Document Title	Specification of AUTOSAR Run-Time Interface
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	923

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History			
Date	**Release**	**Changed by**	**Description**
2018-10-31	R4.4.0	AUTOSAR Release Management	• Initial release

Disclaimer

Table of Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Run-Time Interface (“ARTI”) for debugging and tracing AUTOSAR modules. Currently, the specification is in state “draft”.

ARTI defines an interface between build tools and debugging/tracing tools. The debugging/tracing tools shall then forward tracing information to trace/timing analysis tools. The interface shall ease and speed up the debugging, tracing and verification of system behavior as well as round-trip engineering.

Debugging and tracing enables efficient development, integration, optimization and verification of ECU software. For analyzing several aspects - especially timing aspects - it becomes essential to link the debugging and tracing data to the scheduling of an ECU. Knowledge about tasks, interrupts and runnables, in other words: awareness of the operating system (“OS awareness”), is required.

A good interaction of the tool chain provides complete round-trip engineering from model down to hardware and back - covering several software levels and several phases of the V-model.

ARTI shall especially provide

- Support of “OS Awareness”, for example examination of OS specific tasks, threads etc.
- Support of distributes systems and multi-core
- Support of other AUTOSAR modules (e.g. RTE in CP or ARA in AP)
- Support of instrumentation-based tracing and measurement solutions
- Support of TIMEX

The data flow of the tools and the interfaces of ARTI are depicted in figure 1.1.

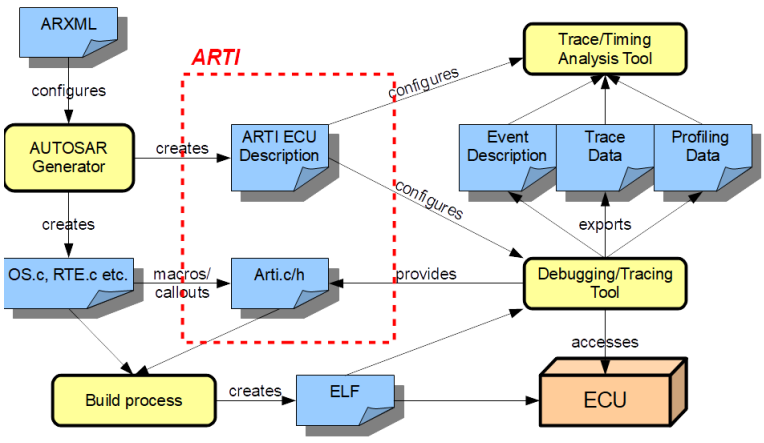

Figure 1.1: ARTI data flow

For some important definitions please read also chapter 1 of RS_FoundationARTI_915.

To implement the features, ARTI uses a similar approach that the former OSEK-ORTI had, but extends this to current requirements. The tools that generate AUTOSAR modules (e.g. OS, RTE, etc.) have to extend the ECU configuration with internal information about this module and emit the extended configuration as a separate file (“ARTI file”). The information therein shall allow to debug and trace the behavior of this module. Additional tools will collect all ARTI files of an ECU and allow selecting specific items to trace and create tracing hook files for a specific trace channel (e.g. internal buffer, hardware trace buffers, etc.). The build environment creates the final application, which then can be used in the ECU. Debugging and tracing tools can read in the ARTI files and are “AUTOSAR aware”, giving additional debugging and tracing features to the developer. These tools can export a trace file, which in turn can be used in trace analysis tools for extended timing analysis, time measurements and optimization runs.

Using the standardized work flow allows interchanging the tools as necessary, and use the tool that fits best for each solution without the need of adapting the work flow.

The work flow of the ARTI file generation and usage is depicted in figure 1.2. ARTI shall only define interfaces within the build process of an AUTOSAR application (i.e. the export of the generators, and the hooks within the AUTOSAR modules). The interfaces for tool communication are post-build and not subject to this specification.

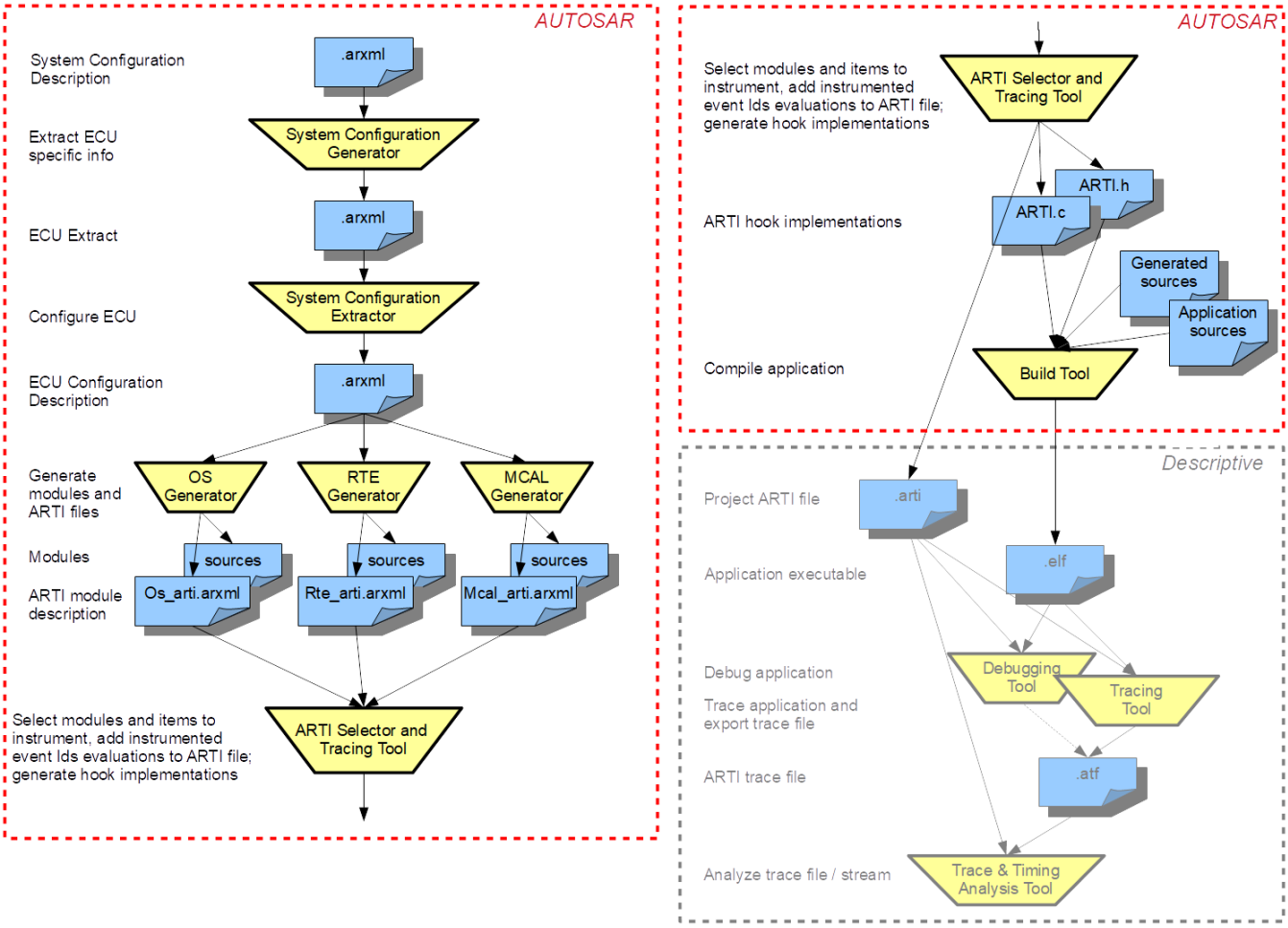

Figure 1.2: ARTI work flow

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the ARTI module that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
ORTI	"OSEK Run Time Interface", an OSEK specification (in its version 2.2) that defines how debuggers can access OSEK OS internal information.

Terms:	Description:
Debugging	"Debugging" refers to halting a system, either as a whole or in parts, for the purpose of • inspecting the contents of the system in a frozen state • single stepping, setting breakpoints, starting and stopping in C or Assembly code

Terms:	Description:
Tracing	"Tracing" refers to collecting run-time information over a certain period of time • either as a pure software solution, or with hardware assistance • may include processor instruction trace, OS scheduling trace, and/or pure data trace • including time-stamping for further timing analysis
Timing Measurement	"Timing Measurement" refers to capturing of timing information • by instrumentation, e.g. via Pre-/PostTaskHooks or other hooks or callouts or • by dedicated hardware support, e.g. hardware performance counters • does not stop execution
Profiling	"Profiling" refers to the process of gaining timing parameters/timing statistics • of functions, tasks, runnables, modules etc. • possibly with minimum/maximum/average statistics • possibly with worst case analysis • possibly calculated out of trace data, repeated snapshots or Timing Measurement

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_TR_Glossary

3.2 Related specification

Not applicable yet.

