Software Debugging over XCP: Effectively Debugging ECUs in the Field

ASAM Technical Seminar 2018

V1.0 | 2018-06-06

Agenda

- **Motivation**
- **Key Features of the Standard**
- The Standard from a Debugger Supplier’s Perspective
- Live Demo

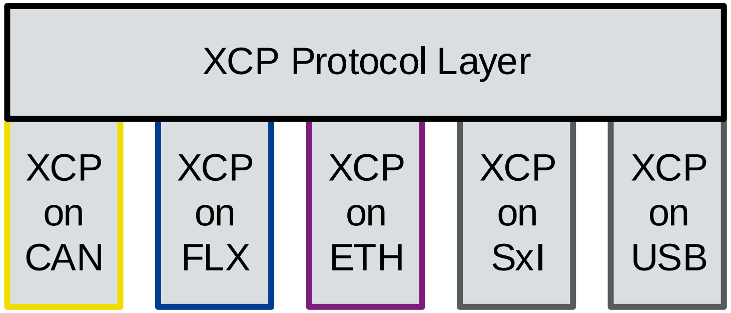
The Roots of XCP

As successor of the CAN Calibration Protocol (CCP) the Universal Measurement and Calibration Protocol (XCP) is primarily used for

- Measurement: acquisition of values of internal variables of an ECU
- Calibration: adjustment of internal variables

XCP is designed as a two layer protocol

- Unique protocol layer
- Transport layer: support for different transport media/busses



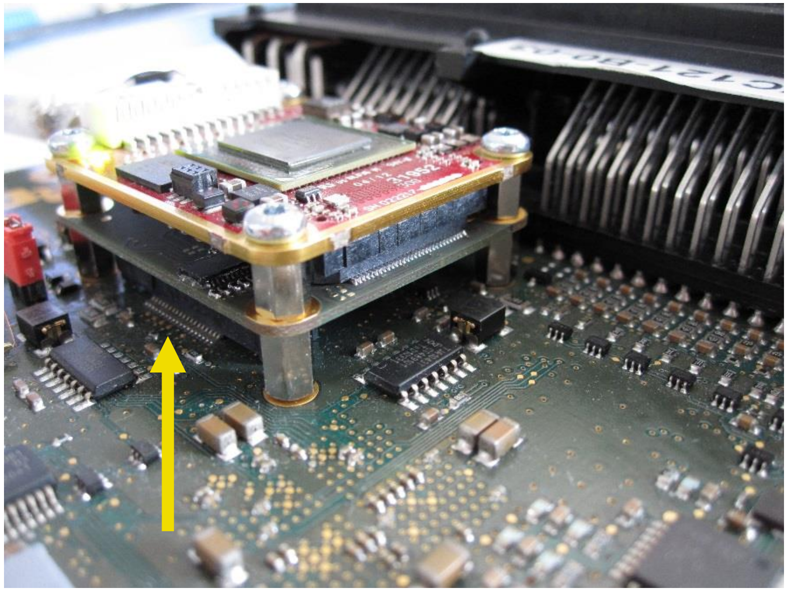
Motivation

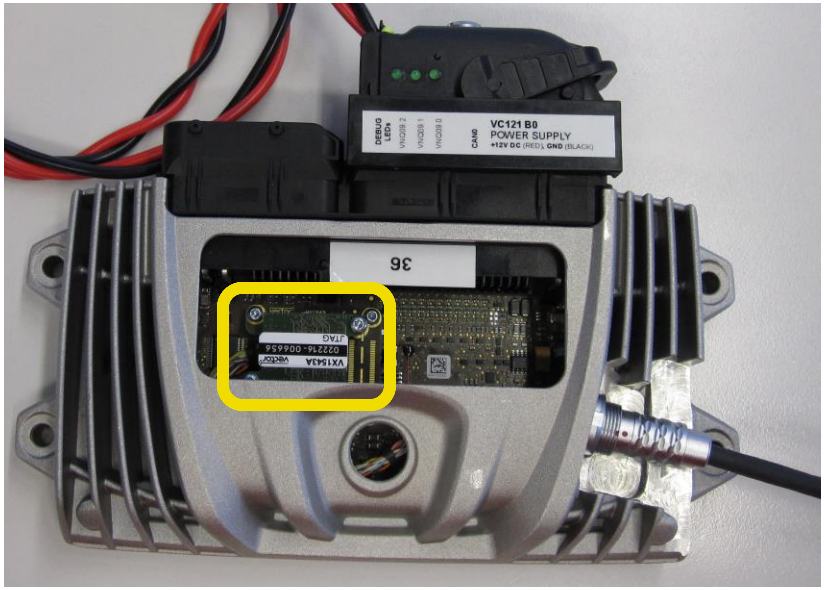
MC and DBG typically rely on the same target debug interface for ECU access

- Switching between MC-HW and Debug Probe is cumbersome
- Mechanical setup might even prevent Debug Probe access to ECU

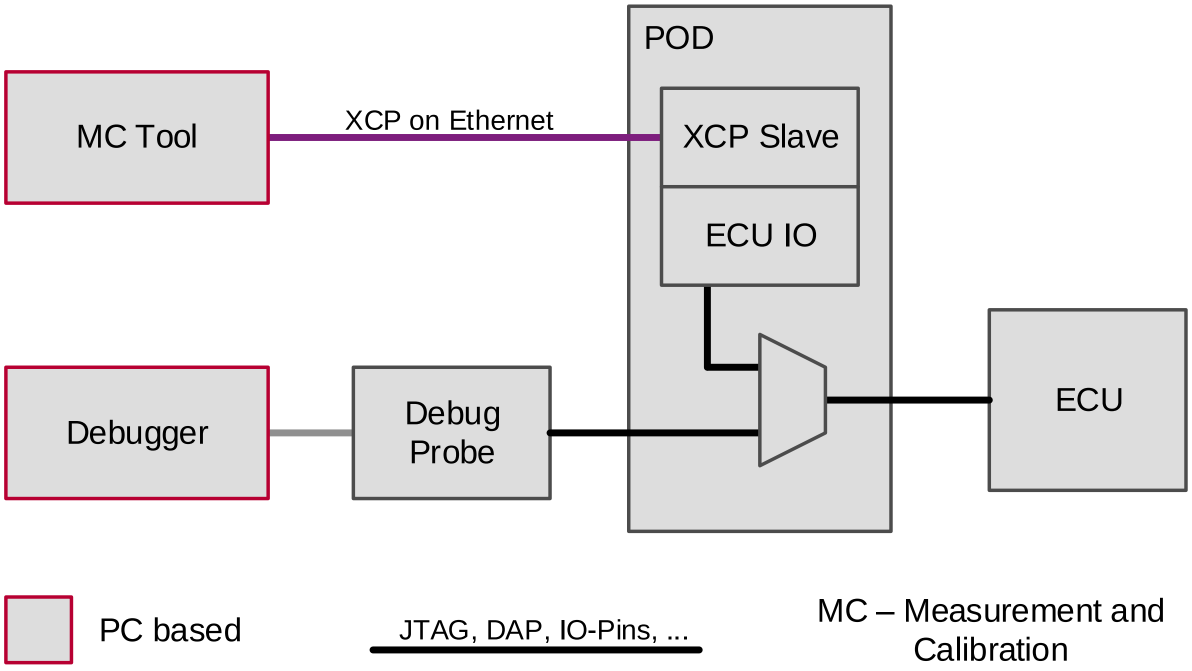
Data acquisition and calibration (MC) and software debugging (DBG) are essential techniques used during all stages of ECU development

