ARTI (ASAM/AUTOSAR Run-Time Interface)

A brief introduction to ARTI

by Peter Gliwa, GLIWA GmbH
And Rudi Dienstbeck, Lauterbach GmbH

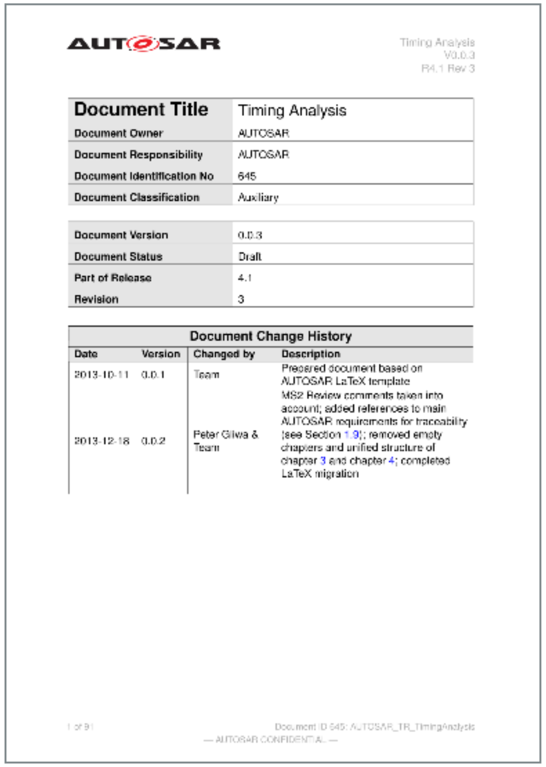
AUTOSAR and timing

- **Timing Extensions**
  ”TIMEX”; since AUTOSAR 4.0
  → Allow specification of timing requirements

- **Timing Analysis**
  First released with 4.1.3
  → Use-cases based guide to timing

- **AUTOSAR OS**
  Contains timing protection mechanisms
  → As of 4.2.2: Execution-, Locking- and
  Inter-Arrival Time Protection

- **ARTI** (AUTOSAR/ASAM Run-Time Interface)
  not a standard yet (probably in 2017)
  → more details later



Timing Analysis
V0.0.3
R4.1 Rev 3

Document Title	Timing Analysis
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	645
Document Classification	Auxiliary

Document Version	0.0.3
Document Status	Draft
Part of Release	4.1
Revision	3

Document Change History

Date	Version	Changed by	Description
2013-10-11	0.0.1	Team	Prepared document based on AUTOSAR LaTeX template
2013-12-18	0.0.2	Peter Gliwa & Team	MS2 Review comments taken into account; added references to main AUTOSAR requirements for traceability (see Section 1.9); removed empty chapters and unified structure of chapter 3 and chapter 4; completed LaTeX migration

1 of 91 Document ID 645: AUTOSAR_TR_TimingAnalysis
— AUTOSAR CONFIDENTIAL —

Timing Analysis document

Other standards

- **ORTI**: OSEK Real-Time Interface: brings OS awareness to debuggers/tracers
  - outdated (“OSEK not AUTOSAR”)
  - no support of tracing runnables, no multi-core support
  - only running state of TASKs can be captured

- Other open trace formats currently not widely used in automotive
  - **CTF**: common trace format, rather common for high-performance computing
  - **LTTng**: open source tracing framework for Linux
  - **BTF**: “best trace format” as defined by the AMALTHEA research project

- Vendor specific formats
  - quite a few different formats
  - often based on older standards like ORTI
  - not AUTOSAR or ASAM standardised

ARTI: ASAM Run-Time Interface

Why another standard?

- **Non-standardized interfaces** require **costly** individual adaptation. This is true for
  - vendor-specific, **non standardized trace formats** and
  - OS-specific interfaces for **instrumentation-based tracing**.

- No existing standard supports
  - Non-OS AUTOSAR events (e.g. RTE)
  - TIMEX
  - AUTOSAR Aadaptive Platform
  - Non-AUTOSAR Systems