4 Constraints and assumptions

The ARTI concept expects to get an own ARTI module description from each module to be debugged, e.g. OS and RTE. This allows mixing modules with ARTI support

with those without ARTI support. However, as ARTI contains internal information, the implementers of the modules have to provide the ARTI file.

4.1 Limitations

ARTI is supposed to work with debug information created by the compilers. This means each module that supports ARTI needs to be compiled with debug information, and the ARTI file has to use the symbol names created by the compiler.

ARTI introduces new hooks. In order to use them, they shall be incorporated into the module's C code. Either they are put therein statically, or they have to be configured.

Tracing internal events is very time critical. ARTI focuses on the solutions with the least impact on timing (in some cases with no timing overhead at all), but this depends on the hardware capabilities of the ECU and the tools. ARTI provides examples that describe the possibilities for tracing, depending on the available hardware and software capabilities.

4.2 Applicability to car domains

ARTI is explicitly designed to be applicable to any car domain.

5 Dependencies to other modules

...

6 Requirements Tracing

The following tables reference the requirements specified in <CITATIONS_OF_CONTRIBUTED_DOCUMENTS> and links to the fulfillment of these. Please note that if column "Satisfied by" is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[Req_Id_0]	No description	[SWS_XYZ_00002]
[Req_Id_1]	No description	[SWS_XYZ_00999]
[Req_Id_2]	No description	[SWS_XYZ_00999]
[Req_Id_3]	No description	[SWS_XYZ_00999]

7 Functional specification

As shown in figure 1.1, ARTI consists of these functional elements:

- ARTI module description
- ARTI hook implementations

The “ARTI Module Description” is intended to be emitted as an ARXML file. Additional files, such as the “project ARTI file” or “ARTI trace file” may be stored in another file format, whereas this format is beyond AUTOSAR and defined elsewhere.

ARTI is not a traditional software module that creates code and changes the system behavior. Instead ARTI is explicitly designed to *not* affect the overall system behavior. Especially the generation and export of the ARTI module description is intended to not influence the module that generates the ARTI export; ARTI should export information that is already internally available. The exported information will then be post-processed and used by further debugging and tracing tools. However, it might be necessary to introduce some special variables or functions to be able to generate requested information. While this causes some slight impact to the code, it is again the intention not to change the overall behavior of the module using ARTI. The same applies to the hooks: while the hooks itself may have some slight impact on the code base and while the hook implementation (done by the tools consuming ARTI) may have some impact on the timing and on the program flow, it is the intention of ARTI to change the module behavior as little as possible – ideally not at all.

ARTI must be applicable “on the road” – this obviously comes with high safety requirements regarding the implementation of the hooks since e.g. some of the ARTI hooks will be executed in the context of the OS. Special care has to be taken in a multi-core context.

If the implementation of the hooks cannot guarantee safe execution, the ECU must not be used “on the road”. “On the road” here refers to situations where the operation or malfunction might cause danger to persons or property.

7.1 ARTI Module Description

An "ARTI Module Description" is an ARXML file that contains detailed information about a specific module (e.g. OS, RTE, etc.). In particular, this is:

- Constants
  A Constant defines a constant value that is specific to this application or environment. E.g. the number of CPUs used in an ECU could be defined as a constant. Constants are used by a debugger to know about the configuration, or to display the value in a convenient way.
  Constants are referred to by an object information (see "Object Information" in chapter 7.1) and are only meaningful in the context of an object.
  A Constant is represented by the container ArtiConstant (see chapter 10.2.1).

- Expressions
  An Expression defines how a specific value can be accessed on the target by a debugger to display a current state of the application. Expressions are like C expression but limited so that they can be evaluated statically. Hence only accesses to global variables are allowed, and only unary, binary and trinary oparators are allowed. Especially accesses to local variables and calls to functions are not allowed.
  This is similar to what is already done in ORTI.
  Expressions are referred to by an object information (see "Object Information" in chapter 7.1) and are used to define the evaluation of parameters values therein.
  An Expression is represented by the container ArtiExpression (see chapter 10.2.2).
- Hook definitions
  Hook definitions contain information about which hooks are present in the module and how they look like. These hook definitions are used to create the hook implementation and to trace the information defined by the hook.
  A Hook definition is represented by the container ArtiHook (see chapter 10.2.3).
- Object information
  Objects within a module (e.g. an “OsTask”) get an own representation in the ARTI module description. The object information contains references to the original object as well as references to the expressions and hooks used for this object. All objects of a specific kind are collected in a container. The detailed layout of an object within a specific module is defined in the according SWS.
- Generic components
  ARTI is able to define objects that should show up in a debugger or when tracing, even if those are not standard AUTOSAR objects (e.g. user defined, or additional OS features like semaphores). See chapter 10.3.

7.2 ARTI Hook Implementation

The ARTI hook implementations are generated by a tool that consumes the ARTI description files. They are mainly represented by two files:

- ARTI.h
  This file contains all macros that are used in the modules supporting ARTI to instrument certain events. It may also contain the implementation of the macro, or may refer to an implementation in ARTI.c.
- ARTI.c
  This file contains the actual implementation of each macro, if it is not empty or not implemented in the ARTI.h file.

All events that are not active will be mapped to an empty macro definition. All events that are active will be expanded to the implementation of the instrumentation. The

actual implementation depends on the hardware and software capabilities of the tracing tool. Thus, it depends on the used tracing tool, how the macros are implemented. ARTI shall provide examples for standard ways of tracing in hardware or software.

8 API specification

8.1 Imported types

This section lists all imported types used by the API. Even if ARTI does not require new types, some RTE or Component types can be used within the configuration of the hook functions. Therefore ARTI also has the standardized include structure (see SRS_BSW_00447) for modules with service interfaces.

[SWS_XYZ_00002] ⌈ ⌋*(Req_Id_0)*

8.2 Type definitions

ARTI does not add any type definitions.

8.3 Function definitions

ARTI does not add any functions.

8.4 Callback notifications

ARTI does not provide any callback functions.

8.5 Scheduled functions

ARTI does not have any functions directly called by Basic Software Scheduler.

8.6 Expected interfaces

In this chapter all interfaces required from other modules are listed.

8.6.1 Mandatory interfaces

8.6.1.1 ARTI Tracing Macro

There is only one ARTI macro with a set of parameters which define the semantic of the macro. This macro is used by all modules with ARTI trace capabilities, therefore ARTI based instrumentation can easily be disabled on a global level.

```
ARTI_TRACE(_contextName, _className, _instanceName,
instanceParameter, _eventName, eventParameter)
```

Some of the parameters come as tokens (literal text) rather than as symbolic identifiers. This allows a macro definition to concatenate these parameters to more specific and efficient macros. Passing and evaluating all parameters as symbolic identifiers at run-time would be very costly especially by means of run-time consumption.

Here is a possible implementation of the generic `ARTI_TRACE` macro:

```
1 #define ARTI_TRACE( _contextName, _className, \
2                     instance_id, _eventName, event_value) \
3     ARTI_TRACE ## _ ## _className ## _ ## _eventName \
4     ## _ ## _contextName ( (instance_id), (event_value) )
```

Such an implementation will generate one hook for all the possible combinations of `_className`, `_eventName`, `_instanceName` and `_contextName` and pass parameters `instance_id` and `event_value` at run-time only. The parameters' meanings are described in the following.

`_contextName` Token, literal text, name of the context. One of the following:

- **`NOSUSP`** indicating that the hook gets called in a context where interrupts are disabled
- **`SPRVSR`** indicating that the called hook may disable interrupts
- **`USER`** indicating the called hook cannot disable interrupts

`_className` Token, literal text, name of the class of macros. Classes can be one of the predefined classes (e.g. `AR_OS`) in this document or user defined.

`_instanceName` Name of an instance

`instanceParameter` Index [uint32] 0..4294967295 of the instance of a particular `_className` and `_instanceName`, the index should start with 0 and be consecutive.

`_eventName` Token, literal text, name of the event as defined for a particular class (e.g. `OsTask_Start`).

`eventParameter` A [uint32] 0..4294967295 value as an argument to an event (e.g. Task Index).

All modules which shall support ARTI tracing shall add calls to this macro with the module specific parameters.

The parameters that are marked as *token, literal text* can't be:

- C macros
- variables
- constants
- enumerations

These parameters are meant to be subject of *token concatenation* by the C preprocessor or the trace tool provider (provider of *ARTI.h*) chooses to map these tokens to symbols within *ARTI.h* depending on the trace tool.