- Techniques have typically been used apart in the past
- Demand of concurrent use in future





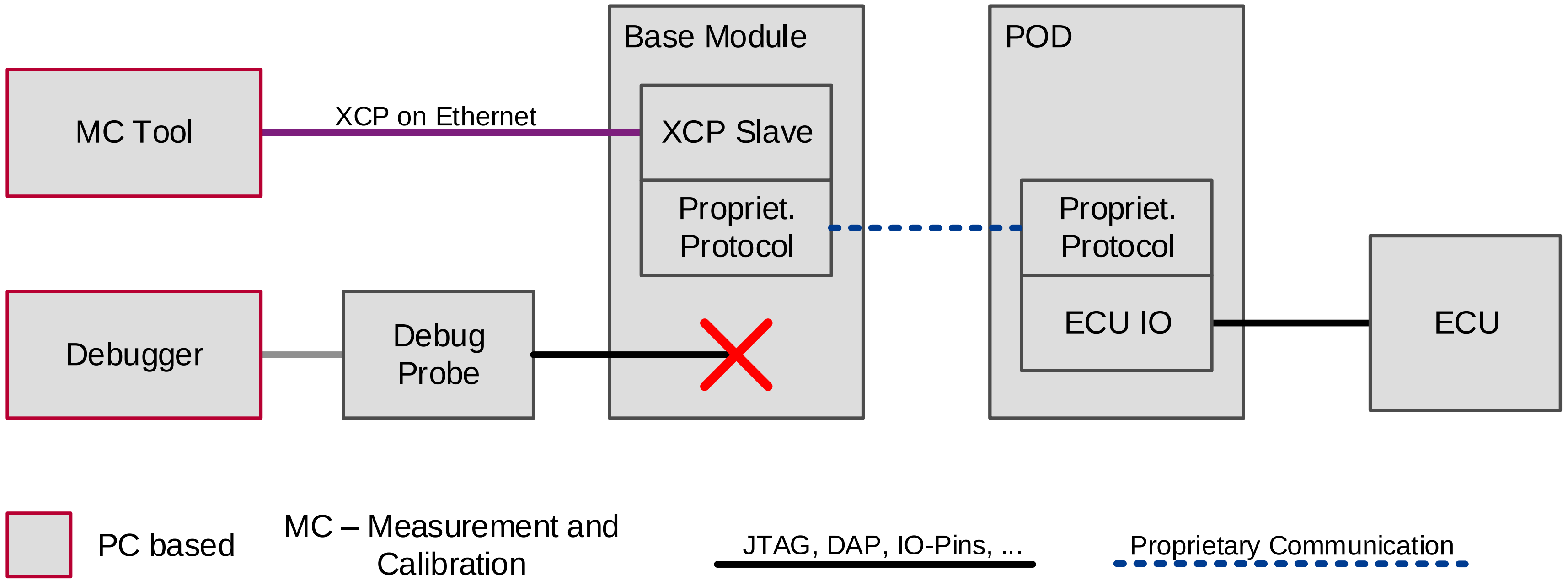
Switching of ECU Debug Signals



Limitations

- Hardware-based arbitration mechanisms lack semantical information of the arbitration request
  - Limits interoperability, system performance and usability
- POD encapsulated within ECU housing
  - Debug Probe unable to access ECU

