EXPLORING THE RUNTIME BEHAVIOUR OF ECU SOFTWARE

Timing Analysis for Embedded AUTOSAR Systems Using OS and RTE Profiling

When developing automotive electronic control units (ECUs), developing error-free software is as essential as understanding the runtime behavior of the application. This is crucial for optimizing resource utilization and performance, and ultimately for meeting all safety requirements. In this article, we present how to perform timing analysis with a solution jointly developed by Vector and Lauterbach based on Vector's AUTOSAR embedded software MICROSAR. The solution is available for a broad range of automotive microcontrollers, such as Infineon AURIX™, Renesas RH850 or Arm® Cortex®-M based MCUs.

Introduction

Microcontrollers like Infineon AURIX™, Renesas RH850 or Arm® Cortex®-M based MCUs are widely used in the automotive industry for the development of electronic control units (ECU), and other applications. For analyzing the runtime behavior of an application, appropriate tools are required to measure execution times and check timing requirements. The user must record hardware events at the instruction level and process the trace data into system events at an abstracted level. Based on this, a timing tool can compare the results with the system requirements as part of the requirement analysis.

Timing Analysis with Lauterbach TRACE32® and Vector's TA Tool Suite

To meet all these requirements, Vector and Lauterbach have jointly developed a solution for Vector's AUTOSAR embedded software MICROSAR. It uses Lauterbach's TRACE32® as well as Vector's TA Tool Suite for performing the Timing Analysis.

This solution is based on the AUTOSAR Run Time Interface (ARTI), which has a similar approach to its predecessor ORTI (OSEK Run-Time Interface). However, it extends the trace capabilities and addresses some shortcomings of the ORTI-based analysis.

