIF**

| Common ParameterInterface attributes | |
|---|---|
| shortName | PlantPIF |
| desc | Interface with all parameters for the plant |
| **properties of the parameters** | |
| **properties of ParameterDataPrototype** | |
| shortName | plnt_EnvFactor |
| desc | Proportionality factor for the heat flow between plant and environment |
| type | EnvFactor |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CALIB |
| **properties of ParameterDataPrototype** | |
| shortName | plnt_HeaterFactor |
| desc | Proportionality factor for the heat flow between plant and the electrical heater |
| type | HeaterFactor |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CALIB |

**Table 3.19: ParameterInterface PlantPIF**

| Common ParameterInterface attributes | |
|---|---|
| shortName | EnvironmentPIF |
| desc | Interface with all parameters for the environment |
| **properties of the parameters** | |
| **properties of ParameterDataPrototype** | |
| shortName | env_TLowLimit |
| desc | Lower limit of the generated temeprature profile |
| type | Temperature_C |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CALIB |
| **properties of ParameterDataPrototype** | |
| shortName | env_TStepSize |
| desc | The maximal temperature diffenrence of the environment in one time step |
| type | Temperature_K |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CALIB |
| **properties of ParameterDataPrototype** | |
| shortName | env_THighLimitDistance |
| desc | Distance of the upper limit from the lower limit for the generated temeprature profile. |
| type | Temperature_K |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CALIB |

**Table 3.20: ParameterInterface EnvironmentPIF**

| Common ParameterInterface attributes | |
|---|---|
| shortName | DtPIF |
| **properties of the parameters** | |
| **properties of ParameterDataPrototype** | |
| shortName | Dt |
| desc | Scheduling time of the components |
| type | Time |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CALIB |

**Table 3.21: ParameterInterface DtPIF**

#### 3.1.7.5 SenderReceiverInterfaces

| Common SenderReceiverInterface attributes | |
|---|---|
| shortName | TemperatureSRIF |
| desc | Interface type for transferring temperatures in [°C] |
| properties of the dataElementss | |
| properties of VariableDataPrototype | |
| shortName | T |
| type | Temperature_C |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | VAR |

**Table 3.22: SenderReceiverInterface TemperatureSRIF**

| Common SenderReceiverInterface attributes | |
|---|---|
| shortName | CurrentSRIF |
| desc | Interface type for transferring a current in [mA] |
| properties of the dataElementss | |
| properties of VariableDataPrototype | |
| shortName | I |
| type | Current |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | VAR |

**Table 3.23: SenderReceiverInterface CurrentSRIF**

#### 3.1.7.6 ApplicationDataTypes, Category VALUE

| Common ApplicationDataType attributes | | | |
|---|---|---|---|
| shortName | Temperature_C | | |
| category | VALUE | | |
| desc | Type for a temperature in [°C] | | |
| swCalibrationAccess | readOnly | | |
| unit | DegreeCelsius | | |
| Range | | | |
| Conversion | | | |
| category | LINEAR | | |
| direction | compuInternalToPhys | | |
| desc | lowerLimit | upperLimit | compuNumerator / compuDenominator |
| - | - | - | $Phys = \frac{-273.15 + 1 * Internal}{1}$ |

Table 3.24: ApplicationDataType Temperature_C

| Common ApplicationDataType attributes | | | |
|---|---|---|---|
| shortName | Current | | |
| category | VALUE | | |
| desc | Type for the current in [mA] | | |
| swCalibrationAccess | readOnly | | |
| unit | MilliAmpere | | |
| Range | | | |
| Conversion | | | |
| category | LINEAR | | |
| direction | compuInternalToPhys | | |
| desc | lowerLimit | upperLimit | compuNumerator / compuDenominator |
| - | - | - | $Phys = \frac{0 + 1000 * Internal}{1}$ |

Table 3.25: ApplicationDataType Current

| Common ApplicationDataType attributes | |
|---|---|
| shortName | EnvFactor |
| category | VALUE |
| desc | Type for the environt factor in [J/Ks] |
| swCalibrationAccess | readOnly |
| unit | JoulePerKelvinSecond |
| Range | |
| Conversion | |
| category | IDENTICAL |
| direction | - |

**Table 3.26: ApplicationDataType EnvFactor**

| Common ApplicationDataType attributes | |
|---|---|
| shortName | Temperature_K |
| category | VALUE |
| desc | Type for a temperature in [K] |
| swCalibrationAccess | readOnly |
| unit | Kelvin |
| Range | |
| Conversion | |
| category | IDENTICAL |
| direction | - |

**Table 3.27: ApplicationDataType Temperature_K**

| Common ApplicationDataType attributes | |
|---|---|
| shortName | Amplification |
| category | VALUE |
| desc | Type for an amplification factor in a controller in [mA/Ks] |
| swCalibrationAccess | readOnly |
| unit | MilliAmperePerKelvinSecond |
| Range | |
| Conversion | |
| category | IDENTICAL |
| direction | - |

**Table 3.28: ApplicationDataType Amplification**

| Common ApplicationDataType attributes | |
|---|---|
| shortName | Energy |
| category | VALUE |
| desc | Type for energy [J] |
| swCalibrationAccess | readOnly |
| unit | Joule |
| Range | |
| Conversion | |
| category | IDENTICAL |
| direction | - |

**Table 3.29: ApplicationDataType Energy**

| Common ApplicationDataType attributes | |
|---|---|
| shortName | ESum |
| category | VALUE |
| desc | Type for the sum of control errors of an I controller in [Ks] |
| swCalibrationAccess | readOnly |
| unit | KelvinSecond |
| Range | |
| Conversion | |
| category | IDENTICAL |
| direction | - |

**Table 3.30: ApplicationDataType ESum**

| Common ApplicationDataType attributes | |
|---|---|
| shortName | HeaterFactor |
| category | VALUE |
| desc | Type of a proportionality factor for the heat flow from an electrical heater to a thermal energy storage in [J/mAs] |
| swCalibrationAccess | readOnly |
| unit | JoulePerMilliAmpereSecond |
| Range | |
| Conversion | |
| category | IDENTICAL |
| direction | - |

**Table 3.31: ApplicationDataType HeaterFactor**

| Common ApplicationDataType attributes | |
|---|---|
| shortName | Time |
| category | VALUE |
| desc | Type for time in [s] |
| swCalibrationAccess | readOnly |
| unit | Second |
| Range | |
| Conversion | |
| category | IDENTICAL |
| direction | - |

**Table 3.32: ApplicationDataType Time**

#### 3.1.7.7 Units

| Common Unit attributes | |
|---|---|
| shortName | DegreeCelsius |
| displayName | °C |
| offsetSiToUnit | -273.15 |
| factorSiToUnit | 1.0 |
| physicalDimension | Temperature |

**Table 3.33: Unit DegreeCelsius**

| Common Unit attributes | |
|---|---|
| shortName | Kelvin |
| displayName | K |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 1.0 |
| physicalDimension | Temperature |

**Table 3.34: Unit Kelvin**

| Common Unit attributes | |
|---|---|
| shortName | Joule |
| displayName | J |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 1.0 |
| physicalDimension | Energy |

**Table 3.35: Unit Joule**

| Common Unit attributes | |
|---|---|
| shortName | MilliAmpere |
| displayName | mA |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 1000.0 |
| physicalDimension | Current |

**Table 3.36: Unit MilliAmpere**

| Common Unit attributes | |
|---|---|
| shortName | KelvinSecond |
| displayName | Ks |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 1.0 |
| physicalDimension | TemperatureTime |

**Table 3.37: Unit KelvinSecond**

| Common Unit attributes | |
|---|---|
| shortName | JoulePerKelvinSecond |
| displayName | J/Ks |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 1.0 |
| physicalDimension | EnergyPerTemperatureTime |

**Table 3.38: Unit JoulePerKelvinSecond**

| Common Unit attributes | |
|---|---|
| shortName | JoulePerMilliAmpereSecond |
| displayName | J/mAs |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 0.001 |
| physicalDimension | EnergyPerCurrentTime |

**Table 3.39: Unit JoulePerMilliAmpereSecond**

| Common Unit attributes | |
|---|---|
| shortName | MilliAmperePerKelvinSecond |
| displayName | mA/Ks |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 1000.0 |
| physicalDimension | CurrentPerTemperatureTime |

**Table 3.40: Unit MilliAmperePerKelvinSecond**

| Common Unit attributes | |
|---|---|
| shortName | Second |
| displayName | s |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 1.0 |
| physicalDimension | Time |

**Table 3.41: Unit Second**

#### 3.1.7.8 PhysicalDimensions

| Common PhysicalDimension attributes | |
|---|---|
| shortName | Energy |
| currentExp | 0 |
| lengthExp | 2 |
| luminousIntensity-Exp | 0 |
| massExp | 1 |
| molarAmountExp | 0 |
| temperatureExp | 0 |
| timeExp | -2 |

**Table 3.42: PhysicalDimension Energy**

| Common PhysicalDimension attributes | |
|---|---|
| shortName | Current |
| currentExp | 1 |
| lengthExp | 0 |
| luminousIntensity-Exp | 0 |
| massExp | 0 |
| molarAmountExp | 0 |
| temperatureExp | 0 |
| timeExp | 0 |

**Table 3.43: PhysicalDimension Current**

| Common PhysicalDimension attributes | |
|---|---|
| shortName | CurrentPerTemperatureTime |
| currentExp | 1 |
| lengthExp | 0 |
| luminousIntensity-Exp | 0 |
| massExp | 0 |
| molarAmountExp | 0 |
| temperatureExp | -1 |
| timeExp | -1 |

**Table 3.44: PhysicalDimension CurrentPerTemperatureTime**

| Common PhysicalDimension attributes | |
|---|---|
| shortName | EnergyPerCurrentTime |
| currentExp | -1 |
| lengthExp | 2 |
| luminousIntensity-Exp | 0 |
| massExp | 1 |
| molarAmountExp | 0 |
| temperatureExp | 0 |
| timeExp | -3 |

**Table 3.45: PhysicalDimension EnergyPerCurrentTime**

| Common PhysicalDimension attributes | |
|---|---|
| shortName | EnergyPerTemperatureTime |
| currentExp | 0 |
| lengthExp | 2 |
| luminousIntensity-Exp | 0 |
| massExp | 1 |
| molarAmountExp | 0 |
| temperatureExp | -1 |
| timeExp | -3 |

**Table 3.46: PhysicalDimension EnergyPerTemperatureTime**

| Common PhysicalDimension attributes | |
|---|---|
| shortName | Time |
| currentExp | 0 |
| lengthExp | 0 |
| luminousIntensity-Exp | 0 |
| massExp | 0 |
| molarAmountExp | 0 |
| temperatureExp | 0 |
| timeExp | 1 |

**Table 3.47: PhysicalDimension Time**

| Common PhysicalDimension attributes | |
|---|---|
| shortName | Temperature |
| currentExp | 0 |
| lengthExp | 0 |
| luminousIntensity-Exp | 0 |
| massExp | 0 |
| molarAmountExp | 0 |
| temperatureExp | 1 |
| timeExp | 0 |

**Table 3.48: PhysicalDimension Temperature**

| Common PhysicalDimension attributes | |
|---|---|
| shortName | TemperatureTime |
| currentExp | 0 |
| lengthExp | 0 |
| luminousIntensity-Exp | 0 |
| massExp | 0 |
| molarAmountExp | 0 |
| temperatureExp | 1 |
| timeExp | 1 |

**Table 3.49: PhysicalDimension TemperatureTime**

#### 3.1.7.9 SwAddrMethods

| Common SwAddrMethod attributes | | |
|---|---|---|
| shortName | | VAR |
| desc | | Memory section for variables |
| | sectionType | var |
| | memoryAllocation-KeywordPolicy | addrMethodShortName |
| | sectionInitializa-tionPolicy | - |
| | option | safetyQM |

**Table 3.50: SwAddrMethod VAR**

| Common SwAddrMethod attributes | | |
|---|---|---|
| shortName | | CALIB |
| desc | | Memory section for calibration parameters |
| | sectionType | var |
| | memoryAllocation-KeywordPolicy | addrMethodShortName |
| | sectionInitializa-tionPolicy | - |
| | option | safetyQM |

**Table 3.51: SwAddrMethod CALIB**

| Common SwAddrMethod attributes | | |
|---|---|---|
| shortName | | CODE |
| desc | | Memory section for code |
| | sectionType | var |
| | memoryAllocation-KeywordPolicy | addrMethodShortName |
| | sectionInitializa-tionPolicy | - |
| | option | safetyQM |

**Table 3.52: SwAddrMethod CODE**

## 3.2 Advanced Show Case

### 3.2.1 General Objectives of the Model Structure

#### 3.2.1.1 The Ecu Description

Since the show case is focusing on measurement and calibration only a minimal system model is provided. Hereby the file Pprj_EcuDescr_U_SystemNodeStub.arxml defines the System SystemU_EcuDescr of category ECU_SYSTEM_DESCRIPTION which contains only the RootSwCompositionPrototype. The file Pprj_EcuDescr_U.arxml contains the according CompositionSwComponentType describing the hierarchical top-level-composition of software components shown in table SystemURootComposition_EcuDescr.

#### 3.2.1.2 The Ecu Extract

The file Pprj_EcuExtract_U_SystemNodeStub.arxml defines the System SystemU_System of category ECU_EXTRACT which contains only the RootSwCompositionPrototype SystemU referencing the ECU Flat Map and the flat top-level-composition SystemU_Root. The file Pprj_EcuExtract_U.arxml contains the according CompositionSwComponentType describing the flat top-level-composition of software components shown in table SystemU_Root.

Please note that the flat top-level-composition uses the identical software component types as the hierarchical top-level-composition. Therefore an identification of component and data instances in the hierarchical software component structure or in the flat structure requires the correct iteration from the according System nodes.

##### 3.2.1.2.1 The ECU Flat Map

The file Pprj_EcuExtract_U_FlatMap.arxml contains the ECU Flat Map.

The ECU Flat Map is utilized to assign unique and comprehensible names to all DataPrototypes representing measurements and characteristics. This is important for the calibration engineers[5]

The applied strategy for the creation of a FlatInstanceDescriptor.shortName is to shorten it to the shortName of the DataPrototype when only a single instance of the DataPrototype is used.

[5] Calibration engineers in this context means the engineers working with measurement and calibration tooling e.g. to determine the correct calibration parameter values in order to adopt functionality in the software components to the mechanical components in the vehicle.

#### 3.2.1.3 Data Types and Data Objects

The components are designed top down coming from the physical function down to the implementation in the target programming language C. Hereby the interfaces of Software Components are typically typed with ApplicationDataTypes in order to describe the physical meaning of the DataPrototypes. The only exceptions are the interfaces to AUTOSAR Services which are typed by ImplementationDataTypes directly as those are standardized. ApplicationPrimitiveDataTypes are mainly of category

- BOOLEAN
- VALUE
- CURVE
- MAP
- COM_AXIS

and the most important CompuMethod categorys are

- LINEAR
- TEXTTABLE

In case of LINEAR conversions it is supported to differentiate the Unit used for the implemented calculations and an additional Unit used in the MCD system. This relationship of such Units are expressed with UnitGroups. The ARElements are structured in a way to support the common usage of elements relevant for the interface description up to the level of PortInterfaces by several Component Descriptions. Those elements are located under Tier1/ARPlatform1/DataDictionary/<KindPackage> in the file Pprj_DataDictionary.arxml.

The CompuMethods and DataConstrs are exclusively used by one ApplicationPrimitiveDataType. The possible reuse between ApplicationPrimitiveDataTypes supported by AUTOSAR is not used in this model structure. When such a ApplicationDataType is defined the intended mapping to the reasonable ImplementationDataType is already considered in order to get an optimal usage of the possible range of the ImplementationDataType. Nevertheless, the several physical meanings are **not** reflected by the definition of individual ImplementationDataType but only the standardized Platform Types [4] are used to describe primitives on implementation level. This has the effect that the RTE APIs are typed by the standardized Platform Types in cases of primitives and arrays of primitives. Only structure types are getting observable in the types of RTE APIs. This approach allows the direct usage of data read from RTE in mathematical or interpolation libraries without any type cast.

The memory allocation of the data objects is controlled by the usage of SwAddrMethods. Those are defined for ParameterDataPrototypes and VariableDataPrototypes on level of the PortInterfaces. A few examples are shown in the chapter 3.2.2.17 for the basic uses cases like calibration parameter, normal data and code.

### 3.2.1.4 Axis, Curves and Maps

The show case contains description for axis, curves and maps which are in AUTOSAR so called compound primitives. In order to understand the structure and the defined attributes in the example it is helpful to understand how such objects are described in AUTOSAR. For this it is necessary to look at the hierarchy of `ApplicationDataType`s, `DataPrototype`s, `PortPrototype`s, `SwComponentType`s and `FlatMap`.

### 3.2.1.5 Axis, Curves and Maps on ApplicationDataType level

Figure 3.13 is based on the example of the `ApplicationPrimitiveDataType` `Map_Time_Lnr_s_uint16`. It shows the relationships between the `ApplicationPrimitiveDataType`s describing the

- MAP itself
- its axis being a group axis
- in turn the properties of a matching working point

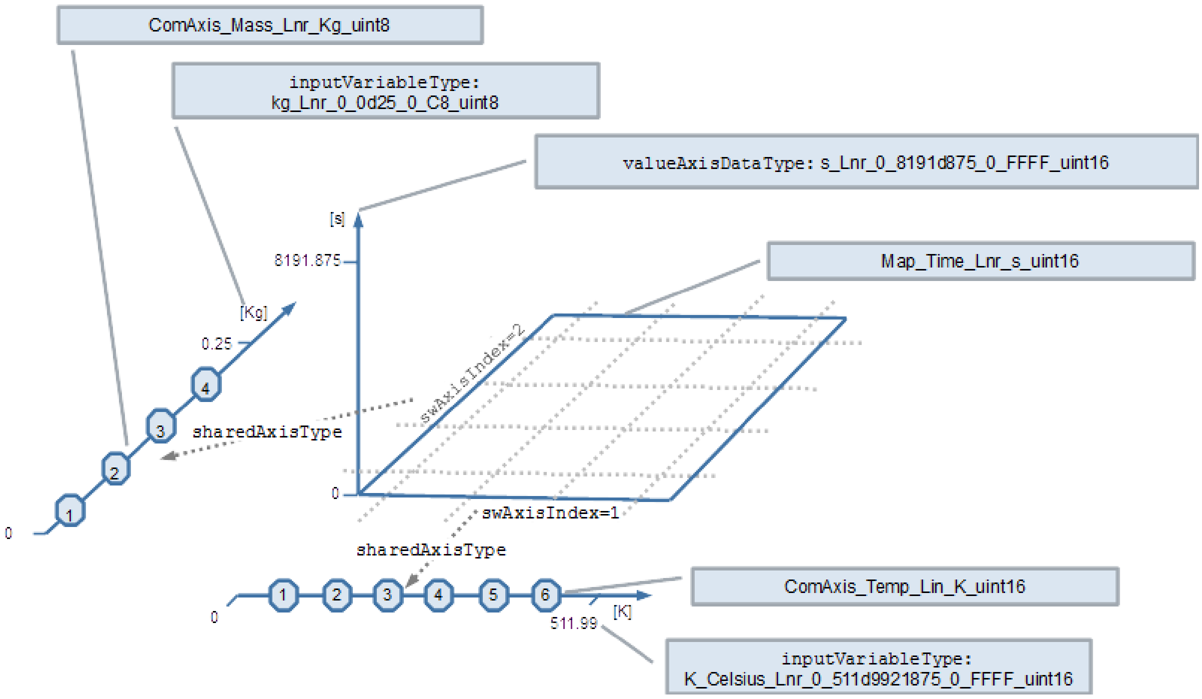


**Figure 3.13: ApplicationPrimitiveDataType of category MAP and its group axes**

The ApplicationPrimitiveDataType Map_Time_Lnr_s_uint16 defines a data type for a MAP with group axes. The physical meaning and range of the contained values is described with the ApplicationPrimitiveDataType s_Lnr_0_8191d875_0_FFFF_uint16. It is referenced with the valueAxisDataType attribute. This means it's a value in the range 0 .. 8191.875 [second] with the resolution of 0.125 [second].

The referenced ApplicationPrimitiveDataType in the role valueAxisDataType represents the primitive data type of the value axis within a compound primitive (e.g. CURVE, MAP). It supersedes CompuMethod, Unit, and BaseType. In the particular example, the valueAxisDataType provides the properties of the primitive elements of the CURVE or MAP via a valueAxisDataType reference to an ApplicationPrimitiveDataType. This in turn defines the attributes:

- dataConstr
- compuMethod
- displayFormat
- unit
- swCalibrationAccess

Thereby, despite being set, the value of swCalibrationAccess of the referenced ApplicationPrimitiveDataType is meaningless for the using CURVE and MAP. Note: The referenced data type needs to be a real primitive (typically of category VALUE. Category BOOLEAN is also supported).

The ApplicationPrimitiveDataType of the CURVE and MAP can additionally define SwDataDefProps which are relevant for the whole compound primitive. Currently the following attributes are used in the example:

- swCalprmAxisSet
- swRecordLayout
- swCalibrationAccess (but will be refined on DataPrototype level)

Further on, via the dataTypeMapping of the using software component, the properties of ImplementationDataType and SwBaseType are described.

As axes of the MAP two group axes are used. The properties of the group axes are described by two ApplicationPrimitiveDataTypes of category COM_AXIS. The attribute swAxisIndex indicates for which dimension the group axis applies (1 = X, 2 = Y). With the attribute sharedAxisType the reference to the ApplicationPrimitiveDataType describing the axis is defined.

In the example, the group axis ComAxis_Temp_Lin_K_uint16 defines the the applicable minimum and maximum number of axis points. Additionally the inputVariableType reference to the ApplicationPrimitiveDataType K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16 defines the properties of the input value for the axis. This in turn corresponds to the values stored as axis point. The same principle applies for the group axis ComAxis_Mass_Lnr_Kg_uint8.

Please note, the above mentioned properties are defined on the level of ApplicationDataTypes and so far not any data instance implementing such properties exists. This requires an instantiation of such ApplicationDataTypes.