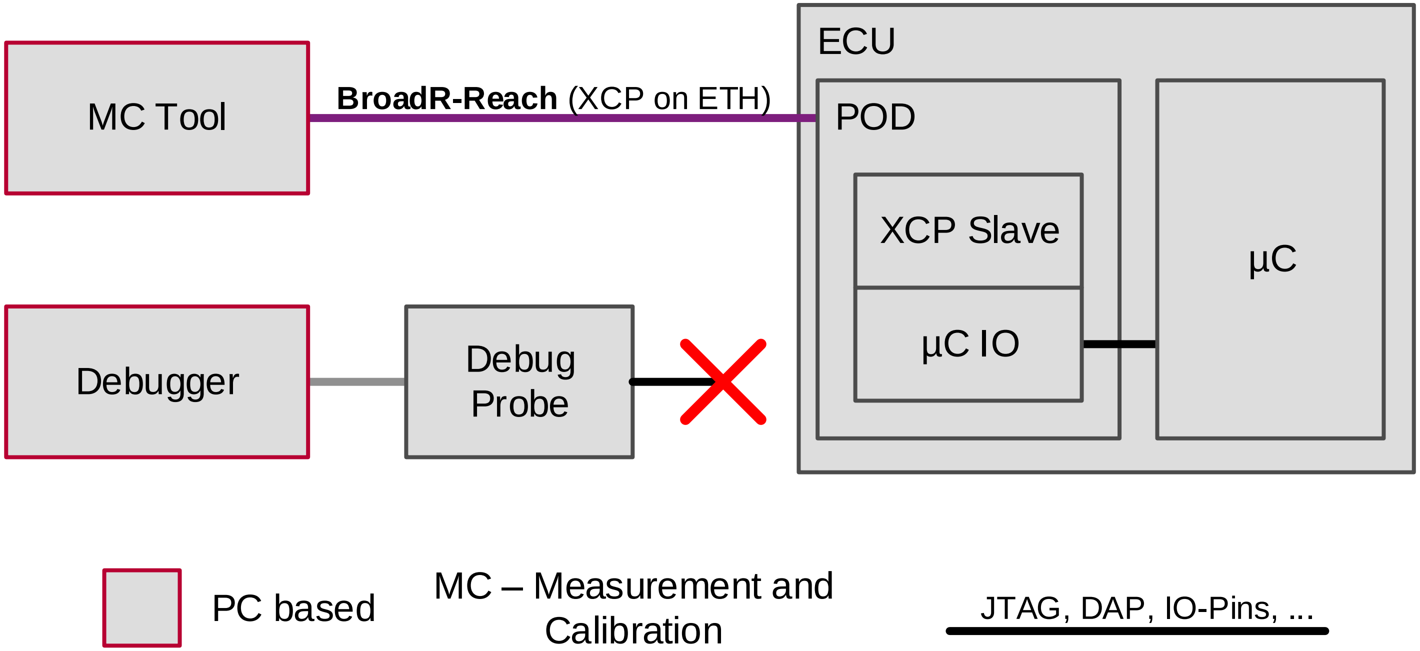
Partitioned MC System



Limitations

- Proprietary protocol used for communication between Base Module and POD prevents relay of Debug Probe signals

BroadR-Reach Tool Access



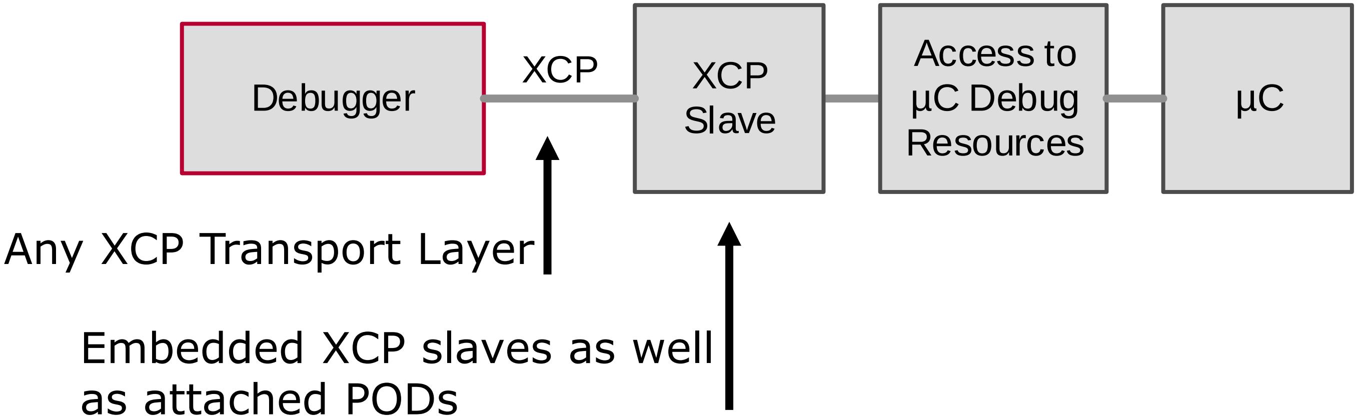
Limitations

- No interface option to connect Debug Probe to POD inside ECU

Debugging over XCP Standard

Standardization of manufacturer-independent mechanisms addressing todays and future needs of ECU debugging

- Standard shall enable the interoperability of different debuggers with different PODs and different MC Tools
- Extension of the widely used Universal Measurement and Calibration Protocol
  - By means of the ASAM Standard Debugging over XCP, associated to XCP
- Definition of generic mechanisms
  - Shall be applicable to embedded XCP Slaves and PODs



