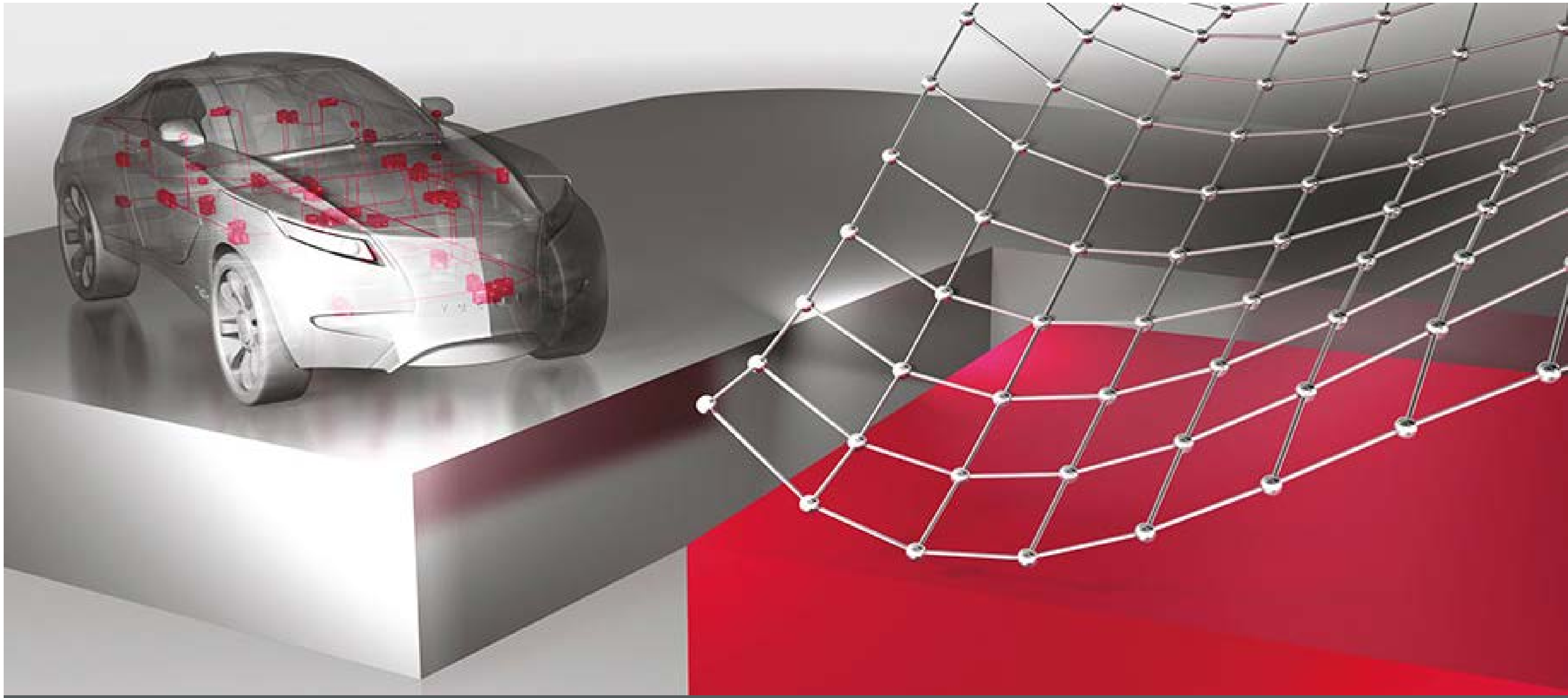


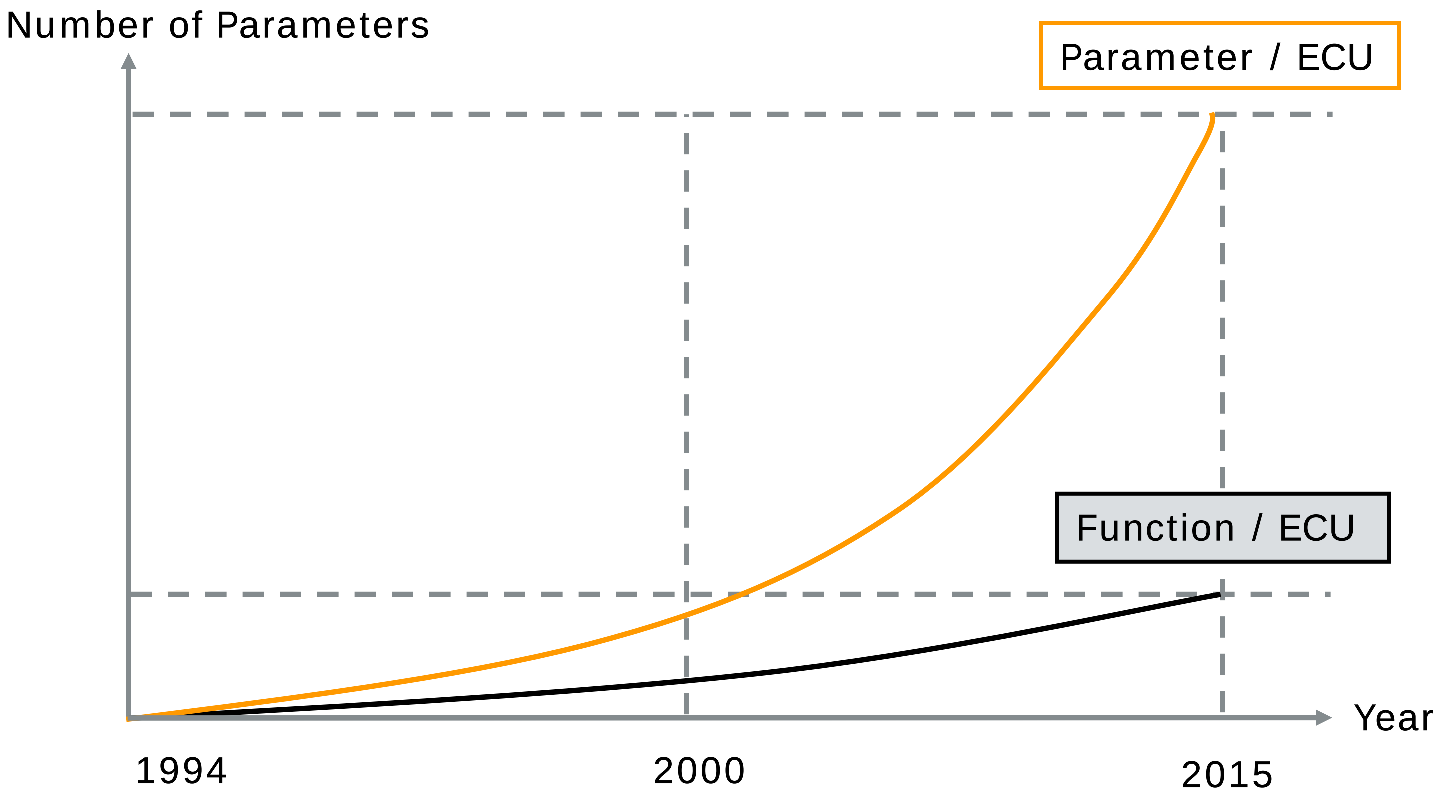
XCP Use Cases

For the Development of Distributed Embedded Systems

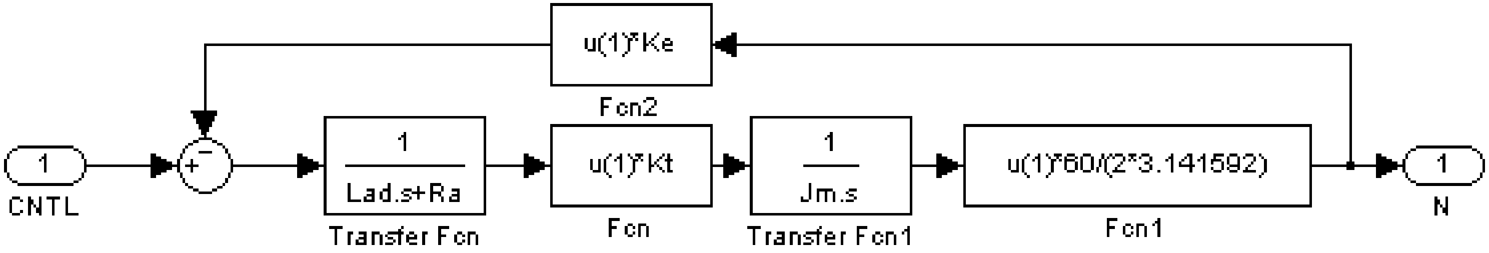
V1.1 | 2016-04-01

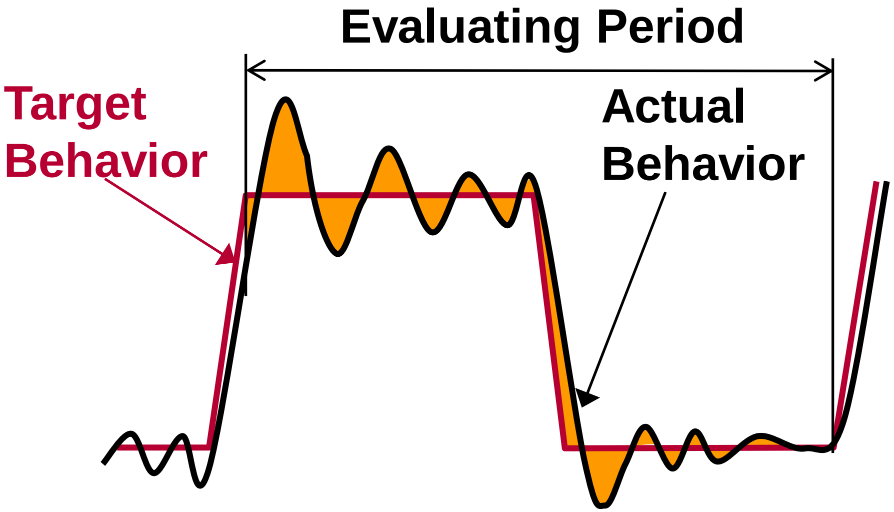
Motivation

- An ECU can have more than some 10,000 different parameters



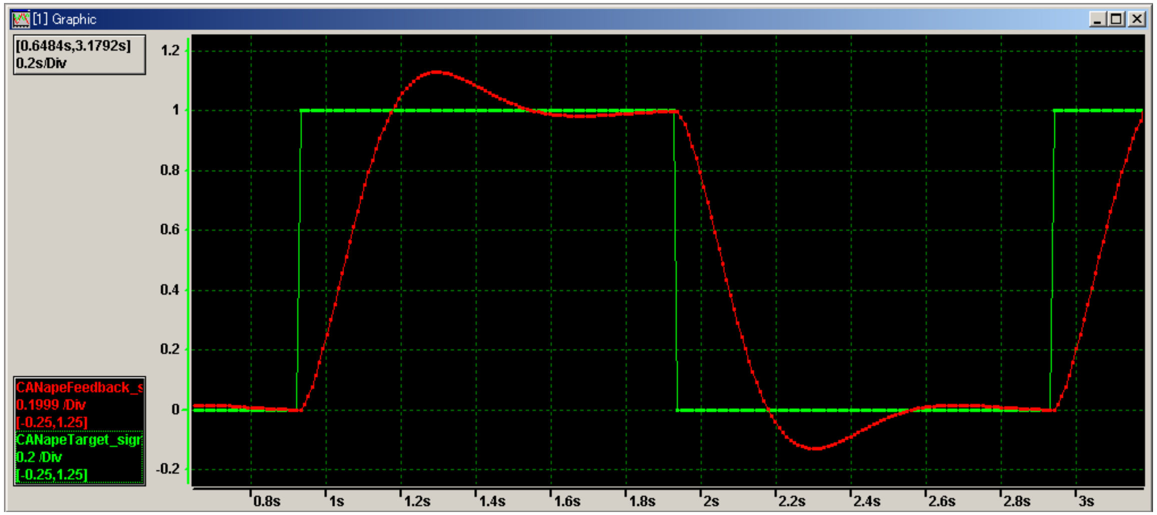