#### 3.2.1.6 Axis, Curves and Maps on DataPrototype and SwComponentPrototype level

##### 3.2.1.6.1 Instantiation of Axis, Curves and Maps

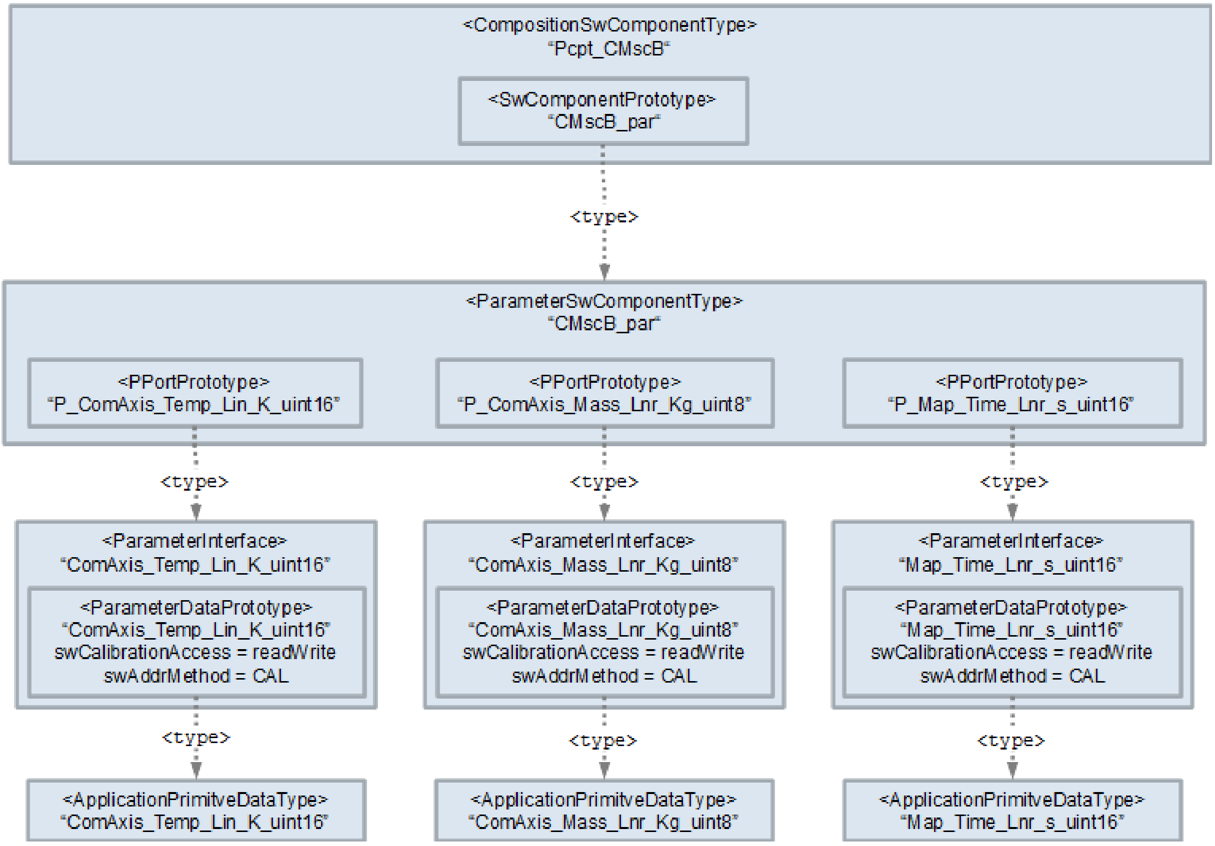


**Figure 3.14: Instantiation of a MAP and its group axes**

Figure 3.14 shows the instantiation of the ApplicationPrimitiveDataType ComAxis_Temp_Lin_K_uint16, ComAxis_Mass_Lnr_Kg_uint8, and Map_Time_Lnr_s_uint16 up to the level of the CompositionSwComponentType Pcpt_CMscB.

Thereby ParameterDataPrototypes are typed by the mentioned ApplicationPrimitiveDataTypes. Each ParameterDataPrototype is owned by an own ParameterInterface. This offers the most flexibility to instantiate the map and axes independently from each other. On the level of the ParameterDataPrototype additionally the swCalibrationAccess and the swAddrMethod is defined. Further on, the ParameterSwComponentType CMscB_par defines three PPortPrototypes typed by the ParameterInterfaces.

Please note that a group axes of a curve or map are not necessarily provided by the same ParameterSwComponentType as the one providing the curve or map. This case is illustrated with the map Arr1dMap_Time_Lnr_s_uint16 using the group axes Arr1dComAxis_Temp_Lin_K_uint16 provided by CMscD_par and ComAxis_Mass_Lnr_Kg_uint8 provided by CMscB_par.

#### 3.2.1.6.2 Usage of Axis, Curves and Maps by Software Components

#### 3.2.1.6.3 Linking map and curve instances to its axes instances

Consider a software component that uses curves and maps with group axes. It is than required to denote which instance of curve and map uses which instance of a group axis as axis of abscissae and, in case of a map, as axis of ordinate.

The AUTOSAR meta model provides hereby two possibilities:

- RunnableEntity.parameterAccess.swDataDefProps

  or

- SwcInternalBehavior.instantiationDataDefProps.swDataDefProps.

Inside one software component it's very unlikely, that the same curve or map is used with different axes by different RunnableEntitys (note that this cannot be expressed by ASAM MCD-2MC, also) . Therefore, in this show case the second ability is used. This avoids the risk of inconsistencies when several RunnableEntitys are defining parameterAccesses to the same curve or map instance.

The according instantiationDataDefProps.parameterInstance references the map instance in the scope of the SwComponentType and the swDataDefProps.swCalprmAxisSet.swCalprmAxis.swCalprmAxisTypeProps.swCalprmRef references the applied group axes with the according SwCalprmAxis.swAxisIndex

**Listing 3.28: Example of an InstantiationDataDefProps for an map**

```
<INSTANTIATION-DATA-DEF-PROPS>
  <PARAMETER-INSTANCE>
    <AUTOSAR-PARAMETER-IREF>
      <PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">/Tier1/ARPlatform1/
          Pcpt_CMscB/CMscB/R_Map_Time_Lnr_s_uint16</PORT-PROTOTYPE-REF>
      <TARGET-DATA-PROTOTYPE-REF DEST="PARAMETER-DATA-PROTOTYPE">/Tier1
          /ARPlatform1/DataDictionary/PortInterfaces/V1_0_0/
          Map_Time_Lnr_s_uint16/Map_Time_Lnr_s_uint16</TARGET-DATA-
          PROTOTYPE-REF>
    </AUTOSAR-PARAMETER-IREF>
  </PARAMETER-INSTANCE>
  <SW-DATA-DEF-PROPS>
```

```
        <SW-DATA-DEF-PROPS-VARIANTS>
          <SW-DATA-DEF-PROPS-CONDITIONAL>
            <SW-CALPRM-AXIS-SET>
              <SW-CALPRM-AXIS>
                <SW-AXIS-INDEX>1</SW-AXIS-INDEX>
                <SW-AXIS-GROUPED>
                  <AR-PARAMETER>
                    <AUTOSAR-PARAMETER-IREF>
                      <PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">/Tier1/
                          ARPlatform1/Pcpt_CMscB/CMscB/
                          R_ComAxis_Temp_Lin_K_uint16</PORT-PROTOTYPE-REF>
                      <TARGET-DATA-PROTOTYPE-REF DEST="PARAMETER-DATA-
                          PROTOTYPE">/Tier1/ARPlatform1/DataDictionary/
                          PortInterfaces/V1_0_0/ComAxis_Temp_Lin_K_uint16/
                          ComAxis_Temp_Lin_K_uint16</TARGET-DATA-PROTOTYPE-
                          REF>
                    </AUTOSAR-PARAMETER-IREF>
                  </AR-PARAMETER>
                </SW-AXIS-GROUPED>
              </SW-CALPRM-AXIS>
              <SW-CALPRM-AXIS>
                <SW-AXIS-INDEX>2</SW-AXIS-INDEX>
                <SW-AXIS-GROUPED>
                  <AR-PARAMETER>
                    <AUTOSAR-PARAMETER-IREF>
                      <PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">/Tier1/
                          ARPlatform1/Pcpt_CMscB/CMscB/
                          R_ComAxis_Mass_Lnr_Kg_uint8</PORT-PROTOTYPE-REF>
                      <TARGET-DATA-PROTOTYPE-REF DEST="PARAMETER-DATA-
                          PROTOTYPE">/Tier1/ARPlatform1/DataDictionary/
                          PortInterfaces/V1_0_0/ComAxis_Mass_Lnr_Kg_uint8/
                          ComAxis_Mass_Lnr_Kg_uint8</TARGET-DATA-PROTOTYPE-
                          REF>
                    </AUTOSAR-PARAMETER-IREF>ARPlatform1
                  </AR-PARAMETER>
                </SW-AXIS-GROUPED>
              </SW-CALPRM-AXIS>
            </SW-CALPRM-AXIS-SET>
          </SW-DATA-DEF-PROPS-CONDITIONAL>
        </SW-DATA-DEF-PROPS-VARIANTS>
      </SW-DATA-DEF-PROPS>
    </INSTANTIATION-DATA-DEF-PROPS>
```

#### 3.2.1.6.4 Linking axes instances to its working point instances

When a software component uses compound primitives containing axes (e.g. curves, maps, or group axes) it's beneficial to indicate which data is used as input for the according axis. This enables the measurement and calibration tool to display the current working point. Like explained in section 3.2.1.6.3, this information can be provided

- at the ParameterAccess.swDataDefProps of the compound primitives containing the axis **or**

- by means of instantiationDataDefProps.swDataDefProps.

In this show case the second ability is used for the same reasons as discussed in section 3.2.1.6.3.

The according instantiationDataDefProps.parameterInstance references the axes instance in the scope of the SwComponentType CMscB. The swDataDefProps.swCalprmAxisSet.swCalprmAxis.swCalprmAxisTypeProps.swVariableRef references the applied working point variable (in this case, a dataElement in a RPortPrototype) with the according SwCalprmAxis.swAxisIndex.

**Listing 3.29: Example of an InstantiationDataDefProps for an axis**

```
<INSTANTIATION-DATA-DEF-PROPS>
  <PARAMETER-INSTANCE>
    <AUTOSAR-PARAMETER-IREF>
      <PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">/Tier1/ARPlatform1/
          Pcpt_CMscB/CMscB/R_ComAxis_Temp_Lin_K_uint16</PORT-PROTOTYPE-
          REF>
      <TARGET-DATA-PROTOTYPE-REF DEST="PARAMETER-DATA-PROTOTYPE">/Tier1
          /ARPlatform1/DataDictionary/PortInterfaces/V1_0_0/
          ComAxis_Temp_Lin_K_uint16/ComAxis_Temp_Lin_K_uint16</TARGET-
          DATA-PROTOTYPE-REF>
    </AUTOSAR-PARAMETER-IREF>
  </PARAMETER-INSTANCE>
  <SW-DATA-DEF-PROPS>
    <SW-DATA-DEF-PROPS-VARIANTS>
      <SW-DATA-DEF-PROPS-CONDITIONAL>
        <SW-CALPRM-AXIS-SET>
          <SW-CALPRM-AXIS>
            <SW-AXIS-INDEX>1</SW-AXIS-INDEX>
            <SW-AXIS-INDIVIDUAL>
              <SW-VARIABLE-REFS>
                <AUTOSAR-VARIABLE>
                  <AUTOSAR-VARIABLE-IREF>
                    <PORT-PROTOTYPE-REF DEST="R-PORT-PROTOTYPE">/Tier1/
                        ARPlatform1/Pcpt_CMscB/CMscB/
                        R_PrimData_Temperature_Lin_K_C_uint16</PORT-
                        PROTOTYPE-REF>
                    <TARGET-DATA-PROTOTYPE-REF DEST="VARIABLE-DATA-
                        PROTOTYPE">/Tier1/ARPlatform1/DataDictionary/
                        PortInterfaces/V1_0_0/
                        PrimData_Temperature_Lin_K_C_uint16/
                        PrimData_Temperature_Lin_K_C_uint16</TARGET-
                        DATA-PROTOTYPE-REF>
                  </AUTOSAR-VARIABLE-IREF>
                </AUTOSAR-VARIABLE>
              </SW-VARIABLE-REFS>
            </SW-AXIS-INDIVIDUAL>
          </SW-CALPRM-AXIS>
        </SW-CALPRM-AXIS-SET>
      </SW-DATA-DEF-PROPS-CONDITIONAL>
    </SW-DATA-DEF-PROPS-VARIANTS>
  </SW-DATA-DEF-PROPS>
</INSTANTIATION-DATA-DEF-PROPS>
```

#### 3.2.1.6.5 Axis, Curves and Maps in the ECU Flat Map

The ECU Flat Map contains entries for all curves, maps, axes and working point variables. The used naming patterns are described in 3.2.1.2.1.

**Listing 3.30: Example of a FlatInstanceDescriptor for map axis and working point variable**

```
<FLAT-INSTANCE-DESCRIPTOR>
  <SHORT-NAME>Map_Time_Lnr_s_uint16</SHORT-NAME>
  <ECU-EXTRACT-REFERENCE-IREF>
    <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/Tier1/
       ARPlatform1/System/SystemU_System/SystemU</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/Tier1/
       ARPlatform1/System/CompositionSwComponentTypes/SystemU_Root/
       CMscB_par</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="P-PORT-PROTOTYPE">/Tier1/ARPlatform1/
       Pcpt_CMscB/CMscB_par/P_Map_Time_Lnr_s_uint16</CONTEXT-ELEMENT-
       REF>
    <TARGET-REF DEST="PARAMETER-DATA-PROTOTYPE">/Tier1/ARPlatform1/
       DataDictionary/PortInterfaces/V1_0_0/Map_Time_Lnr_s_uint16/
       Map_Time_Lnr_s_uint16</TARGET-REF>
  </ECU-EXTRACT-REFERENCE-IREF>
</FLAT-INSTANCE-DESCRIPTOR>
<FLAT-INSTANCE-DESCRIPTOR>
  <SHORT-NAME>ComAxis_Temp_Lin_K_uint16</SHORT-NAME>
  <ECU-EXTRACT-REFERENCE-IREF>
    <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/Tier1/
       ARPlatform1/System/SystemU_System/SystemU</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/Tier1/
       ARPlatform1/System/CompositionSwComponentTypes/SystemU_Root/
       CMscB_par</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="P-PORT-PROTOTYPE">/Tier1/ARPlatform1/
       Pcpt_CMscB/CMscB_par/P_ComAxis_Temp_Lin_K_uint16</CONTEXT-
       ELEMENT-REF>
    <TARGET-REF DEST="PARAMETER-DATA-PROTOTYPE">/Tier1/ARPlatform1/
       DataDictionary/PortInterfaces/V1_0_0/ComAxis_Temp_Lin_K_uint16/
       ComAxis_Temp_Lin_K_uint16</TARGET-REF>
  </ECU-EXTRACT-REFERENCE-IREF>
</FLAT-INSTANCE-DESCRIPTOR>
<FLAT-INSTANCE-DESCRIPTOR>
  <SHORT-NAME>PrimData_Temperature_Lin_K_C_uint16</SHORT-NAME>
  <ECU-EXTRACT-REFERENCE-IREF>
    <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/Tier1/
       ARPlatform1/System/SystemU_System/SystemU</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/Tier1/
       ARPlatform1/System/CompositionSwComponentTypes/SystemU_Root/
       CMscA</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="P-PORT-PROTOTYPE">/Tier1/ARPlatform1/
       Pcpt_CMscA/CMscA/P_PrimData_Temperature_Lin_K_C_uint16</CONTEXT
       -ELEMENT-REF>
    <TARGET-REF DEST="VARIABLE-DATA-PROTOTYPE">/Tier1/ARPlatform1/
       DataDictionary/PortInterfaces/V1_0_0/
       PrimData_Temperature_Lin_K_C_uint16/
       PrimData_Temperature_Lin_K_C_uint16</TARGET-REF>
  </ECU-EXTRACT-REFERENCE-IREF>
```

```
</FLAT-INSTANCE-DESCRIPTOR>
```

#### 3.2.1.7 Arrays of Maps and Axes

The ability of curves, maps and cuboids is usually used to describe the physical dependency of a characteristic on other physical input values. Hereby each input value is described by an orthogonal axis. In contrast to this, arrays are used to group a set of values of the same nature which can be handled by the same algorithm. Typically, in this case the algorithm iterates over the array with an index. Nevertheless, each array element may represent a particular part of the vehicle, e.g. a specific cylinder or a specific sensor. It's possible to combine these design principles. This ends up in the need to describe arrays of curves, maps, cuboids and the according axes.

The show case illustrates the model of those objects by the following elements:

- Arr1dMap_Time_Lnr_s_uint16
- Arr1dComAxis_Temp_Lin_K_uint16
- Arr1dPrimData_Temperature_Lin_K_C_uint16

Hereby, the array of the map Arr1dMap_Time_Lnr_s_uint16 uses for the x-axis an array of group axes Arr1dComAxis_Temp_Lin_K_uint16 which in turn uses an array of primitive values as working points Arr1dPrimData_Temperature_Lin_K_C_uint16. In this case, the n'th map uses the n'th x-axes which uses the n'th value as working point. In contrast, the map uses one group axis ComAxis_Mass_Lnr_Kg_uint8 for the y-axis. In this case all maps in the array are using the same y-axis.

##### 3.2.1.7.1 Arrays of Maps and Axes in the ECU Flat Map

In the ECU Flat Map the ability to reference ApplicationCompositeElementDataPrototypes is used to express the specific meaning of each array element in the array of map and group axis.

For instance, each element in the array Arr1dMap_Time_Lnr_s_uint16 is named in a way to indicate the specific meaning:

- Arr1dMap_Time_Lnr_s_uint16_FrontLeft
- Arr1dMap_Time_Lnr_s_uint16_FrontRight
- Arr1dMap_Time_Lnr_s_uint16_RearLeft
- Arr1dMap_Time_Lnr_s_uint16_RearRight

The following listing shows the structure of such an FlatInstanceDescriptior on one example:

**Listing 3.31: Example of a FlatInstanceDescriptor for an ApplicationCompositeElementDataPrototype**

```
<FLAT-INSTANCE-DESCRIPTOR>
  <SHORT-NAME>Arr1dMap_Time_Lnr_s_uint16_FrontLeft</SHORT-NAME>
  <ECU-EXTRACT-REFERENCE-IREF>
    <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/Tier1/
       ARPlatform1/System/SystemU_System/SystemU</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/Tier1/
       ARPlatform1/System/CompositionSwComponentTypes/SystemU_Root/
       CMscD_par</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="P-PORT-PROTOTYPE">/Tier1/ARPlatform1/
       Pcpt_CMscD/CMscD_par/P_Arr1dMap_Time_Lnr_s_uint16</CONTEXT-
       ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="PARAMETER-DATA-PROTOTYPE">/Tier1/
       ARPlatform1/DataDictionary/PortInterfaces/V1_0_0/
       Arr1dMap_Time_Lnr_s_uint16/Arr1dMap_Time_Lnr_s_uint16</CONTEXT-
       ELEMENT-REF>
    <TARGET-REF DEST="APPLICATION-ARRAY-ELEMENT" INDEX="0">/Tier1/
       ARPlatform1/DataDictionary/ApplicationDataTypes/
       Map_Time_Lnr_s_uint16_ScNoOfWheels/
       Map_Time_Lnr_s_uint16_ScNoOfWheels</TARGET-REF>
  </ECU-EXTRACT-REFERENCE-IREF>
</FLAT-INSTANCE-DESCRIPTOR>
```

Please note the usage of the index attribute in the target reference.

#### 3.2.1.8 Measurement of Modes

##### 3.2.1.8.1 Enabling Measurement of Modes

The measurement of a mode is enabled in the software-component description by setting the ModeDeclarationGroupPrototype.swCalibrationAccess to readOnly. See ModeDirection.

##### 3.2.1.8.2 Modes in the ECU Flat Map

AUTOSAR supports the measurement of the current mode, the previous mode and the next mode. Hereby the last two are useful when the mode is measured during a ongoing transition to identify the kind of transition. In this show case only the measurement of the current mode is illustrated. For this, the FlatMap contains a FlatInstanceDescriptor pointing to the ModeDeclarationGroupPrototype which is to be measured. The role attribute of the FlatInstanceDescriptor is set to CURRENT_MODE

**Listing 3.32: Example of a FlatInstanceDescriptor for a ModeDeclarationGroupPrototype**

```
<FLAT-INSTANCE-DESCRIPTOR>
  <SHORT-NAME>ModeDirection</SHORT-NAME>
  <ROLE>CURRENT_MODE</ROLE>
```

```
  <ECU-EXTRACT-REFERENCE-IREF>
    <CONTEXT-ELEMENT-REF DEST="ROOT-SW-COMPOSITION-PROTOTYPE">/Tier1/
       ARPlatform1/System/SystemU_System/SystemU</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="SW-COMPONENT-PROTOTYPE">/Tier1/
       ARPlatform1/System/CompositionSwComponentTypes/SystemU_Root/
       CMscA</CONTEXT-ELEMENT-REF>
    <CONTEXT-ELEMENT-REF DEST="P-PORT-PROTOTYPE">/Tier1/ARPlatform1/
       Pcpt_CMscA/CMscA/P_ModeDirection</CONTEXT-ELEMENT-REF>
    <TARGET-REF DEST="MODE-DECLARATION-GROUP-PROTOTYPE">/Tier1/
       ARPlatform1/DataDictionary/PortInterfaces/V1_0_0/ModeDirection/
       ModeDirection</TARGET-REF>
  </ECU-EXTRACT-REFERENCE-IREF>
</FLAT-INSTANCE-DESCRIPTOR>
```

### 3.2.2 Show cases in the Example

#### 3.2.2.1 CompositionSwComponentTypes

| Common CompositionSwComponentType attributes | |
|---|---|
| shortName | Pcpt_CMscA |
| desc | Modeling show case for primitive measurement and calculation. |
| **properties of the ports** | |
| **properties of PPortPrototype** | |
| shortName | P_ModeDirection |
| desc | Mode to indicate a direction |
| providedInterface | ModeDirection |
| **properties of PPortPrototype** | |
| shortName | P_PrimCal_Mass_Lnr_Kg |
| desc | Primitive calibration parameter for minimum egg mass. |
| providedInterface | PrimCal_Mass_Lnr_Kg |
| **properties of PPortPrototype** | |
| shortName | P_PrimData_Mass_Lnr_Kg_uint8 |
| desc | Mass in kilogram |
| providedInterface | PrimData_Mass_Lnr_Kg_uint8 |
| **properties of PPortPrototype** | |
| shortName | P_PrimData_StepsSpeed_Txt_sint8 |
| desc | Stepwise speed indication |
| providedInterface | PrimData_StepsSpeed_Txt_sint8 |
| **properties of PPortPrototype** | |
| shortName | P_PrimData_Temperature_Lin_K_C_uint16 |
| desc | Temperature 1 in Kelvin but displayed as degree Celsius |
| providedInterface | PrimData_Temperature_Lin_K_C_uint16 |
| **properties of RPortPrototype** | |
| shortName | R_PrimData_StepsSpeed_Txt_sint8 |
| desc | Stepwise speed indication |
| requiredInterface | PrimData_StepsSpeed_Txt_sint8 |
| **properties of the components** | |
| **properties of SwComponentPrototype** | |
| shortName | CMscA |
| type | CMscA |
| **properties of SwComponentPrototype** | |
| shortName | CMscA_par |
| type | CMscA_par |

**Table 3.53: CompositionSwComponentType Pcpt_CMscA**

| Common CompositionSwComponentType attributes | |
|---|---|
| shortName | SystemU_Root |
| properties of the components | |
| properties of SwComponentPrototype | |
| shortName | CMscA |
| type | CMscA |
| properties of SwComponentPrototype | |
| shortName | CMscA_par |
| type | CMscA_par |
| properties of SwComponentPrototype | |
| shortName | CMscB |
| type | CMscB |
| properties of SwComponentPrototype | |
| shortName | CMscB_par |
| type | CMscB_par |
| properties of SwComponentPrototype | |
| shortName | CMscC_nvm |
| type | CMscC_nvm |
| properties of SwComponentPrototype | |
| shortName | CMscD |
| type | CMscD |
| properties of SwComponentPrototype | |
| shortName | CMscD_par |
| type | CMscD_par |

**Table 3.54: CompositionSwComponentType SystemU_Root**

| Common CompositionSwComponentType attributes | |
|---|---|
| shortName | SystemURootComposition_EcuDescr |
| properties of the components | |
| properties of SwComponentPrototype | |
| shortName | CMscC |
| type | Pcpt_CMscC |

**Table 3.55: CompositionSwComponentType SystemURootComposition_EcuDescr**

| Common CompositionSwComponentType attributes | |
|---|---|
| shortName | Pcpt_CMscD |
| desc | Modeling show case for arrays of axes and mapes. |
| **properties of the ports** | |
| **properties of RPortPrototype** | |
| shortName | R_ComAxis_Mass_Lnr_Kg_uint8 |
| desc | Shared axis for mass |
| requiredInterface | ComAxis_Mass_Lnr_Kg_uint8 |
| **properties of RPortPrototype** | |
| shortName | R_PrimData_MassCorrected_Lnr_Kg_uint8 |
| desc | Primitve data for the corrected mass in kg. |
| requiredInterface | PrimData_MassCorrected_Lnr_Kg_uint8 |
| **properties of the components** | |
| **properties of SwComponentPrototype** | |
| shortName | CMscD |
| type | CMscD |
| **properties of SwComponentPrototype** | |
| shortName | CMscD_par |
| type | CMscD_par |