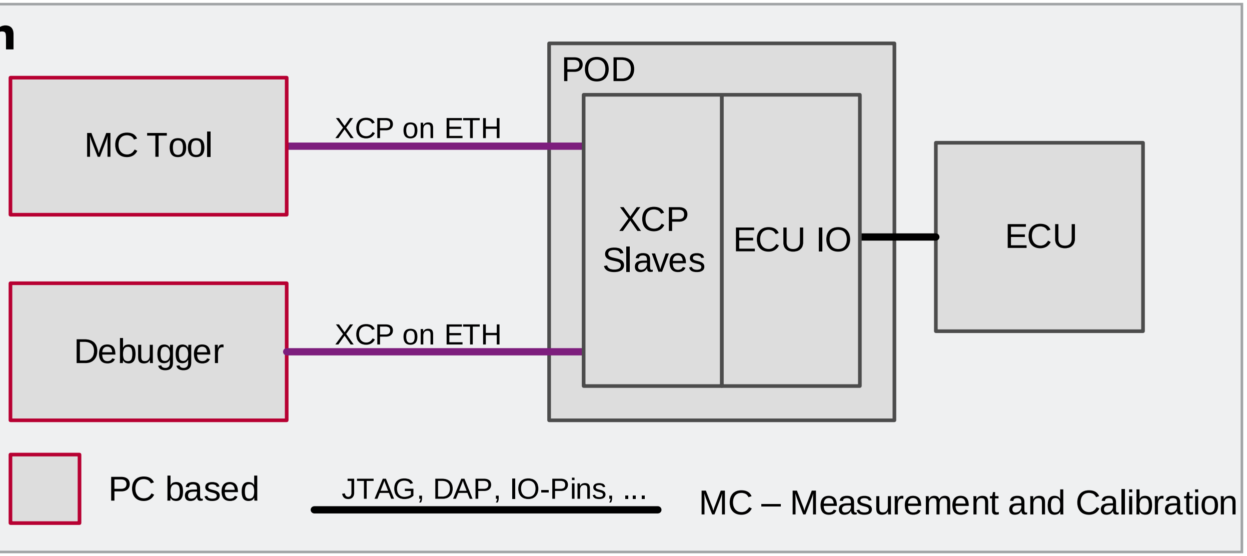
Any XCP Transport Layer

Embedded XCP slaves as well as attached PODs

Range of Covered Technologies

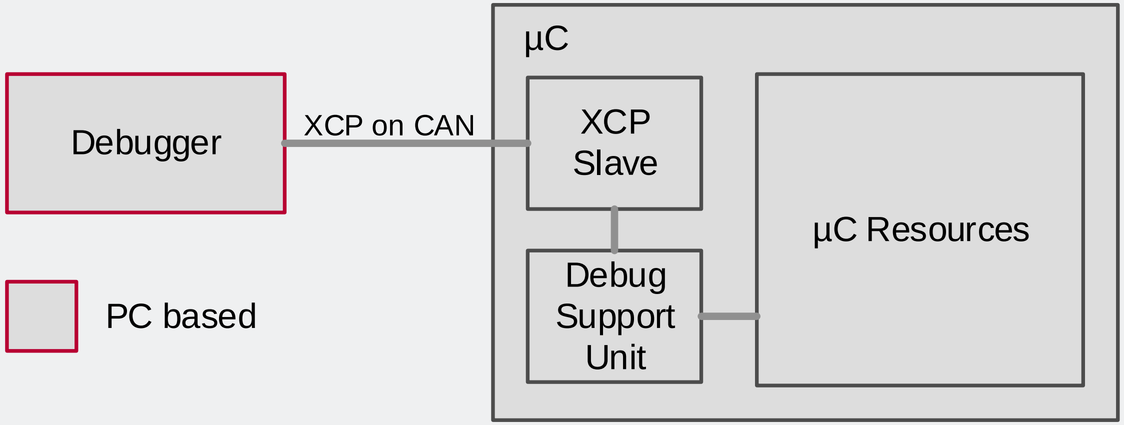
POD based solution

largest range of debug functionality



Embedded XCP Slave

reduced range of debug functionality



In depth comparison is given in chapter 3.5 of the standard

Essential Features Enabling ECU Debugging

- High level commands: reading and writing of arbitrary memory locations
  - Most efficient method for interaction of debugger and target
  - POD translates high level command in possibly several low level target accesses
  - Similar to classical XCP memory access mechanisms but without address translation
- Low level target access
  - Method for sending JTAG and DAP commands
  - Fallback solution if
    - resources are not memory mapped
    - POD is not aware of accessing arbitrary memory locations
  - For more complex, atomic accesses exclusive bus access can be requested
- I/O control
  - Enables debugger to control target reset, watchdog disable and other functions a POD might not be aware of

Methods Improving Parallel Use of MC and Debugging

Definition of Service Levels

- The XCP slave determines the service level
- The service level might change during run time
  - An event is sent to the debugger upon a service level change
- The debugger
  - shall adapt the feature set offered to the user according to the service level
  - shall adapt the XCP command sequence, e.g. shorten time span of exclusive target access
- 4 service level are defined
  - Service level 1 – debugging not possible
  - Service level 2 – exclusive debugger access to target
  - Service level 3 – high bandwidth assigned to debugger
  - Service level 4 – low bandwidth assigned to debugger

Methods Improving Parallel Use of MC and Debugging – cont’d

Semantical awareness of debugger activities

- Debugger uses XCP commands rather than a primitive hardware arbitration mechanism
- POD can optimize scheduling of XCP commands from different XCP masters (MC, debugger) to improve system performance
  - When needed, the debugger can request exclusive target access