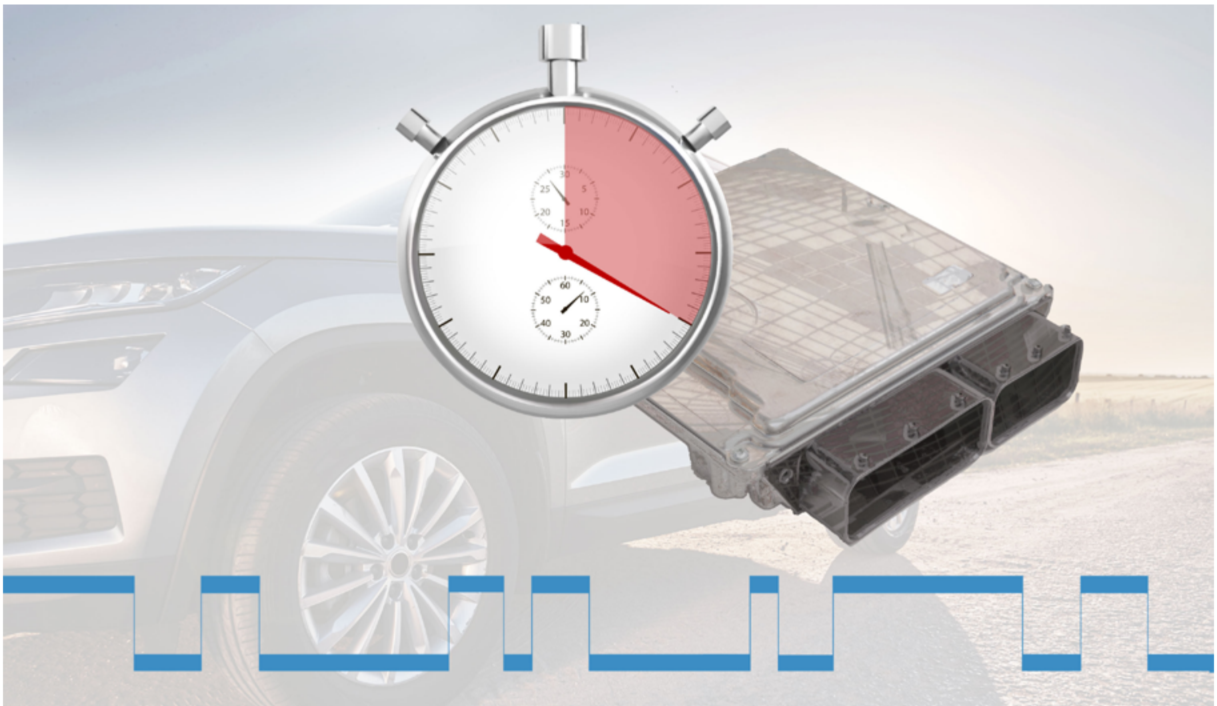
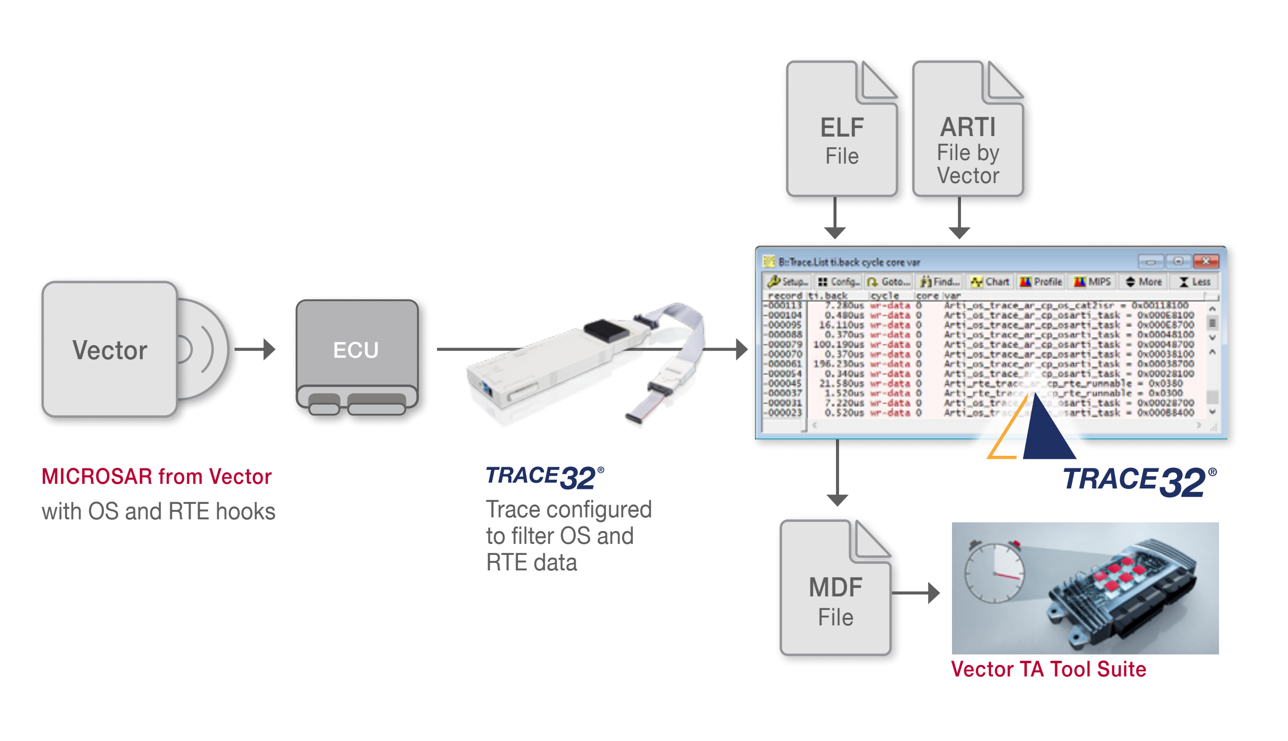