**Table 3.56: CompositionSwComponentType Pcpt_CMscD**

| Common CompositionSwComponentType attributes | |
|---|---|
| shortName | Pcpt_CMscC |
| desc | Composit of modeling show case C |
| **properties of the ports** | |
| **properties of PPortPrototype** | |
| shortName | P_PrimData_Time_Lnr_s_uint16 |
| desc | Primitve data holding a time value. |
| providedInterface | PrimData_Time_Lnr_s_uint16 |
| **properties of PPortPrototype** | |
| shortName | P_PrimData_ValidState_Txt_noUnit_boolean |
| desc | Boolean representing the data validity |
| providedInterface | PrimData_ValidState_Txt_noUnit_boolean |
| **properties of the components** | |
| **properties of SwComponentPrototype** | |
| shortName | CMscA |
| type | Pcpt_CMscA |
| **properties of SwComponentPrototype** | |
| shortName | CMscB |
| type | Pcpt_CMscB |
| **properties of SwComponentPrototype** | |
| shortName | CMscC_nvm |
| type | CMscC_nvm |
| **properties of SwComponentPrototype** | |
| shortName | CMscD |
| type | Pcpt_CMscD |

**Table 3.57: CompositionSwComponentType Pcpt_CMscC**

| Common CompositionSwComponentType attributes | |
|---|---|
| shortName | Pcpt_CMscB |
| desc | Modeling show case for axes, curves and mapes. |
| properties of the ports | |
| properties of PPortPrototype | |
| shortName | P_ComAxis_Mass_Lnr_Kg_uint8 |
| desc | Shared axis for mass |
| providedInterface | ComAxis_Mass_Lnr_Kg_uint8 |
| properties of PPortPrototype | |
| shortName | P_PrimData_MassCorrected_Lnr_Kg_uint8 |
| desc | Primitve data for the corrected mass in kg. |
| providedInterface | PrimData_MassCorrected_Lnr_Kg_uint8 |
| properties of PPortPrototype | |
| shortName | P_PrimData_Time_Lnr_s_uint16 |
| desc | Primitve data holding a time value. |
| providedInterface | PrimData_Time_Lnr_s_uint16 |
| properties of PPortPrototype | |
| shortName | P_PrimData_ValidState_Txt_noUnit_boolean |
| desc | Boolean representing the data validity |
| providedInterface | PrimData_ValidState_Txt_noUnit_boolean |
| properties of RPortPrototype | |
| shortName | R_ModeDirection |
| desc | Mode to indicate a direction |
| requiredInterface | ModeDirection |
| properties of RPortPrototype | |
| shortName | R_PrimData_Mass_Lnr_Kg_uint8 |
| desc | Mass in kilogram |
| requiredInterface | PrimData_Mass_Lnr_Kg_uint8 |
| properties of RPortPrototype | |
| shortName | R_PrimData_StepsSpeed_Txt_sint8 |
| desc | Stepwise speed indication |
| requiredInterface | PrimData_StepsSpeed_Txt_sint8 |
| properties of RPortPrototype | |
| shortName | R_PrimData_Temperature_Lin_K_C_uint16 |
| desc | Temperature 1 in Kelvin but displayed as degree Celsius |
| requiredInterface | PrimData_Temperature_Lin_K_C_uint16 |
| properties of the components | |
| properties of SwComponentPrototype | |

| | | |
|---|---|---|
| | shortName | CMscB |
| | type | CMscB |
| properties of SwComponentPrototype | | |
| | shortName | CMscB_par |
| | type | CMscB_par |

**Table 3.58: CompositionSwComponentType Pcpt_CMscB**

#### 3.2.2.2 ParameterSwComponentTypes

| Common ParameterSwComponentType attributes | |
|---|---|
| shortName | CMscA_par |
| desc | Modeling show case for primitive measurement and calculation. |
| properties of the ports | |
| properties of PPortPrototype | |
| shortName | P_PrimCal_Mass_Lnr_Kg |
| desc | Primitive calibration parameter for minimum egg mass. |
| providedInterface | PrimCal_Mass_Lnr_Kg |

**Table 3.59: ParameterSwComponentType CMscA_par**

| Common ParameterSwComponentType attributes | |
|---|---|
| shortName | CMscD_par |
| desc | Modeling show case for arrays of axes and mapes. |
| properties of the ports | |
| properties of PPortPrototype | |
| shortName | P_Arr1dComAxis_Temp_Lin_K_uint16 |
| desc | Array of shared axis for temperature |
| providedInterface | Arr1dComAxis_Temp_Lin_K_uint16 |
| properties of PPortPrototype | |
| shortName | P_Arr1dMap_Time_Lnr_s_uint16 |
| desc | Map to get time dependent on temperature and mass. |
| providedInterface | Arr1dMap_Time_Lnr_s_uint16 |

**Table 3.60: ParameterSwComponentType CMscD_par**

| Common ParameterSwComponentType attributes | |
|---|---|
| shortName | CMscB_par |
| desc | Modeling show case for axes, curves and mapes. |
| **properties of the ports** | |
| **properties of PPortPrototype** | |
| shortName | P_ComAxis_Mass_Lnr_Kg_uint8 |
| desc | Shared axis for mass |
| providedInterface | ComAxis_Mass_Lnr_Kg_uint8 |
| **properties of PPortPrototype** | |
| shortName | P_ComAxis_Steps_Txt_sint8 |
| desc | Shared axis for speed steps |
| providedInterface | ComAxis_Steps_Txt_sint8 |
| **properties of PPortPrototype** | |
| shortName | P_ComAxis_Temp_Lin_K_uint16 |
| desc | Shared axis for temperature |
| providedInterface | ComAxis_Temp_Lin_K_uint16 |
| **properties of PPortPrototype** | |
| shortName | P_Curve_Mass_Lnr_Kg_uint8 |
| desc | Curve to get mass according differnt speed steps. |
| providedInterface | Curve_Mass_Lnr_Kg_uint8 |
| **properties of PPortPrototype** | |
| shortName | P_Map_Time_Lnr_s_uint16 |
| desc | Map to get time dependent on temperature and mass. |
| providedInterface | Map_Time_Lnr_s_uint16 |

**Table 3.61: ParameterSwComponentType CMscB_par**

#### 3.2.2.3 ApplicationSwComponentTypes

| Common ApplicationSwComponentType attributes | |
|---|---|
| shortName | CMscD |
| desc | Modeling show case for arrays of axes and mapes. |
| **properties of the ports** | |
| **properties of PPortPrototype** | |
| shortName | P_Arr1dPrimData_Temperature_Lin_K_C_uint16 |
| desc | Temperature 1 in Kelvin but displayed as degree Celsius |
| providedInterface | Arr1dPrimData_Temperature_Lin_K_C_uint16 |
| **properties of RPortPrototype** | |
| shortName | R_Arr1dComAxis_Temp_Lin_K_uint16 |
| desc | Array of shared axis for temperature |
| requiredInterface | Arr1dComAxis_Temp_Lin_K_uint16 |
| **properties of RPortPrototype** | |
| shortName | R_Arr1dMap_Time_Lnr_s_uint16 |
| desc | Map to get time dependent on temperature and mass. |
| requiredInterface | Arr1dMap_Time_Lnr_s_uint16 |
| **properties of RPortPrototype** | |
| shortName | R_Arr1dPrimData_Temperature_Lin_K_C_uint16 |
| desc | Temperature 1 in Kelvin but displayed as degree Celsius |
| requiredInterface | Arr1dPrimData_Temperature_Lin_K_C_uint16 |
| **properties of RPortPrototype** | |
| shortName | R_ComAxis_Mass_Lnr_Kg_uint8 |
| desc | Shared axis for mass |
| requiredInterface | ComAxis_Mass_Lnr_Kg_uint8 |
| **properties of RPortPrototype** | |
| shortName | R_PrimData_MassCorrected_Lnr_Kg_uint8 |
| desc | Primitve data for the corrected mass in kg. |
| requiredInterface | PrimData_MassCorrected_Lnr_Kg_uint8 |
| internalBehavior | CMscD |

**Table 3.62: ApplicationSwComponentType CMscD**

| Common SwcInternalBehavior attributes | |
|---|---|
| shortName | CMscD |
| properties of the runnables | |
| properties of RunnableEntity | |
| shortName | CMscD_Process |
| symbol | CMscD_Process |

Table 3.63: SwcInternalBehavior CMscD

| Common ApplicationSwComponentType attributes | |
|---|---|
| shortName | CMscB |
| desc | Modeling show case for axes, curves and mapes. |
| **properties of the ports** | |
| **properties of PPortPrototype** | |
| shortName | P_PrimData_MassCorrected_Lnr_Kg_uint8 |
| desc | Primitve data for the corrected mass in kg. |
| providedInterface | PrimData_MassCorrected_Lnr_Kg_uint8 |
| **properties of PPortPrototype** | |
| shortName | P_PrimData_Time_Lnr_s_uint16 |
| desc | Primitve data holding a time value. |
| providedInterface | PrimData_Time_Lnr_s_uint16 |
| **properties of PPortPrototype** | |
| shortName | P_PrimData_ValidState_Txt_noUnit_boolean |
| desc | Boolean representing the data validity |
| providedInterface | PrimData_ValidState_Txt_noUnit_boolean |
| **properties of RPortPrototype** | |
| shortName | R_ComAxis_Mass_Lnr_Kg_uint8 |
| desc | Shared axis for mass |
| requiredInterface | ComAxis_Mass_Lnr_Kg_uint8 |
| **properties of RPortPrototype** | |
| shortName | R_ComAxis_Steps_Txt_sint8 |
| desc | Shared axis for speed steps |
| requiredInterface | ComAxis_Steps_Txt_sint8 |
| **properties of RPortPrototype** | |
| shortName | R_ComAxis_Temp_Lin_K_uint16 |
| desc | Shared axis for temperature |
| requiredInterface | ComAxis_Temp_Lin_K_uint16 |
| **properties of RPortPrototype** | |
| shortName | R_Curve_Mass_Lnr_Kg_uint8 |
| desc | Curve to get mass according differnt speed steps. |
| requiredInterface | Curve_Mass_Lnr_Kg_uint8 |
| **properties of RPortPrototype** | |
| shortName | R_Map_Time_Lnr_s_uint16 |
| desc | Map to get time dependent on temperature and mass. |
| requiredInterface | Map_Time_Lnr_s_uint16 |
| **properties of RPortPrototype** | |
| shortName | R_ModeDirection |
| desc | Mode to indicate a direction |

| | |
|---|---|
| requiredInterface | ModeDirection |
| properties of RPortPrototype | |
| shortName | R_PrimData_Mass_Lnr_Kg_uint8 |
| desc | Mass in kilogram |
| requiredInterface | PrimData_Mass_Lnr_Kg_uint8 |
| properties of RPortPrototype | |
| shortName | R_PrimData_MassCorrected_Lnr_Kg_uint8 |
| desc | Primitve data for the corrected mass in kg. |
| requiredInterface | PrimData_MassCorrected_Lnr_Kg_uint8 |
| properties of RPortPrototype | |
| shortName | R_PrimData_StepsSpeed_Txt_sint8 |
| desc | Stepwise speed indication |
| requiredInterface | PrimData_StepsSpeed_Txt_sint8 |
| properties of RPortPrototype | |
| shortName | R_PrimData_Temperature_Lin_K_C_uint16 |
| desc | Temperature 1 in Kelvin but displayed as degree Celsius |
| requiredInterface | PrimData_Temperature_Lin_K_C_uint16 |
| properties of RPortPrototype | |
| shortName | R_PrimData_Time_Lnr_s_uint16 |
| desc | Primitve data holding a time value. |
| requiredInterface | PrimData_Time_Lnr_s_uint16 |
| properties of RPortPrototype | |
| shortName | R_PrimData_ValidState_Txt_noUnit_boolean |
| desc | Boolean representing the data validity |
| requiredInterface | PrimData_ValidState_Txt_noUnit_boolean |
| internalBehavior | CMscB |

**Table 3.64: ApplicationSwComponentType CMscB**

| Common SwcInternalBehavior attributes | |
|---|---|
| shortName | CMscB |
| properties of the runnables | |
| properties of RunnableEntity | |
| shortName | CMscB_Process |
| desc | cyclic process for calculation |
| symbol | CMscB_Process |

**Table 3.65: SwcInternalBehavior CMscB**

### 3.2.2.4 ParameterInterfaces

| Common ParameterInterface attributes | |
|---|---|
| shortName | Arr1dComAxis_Temp_Lin_K_uint16 |
| desc | Array of shared axis for temperature |
| properties of the parameters | |
| properties of ParameterDataPrototype | |
| shortName | Arr1dComAxis_Temp_Lin_K_uint16 |
| desc | Array of shared axis for temperature |
| type | ComAxis_Temp_Lin_K_uint16_ScNoOfWheels |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CAL |

Table 3.66: ParameterInterface Arr1dComAxis_Temp_Lin_K_uint16

| Common ParameterInterface attributes | |
|---|---|
| shortName | Arr1dMap_Time_Lnr_s_uint16 |
| desc | Map to get time dependent on temperature and mass. |
| properties of the parameters | |
| properties of ParameterDataPrototype | |
| shortName | Arr1dMap_Time_Lnr_s_uint16 |
| desc | Map to get time dependent on temperature and mass. |
| type | Map_Time_Lnr_s_uint16_ScNoOfWheels |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CAL |

Table 3.67: ParameterInterface Arr1dMap_Time_Lnr_s_uint16

| Common ParameterInterface attributes | |
|---|---|
| shortName | ComAxis_Mass_Lnr_Kg_uint8 |
| desc | Shared axis for mass |
| properties of the parameters | |
| properties of ParameterDataPrototype | |
| shortName | ComAxis_Mass_Lnr_Kg_uint8 |
| desc | Shared axis for mass |
| type | ComAxis_Mass_Lnr_Kg_uint8 |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CAL |

Table 3.68: ParameterInterface ComAxis_Mass_Lnr_Kg_uint8

| Common ParameterInterface attributes | |
|---|---|
| shortName | ComAxis_Steps_Txt_sint8 |
| desc | Shared axis for speed steps |
| properties of the parameters | |
| properties of ParameterDataPrototype | |
| shortName | ComAxis_Steps_Txt_sint8 |
| desc | Shared axis for speed steps |
| type | ComAxis_Steps_Txt_sint8 |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CAL |

**Table 3.69: ParameterInterface ComAxis_Steps_Txt_sint8**

| Common ParameterInterface attributes | |
|---|---|
| shortName | ComAxis_Temp_Lin_K_uint16 |
| desc | Shared axis for temperature |
| properties of the parameters | |
| properties of ParameterDataPrototype | |
| shortName | ComAxis_Temp_Lin_K_uint16 |
| desc | Shared axis for temperature |
| type | ComAxis_Temp_Lin_K_uint16 |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CAL |

**Table 3.70: ParameterInterface ComAxis_Temp_Lin_K_uint16**

| Common ParameterInterface attributes | |
|---|---|
| shortName | Curve_Mass_Lnr_Kg_uint8 |
| desc | Curve to get mass according differnt speed steps. |
| properties of the parameters | |
| properties of ParameterDataPrototype | |
| shortName | Curve_Mass_Lnr_Kg_uint8 |
| desc | Curve to get mass according differnt speed steps. |
| type | Curve_Mass_Lnr_Kg_uint8 |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CAL |

**Table 3.71: ParameterInterface Curve_Mass_Lnr_Kg_uint8**

| Common ParameterInterface attributes | |
|---|---|
| shortName | Map_Time_Lnr_s_uint16 |
| desc | Map to get time dependent on temperature and mass. |
| **properties of the parameters** | |
| **properties of ParameterDataPrototype** | |
| shortName | Map_Time_Lnr_s_uint16 |
| desc | Map to get time dependent on temperature and mass. |
| type | Map_Time_Lnr_s_uint16 |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CAL |

**Table 3.72: ParameterInterface Map_Time_Lnr_s_uint16**

| Common ParameterInterface attributes | |
|---|---|
| shortName | PrimCal_Mass_Lnr_Kg |
| desc | Primitive calibration parameter for minimum egg mass. |
| **properties of the parameters** | |
| **properties of ParameterDataPrototype** | |
| shortName | PrimCal_Mass_Lnr_Kg |
| desc | Primitive calibration parameter for minimum egg mass. |
| type | kg_Lnr_0_0d25_0_C8_uint8 |
| swImplPolicy | standard |
| swCalibrationAccess | readWrite |
| swAddrMethod | CAL |

**Table 3.73: ParameterInterface PrimCal_Mass_Lnr_Kg**

#### 3.2.2.5 ModeSwitchInterfaces

| Common ModeSwitchInterface attributes | | |
|---|---|---|
| shortName | | ModeDirection |
| desc | | Mode to indicate a direction |
| properties of the modeGroups | | |
| | shortName | ModeDirection |
| | swCalibrationAccess | readOnly |
| | type | Direction |

Table 3.74: ModeSwitchInterface ModeDirection

#### 3.2.2.6 SenderReceiverInterfaces

| Common SenderReceiverInterface attributes | |
|---|---|
| shortName | Arr1dPrimData_Temperature_Lin_K_C_uint16 |
| desc | Temperature 1 in Kelvin but displayed as degree Celsius |
| properties of the dataElementss | |
| properties of VariableDataPrototype | |
| shortName | Arr1dPrimData_Temperature_Lin_K_C_uint16 |
| desc | Temperature 1 in Kelvin but displayed as degree Celsius |
| type | K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16_ScNoOfWheels |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | DATA |

**Table 3.75: SenderReceiverInterface Arr1dPrimData_Temperature_Lin_K_C_uint16**

| Common SenderReceiverInterface attributes | |
|---|---|
| shortName | PrimData_Mass_Lnr_Kg_uint8 |
| desc | Mass in kilogram |
| properties of the dataElementss | |
| properties of VariableDataPrototype | |
| shortName | PrimData_Mass_Lnr_Kg_uint8 |
| desc | Mass in kilogram |
| type | kg_Lnr_0_0d25_0_C8_uint8 |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | DATA |

**Table 3.76: SenderReceiverInterface PrimData_Mass_Lnr_Kg_uint8**

| Common SenderReceiverInterface attributes | |
|---|---|
| shortName | PrimData_MassCorrected_Lnr_Kg_uint8 |
| desc | Primitve data for the corrected mass in kg. |
| properties of the dataElementss | |
| properties of VariableDataPrototype | |
| shortName | PrimData_MassCorrected_Lnr_Kg_uint8 |
| desc | Primitve data for the corrected mass in kg. |
| type | kg_Lnr_0_0d25_0_C8_uint8 |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | DATA |

**Table 3.77: SenderReceiverInterface PrimData_MassCorrected_Lnr_Kg_uint8**

| Common SenderReceiverInterface attributes | |
|---|---|
| shortName | PrimData_Temperature_Lin_K_C_uint16 |
| desc | Temperature 1 in Kelvin but displayed as degree Celsius |
| properties of the dataElementss | |
| properties of VariableDataPrototype | |
| shortName | PrimData_Temperature_Lin_K_C_uint16 |
| desc | Temperature 1 in Kelvin but displayed as degree Celsius |
| type | K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16 |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | DATA |

**Table 3.78: SenderReceiverInterface PrimData_Temperature_Lin_K_C_uint16**

| Common SenderReceiverInterface attributes | |
|---|---|
| shortName | PrimData_Time_Lnr_s_uint16 |
| desc | Primitve data holding a time value. |
| properties of the dataElementss | |
| properties of VariableDataPrototype | |
| shortName | PrimData_Time_Lnr_s_uint16 |
| desc | Primitve data holding a time value. |
| type | s_Lnr_0_8191d875_0_FFFF_uint16 |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | DATA |

**Table 3.79: SenderReceiverInterface PrimData_Time_Lnr_s_uint16**

| Common SenderReceiverInterface attributes | |
|---|---|
| shortName | PrimData_ValidState_Txt_noUnit_boolean |
| desc | Boolean representing the data validity |
| properties of the dataElementss | |
| properties of VariableDataPrototype | |
| shortName | PrimData_ValidState_Txt_noUnit_boolean |
| desc | Boolean representing the data validity |
| type | DataValidityType |
| swImplPolicy | standard |
| swCalibrationAccess | readOnly |
| swAddrMethod | DATA |

**Table 3.80: SenderReceiverInterface PrimData_ValidState_Txt_noUnit_boolean**

#### 3.2.2.7 ApplicationDataTypes, Category BOOLEAN

| Common ApplicationDataType attributes | | | | |
|---|---|---|---|---|
| shortName | DataValidityType | | | |
| category | BOOLEAN | | | |
| desc | Boolean to represent the data validity | | | |
| swCalibrationAccess | notAccessible | | | |
| unit | NoUnit | | | |
| Range | | | | |
| | lowerLimit | upperLimit | | |
| physConstrs | 0 | 1 | | |
| Conversion | | | | |
| category | TEXTTABLE | | | |
| direction | compuInternalToPhys | | | |
| desc | lowerLimit | upperLimit | vt | symbol |
| - | 0 | 0 | Invalid | |
| - | 1 | 1 | Valid | |

Table 3.81: ApplicationDataType DataValidityType

#### 3.2.2.8 ApplicationDataTypes, Category VALUE

| Common ApplicationDataType attributes | | | | |
|---|---|---|---|---|
| shortName | K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16 | | | |
| category | VALUE | | | |
| desc | Temperature | | | |
| swCalibrationAccess | notAccessible | | | |
| unit | K | | | |
| Range | | | | |
| | | lowerLimit | upperLimit | |
| | physConstrs | 0 | 511.9921875 | |
| Conversion | | | | |
| | category | LINEAR | | |
| | direction | compuInternalToPhys | | |
| | desc | lowerLimit | upperLimit | compuNumerator / compuDenominator |
| | - | - | - | $Phys = \frac{0 + 0.0078125 * Internal}{1}$ |

Table 3.82: ApplicationDataType K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16

| Common ApplicationDataType attributes | | | | |
|---|---|---|---|---|
| shortName | kg_Lnr_0_0d25_0_C8_uint8 | | | |
| category | VALUE | | | |
| desc | Mass | | | |
| swCalibrationAccess | notAccessible | | | |
| unit | kg | | | |
| Range | | | | |
| | | lowerLimit | upperLimit | |
| | physConstrs | 0 | 0.25 | |
| Conversion | | | | |
| | category | LINEAR | | |
| | direction | compuInternalToPhys | | |
| | desc | lowerLimit | upperLimit | compuNumerator / compuDenominator |
| | - | - | - | $Phys = \frac{0 + 0.00125 * Internal}{1}$ |

Table 3.83: ApplicationDataType kg_Lnr_0_0d25_0_C8_uint8

| Common ApplicationDataType attributes | | | |
|---|---|---|---|
| shortName | NoUnit_Lnr_1_4_1_4_uint8 | | |
| category | VALUE | | |
| swCalibrationAccess | notAccessible | | |
| unit | NoUnit | | |
| Range | | | |
| | lowerLimit | upperLimit | |
| physConstrs | 1 | 4 | |
| Conversion | | | |
| category | LINEAR | | |
| direction | compuInternalToPhys | | |
| desc | lowerLimit | upperLimit | compuNumerator / compuDenominator |
| - | - | - | $Phys = \frac{0 + 1 * Internal}{1}$ |