Examples:

1 OS on 2 cores the OS short name is *OsA*, the OS manages three physical CPU cores.

- `ARTI_TRACE(NOSUSP, AR_TASK_SCHEDULER, OsA, 0, OsTask_Start, 0); /* OS OsA start of Task with index 0 on it's own Core 0 */`
- `ARTI_TRACE(NOSUSP, AR_TASK_SCHEDULER, OsA, 1, OsTask_Start, 0); /* OS OsA start of Task with index 0 on it's own Core 1 */`

2 OSs on 1 physical core the OS short names are *OsA* and *OsB*, both run on the same physical CPU core (e.g. Hypervisor)

- `ARTI_TRACE(NOSUSP, AR_TASK_SCHEDULER, OsA, 0, OsTask_Start, 0); /* OS OsA start of Task with index 0 on it's own Core 0 */`
- `ARTI_TRACE(NOSUSP, AR_TASK_SCHEDULER, OsB, 0, OsTask_Start, 0); /* OS OsB start of Task with index 0 on it's own Core 0 */`

2 OSs on 4 cores the OS short names are *OsA* and *OsB* each OS manages two physical CPU cores.

- `ARTI_TRACE(NOSUSP, AR_TASK_SCHEDULER, OsA, 0, OsTask_Start, 0); /* OS OsA start of Task with index 0 on it's own Core 0 */`
- `ARTI_TRACE(NOSUSP, AR_TASK_SCHEDULER, OsA, 1, OsTask_Start, 0); /* OS OsA start of Task with index 0 on it's own Core 1 */`
- `ARTI_TRACE(NOSUSP, AR_TASK_SCHEDULER, OsB, 0, OsTask_Start, 0); /* OS OsB start of Task with index 0 on it's own Core 0 */`

- `ARTI_TRACE(NOSUSP, AR_TASK_SCHEDULER, OsB, 1, OsTask_Start, 0); /* OS OsB start of Task with index 0 on it's own Core 1 */`

2 OSs, 2 virtual cores each and 3 physical cores the OS short names are *OsA* and *OsB* each OS manages two virtual CPU cores (e.g. Hypervisor manages the three physical CPU cores).

- `ARTI_TRACE(NOSUSP, AR_TASK_SCHEDULER, OsA, 0, OsTask_Start, 0); /* OS OsA start of Task with index 0 on it's own Core 0 */`
- `ARTI_TRACE(NOSUSP, AR_TASK_SCHEDULER, OsA, 1, OsTask_Start, 0); /* OS OsA start of Task with index 0 on it's own Core 1 */`
- `ARTI_TRACE(NOSUSP, AR_TASK_SCHEDULER, OsB, 0, OsTask_Start, 0); /* OS OsB start of Task with index 0 on it's own Core 0 */`
- `ARTI_TRACE(NOSUSP, AR_TASK_SCHEDULER, OsB, 1, OsTask_Start, 0); /* OS OsB start of Task with index 0 on it's own Core 1 */`

`AMODULE`, a defined class with a single instance called `AModule1`.

- `ARTI_TRACE(SPRVSR, AMODULE, AModule1, 0, Thing_Start, 123);`

8.6.2 Optional interfaces

ARTI does not define optional interfaces.

8.6.3 Configurable interfaces

ARTI does not define configurable interfaces.

8.7 Service Interfaces

ARTI does not provide any service interfaces.

9 Sequence diagrams

Not applicable yet.

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters used by the ARTI containers defined in chapters 10.3 and in the SWS documents of other modules.

Chapter 10.3 specifies the structure (containers) and the parameters of generic (i.e. vendor or user specific) ARTI objects.

Chapter 10.4 specifies published information of ARTI.

Containers and parameters that are related to the OS module are specified in SWS_OS, chapter "Containers and configuration parameters for ARTI".

Containers and parameters that are related to the RTE module are specified in SWS_RTE, chapter "Configuration of ARTI Information".

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in SWS_BSWGeneral.

10.2 ARTI Parameters

Module SWS Item	**ARTI_Arti_xxxx**
Module Name	Arti
Module Description	The “Arti” module specifies all configuration parameters used by the ARTI containers "ArtiGenericComponentClass" and "ArtiGenericComponentInstance" and SWS documents of other modules.
Post-Build Variant Support	true
Supported Config Variangs	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE
Included Containers	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
ArtiConstant	0..*	This container holds a constant value.
ArtiExpression	0..*	This container holds a C like expression that a debugger can evaluate. This is similar to what is already done in ORTI.
ArtiHook	0..*	This container represents an ARTI hook that is present in the module.
ArtiObjectClassParameter	0..*	This container represents a parameter of an Arti object class definition.
ArtiObjectInstanceParameter	0..*	This container represents a parameter of an Arti object instance.
ArtiParameterTypeMap	0..*	This container represents a map of key/value pairs to map a parameter value to a display string.

Example 10.1

Examplary Values of the Arti Container

```
<AUTOSAR>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>Vendor1</SHORT-NAME>
      <ELEMENTS>
        <ECUC-MODULE-CONFIGURATION-VALUES>
          <SHORT-NAME>Vendor1Arti</SHORT-NAME>
          <DEFINITION-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/ArtiDefs/Arti
              </DEFINITION-REF>
          <CONTAINERS>
            <ECUC-CONTAINER-VALUE>
              <SHORT-NAME>ArtiConstant_ArtiSwc_WiperLocation_Front</
                  SHORT-NAME>
              <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
                  AUTOSAR/ArtiDefs/Arti/ArtiConstant</DEFINITION-REF>
              <...>
            </ECUC-CONTAINER-VALUE>
            <ECUC-CONTAINER-VALUE>
              <SHORT-NAME>ArtiExpression_ArtiHwCore_CurrentTaskOnCore0<
                  /SHORT-NAME>
              <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
                  AUTOSAR/ArtiDefs/Arti/ArtiExpression</DEFINITION-REF>
              <...>
            </ECUC-CONTAINER-VALUE>
            <ECUC-CONTAINER-VALUE>
              <SHORT-NAME>ArtiHook_ArtiOs_TaskStart</SHORT-NAME>
              <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
                  AUTOSAR/ArtiDefs/Arti/ArtiHook</DEFINITION-REF>
              <...>
```

```
        </ECUC-CONTAINER-VALUE>
        <ECUC-CONTAINER-VALUE>
          <SHORT-NAME>
              ArtiObjectClassParameter_ArtiHwCore_CurrentApplication
              </SHORT-NAME>
          <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
              AUTOSAR/ArtiDefs/Arti/ArtiObjectClassParameter</
              DEFINITION-REF>
          <...>
        </ECUC-CONTAINER-VALUE>
        <ECUC-CONTAINER-VALUE>
          <SHORT-NAME>
              ArtiObjectInstanceParameter_CurrentApplicationOnCore0
              </SHORT-NAME>
          <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
              AUTOSAR/ArtiDefs/Arti/ArtiObjectInstanceParameter</
              DEFINITION-REF>
          <...>
        </ECUC-CONTAINER-VALUE>
        <ECUC-CONTAINER-VALUE>
          <SHORT-NAME>ArtiParameterTypeMap_Core</SHORT-NAME>
          <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
              AUTOSAR/ArtiDefs/Arti/ArtiParameterTypeMap</
              DEFINITION-REF>
          <...>
        </ECUC-CONTAINER-VALUE>
      </CONTAINERS>
    </ECUC-MODULE-CONFIGURATION-VALUES>
    <...>
```

10.2.1 ArtiConstant

SWS Item	**ARTI_Arti_xxxx**
Container Name	ArtiConstant
Description	The ArtiConstant container holds a constant value. See "Constants" in chapter "Functional specification".
Post-Build Variant Multiplicity	False
Configuration Parameters	

Name	ArtiConstantString
Description	This is the constant value for a specific parameter
Type	EcucStringParamDef

▽

△

Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Example 10.2

Examplary Value of an ArtiConstant Container

```
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiConstant_ArtiSwc_WiperLocation_Front</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
     ArtiDefs/Arti/ArtiConstant</DEFINITION-REF>
  <PARAMETER-VALUES>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
         Arti/ArtiConstant/ArtiConstantString</DEFINITION-REF>
      <VALUE>Front</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
  </PARAMETER-VALUES>
</ECUC-CONTAINER-VALUE>
```

10.2.2 ArtiExpression

SWS Item	**ARTI_Arti_xxxx**
Container Name	ArtiExpression
Description	This container holds a C like expression that a debugger can evaluate. See "Expressions" in chapter "Functional specification".
Post-Build Variant Multiplicity	False
Configuration Parameters	

Name	ArtiExpressionString
Description	This string represents a C like expresion that a debugger can evaluate.
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Example 10.3

Examplary Value of an ArtiExpression Container

```
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiExpression_ArtiHwCore_CurrentTaskOnCore0</SHORT-
     NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
     ArtiDefs/Arti/ArtiExpression</DEFINITION-REF>
  <PARAMETER-VALUES>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
         Arti/ArtiExpression/ArtiExpressionString</DEFINITION-REF>
      <VALUE>Os_ControlledCoreInfo[0U].RunningTask</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
  </PARAMETER-VALUES>
</ECUC-CONTAINER-VALUE>
```

10.2.3 ArtiHook

SWS Item	**ARTI_Arti_xxxx**
Container Name	ArtiHook
Description	This container represents an ARTI hook that is present in the module. See "Hook definitions" in chapter "Functional specification".