Figure 1. The perfect match for Automotive Timing Analysis: TRACE32® and Vector's TA Tool Suite



```
B::Trace.List ti.back cycle core var
record  ti.back    cycle    core var
-000113    7.280us wr-data  0    Arti_os_trace_ar_cp_os_cat2isr = 0x00118100
-000104    0.480us wr-data  0    Arti_os_trace_ar_cp_osarti_task = 0x000E8100
-000095   16.110us wr-data  0    Arti_os_trace_ar_cp_osarti_task = 0x000E8700
-000088    0.370us wr-data  0    Arti_os_trace_ar_cp_osarti_task = 0x00048100
-000079  100.190us wr-data  0    Arti_os_trace_ar_cp_osarti_task = 0x00048700
-000070    0.370us wr-data  0    Arti_os_trace_ar_cp_osarti_task = 0x00038100
-000061  196.230us wr-data  0    Arti_os_trace_ar_cp_osarti_task = 0x00038700
-000054    0.340us wr-data  0    Arti_os_trace_ar_cp_osarti_task = 0x00028100
-000045   21.580us wr-data  0    Arti_rte_trace_ar_cp_rte_runnable = 0x0380
-000037    1.520us wr-data  0    Arti_rte_trace_ar_cp_rte_runnable = 0x0300
-000031    7.220us wr-data  0    Arti_os_trace_ar_cp_osarti_task = 0x00028700
-000023    0.520us wr-data  0    Arti_os_trace_ar_cp_osarti_task = 0x00088400
```

MICROSAR from Vector
with OS and RTE hooks

TRACE32®
Trace configured to filter OS and RTE data

Vector TA Tool Suite

Figure 2. Workflow for AUTOSAR timing analysis using Lauterbach TRACE32® and Vector‘s TA Tool Suite

Advantages of AUTOSAR ARTI vs. ORTI

ORTI-based timing analysis focuses on the operating system and ignores other AUTOSAR modules such as Software Components (SWCs) and the Runtime Environment (RTE). Although the scheduling of AUTOSAR runnables is often the authoritative object of investigation in timing analysis, runnables are not covered by the ORTI file, nor is communication between software components. The developer is usually not able to distinguish between task completion, interruption, and waiting in the trace, since only task changes can be recorded, and thus cannot perform a detailed timing analysis on activation delays or total response time, for example.

AUTOSAR ARTI provides extended information, which is important for the automotive industry, about tasks, ISRs (interrupt service routines), runnables, RTE communication, or spinlocks in an ARXML format. Thus, ARTI support enables an in-depth analysis of the runtime behavior of AUTOSAR-based systems. ARTI defines a comprehensive, standardized interface between the build tools and the debug and trace tools. Because ARTI consistently records only this relevant information, it can easily be implemented on systems with a relatively small trace bandwidth.

Configuring Trace-Based Event Capture

Before the trace starts, the trace hooks to generate trace data for OS and RTE events have to be configured in MICROSAR. For Infineon AURIX™ for example the ARTI trace hooks write the details of the OS and RTE events to either the Online Data Acquisition (OLDA) address space or to an address space within the Local Memory Unit (LMU). Each 32-bit write contains the core identifier and the encoded event data.

In Lauterbach's TRACE32® PowerView Software microcontroller dependent configurations must also be made. For AURIX™ the MCDS trace and trigger logic has to be configured to generate trace data for OS and RTE events only. To facilitate this programming, Lauterbach provides a ready-made trigger program.

The trace data generated for ARTI profiling can be streamed to the host computer at program runtime, allowing very long recording times. It can be used to cover CPU load analysis, event chain analysis, calculation of OS metrics, and much more, in addition to validation of timing requirements.

Lauterbach's TRACE32® trace tools are an established part of the AUTOSAR Classic timing toolchain and also support ARTI profiling for the AUTOSAR Adaptive Platform. TRACE32® uses the ARTI file (arxml) generated during the build process to display all relevant debug information in a suitable way. As an example, figure 3 shows a runnable timing diagram for all six CPUs of an AURIX™ TC397XE microcontroller.