Table 3.84: ApplicationDataType NoUnit_Lnr_1_4_1_4_uint8

| Common ApplicationDataType attributes | | | |
|---|---|---|---|
| shortName | NoUnit_Lnr_1_65535_1_FFFF_uint16 | | |
| category | VALUE | | |
| swCalibrationAccess | notAccessible | | |
| unit | NoUnit | | |
| Range | | | |
| | lowerLimit | upperLimit | |
| physConstrs | 1 | 65535 | |
| Conversion | | | |
| category | LINEAR | | |
| direction | compuInternalToPhys | | |
| desc | lowerLimit | upperLimit | compuNumerator / compuDenominator |
| - | - | - | $Phys = \frac{0 + 1 * Internal}{1}$ |

Table 3.85: ApplicationDataType NoUnit_Lnr_1_65535_1_FFFF_uint16

| Common ApplicationDataType attributes | | | |
|---|---|---|---|
| shortName | s_Lnr_0_8191d875_0_FFFF_uint16 | | |
| category | VALUE | | |
| desc | cooking time in seconds | | |
| swCalibrationAccess | notAccessible | | |
| unit | s | | |
| Range | | | |
| | lowerLimit | upperLimit | |
| physConstrs | 0 | 8191.875 | |
| Conversion | | | |
| category | LINEAR | | |
| direction | compuInternalToPhys | | |
| desc | lowerLimit | upperLimit | compuNumerator / compuDenominator |
| - | - | - | $Phys = \frac{0 + 0.125 * Internal}{1}$ |

Table 3.86: ApplicationDataType s_Lnr_0_8191d875_0_FFFF_uint16

| Common ApplicationDataType attributes | | | | |
|---|---|---|---|---|
| shortName | speedSteps | | | |
| category | VALUE | | | |
| desc | Possible speed steps | | | |
| swCalibrationAccess | notAccessible | | | |
| unit | NoUnit | | | |
| Range | | | | |
| | lowerLimit | upperLimit | | |
| physConstrs | -1 | 2 | | |
| Conversion | | | | |
| category | TEXTTABLE | | | |
| direction | compuInternalToPhys | | | |
| desc | lowerLimit | upperLimit | vt | symbol |
| - | -1 | -1 | Stop | |
| - | 0 | 0 | LightSpeed | |
| - | 1 | 1 | RidiculousSpeed | |
| - | 2 | 2 | LudicrousSpeed | |

Table 3.87: ApplicationDataType speedSteps

| Common ApplicationDataType attributes | | | | |
|---|---|---|---|---|
| shortName | TxWheelNames | | | |
| category | VALUE | | | |
| desc | Wheel names | | | |
| swCalibrationAccess | notAccessible | | | |
| unit | NoUnit | | | |
| Range | | | | |
| | lowerLimit | | upperLimit | |
| physConstrs | 0 | | 3 | |
| Conversion | | | | |
| category | TEXTTABLE | | | |
| direction | compuInternalToPhys | | | |
| desc | lowerLimit | upperLimit | vt | symbol |
| - | 0 | 0 | FrontLeft | |
| - | 1 | 1 | FrontRight | |
| - | 2 | 2 | RearLeft | |
| - | 3 | 3 | RearRight | |

**Table 3.88: ApplicationDataType TxWheelNames**

#### 3.2.2.9 ApplicationDataTypes, Category COM_AXIS

| Common ApplicationDataType attributes | |
|---|---|
| shortName | ComAxis_Temp_Lin_K_uint16 |
| category | COM_AXIS |
| swCalibrationAccess | notAccessible |
| swRecordLayout | RL20_ME_Axis |
| properties of the axes (swCalprmAxisSet) | |
| properties of SwAxisIndividual (swCalprmAxis and swCalprmAxisTypeProps) | |
| swAxisIndex | 1 |
| category | COM_AXIS |
| inputVariableType | K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16 |
| swMaxAxisPoints | 6 |
| swMinAxisPoints | 6 |

Table 3.89: ApplicationDataType ComAxis_Temp_Lin_K_uint16

| Common ApplicationDataType attributes | |
|---|---|
| shortName | ComAxis_Steps_Txt_sint8 |
| category | COM_AXIS |
| swCalibrationAccess | notAccessible |
| swRecordLayout | RL20_ME_Axis |
| properties of the axes (swCalprmAxisSet) | |
| properties of SwAxisIndividual (swCalprmAxis and swCalprmAxisTypeProps) | |
| swAxisIndex | 1 |
| category | COM_AXIS |
| inputVariableType | speedSteps |
| swMaxAxisPoints | 4 |
| swMinAxisPoints | 4 |

Table 3.90: ApplicationDataType ComAxis_Steps_Txt_sint8

| Common ApplicationDataType attributes | |
|---|---|
| shortName | ComAxis_Mass_Lnr_Kg_uint8 |
| category | COM_AXIS |
| swCalibrationAccess | notAccessible |
| swRecordLayout | RL20_ME_Axis |
| properties of the axes (swCalprmAxisSet) | |
| properties of SwAxisIndividual (swCalprmAxis and swCalprmAxisTypeProps) | |
| swAxisIndex | 1 |
| category | COM_AXIS |
| inputVariableType | kg_Lnr_0_0d25_0_C8_uint8 |
| swMaxAxisPoints | 4 |
| swMinAxisPoints | 4 |

**Table 3.91: ApplicationDataType ComAxis_Mass_Lnr_Kg_uint8**

#### 3.2.2.10 ApplicationDataTypes, Category CURVE

| Common ApplicationDataType attributes | |
|---|---|
| shortName | Curve_Mass_Lnr_Kg_uint8 |
| category | CURVE |
| swCalibrationAccess | notAccessible |
| swRecordLayout | RL20_ME_1DimMap |
| valueAxisDataType | kg_Lnr_0_0d25_0_C8_uint8 |
| properties of the axes (swCalprmAxisSet) | |
| properties of SwAxisGrouped (swCalprmAxis and swCalprmAxisTypeProps) | |
| swAxisIndex | 1 |
| category | COM_AXIS |
| sharedAxisType | ComAxis_Steps_Txt_sint8 |

**Table 3.92: ApplicationDataType Curve_Mass_Lnr_Kg_uint8**

#### 3.2.2.11 ApplicationDataTypes, Category MAP

| Common ApplicationDataType attributes | |
|---|---|
| shortName | Map_Time_Lnr_s_uint16 |
| category | MAP |
| swCalibrationAccess | notAccessible |
| swRecordLayout | RL20_ME_2DimMap |
| valueAxisDataType | s_Lnr_0_8191d875_0_FFFF_uint16 |
| properties of the axes (swCalprmAxisSet) | |
| properties of SwAxisGrouped (swCalprmAxis and swCalprmAxisTypeProps) | |
| swAxisIndex | 1 |
| category | COM_AXIS |
| sharedAxisType | ComAxis_Temp_Lin_K_uint16 |
| properties of SwAxisGrouped (swCalprmAxis and swCalprmAxisTypeProps) | |
| swAxisIndex | 2 |
| category | COM_AXIS |
| sharedAxisType | ComAxis_Mass_Lnr_Kg_uint8 |

**Table 3.93: ApplicationDataType Map_Time_Lnr_s_uint16**

#### 3.2.2.12 ApplicationArrayDataTypes

| Common ApplicationArrayDataType attributes | |
|---|---|
| shortName | ComAxis_Temp_Lin_K_uint16_ScNoOfWheels |
| category | ARRAY |
| swCalibrationAccess | notAccessible |
| properties of the elements | |
| properties of ApplicationArrayElement | |
| shortName | ComAxis_Temp_Lin_K_uint16_ScNoOfWheels |
| category | COM_AXIS |
| type | ComAxis_Temp_Lin_K_uint16 |
| arraySizeSemantics | fixedSize |
| maxNumberOfElements | |

**Table 3.94: ApplicationArrayDataType ComAxis_Temp_Lin_K_uint16_ScNoOfWheels**

| Common ApplicationArrayDataType attributes | |
|---|---|
| shortName | K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16_ScNoOfWheels |
| category | ARRAY |
| swCalibrationAccess | notAccessible |
| properties of the elements | |
| properties of ApplicationArrayElement | |
| shortName | K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16_ScNoOfWheels |
| category | VALUE |
| type | K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16 |
| arraySizeSemantics | fixedSize |
| maxNumberOfElements | |

**Table 3.95: ApplicationArrayDataType K_Celsius_Lnr_0_511d9921875_0_FFFF_uint16_ScNoOfWheels**

| Common ApplicationArrayDataType attributes | |
|---|---|
| shortName | Map_Time_Lnr_s_uint16_ScNoOfWheels |
| category | ARRAY |
| swCalibrationAccess | notAccessible |
| properties of the elements | |
| properties of ApplicationArrayElement | |
| shortName | Map_Time_Lnr_s_uint16_ScNoOfWheels |
| category | MAP |
| type | Map_Time_Lnr_s_uint16 |
| arraySizeSemantics | fixedSize |
| maxNumberOfElements | |

**Table 3.96: ApplicationArrayDataType Map_Time_Lnr_s_uint16_ScNoOfWheels**

#### 3.2.2.13 ApplicationRecordDataTypes

| Common ApplicationRecordDataType attributes | |
|---|---|
| shortName | CMscC_nvm_NvBlockATyp |
| category | STRUCTURE |
| swCalibrationAccess | notAccessible |
| properties of the elements | |
| properties of ApplicationRecordElement | |
| shortName | PrimData_StepsSpeed_Txt_sint8 |
| category | VALUE |
| type | speedSteps |

**Table 3.97: ApplicationRecordDataType CMscC_nvm_NvBlockATyp**

#### 3.2.2.14 ModeDeclarationGroups

| Common ModeDeclarationGroup attributes | |
|---|---|
| shortName | Direction |
| category | EXPLICIT_ORDER |
| initialMode | Halt |
| **properties of the modeDeclarations** | |
| **properties of ModeDeclaration** | |
| shortName | Backward |
| desc | Backward direction |
| value | 2 |
| **properties of ModeDeclaration** | |
| shortName | Forward |
| desc | Forward direction |
| value | 1 |
| **properties of ModeDeclaration** | |
| shortName | Halt |
| desc | Standstill |
| value | 0 |

**Table 3.98: ModeDeclarationGroup Direction**

#### 3.2.2.15 Units

| Common Unit attributes | |
|---|---|
| shortName | Celsius |
| desc | Degrees Celsius |
| displayName | °C |
| offsetSiToUnit | -273.15 |
| factorSiToUnit | 1.0 |
| physicalDimension | PD_K |

**Table 3.99: Unit Celsius**

| Common Unit attributes | |
|---|---|
| shortName | K |
| desc | Temperature |
| displayName | K |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 1.0 |
| physicalDimension | PD_K |

**Table 3.100: Unit K**

| Common Unit attributes | |
|---|---|
| shortName | kg |
| desc | Mass |
| displayName | kg |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 1.0 |
| physicalDimension | PD_kg |

**Table 3.101: Unit kg**

| Common Unit attributes | |
|---|---|
| shortName | NoUnit |
| desc | No Unit |
| displayName | - |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 1.0 |
| physicalDimension | PD_NoUnit |

**Table 3.102: Unit NoUnit**

| Common Unit attributes | |
|---|---|
| shortName | s |
| desc | Time |
| displayName | s |
| offsetSiToUnit | 0.0 |
| factorSiToUnit | 1.0 |
| physicalDimension | PD_s |

**Table 3.103: Unit s**

#### 3.2.2.16 PhysicalDimensions

| Common PhysicalDimension attributes | |
|---|---|
| shortName | PD_K |
| currentExp | 0 |
| lengthExp | 0 |
| luminousIntensity-Exp | 0 |
| massExp | 0 |
| molarAmountExp | 0 |
| temperatureExp | 1 |
| timeExp | 0 |

**Table 3.104: PhysicalDimension PD_K**

| Common PhysicalDimension attributes | |
|---|---|
| shortName | PD_kg |
| currentExp | 0 |
| lengthExp | 0 |
| luminousIntensity-Exp | 0 |
| massExp | 1 |
| molarAmountExp | 0 |
| temperatureExp | 0 |
| timeExp | 0 |

**Table 3.105: PhysicalDimension PD_kg**

| Common PhysicalDimension attributes | |
|---|---|
| shortName | PD_NoUnit |
| currentExp | 0 |
| lengthExp | 0 |
| luminousIntensity-Exp | 0 |
| massExp | 0 |
| molarAmountExp | 0 |
| temperatureExp | 0 |
| timeExp | 0 |

**Table 3.106: PhysicalDimension PD_NoUnit**

| Common PhysicalDimension attributes | |
|---|---|
| shortName | PD_s |
| currentExp | 0 |
| lengthExp | 0 |
| luminousIntensity-Exp | 0 |
| massExp | 0 |
| molarAmountExp | 0 |
| temperatureExp | 0 |
| timeExp | 1 |

**Table 3.107: PhysicalDimension PD_s**

#### 3.2.2.17 SwAddrMethods

| Common SwAddrMethod attributes | |
|---|---|
| shortName | CAL |
| desc | Calibratable constants; safety level QM. Constants will be located in different memory sections depending on the alignment of the constant. |
| sectionType | calprm |
| memoryAllocation-KeywordPolicy | addrMethodShortNameAndAlignment |
| sectionInitializa-tionPolicy | - |
| option | safetyQM |

**Table 3.108: SwAddrMethod CAL**

| Common SwAddrMethod attributes | |
|---|---|
| shortName | CODE_10MS |
| desc | Code of ECU-functions called every 10 ms; safety level QM. |
| sectionType | code |
| memoryAllocation-KeywordPolicy | addrMethodShortName |
| sectionInitializa-tionPolicy | - |
| option | safetyQM |

**Table 3.109: SwAddrMethod CODE_10MS**

| Common SwAddrMethod attributes | |
|---|---|
| shortName | CONST_SLOW |
| desc | Non calibratable constants of ECU-functions called seldom; safety level QM. |
| sectionType | const |
| memoryAllocation-KeywordPolicy | addrMethodShortName |
| sectionInitializa-tionPolicy | - |
| option | safetyQM |

**Table 3.110: SwAddrMethod CONST_SLOW**

| Common SwAddrMethod attributes | |
|---|---|
| shortName | DATA |
| desc | Variables of ECU-functions; safety level QM. Variables will be located in different memory sections depending on the alignment of the variable. |
| sectionType | var |
| memoryAllocation-KeywordPolicy | addrMethodShortNameAndAlignment |
| sectionInitializa-tionPolicy | INIT |
| option | safetyQM |

**Table 3.111: SwAddrMethod DATA**

| Common SwAddrMethod attributes | |
|---|---|
| shortName | DATA_NVDAT |
| desc | Variables stored in non-volatile memory; safety level QM. |
| sectionType | var |
| memoryAllocation-KeywordPolicy | addrMethodShortName |
| sectionInitializa-tionPolicy | NO-INIT |
| option | nvData, safetyQM |

**Table 3.112: SwAddrMethod DATA_NVDAT**

# A Mentioned Class Tables

For the sake of completeness, this chapter contains a set of class tables representing meta-classes mentioned in the context of this document but which are not contained directly in the scope of describing specific meta-model semantics.

| **Class** | **ARElement (abstract)** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage | | | |
| **Note** | An element that can be defined stand-alone, i.e. without being part of another element (except for packages of course). | | | |
| **Base** | ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| – | – | – | – | – |

**Table A.1: ARElement**

| **Class** | **ARPackage** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::ARPackage | | | |
| **Note** | AUTOSAR package, allowing to create top level packages to structure the contained ARElements.<br><br>ARPackages are open sets. This means that in a file based description system multiple files can be used to partially describe the contents of a package.<br><br>This is an extended version of MSR's SW-SYSTEM. | | | |
| **Base** | ARObject, AtpBlueprint, AtpBlueprintable, CollectableElement, Identifiable, MultilanguageReferrable, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| arPackage | ARPackage | * | aggr | This represents a sub package within an ARPackage, thus allowing for an unlimited package hierarchy.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=blueprintDerivationTime<br>xml.sequenceOffset=30 |
| element | PackageableElement | * | aggr | Elements that are part of this package<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=systemDesignTime<br>xml.sequenceOffset=20 |

| | | | | |
|---|---|---|---|---|
| referenceBase | ReferenceBase | * | aggr | This denotes the reference bases for the package. This is the basis for all relative references within the package. The base needs to be selected according to the base attribute within the references.<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=shortLabel<br>xml.sequenceOffset=10 |

**Table A.2: ARPackage**

| *Class* | **AliasNameSet** | | | |
|---|---|---|---|---|
| *Package* | M2::AUTOSARTemplates::CommonStructure::FlatMap | | | |
| *Note* | This meta-class represents a set of AliasNames. The AliasNameSet can for example be an input to the A2L-Generator. It shall not be used by the RTE generator to generate the MC-Support.<br><br>In a given instance of AliasNameSet in the bound system there must be at most one aliasName per FlatInstanceDescriptor.<br><br>**Tags:** atp.recommendedPackage=AliasNameSets | | | |
| *Base* | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| *Attribute* | *Type* | *Mul.* | *Kind* | *Note* |
| aliasName | AliasNameAssignment | 1..* | aggr | AliasNames contained in the AliasNameSet.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortLabel<br>vh.latestBindingTime=preCompileTime |

**Table A.3: AliasNameSet**

| *Class* | **AnyInstanceRef** | | | |
|---|---|---|---|---|
| *Package* | M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::AnyInstanceRef | | | |
| *Note* | Describes a reference to any instance in an AUTOSAR model. This is the most generic form of an instance ref. Refer to the superclass notes for more details. | | | |
| *Base* | ARObject,AtpInstanceRef | | | |
| *Attribute* | *Type* | *Mul.* | *Kind* | *Note* |
| base | AtpClassifier | 1 | ref | This is the base from which navigation path begins.<br><br>**Stereotypes:** atpDerived |
| contextElement | AtpFeature | * | ref | This is one step in the navigation path specified by the instance ref. |
| target | AtpFeature | 1 | ref | This is the target of the instance ref. |

**Table A.4: AnyInstanceRef**

| **Class** | **ApplicationArrayDataType** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| **Note** | An application data type which is an array, each element is of the same application data type.<br><br>**Tags:** atp.recommendedPackage=ApplicationDataTypes | | | |
| **Base** | ARElement, ARObject, ApplicationCompositeDataType, ApplicationDataType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| dynamicArraySizeProfile | String | 0..1 | attr | Specifies the profile which the array will follow if it is a variable size array. |
| element | ApplicationArrayElement | 1 | aggr | This association implements the concept of an array element. That is, in some cases it is necessary to be able to identify single array elements, e.g. as input values for an interpolation routine. |

**Table A.5: ApplicationArrayDataType**

| **Class** | **ApplicationArrayElement** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes | | | |
| **Note** | Describes the properties of the elements of an application array data type. | | | |
| **Base** | ARObject, ApplicationCompositeElementDataPrototype, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| arraySizeHandling | ArraySizeHandlingEnum | 0..1 | attr | The way how the size of the array is handled. |
| arraySizeSemantics | ArraySizeSemanticsEnum | 0..1 | attr | This attribute controls how the information about the array size shall be interpreted. |
| indexDataType | ApplicationPrimitiveDataType | 0..1 | ref | This reference can be taken to assign a CompuMethod of category TEXTTABLE to the array. The texttable entries associate a textual value to an index number such that the element with that index number is represented by a symbolic name. |
| maxNumberOfElements | PositiveInteger | 0..1 | attr | The maximum number of elements that the array can contain.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime |

**Table A.6: ApplicationArrayElement**

| | | | | |
|---|---|---|---|---|
| ***Class*** | **ApplicationCompositeDataType (abstract)** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| ***Note*** | Abstract base class for all application data types composed of other data types. | | | |
| ***Base*** | ARElement, ARObject, ApplicationDataType, AtpBlueprint, AtpBlueprintable, Atp Classifier, AtpType, AutosarDataType, CollectableElement, Identifiable, Multilanguage Referrable, PackageableElement, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table A.7: ApplicationCompositeDataType**

| | | | | |
|---|---|---|---|---|
| ***Class*** | **ApplicationCompositeElementDataPrototype (abstract)** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes | | | |
| ***Note*** | This class represents a data prototype which is aggregated within a composite application data type (record or array). It is introduced to provide a better distinction between target and context in instanceRefs. | | | |
| ***Base*** | ARObject, AtpFeature, AtpPrototype, DataPrototype, Identifiable, Multilanguage Referrable, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| type | ApplicationData Type | 1 | tref | This represents the corresponding data type.<br><br>**Stereotypes:** isOfType |

**Table A.8: ApplicationCompositeElementDataPrototype**

| | | | | |
|---|---|---|---|---|
| ***Class*** | **ApplicationDataType (abstract)** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| ***Note*** | ApplicationDataType defines a data type from the application point of view. Especially it should be used whenever something "physical" is at stake.<br><br>An ApplicationDataType represents a set of values as seen in the application model, such as measurement units. It does not consider implementation details such as bit-size, endianess, etc.<br><br>It should be possible to model the application level aspects of a VFB system by using ApplicationDataTypes only. | | | |
| ***Base*** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table A.9: ApplicationDataType**

| *Class* | **ApplicationPrimitiveDataType** | | | |
|---|---|---|---|---|
| *Package* | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| *Note* | A primitive data type defines a set of allowed values.<br><br>**Tags:** atp.recommendedPackage=ApplicationDataTypes | | | |
| *Base* | ARElement, ARObject, ApplicationDataType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| *Attribute* | *Type* | *Mul.* | *Kind* | *Note* |
| – | – | – | – | – |

**Table A.10: ApplicationPrimitiveDataType**

| *Class* | **ApplicationRecordDataType** | | | |
|---|---|---|---|---|
| *Package* | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| *Note* | An application data type which can be decomposed into prototypes of other application data types.<br><br>**Tags:** atp.recommendedPackage=ApplicationDataTypes | | | |
| *Base* | ARElement, ARObject, ApplicationCompositeDataType, ApplicationDataType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| *Attribute* | *Type* | *Mul.* | *Kind* | *Note* |
| element (ordered) | ApplicationRecordElement | 1..* | aggr | Specifies an element of a record.<br><br>The aggregation of ApplicationRecordElement is subject to variability with the purpose to support the conditional existence of elements inside a ApplicationrecordDataType.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime |

**Table A.11: ApplicationRecordDataType**

| *Class* | **ApplicationRecordElement** | | | |
|---|---|---|---|---|
| *Package* | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes | | | |
| *Note* | Describes the properties of one particular element of an application record data type. | | | |
| *Base* | ARObject, ApplicationCompositeElementDataPrototype, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable | | | |
| *Attribute* | *Type* | *Mul.* | *Kind* | *Note* |
| – | – | – | – | – |

**Table A.12: ApplicationRecordElement**

| **Class** | **ApplicationSwComponentType** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::Components | | | |
| **Note** | The ApplicationSwComponentType is used to represent the application software.<br><br>**Tags:** atp.recommendedPackage=SwComponentTypes | | | |
| **Base** | ARElement, ARObject, AtomicSwComponentType, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| – | – | – | – | – |

**Table A.13: ApplicationSwComponentType**

| **Enumeration** | **ArraySizeSemanticsEnum** |
|---|---|
| **Package** | M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes |
| **Note** | This type controls how the information about the number of elements in an ApplicationArrayDataType is to be interpreted. |
| **Literal** | **Description** |
| fixedSize | This means that the ApplicationArrayDataType will always have a fixed number of elements.<br><br>**Tags:** atp.EnumerationValue=0 |
| variableSize | This implies that the actual number of elements in the ApplicationArrayDataType might vary at run-time. The value of arraySize represents the maximum number of elements in the array.<br><br>**Tags:** atp.EnumerationValue=1 |

**Table A.14: ArraySizeSemanticsEnum**

| **Class** | **AssemblySwConnector** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::Composition | | | |
| **Note** | AssemblySwConnectors are exclusively used to connect SwComponentPrototypes in the context of a CompositionSwComponentType. | | | |
| **Base** | ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable, SwConnector | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| provider | AbstractProvidedPortPrototype | 0..1 | iref | Instance of providing port. |
| requester | AbstractRequiredPortPrototype | 0..1 | iref | Instance of requiring port. |