▽

△

Post-Build Variant Multiplicity	False
Configuration Parameters	

Name	ArtiHookClass
Description	Name of the (schedule) class of macros. Classes can be one of the predefined classes or user defined.
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiHookEventName
Description	The name of the event as defined for a particular class, or an arbitrary name for generic classes.
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiHookEventParameterTypeRef
Description	Refers to a parameter type to interpret the hook event number.
Type	Reference to ArtiParameterTypeMap
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiHookInstance
Description	Name of an instance of the (schedule) class.
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiHookInstanceParameterTypeRef
Description	Refers to a parameter type to interpret the hook instance number.
Type	Reference to ArtiParameterTypeMap
Unit	–
Range	–
Post-Build Variant Multiplicity	False

▽

△

Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Example 10.4

Examplary Value of an ArtiHook Container

```
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiHook_ArtiOs_TaskStart</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
     ArtiDefs/Arti/ArtiHook</DEFINITION-REF>
  <PARAMETER-VALUES>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
         Arti/ArtiHook/ArtiHookClass</DEFINITION-REF>
      <VALUE>AR_TASK_SCHEDULER</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
         Arti/ArtiHook/ArtiHookEventName</DEFINITION-REF>
      <VALUE>OsTask_Start</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
         Arti/ArtiHook/ArtiHookInstance</DEFINITION-REF>
      <VALUE>Vendor1OsCore</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
  </PARAMETER-VALUES>
  <REFERENCE-VALUES>
    <ECUC-REFERENCE-VALUE>
      <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs/
         Arti/ArtiHook/ArtiHookEventParameterTypeRef</DEFINITION-REF
         >
      <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti/
         ArtiParameterTypeMap_TaskId</VALUE-REF>
    </ECUC-REFERENCE-VALUE>
    <ECUC-REFERENCE-VALUE>
      <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs/
         Arti/ArtiHook/ArtiHookInstanceParameterTypeRef</DEFINITION-
         REF>
      <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti/
         ArtiParameterTypeMap_Core</VALUE-REF>
    </ECUC-REFERENCE-VALUE>
  </REFERENCE-VALUES>
</ECUC-CONTAINER-VALUE>
```

10.2.4 ArtiObjectClassParameter

SWS Item	**ARTI_Arti_xxxx**
Container Name	ArtiObjectClassParameter
Description	Represents a parameter of an Arti object class.
Post-Build Variant Multiplicity	False
Configuration Parameters	

Name	ArtiObjectClassParameterDescription
Description	Description of the Parameter.
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiParameterTypeMapRef
Description	Refers to a parameter type to interpret the instance parameter value.
Type	Reference to ArtiParameterTypeMap
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Example 10.5

Examplary Value of an ArtiObjectClassParameter Container

```
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiObjectClassParameter_ArtiHwCore_CurrentTask</SHORT-
     NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
     ArtiDefs/Arti/ArtiObjectClassParameter</DEFINITION-REF>
  <PARAMETER-VALUES>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
         Arti/ArtiObjectClassParameter/
         ArtiObjectClassParameterDescription</DEFINITION-REF>
      <VALUE>Current Running AUTOSAR Task</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
  </PARAMETER-VALUES>
  <REFERENCE-VALUES>
    <ECUC-REFERENCE-VALUE>
      <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs/
         Arti/ArtiObjectClassParameter/ArtiParameterTypeMapRef</
         DEFINITION-REF>
      <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti/
         ArtiParameterTypeMap_TaskExpr</VALUE-REF>
    </ECUC-REFERENCE-VALUE>
  </REFERENCE-VALUES>
</ECUC-CONTAINER-VALUE>
```

10.2.5 ArtiObjectInstanceParameter

SWS Item	**ARTI_Arti_xxxx**
Container Name	ArtiObjectInstanceParameter
Description	Represents a parameter of an Arti object instance.
Post-Build Variant Multiplicity	False
Configuration Parameters	

Name	ArtiConstantRef
Description	Refers to a constant representing the value of this parameter.
Type	Reference to ArtiConstant
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False

▽

△

Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiExpressionRef
Description	Refers to an expression that evaluates the value of this parameter.
Type	Reference to ArtiExpression
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiHookRef
Description	Refers to a hook that records this parameter.
Type	Reference to ArtiHook
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Example 10.6

Examplary Value of an ArtiObjectInstanceParameter Container

```
<ECUC-CONTAINER-VALUE>
```

```
<SHORT-NAME>ArtiObjectInstanceParameter_CurrentTaskOnCore0</SHORT-
    NAME>
<DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
    ArtiDefs/Arti/ArtiObjectInstanceParameter</DEFINITION-REF>
<REFERENCE-VALUES>
  <ECUC-REFERENCE-VALUE>
    <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs/
        Arti/ArtiObjectInstanceParameter/ArtiExpressionRef</
        DEFINITION-REF>
    <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti/
        ArtiExpression_ArtiHwCore_CurrentTaskOnCore0</VALUE-REF>
  </ECUC-REFERENCE-VALUE>
</REFERENCE-VALUES>
</ECUC-CONTAINER-VALUE>
```

10.2.6 ArtiParameterTypeMap

SWS Item	**ARTI_Arti_xxxx**
Container Name	ArtiParameterTypeMap
Description	A map of key/value pairs to map a parameter value to a display string.
Post-Build Variant Multiplicity	False
Included Containers	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
ArtiParameterTypeMapPair	1..*	A key/value pair to map a parameter value to a display string.

SWS Item	**ARTI_Arti_xxxx**
Container Name	ArtiParameterTypeMapPair
Description	A key/value pair to map a parameter value to a display string.
Post-Build Variant Multiplicity	False
Configuration Parameters	

Name	Input
Description	The value given by a parameter to translate.
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	Output
Description	The string to display for the Input value.
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Example 10.7

Examplary Value of an ArtiParameterTypeMap Container

```
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiParameterTypeMap_TaskId</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
     ArtiDefs/Arti/ArtiParameterTypeMap</DEFINITION-REF>
  <SUB-CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>TaskHighPrio</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
         ArtiDefs/Arti/ArtiParameterTypeMap/ArtiParameterTypeMapPair
         </DEFINITION-REF>
```

```
        <PARAMETER-VALUES>
        <ECUC-TEXTUAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
              ArtiDefs/Arti/ArtiParameterTypeMap/
              ArtiParameterTypeMapPair/Input</DEFINITION-REF>
          <VALUE>1</VALUE>
        </ECUC-TEXTUAL-PARAM-VALUE>
        <ECUC-TEXTUAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
              ArtiDefs/Arti/ArtiParameterTypeMap/
              ArtiParameterTypeMapPair/Output</DEFINITION-REF>
          <VALUE>HighPriority</VALUE>
        </ECUC-TEXTUAL-PARAM-VALUE>
      </PARAMETER-VALUES>
    </ECUC-CONTAINER-VALUE>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>TaskLowPrio</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
          ArtiDefs/Arti/ArtiParameterTypeMap/ArtiParameterTypeMapPair
          </DEFINITION-REF>
      <PARAMETER-VALUES>
        <ECUC-TEXTUAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
              ArtiDefs/Arti/ArtiParameterTypeMap/
              ArtiParameterTypeMapPair/Input</DEFINITION-REF>
          <VALUE>2</VALUE>
        </ECUC-TEXTUAL-PARAM-VALUE>
        <ECUC-TEXTUAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
              ArtiDefs/Arti/ArtiParameterTypeMap/
              ArtiParameterTypeMapPair/Output</DEFINITION-REF>
          <VALUE>LowPriority</VALUE>
        </ECUC-TEXTUAL-PARAM-VALUE>
      </PARAMETER-VALUES>
    </ECUC-CONTAINER-VALUE>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>idle</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
          ArtiDefs/Arti/ArtiParameterTypeMap/ArtiParameterTypeMapPair
          </DEFINITION-REF>
      <PARAMETER-VALUES>
        <ECUC-TEXTUAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
              ArtiDefs/Arti/ArtiParameterTypeMap/
              ArtiParameterTypeMapPair/Input</DEFINITION-REF>
          <VALUE>0</VALUE>
        </ECUC-TEXTUAL-PARAM-VALUE>
        <ECUC-TEXTUAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
              ArtiDefs/Arti/ArtiParameterTypeMap/
              ArtiParameterTypeMapPair/Output</DEFINITION-REF>
          <VALUE>idle</VALUE>
        </ECUC-TEXTUAL-PARAM-VALUE>
      </PARAMETER-VALUES>
    </ECUC-CONTAINER-VALUE>
  </SUB-CONTAINERS>
```

```
</ECUC-CONTAINER-VALUE>
```

10.3 ARTI Generic Container

SWS Item	**ARTI_ArtiGeneric_xxxx**
Container Name	ArtiGeneric
Description	The “ArtiGeneric” container contains definitions for generic objects, i.e. not belonging to a standard AUTOSAR module.
Post-Build Variant Multiplicity	False
Included Containers	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
ArtiGenericComponentClass	0..*	The "class" definition describes the layout of the object (similar to a "class" definition in C).
ArtiGenericComponentInstance	0..*	The "instance" definition describes a specific instantiated object.

