

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

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R19-11

Document Change History			
Date	Release	Changed by	Description
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> • Added expression syntax • Corrected trace macros and ARTI class names • Added and extended several configuration parameters • Corrected SWS item references • Changed Document Status from Final to published
2018-10-31	R4.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"> • Initial release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1	Introduction and functional overview	5
2	Acronyms and Abbreviations	7
3	Related documentation	8
3.1	Input documents & related standards and norms	8
3.2	Related specification	8
4	Constraints and assumptions	8
4.1	Limitations	9
4.2	Applicability to car domains	9
5	Dependencies to other modules	9
6	Requirements Tracing	9
7	Functional specification	10
7.1	ARTI Module Description	10
7.2	ARTI Hook Implementation	11
8	API specification	12
8.1	Imported types	12
8.2	Type definitions	12
8.3	Function definitions	12
8.4	Callback notifications	12
8.5	Scheduled functions	12
8.6	Expected interfaces	12
8.6.1	Mandatory interfaces	13
8.6.1.1	ARTI Tracing Macro	13
8.6.2	Optional interfaces	15
8.6.2.1	ARTI Generic Stopwatch	15
8.6.2.2	ARTI Generic Dataflow Stopwatch	16
8.6.2.3	ARTI Generic Datapoint	17
8.6.3	Configurable interfaces	18
8.7	Service Interfaces	18
9	Sequence diagrams	19
10	Configuration specification	19
10.1	How to read this chapter	19
10.2	ARTI Parameters	19
10.2.1	ArtiConstant	21
10.2.2	ArtiExpression	23
10.2.3	ArtiHook	24
10.2.4	ArtiObjectClassParameter	28

10.2.5	ArtiObjectInstanceParameter	29
10.2.6	ArtiParameterTypeMap	31
10.2.7	ArtiStates	36
10.3	ARTI Generic Container	38
10.3.1	ArtiGenericComponentClass	40
10.3.2	ArtiGenericComponentInstance	43
10.4	Published Information	48
A	Not applicable requirements	48
B	Example	48
B.1	ARTI Instrumentation	48
B.1.1	ARTI Tool Binding (ARTI.h)	48
B.1.2	ARTI OS Instrumentation	53
B.1.3	ARTI Arbitrary Instrumentation	54
B.2	ARXML Representation of Instrumentation	55
C	Expression Syntax	60

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 distributed 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.

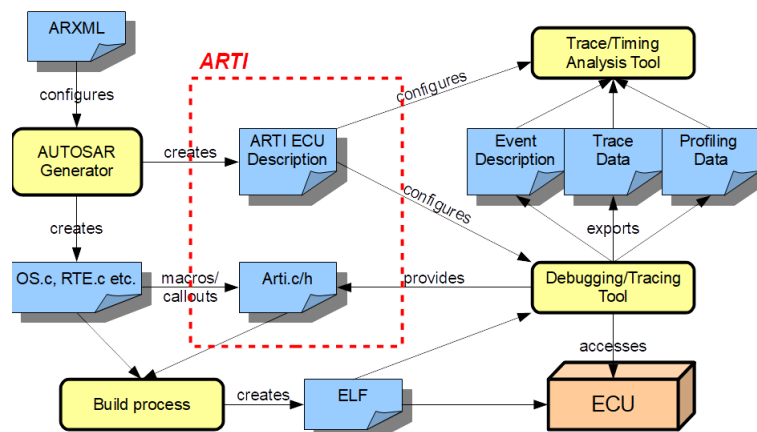


Figure 1.1: ARTI data flow

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.

