



Standardized debugging in AUTOSAR

6° Automotive SPIN Italia Workshop

Milano - December 4th, 2009

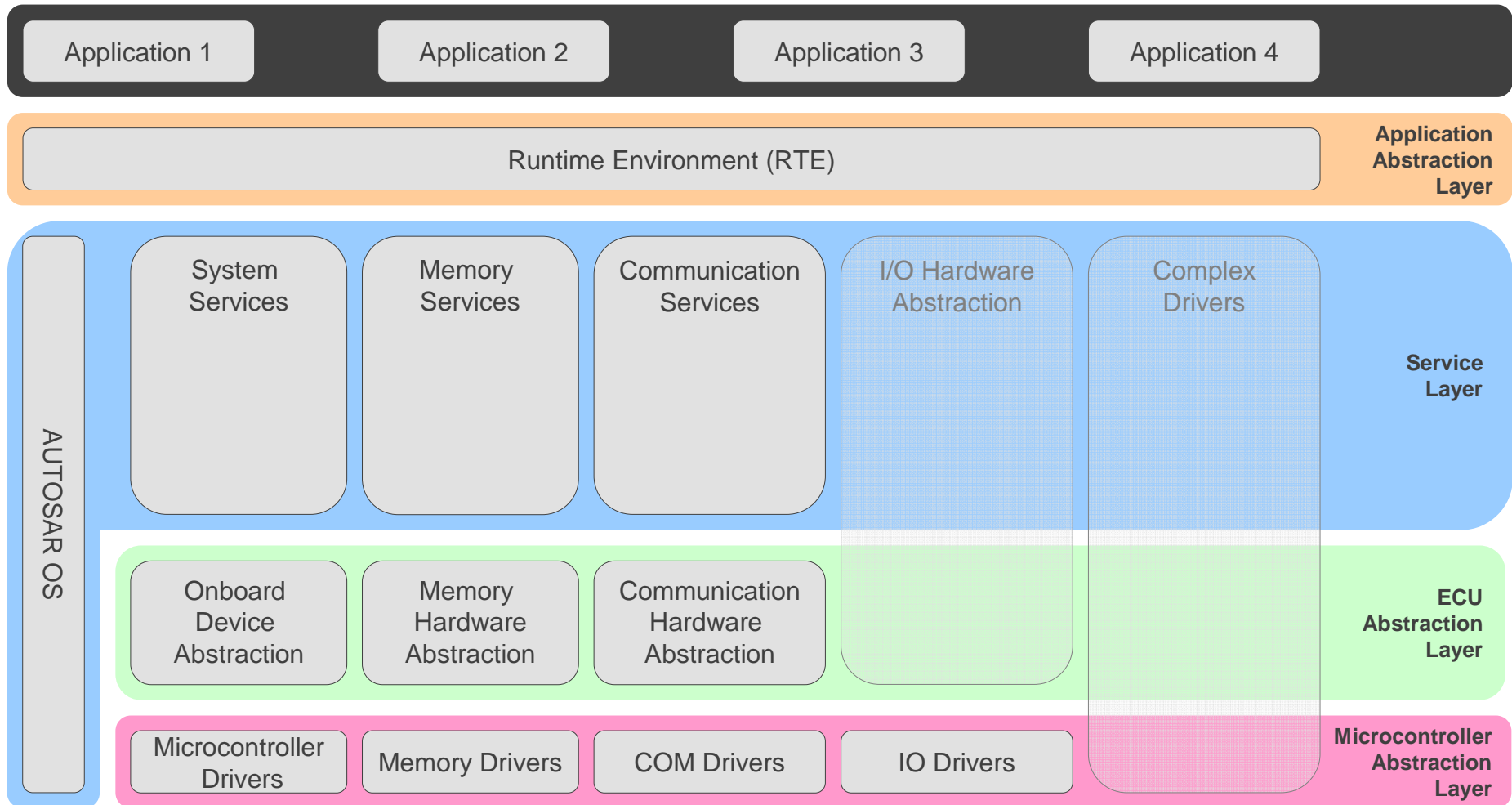
Jochen Olig

Overview

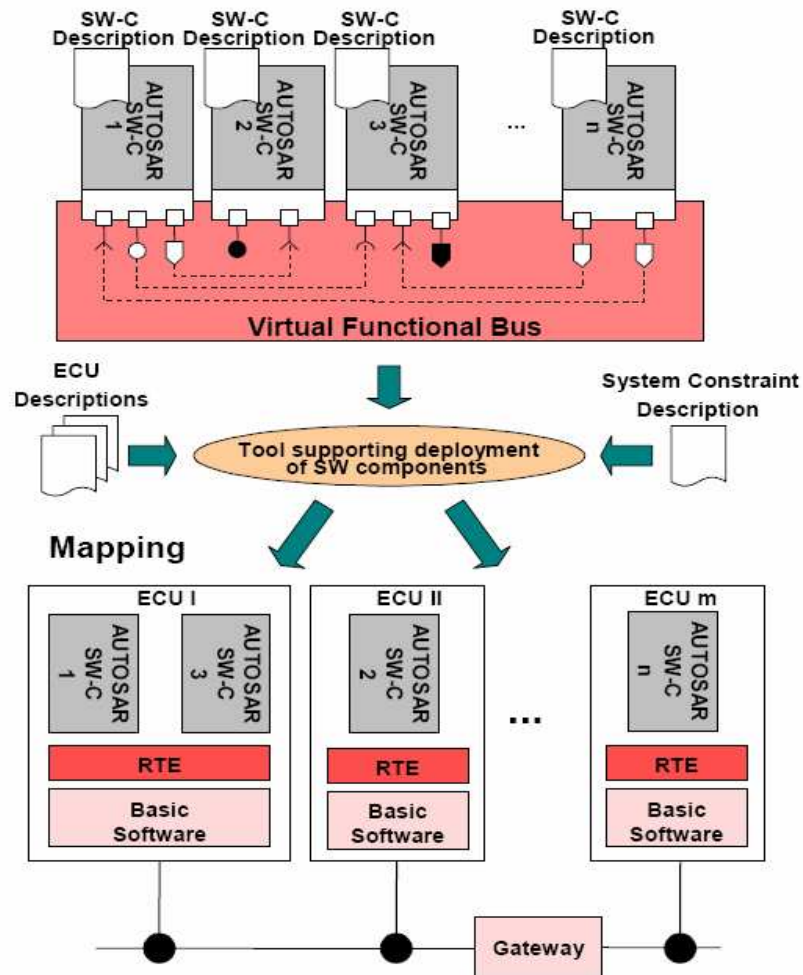
- Basic Concepts in AUTOSAR: Layered SW Architecture, VFB/RTE
- Challenges in AUTOSAR development
- AUTOSAR Debugging module
- Implementation of a debugging module
- Conclusion



The AUTOSAR Software Layers



Virtual function bus and SW Components



- Application Software is described in SW Components (SW-C)
- SW-C communication through
 - Sender/Receiver interfaces
 - Client/Server interfaces
- Specified at design time, without knowledge of HW and network topology

- Mapping of “Software Components“ to ECUs and configuration of basic software
- RTE then implements the VFB for the current ECU specific configuration

AUTOSAR – Challenges

- ~ 50 configurable AUTOSAR Basic SW modules
- Huge number of parameters to configure (> 6000)
- Complex configuration process
- Optimization for execution speed and memory footprint necessary
- Integration of various SW-Cs from different manufacturers
- Consistency of data and runtime behavior must be correct to assure the functionality of the whole system



AUTOSAR – Roles and Challenges

- Function developer
 - Get quick overview of the system status
 - Help to find problems without detailed AUTOSAR module know-how
- Stack integrator
 - Smart methodology for step-by-step integration needed
 - Wizards which guide the user during the configuration are needed
 - Debug and trace tools for seeking of problems needed



AUTOSAR Debugging – Key Features

- Basic idea: Debug and trace Basic SW modules during runtime using standardized functions and bus systems
- Available from AUTOSAR 4.0 onwards
- Collects raw data on the target
- Transmits collected data to a host system using standard AUTOSAR interfaces
- Let the host analyze/visualize the data