Optimization of PID control algorithm



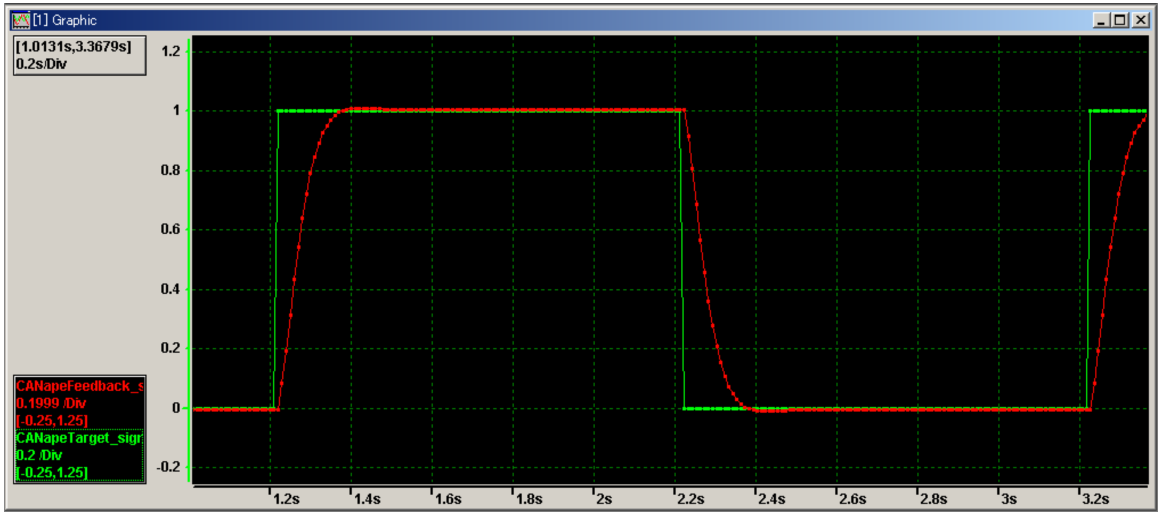


Optimization of PID control algorithm

Before calibration



After calibration



🟩 Target Value

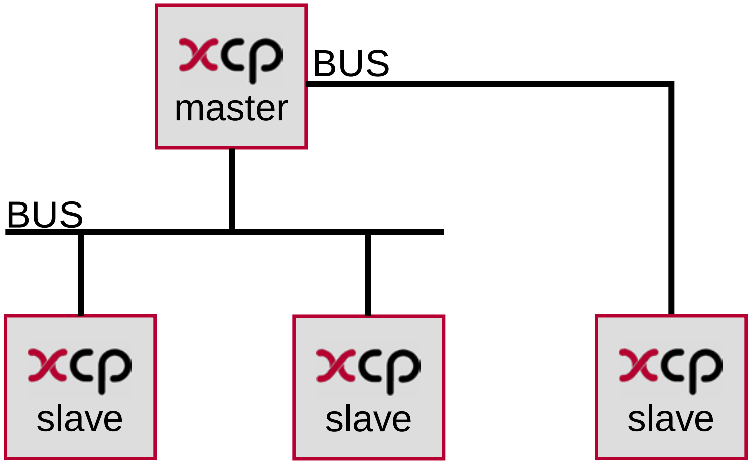
🟥 Actual Value

Goal of XCP

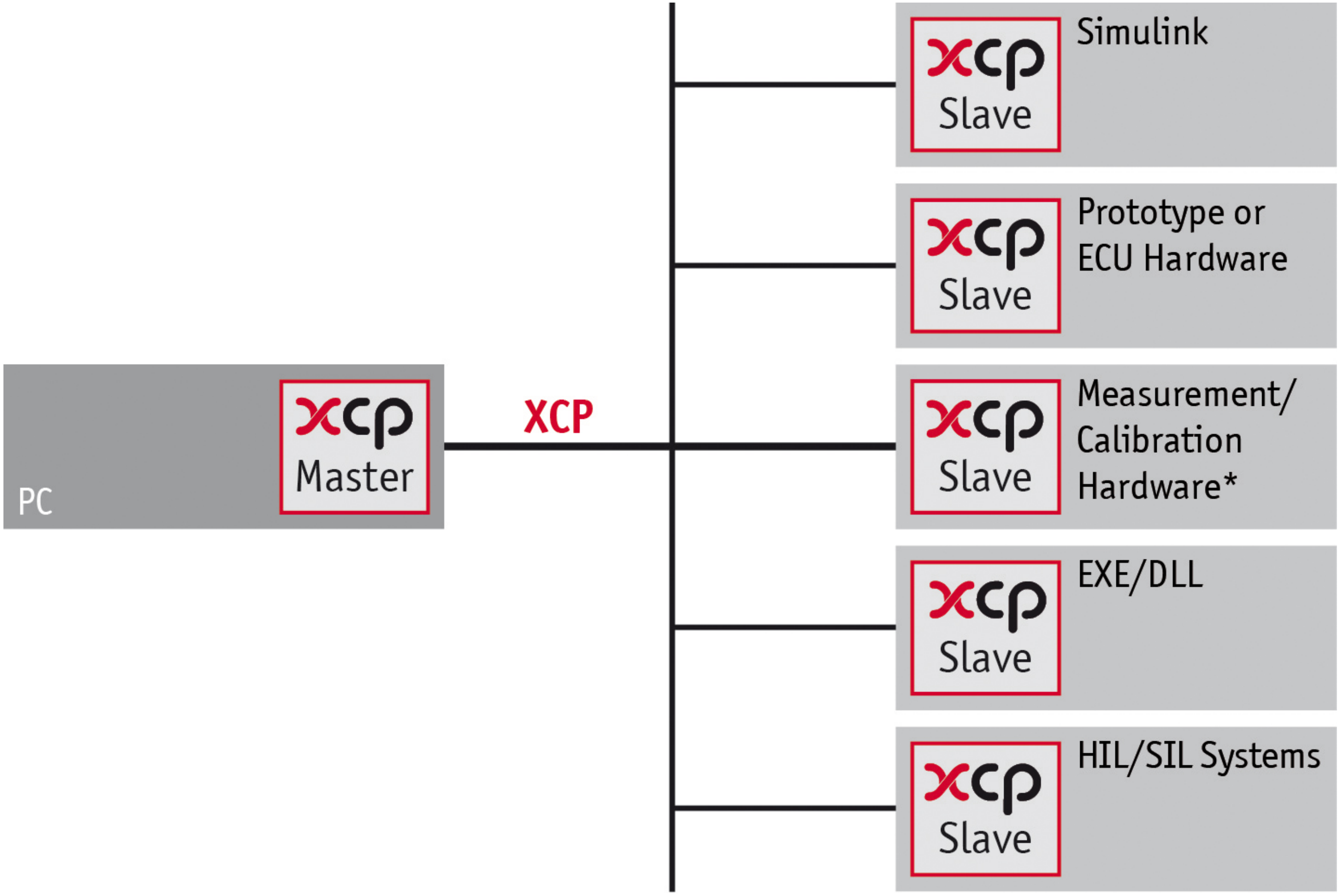
- Communication solution for various applications and use cases
- XCP was designed according to the following principles:
  - Minimal resource consumption in the ECUs (RAM, ROM, runtime)
  - Efficient communication
  - Simple ECU (slave) implementation
  - “Plug & Play”: easy configuration with small amount of parameters
  - Scalability

Basics

- XCP is the successor of CCP (CAN Calibration Protocol)
- XCP stands for Universal Calibration Protocol
- The **“*X*”** generalizes the “various” transportation layers used by the members of the protocol family e.g. “*X*CP on CAN”, “*X*CP on Ethernet”, “*X*CP on UART/SPI”, “*X*CP on LIN”, etc.
- ASAM Measurement and Calibration Interface, standard since 2003
- Single Master, Multi Slave concept