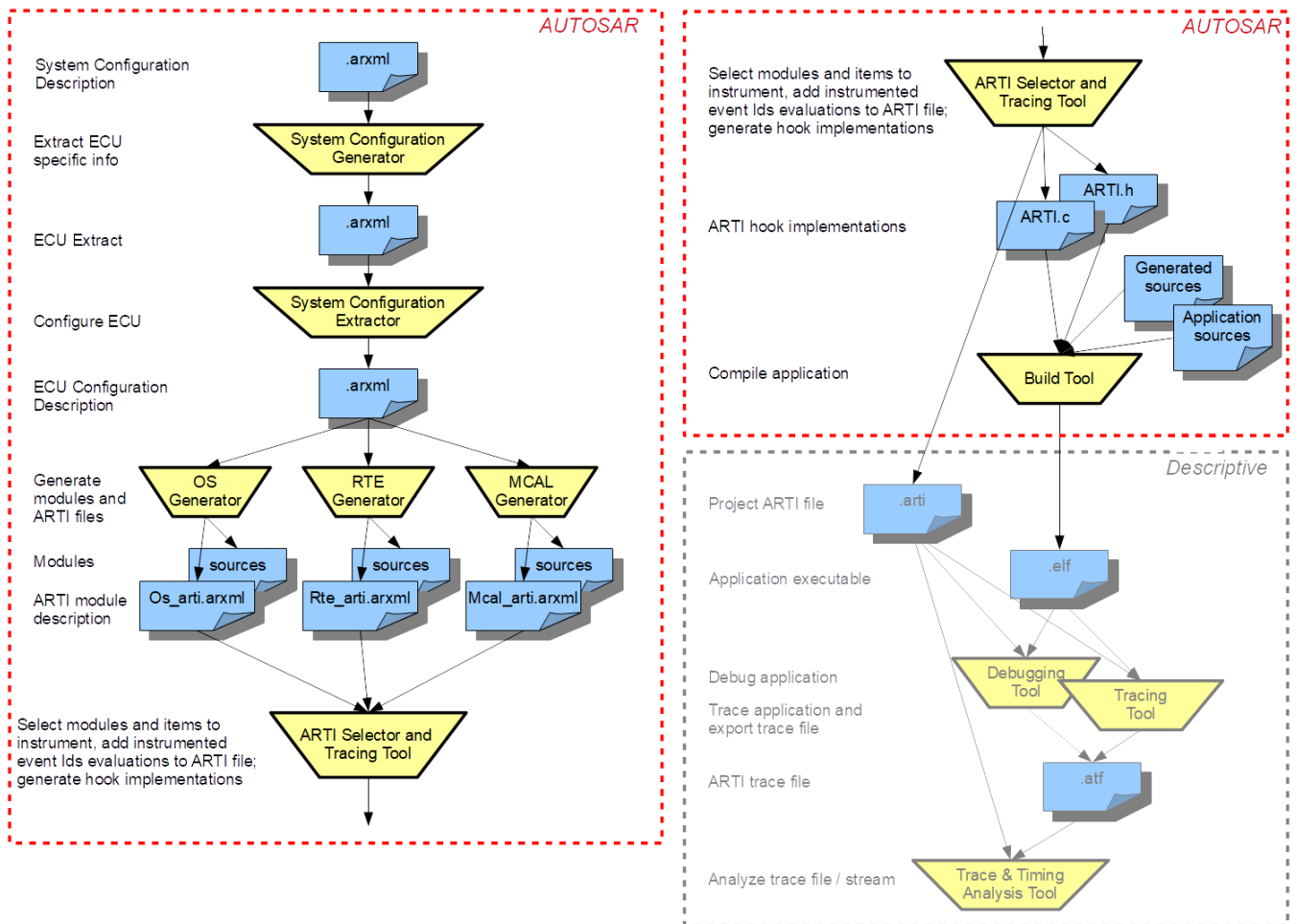


Figure 1.2: ARTI work flow

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 <ul style="list-style-type: none"> 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	<p>"Tracing" refers to collecting run-time information over a certain period of time</p> <ul style="list-style-type: none"> • 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	<p>"Timing Measurement" refers to capturing of timing information</p> <ul style="list-style-type: none"> • 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	<p>"Profiling" refers to the process of gaining timing parameters/timing statistics</p> <ul style="list-style-type: none"> • 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 the current state of the application. Expressions are like C expressions but limited so that they can be evaluated statically. Hence only accesses to global variables are allowed, and only unary, binary and trinary operators are allowed. Especially accesses to local variables and calls to functions are not allowed. See Appendix C for a full syntax specification of Expressions. Expressions are referred to by an object information (see "Object Information" in chapter 7.1) and are used to define the evaluation of parameter 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, _instanceName, \
2                   instanceParameter, _eventName, eventParameter) \
3   ARTI_TRACE ## _ ## _className ## _ ## _instanceName \
4   ## _ ## _eventName ## _ ## _contextName \
5   ( (instanceParameter), (eventParameter) )
```

Such an implementation will generate one hook for all the possible combinations of `_contextName`, `_className`, `_instanceName` and `_eventName` and pass parameters `instanceParameter` and `eventParameter` 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_CP_OS_TASK`) or user defined. The predefined classes are specified in the SWS of the according BSW module (e.g. `SWS_OS`).

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

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

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

8.6.2 Optional interfaces

This section defines all interfaces, which are required to fulfill an optional functionality of the module.

8.6.2.1 ARTI Generic Stopwatch

A stopwatch can be used to time between two user defined points in an application. The user can put the corresponding `ARTI_TRACE` macro calls of the class `USER_STOPWATCH` anywhere in the code. An arbitrary number of stopwatches are supported by using different instance names (`_instanceName`). Please note that the trace tool provider might put limits on the number of active stopwatches.

The trace tool shall at least consider the time between the first `START` event and the first `STOP` event in a given sequence and doesn't need to consider nested `START` and `STOP` events. E.g.

1. **START**
2. `START` (ignored, already started)
3. **STOP**
4. `STOP` (ignored, no matching `START`)
5. **START**
6. **STOP**

Only events in **bold** are considered, time is calculated between *1 and 3* and *5 and 6*.

ARTI_TRACE Parameters

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

Parameter	Type	Description
_contextName	Token, literal text	see "8.6.1.1 ARTI Tracing Macro"
_className	Token, literal text	USER_STOPWATCH
_instanceName	Token, literal text	value that identifies the instance of the stopwatch
instanceParameter	uint32	Not used, should be set to 0
_eventName	Token, literal text	value that identifies the event of the timer, one of START or STOP
eventParameter	uint32	Not used, should be set to 0

Example 8.1

```
1 ARTI_TRACE(USER, USER_STOPWATCH, myStopwatch, 0, START, 0);
2 ARTI_TRACE(USER, USER_STOPWATCH, myStopwatch, 0, STOP, 0);
```

8.6.2.2 ARTI Generic Dataflow Stopwatch

A dataflow stopwatch can be used to time between *write* and *read* accesses to a given variable. The user can put the corresponding `ARTI_TRACE` macro calls of the class `USER_DATAFLOW_STOPWATCH` anywhere in the code. An arbitrary number of dataflow stopwatches are supported by using different instance names (`_instanceName`). Please note that the trace tool provider might put limits on the number of active dataflow stopwatches.

The trace tool shall at least consider the time between the last `WRITE` event, the first `READ` and the last `READ` event in a given sequence and doesn't need to consider nested `WRITE` and `READ` events. E.g.

1. `WRITE` (ignored as it gets overwritten in 2)
2. `WRITE`
3. `READ`
4. `WRITE`
5. `READ` (min)
6. `READ` (ignored, if only consider min and max)
7. `READ` (max)

Only events in **bold** are considered, time is calculated between *2 and 3* and *4 and 5/7*. The time between 4 and the 5 yields the **min** data age time, likewise the time between 4 and 7 yields the **max** data age time for the second sequence.

ARTI_TRACE Parameters

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

Parameter	Type	Description
_contextName	Token, literal text	see "8.6.1.1 ARTI Tracing Macro"
_className	Token, literal text	USER_DATAFLOW_STOPWATCH
_instanceName	Token, literal text	value that identifies the instance of the dataflow stopwatch
instanceParameter	uint32	Not used, should be set to 0
_eventName	Token, literal text	value that identifies the event of the timer, one of WRITE or READ
eventParameter	uint32	Not used, should be set to 0

Example 8.2