Example 10.8

Examplary Values of the ArtiGeneric Container

```
<AUTOSAR>
  <AR-PACKAGES>
    <AR-PACKAGE>
      <SHORT-NAME>Vendor1</SHORT-NAME>
      <ELEMENTS>
        <ECUC-MODULE-CONFIGURATION-VALUES>
          <SHORT-NAME>Vendor1ArtiGeneric</SHORT-NAME>
          <DEFINITION-REF DEST="ECUC-MODULE-DEF">/AUTOSAR/ArtiDefs/
              ArtiGeneric</DEFINITION-REF>
            <ECUC-CONTAINER-VALUE>
              <SHORT-NAME>ArtiGenericComponentClass_AMODULE</SHORT-NAME
                  >
              <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
                  AUTOSAR/ArtiDefs/ArtiGeneric/
                  ArtiGenericComponentClass</DEFINITION-REF>
              <...>
            </ECUC-CONTAINER-VALUE>
            <ECUC-CONTAINER-VALUE>
              <SHORT-NAME>ArtiGenericComponentClass_RteWiperSwc</SHORT-
                  NAME>
```

```
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
          AUTOSAR/ArtiDefs/ArtiGeneric/
          ArtiGenericComponentClass</DEFINITION-REF>
      <...>
    </ECUC-CONTAINER-VALUE>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>ArtiGenericComponentClass_Vendor1Task</SHORT-
          NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
          AUTOSAR/ArtiDefs/ArtiGeneric/
          ArtiGenericComponentClass</DEFINITION-REF>
      <...>
    </ECUC-CONTAINER-VALUE>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>ArtiGenericComponentInstance_AModule1</SHORT-
          NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
          AUTOSAR/ArtiDefs/ArtiGeneric/
          ArtiGenericComponentInstance</DEFINITION-REF>
      <...>
    </ECUC-CONTAINER-VALUE>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>ArtiGenericComponentInstance_TaskHighPriority
          </SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
          AUTOSAR/ArtiDefs/ArtiGeneric/
          ArtiGenericComponentInstance</DEFINITION-REF>
      <...>
    </ECUC-CONTAINER-VALUE>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>ArtiGenericComponentInstance_Wiper</SHORT-
          NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/
          AUTOSAR/ArtiDefs/ArtiGeneric/
          ArtiGenericComponentInstance</DEFINITION-REF>
      <...>
    </ECUC-CONTAINER-VALUE>
  </CONTAINERS>
</ECUC-MODULE-CONFIGURATION-VALUES>
<...>
```

10.3.1 ArtiGenericComponentClass

SWS Item	**ARTI_ArtiGeneric_xxxx**
Container Name	ArtiGenericComponentClass
Description	The "class" definition describes the layout of the object (similar to a "class" definition in C).

▽

△

Post-Build Variant Multiplicity	False
Configuration Parameters	

Name	ArtiGenericComponentClassDescription
Description	Description of the class.
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiGenericComponentClassName
Description	Name of the class.
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
ArtiGenericComponentClassParameter	0..*	Parameter definition of a class.

SWS Item	ARTI_ArtiGeneric_xxxx
Container Name	ArtiGenericComponentClassParameter
Description	Parameter definition of a class.
Post-Build Variant Multiplicity	False
Configuration Parameters	

Name	ArtiGenericComponentClassParameterDescription
Description	Description of the parameter
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiGenericComponentClassParameterName
Description	Name of the parameter
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiParameterTypeMapRef
Description	Refers to a parameter type to interpret the parameter value.
Type	Reference to ArtiParameterTypeMap
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Example 10.9

Examplary Value of an ArtiGenericComponentClass Container

```
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiGenericComponentClass_AMODULE</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
     ArtiDefs/ArtiGeneric/ArtiGenericComponentClass</DEFINITION-REF>
  <PARAMETER-VALUES>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
         ArtiGeneric/ArtiGenericComponentClass/
         ArtiGenericComponentClassName</DEFINITION-REF>
      <VALUE>AMODULE</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
  </PARAMETER-VALUES>
  <SUB-CONTAINERS>
    <ECUC-CONTAINER-VALUE UUID="">
      <SHORT-NAME>AMODULE_RUNNINGTHING</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
         ArtiDefs/ArtiGeneric/ArtiGenericComponentClass/
         ArtiGenericComponentClassParameter</DEFINITION-REF>
      <PARAMETER-VALUES>
        <ECUC-TEXTUAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
             ArtiDefs/ArtiGeneric/ArtiGenericComponentClass/
             ArtiGenericComponentClassParameter/
             ArtiGenericComponentClassParameterDescription</
             DEFINITION-REF>
          <VALUE>Running Thing</VALUE>
        </ECUC-TEXTUAL-PARAM-VALUE>
        <ECUC-TEXTUAL-PARAM-VALUE>
```

```
            <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
                ArtiDefs/ArtiGeneric/ArtiGenericComponentClass/
                ArtiGenericComponentClassParameter/
                ArtiGenericComponentClassParameterName</DEFINITION-REF>
            <VALUE>RUNNINGTHING</VALUE>
          </ECUC-TEXTUAL-PARAM-VALUE>
        </PARAMETER-VALUES>
        <REFERENCE-VALUES>
          <ECUC-REFERENCE-VALUE>
            <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs
                /ArtiGeneric/ArtiGenericComponentClass/
                ArtiGenericComponentClassParameter/
                ArtiParameterTypeMapRef</DEFINITION-REF>
            <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti
                /ArtiParameterTypeMap_RunningThing</VALUE-REF>
          </ECUC-REFERENCE-VALUE>
        </REFERENCE-VALUES>
      </ECUC-CONTAINER-VALUE>
      <ECUC-CONTAINER-VALUE UUID="">
        <SHORT-NAME>AMOULE_THINGSTART</SHORT-NAME>
        <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
            ArtiDefs/ArtiGeneric/ArtiGenericComponentClass/
            ArtiGenericComponentClassParameter</DEFINITION-REF>
        <PARAMETER-VALUES>
          <ECUC-TEXTUAL-PARAM-VALUE>
            <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
                ArtiDefs/ArtiGeneric/ArtiGenericComponentClass/
                ArtiGenericComponentClassParameter/
                ArtiGenericComponentClassParameterDescription</
                DEFINITION-REF>
            <VALUE>Thing start</VALUE>
          </ECUC-TEXTUAL-PARAM-VALUE>
          <ECUC-TEXTUAL-PARAM-VALUE>
            <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
                ArtiDefs/ArtiGeneric/ArtiGenericComponentClass/
                ArtiGenericComponentClassParameter/
                ArtiGenericComponentClassParameterName</DEFINITION-REF>
            <VALUE>THING_START</VALUE>
          </ECUC-TEXTUAL-PARAM-VALUE>
        </PARAMETER-VALUES>
        <REFERENCE-VALUES>
          <ECUC-REFERENCE-VALUE>
            <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs
                /ArtiGeneric/ArtiGenericComponentClass/
                ArtiGenericComponentClassParameter/
                ArtiParameterTypeMapRef</DEFINITION-REF>
            <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti
                /ArtiParameterTypeMap_ThingStart</VALUE-REF>
          </ECUC-REFERENCE-VALUE>
        </REFERENCE-VALUES>
      </ECUC-CONTAINER-VALUE>
    </SUB-CONTAINERS>
  </ECUC-CONTAINER-VALUE>
```

10.3.2 ArtiGenericComponentInstance

SWS Item	**ARTI_ArtiGeneric_xxxx**
Container Name	ArtiGenericComponentInstance
Description	The "instance" definition describes a specific instantiated object.
Post-Build Variant Multiplicity	False
Configuration Parameters	

Name	ArtiGenericComponentInstanceDescription
Description	Description of the instance.
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiGenericComponentInstanceName
Description	Name of the instance.
Type	EcucStringParamDef
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiGenericComponentClassRef
Description	Refers to a ArtiGenericComponentClass of which this object is instantiated.
Type	Reference to ArtiGenericComponentClass
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
ArtiGenericComponentInstanceParameter	0..*	Parameter definition of an instance.