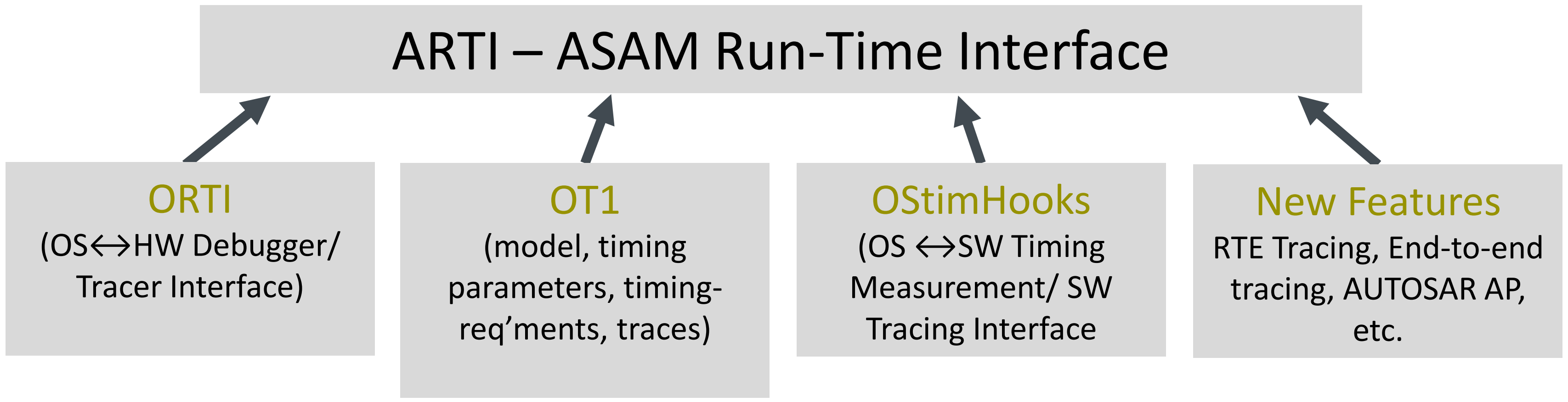
- Strong demand from **OEMs** and **Suppliers** for a unified approach.

ARTI: ASAM Run-Time Interface

- A new ASAM Project

“ARTI – ASAM Run-Time Interface”

- It aims at **creating a new standard** which becomes available as soon as the project successfully completes.


ARTI – ASAM Run-Time Interface
ORTI
(OS↔HW Debugger/ Tracer Interface)
OT1
(model, timing parameters, timing-req’ments, traces)
OStimHooks
(OS ↔SW Timing Measurement/ SW Tracing Interface
New Features
RTE Tracing, End-to-end tracing, AUTOSAR AP, etc.

ARTI goals

- **Debugging** – halting a system, either as a whole or in parts, for the purpose of
  - inspecting the contents of the system in a frozen state
  - single stepping, setting breakpoints, starting and stopping in C or Assembly code
- **Tracing** – collecting run-time information over a certain period of time
  - either as a pure software solution, or with hardware assistance
  - may include processor instruction trace, OS scheduling trace, and/or pure data trace
  - including time-stamping for further timing analysis
- **Timing Measurement** – capturing of timing information
  - by instrumentation, e.g. via Pre-/PostTaskHooks or other hooks or callouts or
  - by dedicated hardware support, e.g. hardware performance counters
  - does not stop execution
- **Profiling** – gaining timing parameters/timing statistics
  - of functions, tasks, runnables, modules etc.
  - possibly with minimum/maximum/average statistics
  - possibly with worst case analysis
  - possibly calculated out of trace data, repeated snapshots or Timing Measurement

ARTI aspects to consider

- ARTI shall support
  - Multi-core
  - Runnables
  - Instrumentation-based tracing and measurement solutions
  - The actual AUTOSAR-OS implementation
  - TIMEX
  - debugging and tracing beyond ECU level, e.g. end-to-end timing taking several ECUs and buses into account
  - AP (adaptive platform)
  - Non-AUTOSAR systems

ARTI: who is behind it?