Data collection on the target

- Collection is based on so called DIDs (**D**ebugging **I**dentifiers). DIDs contain the data traced by the debugging module
- There are two types of DIDs with different behavior:
 - Standard DIDs: Can be used to trace data (plain memory dump or data structure dump) requested based on time, external event or internal Dbg API function call
 - Predefined DIDs: Can be used to trace function calls of
 - DET: Trace DET error codes
 - RTE: VFB tracing
 - OS: Tracing of Pre- and PostTaskHook
 - All BSW: Tracing of function entry and exit
- Data tracing
 - Cyclically: Standard DIDs only
 - Explicitly: Standard and Predefined DIDs
- Tracing data can be stored in a buffer to handle bursts

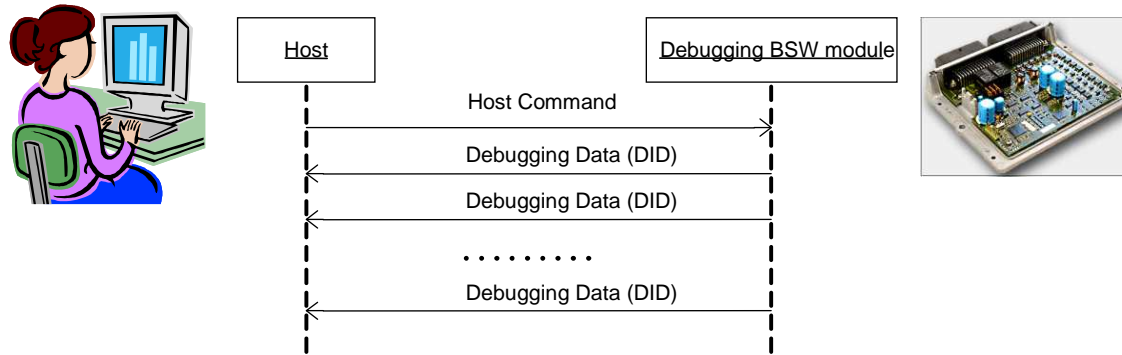


Transmission of data to a host system

- The message format between target and host system is standardized → Hosts and target systems of different vendors can be combined
- The format is optimized to be usable on CAN (bandwidth, maximum message size)
 - Other interfaces like FlexRay or Ethernet will yield a much better tracing performance
- A Transport Layer is part of the Debugging module which allows transmission of large DIDs
- Also support proprietary interfaces by partitioning the debugging module into a data collection and a transmission part
- Sending of data
 - Continuously: Whenever data was traced it is sent immediately
 - On request: Explicit request to send by host