For more information about Vector
and our products please visit

www.vector.com

Author:
König, Ralf
Vector Germany

ASAM Technical Workshop 2018

Software Debugging over XCP: Effectively Debugging ECUs in the Field

Michael Eick

2018 / 06 / 14

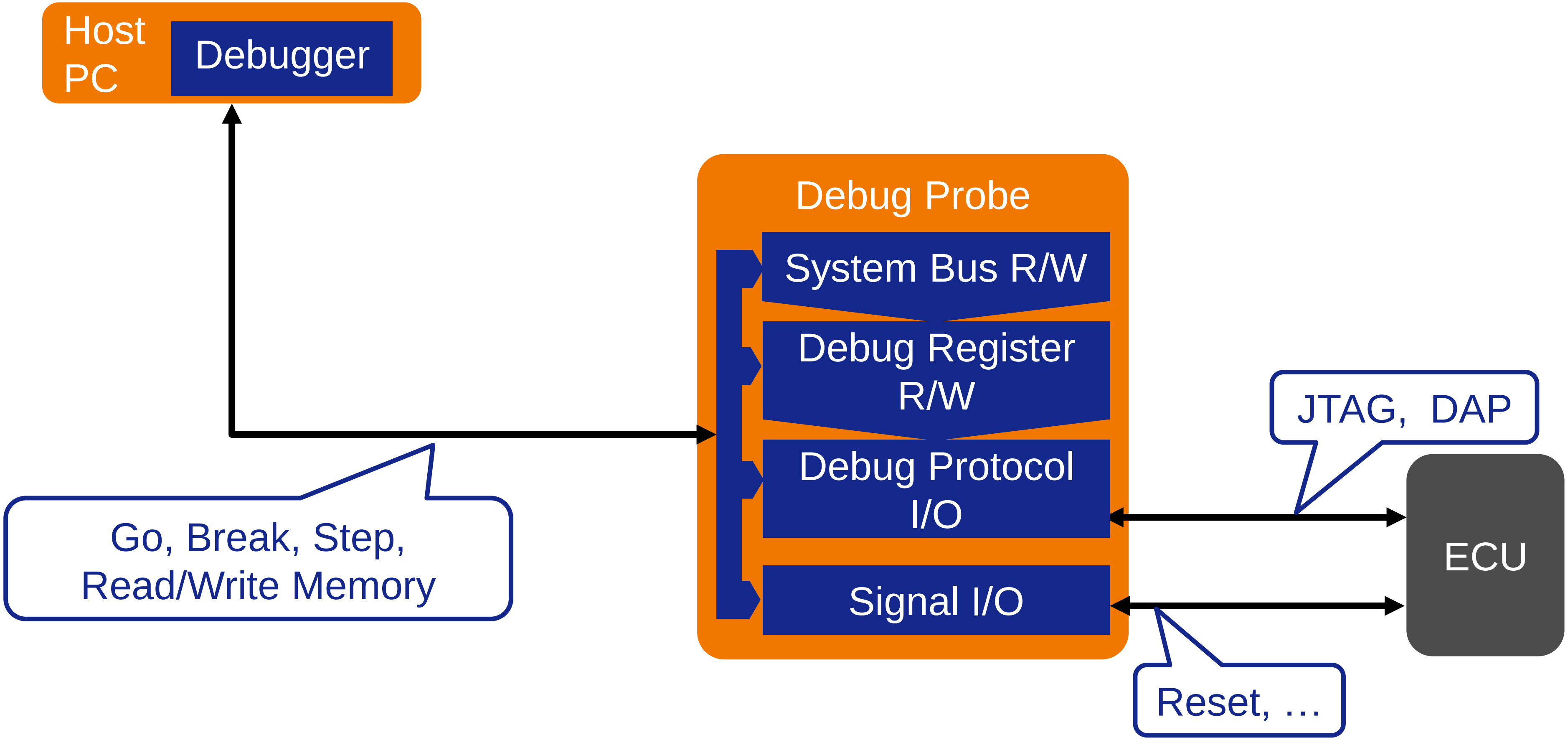
www.lauterbach.com

LAUTERBACH
DEVELOPMENT TOOLS

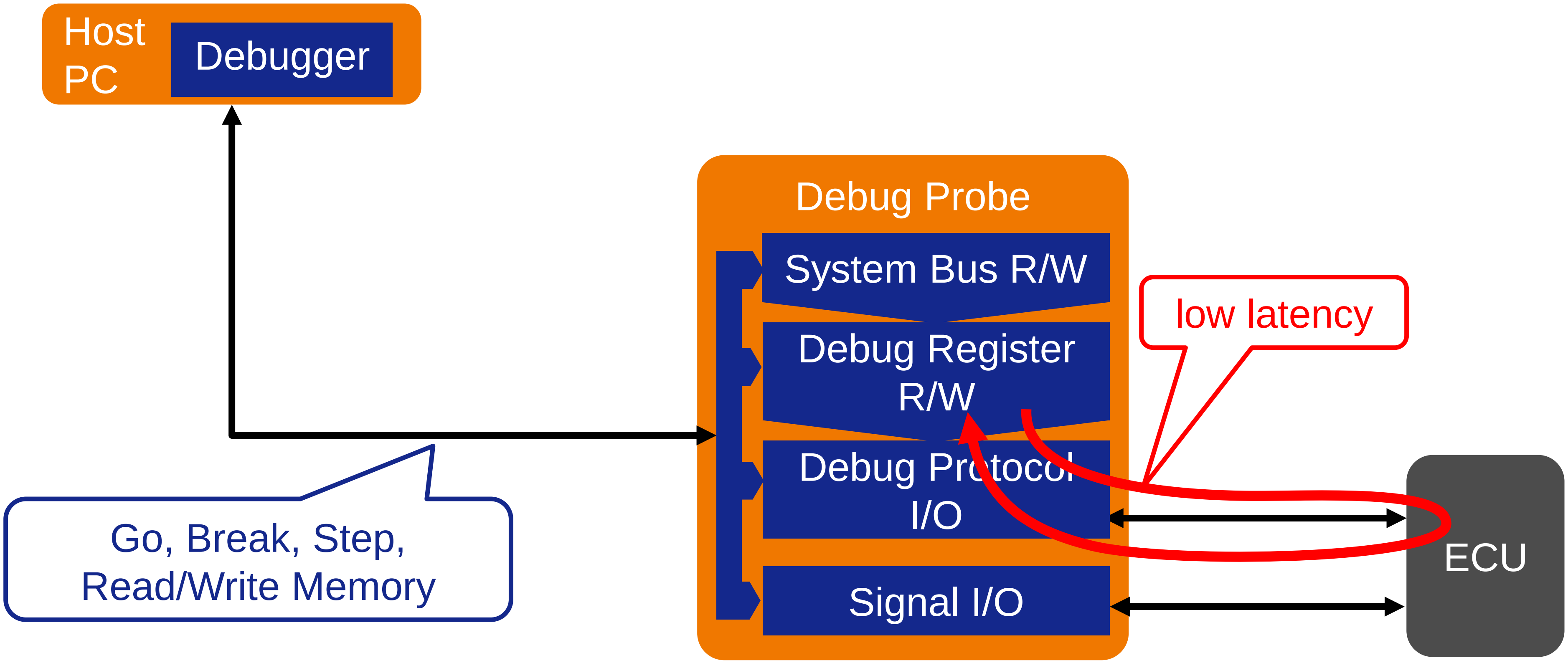
Agenda

- Motivation
- Key Features of the Standard
- The Standard from a Debugger Supplier’s Perspective
- Live Demo