AUTOSAR/ OS vendors

Timing Tool vendors

Debug/Trace Tool vendors

Users, AUTOSAR experts

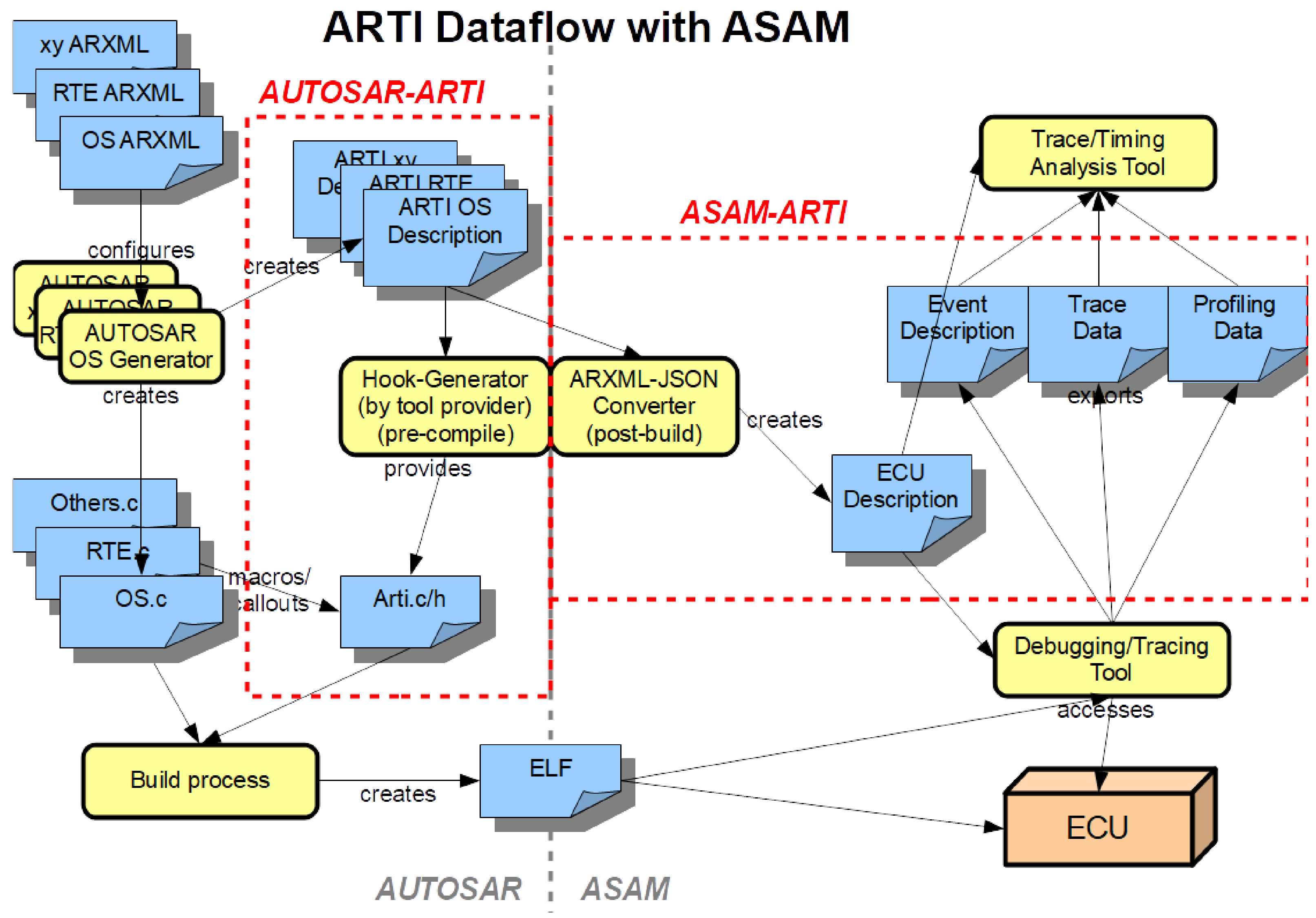
Relevance for the market

- Most embedded applications come with
  - Timing requirements
  - The need to understand and debug the software
  - The need for optimization
- These aspects are not limited to automotive

- Important automotive embedded software players are involved already
  - Lauterbach -is the world’s #1 debugger vendor
  - Elektrobit, ETAS and Vector cover >95% of the automotive RTOS market
  - BOSCH and Conti are among the biggest automotive tier-1s in the world

- The ARTI tool vendors strive for more relevance in other markets

ARTI Dataflow



Deliverables

- Specification
  - Explanatory aspects
  - UML models
  - Examples
- Schema files
- Prototype(s) demonstrating the interfaces and tools in (inter-) action

Standardization approach

- Relation to AUTOSAR
  - ASAM ARTI smoothly interfaces to AUTOSAR ARTI (cf. A2L)
  - ASAM ARTI is independent of AUTOSAR ARTI, can “live” without AUTOSAR