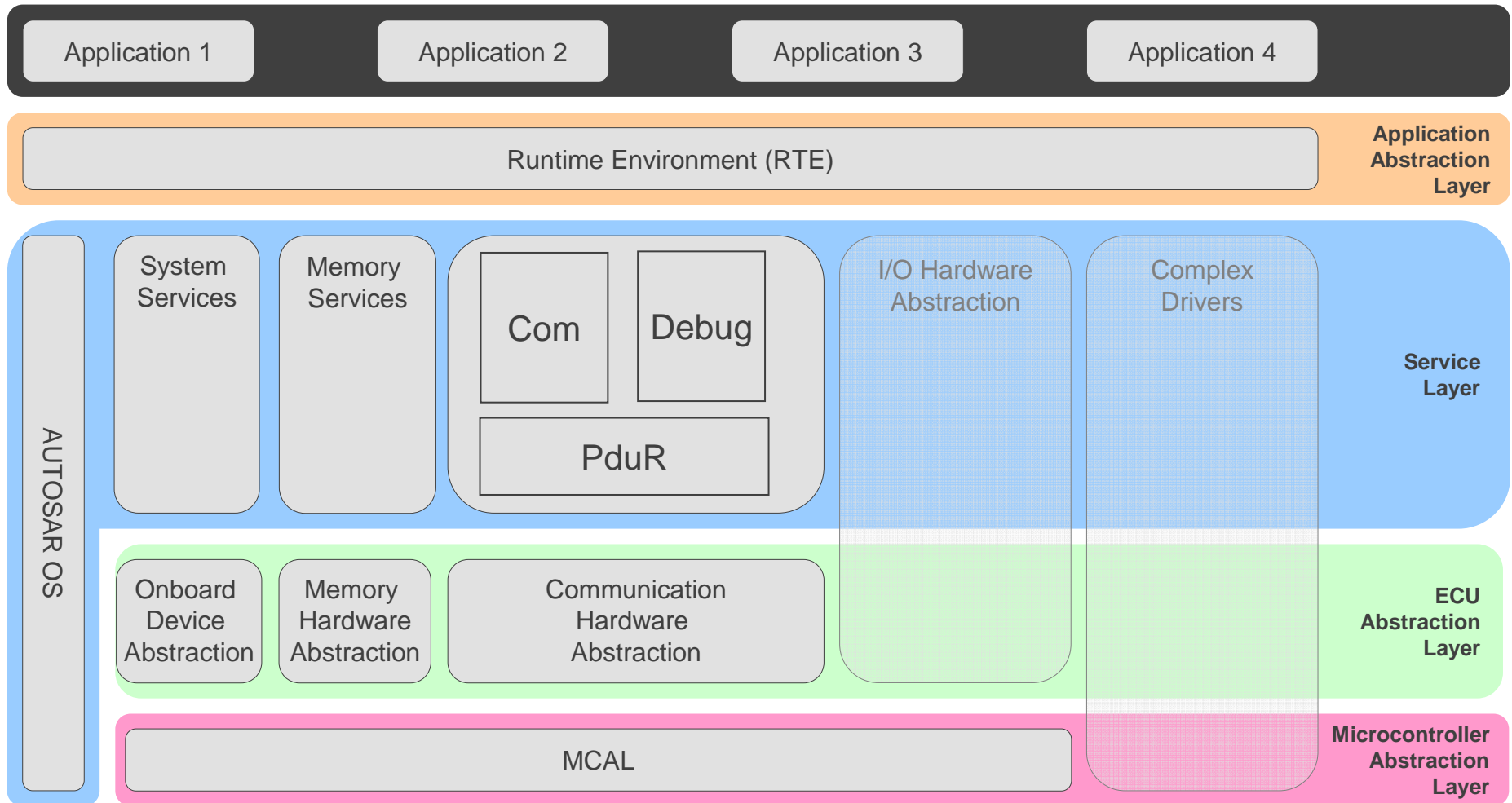


AUTOSAR Debugging



- The debugging host sends commands to the target system (via control channel)
- Among others, the following properties can be changed during runtime
 - Switching ON/OFF of tracing of individual DIDs
 - Buffering ON/OFF: Direct sending or buffering of DIDs
 - Continuous sending ON/OFF: Buffer content is sent when new data was stored