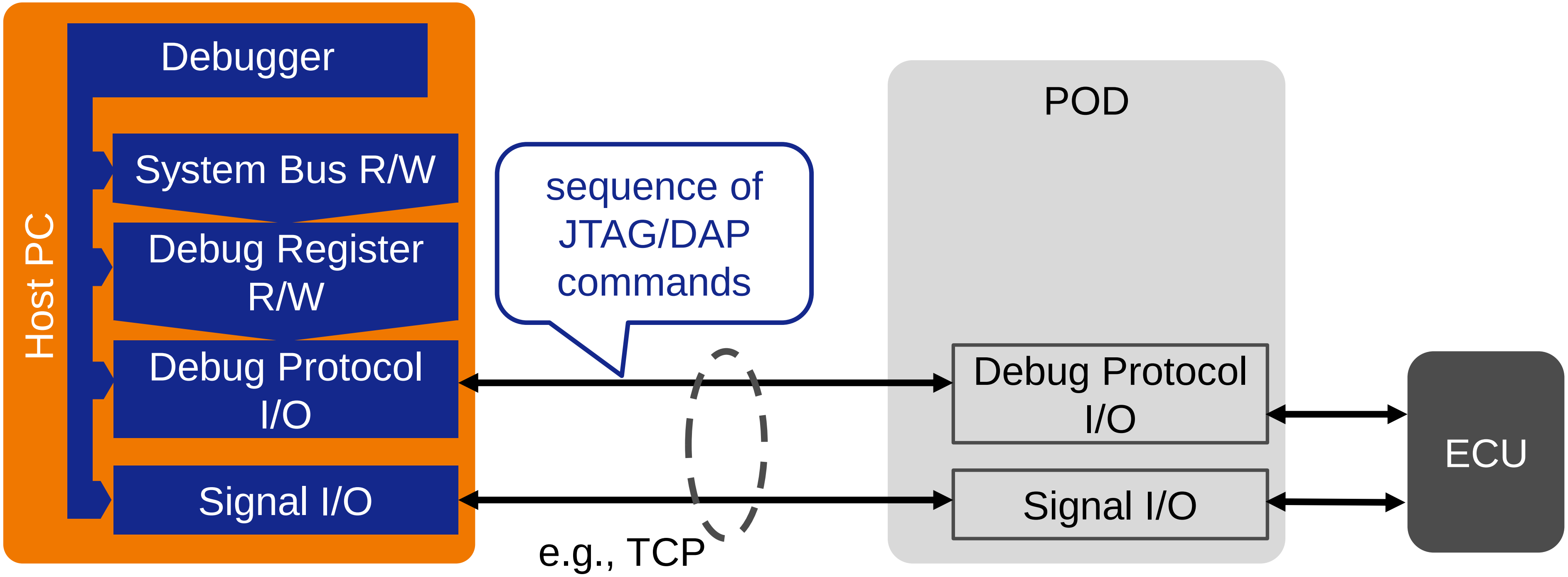
Debug System Overview



Debug System Overview

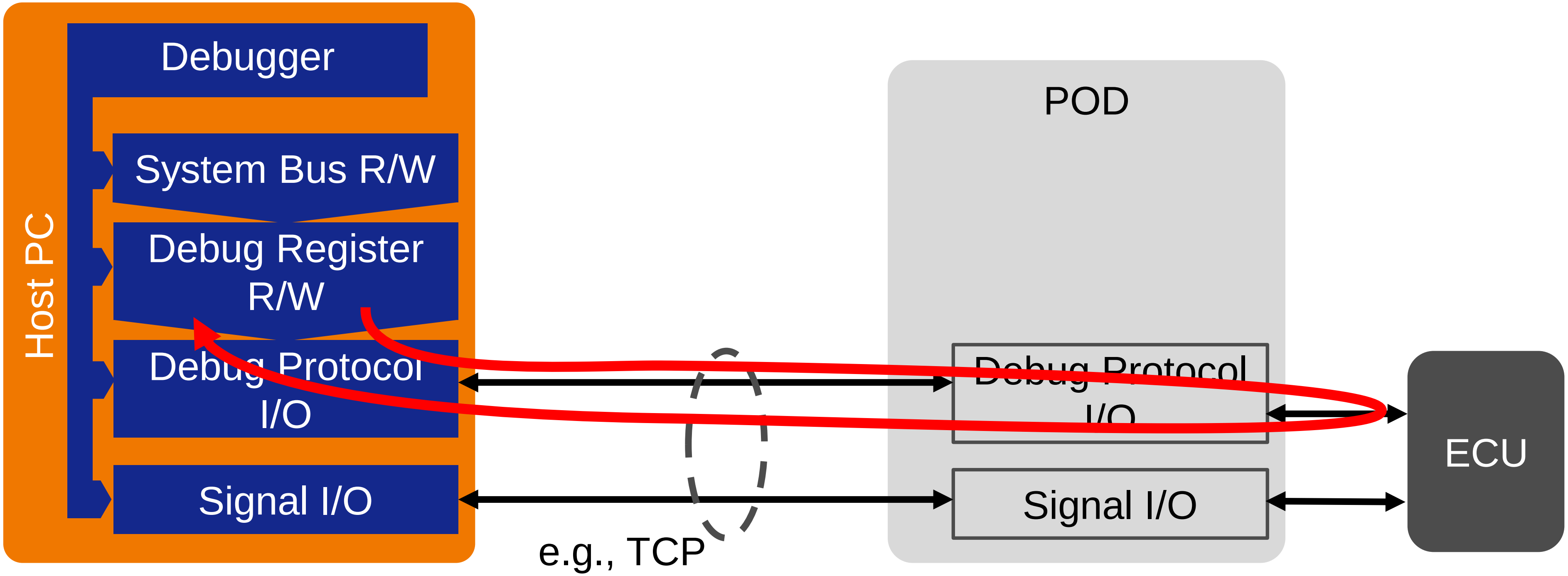


Low-Level Tunneling Approach



- Supports all operations required for debugging

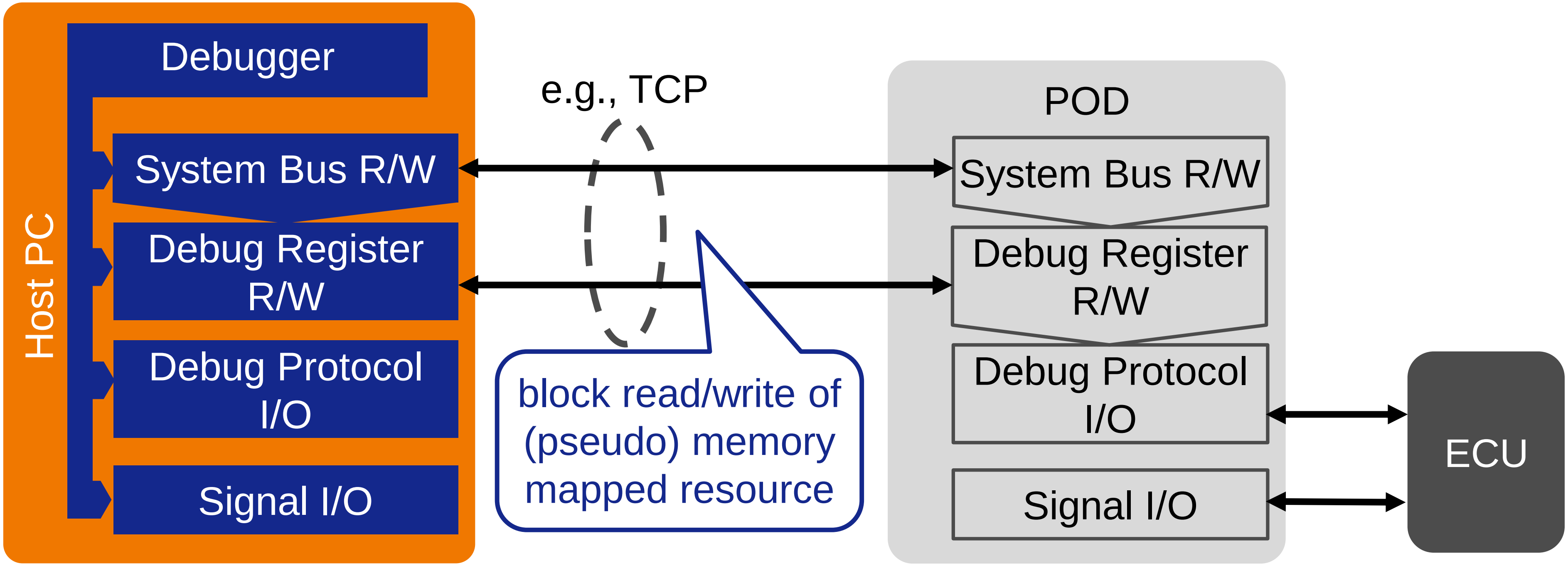
Low-Level Tunneling Approach



- Supports all operations required for debugging
- Example bus read:
  - several debug register operations
  - TCP latency adds to every operation
- Primarily relevant for POD use case