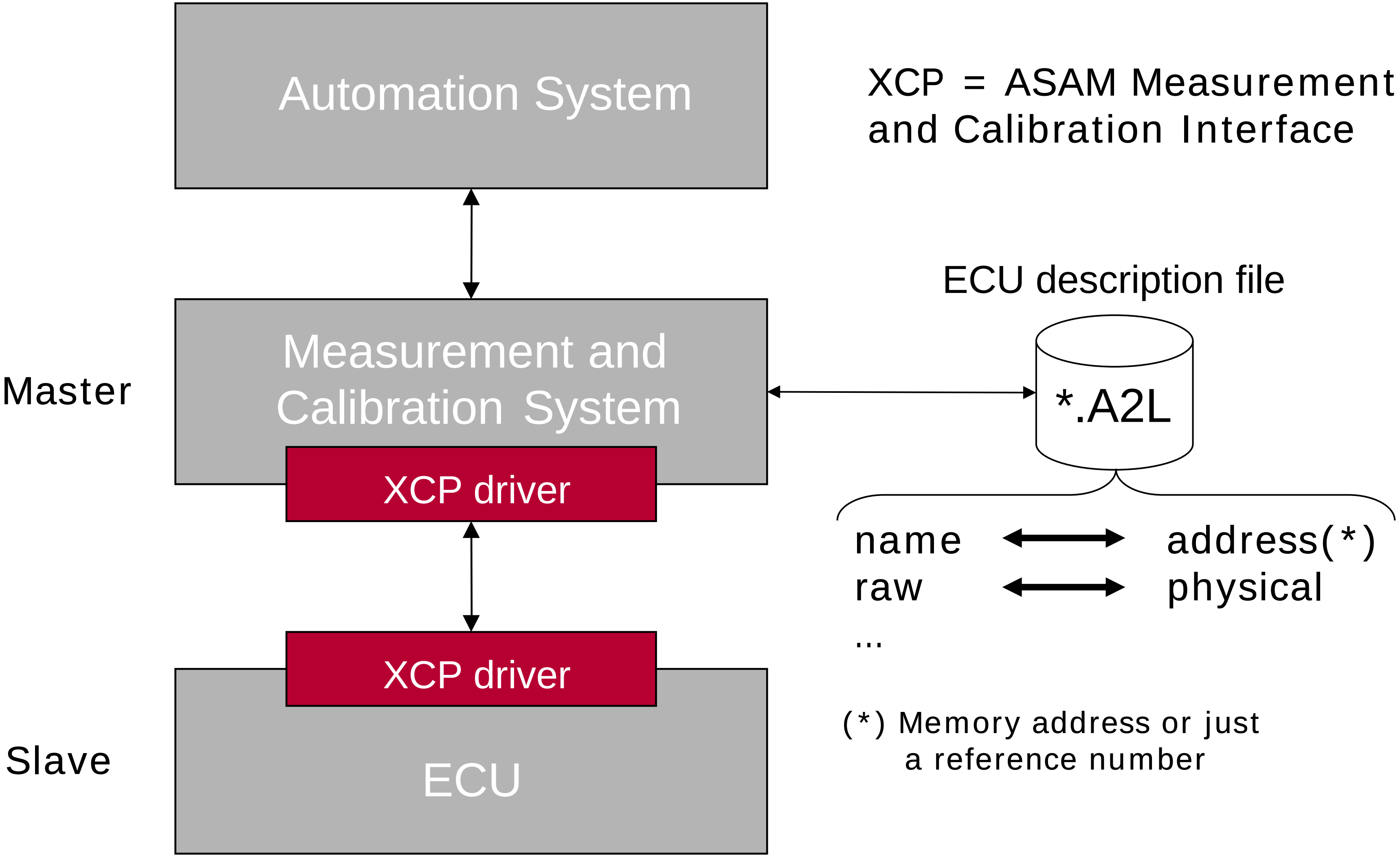


One Protocol for Every Purpose



* Debug Interfaces, Memory Emulator, ...

ASAM Measurement and Calibration Interface



XCP = ASAM Measurement and Calibration Interface

Master

Automation System

Measurement and Calibration System

XCP driver

XCP driver

ECU

Slave

ECU description file

*.A2L

name ⟷ address(*)

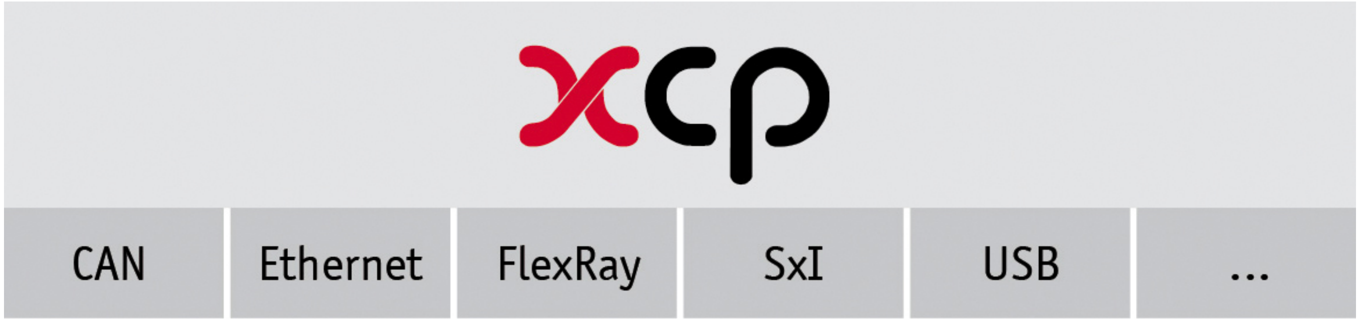
raw ⟷ physical

...

(*) Memory address or just a reference number

Different Layers

- XCP = one protocol layer and several transport layer(s):
  - Protocol layer
    - Generic measurement and calibration protocol which is independent from the network type being used
  - Transport layer
    - How XCP is transported in the different network types