```
1 ARTI_TRACE(USER, USER_DATAFLOW_STOPWATCH, myDataflowStopwatch, 0, WRITE, 0);
2 myVariable = 1;
3 ...
4 uint32 temp = myVariable;
5 ARTI_TRACE(USER, USER_DATAFLOW_STOPWATCH, myDataFlowStopwatch, 0, READ, 0);
```

8.6.2.3 ARTI Generic Datapoint

A *datapoint* provides the possibility to record different values at user defined locations in the code. The user can put the corresponding ARTI_TRACE macro calls of the class USER_DATAPOINT anywhere in the code. An arbitrary number of data points are supported by using different instance names (_instanceName). Please note that the trace tool provider might put limits on the number of active data points. There are predefined event names (_eventName) for different data types as defined by AUTOSAR (see AUTOSAR_SWS_PlatformTypes, e.g. UINT32) this information might be used by the trace tool for optimized storage and visualization.

ARTI_TRACE Parameters

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

Parameter	Type	Description
<code>_contextName</code>	Token, literal text	see "8.6.1.1 ARTI Tracing Macro"
<code>_className</code>	Token, literal text	USER_DATAPOINT
<code>_instanceName</code>	Token, literal text	value that identifies the instance of the data point
<code>instanceParameter</code>	uint32	Not used, should be set to 0
<code>_eventName</code>	Token, literal text	value that identifies the event of the data point, shall be one of the following: <ul style="list-style-type: none"> • BOOLEAN • UINT8 • UINT16 • UINT32 • SINT8 • SINT16 • SINT32 • FLOAT32
<code>eventParameter</code>	uint32	Value that shall be recorded by the event (up to 32-bits)

Example 8.3

```

1 ARTI_TRACE(USER, USER_DATAPOINT, myDatapoint0, 0, UINT32, 2ul);
2 ARTI_TRACE(USER, USER_DATAPOINT, myDatapoint1, 0, SINT8, s8_Data);

```

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	ECUC_Arti_00001	
Module Name	Arti	
Module Description	The Arti Module contains all parameters for ARTI that are filled by the generators (OS, RTE, ...).	
Post-Build Variant Support	true	
Supported Config Variants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE	
Included Containers		
Container Name	Multiplicity	Scope / Dependency

ArtiConstant	0..*	This container holds a constant value. Tags: atp.Status=draft
ArtiExpression	0..*	This container holds a C like expression that a debugger can evaluate. This is similar to what is already done in ORTI. Tags: atp.Status=draft
ArtiHook	0..*	This container represents an ARTI hook that is present in the module. Tags: atp.Status=draft
ArtiObjectClassParameter	0..*	This container represents a parameter of an Arti object class definition. Tags: atp.Status=draft
ArtiObjectInstance Parameter	0..*	This container represents a parameter of an Arti object instance. Tags: atp.Status=draft
ArtiParameterTypeMap	0..*	A map of key/value pairs to map a parameter value to a display string and/or an Arti or EcuC object. Tags: atp.Status=draft
ArtiStates	0..*	This container contains all states of tasks, isrs... that the EcuC uses. Tags: atp.Status=draft

Example 10.1

Exemplary 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>ArtiParamTypeMap_Core</SHORT-NAME>
    <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF"/>
      AUTOSAR/ArtiDefs/Arti/ArtiParamTypeMap</
      DEFINITION-REF>
    <...>
  </ECUC-CONTAINER-VALUE>
</CONTAINERS>
</ECUC-MODULE-CONFIGURATION-VALUES>
<...>

```

10.2.1 ArtiConstant

SWS Item	[ECUC_Arti_00002]
Container Name	ArtiConstant
Parent Container	Arti

Description	This container holds a constant value. Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	ArtiConstantString [ECUC_Arti_00008]		
Parent Container	ArtiConstant		
Description	This is the constant value for a specific parameter. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

Example 10.2

Exemplary 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	[ECUC_Arti_00003]		
Container Name	ArtiExpression		
Parent Container	Arti		
Description	This container holds a C like expression that a debugger can evaluate. This is similar to what is already done in ORTI. Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	ArtiExpressionString [ECUC_Arti_00009]		
Parent Container	ArtiExpression		
Description	This string represents a C like expression that a debugger can evaluate. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

Example 10.3

Exemplary 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	[ECUC_Arti_00004]		
Container Name	ArtiHook		
Parent Container	Arti		
Description	This container represents an ARTI hook that is present in the module. Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	ArtiHookClass [ECUC_Arti_00010]		
Parent Container	ArtiHook		
Description	Name of the (schedule) class of macros. Classes can be one of the predefined classes or user defined. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	ArtiHookContext [ECUC_Arti_00023]		
Parent Container	ArtiHook		
Description	Name of the execution context. One of NOSUSP, SPRVSR, or USER. See also chapter "ARTI Tracing Macro". Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	ArtiHookEventName [ECUC_Arti_00011]		
Parent Container	ArtiHook		
Description	The name of the event as defined for a particular class, or an arbitrary name for generic classes. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	ArtiHookInstance [ECUC_Arti_00013]		
Parent Container	ArtiHook		
Description	Name of an instance of the (schedule) class. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			

Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	ArtiHookEventParameterTypeRef [ECUC_Arti_00012]		
Parent Container	ArtiHook		
Description	Refers to a parameter type to interpret the hook event number. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to ArtiParameterTypeMap		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	ArtiHookInstanceParameterTypeRef [ECUC_Arti_00014]		
Parent Container	ArtiHook		
Description	Refers to a parameter type to interpret the hook instance number. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to ArtiParameterTypeMap		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	