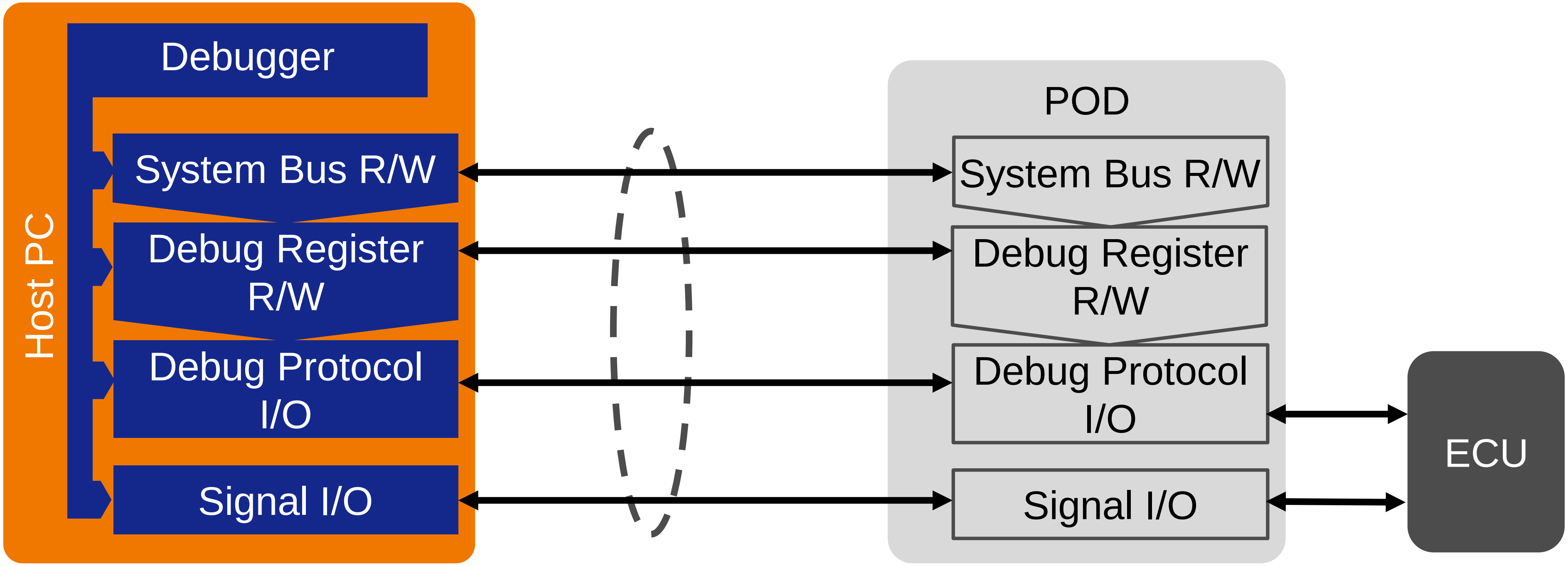
slow

High-Level Tunneling Approach



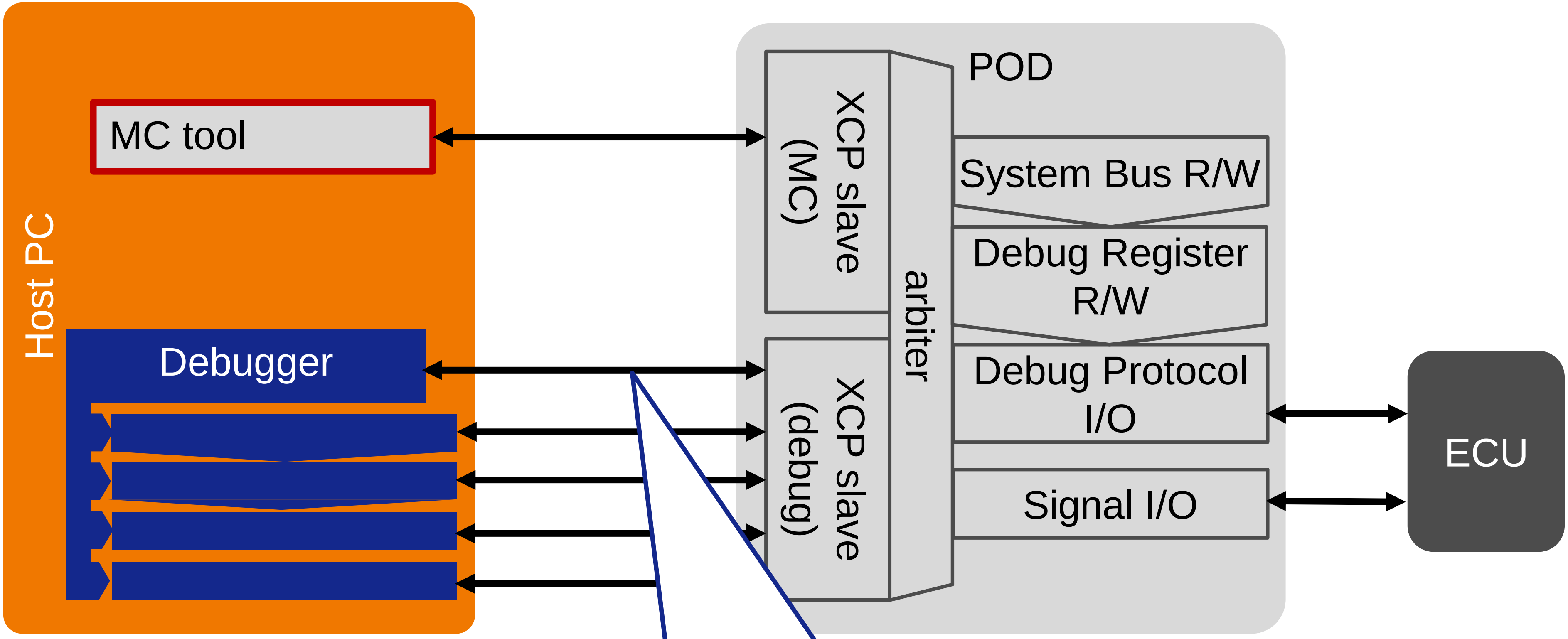
- System bus read/write only one operation → good performance
- Can be implemented for embedded XCP slaves
- No actions possible requiring operations on debug protocol or signal level → impact depends on ECU CPU type