SWS Item	**ARTI_ArtiGeneric_xxxx**
Container Name	ArtiGenericComponentInstanceParameter
Description	Parameter definition of an instance.
Post-Build Variant Multiplicity	False
Configuration Parameters	

Name	ArtiConstantRef
Description	Refers to an ArtiConstant that represents the value of this parameter.
Type	Reference to ArtiConstant
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile

▽

△

Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiExpressionRef
Description	Refers to an ArtiExpression that evaluates the value of this parameter.
Type	Reference to ArtiExpression
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiGenericComponentClassParameterRef
Description	Refers to an ArtiGenericComponentClassParameter that defines this parameter.
Type	Reference to ArtiGenericComponentClassParameter
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Name	ArtiHookRef
Description	Refers to a hook that records this parameter.
Type	Reference to ArtiHook
Unit	–
Range	–
Post-Build Variant Multiplicity	False
Post-Build Variant Value	False
Multiplicity Configuration Class	Pre-Compile
Value Configuration Class	Pre-Compile
Scope	ECU
Dependency	–

Example 10.10

Examplary Value of an ArtiGenericComponentInstance Container

```
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiGenericComponentInstance_AModule1</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
     ArtiDefs/ArtiGeneric/ArtiGenericComponentInstance</DEFINITION-
     REF>
  <PARAMETER-VALUES>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
         ArtiGeneric/ArtiGenericComponentInstance/
         ArtiGenericComponentInstanceName</DEFINITION-REF>
      <VALUE>AModule1</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
  </PARAMETER-VALUES>
  <REFERENCE-VALUES>
    <ECUC-REFERENCE-VALUE>
      <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs/
         ArtiGeneric/ArtiGenericComponentInstance/
         ArtiGenericComponentClassRef</DEFINITION-REF>
      <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/
         Vendor1ArtiGeneric/ArtiGenericComponentClass_AMODULE</VALUE
         -REF>
    </ECUC-REFERENCE-VALUE>
  </REFERENCE-VALUES>
  <SUB-CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>AModule1_RUNNINGTHING</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
         ArtiDefs/ArtiGeneric/ArtiGenericComponentInstance/
         ArtiGenericComponentInstanceParameter</DEFINITION-REF>
      <REFERENCE-VALUES>
```

```
            <ECUC-REFERENCE-VALUE>
              <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs
                  /ArtiGeneric/ArtiGenericComponentInstance/
                  ArtiGenericComponentInstanceParameter/ArtiExpressionRef
                  </DEFINITION-REF>
              <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti
                  /ArtiExpression_ArtiGeneric_AModule1_RunningThing</
                  VALUE-REF>
            </ECUC-REFERENCE-VALUE>
            <ECUC-REFERENCE-VALUE>
              <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs
                  /ArtiGeneric/ArtiGenericComponentInstance/
                  ArtiGenericComponentInstanceParameter/
                  ArtiGenericComponentClassParameterRef</DEFINITION-REF>
              <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/
                  Vendor1ArtiGeneric/ArtiGenericComponentClass_AMODULE/
                  AMODULE_RUNNINGTHING</VALUE-REF>
            </ECUC-REFERENCE-VALUE>
          </REFERENCE-VALUES>
        </ECUC-CONTAINER-VALUE>
      </SUB-CONTAINERS>
    </ECUC-CONTAINER-VALUE>
```

10.4 Published Information

For details refer to the chapter 10.3 “Published Information” in SWS_BSWGeneral.

A Not applicable requirements

[SWS_XYZ_00999] ⌈ These requirements are not applicable to this specification. ⌋ *(Req_Id_1, Req_Id_2, Req_Id_3)*

B Example

The example provided in this chapter demonstrates how to apply ARTI to an operating system and also how to use ARTI from within the application to trace some user-defined data. It also shows how the generic `ARTI_TRACE` macro can be mapped to different tracing implementations. In the example, these first tracing implementations is provided by `VENDOR_A` the second by `VENDOR_B`.

The C code of the example compiles but is not functional. The operating system is boiled down to three functions: `SuspendAllInterrupts`, `ResumeAllInterrupts` and `StartOS`. The application code defined the `main` function and two tasks: `Task_Cylinder0` and `Task_Cylinder1`.

Section B.1 holds all the C code demonstrating the ARTI instrumentation and section B.1.3 contains the corresponding ARXML code.

B.1 ARTI Instrumentation

B.1.1 ARTI Tool Binding (ARTI.h)

Listing B.1: Example for ARTI.h

```
#ifndef _TOOL_VENDOR_BINDING_H_
#define _TOOL_VENDOR_BINDING_H_

#include <stdint.h>

#if defined VENDOR_A
/* ARTI Trace Macro */
#   define ARTI_TRACE(_contextName, _className, _instanceName, instanceParameter, _eventName, event_value) \
        (void)TraceImpl ## _ ## _className ## _ ## _eventName ## _ ## _instanceName ## _ ## _contextName( (instanceParameter), (event_value) )

/* Prototypes for AR_CP_OS_TASKSCHEDULER */
void TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Start_OS_SHORT_NAME_SPRVSR(uint32_t instanceParameter, uint32_t event_value);
void TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Stop_OS_SHORT_NAME_SPRVSR(uint32_t instanceParameter, uint32_t event_value);

void TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Start_OS_SHORT_NAME_USER(uint32_t instanceParameter, uint32_t event_value);
void TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Stop_OS_SHORT_NAME_USER(uint32_t instanceParameter, uint32_t event_value);

void TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Start_OS_SHORT_NAME_NOSUSP(uint32_t instanceParameter, uint32_t event_value);
void TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Stop_OS_SHORT_NAME_NOSUSP(uint32_t instanceParameter, uint32_t event_value);

/* Prototypes for Ingnition_Control */
void TraceImpl_Ingnition_Control_IgnitionStart_Cylinder0_USER(uint32_t instanceParameter, uint32_t event_value);
void TraceImpl_Ingnition_Control_IgnitionStop_Cylinder0_USER(uint32_t instanceParameter, uint32_t event_value);

void TraceImpl_Ingnition_Control_IgnitionStart_Cylinder1_USER(uint32_t instanceParameter, uint32_t event_value);
void TraceImpl_Ingnition_Control_IgnitionStop_Cylinder1_USER(uint32_t instanceParameter, uint32_t event_value);

#elif defined VENDOR_B
/* ARTI Trace Macro */
#   define ARTI_TRACE(_contextName, _className, _instanceName, instanceParameter, _eventName, event_value) \
```

```
31          (void)TraceImpl ## _ ## _className ## _ ## _contextName( (
               _instanceName), (instanceParameter), (_eventName), (
               event_value) )
32
33  /* Defines for AR_CP_OS_TASKSCHEDULER */
34  /* Instance Names */
35  #define OS_SHORT_NAME (0)
36  /* Event Names */
37  #define OsTask_Start  (0)
38  #define OsTask_Stop   (1)
39
40  /* Defines for Ingnition_Control */
41  /* Instance Names */
42  #define Cylinder0     (0)
43  #define Cylinder1     (1)
44  /* Event Names */
45  #define IgnitionStart (0)
46  #define IgnitionStop  (1)
47
48  /* Prototypes for AR_CP_OS_TASKSCHEDULER */
49  void TraceImpl_AR_CP_OS_TASKSCHEDULER_SPRVSR(uint32_t instanceName,
       uint32_t instanceParameter, uint32_t eventName, uint32_t
       event_value);
50  void TraceImpl_AR_CP_OS_TASKSCHEDULER_USER(uint32_t instanceName,
       uint32_t instanceParameter, uint32_t eventName, uint32_t
       event_value);
51  void TraceImpl_AR_CP_OS_TASKSCHEDULER_NOSUSP(uint32_t instanceName,
       uint32_t instanceParameter, uint32_t eventName, uint32_t
       event_value);
52
53  /* Prototypes for Ingnition_Control */
54  void TraceImpl_Ingnition_Control_SPRVSR(uint32_t instanceName, uint32_t
        instanceParameter, uint32_t eventName, uint32_t event_value);
55  void TraceImpl_Ingnition_Control_USER(uint32_t instanceName, uint32_t
       instanceParameter, uint32_t eventName, uint32_t event_value);
56  void TraceImpl_Ingnition_Control_NOSUSP(uint32_t instanceName, uint32_t
        instanceParameter, uint32_t eventName, uint32_t event_value);
57
58  #else
59  #   define ARTI_TRACE(_contextName, _className, _instanceName,
       instanceParameter, _eventName, event_value) (void)0
60  #endif
61
62
63  #endif
```