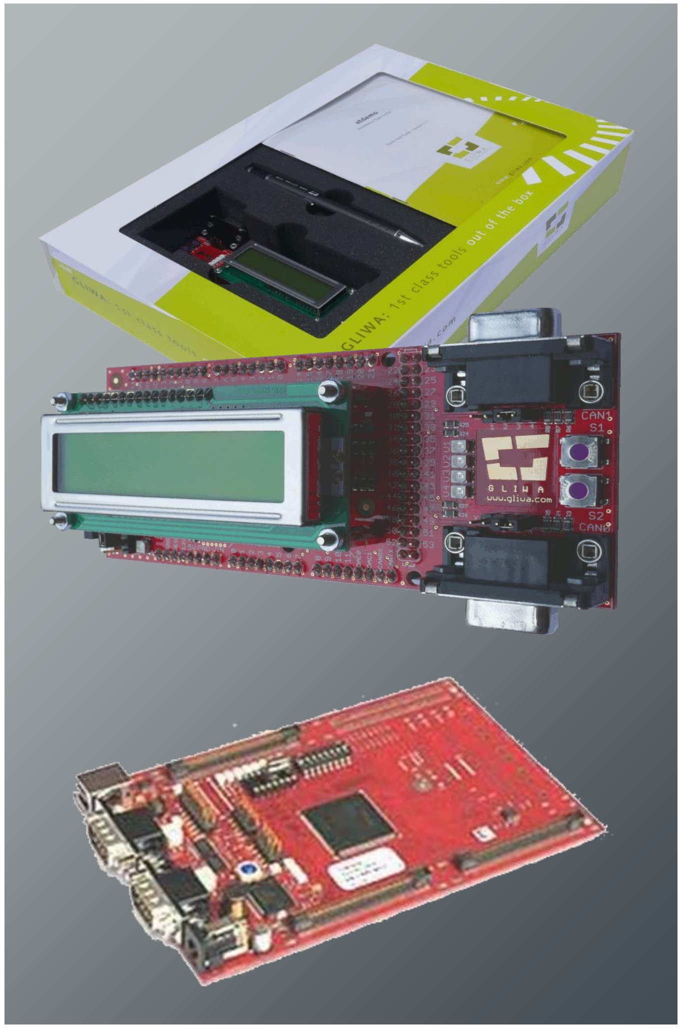
- Within AUTOSAR ARTI, already two subgroups exist
  - ARTI Hooks and ARXML
    → in a future set-up this becomes the **AUTOSAR** ARTI
  - ARTI data exchange formats
    → in a future set-up this becomes the **ASAM** ARTI

ARTI: current status

- AUTOSAR-ARTI first version specified in AUTOSAR 4.4
- Currently waiting for 4.4 implementation in OS
- Further planning:
  - ASAM-ARTI Project Start: February 2019
  - ASAM-ARTI Project Release: December 2019

ARTI: reference platform

- Demo platform for automotive multi-core SW development
- Infineon AURIX TC27x
- **GLIWA ATdemo** (TC275TP, <u>production</u> device):
  Multi-core demo software incl. development environment allowing to **build, flash and analyze application in minutes**
- **Infineon TriBoard** (TC277TF, <u>emulation</u> device):
  Second evaluation platform allowing **MCDS hardware-based tracing**
- Both platforms are code compatible



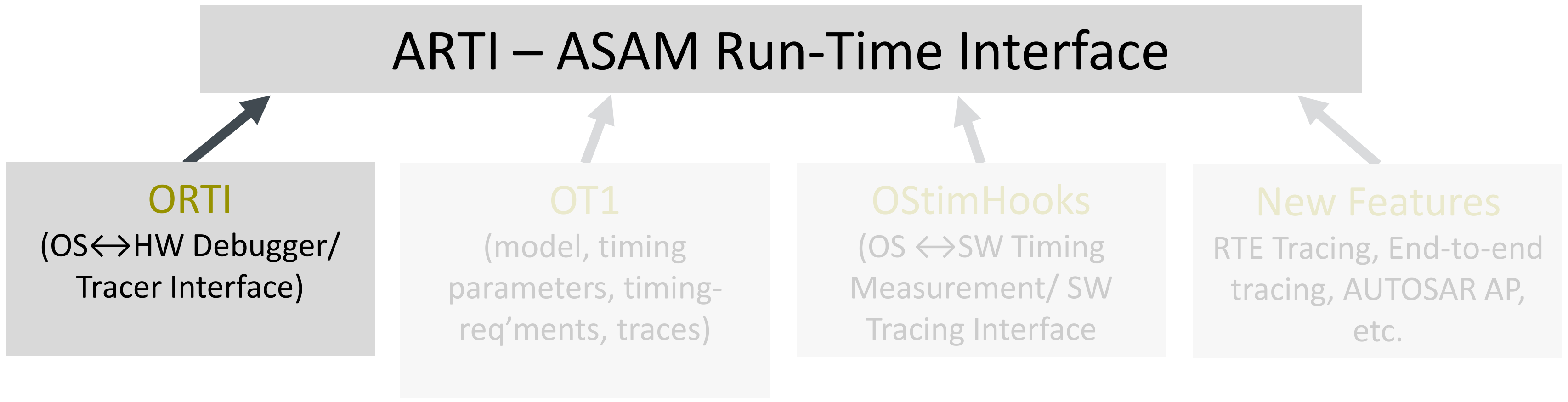
ARTI: details on **ORTI** (one of the building blocks)