**Table A.15: AssemblySwConnector**

| **Class** | **AtomicSwComponentType (abstract)** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::Components | | | |
| **Note** | An atomic software component is atomic in the sense that it cannot be further decomposed and distributed across multiple ECUs. | | | |
| **Base** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| internalBehavior | SwcInternalBehavior | 0..1 | aggr | The SwcInternalBehaviors owned by an AtomicSwComponentType can be located in a different physical file. Therefore the aggregation is «atpSplitable».<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=internalBehavior, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| symbolProps | SymbolProps | 0..1 | aggr | This represents the SymbolProps for the AtomicSwComponentType.<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=shortName |

**Table A.16: AtomicSwComponentType**

| **Class** | **AutosarDataPrototype (abstract)** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes | | | |
| **Note** | Base class for prototypical roles of an AutosarDataType. | | | |
| **Base** | ARObject, AtpFeature, AtpPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| type | AutosarDataType | 1 | tref | This represents the corresponding data type.<br><br>**Stereotypes:** isOfType |

**Table A.17: AutosarDataPrototype**

| **Class** | **CompositionSwComponentType** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::Composition | | | |
| **Note** | A CompositionSwComponentType aggregates SwComponentPrototypes (that in turn are typed by SwComponentTypes) as well as SwConnectors for primarily connecting SwComponentPrototypes among each others and towards the surface of the CompositionSwComponentType. By this means hierarchical structures of software-components can be created.<br><br>**Tags:** atp.recommendedPackage=SwComponentTypes | | | |
| **Base** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |

| | | | | |
|---|---|---|---|---|
| component | SwComponentPrototype | * | aggr | The instantiated components that are part of this composition. The aggregation of SwComponentPrototype is subject to variability with the purpose to support the conditional existence of a SwComponentPrototype. Please be aware: if the conditional existence of SwComponentPrototypes is resolved post-build the deselected SwComponentPrototypes are still contained in the ECUs build but the instances are inactive in in that they are not scheduled by the RTE.<br><br>The aggregation is marked as atpSplitable in order to allow the addition of service components to the ECU extract during the ECU integration.<br><br>The use case for having 0 components owned by the CompositionSwComponentType could be to deliver an empty CompositionSwComponentType to e.g. a supplier for filling the internal structure.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=postBuild |
| connector | SwConnector | * | aggr | SwConnectors have the principal ability to establish a connection among PortPrototypes. They can have many roles in the context of a CompositionSwComponentType. Details are refined by subclasses.<br><br>The aggregation of SwConnectors is subject to variability with the purpose to support variant data flow.<br><br>The aggregation is marked as atpSplitable in order to allow the extension of the ECU extract with AssemblySwConnectors between ApplicationSwComponentTypes and ServiceSwComponentTypes during the ECU integration.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=postBuild |
| constantValueMapping | ConstantSpecificationMappingSet | * | ref | Reference to the ConstantSpecificationMapping to be applied for initValues of PPortComSpecs and RPortComSpec.<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=constantValueMapping |

| | | | | |
|---|---|---|---|---|
| dataTypeMapping | DataTypeMappingSet | * | ref | Reference to the DataTypeMapping to be applied for the used ApplicationDataTypes in PortInterfaces.<br><br>Background: when developing subsystems it may happen that ApplicationDataTypes are used on the surface of CompositionSwComponentTypes. In this case it would be reasonable to be able to also provide the intended mapping to the ImplementationDataTypes. However, this mapping shall be informal and not technically binding for the implementers mainly because the RTE generator is not concerned about the CompositionSwComponentTypes.<br><br>Rationale: if the mapping of ApplicationDataTypes on the delegated and inner PortPrototype matches then the mapping to ImplementationDataTypes is not impacting compatibility.<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=dataTypeMapping |
| instantiationRTEEventProps | InstantiationRTEEventProps | * | aggr | This allows to define instantiation specific properties for RTE Events, in particular for instance specific scheduling.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortLabel, variationPoint.shortLabel<br>vh.latestBindingTime=codeGenerationTime |

**Table A.18: CompositionSwComponentType**

| ***Class*** | **CompuConstTextContent** | | | |
|---|---|---|---|---|
| ***Package*** | M2::MSR::AsamHdo::ComputationMethod | | | |
| ***Note*** | This meta-class represents the textual content of a scale. | | | |
| ***Base*** | ARObject, CompuConstContent | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| vt | VerbatimString | 1 | attr | This represents a textual constant in the computation method. |

**Table A.19: CompuConstTextContent**

| **Class** | **CompuMethod** | | | |
|---|---|---|---|---|
| **Package** | M2::MSR::AsamHdo::ComputationMethod | | | |
| **Note** | This meta-class represents the ability to express the relationship between a physical value and the mathematical representation.<br><br>Note that this is still independent of the technical implementation in data types. It only specifies the formula how the internal value corresponds to its physical pendant.<br><br>**Tags:** atp.recommendedPackage=CompuMethods | | | |
| **Base** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| compuInternalToPhys | Compu | 0..1 | aggr | This specifies the computation from internal values to physical values.<br><br>**Tags:** xml.sequenceOffset=80 |
| compuPhysToInternal | Compu | 0..1 | aggr | This represents the computation from physical values to the internal values.<br><br>**Tags:** xml.sequenceOffset=90 |
| displayFormat | DisplayFormatString | 0..1 | attr | This property specifies, how the physical value shall be displayed e.g. in documents or measurement and calibration tools.<br><br>**Tags:** xml.sequenceOffset=20 |
| unit | Unit | 0..1 | ref | This is the physical unit of the Physical values for which the CompuMethod applies.<br><br>**Tags:** xml.sequenceOffset=30 |

**Table A.20: CompuMethod**

| **Class** | **CompuRationalCoeffs** | | | |
|---|---|---|---|---|
| **Package** | M2::MSR::AsamHdo::ComputationMethod | | | |
| **Note** | This meta-class represents the ability to express a rational function by specifying the coefficients of nominator and denominator. | | | |
| **Base** | ARObject | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| compuDenominator | CompuNominatorDenominator | 1 | aggr | This is the denominator of the expression.<br><br>**Tags:** xml.sequenceOffset=30 |
| compuNumerator | CompuNominatorDenominator | 1 | aggr | This is the numerator of the rational expression.<br><br>**Tags:** xml.sequenceOffset=20 |

**Table A.21: CompuRationalCoeffs**

| **Class** | **CompuScale** | | | |
|---|---|---|---|---|
| **Package** | M2::MSR::AsamHdo::ComputationMethod | | | |
| **Note** | This meta-class represents the ability to specify one segment of a segmented computation method. | | | |
| **Base** | ARObject | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| desc | MultiLanguageOverviewParagraph | 0..1 | aggr | <desc> represents a general but brief description of the object in question.<br><br>**Tags:** xml.sequenceOffset=30 |
| compuInverseValue | CompuConst | 0..1 | aggr | This is the inverse value of the constraint. This supports the case that the scale is not reversible per se.<br><br>**Tags:** xml.sequenceOffset=60 |
| compuScaleContents | CompuScaleContents | 0..1 | aggr | This represents the computation details of the scale.<br><br>**Tags:** xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=70; xml.typeElement=false; xml.typeWrapperElement=false |
| lowerLimit | Limit | 0..1 | attr | This specifies the lower limit of the scale.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime<br>xml.sequenceOffset=40 |
| mask | PositiveInteger | 0..1 | attr | In difference to all the other computational methods every COMPU-SCALE will be applied including the bit MASK. Therefore it is allowed for this type of COMPU-METHOD, that COMPU-SCALES overlap.<br><br>To calculate the string reverse to a value, the string has to be split and the according value for each substring has to be summed up. The sum is finally transmitted.<br><br>The processing has to be done in order of the COMPU-SCALE elements.<br><br>**Tags:** xml.sequenceOffset=35 |
| shortLabel | Identifier | 0..1 | attr | This element specifies a short name for the particular scale. The name can for example be used to derive a programming language identifier.<br><br>**Tags:** xml.sequenceOffset=20 |
| symbol | CIdentifier | 0..1 | attr | The symbol, if provided, is used by code generators to get a C identifier for the CompuScale. The name will be used as is for the code generation, therefore it needs to be unique within the generation context.<br><br>**Tags:** xml.sequenceOffset=25 |

| | | | | |
|---|---|---|---|---|
| upperLimit | Limit | 0..1 | attr | This specifies the upper limit of a of the scale.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime<br>xml.sequenceOffset=50 |

**Table A.22: CompuScale**

| ***Class*** | **DataConstr** | | | |
|---|---|---|---|---|
| ***Package*** | M2::MSR::AsamHdo::Constraints::GlobalConstraints | | | |
| ***Note*** | This meta-class represents the ability to specify constraints on data.<br><br>**Tags:** atp.recommendedPackage=DataConstrs | | | |
| ***Base*** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| dataConstrRule | DataConstrRule | * | aggr | This is one particular rule within the data constraints.<br><br>**Tags:** xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=30; xml.typeElement=false; xml.typeWrapperElement=false |

**Table A.23: DataConstr**

| ***Class*** | **DataConstrRule** | | | |
|---|---|---|---|---|
| ***Package*** | M2::MSR::AsamHdo::Constraints::GlobalConstraints | | | |
| ***Note*** | This meta-class represents the ability to express one specific data constraint rule. | | | |
| ***Base*** | ARObject | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| constrLevel | Integer | 0..1 | attr | This attribute describes the category of a constraint. One of its functions is in the area of constraint violation, where it can be used from a certain level, to produce error messages.<br><br>The lower the level, the more stringent the check.<br><br>Used to distinguish hard or soft limits.<br><br>**Tags:** xml.sequenceOffset=20 |
| internalConstrs | InternalConstrs | 0..1 | aggr | Describes the limitations applicable on the internal domain (as opposed to the physical domain).<br><br>**Tags:** xml.sequenceOffset=40 |
| physConstrs | PhysConstrs | 0..1 | aggr | Describes the limitations applicable on the physical domain (as opposed to the internal domain).<br><br>**Tags:** xml.sequenceOffset=30 |

**Table A.24: DataConstrRule**

| **Class** | **DataPrototype (abstract)** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes | | | |
| **Note** | Base class for prototypical roles of any data type. | | | |
| **Base** | ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| swDataDefProps | SwDataDefProps | 0..1 | aggr | This property allows to specify data definition properties which apply on data prototype level. |

**Table A.25: DataPrototype**

| **Class** | **DataTypeMap** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| **Note** | This class represents the relationship between ApplicationDataType and its implementing ImplementationDataType. | | | |
| **Base** | ARObject | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| applicationDataType | ApplicationDataType | 1 | ref | This is the corresponding ApplicationDataType |
| implementationDataType | ImplementationDataType | 1 | ref | This is the corresponding ImplementationDataType. |

**Table A.26: DataTypeMap**

| **Class** | **DataTypeMappingSet** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::Datatypes | | | |
| **Note** | This class represents a list of mappings between ApplicationDataTypes and ImplementationDataTypes. In addition, it can contain mappings between ImplementationDataTypes and ModeDeclarationGroups.<br><br>**Tags:** atp.recommendedPackage=DataTypeMappingSets | | | |
| **Base** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| dataTypeMap | DataTypeMap | * | aggr | This is one particular association between an ApplicationDataType and its ImplementationDataType. |
| modeRequestTypeMap | ModeRequestTypeMap | * | aggr | This is one particular association between an ModeDeclarationGroup and its ImplementationDataType. |

**Table A.27: DataTypeMappingSet**

| Class | EcuInstance | | | |
|---|---|---|---|---|
| Package | M2::AUTOSARTemplates::SystemTemplate::Fibex::FibexCore::CoreTopology | | | |
| Note | ECUInstances are used to define the ECUs used in the topology. The type of the ECU is defined by a reference to an ECU specified with the ECU resource description.<br><br>**Tags:** atp.recommendedPackage=EcuInstances | | | |
| Base | ARObject, CollectableElement, FibexElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| associatedComIPduGroup | ISignalIPduGroup | * | ref | With this reference it is possible to identify which ISignalIPduGroups are applicable for which CommunicationConnector/ ECU.<br><br>Only top level ISignalIPduGroups shall be referenced by an EcuInstance. If an ISignalIPduGroup contains other ISignalIPduGroups than these contained ISignalIPduGroups shall not be referenced by the EcuInstance. Contained ISignalIPduGroups are associated to an EcuInstance via the top level ISignalIPduGroup. |
| associatedPdurIPduGroup | PdurIPduGroup | * | ref | With this reference it is possible to identify which PduR IPdu Groups are applicable for which CommunicationConnector/ ECU. |
| clientIdRange | ClientIdRange | 0..1 | aggr | Restriction of the Client Identifier for this Ecu to an allowed range of numerical values. The Client Identifier of the transaction handle is generated by the client RTE for inter-Ecu Client/Server communication. |
| comConfigurationGwTimeBase | TimeValue | 0..1 | attr | The period between successive calls to Com_MainFunctionRouteSignals of the AUTOSAR COM module in seconds. |
| comConfigurationRxTimeBase | TimeValue | 0..1 | attr | The period between successive calls to Com_MainFunctionRx of the AUTOSAR COM module in seconds. |
| comConfigurationTxTimeBase | TimeValue | 0..1 | attr | The period between successive calls to Com_MainFunctionTx of the AUTOSAR COM module in seconds. |
| comEnableMDTForCyclicTransmission | Boolean | 0..1 | attr | Enables for the Com module of this EcuInstance the minimum delay time monitoring for cyclic and repeated transmissions (TransmissionModeTiming has cyclicTiming assigned or eventControlledTiming with numberOfRepetitions > 0). |
| commController | CommunicationController | 1..* | aggr | CommunicationControllers of the ECU. |
| connector | CommunicationConnector | * | aggr | All channels controlled by a single controller. |
| diagnosticAddress | Integer | 0..1 | attr | An ECU specific ID for responses of diagnostic routines. |
| diagnosticProps | DiagnosticEcuProps | 0..1 | aggr | This represents the diagnostic-related properties of an entire ECU. |

| | | | | |
|---|---|---|---|---|
| ethSwitchPortGroupDerivation | Boolean | 0..1 | attr | Defines whether the derivation of SwitchPortGroups based on VLAN and/or CouplingPort.pncMapping shall be performed for this EcuInstance. If not defined the derivation shall not be done. |
| partition | EcuPartition | * | aggr | Optional definition of Partitions within an Ecu. |
| pnResetTime | TimeValue | 0..1 | attr | Specifies the runtime of the reset timer in seconds. This reset time is valid for the reset of PN requests in the EIRA and in the ERA. |
| pncPrepareSleepTimer | TimeValue | 0..1 | attr | Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP. |
| sleepModeSupported | Boolean | 1 | attr | Specifies whether the ECU instance may be put to a "low power mode"<br>• true: sleep mode is supported<br>• false: sleep mode is not supported<br><br>Note: This flag may only be set to "true" if the feature is supported by both hardware and basic software. |
| v2xSupported | V2xSupportEnum | 0..1 | attr | This attribute is used to control the existence of the V2X stack on the given EcuInstance. |
| wakeUpOverBusSupported | Boolean | 1 | attr | Driver support for wakeup over Bus. |

**Table A.28: EcuInstance**

| **Class** | **EcucModuleConfigurationValues** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::ECUCDescriptionTemplate | | | |
| **Note** | Head of the configuration of one Module. A Module can be a BSW module as well as the RTE and ECU Infrastructure.<br><br>As part of the BSW module description, the EcucModuleConfigurationValues element has two different roles:<br><br>The recommendedConfiguration contains parameter values recommended by the BSW module vendor.<br><br>The preconfiguredConfiguration contains values for those parameters which are fixed by the implementation and cannot be changed.<br><br>These two EcucModuleConfigurationValues are used when the base EcucModuleConfigurationValues (as part of the base ECU configuration) is created to fill parameters with initial values.<br><br>**Tags:** atp.recommendedPackage=EcucModuleConfigurationValuess | | | |
| **Base** | ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| container | EcucContainerValue | 1..* | aggr | Aggregates all containers that belong to this module configuration.<br><br>atpVariation: [RS_ECUC_00078]<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=definition, shortName, variationPoint.shortLabel<br>vh.latestBindingTime=postBuild<br>xml.sequenceOffset=10 |
| definition | EcucModuleDef | 1 | ref | Reference to the definition of this EcucModuleConfigurationValues element. Typically, this is a vendor specific module configuration.<br><br>**Tags:** xml.sequenceOffset=-10 |
| ecucDefEdition | RevisionLabelString | 1 | attr | This is the version info of the ModuleDef ECUC Parameter definition to which this values conform to / are based on.<br><br>For the Definition of ModuleDef ECUC Parameters the AdminData shall be used to express the semantic changes. The compatibility rules between the definition and value revision labels is up to the module's vendor. |
| implementationConfigVariant | EcucConfigurationVariantEnum | 1 | attr | Specifies the kind of deliverable this EcucModuleConfigurationValues element provides. If this element is not used in a particular role (e.g. preconfiguredConfiguration or recommendedConfiguration) then the value must be one of VariantPreCompile, VariantLinkTime, VariantPostBuild. |

| | | | | |
|---|---|---|---|---|
| moduleDescription | BswImplementation | 0..1 | ref | Referencing the BSW module description, which this EcucModuleConfigurationValues element is configuring. This is optional because the EcucModuleConfigurationValues element is also used to configure the ECU infrastructure (memory map) or Application SW-Cs. However in case the EcucModuleConfigurationValues are used to configure the module, the reference is mandatory in order to fetch module specific "common" published information. |

**Table A.29: EcucModuleConfigurationValues**

| ***Class*** | **EcucValueCollection** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::ECUCDescriptionTemplate | | | |
| ***Note*** | This represents the anchor point of the ECU configuration description.<br><br>**Tags:** atp.recommendedPackage=EcucValueCollections | | | |
| ***Base*** | ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| ecuExtract | System | 1 | ref | Represents the extract of the System Configuration that is relevant for the ECU configured with that ECU Configuration Description. |
| ecucValue | EcucModuleConfigurationValues | 1..* | ref | References to the configuration of individual software modules that are present on this ECU.<br><br>atpVariation: [RS_ECUC_0079]<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime |

**Table A.30: EcucValueCollection**

| **Class** | **FlatInstanceDescriptor** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::CommonStructure::FlatMap | | | |
| **Note** | Represents exactly one node (e.g. a component instance or data element) of the instance tree of a software system. The purpose of this element is to map the various nested representations of this instance to a flat representation and assign a unique name (shortName) to it.<br><br>Use cases:<br>• Specify unique names of measurable data to be used by MCD tools<br>• Specify unique names of calibration data to be used by MCD tool<br>• Specify a unique name for an instance of a component prototype in the ECU extract of the system description<br><br>Note that in addition it is possible to assign alias names via AliasNameAssignment. | | | |
| **Base** | ARObject, Identifiable, MultilanguageReferrable, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| ecuExtractReference | AtpFeature | 0..1 | iref | Refers to the instance in the ECU extract. This is valid only, if the FlatMap is used in the context of an ECU extract.<br><br>The reference shall be such that it uniquely defines the object instance. For example, if a data prototype is declared as a role within an SwcInternalBehavior, it is not enough to state the SwcInternalBehavior as context and the aggregated data prototype as target. In addition, the reference shall also include the complete path identifying instance of the component prototype and the AtomicSoftwareComponentType, which is refered by the particular SwcInternalBehavior.<br><br>**Tags:** xml.sequenceOffset=40 |
| role | Identifier | 0..1 | attr | The role denotes the particular role of the downstream memory location described by this FlatInstanceDescriptor.<br><br>It applies to use case where one upstream object results in multiple downstream objects, e.g. ModeDeclarationGroupPrototypes which are measurable. In this case the RTE will provide locations for current mode, previous mode and next mode. |
| swDataDefProps | SwDataDefProps | 0..1 | aggr | The properties of this FlatInstanceDescriptor. |

| | | | | |
|---|---|---|---|---|
| upstreamReference | AtpFeature | 0..1 | iref | Refers to the instance in the context of an "upstream" descriptions, wich could be the system or system extract description, the basic software module description or (if a flat map is used in preliminary context) a description of an atomic component or composition. This reference is optional in case the flat map is used in ECU context.<br><br>The reference shall be such that it uniquely defines the object instance in the given context. For example, if a data prototype is declared as a role within an SwcInternalBehavior, it is not enough to state the SwcInternalBehavior as context and the aggregated data prototype as target. In addition, the reference shall also include the complete path identifying the instance of the component prototype that contains the particular instance of SwcInternalBehavior.<br><br>**Tags:** xml.sequenceOffset=20 |

**Table A.31: FlatInstanceDescriptor**

| ***Class*** | **FlatMap** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::FlatMap | | | |
| ***Note*** | Contains a flat list of references to software objects. This list is used to identify instances and to resolve name conflicts. The scope is given by the RootSwCompositionPrototype for which it is used, i.e. it can be applied to a system, system extract or ECU-extract.<br><br>An instance of FlatMap may also be used in a preliminary context, e.g. in the scope of a software component before integration into a system. In this case it is not referred by a RootSwCompositionPrototype.<br><br>**Tags:** atp.recommendedPackage=FlatMaps | | | |
| ***Base*** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |

| | | | | |
|---|---|---|---|---|
| instance | FlatInstanceDescriptor | 1..* | aggr | A descriptor instance aggregated in the flat map.<br><br>The variation point accounts for the fact, that the system in scope can be subject to variability, and thus the existence of some instances is variable.<br><br>The aggregation has been made splitable because the content might be contributed by different stakeholders at different times in the workflow. Plus, the overall size might be so big that eventually it becomes more manageable if it is distributed over several files.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=postBuild |

**Table A.32: FlatMap**

| ***Class*** | **Identifiable (abstract)** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable | | | |
| ***Note*** | Instances of this class can be referred to by their identifier (within the namespace borders). In addition to this, Identifiables are objects which contribute significantly to the overall structure of an AUTOSAR description. In particular, Identifiables might contain Identifiables. | | | |
| ***Base*** | ARObject, MultilanguageReferrable, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| desc | MultiLanguageOverviewParagraph | 0..1 | aggr | This represents a general but brief (one paragraph) description what the object in question is about. It is only one paragraph! Desc is intended to be collected into overview tables. This property helps a human reader to identify the object in question.<br><br>More elaborate documentation, (in particular how the object is built or used) should go to "introduction".<br><br>**Tags:** xml.sequenceOffset=-60 |
| category | CategoryString | 0..1 | attr | The category is a keyword that specializes the semantics of the Identifiable. It affects the expected existence of attributes and the applicability of constraints.<br><br>**Tags:** xml.sequenceOffset=-50 |
| adminData | AdminData | 0..1 | aggr | This represents the administrative data for the identifiable object.<br><br>**Tags:** xml.sequenceOffset=-40 |

| | | | | |
|---|---|---|---|---|
| annotation | Annotation | * | aggr | Possibility to provide additional notes while defining a model element (e.g. the ECU Configuration Parameter Values). These are not intended as documentation but are mere design notes.<br><br>**Tags:** xml.sequenceOffset=-25 |
| introduction | Documentation Block | 0..1 | aggr | This represents more information about how the object in question is built or is used. Therefore it is a DocumentationBlock.<br><br>**Tags:** xml.sequenceOffset=-30 |
| uuid | String | 0..1 | attr | The purpose of this attribute is to provide a globally unique identifier for an instance of a meta-class. The values of this attribute should be globally unique strings prefixed by the type of identifier. For example, to include a DCE UUID as defined by The Open Group, the UUID would be preceded by "DCE:". The values of this attribute may be used to support merging of different AUTOSAR models. The form of the UUID (Universally Unique Identifier) is taken from a standard defined by the Open Group (was Open Software Foundation). This standard is widely used, including by Microsoft for COM (GUIDs) and by many companies for DCE, which is based on CORBA. The method for generating these 128-bit IDs is published in the standard and the effectiveness and uniqueness of the IDs is not in practice disputed. If the id namespace is omitted, DCE is assumed. An example is "DCE:2fac1234-31f8-11b4-a222-08002b34c003". The uuid attribute has no semantic meaning for an AUTOSAR model and there is no requirement for AUTOSAR tools to manage the timestamp.<br><br>**Tags:** xml.attribute=true |