Listing B.2: Example for ARTI.c

```
1  #include <stdint.h>
2
3  #include "os.h"
4  #include "tool-vendor_binding.h"
5
6  /* Stubs for intrinsics */
7  #define __disable() ((void)(0))
8  #define __enable() ((void)(0))
```

```
9
10 #if defined VENDOR_A
11
12 void TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Start_OS_SHORT_NAME_SPRVSR
       (uint32_t instanceParameter, uint32_t event_value)
13 {
14     __disable();
15     TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Start_OS_SHORT_NAME_NOSUSP(
           instanceParameter, event_value);
16     __enable();
17 }
18
19 void TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Stop_OS_SHORT_NAME_SPRVSR(
       uint32_t instanceParameter, uint32_t event_value)
20 {
21     __disable();
22     TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Stop_OS_SHORT_NAME_NOSUSP(
           instanceParameter, event_value);
23     __enable();
24 }
25
26 void TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Start_OS_SHORT_NAME_USER(
       uint32_t instanceParameter, uint32_t event_value)
27 {
28     SuspendAllInterrupts();
29     TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Start_OS_SHORT_NAME_NOSUSP(
           instanceParameter, event_value);
30     ResumeAllInterrupts();
31 }
32
33 void TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Stop_OS_SHORT_NAME_USER(
       uint32_t instanceParameter, uint32_t event_value)
34 {
35     SuspendAllInterrupts();
36     TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Stop_OS_SHORT_NAME_NOSUSP(
           instanceParameter, event_value);
37     ResumeAllInterrupts();
38 }
39
40 void TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Start_OS_SHORT_NAME_NOSUSP
       (uint32_t instanceParameter, uint32_t event_value)
41 {
42     (void)instanceParameter; // avoid warning "unused parameter"
43     (void)event_value; // avoid warning "unused parameter"
44
45     // actual tracing code goes here
46 }
47
48 void TraceImpl_AR_CP_OS_TASKSCHEDULER_OsTask_Stop_OS_SHORT_NAME_NOSUSP(
       uint32_t instanceParameter, uint32_t event_value)
49 {
50     (void)instanceParameter; // avoid warning "unused parameter"
51     (void)event_value; // avoid warning "unused parameter"
52
53     // actual tracing code goes here
54 }
```

```
55
56 void TraceImpl_Ingnition_Control_IgnitionStart_Cylinder0_USER(uint32_t
       instanceParameter, uint32_t event_value)
57 {
58     (void)instanceParameter; // avoid warning "unused parameter"
59     (void)event_value; // avoid warning "unused parameter"
60     SuspendAllInterrupts();
61     // actual tracing code goes here
62     ResumeAllInterrupts();
63 }
64
65 void TraceImpl_Ingnition_Control_IgnitionStop_Cylinder0_USER(uint32_t
       instanceParameter, uint32_t event_value)
66 {
67     (void)instanceParameter; // avoid warning "unused parameter"
68     (void)event_value; // avoid warning "unused parameter"
69     SuspendAllInterrupts();
70     // actual tracing code goes here
71     ResumeAllInterrupts();
72 }
73
74 void TraceImpl_Ingnition_Control_IgnitionStart_Cylinder1_USER(uint32_t
       instanceParameter, uint32_t event_value)
75 {
76     (void)instanceParameter; // avoid warning "unused parameter"
77     (void)event_value; // avoid warning "unused parameter"
78     SuspendAllInterrupts();
79     // actual tracing code goes here
80     ResumeAllInterrupts();
81 }
82
83 void TraceImpl_Ingnition_Control_IgnitionStop_Cylinder1_USER(uint32_t
       instanceParameter, uint32_t event_value)
84 {
85     (void)instanceParameter; // avoid warning "unused parameter"
86     (void)event_value; // avoid warning "unused parameter"
87     SuspendAllInterrupts();
88     // actual tracing code goes here
89     ResumeAllInterrupts();
90 }
91
92 #elif defined VENDOR_B
93
94 void TraceImpl_AR_CP_OS_TASKSCHEDULER_SPRVSR(uint32_t instanceName,
       uint32_t instanceParameter, uint32_t eventName, uint32_t
       event_value)
95 {
96     __disable();
97     TraceImpl_AR_CP_OS_TASKSCHEDULER_NOSUSP(instanceName,
           instanceParameter, eventName, event_value);
98     __enable();
99 }
100
101 void TraceImpl_AR_CP_OS_TASKSCHEDULER_USER(uint32_t instanceName,
       uint32_t instanceParameter, uint32_t eventName, uint32_t
       event_value)
```

```
{
    SuspendAllInterrupts();
    TraceImpl_AR_CP_OS_TASKSCHEDULER_NOSUSP(instanceName,
       instanceParameter, eventName, event_value);
    ResumeAllInterrupts();
}

void TraceImpl_AR_CP_OS_TASKSCHEDULER_NOSUSP(uint32_t instanceName,
   uint32_t instanceParameter, uint32_t eventName, uint32_t
   event_value)
{
    (void)instanceName; // avoid warning "unused parameter"
    (void)instanceParameter; // avoid warning "unused parameter"
    (void)eventName; // avoid warning "unused parameter"
    (void)event_value; // avoid warning "unused parameter"

    // actual tracing code goes here
}

void TraceImpl_Ingnition_Control_SPRVSR(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value)
{
    __disable();
    TraceImpl_Ingnition_Control_NOSUSP(instanceName, instanceParameter,
        eventName, event_value);
    __enable();
}

void TraceImpl_Ingnition_Control_USER(uint32_t instanceName, uint32_t
   instanceParameter, uint32_t eventName, uint32_t event_value)
{
    SuspendAllInterrupts();
    TraceImpl_Ingnition_Control_NOSUSP(instanceName, instanceParameter,
        eventName, event_value);
    ResumeAllInterrupts();
}

void TraceImpl_Ingnition_Control_NOSUSP(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value)
{
    (void)instanceName; // avoid warning "unused parameter"
    (void)instanceParameter; // avoid warning "unused parameter"
    (void)eventName; // avoid warning "unused parameter"
    (void)event_value; // avoid warning "unused parameter"

    // actual tracing code goes here
}

#else

#endif
```

B.1.2 ARTI OS Instrumentation

Listing B.3: Example for OS instrumentation header

```
#ifndef _OS_H_
#define _OS_H_

#define TASK(_taskname)     void OS_TASK ## _ ## _taskname(void)

void SuspendAllInterrupts(void);
void ResumeAllInterrupts(void);

void StartOS(void);

#endif
```

Listing B.4: Example for OS instrumentation source

```
#include "user_main.h"
#include "tool-vendor_binding.h"

void SuspendAllInterrupts(void)
{
    // ...
}

void ResumeAllInterrupts(void)
{
    // ...
}

void StartOS(void)
{
    const int myCoreId = 0;
    const int OS_TASK_Task_Cylinder0_ID = 2;

    // for testing the ARTI interface, we call the task UserTask1 here
       directly (rather than implementing an OS)
    ARTI_TRACE(NOSUSP, AR_CP_OS_TASKSCHEDULER, OS_SHORT_NAME, myCoreId,
        OsTask_Start, OS_TASK_Task_Cylinder0_ID);
    OS_TASK_Task_Cylinder0();
    ARTI_TRACE(NOSUSP, AR_CP_OS_TASKSCHEDULER, OS_SHORT_NAME, myCoreId,
        OsTask_Stop,  OS_TASK_Task_Cylinder0_ID);
}
```

B.1.3 ARTI Arbitrary Instrumentation

Listing B.5: Example for arbitrary (user code) instrumentation header

```
#ifndef _USER_MAIN_H_
#define _USER_MAIN_H_

#include "os.h"
extern TASK(Task_Cylinder0);
extern TASK(Task_Cylinder1);

#endif
```

Listing B.6: Example for arbitrary (user code) instrumentation source

```
#include <stdlib.h>
```

```
2
3  #include "os.h"
4  #include "tool-vendor_binding.h"
5
6  TASK(Task_Cylinder0)
7  {
8      ARTI_TRACE(USER, Ingnition_Control, Cylinder0, 0, IgnitionStart,
           53);
9      // inject
10     ARTI_TRACE(USER, Ingnition_Control, Cylinder0, 0, IgnitionStop, 53)
           ;
11 }
12
13 TASK(Task_Cylinder1)
14 {
15     ARTI_TRACE(USER, Ingnition_Control, Cylinder1, 0, IgnitionStart,
           77);
16     // inject
17     ARTI_TRACE(USER, Ingnition_Control, Cylinder1, 0, IgnitionStop, 77)
           ;
18 }
19
20 int main(void)
21 {
22     StartOS();
23
24     exit(EXIT_SUCCESS);
25
26     return -1;
27 }
```

B.2 ARXML Representation of Instrumentation

Example B.1

Examplary value of the ArtiHook container for OsTask_Start

```
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiHook_ArtiOs_TaskStart</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
      ArtiDefs/Arti/ArtiHook</DEFINITION-REF>
  <PARAMETER-VALUES>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
          Arti/ArtiHook/ArtiHookClass</DEFINITION-REF>
      <VALUE>AR_CP_OS_TASKSCHEDULER</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
          Arti/ArtiHook/ArtiHookEventName</DEFINITION-REF>
      <VALUE>OsTask_Start</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
    <ECUC-TEXTUAL-PARAM-VALUE>
```

```
        <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
            Arti/ArtiHook/ArtiHookInstance</DEFINITION-REF>
        <VALUE>OS_SHORT_NAME</VALUE>
      </ECUC-TEXTUAL-PARAM-VALUE>
    </PARAMETER-VALUES>
    <REFERENCE-VALUES>
      <ECUC-REFERENCE-VALUE>
        <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs/
            Arti/ArtiHook/ArtiHookEventParameterTypeRef</DEFINITION-REF
            >
        <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti/
            ArtiParameterTypeMap_TaskCylinderId</VALUE-REF>
      </ECUC-REFERENCE-VALUE>
      <ECUC-REFERENCE-VALUE>
        <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs/
            Arti/ArtiHook/ArtiHookInstanceParameterTypeRef</DEFINITION-
            REF>
        <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti/
            ArtiParameterTypeMap_Core</VALUE-REF>
      </ECUC-REFERENCE-VALUE>
    </REFERENCE-VALUES>
  </ECUC-CONTAINER-VALUE>
```