- ORTI (OSEK Run-Time Interface)
  - ORTI File informs the Debug/Trace Tool about:

    Structure of the OS (Tasks, ISR2, Stacks, SchedulingTables,…)

    OS Status Update Signaling of the OS, e.g. which Variables are used for Signaling, Encoding.
  - ORTI contains symbolic Information

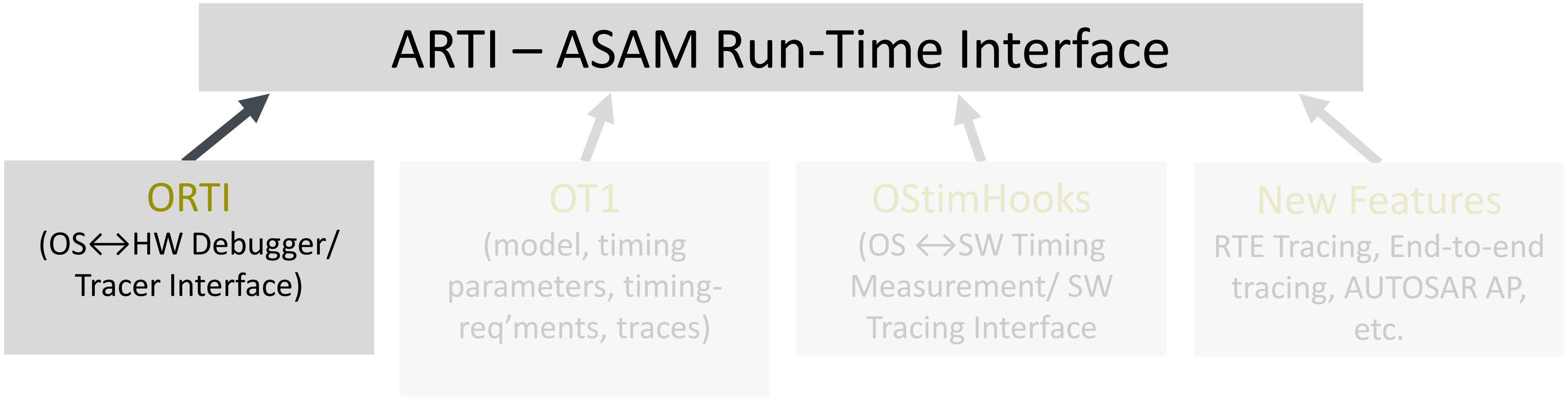
    Address Information for Symbols obtained from ELF File.


ARTI – ASAM Run-Time Interface
ORTI
(OS↔HW Debugger/ Tracer Interface)
OT1
(model, timing parameters, timing-req'ments, traces)
OStimHooks
(OS ↔SW Timing Measurement/ SW Tracing Interface
New Features
RTE Tracing, End-to-end tracing, AUTOSAR AP, etc.

ARTI: details on **ORTI** (example)

- Example: Signaling of currently running Task and ISR2 on Core 0.

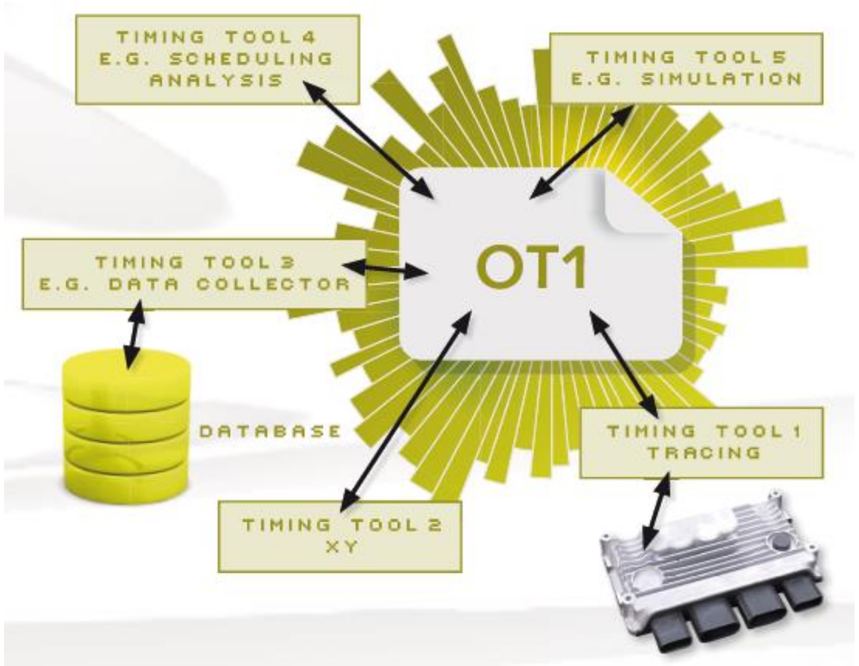
```
                          OS TC277
                              {
RUNNINGTASK[0] = "OS_kernelArray[0].taskCurrent";
 RUNNINGISR2[0] = "OS_kernelArray[0].isrCurrent";
                              …
```