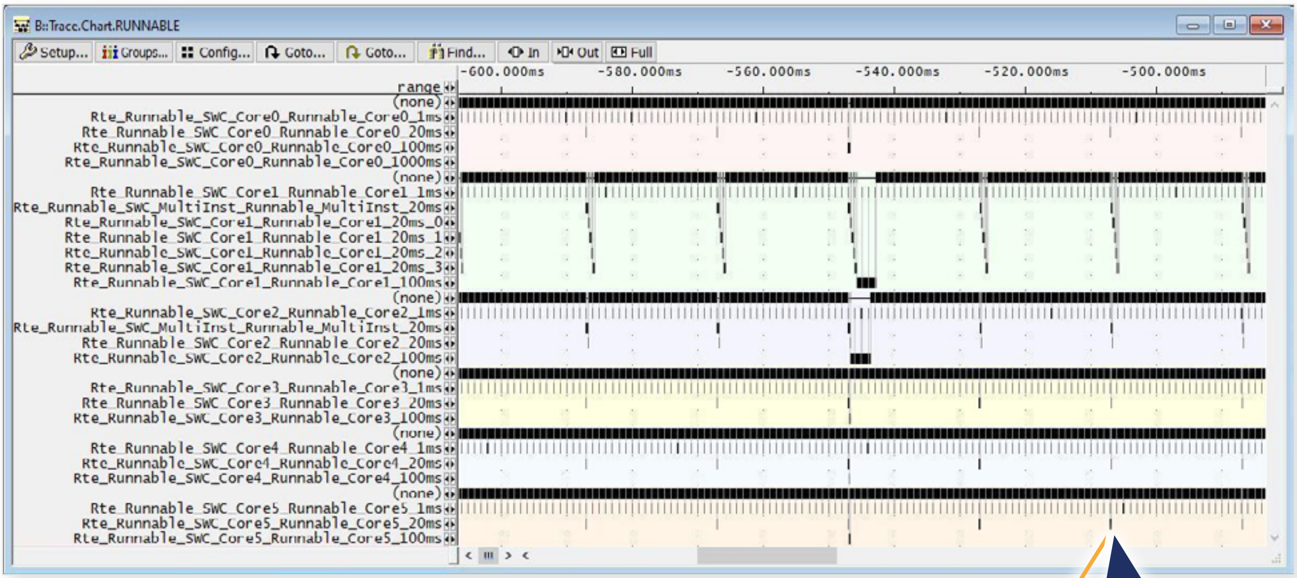

B::Trace.Chart.RUNNABLE
Setup...
Groups...
Config...
Goto...
Goto...
Find...
In
Out
Full
-600.000ms
-580.000ms
-560.000ms
-540.000ms
-520.000ms
-500.000ms
range
(none)
Rte_Runnable_SWC_Core0_Runnable_Core0_1ms
Rte_Runnable_SWC_Core0_Runnable_Core0_20ms
Rte_Runnable_SWC_Core0_Runnable_Core0_100ms
Rte_Runnable_SWC_Core0_Runnable_Core0_1000ms
(none)
Rte_Runnable_SWC_Core1_Runnable_Core1_1ms
Rte_Runnable_SWC_MultiInst_Runnable_MultiInst_20ms
Rte_Runnable_SWC_Core1_Runnable_Core1_20ms_0
Rte_Runnable_SWC_Core1_Runnable_Core1_20ms_1
Rte_Runnable_SWC_Core1_Runnable_Core1_20ms_2
Rte_Runnable_SWC_Core1_Runnable_Core1_20ms_3
Rte_Runnable_SWC_Core1_Runnable_Core1_100ms
(none)
Rte_Runnable_SWC_Core2_Runnable_Core2_1ms
Rte_Runnable_SWC_MultiInst_Runnable_MultiInst_20ms
Rte_Runnable_SWC_Core2_Runnable_Core2_20ms
Rte_Runnable_SWC_Core2_Runnable_Core2_100ms
(none)
Rte_Runnable_SWC_Core3_Runnable_Core3_1ms
Rte_Runnable_SWC_Core3_Runnable_Core3_20ms
Rte_Runnable_SWC_Core3_Runnable_Core3_100ms
(none)
Rte_Runnable_SWC_Core4_Runnable_Core4_1ms
Rte_Runnable_SWC_Core4_Runnable_Core4_20ms
Rte_Runnable_SWC_Core4_Runnable_Core4_100ms
(none)
Rte_Runnable_SWC_Core5_Runnable_Core5_1ms
Rte_Runnable_SWC_Core5_Runnable_Core5_20ms
Rte_Runnable_SWC_Core5_Runnable_Core5_100ms

Figure 3. Runnable time diagram for all six cores of an AURIX™ TC397XE

Advanced Timing Analysis with Vector TA.Inspection

After performing the ARTI Real Time Trace, the recorded trace data can then be exported as ASAM MDF (Measurement Data Format) and subsequently processed using the Vector timing tool TA.Inspection.