**Table A.33: Identifiable**

| ***Class*** | **ImplementationDataType** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::ImplementationDataTypes | | | |
| ***Note*** | Describes a reusable data type on the implementation level. This will typically correspond to a typedef in C-code.<br><br>**Tags:** atp.recommendedPackage=ImplementationDataTypes | | | |
| ***Base*** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, AutosarDataType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| dynamicArraySizeProfile | String | 0..1 | attr | Specifies the profile which the array will follow in case this data type is a variable size array. |

| | | | | |
|---|---|---|---|---|
| subElement (ordered) | ImplementationDataTypeElement | * | aggr | Specifies an element of an array, struct, or union data type.<br><br>The aggregation of ImplementionDataTypeElement is subject to variability with the purpose to support the conditional existence of elements inside a ImplementationDataType representing a structure.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime |
| symbolProps | SymbolProps | 0..1 | aggr | This represents the SymbolProps for the ImplementationDataType.<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=shortName |
| typeEmitter | NameToken | 0..1 | attr | This attribute is used to control which part of the AUTOSAR toolchain is supposed to trigger data type definitions. |

**Table A.34: ImplementationDataType**

| ***Class*** | **InstantiationDataDefProps** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::InstantiationDataDefProps | | | |
| ***Note*** | This is a general class allowing to apply additional SwDataDefProps to particular instantiations of a DataPrototype.<br><br>Typically the accessibility and further information like alias names for a particular data is modeled on the level of DataPrototypes (especially VariableDataPrototypes, ParameterDataPrototypes). But due to the recursive structure of the meta-model concerning data types (a composite (data) type consists out of data prototypes) a part of the MCD information is described in the data type (in case of ApplicationCompositeDataType).<br><br>This is a strong restriction in the reuse of data typed because the data type should be re-used for different VariableDataPrototypes and ParameterDataPrototypes to guarantee type compatibility on C-implementation level (e.g. data of a Port is stored in PIM or a ParameterDataPrototype used as ROM Block and shall be typed by the same data type as NVRAM Block).<br><br>This class overcomes such a restriction if applied properly. | | | |
| ***Base*** | ARObject | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| parameterInstance | AutosarParameterRef | 0..1 | aggr | This is the particular ParameterDataPrototypes on which the swDataDefProps shall be applied. |
| swDataDefProps | SwDataDefProps | 1 | aggr | These are the particular data definition properties which shall be applied |
| variableInstance | AutosarVariableRef | 0..1 | aggr | This is the particular VariableDataPrototypes on which the swDataDefProps shall be applied. |

**Table A.35: InstantiationDataDefProps**

| **Class** | **InternalBehavior (abstract)** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::CommonStructure::InternalBehavior | | | |
| **Note** | Common base class (abstract) for the internal behavior of both software components and basic software modules/clusters. | | | |
| **Base** | ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| constantMemory | ParameterDataPrototype | * | aggr | Describes a read only memory object containing characteristic value(s) implemented by this InternalBehavior.<br><br>The shortName of ParameterDataPrototype has to be equal to the "C' identifier of the described constant.<br><br>The characteristic value(s) might be shared between SwComponentPrototypes of the same SwComponentType.<br><br>The aggregation of constantMemory is subject to variability with the purpose to support variability in the software component or module implementations. Typically different algorithms in the implementation are requiring different number of memory objects.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| constantValueMapping | ConstantSpecificationMappingSet | * | ref | Reference to the ConstanSpecificationMapping to be applied for the particular InternalBehavior<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=constantValueMapping |
| dataTypeMapping | DataTypeMappingSet | * | ref | Reference to the DataTypeMapping to be applied for the particular InternalBehavior<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=dataTypeMapping |
| exclusiveArea | ExclusiveArea | * | aggr | This specifies an ExclusiveArea for this InternalBehavior. The exclusiveArea is local to the component resp. module. The aggregation of ExclusiveAreas is subject to variability. Note: the number of ExclusiveAreas might vary due to the conditional existence of RunnableEntities or BswModuleEntities.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |

| | | | | |
|---|---|---|---|---|
| exclusiveAreaNestingOrder | ExclusiveAreaNestingOrder | * | aggr | This represents the set of ExclusiveAreaNestingOrder owned by the InternalBehavior.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| staticMemory | VariableDataPrototype | * | aggr | Describes a read and writeable static memory object representing measurerment variables implemented by this software component. The term "static" is used in the meaning of "non-temporary" and does not necessarily specify a linker encapsulation. This kind of memory is only supported if supportsMultipleInstantiation is FALSE.<br><br>The shortName of the VariableDataPrototype has to be equal with the "C' identifier of the described variable.<br><br>The aggregation of staticMemory is subject to variability with the purpose to support variability in the software component's implementations.<br><br>Typically different algorithms in the implementation are requiring different number of memory objects.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |

**Table A.36: InternalBehavior**

| *Class* | **McDataInstance** | | | |
|---|---|---|---|---|
| *Package* | M2::AUTOSARTemplates::CommonStructure::MeasurementCalibrationSupport | | | |
| *Note* | Describes the specific properties of one data instance in order to support measurement and/or calibration of this data instance.<br><br>The most important attributes are:<br>• Its shortName is copied from the ECU Flat map (if applicable) and will be used as identifier and for display by the MC system.<br>• The category is copied from the corresponding data type (ApplicationDataType if defined, otherwise ImplementationDataType) as far as applicable.<br>• The symbol is the one used in the programming language. It will be used to find out the actual memory address by the final generation tool with the help of linker generated information.<br><br>It is assumed that in the M1 model this part and all the aggregated and referred elements (with the exception of the Flat Map and the references from ImplementationElementInParameterInstanceRef and McAccessDetails) are completely generated from "upstream" information. This means, that even if an element like e.g. a CompuMethod is only used via reference here, it will be copied into the M1 artifact which holds the complete McSupportData for a given Implementation. | | | |
| *Base* | ARObject, Identifiable, MultilanguageReferrable, Referrable | | | |
| *Attribute* | *Type* | *Mul.* | *Kind* | *Note* |
| arraySize | PositiveInteger | 0..1 | attr | The existence of this attribute turns the data instance into an array of data. The attribute determines the size of the array in terms of number of elements. |
| displayIdentifier | McdIdentifier | 0..1 | attr | An optional attribute to be used to set the ASAM ASAP2 DISPLAY_IDENTIFIER attribute. |
| flatMapEntry | FlatInstanceDescriptor | 0..1 | ref | Reference to the corresponding entry in the ECU Flat Map. This allows to trace back to the original specification of the generated data instance. This link shall be added by the RTE generator mainly for documentation purposes.<br><br>The reference is optional because<br>• The McDataInstance may represent an array or struct in which only the subElements correspond to FlatMap entries.<br>• The McDataInstance may represent a task local buffer for rapid prototyping access which is different from the "main instance" used for measurement access. |
| instanceInMemory | ImplementationElementInParameterInstanceRef | 0..1 | aggr | Reference to the corresponding data instance in the description of calibration data structures published by the RTE generator. This is used to support emulation methods inside the ECU, it is not required for A2L generation. |

| | | | | |
|---|---|---|---|---|
| mcDataAccessDetails | McDataAccessDetails | 0..1 | aggr | Refers to "upstream" information on how the RTE uses this data instance. Use Case: Rapid Prototyping |
| mcDataAssignment | RoleBasedMcDataAssignment | * | aggr | An assignment between McDataInstances. This supports the indication of related McDataElement implementing the of ?RP global buffer", ?RP global measurement buffer", ?RP enabler flag". |
| resultingProperties | SwDataDefProps | 0..1 | aggr | These are the generated properties resulting from decisions taken by the RTE generator for the actually implemented data instance. Only those properties are relevant here, which are needed for the measurement and calibration system. |
| resultingRptSwPrototypingAccess | RptSwPrototypingAccess | 0..1 | aggr | Describes the implemented accessibility of data and modes by the rapid prototyping tooling. |
| role | Identifier | 0..1 | attr | An optional attribute to be used for additional information on the role of this data instance, for example in the context of rapid prototyping. |
| rptImplPolicy | RptImplPolicy | 0..1 | aggr | Describes the implemented code preparation for rapid prototyping at data accesses for a hook based bypassing. |
| subElement (ordered) | McDataInstance | * | aggr | This relation indicates, that the target element is part of a "struct" which is given by the source element. This information will be used by the final generator to set up the correct addressing scheme.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime |

| | | | | |
|---|---|---|---|---|
| symbol | SymbolString | 0..1 | attr | This String is used to determine the memory address during final generation of the MC configuration data (e.g. "A2L" file) . It shall be the name of the element in the programming language such that it can be identified in linker generated information.<br><br>In case the McDataInstance is part of composite data in the programming language, the symbol String may include parts denoting the element context, unless the context is given by the symbol attribute of an enclosing McDataInstance. This means in particular for the C language that the "." character shall be used as a separator between the name of a "struct" variable the name of one of its elements.<br><br>The symbol can differ from the shortName in case of generated C data declarations.<br><br>It is an optional attribute since it may be missing in case the instance represents an element (e.g. a single array element) which has no name in the linker map. |

**Table A.37: McDataInstance**

| ***Enumeration*** | **MemoryAllocationKeywordPolicyType** |
|---|---|
| ***Package*** | M2::MSR::DataDictionary::AuxillaryObjects |
| ***Note*** | Enumeration to specify the name pattern of the Memory Allocation Keyword. |
| ***Literal*** | ***Description*** |
| addrMethod ShortName | The MemorySection shortNames of referring MemorySections and therefore the belonging Memory Allocation Keywords in the code are build with the shortName of the SwAddrMethod. This is the default value if the attribute does not exist.<br><br>**Tags:** atp.EnumerationValue=0 |
| addrMethod ShortName AndAlign- ment | The MemorySection shortNames of referring MemorySections and therefore the belonging Memory Allocation Keywords in the code are build with the shortName of the SwAddrMethod and a variable alignment postfix.<br><br>Thereby the alignment postfix needs to be consistent with the alignment attribute of the related MemorySection.<br><br>**Tags:** atp.EnumerationValue=1 |

**Table A.38: MemoryAllocationKeywordPolicyType**

| ***Enumeration*** | **MemorySectionType** |
|---|---|
| ***Package*** | M2::MSR::DataDictionary::AuxillaryObjects |
| ***Note*** | Enumeration to specify the essential nature of the data which can be allocated in a common memory class by the means of the AUTOSAR Memory Mapping. |

| Literal | Description |
|---|---|
| calibration Variables | This memory section is reserved for "virtual variables" that are computed by an MCD system during a measurement session but do not exist in the ECU memory.<br><br>**Tags:** atp.EnumerationValue=2 |
| calprm | To be used for calibratable constants of ECU-functions.<br><br>**Tags:** atp.EnumerationValue=3 |
| code | To be used for mapping code to application block, boot block, external flash etc.<br><br>**Tags:** atp.EnumerationValue=4 |
| configData | Constants with attributes that show that they reside in one segment for module configuration.<br><br>**Tags:** atp.EnumerationValue=5 |
| const | To be used for global or static constants.<br><br>**Tags:** atp.EnumerationValue=6 |
| excludeFrom Flash | This memory section is reserved for "virtual parameters" that are taken for computing the values of so-called dependent parameter of an MCD system. Dependent Parameters that are not at the same time "virtual parameters" are allocated in the ECU memory.<br><br>Virtual parameters, on the other hand, are not allocated in the ECU memory. Virtual parameters exist in the ECU Hex file for the purpose of being considered (for computing the values of dependent parameters) during an offline-calibration session.<br><br>**Tags:** atp.EnumerationValue=7 |
| var | To be used for global or static variables. The expected initialization is specified with the attribute sectionInitializationPolicy.<br><br>**Tags:** atp.EnumerationValue=9 |

**Table A.39: MemorySectionType**

| ***Class*** | **ModeDeclaration** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::ModeDeclaration | | | |
| ***Note*** | Declaration of one Mode. The name and semantics of a specific mode is not defined in the meta-model. | | | |
| ***Base*** | ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| value | PositiveInteger | 0..1 | attr | The RTE shall take the value of this attribute for generating the source code representation of this ModeDeclaration. |

**Table A.40: ModeDeclaration**

| Class | ModeDeclarationGroup | | | |
|---|---|---|---|---|
| Package | M2::AUTOSARTemplates::CommonStructure::ModeDeclaration | | | |
| Note | A collection of Mode Declarations. Also, the initial mode is explicitly identified.<br><br>**Tags:** atp.recommendedPackage=ModeDeclarationGroups | | | |
| Base | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| Attribute | Type | Mul. | Kind | Note |
| initialMode | ModeDeclaration | 1 | ref | The initial mode of the ModeDeclarationGroup. This mode is active before any mode switches occurred. |
| modeDeclaration | ModeDeclaration | 1..* | aggr | The ModeDeclarations collected in this ModeDeclarationGroup.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=blueprintDerivationTime |
| modeManagerErrorBehavior | ModeErrorBehavior | 0..1 | aggr | This represents the ability to define the error behavior expected by the mode manager in case of errors on the mode user side (e.g. terminated mode user). |
| modeTransition | ModeTransition | * | aggr | This represents the avaliable ModeTransitions of the ModeDeclarationGroup |
| modeUserErrorBehavior | ModeErrorBehavior | 0..1 | aggr | This represents the definition of the error behavior expected by the mode user in case of errors on the mode manager side (e.g. terminated mode manager). |
| onTransitionValue | PositiveInteger | 0..1 | attr | The value of this attribute shall be taken into account by the RTE generator for programmatically representing a value used for the transition between two statuses. |

**Table A.41: ModeDeclarationGroup**

| Class | ModeDeclarationGroupPrototype | | | |
|---|---|---|---|---|
| Package | M2::AUTOSARTemplates::CommonStructure::ModeDeclaration | | | |
| Note | The ModeDeclarationGroupPrototype specifies a set of Modes (ModeDeclarationGroup) which is provided or required in the given context. | | | |
| Base | ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable | | | |
| Attribute | Type | Mul. | Kind | Note |
| swCalibrationAccess | SwCalibrationAccessEnum | 0..1 | attr | This allows for specifying whether or not the enclosing ModeDeclarationGroupPrototype can be measured at run-time. |
| type | ModeDeclarationGroup | 1 | tref | The "collection of ModeDeclarations" ( = ModeDeclarationGroup) supported by a component<br><br>**Stereotypes:** isOfType |

**Table A.42: ModeDeclarationGroupPrototype**

| ***Class*** | **ModeSwitchInterface** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::PortInterface | | | |
| ***Note*** | A mode switch interface declares a ModeDeclarationGroupPrototype to be sent and received.<br><br>**Tags:** atp.recommendedPackage=PortInterfaces | | | |
| ***Base*** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Port Interface, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| modeGroup | ModeDeclarationGroupPrototype | 1 | aggr | The ModeDeclarationGroupPrototype of this mode interface. |

**Table A.43: ModeSwitchInterface**

| ***Class*** | **NumericalValueSpecification** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::Constants | | | |
| ***Note*** | A numerical ValueSpecification which is intended to be assigned to a Primitive data element. Note that the numerical value is a variant, it can be computed by a formula. | | | |
| ***Base*** | ARObject, ValueSpecification | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| value | Numerical | 1 | attr | This is the value itself.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime |

**Table A.44: NumericalValueSpecification**

| ***Class*** | **PPortPrototype** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Components | | | |
| ***Note*** | Component port providing a certain port interface. | | | |
| ***Base*** | ARObject, AbstractProvidedPortPrototype, AtpBlueprintable, AtpFeature, Atp Prototype, Identifiable, MultilanguageReferrable, PortPrototype, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| providedInterface | PortInterface | 1 | tref | The interface that this port provides.<br><br>**Stereotypes:** isOfType |

**Table A.45: PPortPrototype**

| Class | ParameterAccess | | | |
|---|---|---|---|---|
| Package | M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::DataElements | | | |
| Note | The presence of a ParameterAccess implies that a RunnableEntity needs access to a ParameterDataPrototype. | | | |
| Base | ARObject, AbstractAccessPoint, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable | | | |
| Attribute | Type | Mul. | Kind | Note |
| accessedParameter | AutosarParameterRef | 1 | aggr | Refernce to the accessed calibration parameter. |
| swDataDefProps | SwDataDefProps | 0..1 | aggr | This allows denote instance and access specific properties, mainly input values and common axis. |

**Table A.46: ParameterAccess**

| Class | ParameterDataPrototype | | | |
|---|---|---|---|---|
| Package | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes | | | |
| Note | A parameter element used for parameter interface and internal behavior, supporting signal like parameter and characteristic value communication patterns and parameter and characteristic value definition. | | | |
| Base | ARObject, AtpFeature, AtpPrototype, AutosarDataPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable | | | |
| Attribute | Type | Mul. | Kind | Note |
| initValue | ValueSpecification | 0..1 | aggr | Specifies initial value(s) of the ParameterDataPrototype |

**Table A.47: ParameterDataPrototype**

| Class | ParameterInterface | | | |
|---|---|---|---|---|
| Package | M2::AUTOSARTemplates::SWComponentTemplate::PortInterface | | | |
| Note | A parameter interface declares a number of parameter and characteristic values to be exchanged between parameter components and software components.<br><br>**Tags:** atp.recommendedPackage=PortInterfaces | | | |
| Base | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DataInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable | | | |
| Attribute | Type | Mul. | Kind | Note |
| parameter | ParameterDataPrototype | 1..* | aggr | The ParameterDataPrototype of this ParameterInterface. |

**Table A.48: ParameterInterface**

| **Class** | **ParameterProvideComSpec** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::Communication | | | |
| **Note** | "Communication" specification that applies to parameters on the provided side of a connection. | | | |
| **Base** | ARObject, PPortComSpec | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| initValue | ValueSpecification | 0..1 | aggr | The initial value applicable for the corresponding ParameterDataPrototype. |
| parameter | ParameterDataPrototype | 1 | ref | The ParameterDataPrototype to which the ParameterComSpec applies. |

**Table A.49: ParameterProvideComSpec**

| **Class** | **ParameterSwComponentType** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::Components | | | |
| **Note** | The ParameterSwComponentType defines parameters and characteristic values accessible via provided Ports. The provided values are the same for all connected SwComponentPrototypes<br><br>**Tags:** atp.recommendedPackage=SwComponentTypes | | | |
| **Base** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable, SwComponentType | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| constantMapping | ConstantSpecificationMappingSet | * | ref | Reference to the ConstanSpecificationMapping to be applied for the particular ParameterSwComponentType<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=constantMapping |
| dataTypeMapping | DataTypeMappingSet | * | ref | Reference to the DataTypeMapping to be applied for the particular ParameterSwComponentType<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=dataTypeMapping |
| instantiationDataDefProps | InstantiationDataDefProps | * | aggr | The purpose of this is that within the context of a given SwComponentType some data def properties of individual instantiations can be modified.<br><br>The aggregation of InstantiationDataDefProps is subject to variability with the purpose to support the conditional existence of PortPrototypes<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime |

**Table A.50: ParameterSwComponentType**

| **Class** | **PhysConstrs** | | | |
|---|---|---|---|---|
| **Package** | M2::MSR::AsamHdo::Constraints::GlobalConstraints | | | |
| **Note** | This meta-class represents the ability to express physical constraints. Therefore it has (in opposite to InternalConstrs) a reference to a Unit. | | | |
| **Base** | ARObject | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| lowerLimit | Limit | 0..1 | attr | This specifies the lower limit of the constraint.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime<br>xml.sequenceOffset=20 |
| maxDiff | Numerical | 0..1 | attr | Maximum difference that is permitted between two consecutive values if the constraint is applied to an axis.<br><br>**Tags:** xml.sequenceOffset=60 |
| maxGradient | Numerical | 0..1 | attr | This element specifies the maximum slope that may be used in curves and maps.<br><br>**Tags:** xml.sequenceOffset=50 |
| monotony | MonotonyEnum | 0..1 | attr | This specifies the monotony constraints on the data object. Note that this applies only to curves and maps.<br><br>**Tags:** xml.sequenceOffset=70 |
| scaleConstr (ordered) | ScaleConstr | * | aggr | This is one particular scale which contributes to the data constraints.<br><br>**Tags:** xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=40; xml.typeElement=false; xml.typeWrapperElement=false |
| unit | Unit | 0..1 | ref | This is the unit to which the physical constraints relate to. In particular, it is the physical unit of the specified limits.<br><br>**Tags:** xml.sequenceOffset=80 |
| upperLimit | Limit | 0..1 | attr | This specifies the upper limit of the constraint.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime<br>xml.sequenceOffset=30 |

**Table A.51: PhysConstrs**

| **Class** | **PhysicalDimension** | | | |
|---|---|---|---|---|
| **Package** | M2::MSR::AsamHdo::Units | | | |
| **Note** | This class represents a physical dimension. If the physical dimension of two units is identical, then a conversion between them is possible. The conversion between units is related to the definition of the physical dimension.<br><br>Note that the equivalence of the exponents does not per se define the convertibility. For example Energy and Torque share the same exponents (Nm).<br><br>Please note further the value of an exponent does not necessarily have to be an integer number. It is also possible that the value yields a rational number, e.g. to compute the square root of a given physical quantity. In this case the exponent value would be a rational number where the numerator value is 1 and the denominator value is 2.<br><br>**Tags:** atp.recommendedPackage=PhysicalDimensions | | | |
| **Base** | ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| currentExp | Numerical | 0..1 | attr | This attribute represents the exponent of the physical dimension "electric current".<br><br>**Tags:** xml.sequenceOffset=50 |
| lengthExp | Numerical | 0..1 | attr | The exponent of the physical dimension "length".<br><br>**Tags:** xml.sequenceOffset=20 |
| luminousIntensityExp | Numerical | 0..1 | attr | The exponent of the physical dimension "luminous intensity".<br><br>**Tags:** xml.sequenceOffset=80 |
| massExp | Numerical | 0..1 | attr | The exponent of the physical dimension "mass".<br><br>**Tags:** xml.sequenceOffset=30 |
| molarAmountExp | Numerical | 0..1 | attr | The exponent of the physical dimension "quantity of substance".<br><br>**Tags:** xml.sequenceOffset=70 |
| temperatureExp | Numerical | 0..1 | attr | The exponent of the physical dimension "temperature".<br><br>**Tags:** xml.sequenceOffset=60 |
| timeExp | Numerical | 0..1 | attr | The exponent of the physical dimension "time".<br><br>**Tags:** xml.sequenceOffset=40 |

**Table A.52: PhysicalDimension**

| Class | PortInterface (abstract) | | | |
|---|---|---|---|---|
| Package | M2::AUTOSARTemplates::SWComponentTemplate::PortInterface | | | |
| Note | Abstract base class for an interface that is either provided or required by a port of a software component. | | | |
| Base | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| Attribute | Type | Mul. | Kind | Note |
| isService | Boolean | 1 | attr | This flag is set if the PortInterface is to be used for communication between an<br>• ApplicationSwComponentType or<br>• ServiceProxySwComponentType or<br>• SensorActuatorSwComponentType or<br>• ComplexDeviceDriverSwComponentType<br>• ServiceSwComponentType<br>• EcuAbstractionSwComponentType<br><br>and a ServiceSwComponentType (namely an AUTOSAR Service) located on the same ECU. Otherwise the flag is not set. |
| serviceKind | ServiceProviderEnum | 0..1 | attr | This attribute provides further details about the nature of the applied service. |