ARTI – ASAM Run-Time Interface
ORTI
(OS↔HW Debugger/ Tracer Interface)
OT1
(model, timing parameters, timing-req'ments, traces)
OStimHooks
(OS ↔SW Timing Measurement/ SW Tracing Interface
New Features
RTE Tracing, End-to-end tracing, AUTOSAR AP, etc.

ARTI: details on **OT1** (one of the building blocks)

- **Open data exchange format for**
  - **System configuration** (tasks, priorities, runnables, etc. or buses, messages, etc.)
  - **Traces** (log of e.g. scheduling related events)
  - **Timing information** (core execution time, response times, etc.)
  - **Timing requirements** (e.g. max. allowed response times)
- **Any tool can provide/retrieve information**

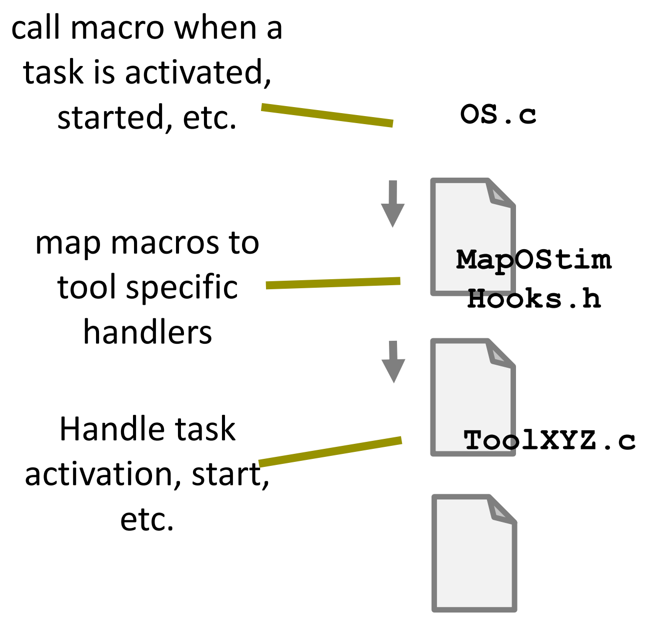


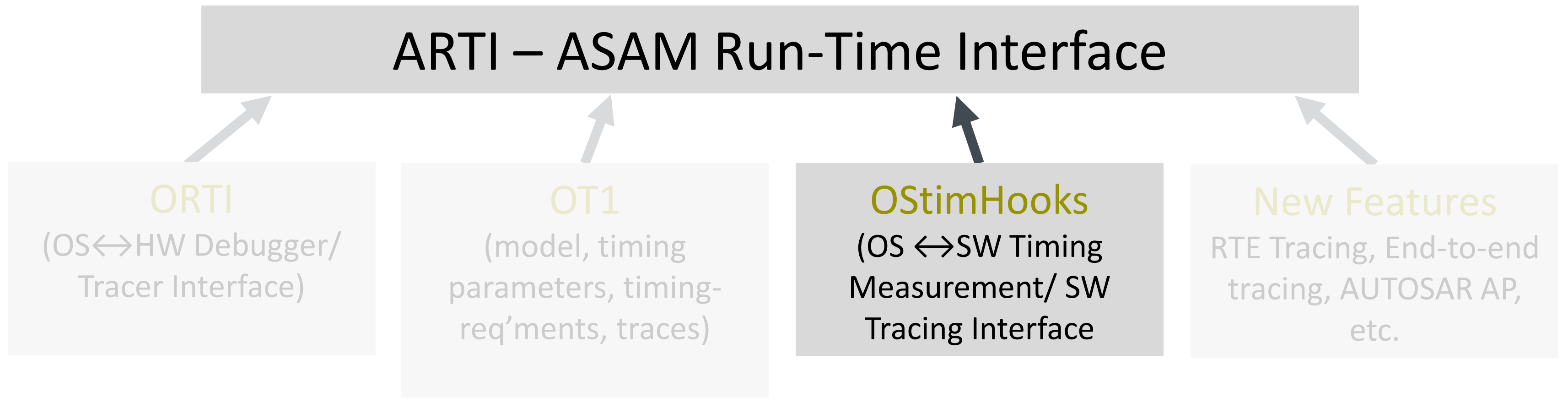
ARTI – ASAM Run-Time Interface

ORTI (OS↔HW Debugger/ Tracer Interface)	OT1 (model, timing parameters, timing-req'ments, traces)	OStimHooks (OS ↔SW Timing Measurement/ SW Tracing Interface	New Features RTE Tracing, End-to-end tracing, AUTOSAR AP, etc.