The Trace.EXPORT.MDF <file> command decodes the TRACE32® trace recording using the ELF and ARTI files and generates an MDF file for export to TA.Inspection, where further analysis can then be performed (Figure 4).

TA.Inspection allows the verification of the timing behavior of the application software and of the operating system. By integrating the TA.Inspection to a CI/CD/CT pipeline the automatic supervision of key timing metrics and constraints can be realized. Further information you can find in Vector's know-ledgebase: **kb.vector.com**

Conclusion

Vector and Lauterbach's joint solution makes it possible to provide accurate timing data for embedded AUTOSAR systems running automotive microcontrollers. The solution is based on the AUTOSAR software MICROSAR. The TA.Inspection option, part of Vector's TA Tool Suite, imports TRACE32® trace measurements for analysis in terms of response times, utilization, and other metrics.

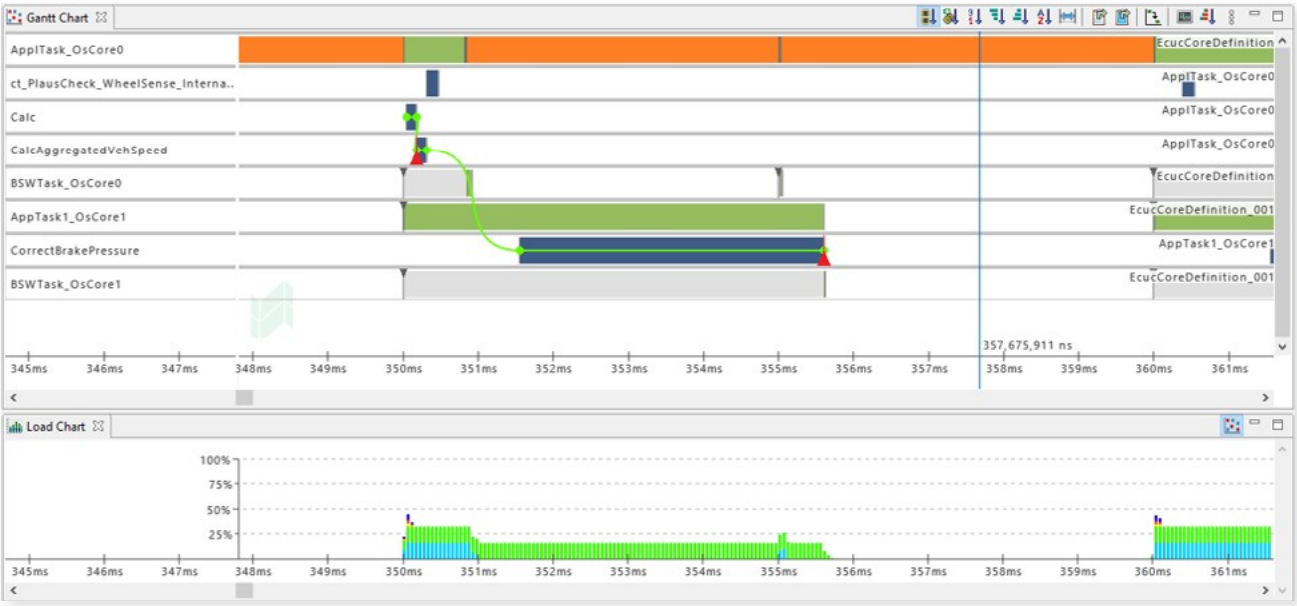


Figure 4. Timing analysis in Vector's TA Tool Suite

MORE INFORMATION

Lauterbach offers a support package „rte_profiling“ for this solution for Infineon AURIX™ on its website: **www.lauterbach.com/scripts**