Scope / Dependency	scope: ECU
No Included Containers	

Example 10.4

Exemplary Value of an ArtiHook Container

```

<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiHook_ArtiOs_TaskStart</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF"/>/AUTOSAR/
    EcucDefs/Arti/ArtiHook</DEFINITION-REF>
  <PARAMETER-VALUES>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF"/>/AUTOSAR/EcucDefs/
        Arti/ArtiHook/ArtiHookClass</DEFINITION-REF>
      <VALUE>AR_CP_OS_TASK</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF"/>/AUTOSAR/EcucDefs/
        Arti/ArtiHook/ArtiHookContext</DEFINITION-REF>
      <VALUE>NOSUSP</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF"/>/AUTOSAR/EcucDefs/
        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/EcucDefs/
        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/EcucDefs/
        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/EcucDefs/
        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	[ECUC_Arti_00005]		
Container Name	ArtiObjectClassParameter		
Parent Container	Arti		
Description	This container represents a parameter of an Arti object class definition. Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	ArtiParameterTypeMapRef [ECUC_Arti_00016]		
Parent Container	ArtiObjectClassParameter		
Description	Refers to a parameter type to interpret the instance parameter value. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to ArtiParameterTypeMap		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

Example 10.5

Exemplary 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	[ECUC_Arti_00006]		
Container Name	ArtiObjectInstanceParameter		
Parent Container	Arti		
Description	This container represents a parameter of an Arti object instance. Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	ArtiConstantRef [ECUC_Arti_00017]		
Parent Container	ArtiObjectInstanceParameter		
Description	Refers to a constant representing the value of this parameter. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to ArtiConstant		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		

Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	ArtiExpressionRef [ECUC_Arti_00018]		
Parent Container	ArtiObjectInstanceParameter		
Description	Refers to an expression that evaluates the value of this parameter. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to ArtiExpression		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	ArtiHookRef [ECUC_Arti_00019]		
Parent Container	ArtiObjectInstanceParameter		
Description	Refers to a hook that records this parameter. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to ArtiHook		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

Example 10.6

Exemplary 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	[ECUC_Arti_00007]		
Container Name	ArtiParameterTypeMap		
Parent Container	Arti		
Description	A map of key/value pairs to map a parameter value to a display string and/or an Arti or EcuC object. Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Containers		
Container Name	Multiplicity	Scope / Dependency
ArtiParameterTypeMapPair	1..*	A key/value pair to map a parameter value to a display string and/or an Arti or EcuC object. Tags: atp.Status=draft

SWS Item	[ECUC_Arti_00020]		
Container Name	ArtiParameterTypeMapPair		
Parent Container	ArtiParameterTypeMap		
Description	A key/value pair to map a parameter value to a display string and/or an Arti or EcuC object. Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	ArtiParameterTypeMapPairInput [ECUC_Arti_00021]		
Parent Container	ArtiParameterTypeMapPair		
Description	The numerical value given by a parameter to translate. When used with Hooks (ArtiHooks), this parameter is mandatory (multiplicity 1) and its value is limited to the range of 0..65535. When used with Hooks, this parameter may be used to map the values given by "instanceParameter" and/or the "eventParameter" of the ARTI_TRACE macro. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcuIntegerParamDef		
Range	0 .. 65535		
Default Value			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	ArtiParameterTypeMapPairOutput [ECUC_Arti_00022]		
Parent Container	ArtiParameterTypeMapPair		
Description	The string to display for the Input value. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	ArtiParameterTypeMapPairInputExpressionRef [ECUC_Arti_00024]		
Parent Container	ArtiParameterTypeMapPair		
Description	Refers to an expression that evaluates to a numerical value to translate. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to ArtiExpression		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	ArtiParameterTypeMapPairOutputRef [ECUC_Arti_00025]		
Parent Container	ArtiParameterTypeMapPair		
Description	Choice Reference to ArtiOsTaskInstance, ArtiOsIsrcInstance or ArtiTaskState or OsAppMode. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Choice reference to [ArtiOsIsrcInstance,ArtiOsTaskInstance,ArtiTaskState,OsAppMode]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers

Example 10.7

Exemplary Values of an ArtiParameterTypeMap Containers

```

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

```

```

    <DEFINITION-REF DEST="ECUC-INTEGGER-PARAM-DEF"/>/AUTOSAR/
      EcucDefs/Arti/ArtiParameterTypeMap/
      ArtiParameterTypeMapPair/ArtiParameterTypeMapPairInput<
        /DEFINITION-REF>
    <VALUE>1</VALUE>
  </ECUC-TEXTUAL-PARAM-VALUE>
<ECUC-TEXTUAL-PARAM-VALUE>
  <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF"/>/AUTOSAR/
    EcucDefs/Arti/ArtiParameterTypeMap/
    ArtiParameterTypeMapPair/ArtiParameterTypeMapPairOutput
  </DEFINITION-REF>
  <VALUE>HighPriority</VALUE>
</ECUC-TEXTUAL-PARAM-VALUE>
</PARAMETER-VALUES>
</ECUC-CONTAINER-VALUE>
</SUB-CONTAINERS>
</ECUC-CONTAINER-VALUE>
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiParameterTypeMap_OsAppMode</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF"/>/AUTOSAR/
    EcucDefs/Arti/ArtiParameterTypeMap</DEFINITION-REF>
  <SUB-CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>AppModeDefault</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF"/>/AUTOSAR/
        EcucDefs/Arti/ArtiParameterTypeMap/ArtiParameterTypeMapPair
      </DEFINITION-REF>
    <PARAMETER-VALUES>
      <ECUC-TEXTUAL-PARAM-VALUE>
        <DEFINITION-REF DEST="ECUC-INTEGGER-PARAM-DEF"/>/AUTOSAR/
          EcucDefs/Arti/ArtiParameterTypeMap/
          ArtiParameterTypeMapPair/ArtiParameterTypeMapPairInput<
            /DEFINITION-REF>
        <VALUE>1</VALUE>
      </ECUC-TEXTUAL-PARAM-VALUE>
      <ECUC-TEXTUAL-PARAM-VALUE>
        <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF"/>/AUTOSAR/
          EcucDefs/Arti/ArtiParameterTypeMap/
          ArtiParameterTypeMapPair/ArtiParameterTypeMapPairOutput
        </DEFINITION-REF>
        <VALUE>OSDEFAULTAPPMODE</VALUE>
      </ECUC-TEXTUAL-PARAM-VALUE>
    </PARAMETER-VALUES>
    </ECUC-CONTAINER-VALUE>
  <ECUC-CONTAINER-VALUE>
    <SHORT-NAME>AppModeNone</SHORT-NAME>
    <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF"/>/AUTOSAR/
      EcucDefs/Arti/ArtiParameterTypeMap/ArtiParameterTypeMapPair
    </DEFINITION-REF>
  <PARAMETER-VALUES>
    <ECUC-TEXTUAL-PARAM-VALUE>
      <DEFINITION-REF DEST="ECUC-INTEGGER-PARAM-DEF"/>/AUTOSAR/
        EcucDefs/Arti/ArtiParameterTypeMap/
        ArtiParameterTypeMapPair/ArtiParameterTypeMapPairInput<
          /DEFINITION-REF>
      <VALUE>0</VALUE>
    </ECUC-TEXTUAL-PARAM-VALUE>
  </PARAMETER-VALUES>
</ECUC-CONTAINER-VALUE>
</SUB-CONTAINERS>
</ECUC-CONTAINER-VALUE>

```