ARTI: details on **OStimHooks** (one of the building blocks)

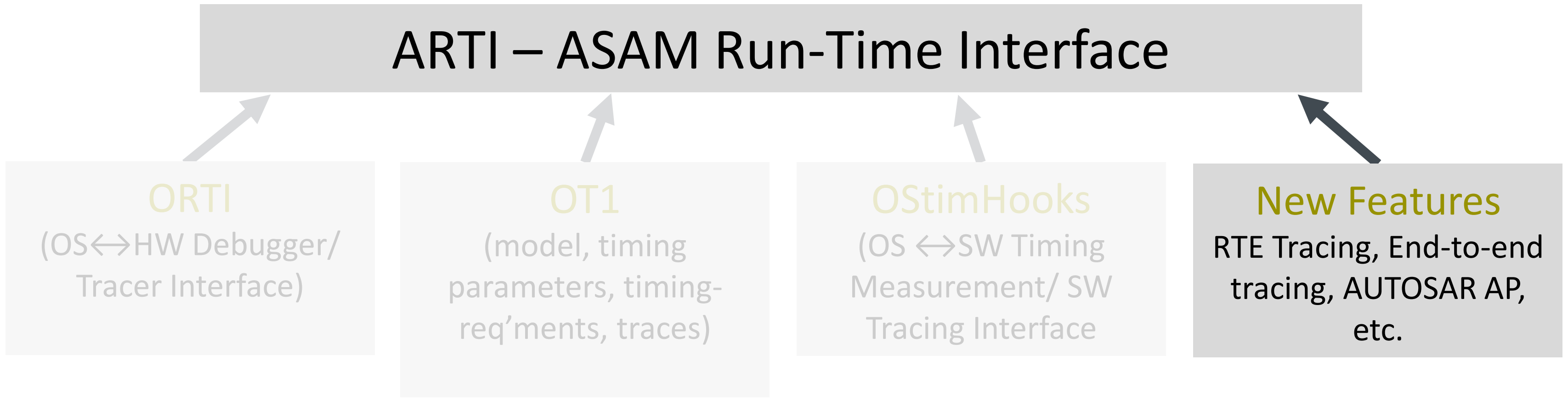
- Standard defined in 2010 as an **interface between** the **OS** and any **Timing Measurement** or **scheduling Tracing tool**.
- The OS is required to define **macros/callouts** for
  - TASKs: activation, failed activation, start and termination
  - ISRs: start, end
- Adapted by few OS vendors


call macro when a task is activated, started, etc.
OS.c
map macros to tool specific handlers
MapOStim Hooks.h
Handle task activation, start, etc.
ToolXYZ.c


ARTI – ASAM Run-Time Interface
ORTI
(OS↔HW Debugger/ Tracer Interface)
OT1
(model, timing parameters, timing-req'ments, traces)
OStimHooks
(OS ↔SW Timing Measurement/ SW Tracing Interface
New Features
RTE Tracing, End-to-end tracing, AUTOSAR AP, etc.

ARTI: details on **New Features**

- Add not only OS awareness to tracing but system awareness
  - RTE events/communication
  - SW-C related information/filtering
- Support end-to-end timing, i.e. synchronize traces of several buses/ECUs
- Support AUTOSAR AP ("Adaptive Platform", the future standard for high performance ECUs and more flexibility, e.g. dynamic task creation at run-time)


ARTI – ASAM Run-Time Interface
ORTI
(OS↔HW Debugger/ Tracer Interface)
OT1
(model, timing parameters, timing-req'ments, traces)
OStimHooks
(OS ↔SW Timing Measurement/ SW Tracing Interface
New Features
RTE Tracing, End-to-end tracing, AUTOSAR AP, etc.