Example B.2

Examplary value of the ArtiOsInstance container using the hooks

```
  <ECUC-CONTAINER-VALUE>
    <SHORT-NAME>ArtiOsInstance_Conf</SHORT-NAME>
    <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
        ArtiDefs/ArtiOs/ArtiOsInstance</DEFINITION-REF>
    <REFERENCE-VALUES>
      <ECUC-REFERENCE-VALUE>
        <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs/
            ArtiOs/ArtiOsInstance/ArtiOsEcucRef</DEFINITION-REF>
        <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1EcucOs/
            Vendor1Os</VALUE-REF>
      </ECUC-REFERENCE-VALUE>
      <ECUC-REFERENCE-VALUE>
        <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs/
            ArtiOs/ArtiOsInstance/ArtiOsTaskHookRef</DEFINITION-REF>
        <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti/
            ArtiHook_ArtiOs_TaskStart</VALUE-REF>
      </ECUC-REFERENCE-VALUE>
      <ECUC-REFERENCE-VALUE>
        <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs/
            ArtiOs/ArtiOsInstance/ArtiOsTaskHookRef</DEFINITION-REF>
        <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti/
            ArtiHook_ArtiOs_TaskStop</VALUE-REF>
      </ECUC-REFERENCE-VALUE>
    </REFERENCE-VALUES>
  </ECUC-CONTAINER-VALUE>
```

Example B.3

Examplary value of the ArtiHook container for arbitrary use

```
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiHook_IgnitionControl_Cyl0_IgnitionStart</SHORT-NAME
     >
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
     ArtiDefs/Arti/ArtiHook</DEFINITION-REF>
  <PARAMETER-VALUES>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
         Arti/ArtiHook/ArtiHookClass</DEFINITION-REF>
      <VALUE>Ignition_Control</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
         Arti/ArtiHook/ArtiHookEventName</DEFINITION-REF>
      <VALUE>IgnitionStart</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
         Arti/ArtiHook/ArtiHookInstance</DEFINITION-REF>
      <VALUE>Cylinder0</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
  </PARAMETER-VALUES>
</ECUC-CONTAINER-VALUE>
```

Example B.4

Examplary value of an ArtiGenericComponentClass container with parameters holding hooks

```
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiGenericComponentClass_IgnitionControl</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
     ArtiDefs/ArtiGeneric/ArtiGenericComponentClass</DEFINITION-REF>
  <PARAMETER-VALUES>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
         ArtiGeneric/ArtiGenericComponentClass/
         ArtiGenericComponentClassName</DEFINITION-REF>
      <VALUE>ADIFFERENT</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
  </PARAMETER-VALUES>
  <SUB-CONTAINERS>
    <ECUC-CONTAINER-VALUE UUID="">
      <SHORT-NAME>IgnitionStart</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
         ArtiDefs/ArtiGeneric/ArtiGenericComponentClass/
         ArtiGenericComponentClassParameter</DEFINITION-REF>
      <PARAMETER-VALUES>
        <ECUC-TEXTUAL-PARAM-VALUE>
```

```
            <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
               ArtiDefs/ArtiGeneric/ArtiGenericComponentClass/
               ArtiGenericComponentClassParameter/
               ArtiGenericComponentClassParameterDescription</
               DEFINITION-REF>
            <VALUE>Ignition Start</VALUE>
          </ECUC-TEXTUAL-PARAM-VALUE>
          <ECUC-TEXTUAL-PARAM-VALUE>
            <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
               ArtiDefs/ArtiGeneric/ArtiGenericComponentClass/
               ArtiGenericComponentClassParameter/
               ArtiGenericComponentClassParameterName</DEFINITION-REF>
            <VALUE>IGNITION_START</VALUE>
          </ECUC-TEXTUAL-PARAM-VALUE>
        </PARAMETER-VALUES>
      </ECUC-CONTAINER-VALUE>
      <ECUC-CONTAINER-VALUE UUID="">
        <SHORT-NAME>IgnitionStop</SHORT-NAME>
        <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
           ArtiDefs/ArtiGeneric/ArtiGenericComponentClass/
           ArtiGenericComponentClassParameter</DEFINITION-REF>
        <PARAMETER-VALUES>
          <ECUC-TEXTUAL-PARAM-VALUE>
            <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
               ArtiDefs/ArtiGeneric/ArtiGenericComponentClass/
               ArtiGenericComponentClassParameter/
               ArtiGenericComponentClassParameterDescription</
               DEFINITION-REF>
            <VALUE>Ignition Stop</VALUE>
          </ECUC-TEXTUAL-PARAM-VALUE>
          <ECUC-TEXTUAL-PARAM-VALUE>
            <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/
               ArtiDefs/ArtiGeneric/ArtiGenericComponentClass/
               ArtiGenericComponentClassParameter/
               ArtiGenericComponentClassParameterName</DEFINITION-REF>
            <VALUE>IGNITION_STOP</VALUE>
          </ECUC-TEXTUAL-PARAM-VALUE>
        </PARAMETER-VALUES>
      </ECUC-CONTAINER-VALUE>
    </SUB-CONTAINERS>
  </ECUC-CONTAINER-VALUE>
```

Example B.5

Examplary value of an ArtiGenericComponentInstance container using the hooks

```
  <ECUC-CONTAINER-VALUE>
    <SHORT-NAME>ArtiGenericComponentInstance_IgnitionCyl0</SHORT-NAME>
    <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
       ArtiDefs/ArtiGeneric/ArtiGenericComponentInstance</DEFINITION-
       REF>
    <PARAMETER-VALUES>
      <ECUC-TEXTUAL-PARAM-VALUE>
```

```
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF">/AUTOSAR/ArtiDefs/
          ArtiGeneric/ArtiGenericComponentInstance/
          ArtiGenericComponentInstanceName</DEFINITION-REF>
      <VALUE>Ignition Cylinder 0</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
  </PARAMETER-VALUES>
  <REFERENCE-VALUES>
    <ECUC-REFERENCE-VALUE>
      <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs/
          ArtiGeneric/ArtiGenericComponentInstance/
          ArtiGenericComponentClassRef</DEFINITION-REF>
      <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/
          Vendor1ArtiGeneric/
          ArtiGenericComponentClass_IgnitionControl</VALUE-REF>
    </ECUC-REFERENCE-VALUE>
  </REFERENCE-VALUES>
  <SUB-CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>IgnitionCyl0Start</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
          ArtiDefs/ArtiGeneric/ArtiGenericComponentInstance/
          ArtiGenericComponentInstanceParameter</DEFINITION-REF>
      <REFERENCE-VALUES>
        <ECUC-REFERENCE-VALUE>
          <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs
              /ArtiGeneric/ArtiGenericComponentInstance/
              ArtiGenericComponentInstanceParameter/
              ArtiGenericComponentClassParameterRef</DEFINITION-REF>
          <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/
              Vendor1ArtiGeneric/
              ArtiGenericComponentClass_IgnitionControl/IgnitionStart
              </VALUE-REF>
        </ECUC-REFERENCE-VALUE>
        <ECUC-REFERENCE-VALUE>
          <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs
              /ArtiGeneric/ArtiGenericComponentInstance/
              ArtiGenericComponentInstanceParameter/ArtiHookRef</
              DEFINITION-REF>
          <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti
              /ArtiHook_IgnitionControl_Cyl0_IgnitionStart</VALUE-REF
              >
        </ECUC-REFERENCE-VALUE>
      </REFERENCE-VALUES>
    </ECUC-CONTAINER-VALUE>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>IgnitionCyl0Stop</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF">/AUTOSAR/
          ArtiDefs/ArtiGeneric/ArtiGenericComponentInstance/
          ArtiGenericComponentInstanceParameter</DEFINITION-REF>
      <REFERENCE-VALUES>
        <ECUC-REFERENCE-VALUE>
          <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs
              /ArtiGeneric/ArtiGenericComponentInstance/
              ArtiGenericComponentInstanceParameter/
              ArtiGenericComponentClassParameterRef</DEFINITION-REF>
```

```
            <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/
                Vendor1ArtiGeneric/
                ArtiGenericComponentClass_IgnitionControl/IgnitionStop<
                /VALUE-REF>
          </ECUC-REFERENCE-VALUE>
          <ECUC-REFERENCE-VALUE>
            <DEFINITION-REF DEST="ECUC-REFERENCE-DEF">/AUTOSAR/ArtiDefs
                /ArtiGeneric/ArtiGenericComponentInstance/
                ArtiGenericComponentInstanceParameter/ArtiHookRef</
                DEFINITION-REF>
            <VALUE-REF DEST="ECUC-CONTAINER-VALUE">/Vendor1/Vendor1Arti
                /ArtiHook_IgnitionControl_Cyl0_IgnitionStop</VALUE-REF>
          </ECUC-REFERENCE-VALUE>
        </REFERENCE-VALUES>
      </ECUC-CONTAINER-VALUE>
    </SUB-CONTAINERS>
  </ECUC-CONTAINER-VALUE>
```