```

</ECUC-TEXTUAL-PARAM-VALUE>
<ECUC-TEXTUAL-PARAM-VALUE>
  <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF"/>/AUTOSAR/
    EcucDefs/Arti/ArtiParameterTypeMap/
      ArtiParameterTypeMapPair/ArtiParameterTypeMapPairOutput
  </DEFINITION-REF>
  <VALUE>OS_APPMODE_NONE</VALUE>
</ECUC-TEXTUAL-PARAM-VALUE>
</PARAMETER-VALUES>
</ECUC-CONTAINER-VALUE>
</SUB-CONTAINERS>
</ECUC-CONTAINER-VALUE>
<ECUC-CONTAINER-VALUE>
  <SHORT-NAME>ArtiParameterTypeMap_TaskExpr</SHORT-NAME>
  <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF"/>/AUTOSAR/
    EcucDefs/Arti/ArtiParameterTypeMap</DEFINITION-REF>
  <SUB-CONTAINERS>
    <ECUC-CONTAINER-VALUE>
      <SHORT-NAME>Task_1</SHORT-NAME>
      <DEFINITION-REF DEST="ECUC-PARAM-CONF-CONTAINER-DEF"/>/AUTOSAR/
        EcucDefs/Arti/ArtiParameterTypeMap/ArtiParameterTypeMapPair
      </DEFINITION-REF>
      <PARAMETER-VALUES>
        <ECUC-TEXTUAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-INTEGGER-PARAM-DEF"/>/AUTOSAR/
            EcucDefs/Arti/ArtiParameterTypeMap/
              ArtiParameterTypeMapPair/ArtiParameterTypeMapPairInput<
            </DEFINITION-REF>
          <VALUE>&Task_1</VALUE>
        </ECUC-TEXTUAL-PARAM-VALUE>
        <ECUC-TEXTUAL-PARAM-VALUE>
          <DEFINITION-REF DEST="ECUC-STRING-PARAM-DEF"/>/AUTOSAR/
            EcucDefs/Arti/ArtiParameterTypeMap/
              ArtiParameterTypeMapPair/ArtiParameterTypeMapPairOutput
          </DEFINITION-REF>
          <VALUE>Task_1</VALUE>
        </ECUC-TEXTUAL-PARAM-VALUE>
      </PARAMETER-VALUES>
    </ECUC-CONTAINER-VALUE>
  </SUB-CONTAINERS>
</ECUC-CONTAINER-VALUE>

```

10.2.7 ArtiStates

SWS Item	[ECUC_Arti_00026]
Container Name	ArtiStates
Parent Container	Arti
Description	This container contains all states of tasks, isrs... that the EcuC uses. Tags: atp.Status=draft

Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Containers		
Container Name	Multiplicity	Scope / Dependency
ArtiTaskState	0..*	Each state used by the OS has to be listed as ArtiTaskState Parameter with a choice of the states. Tags: atp.Status=draft

SWS Item	[ECUC_Arti_00027]		
Container Name	ArtiTaskState		
Parent Container	ArtiStates		
Description	Each state used by the OS has to be listed as ArtiTaskState Parameter with a choice of the states. Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	ArtiTaskStateEnhanced [ECUC_Arti_00028]		
Parent Container	ArtiTaskState		
Description	Set to true, if the OS provides an "enhanced" state model with "READY" split to "Activated", "Preempted", "Released".		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default Value			
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

Name	ArtiTaskStateEnum [ECUC_Arti_00029]		
Parent Container	ArtiTaskState		
Description	ArtiTaskState choice of the states.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	ArtiTaskStateActivated	activated	
	ArtiTaskStatePreempted	preempted	
	ArtiTaskStateReady	ready	
	ArtiTaskStateReleased	released	
	ArtiTaskStateRunning	running	
	ArtiTaskStateSuspended	suspended	
	ArtiTaskStateWaiting	waiting	
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3 ARTI Generic Container

Module SWS Item	ECUC_ArtiGeneric_00001		
Module Name	ArtiGeneric		
Module Description	The ArtiGeneric contains definitions for generic objects, i.e. not belonging to a standard AUTOSAR module.		
Post-Build Variant Support	false		
Supported Config Variants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE		
Included Containers			
Container Name	Multiplicity	Scope / Dependency	
ArtiGenericComponent Class	0..*	The class definition describes the layout of the object (similar to a "class" definition in C). Tags: atp.Status=draft	
ArtiGenericComponent Instance	0..*	The instance definition describes a specific instantiated object. Tags: atp.Status=draft	

Example 10.8

Exemplary 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>
        </ECUC-MODULE-CONFIGURATION-VALUES>
      </ELEMENTS>
    </AR-PACKAGE>
  </AR-PACKAGES>
</AUTOSAR>