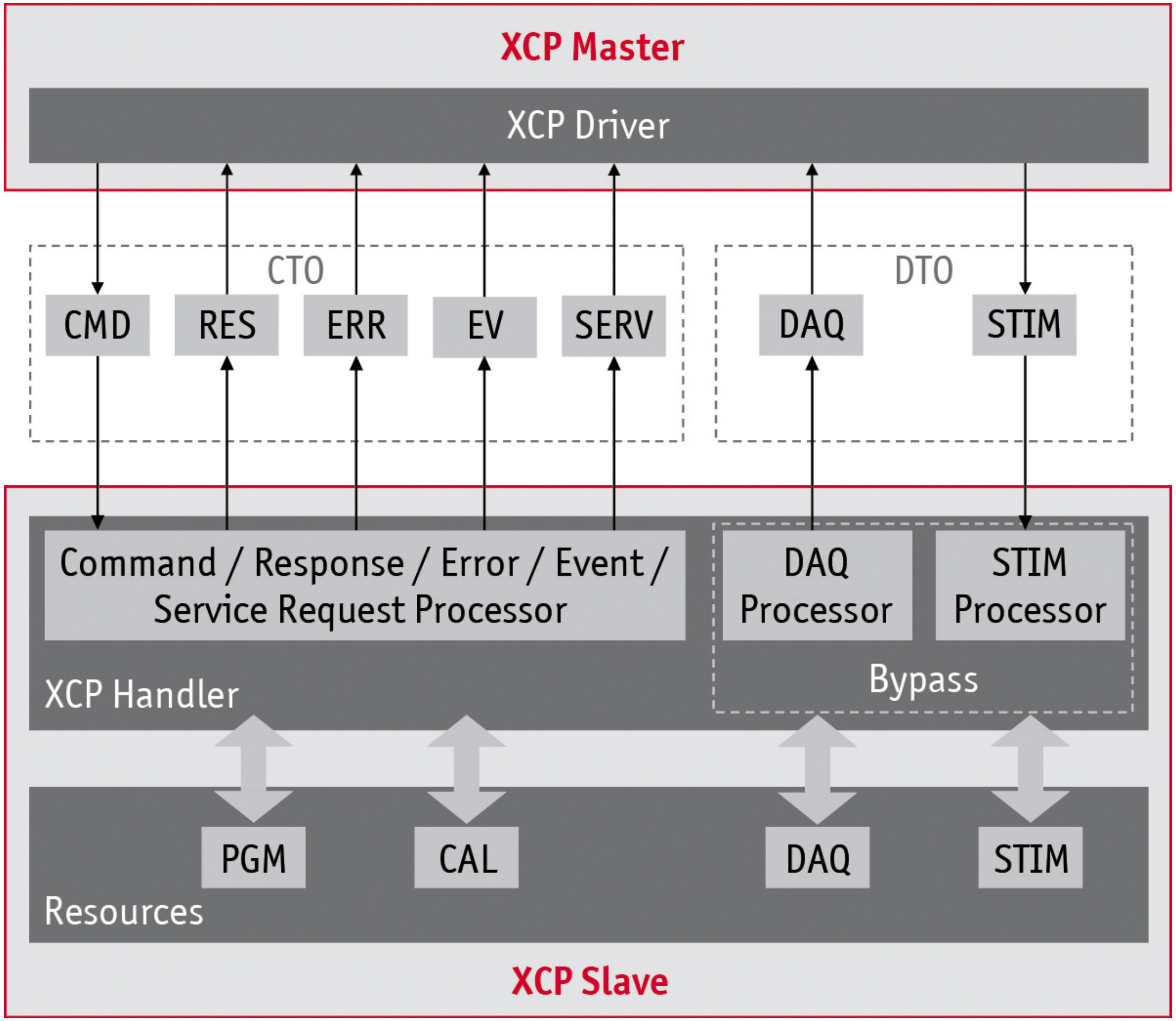
Transport Layers

	ASAM Standard
▶ XCP on CAN	✓
▶ XCP on SxI (SPI, SCI)	✓
▶ XCP on Ethernet (TCP/IP and UDP/IP)	✓
▶ XCP on USB	✓
▶ XCP on FlexRay	✓
▶ XCP on LIN	not yet
▶ XCP on MOST	not yet
▶ XCP on K-LINE	not yet

Easy migration from a fast interface in the development phase into e.g. CAN in the series phase!