Command Space of Standardized Protocol



- Low-level commands are optional
  - Debugger determines XCP slave (POD) capabilities via protocol
  - restrictions may result if not available
- Debugger functions can be implemented using optimal combination of high and low level commands

Parallel Usage of MC Tool and Debugger



Supported by standard through exclusive access request service level events

Supported Debug Functions

- Start and stop program execution, single stepping
- Program and read/write breakpoints
- Debugging from the reset vector
- Run-time access to arbitrary memory locations, high-level (C/C++/…) variables, peripherals
- Flash programming
- On-chip trace (if not in use by MC tool)

Limits

- No off-chip trace
- Performance compared to system with debug probe
  - Operations can take longer
    (e.g., time required for single step, start, stop)
  - Higher reaction time

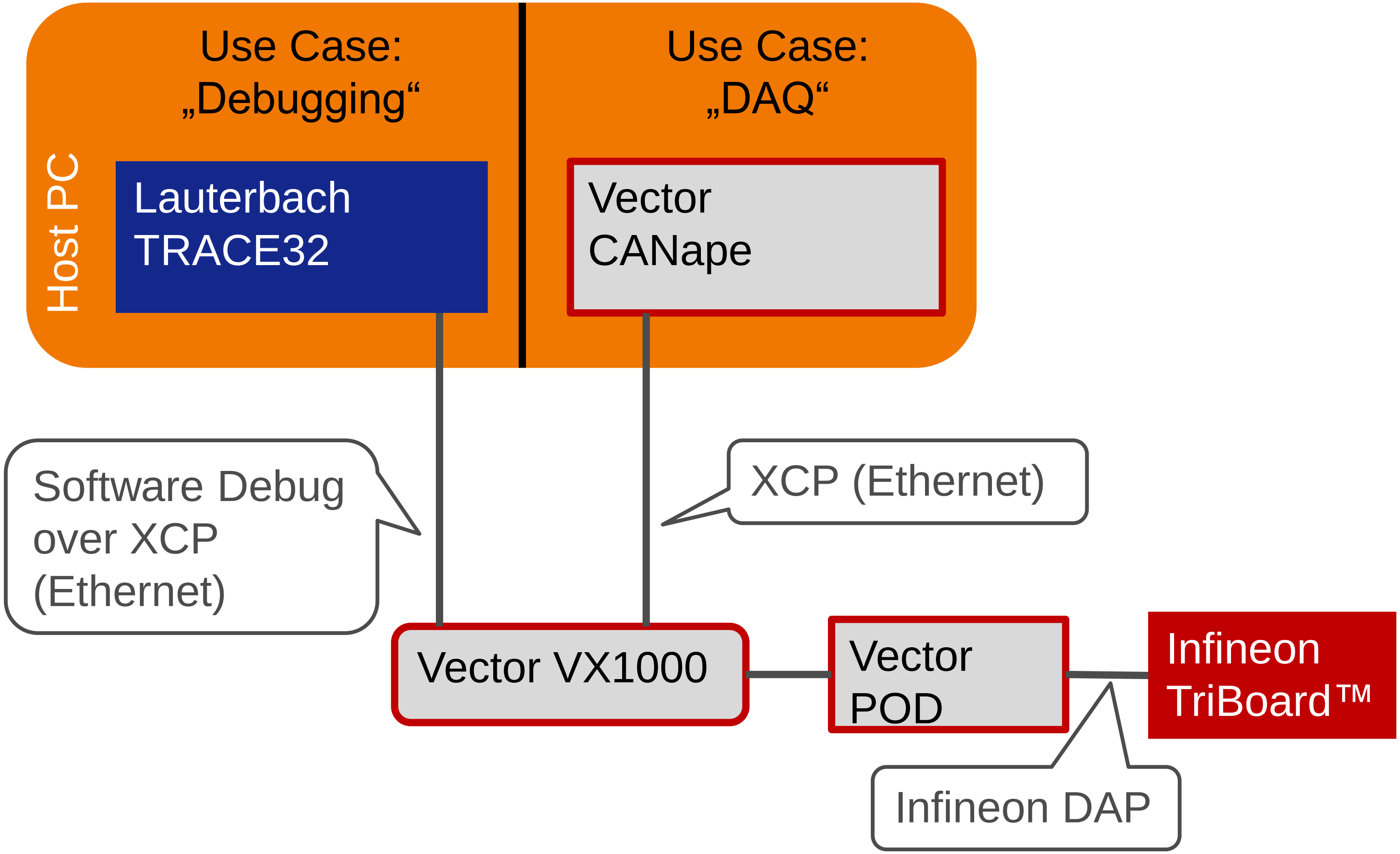
Supported CPUs

- Standard is CPU independent
- High-Level Commands
  - Mapping to target resources needs to be CPU specific
  - Mapping currently defined in appendix for

    Infineon TriCore™ Renesas RH850 MPC5xxx
  - Can be easily extend to new CPUs
- Low-Level Commands

  Infineon DAP JTAG

Live Demo



Use Case: „Debugging“

Use Case: „DAQ“

Host PC

Lauterbach TRACE32

Vector CANape

Software Debug over XCP (Ethernet)

XCP (Ethernet)

Vector VX1000

Vector POD

Infineon TriBoard™

Infineon DAP

www.lauterbach.com

2018 / JUNE

THANK YOU!

QUESTIONS?

Michael Eick
Michael.Eick@lauterbach.com

LAUTERBACH
DEVELOPMENT TOOLS