**Table A.53: PortInterface**

| Class | PortPrototype (abstract) | | | |
|---|---|---|---|---|
| Package | M2::AUTOSARTemplates::SWComponentTemplate::Components | | | |
| Note | Base class for the ports of an AUTOSAR software component.<br><br>The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports. | | | |
| Base | ARObject, AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable | | | |
| Attribute | Type | Mul. | Kind | Note |
| clientServerAnnotation | ClientServerAnnotation | * | aggr | Annotation of this PortPrototype with respect to client/server communication. |
| delegatedPortAnnotation | DelegatedPortAnnotation | 0..1 | aggr | Annotations on this delegated port. |
| ioHwAbstractionServerAnnotation | IoHwAbstractionServerAnnotation | * | aggr | Annotations on this IO Hardware Abstraction port. |
| modePortAnnotation | ModePortAnnotation | * | aggr | Annotations on this mode port. |

| | | | | |
|---|---|---|---|---|
| nvDataPortAnnotation | NvDataPortAnnotation | * | aggr | Annotations on this non voilatile data port. |
| parameterPortAnnotation | ParameterPortAnnotation | * | aggr | Annotations on this parameter port. |
| senderReceiverAnnotation | SenderReceiverAnnotation | * | aggr | Collection of annotations of this ports sender/receiver communication. |
| triggerPortAnnotation | TriggerPortAnnotation | * | aggr | Annotations on this trigger port. |

**Table A.54: PortPrototype**

| ***Class*** | **RPortPrototype** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Components | | | |
| ***Note*** | Component port requiring a certain port interface. | | | |
| ***Base*** | ARObject, AbstractRequiredPortPrototype, AtpBlueprintable, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, PortPrototype, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| requiredInterface | PortInterface | 1 | tref | The interface that this port requires, i.e. the port depends on another port providing the specified interface.<br><br>**Stereotypes:** isOfType |

**Table A.55: RPortPrototype**

| ***Class*** | **RTEEvent (abstract)** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents | | | |
| ***Note*** | Abstract base class for all RTE-related events | | | |
| ***Base*** | ARObject, AbstractEvent, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| disabledMode | ModeDeclaration | * | iref | Reference to the Modes that disable the Event.<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=contextPort, contextModeDeclarationGroupPrototype, targetModeDeclaration |
| startOnEvent | RunnableEntity | 0..1 | ref | RunnableEntity starts when the corresponding RTEEvent occurs. |

**Table A.56: RTEEvent**

| ***Primitive*** | **Ref** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Primitive Types | | | |
| ***Note*** | This primitive denotes a name based reference. For detailed syntax see the xsd.pattern.<br>• first slash (relative or absolute reference) [optional]<br>• Identifier [required]<br>• a sequence of slashes and Identifiers [optional]<br><br>This primitive is used by the meta-model tools to create the references.<br><br>**Tags:** xml.xsd.customType=REF; xml.xsd.pattern=/?[a-zA-Z][a-zA-Z0-9_]{0,127}(/[a-zA-Z][a-zA-Z0-9_]{0,127})*; xml.xsd.type=string | | | |
| ***Attribute*** | ***Datatype*** | ***Mul.*** | ***Kind*** | ***Note*** |
| base | Identifier | 0..1 | attr | This attribute reflects the base to be used for this reference.<br><br>**Tags:** xml.attribute=true |
| index | PositiveInteger | 0..1 | attr | This attribute supports the use case to point on specific elements in an array. This is in particular required if arrays are used to implement particular data objects.<br><br>**Tags:** xml.attribute=true |

**Table A.57: Ref**

**[constr_2552] Index attribute is only valid for arrays** ⌈The index attribute in references is valid only if the reference target is an ApplicationArrayElement or if the reference target is an ImplementationDataTypeElement owned by an ImplementationDataType/ImplementationDataTypeElement of category ARRAY and has an attribute maxNumberOfElements/arraySize.⌋*()*

| ***Class*** | **Referrable (abstract)** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::GenericStructure::GeneralTemplateClasses::Identifiable | | | |
| ***Note*** | Instances of this class can be referred to by their identifier (while adhering to namespace borders). | | | |
| ***Base*** | ARObject | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| shortName | Identifier | 1 | attr | This specifies an identifying shortName for the object. It needs to be unique within its context and is intended for humans but even more for technical reference.<br><br>**Tags:** xml.enforceMinMultiplicity=true; xml.sequenceOffset=-100 |

| | | | | |
|---|---|---|---|---|
| shortName Fragment | ShortNameFragment | * | aggr | This specifies how the Referrable.shortName is composed of several shortNameFragments.<br><br>**Tags:** xml.sequenceOffset=-90 |

**Table A.58: Referrable**

| ***Class*** | **RootSwCompositionPrototype** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate | | | |
| ***Note*** | The RootSwCompositionPrototype represents the top-level-composition of software components within a given System. According to the use case of the System, this may for example be the a more or less complete VFB description, the software of a System Extract or the software of a flat ECU Extract with only atomic SWCs.<br><br>Therefore the RootSwComposition will only occasionally contain all atomic software components that are used in a complete VFB System. The OEM is primarily interested in the required functionality and the interfaces defining the integration of the Software Component into the System. The internal structure of such a component contains often substantial intellectual property of a supplier. Therefore a top-level software composition will often contain empty compositions which represent subsystems.<br><br>The contained SwComponentPrototypes are fully specified by their SwComponentTypes (including PortPrototypes, PortInterfaces, VariableDataPrototypes, SwcInternalBehavior etc.), and their ports are interconnected using SwConnectorPrototypes. | | | |
| ***Base*** | ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| calibration Parameter ValueSet | CalibrationParameterValueSet | * | ref | Used CalibrationParameterValueSet for instance specific initialization of calibration parameters.<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=calibrationParameterValueSet |
| flatMap | FlatMap | 0..1 | ref | The FlatMap used in the scope of this RootSwCompositionPrototype.<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=flatMap |
| softwareComposition | CompositionSwComponentType | 1 | tref | We assume that there is exactly one top-level composition that includes all Component instances of the system<br><br>**Stereotypes:** isOfType |

**Table A.59: RootSwCompositionPrototype**

| **Class** | **RunnableEntity** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior | | | |
| **Note** | A RunnableEntity represents the smallest code-fragment that is provided by an AtomicSwComponentType and are executed under control of the RTE. RunnableEntities are for instance set up to respond to data reception or operation invocation on a server. | | | |
| **Base** | ARObject, AtpClassifier, AtpFeature, AtpStructureElement, ExecutableEntity, Identifiable, MultilanguageReferrable, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| argument (ordered) | RunnableEntity Argument | * | aggr | This represents the formal definition of a an argument to a RunnableEntity. |
| asynchronousServerCallResultPoint | AsynchronousServerCallResultPoint | * | aggr | The server call result point admits a runnable to fetch the result of an asynchronous server call.<br><br>The aggregation of AsynchronousServerCallResultPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes and the variant existence of server call result points in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| canBeInvokedConcurrently | Boolean | 1 | attr | If the value of this attribute is set to "true" the enclosing RunnableEntity can be invoked concurrently (even for one instance of the corresponding AtomicSwComponentType). This implies that it is the responsibility of the implementation of the RunnableEntity to take care of this form of concurrency. Note that the default value of this attribute is set to "false". |
| dataReadAccess | VariableAccess | * | aggr | RunnableEntity has implicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.<br><br>The aggregation of dataReadAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataReadAccess in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |

| | | | | |
|---|---|---|---|---|
| dataReceivePointByArgument | VariableAccess | * | aggr | RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype. The result is passed back to the application by means of an argument in the function signature.<br><br>The aggregation of dataReceivePointByArgument is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data receive points in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| dataReceivePointByValue | VariableAccess | * | aggr | RunnableEntity has explicit read access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.<br><br>The result is passed back to the application by means of the return value. The aggregation of dataReceivePointByValue is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of data receive points in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| dataSendPoint | VariableAccess | * | aggr | RunnableEntity has explicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.<br><br>The aggregation of dataSendPoint is subject to variability with the purpose to support the conditional existence of sender receiver PortPrototype or the variant existence of data send points in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |

| | | | | |
|---|---|---|---|---|
| dataWriteAccess | VariableAccess | * | aggr | RunnableEntity has implicit write access to dataElement of a sender-receiver PortPrototype or nv data of a nv data PortPrototype.<br><br>The aggregation of dataWriteAccess is subject to variability with the purpose to support the conditional existence of sender receiver ports or the variant existence of dataWriteAccess in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| externalTriggeringPoint | ExternalTriggeringPoint | * | aggr | The aggregation of ExternalTriggeringPoint is subject to variability with the purpose to support the conditional existence of trigger ports or the variant existence of external triggering points in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=externalTriggeringPoint, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| internalTriggeringPoint | InternalTriggeringPoint | * | aggr | The aggregation of InternalTriggeringPoint is subject to variability with the purpose to support the variant existence of internal triggering points in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| modeAccessPoint | ModeAccessPoint | * | aggr | The runnable has a mode access point. The aggregation of ModeAccessPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode access points in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=modeAccessPoint, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |

| | | | | |
|---|---|---|---|---|
| modeSwitchPoint | ModeSwitchPoint | * | aggr | The runnable has a mode switch point. The aggregation of ModeSwitchPoint is subject to variability with the purpose to support the conditional existence of mode ports or the variant existence of mode switch points in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| parameterAccess | ParameterAccess | * | aggr | The presence of a ParameterAccess implies that a RunnableEntity needs read only access to a ParameterDataPrototype which may either be local or within a PortPrototype.<br><br>The aggregation of ParameterAccess is subject to variability with the purpose to support the conditional existence of parameter ports and component local parameters as well as the variant existence of ParameterAccess (points) in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| readLocalVariable | VariableAccess | * | aggr | The presence of a readLocalVariable implies that a RunnableEntity needs read access to a VariableDataPrototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable.<br><br>The aggregation of readLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicitInterRunnableVariable or the variant existence of readLocalVariable (points) in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |

| | | | | |
|---|---|---|---|---|
| serverCall Point | ServerCallPoint | * | aggr | The RunnableEntity has a ServerCallPoint. The aggregation of ServerCallPoint is subject to variability with the purpose to support the conditional existence of client server PortPrototypes or the variant existence of server call points in the implementation.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variation Point.shortLabel<br>vh.latestBindingTime=preCompileTime |
| symbol | CIdentifier | 1 | attr | The symbol describing this RunnableEntity's entry point. This is considered the API of the RunnableEntity and is required during the RTE contract phase. |
| waitPoint | WaitPoint | * | aggr | The WaitPoint associated with the RunnableEntity. |
| writtenLoc alVariable | VariableAccess | * | aggr | The presence of a writtenLocalVariable implies that a RunnableEntity needs write access to a VariableDataPrototype in the role of implicitInterRunnableVariable or explicitInterRunnableVariable.<br><br>The aggregation of writtenLocalVariable is subject to variability with the purpose to support the conditional existence of implicitInterRunnableVariable and explicitInterRunnableVariable or the variant existence of writtenLocalVariable (points) in the implementation.<br><br>**Stereotypes:** atpVariation<br>**Tags:** atp.Splitkey=shortName, variation Point.shortLabel<br>vh.latestBindingTime=preCompileTime |

**Table A.60: RunnableEntity**

| ***Class*** | **SenderReceiverInterface** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::PortInterface | | | |
| ***Note*** | A sender/receiver interface declares a number of data elements to be sent and received.<br><br>**Tags:** atp.recommendedPackage=PortInterfaces | | | |
| ***Base*** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, DataInterface, Identifiable, MultilanguageReferrable, PackageableElement, PortInterface, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| dataEleme nt | VariableDataPr ototype | 1..* | aggr | The data elements of this SenderReceiverInterface. |
| invalidation Policy | InvalidationPolic y | * | aggr | InvalidationPolicy for a particular dataElement |

**Table A.61: SenderReceiverInterface**

| Class | SwAddrMethod | | | |
|---|---|---|---|---|
| Package | M2::MSR::DataDictionary::AuxillaryObjects | | | |
| Note | Used to assign a common addressing method, e.g. common memory section, to data or code objects. These objects could actually live in different modules or components.<br><br>**Tags:** atp.recommendedPackage=SwAddrMethods | | | |
| Base | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| memoryAllocationKeywordPolicy | MemoryAllocationKeywordPolicyType | 0..1 | attr | Enumeration to specify the name pattern of the Memory Allocation Keyword. |
| option | Identifier | * | attr | This attribute introduces the ability to specify further intended properties of the MemorySection in with the related objects shall be placed.<br><br>These properties are handled as to be selected. The intended options are mentioned in the list.<br><br>In the Memory Mapping configuration, this option list is used to determine an appropriate MemMapAddressingModeSet. |
| sectionInitializationPolicy | SectionInitializationPolicyType | 0..1 | attr | Specifies the expected initialization of the variables (inclusive those which are implementing VariableDataPrototypes). Therefore this is an implementation constraint for initialization code of BSW modules (especially RTE) as well as the start-up code which initializes the memory segment to which the AutosarDataPrototypes referring to the SwAddrMethod's are later on mapped.<br><br>If the attribute is not defined it has the identical semantic as the attribute value "INIT" |
| sectionType | MemorySectionType | 0..1 | attr | Defines the type of memory sections which can be associated with this addresssing method. |

**Table A.62: SwAddrMethod**

| Class | SwAxisGrouped | | | |
|---|---|---|---|---|
| Package | M2::MSR::DataDictionary::Axis | | | |
| Note | An SwAxisGrouped is an axis which is shared between multiple calibration parameters. | | | |
| Base | ARObject, SwCalprmAxisTypeProps | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| sharedAxisType | ApplicationPrimitiveDataType | 0..1 | ref | This is the datatype of the calibration parameter providing the shared axis. |

| | | | | |
|---|---|---|---|---|
| swAxisIndex | AxisIndexType | 0..1 | attr | Describes which axis of the referenced calibration parameter provides the values for the group axis. The index satisfies the following convention:<br>• 0 = value axis. in this case, the interpolation result of the referenced parameter is used as a base point index.<br>• The index should only be specified if the parameter under swCalprm contains more than one axis. It is standard practice for the axis index of parameters with more than one axis, to be set to 1, if data has not been assigned to swAxisIndex.<br><br>**Tags:** xml.sequenceOffset=20 |
| swCalprmRef | SwCalprmRefProxy | 1 | aggr | This property specifes the calibration parameter which serves as the input axis. In AUTOSAR, the type of the referenced Calibration parameter shall be compatible to the type specified by sharedAxisType.<br><br>**Tags:** xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=30; xml.typeElement=false; xml.typeWrapperElement=false |

**Table A.63: SwAxisGrouped**

| ***Class*** | **SwAxisIndividual** | | | |
|---|---|---|---|---|
| ***Package*** | M2::MSR::DataDictionary::Axis | | | |
| ***Note*** | This meta-class describes an axis integrated into a parameter (field etc.). The integration makes this individual to each parameter. The so-called grouped axis represents the counterpart to this. It is conceived as an independent parameter (see class SwAxisGrouped). | | | |
| ***Base*** | ARObject, SwCalprmAxisTypeProps | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| compuMethod | CompuMethod | 0..1 | ref | This is the compuMethod which is expected for the axis. It is used in early stages if the particular input-value is not yet available.<br><br>**Tags:** xml.sequenceOffset=30 |
| dataConstr | DataConstr | 0..1 | ref | Refers to constraints, e.g. for plausibility checks.<br><br>**Tags:** xml.sequenceOffset=80 |
| inputVariableType | ApplicationPrimitiveDataType | 0..1 | ref | This is the datatype of the input value for the axis. This allows to define e.g. a type of curve, where the input value is finalized at the access point. |
| swAxisGeneric | SwAxisGeneric | 0..1 | aggr | this specifies the properties of a generic axis if applicable.<br><br>**Tags:** xml.sequenceOffset=90 |

| | | | | |
|---|---|---|---|---|
| swMaxAxisPoints | Integer | 1 | attr | Maximum number of base points contained in the axis of a map or curve.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime<br>xml.sequenceOffset=60 |
| swMinAxisPoints | Integer | 1 | attr | Minimum number of base points contained in the axis of a map or curve.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime<br>xml.sequenceOffset=70 |
| swVariableRef (ordered) | SwVariableRefProxy | * | aggr | Refers to input variables of the axis. It is possible to specify more than one variable. Here the following is valid:<br>• The variable with the highest priority shall be given first. It is used in the generation of the code and is also displayed first in the application system.<br>• All variables referenced shall be of the same physical nature. This is usually detected in that the conversion formulae affected refer back to the same SI-units.<br><br>In AUTOSAR this ensured by the constraint, that the referenced input variables shall use a type compatible to "inputVariableType".<br>• This multiple referencing allows a base point distribution for more than one input variable to be used. One example of this are the temperature curves which can depend both on the induction air temperature and the engine temperature.<br><br>These variables can be displayed simultaneously by MCD systems (adjustment systems), enabling operating points to be shown in the curves.<br><br>**Tags:** xml.roleElement=false; xml.roleWrapperElement=true; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false |
| unit | Unit | 0..1 | ref | This represents the physical unit of the input value of the axis. It is provided to support the case that the particular input variable is not yet known.<br><br>**Tags:** xml.sequenceOffset=40 |

**Table A.64: SwAxisIndividual**

| **Class** | **SwBaseType** | | | |
|---|---|---|---|---|
| **Package** | M2::MSR::AsamHdo::BaseTypes | | | |
| **Note** | This meta-class represents a base type used within ECU software.<br><br>**Tags:** atp.recommendedPackage=BaseTypes | | | |
| **Base** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, BaseType, Collectable Element, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| – | – | – | – | – |

**Table A.65: SwBaseType**

| **Enumeration** | **SwCalibrationAccessEnum** |
|---|---|
| **Package** | M2::MSR::DataDictionary::DataDefProperties |
| **Note** | Determines the access rights to a data object w.r.t. measurement and calibration. |
| ***Literal*** | ***Description*** |
| notAccessible | The element will not be accessible via MCD tools, i.e. will not appear in the ASAP file.<br><br>**Tags:** atp.EnumerationValue=0 |
| readOnly | The element will only appear as read-only in an ASAP file.<br><br>**Tags:** atp.EnumerationValue=1 |
| readWrite | The element will appear in the ASAP file with both read and write access.<br><br>**Tags:** atp.EnumerationValue=2 |

**Table A.66: SwCalibrationAccessEnum**

| **Class** | **SwCalprmAxis** | | | |
|---|---|---|---|---|
| **Package** | M2::MSR::DataDictionary::CalibrationParameter | | | |
| **Note** | This element specifies an individual input parameter axis (abscissa). | | | |
| **Base** | ARObject | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| category | CalprmAxisCategoryEnum | 0..1 | attr | This property specifies the category of a particular axis.<br><br>**Tags:** xml.sequenceOffset=30 |
| baseType | SwBaseType | 0..1 | ref | The SwBaseType to be used for the axis. Note that this is not applicable for ApplicationDataTypes. The value shall be ignored.<br><br>**Tags:** atp.Status=removed<br>xml.sequenceOffset=110 |
| displayFormat | DisplayFormatString | 0..1 | attr | This property specifies how the axis values shall be displayed e.g. in documents or in measurement and calibration tools.<br><br>**Tags:** xml.sequenceOffset=100 |

| | | | | |
|---|---|---|---|---|
| swAxisIndex | AxisIndexType | 0..1 | attr | This attribute specifies which axis is specified by the containing SwCalprmAxis.<br><br>For example in a curve this is usually "1". In a map this is "1" or "2".<br><br>**Tags:** xml.sequenceOffset=20 |
| swCalibrationAccess | SwCalibrationAccessEnum | 0..1 | attr | Describes the applicability of parameters and variables.<br><br>**Tags:** xml.sequenceOffset=90 |
| swCalprmAxisTypeProps | SwCalprmAxisTypeProps | 1 | aggr | specific properties depending on the type of the axis.<br><br>**Tags:** xml.roleElement=false; xml.roleWrapperElement=false; xml.sequenceOffset=40; xml.typeElement=true; xml.typeWrapperElement=false |

**Table A.67: SwCalprmAxis**

| ***Class*** | **SwCalprmAxisSet** | | | |
|---|---|---|---|---|
| ***Package*** | M2::MSR::DataDictionary::CalibrationParameter | | | |
| ***Note*** | This element specifies the input parameter axes (abscissas) of parameters (and variables, if these are used adaptively). | | | |
| ***Base*** | ARObject | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| swCalprmAxis | SwCalprmAxis | * | aggr | One axis belonging to this SwCalprmAxisSet<br><br>**Tags:** xml.roleElement=true; xml.roleWrapperElement=false; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false |

**Table A.68: SwCalprmAxisSet**

| ***Class*** | **SwComponentPrototype** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Composition | | | |
| ***Note*** | Role of a software component within a composition. | | | |
| ***Base*** | ARObject, AtpFeature, AtpPrototype, Identifiable, MultilanguageReferrable, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| type | SwComponentType | 1 | tref | Type of the instance.<br><br>**Stereotypes:** isOfType |

**Table A.69: SwComponentPrototype**

| | | | | |
|---|---|---|---|---|
| ***Class*** | **SwComponentType (abstract)** | | | |
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Components | | | |
| ***Note*** | Base class for AUTOSAR software components. | | | |
| ***Base*** | ARElement, ARObject, AtpBlueprint, AtpBlueprintable, AtpClassifier, AtpType, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| consistencyNeeds | ConsistencyNeeds | * | aggr | This represents the colelction of ConsistencyNeeds owned by the enclosing SwComponentType.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| port | PortPrototype | * | aggr | The PortPrototypes through which this SwComponentType can communicate.<br><br>The aggregation of PortPrototype is subject to variability with the purpose to support the conditional existence of PortPrototypes.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| portGroup | PortGroup | * | aggr | A port group being part of this component.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime |
| swComponentDocumentation | SwComponentDocumentation | 0..1 | aggr | This adds a documentation to the SwComponentType.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=swComponentDocumentation, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime<br>xml.sequenceOffset=-10 |
| unitGroup | UnitGroup | * | ref | This allows for the specification of which UnitGroups are relevant in the context of referencing SwComponentType. |