```

```

    </ECUC-CONTAINER-VALUE>
  </CONTAINERS>
</ECUC-MODULE-CONFIGURATION-VALUES>
<...>

```

10.3.1 ArtiGenericComponentClass

SWS Item	[ECUC_ArtiGeneric_00002]		
Container Name	ArtiGenericComponentClass		
Parent Container	ArtiGeneric		
Description	The class definition describes the layout of the object (similar to a "class" definition in C). Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	ArtiGenericComponentClassName [ECUC_ArtiGeneric_00004]		
Parent Container	ArtiGenericComponentClass		
Description	Name of the class. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
ArtiGenericComponentClassParameter	0..*	Parameter definition of a class. Tags: atp.Status=draft

SWS Item	[ECUC_ArtiGeneric_00006]		
Container Name	ArtiGenericComponentClassParameter		
Parent Container	ArtiGenericComponentClass		
Description	Parameter definition of a class. Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	ArtiGenericComponentClassNameParameterName [ECUC_ArtiGeneric_00007]		
Parent Container	ArtiGenericComponentClassParameter		
Description	Name of the parameter. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	ArtiParamTypeMapRef [ECUC_ArtiGeneric_00009]		
Parent Container	ArtiGenericComponentClassParameter		
Description	Refers to a parameter type to interpret the parameter value. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to ArtiParamTypeMap		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	Pre-compile time	X	All Variants
	Link time	-	
	Post-build time	-	
Scope / Dependency	scope: ECU		
No Included Containers			

Example 10.9

Exemplary 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>
        </ECUC-REFERENCE-VALUE>
      </REFERENCE-VALUES>
    </ECUC-CONTAINER-VALUE>
  </SUB-CONTAINERS>
</ECUC-CONTAINER-VALUE>

```

```

    <VALUE-REF DEST="ECUC-CONTAINER-VALUE"/>/Vendor1/Vendor1Arti
      /ArtiParamTypeMap_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/
          ArtiParamTypeMapRef</DEFINITION-REF>
    <VALUE-REF DEST="ECUC-CONTAINER-VALUE"/>/Vendor1/Vendor1Arti
      /ArtiParamTypeMap_ThingStart</VALUE-REF>
  </ECUC-REFERENCE-VALUE>
</REFERENCE-VALUES>
</ECUC-CONTAINER-VALUE>
</SUB-CONTAINERS>
</ECUC-CONTAINER-VALUE>

```

10.3.2 ArtiGenericComponentInstance

SWS Item	[ECUC_ArtiGeneric_00003]
Container Name	ArtiGenericComponentInstance
Parent Container	ArtiGeneric
Description	The instance definition describes a specific instantiated object. Tags: atp.Status=draft
Post-Build Variant Multiplicity	false

Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	ArtiGenericComponentInstanceName [ECUC_ArtiGeneric_00010]		
Parent Container	ArtiGenericComponentInstance		
Description	Name of the instance. Tags: atp.Status=draft		
Multiplicity	1		
Type	EcucStringParamDef		
Default Value			
Regular Expression			
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Name	ArtiGenericComponentClassRef [ECUC_ArtiGeneric_00012]		
Parent Container	ArtiGenericComponentInstance		
Description	Refers to a ArtGenericClass of which this object is instantiated. Tags: atp.Status=draft		
Multiplicity	1		
Type	Reference to ArtiGenericComponentClass		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
ArtiGenericComponentInstanceParameter	0..*	Parameter definition of an instance. Tags: atp.Status=draft

SWS Item	[ECUC_ArtiGeneric_00013]
-----------------	--------------------------

Container Name	ArtiGenericComponentInstanceParameter		
Parent Container	ArtiGenericComponentInstance		
Description	Parameter definition of an instance. Tags: atp.Status=draft		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Name	ArtiConstantRef [ECUC_ArtiGeneric_00014]		
Parent Container	ArtiGenericComponentInstanceParameter		
Description	Refers to an ArtiConstant that represents the value of this parameter. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to ArtiConstant		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

Name	ArtiExpressionRef [ECUC_ArtiGeneric_00015]		
Parent Container	ArtiGenericComponentInstanceParameter		
Description	Refers to an ArtiExpression that evaluates the value of this parameter. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to ArtiExpression		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		

Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

Name	ArtiGenericComponentClassParameterRef [ECUC_ArtiGeneric_00016]		
Parent Container	ArtiGenericComponentInstanceParameter		
Description	Refers to an ArtiGenericComponentClassParameter that defines this parameter. Tags: atp.Status=draft		
Multiplicity	0..*		
Type	Reference to ArtiGenericComponentClassParameter		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

Name	ArtiHookRef [ECUC_ArtiGeneric_00017]		
Parent Container	ArtiGenericComponentInstanceParameter		
Description	Refers to a hook that records this parameter. Tags: atp.Status=draft		
Multiplicity	0..1		
Type	Reference to ArtiHook		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	

Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency			

No Included Containers

Example 10.10

Exemplary 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>
    </ECUC-REFERENCE-VALUE>
  </SUB-CONTAINERS>