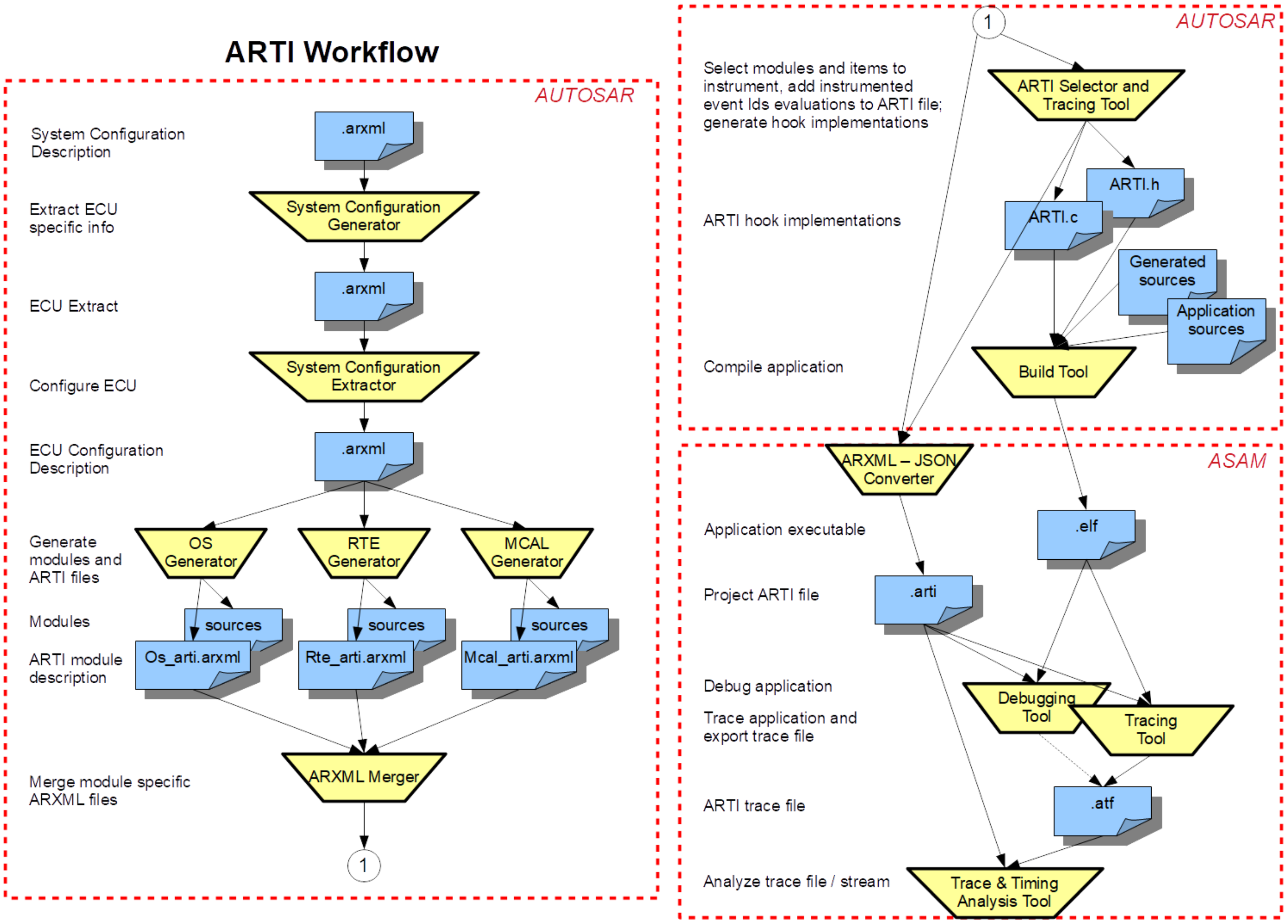
Thank you

- Any questions, remarks?

Please contact peter.gliwa@gliwa.com or rudi.dienstbeck@lauterbach.com

Backup

ARTI Workflow


ARTI Workflow
AUTOSAR
System Configuration Description
.arxml
Extract ECU specific info
System Configuration Generator
ECU Extract
.arxml
Configure ECU
System Configuration Extractor
ECU Configuration Description
.arxml
Generate modules and ARTI files
OS Generator
RTE Generator
MCAL Generator
Modules
sources
sources
sources
ARTI module description
Os_arti.arxml
Rte_arti.arxml
Mcal_arti.arxml
Merge module specific ARXML files
ARXML Merger
1
1
AUTOSAR
Select modules and items to instrument, add instrumented event Ids evaluations to ARTI file; generate hook implementations
ARTI Selector and Tracing Tool
ARTI.h
ARTI.c
ARTI hook implementations
Generated sources
Application sources
Compile application
Build Tool
ASAM
ARXML – JSON Converter
Application executable
.elf
Project ARTI file
.arti
Debug application
Debugging Tool
Trace application and export trace file
Tracing Tool
ARTI trace file
.atf
Analyze trace file / stream
Trace & Timing Analysis Tool