Embedding AUTOSAR Debugging

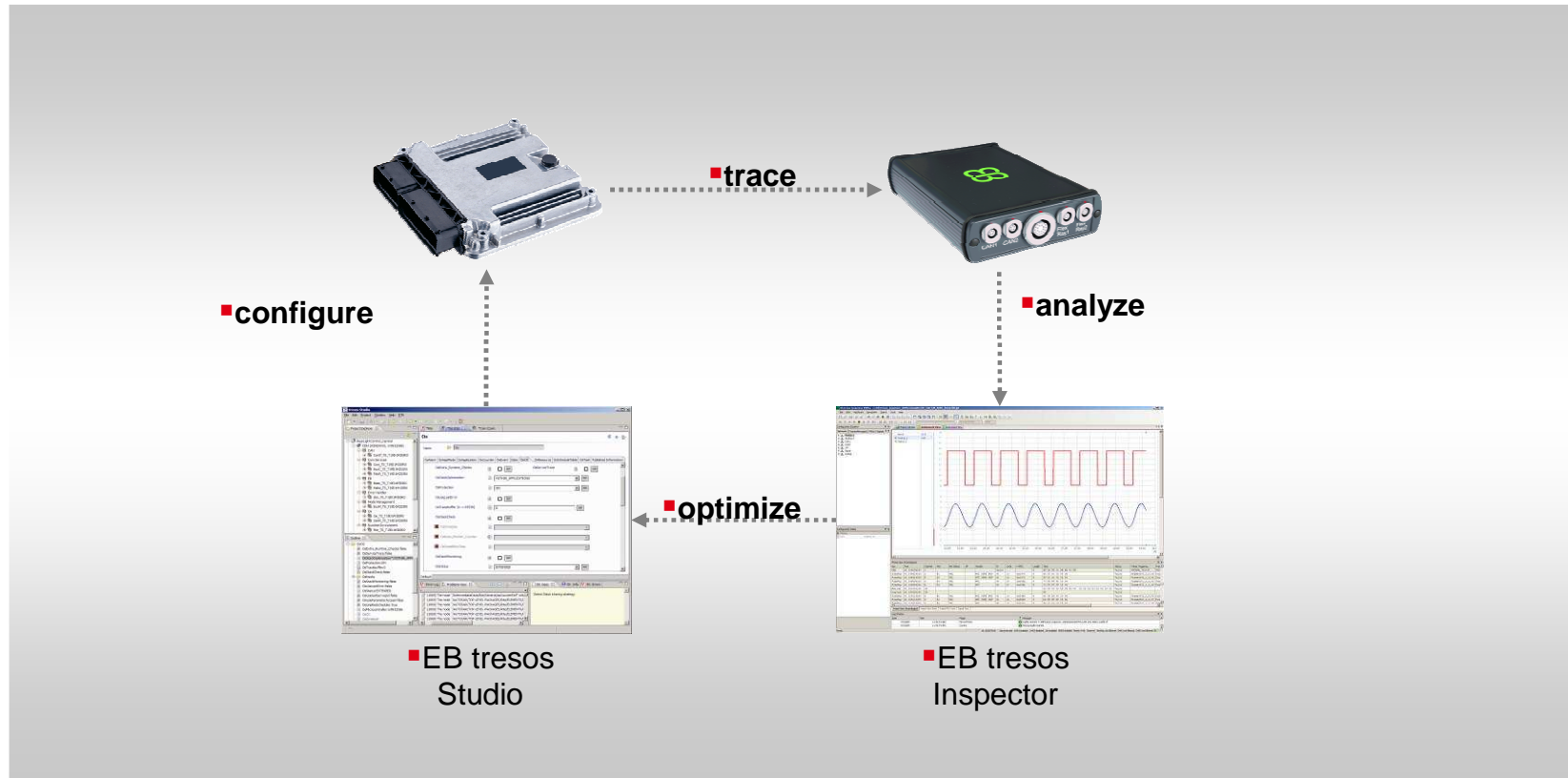


EB tresos *Debug & Trace* - Features

- Transmission of *Debug & Trace* data via CAN or FlexRay
- Buffering of data in ECU memory
- Switching traced data on/off during run-time
- RTE tracing
 - Sender/Receiver signals
 - Client/Server calls
 - Enter/Exit of Runnable entities
- DET tracing
- OS Pre-/Post-Task-Hooks
- Generic function tracing (Extension to AUTOSAR)
 - Enter/Exit of function calls
 - Parameter values
- Generic state machine tracing (Extension to AUTOSAR)
 - Old and new state of configured state variables