VECTOR

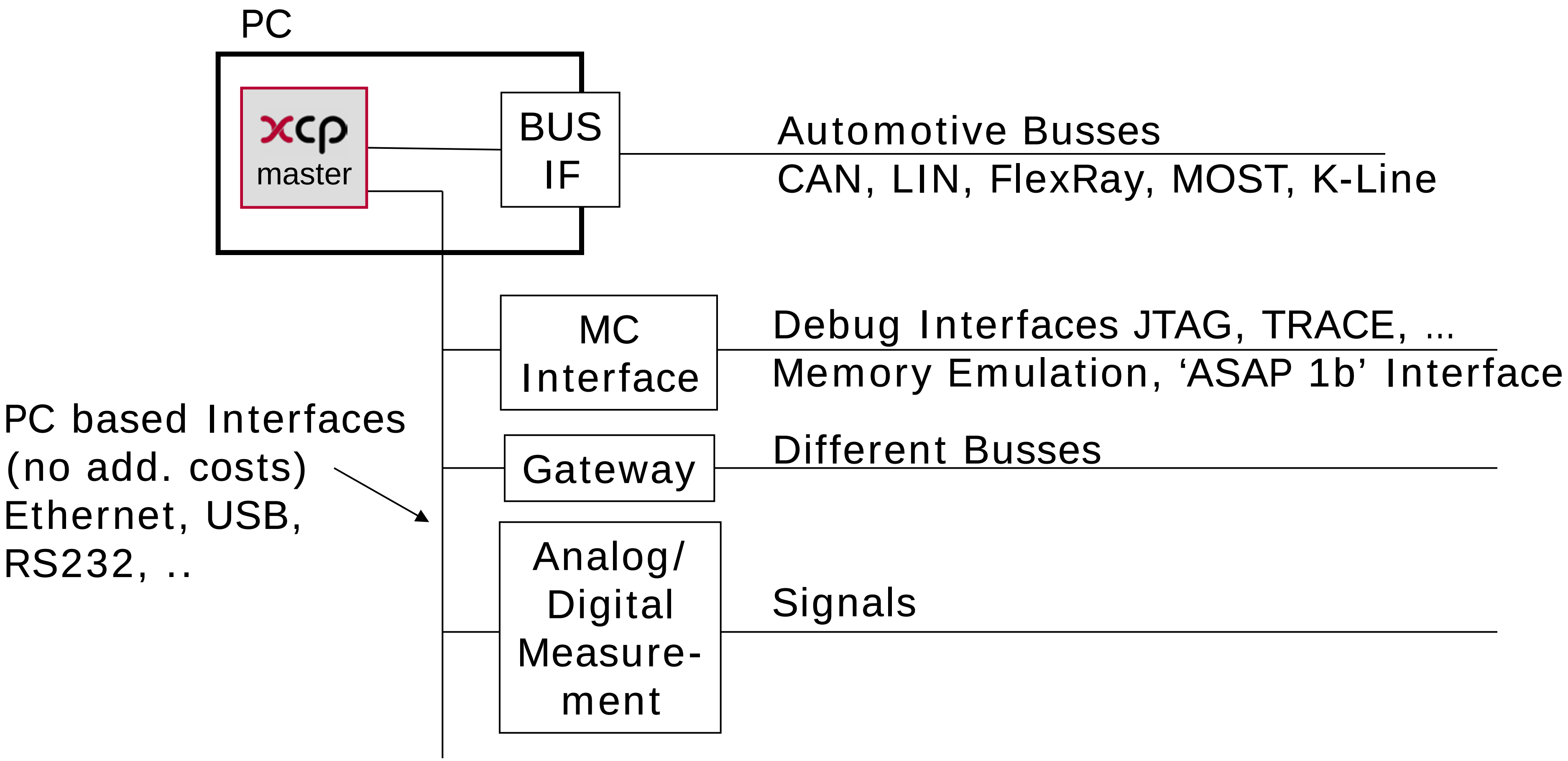
XCP Communication Model with CTO/DTO



CTO: Command Transfer Object

DTO: Data Transfer Object

Universal Communication Protocol



PC

xcp master

BUS IF

Automotive Busses
CAN, LIN, FlexRay, MOST, K-Line

MC Interface

Debug Interfaces JTAG, TRACE, ...
Memory Emulation, ‘ASAP 1b’ Interface

Gateway

Different Busses

Analog/ Digital Measure-ment

Signals

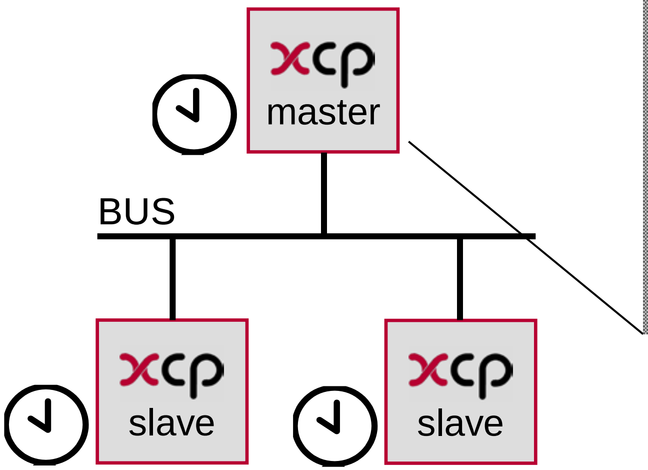
PC based Interfaces
(no add. costs)
Ethernet, USB,
RS232, ..

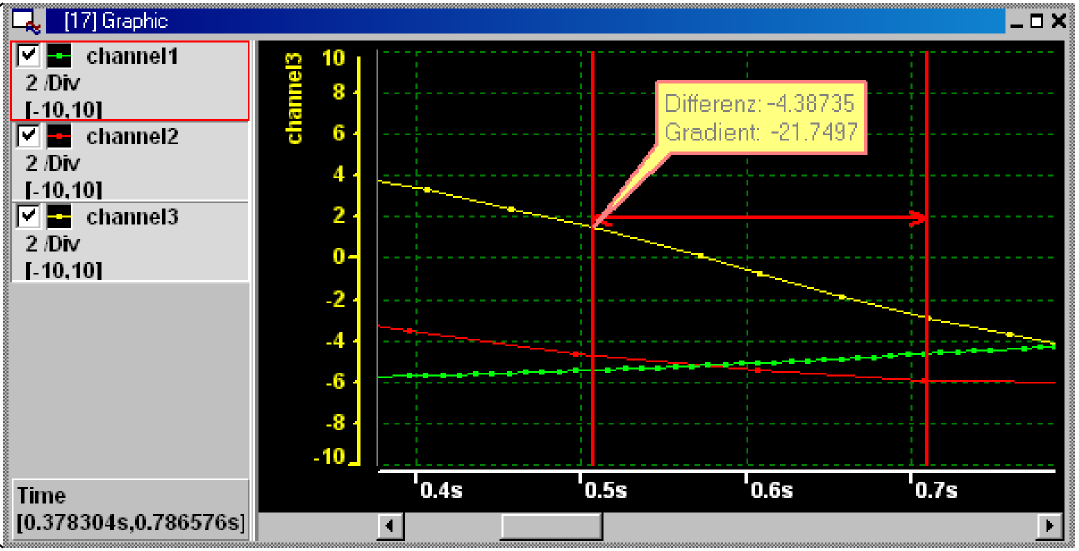
Universal Communication Protocol

- Same XCP driver code for
  - Sending few bytes out of small controllers and interfaces, e.g. 8-bit processor with serial interface
  - Sending megabytes per second over fast interface, e.g. Ethernet with 32-bit processor
- Scalability
  - Driver size: There are mandatory and optional functions to optimize necessary ROM/Flash size
  - ECU resource consumption: High throughput vs. controller runtime and RAM size
  - Bus load: Number of signals vs. bus bandwidth
- Simple implementation with only small amount of parameters