```

```

<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

```

1  #ifndef _TOOL_VENDOR_BINDING_H_
2  #define _TOOL_VENDOR_BINDING_H_
3
4  #include <stdint.h>
5
6  #if defined VENDOR_A
7  /* ARTI Trace Macro */
8  #   define ARTI_TRACE(_contextName, _className, _instanceName,
9      instanceParameter, _eventName, event_value) \
10      (void)TraceImpl ## _ ## _className ## _ ## _eventName ## _ ##
11      _instanceName ## _ ## _contextName( (instanceParameter), (
12      event_value) )
13
14  /* Prototypes for AR_CP_OS_TASK */
15  void TraceImpl_AR_CP_OS_TASK_OsTask_Start_OS_SHORT_NAME_SPRVSR(uint32_t
16      instanceParameter, uint32_t event_value);
17  void TraceImpl_AR_CP_OS_TASK_OsTask_Stop_OS_SHORT_NAME_SPRVSR(uint32_t
18      instanceParameter, uint32_t event_value);
19
20  void TraceImpl_AR_CP_OS_TASK_OsTask_Start_OS_SHORT_NAME_USER(uint32_t
21      instanceParameter, uint32_t event_value);
22  void TraceImpl_AR_CP_OS_TASK_OsTask_Stop_OS_SHORT_NAME_USER(uint32_t
23      instanceParameter, uint32_t event_value);
24
25  void TraceImpl_AR_CP_OS_TASK_OsTask_Start_OS_SHORT_NAME_NOSUSP(uint32_t
26      instanceParameter, uint32_t event_value);
27  void TraceImpl_AR_CP_OS_TASK_OsTask_Stop_OS_SHORT_NAME_NOSUSP(uint32_t
28      instanceParameter, uint32_t event_value);
29
30  /* Prototypes for Ignition_Control */
31  void TraceImpl_Ignition_Control_IgnitionStart_Cylinder0_USER(uint32_t
32      instanceParameter, uint32_t event_value);
33  void TraceImpl_Ignition_Control_IgnitionStop_Cylinder0_USER(uint32_t
34      instanceParameter, uint32_t event_value);
35
36  void TraceImpl_Ignition_Control_IgnitionStart_Cylinder1_USER(uint32_t
37      instanceParameter, uint32_t event_value);
38  void TraceImpl_Ignition_Control_IgnitionStop_Cylinder1_USER(uint32_t
39      instanceParameter, uint32_t event_value);
40
41  #elif defined VENDOR_B
42  /* ARTI Trace Macro */
43  #   define ARTI_TRACE(_contextName, _className, _instanceName,
44      instanceParameter, _eventName, event_value) \
45      (void)TraceImpl ## _ ## _className ## _ ## _contextName( (
46      _instanceName), (instanceParameter), (_eventName), (
47      event_value) )
48
49  /* Defines for AR_CP_OS_TASK */
50  /* Instance Names */
51  #define OS_SHORT_NAME (0)
52  /* Event Names */
53  #define OsTask_Start (0)
54  #define OsTask_Stop (1)

```

```

39
40 /* Defines for Ignition_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_TASK */
49 void TraceImpl_AR_CP_OS_TASK_SPRVSR(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value);
50 void TraceImpl_AR_CP_OS_TASK_USER(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value);
51 void TraceImpl_AR_CP_OS_TASK_NOSUSP(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value);
52
53 /* Prototypes for Ignition_Control */
54 void TraceImpl_Ignition_Control_SPRVSR(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value);
55 void TraceImpl_Ignition_Control_USER(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value);
56 void TraceImpl_Ignition_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_TASK_OsTask_Start_OS_SHORT_NAME_SPRVSR(uint32_t
    instanceParameter, uint32_t event_value)
13 {
14     __disable();
15     TraceImpl_AR_CP_OS_TASK_OsTask_Start_OS_SHORT_NAME_NOSUSP(
    instanceParameter, event_value);
16     __enable();
17 }
18

```

```
19 void TraceImpl_AR_CP_OS_TASK_OsTask_Stop_OS_SHORT_NAME_SPRVSR(uint32_t
    instanceParameter, uint32_t event_value)
20 {
21     __disable();
22     TraceImpl_AR_CP_OS_TASK_OsTask_Stop_OS_SHORT_NAME_NOSUSP (
        instanceParameter, event_value);
23     __enable();
24 }
25
26 void TraceImpl_AR_CP_OS_TASK_OsTask_Start_OS_SHORT_NAME_USER(uint32_t
    instanceParameter, uint32_t event_value)
27 {
28     SuspendAllInterrupts();
29     TraceImpl_AR_CP_OS_TASK_OsTask_Start_OS_SHORT_NAME_NOSUSP (
        instanceParameter, event_value);
30     ResumeAllInterrupts();
31 }
32
33 void TraceImpl_AR_CP_OS_TASK_OsTask_Stop_OS_SHORT_NAME_USER(uint32_t
    instanceParameter, uint32_t event_value)
34 {
35     SuspendAllInterrupts();
36     TraceImpl_AR_CP_OS_TASK_OsTask_Stop_OS_SHORT_NAME_NOSUSP (
        instanceParameter, event_value);
37     ResumeAllInterrupts();
38 }
39
40 void TraceImpl_AR_CP_OS_TASK_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_TASK_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_Ignition_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_Ignition_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_Ignition_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_Ignition_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_TASK_SPRVSR(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value)
95 {
96     __disable();
97     TraceImpl_AR_CP_OS_TASK_NOSUSP(instanceName, instanceParameter,
        eventName, event_value);
98     __enable();
99 }
100
101 void TraceImpl_AR_CP_OS_TASK_USER(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value)
102 {
103     SuspendAllInterrupts();
104     TraceImpl_AR_CP_OS_TASK_NOSUSP(instanceName, instanceParameter,
        eventName, event_value);
105     ResumeAllInterrupts();
106 }
107
108 void TraceImpl_AR_CP_OS_TASK_NOSUSP(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value)
109 {
110     (void)instanceName; // avoid warning "unused parameter"
111     (void)instanceParameter; // avoid warning "unused parameter"
112     (void)eventName; // avoid warning "unused parameter"
```

```

113     (void)event_value; // avoid warning "unused parameter"
114
115     // actual tracing code goes here
116 }
117
118 void TraceImpl_Ignition_Control_SPRVSR(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value)
119 {
120     __disable();
121     TraceImpl_Ignition_Control_NOSUSP(instanceName, instanceParameter,
    eventName, event_value);
122     __enable();
123 }
124
125 void TraceImpl_Ignition_Control_USER(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value)
126 {
127     SuspendAllInterrupts();
128     TraceImpl_Ignition_Control_NOSUSP(instanceName, instanceParameter,
    eventName, event_value);
129     ResumeAllInterrupts();
130 }
131
132 void TraceImpl_Ignition_Control_NOSUSP(uint32_t instanceName, uint32_t
    instanceParameter, uint32_t eventName, uint32_t event_value)
133 {
134     (void)instanceName; // avoid warning "unused parameter"
135     (void)instanceParameter; // avoid warning "unused parameter"
136     (void)eventName; // avoid warning "unused parameter"
137     (void)event_value; // avoid warning "unused parameter"
138
139     // actual tracing code goes here
140 }
141
142 #else
143
144 #endif

```

B.1.2 ARTI OS Instrumentation

Listing B.3: Example for OS instrumentation header