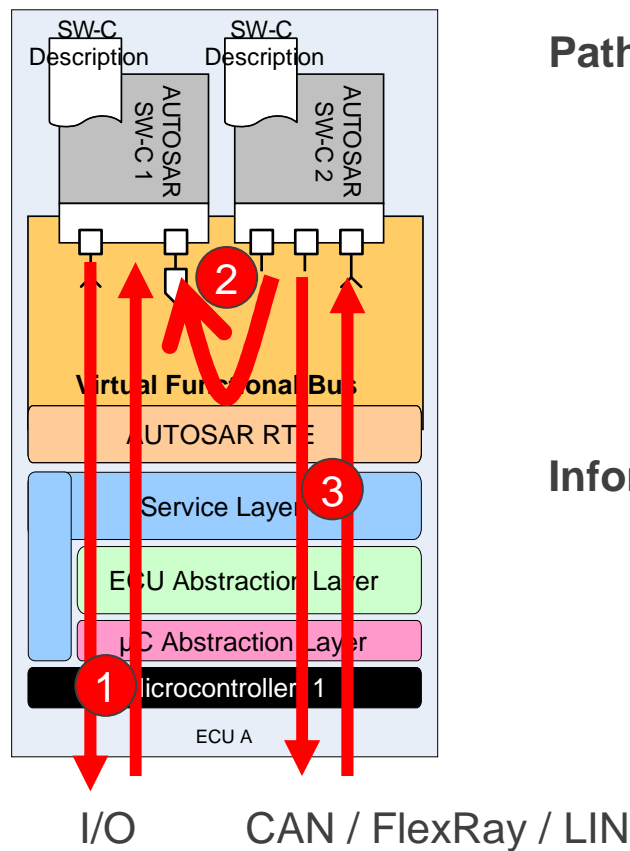
EB tresos *Debug & Trace*



Copyright Elektrobit Corporation 2009
Elektrobit.com, March 10, 2009, Slide 13



Functional Overview - Rte Tracing



Paths of TX/RX Signals to be traced

- SW-C to ECU I/O 1
- SW-C to SW-C of same ECU 2
- SW-C to Network (CAN, FlexRay, LIN) 3

Information traced

- Signal Id
- Time stamp of Rte-Call Rte_Send_Signal / Rte_Receive_Signal
- Value transmitted (Basic data types, no arrays)

Analysis with EB tresos Inspector

The screenshot displays the EB tresos Inspector interface, which is divided into several main sections:

- Configuration Explorer:** A tree view on the left showing the project structure for 'can-demo', including components like DBG_RX_PDU, DBG_TX_PDU, GenericFunction, GenericStatemachine, PostTaskHook, PreTaskHook, RunnableTermination, RunnableStart, RTECall, RTEVfbSignalReceive, RTEVfbSignalSend, and DetCall.
- Instrument View:** A graph showing two data series: 'Os_Task_TASK1_NewValue' (blue line) and 'Os_Task_TASK5_NewValue' (yellow line). The x-axis represents time in seconds, ranging from approximately 50.54 to 50.62. The y-axis represents values from 4 to 5. The graph shows a periodic, sawtooth-like pattern for both signals.
- Debug and Trace Control View:** A table for configuring debugging options for the 'can-demo' ECU. It includes columns for Name, DID Collection, Timestamp, Buffering, Buffer, and Continuous Send. The 'GenericFunction' and 'GenericStatemachine' entries have their 'DID Collection' and 'Timestamp' options checked.
- Debug and Trace View chronological:** A table showing a chronological list of events. It includes columns for ECU Time, Bus Time, Signal Name, Name, Value, and Text. The events include 'GenericFunction' calls and 'Os_Api_ActivateTask' operations, with associated signal values and text descriptions like 'Entry', 'Exit', 'SUSPENDED', and 'NEW'.
- Configuration Editor:** A panel at the bottom left showing the configuration for the selected object, including its name ('GenericStatemac') and identifier (238).

Copyright Elektrobit Corporation 2009
 Elektrobit.com, March 10, 2009, Slide 15



Debug & Trace - Outlook

- OS task tracing
 - Show task statistics (min. / max. / avr. runtime)
 - Show task activation and preemptions
 - Show ISR
- Show online status of AUTOSAR Modules
- Tracing of multiple ECUs at the same time
- PDU tracing
- Back annotation to tresos Studio



Conclusion

- Integration of application software components and assuring their interoperability is crucial
- Complex configuration of basic software in AUTOSAR based development
- AUTOSAR debugging
 - is seamlessly integrated in AUTOSAR SW architecture
 - uses existing communication channels
 - enables in deep analysis of ECU Software during runtime
 - supports both function developers and stack integrators
- Implementation as *EB Debug & Trace* is available!



Thank you!



Discover the Experience