**Table A.70: SwComponentType**

| **Class** | ≪atpVariation≫ **SwDataDefProps** | | | |
|---|---|---|---|---|
| **Package** | M2::MSR::DataDictionary::DataDefProperties | | | |
| **Note** | This class is a collection of properties relevant for data objects under various aspects. One could consider this class as a "pattern of inheritance by aggregation". The properties can be applied to all objects of all classes in which SwDataDefProps is aggregated.<br><br>Note that not all of the attributes or associated elements are useful all of the time. Hence, the process definition (e.g. expressed with an OCL or a Document Control Instance MSR-DCI) has the task of implementing limitations.<br><br>SwDataDefProps covers various aspects:<br>• Structure of the data element for calibration use cases: is it a single value, a curve, or a map, but also the recordLayouts which specify how such elements are mapped/converted to the DataTypes in the programming language (or in AUTOSAR). This is mainly expressed by properties like swRecordLayout and swCalprmAxisSet<br>• Implementation aspects, mainly expressed by swImplPolicy, swVariableAccessImplPolicy, swAddrMethod, swPointerTagetProps, baseType, implementationDataType and additionalNativeTypeQualifier<br>• Access policy for the MCD system, mainly expressed by swCalibrationAccess<br>• Semantics of the data element, mainly expressed by compuMethod and/or unit, dataConstr, invalidValue<br>• Code generation policy provided by swRecordLayout<br><br>**Tags:** vh.latestBindingTime=codeGenerationTime | | | |
| **Base** | ARObject | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| additionalNativeTypeQualifier | NativeDeclarationString | 0..1 | attr | This attribute is used to declare native qualifiers of the programming language which can neither be deduced from the baseType (e.g. because the data object describes a pointer) nor from other more abstract attributes. Examples are qualifiers like "volatile", "strict" or "enum" of the C-language. All such declarations have to be put into one string.<br><br>**Tags:** xml.sequenceOffset=235 |
| annotation | Annotation | * | aggr | This aggregation allows to add annotations (yellow pads ...) related to the current data object.<br><br>**Tags:** xml.roleElement=true; xml.roleWrapperElement=true; xml.sequenceOffset=20; xml.typeElement=false; xml.typeWrapperElement=false |
| baseType | SwBaseType | 0..1 | ref | Base type associated with the containing data object.<br><br>**Tags:** xml.sequenceOffset=50 |

| | | | | |
|---|---|---|---|---|
| compuMethod | CompuMethod | 0..1 | ref | Computation method associated with the semantics of this data object.<br><br>**Tags:** xml.sequenceOffset=180 |
| dataConstr | DataConstr | 0..1 | ref | Data constraint for this data object.<br><br>**Tags:** xml.sequenceOffset=190 |
| displayFormat | DisplayFormatString | 0..1 | attr | This property describes how a number is to be rendered e.g. in documents or in a measurement and calibration system.<br><br>**Tags:** xml.sequenceOffset=210 |
| implementationDataType | ImplementationDataType | 0..1 | ref | This association denotes the ImplementationDataType of a data declaration via its aggregated SwDataDefProps. It is used whenever a data declaration is not directly referring to a base type. Especially<br>• redefinition of an ImplementationDataType via a "typedef" to another ImplementationDatatype<br>• the target type of a pointer (see SwPointerTargetProps), if it does not refer to a base type directly<br>• the data type of an array or record element within an ImplementationDataType, if it does not refer to a base type directly<br>• the data type of an SwServiceArg, if it does not refer to a base type directly<br><br>**Tags:** xml.sequenceOffset=215 |
| invalidValue | ValueSpecification | 0..1 | aggr | Optional value to express invalidity of the actual data element.<br><br>**Tags:** xml.sequenceOffset=255 |
| stepSize | Float | 0..1 | attr | This attribute can be used to define a value which is added to or subtracted from the value of a DataPrototype when using up/down keys while calibrating. |
| swAddrMethod | SwAddrMethod | 0..1 | ref | Addressing method related to this data object. Via an association to the same SwAddrMethod it can be specified that several DataPrototypes shall be located in the same memory without already specifying the memory section itself.<br><br>**Tags:** xml.sequenceOffset=30 |

| | | | | |
|---|---|---|---|---|
| swAlignment | AlignmentType | 0..1 | attr | The attribute describes the intended alignment of the DataPrototype. If the attribute is not defined the alignment is determined by the swBaseType size and the memoryAllocationKeywordPolicy of the referenced SwAddrMethod.<br><br>**Tags:** xml.sequenceOffset=33 |
| swBitRepresentation | SwBitRepresentation | 0..1 | aggr | Description of the binary representation in case of a bit variable.<br><br>**Tags:** xml.sequenceOffset=60 |
| swCalibrationAccess | SwCalibrationAccessEnum | 0..1 | attr | Specifies the read or write access by MCD tools for this data object.<br><br>**Tags:** xml.sequenceOffset=70 |
| swCalprmAxisSet | SwCalprmAxisSet | 0..1 | aggr | This specifies the properties of the axes in case of a curve or map etc. This is mainly applicable to calibration parameters.<br><br>**Tags:** xml.sequenceOffset=90 |
| swComparisonVariable | SwVariableRefProxy | * | aggr | Variables used for comparison in an MCD process.<br><br>**Tags:** xml.sequenceOffset=170; xml.typeElement=false |
| swDataDependency | SwDataDependency | 0..1 | aggr | Describes how the value of the data object has to be calculated from the value of another data object (by the MCD system).<br><br>**Tags:** xml.sequenceOffset=200 |
| swHostVariable | SwVariableRefProxy | 0..1 | aggr | Contains a reference to a variable which serves as a host-variable for a bit variable. Only applicable to bit objects.<br><br>**Tags:** xml.sequenceOffset=220; xml.typeElement=false |
| swImplPolicy | SwImplPolicyEnum | 0..1 | attr | Implementation policy for this data object.<br><br>**Tags:** xml.sequenceOffset=230 |

| | | | | |
|---|---|---|---|---|
| swIntendedResolution | Numerical | 0..1 | attr | The purpose of this element is to describe the requested quantization of data objects early on in the design process.<br><br>The resolution ultimately occurs via the conversion formula present (compuMethod), which specifies the transition from the physical world to the standardized world (and vice-versa) (here, "the slope per bit" is present implicitly in the conversion formula).<br><br>In the case of a development phase without a fixed conversion formula, a pre-specification can occur through swIntendedResolution.<br><br>The resolution is specified in the physical domain according to the property "unit".<br><br>**Tags:** xml.sequenceOffset=240 |
| swInterpolationMethod | Identifier | 0..1 | attr | This is a keyword identifying the mathematical method to be applied for interpolation. The keyword needs to be related to the interpolation routine which needs to be invoked.<br><br>**Tags:** xml.sequenceOffset=250 |
| swIsVirtual | Boolean | 0..1 | attr | This element distinguishes virtual objects. Virtual objects do not appear in the memory, their derivation is much more dependent on other objects and hence they shall have a swDataDependency .<br><br>**Tags:** xml.sequenceOffset=260 |
| swPointerTargetProps | SwPointerTargetProps | 0..1 | aggr | Specifies that the containing data object is a pointer to another data object.<br><br>**Tags:** xml.sequenceOffset=280 |
| swRecordLayout | SwRecordLayout | 0..1 | ref | Record layout for this data object.<br><br>**Tags:** xml.sequenceOffset=290 |
| swRefreshTiming | MultidimensionalTime | 0..1 | aggr | This element specifies the frequency in which the object involved shall be or is called or calculated. This timing can be collected from the task in which write access processes to the variable run. But this cannot be done by the MCD system.<br><br>So this attribute can be used in an early phase to express the desired refresh timing and later on to specify the real refresh timing.<br><br>**Tags:** xml.sequenceOffset=300 |

| | | | | |
|---|---|---|---|---|
| swTextProps | SwTextProps | 0..1 | aggr | the specific properties if the data object is a text object.<br><br>**Tags:** xml.sequenceOffset=120 |
| swValueBlockSize | Numerical | 0..1 | attr | This represents the size of a Value Block<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime<br>xml.sequenceOffset=80 |
| unit | Unit | 0..1 | ref | Physical unit associated with the semantics of this data object. This attribute applies if no compuMethod is specified. If both units (this as well as via compuMethod) are specified the units shall be compatible.<br><br>**Tags:** xml.sequenceOffset=350 |
| valueAxisDataType | ApplicationPrimitiveDataType | 0..1 | ref | The referenced ApplicationPrimitiveDataType represents the primitive data type of the value axis within a compound primitive (e.g. curve, map). It supersedes CompuMethod, Unit, and BaseType.<br><br>**Tags:** xml.sequenceOffset=355 |

**Table A.71: SwDataDefProps**

| ***Enumeration*** | **SwImplPolicyEnum** |
|---|---|
| ***Package*** | M2::MSR::DataDictionary::DataDefProperties |
| ***Note*** | Specifies the implementation strategy with respect to consistency mechanisms of variables. |
| ***Literal*** | ***Description*** |
| const | forced implementation such that the running software within the ECU shall not modify it. For example implemented with the "const" modifier in C. This can be applied for parameters (not for those in NVRAM) as well as argument data prototypes.<br><br>**Tags:** atp.EnumerationValue=0 |
| fixed | This data element is fixed. In particular this indicates, that it might also be implemented e.g. as in place data, (#DEFINE).<br><br>**Tags:** atp.EnumerationValue=1 |
| measurementPoint | The data element is created for measurement purposes only. The data element is never read directly within the ECU software. In contrast to a "standard" data element in an unconnected provide port is, this unconnection is guaranteed for measurementPoint data elements.<br><br>**Tags:** atp.EnumerationValue=2 |
| queued | The content of the data element is queued and the data element has 'event' semantics, i.e. data elements are stored in a queue and all data elements are processed in 'first in first out' order. The queuing is intended to be implemented by RTE Generator. This value is not applicable for parameters.<br><br>**Tags:** atp.EnumerationValue=3 |

| | |
|---|---|
| standard | This is applicable for all kinds of data elements. For variable data prototypes the 'last is best' semantics applies. For parameter there is no specific implementation directive.<br><br>**Tags:** atp.EnumerationValue=4 |

**Table A.72: SwImplPolicyEnum**

| ***Class*** | **SwcImplementation** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::SwcImplementation | | | |
| ***Note*** | This meta-class represents a specialization of the general Implementation meta-class with respect to the usage in application software.<br><br>**Tags:** atp.recommendedPackage=SwcImplementations | | | |
| ***Base*** | ARElement, ARObject, CollectableElement, Identifiable, Implementation, MultilanguageReferrable, PackageableElement, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| behavior | SwcInternalBehavior | 1 | ref | The internal behavior implemented by this Implementation. |
| perInstanceMemorySize | PerInstanceMemorySize | * | aggr | Allows a definition of the size of the per-instance memory for this implementation. The aggregation of PerInstanceMemorySize is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects, in this case PerInstanceMemory.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=preCompileTime |
| requiredRTEVendor | String | 0..1 | attr | Identify a specific RTE vendor. This information is potentially important at the time of integrating (in particular: linking) the application code with the RTE. The semantics is that (if the association exists) the corresponding code has been created to fit to the vendor-mode RTE provided by this specific vendor. Attempting to integrate the code with another RTE generated in vendor mode is in general not possible. |

**Table A.73: SwcImplementation**

| ***Class*** | **SwcInternalBehavior** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior | | | |
| ***Note*** | The SwcInternalBehavior of an AtomicSwComponentType describes the relevant aspects of the software-component with respect to the RTE, i.e. the RunnableEntities and the RTEEvents they respond to. | | | |
| ***Base*** | ARObject, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, InternalBehavior, MultilanguageReferrable, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |

| | | | | |
|---|---|---|---|---|
| arTypedPerInstanceMemory | VariableDataPrototype | * | aggr | Defines an AUTOSAR typed memory-block that needs to be available for each instance of the SW-component.<br><br>This is typically only useful if supportsMultipleInstantiation is set to "true" or if the component defines NVRAM access via permanent blocks.<br><br>The aggregation of arTypedPerInstanceMemory is subject to variability with the purpose to support variability in the software component's implementations. Typically different algorithms in the implementation are requiring different number of memory objects.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| event | RTEEvent | * | aggr | This is a RTEEvent specified for the particular SwcInternalBehavior.<br><br>The aggregation of RTEEvent is subject to variability with the purpose to support the conditional existence of RTE events. Note: the number of RTE events might vary due to the conditional existence of PortPrototypes using DataReceivedEvents or due to different scheduling needs of algorithms.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| exclusiveAreaPolicy | SwcExclusiveAreaPolicy | * | aggr | Options how to generate the ExclusiveArea related APIs. When no SwcExclusiveAreaPolicy is specified for an ExclusiveArea the default values apply.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=exclusiveAreaPolicy<br>vh.latestBindingTime=preCompileTime |

| | | | | |
|---|---|---|---|---|
| explicitInterRunnableVariable | VariableDataPrototype | * | aggr | Implement state message semantics for establishing communication among runnables of the same component. The aggregation of explicitInterRunnableVariable is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| handleTerminationAndRestart | HandleTerminationAndRestartEnum | 1 | attr | This attribute controls the behavior with respect to stopping and restarting. The corresponding AtomicSwComponentType may either not support stop and restart, or support only stop, or support both stop and restart. |
| implicitInterRunnableVariable | VariableDataPrototype | * | aggr | Implement state message semantics for establishing communication among runnables of the same component. The aggregation of implicitInterRunnableVariable is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| includedDataTypeSet | IncludedDataTypeSet | * | aggr | The includedDataTypeSet is used by a software component for its implementation.<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=includedDataTypeSet |
| includedModeDeclarationGroupSet | IncludedModeDeclarationGroupSet | * | aggr | This aggregation represents the included ModeDeclarationGroups<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=includedModeDeclarationGroupSet |

| | | | | |
|---|---|---|---|---|
| instantiationDataDefProps | InstantiationDataDefProps | * | aggr | The purpose of this is that within the context of a given SwComponentType some data def properties of individual instantiations can be modified. The aggregation of InstantiationDataDefProps is subject to variability with the purpose to support the conditional existence of PortPrototypes and component local memories like "perInstanceParameter" or "arTypedPerInstanceMemory".<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=instantiationDataDefProps, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| perInstanceMemory | PerInstanceMemory | * | aggr | Defines a per-instance memory object needed by this software component. The aggregation of PerInstanceMemory is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| perInstanceParameter | ParameterDataPrototype | * | aggr | Defines parameter(s) or characteristic value(s) that needs to be available for each instance of the software-component. This is typically only useful if supportsMultipleInstantiation is set to "true". The aggregation of perInstanceParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |
| portAPIOption | PortAPIOption | * | aggr | Options for generating the signature of port-related calls from a runnable to the RTE and vice versa. The aggregation of PortPrototypes is subject to variability with the purpose to support the conditional existence of ports.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=portAPIOption, variationPoint.shortLabel<br>vh.latestBindingTime=preCompileTime |

| | | | | |
|---|---|---|---|---|
| runnable | RunnableEntity | * | aggr | This is a RunnableEntity specified for the particular SwcInternalBehavior.<br><br>The aggregation of RunnableEntity is subject to variability with the purpose to support the conditional existence of RunnableEntities. Note: the number of RunnableEntities might vary due to the conditional existence of PortPrototypes using DataReceivedEvents or due to different scheduling needs of algorithms.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variation Point.shortLabel<br>vh.latestBindingTime=preCompileTime |
| serviceDep endency | SwcServiceDep endency | * | aggr | Defines the requirements on AUTOSAR Services for a particular item.<br><br>The aggregation of SwcServiceDependency is subject to variability with the purpose to support the conditional existence of ports as well as the conditional existence of ServiceNeeds.<br><br>The SwcServiceDependency owned by an SwcInternalBehavior can be located in a different physical file in order to support that SwcServiceDependency might be provided in later development steps or even by different expert domain (e.g OBD expert for Obd related Service Needs) tools. Therefore the aggregation is «atpSplitable».<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variation Point.shortLabel<br>vh.latestBindingTime=preCompileTime |
| sharedPar ameter | ParameterData Prototype | * | aggr | Defines parameter(s) or characteristic value(s) shared between SwComponentPrototypes of the same SwComponentType The aggregation of sharedParameter is subject to variability with the purpose to support variability in the software components implementations. Typically different algorithms in the implementation are requiring different number of memory objects.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variation Point.shortLabel<br>vh.latestBindingTime=preCompileTime |
| supportsM ultipleInsta ntiation | Boolean | 1 | attr | Indicate whether the corresponding software-component can be multiply instantiated on one ECU. In this case the attribute will result in an appropriate component API on programming language level (with or without instance handle). |

| | | | | |
|---|---|---|---|---|
| variationPointProxy | VariationPointProxy | * | aggr | Proxy of a variation points in the C/C++ implementation.<br><br>**Stereotypes:** atpSplitable<br>**Tags:** atp.Splitkey=shortName |

**Table A.74: SwcInternalBehavior**

| ***Class*** | **System** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::SystemTemplate | | | |
| ***Note*** | The top level element of the System Description. The System description defines five major elements: Topology, Software, Communication, Mapping and Mapping Constraints.<br><br>The System element directly aggregates the elements describing the Software, Mapping and Mapping Constraints; it contains a reference to an ASAM FIBEX description specifying Communication and Topology.<br><br>**Tags:** atp.recommendedPackage=Systems | | | |
| ***Base*** | ARElement, ARObject, AtpClassifier, AtpFeature, AtpStructureElement, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| clientIdDefinitionSet | ClientIdDefinitionSet | * | ref | Set of Client Identifiers that are used for inter-ECU client-server communication in the System. |
| containerIPduHeaderByteOrder | ByteOrderEnum | 0..1 | attr | Defines the byteOrder of the header in ContainerIPdus. |
| ecuExtractVersion | RevisionLabelString | 0..1 | attr | Version number of the Ecu Extract. |
| fibexElement | FibexElement | * | ref | Reference to ASAM FIBEX elements specifying Communication and Topology.<br><br>All Fibex Elements used within a System Description shall be referenced from the System Element.<br><br>atpVariation: In order to describe a product-line, all FibexElements can be optional.<br><br>**Stereotypes:** atpVariation<br>**Tags:** vh.latestBindingTime=postBuild |
| j1939SharedAddressCluster | J1939SharedAddressCluster | * | aggr | Collection of J1939Clusters that share a common address space for the routing of messages.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variationPoint.shortLabel<br>vh.latestBindingTime=postBuild |

| | | | | |
|---|---|---|---|---|
| mapping | SystemMapping | * | aggr | Aggregation of all mapping aspects (mapping of SW components to ECUs, mapping of data elements to signals, and mapping constraints).<br><br>In order to support OEM / Tier 1 interaction and shared development for one common System this aggregation is atpSplitable and atpVariation. The content of SystemMapping can be provided by several parties using different names for the SystemMapping.<br><br>This element is not required when the System description is used for a network-only use-case.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variation Point.shortLabel<br>vh.latestBindingTime=postBuild |
| pncVector Length | PositiveInteger | 0..1 | attr | Length of the partial networking request release information vector (in bytes). |
| pncVector Offset | PositiveInteger | 0..1 | attr | Absolute offset (with respect to the NM-PDU) of the partial networking request release information vector that is defined in bytes as an index starting with 0. |
| rootSoftwa reComposi tion | RootSwCompos itionPrototype | 0..1 | aggr | Aggregation of the root software composition, containing all software components in the System in a hierarchical structure. This element is not required when the System description is used for a network-only use-case.<br><br>atpVariation: The RootSwCompositionPrototype can vary.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variation Point.shortLabel<br>vh.latestBindingTime=systemDesignTime |
| systemDoc umentation | Chapter | * | aggr | Possibility to provide additional documentation while defining the System. The System documentation can be composed of several chapters.<br><br>**Stereotypes:** atpSplitable; atpVariation<br>**Tags:** atp.Splitkey=shortName, variation Point.shortLabel<br>vh.latestBindingTime=systemDesignTime<br>xml.sequenceOffset=-10 |
| systemVer sion | RevisionLabelSt ring | 1 | attr | Version number of the System Description. |

**Table A.75: System**

| **Class** | **TimingEvent** | | | |
|---|---|---|---|---|
| **Package** | M2::AUTOSARTemplates::SWComponentTemplate::SwcInternalBehavior::RTEEvents | | | |
| **Note** | TimingEvent references the RunnableEntity that need to be started in response to the TimingEvent | | | |
| **Base** | ARObject, AbstractEvent, AtpClassifier, AtpFeature, AtpStructureElement, Identifiable, MultilanguageReferrable, RTEEvent, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| period | TimeValue | 1 | attr | Period of timing event in seconds. The value of this attribute shall be greater than zero. |

**Table A.76: TimingEvent**

| **Class** | **Unit** | | | |
|---|---|---|---|---|
| **Package** | M2::MSR::AsamHdo::Units | | | |
| **Note** | This is a physical measurement unit. All units that might be defined should stem from SI units. In order to convert one unit into another factor and offset are defined.<br><br>For the calculation from SI-unit to the defined unit the factor (factorSiToUnit ) and the offset (offsetSiToUnit ) are applied as follows:<br>`x [{unit}] := y * [{siUnit}] * factorSiToUnit [[unit]/{siUnit}] + offsetSiToUnit [{unit}]`<br><br>For the calculation from a unit to SI-unit the reciprocal of the factor (factorSiToUnit ) and the negation of the offset (offsetSiToUnit ) are applied.<br>`y {siUnit} := (x*{unit} - offsetSiToUnit [{unit}]) / (factorSiToUnit [[unit]/{siUnit}]`<br><br>**Tags:** atp.recommendedPackage=Units | | | |
| **Base** | ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| **Attribute** | **Type** | **Mul.** | **Kind** | **Note** |
| displayName | SingleLanguageUnitNames | 0..1 | aggr | This specifies how the unit shall be displayed in documents or in user interfaces of tools.The displayName corresponds to the Unit.Display in an ASAM MCD-2MC file.<br><br>**Tags:** xml.sequenceOffset=20 |
| factorSiToUnit | Float | 0..1 | attr | This is the factor for the conversion from SI Units to units.<br><br>The inverse is used for conversion from units to SI Units.<br><br>**Tags:** xml.sequenceOffset=30 |
| offsetSiToUnit | Float | 0..1 | attr | This is the offset for the conversion from and to siUnits.<br><br>**Tags:** xml.sequenceOffset=40 |

| | | | | |
|---|---|---|---|---|
| physicalDimension | PhysicalDimension | 0..1 | ref | This association represents the physical dimension to which the unit belongs to. Note that only values with units of the same physical dimensions might be converted.<br><br>**Tags:** xml.sequenceOffset=50 |

**Table A.77: Unit**

| ***Class*** | **UnitGroup** | | | |
|---|---|---|---|---|
| ***Package*** | M2::MSR::AsamHdo::Units | | | |
| ***Note*** | This meta-class represents the ability to specify a logical grouping of units.The category denotes the unit system that the referenced units are associated to.<br><br>In this way, e.g. country-specific unit systems (CATEGORY="COUNTRY") can be defined as well as specific unit systems for certain application domains.<br><br>In the same way a group of equivalent units, can be defined which are used in different countries, by setting CATEGORY="EQUIV_UNITS". KmPerHour and MilesPerHour could such be combined to one group named "vehicle_speed". The unit MeterPerSec would not belong to this group because it is normally not used for vehicle speed. But all of the mentioned units could be combined to one group named "speed".<br><br>Note that the UnitGroup does not ensure the physical compliance of the units. This is maintained by the physical dimension.<br><br>**Tags:** atp.recommendedPackage=UnitGroups | | | |
| ***Base*** | ARElement, ARObject, CollectableElement, Identifiable, MultilanguageReferrable, PackageableElement, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| unit | Unit | * | ref | This represents one particular unit in the UnitGroup.<br><br>**Tags:** xml.sequenceOffset=20 |

**Table A.78: UnitGroup**

| ***Class*** | **ValueSpecification (abstract)** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::CommonStructure::Constants | | | |
| ***Note*** | Base class for expressions leading to a value which can be used to initialize a data object. | | | |
| ***Base*** | ARObject | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| shortLabel | Identifier | 0..1 | attr | This can be used to identify particular value specifications for human readers, for example elements of a record type. |

**Table A.79: ValueSpecification**

| ***Class*** | **VariableDataPrototype** | | | |
|---|---|---|---|---|
| ***Package*** | M2::AUTOSARTemplates::SWComponentTemplate::Datatype::DataPrototypes | | | |
| ***Note*** | A VariableDataPrototype is used to contain values in an ECU application. This means that most likely a VariableDataPrototype allocates "static" memory on the ECU. In some cases optimization strategies might lead to a situation where the memory allocation can be avoided.<br><br>In particular, the value of a VariableDataPrototype is likely to change as the ECU on which it is used executes. | | | |
| ***Base*** | ARObject, AtpFeature, AtpPrototype, AutosarDataPrototype, DataPrototype, Identifiable, MultilanguageReferrable, Referrable | | | |
| ***Attribute*** | ***Type*** | ***Mul.*** | ***Kind*** | ***Note*** |
| initValue | ValueSpecification | 0..1 | aggr | Specifies initial value(s) of the VariableDataPrototype |

**Table A.80: VariableDataPrototype**