```

1 #ifndef _OS_H_
2 #define _OS_H_
3
4 #define TASK(_taskname)      void OS_TASK ## _ ## _taskname(void)
5
6 void SuspendAllInterrupts(void);
7 void ResumeAllInterrupts(void);
8
9 void StartOS(void);
10
11 #endif

```

Listing B.4: Example for OS instrumentation source

```

1 #include "user_main.h"
2 #include "tool-vendor_binding.h"
3
4 void SuspendAllInterrupts(void)
5 {
6     // ...
7 }
8
9 void ResumeAllInterrupts(void)
10 {
11     // ...
12 }
13
14 void StartOS(void)
15 {
16     const int myCoreId = 0;
17     const int OS_TASK_Task_Cylinder0_ID = 2;
18
19     // for testing the ARTI interface, we call the task UserTask1 here
20     // directly (rather than implementing an OS)
21     ARTI_TRACE(NOSUSP, AR_CP_OS_TASK, OS_SHORT_NAME, myCoreId,
22               OsTask_Start, OS_TASK_Task_Cylinder0_ID);
23     OS_TASK_Task_Cylinder0();
24     ARTI_TRACE(NOSUSP, AR_CP_OS_TASK, OS_SHORT_NAME, myCoreId,
25               OsTask_Stop, OS_TASK_Task_Cylinder0_ID);
26 }

```

B.1.3 ARTI Arbitrary Instrumentation

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

```

1 #ifndef _USER_MAIN_H_
2 #define _USER_MAIN_H_
3
4 #include "os.h"
5 extern TASK(Task_Cylinder0);
6 extern TASK(Task_Cylinder1);
7
8 #endif

```

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

```

1 #include <stdlib.h>
2
3 #include "os.h"
4 #include "tool-vendor_binding.h"
5
6 TASK(Task_Cylinder0)
7 {
8     ARTI_TRACE(USER, Ignition_Control, Cylinder0, 0, IgnitionStart,
9               53);
10    // inject
11    ARTI_TRACE(USER, Ignition_Control, Cylinder0, 0, IgnitionStop, 53)
12    ;
13 }

```

```

13 TASK(Task_Cylinder1)
14 {
15     ARTI_TRACE(USER, Ignition_Control, Cylinder1, 0, IgnitionStart,
16         77);
17     // inject
18     ARTI_TRACE(USER, Ignition_Control, Cylinder1, 0, IgnitionStop, 77)
19     ;
20 }
21 int main(void)
22 {
23     StartOS();
24     exit(EXIT_SUCCESS);
25     return -1;
26 }

```

B.2 ARXML Representation of Instrumentation

Example B.1

Exemplary 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_TASK</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>
  </REFERENCE-VALUES>
</ECUC-CONTAINER-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

Exemplary 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

Exemplary 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>
  </PARAMETER-VALUES>
</ECUC-CONTAINER-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

Exemplary 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

Exemplary 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>

```

C Expression Syntax

This section describes the grammar of Expressions using the Extended Backus-Naur Form.

```

expression           = logical_OR_expression { '?' expression ':' expression } ;
logical_OR_expression = logical_AND_expression { '||' logical_AND_expression } ;
logical_AND_expression = inclusive_OR_expression { '&&' inclusive_OR_expression } ;
inclusive_OR_expression = exclusive_OR_expression { '|' exclusive_OR_expression } ;
exclusive_OR_expression = AND_expression { '^' AND_expression } ;
AND_expression       = equality_expression { '&' equality_expression } ;
equality_expression  = relational_expression { ( '==' | '!=' )
    relational_expression } ;
relational_expression = shift_expression { ( '<' | '>' | '<=' | '>=' )
    shift_expression } ;
shift_expression     = additive_expression { ( '<<' | '>>' )
    additive_expression } ;
additive_expression  = multiplicative_expression { ( '+' | '-' )
    multiplicative_expression } ;
multiplicative_expression = cast_expression { ( '*' | '/' | '%' )
    cast_expression } ;
cast_expression      = { '(' type_name ')' } unary_expression ;
unary_expression     = postfix_expression | unary_operator cast_expression
    | 'sizeof' unary_expression | 'sizeof' '(' type_name ')' ;
unary_operator       = '&' | '*' | '+' | '-' | '~' | '!';
postfix_expression   = primary_expression { '[' expression ']' | ( '.' | '->' )
    appl_identifier } ;
primary_expression   = appl_identifier | constant | '(' expression ')' ;
constant             = integer_constant | character_constant |
    floating_constant | enumeration_constant ;
type_name            = type_specifier { type_specifier } [ '*' ] ;
type_specifier       = 'void' | 'char' | 'short' | 'int' | 'long' | 'float' |
    'double' | 'signed' | 'unsigned' | type_def_name ;
type_def_name        = appl_identifier;
    
```

Where:

<code>integer_constant</code>	represents an integer number, where the standard C convention is used for decimal, hexadecimal and octal notation.
<code>character_constant</code>	follows the C definition for a character, including the support of all standard escape sequences, such as <code>'\n'</code> , <code>'\t'</code> etc.
<code>floating_constant</code>	follows the C definition for a floating point number.
<code>enumeration_constant</code>	follows the C definition for an "enum" constant.
<code>appl_identifier</code>	represents any C identifier and represents application symbols. These symbols rely on symbolic information retrieved from the debug information of the application and must have 'external linkage' scope (e.g. global C variables). The symbol value is only valid after the application has executed its initialization ▽
	△ phase (typically this is the system startup code before reaching the applications entry point, which is <code>main()</code> in C). The only exception to this constraint is when using the unary address-operator (&).

Further rules:

- Whitespace (blank, TAB) between terminals is ignored.
- All keywords and identifiers are case-sensitive.