Event Synchronized Measurement of Consistent Data

- Event: task execution, timer signal, user action, single event ...
- A slave event occurs, slave samples all consistent data and transmits the data incl. the event timestamp to the master
- Master synchronizes all slave data to master clock and shows all signals on a global time axis





☑ 🟩 channel1
2 /Div
[-10,10]

☑ 🟥 channel2
2 /Div
[-10,10]

☑ 🟨 channel3
2 /Div
[-10,10]

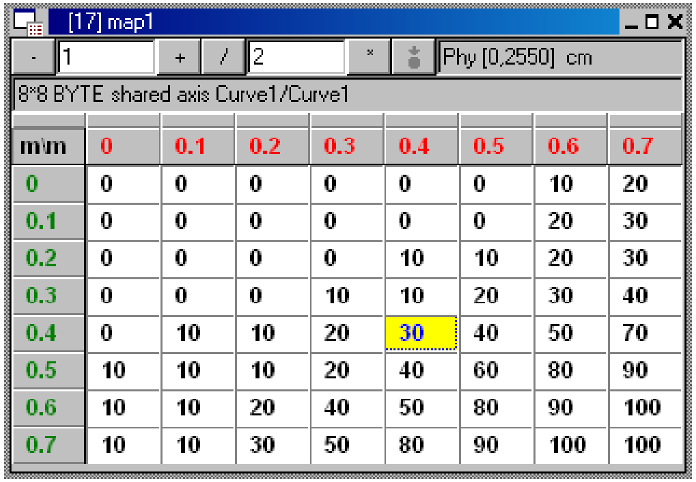
Time
[0.378304s,0.786576s]

Differenz: -4.38735
Gradient: -21.7497

VECTOR

Calibration

- Description of simple scalar parameters up to complex maps and curves with record layout and conversion rules in the A2L file
- Comfortable access to calibrate the parameters online and offline

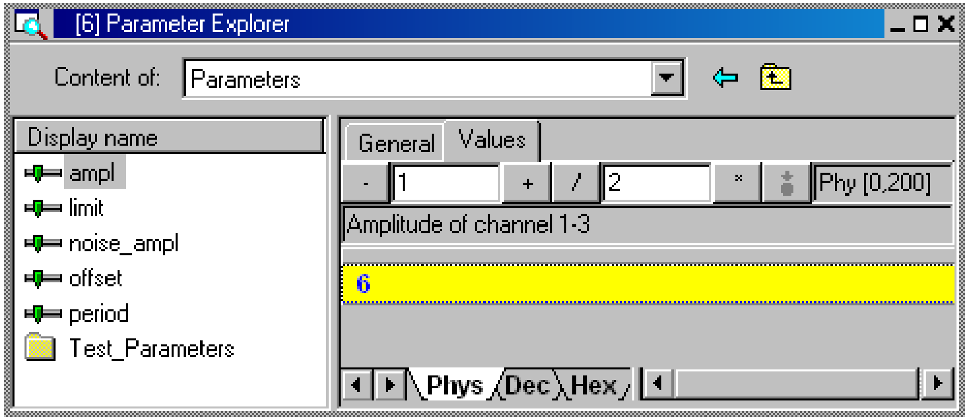


[17] map1

- | 1 | + | / | 2 | * | Phy [0,2550] cm

8*8 BYTE shared axis Curve1/Curve1

m\m	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7
0	0	0	0	0	0	0	10	20
0.1	0	0	0	0	0	0	20	30
0.2	0	0	0	0	10	10	20	30
0.3	0	0	0	10	10	20	30	40
0.4	0	10	10	20	30	40	50	70
0.5	10	10	10	20	40	60	80	90
0.6	10	10	20	40	50	80	90	100
0.7	10	10	30	50	80	90	100	100



[6] Parameter Explorer

Content of: Parameters

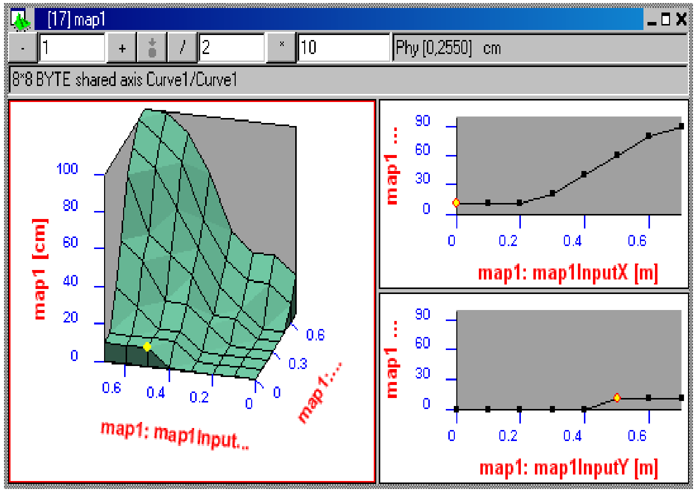
Display name
- ampl
- limit
- noise_ampl
- offset
- period
- Test_Parameters

General | Values

- | 1 | + | / | 2 | * | Phy [0,200]

Amplitude of channel 1-3

6



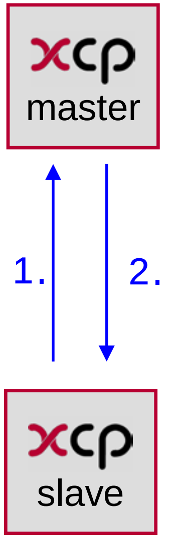
[17] map1

- | 1 | + | / | 2 | * | 10 | Phy [0,2550] cm

8*8 BYTE shared axis Curve1/Curve1

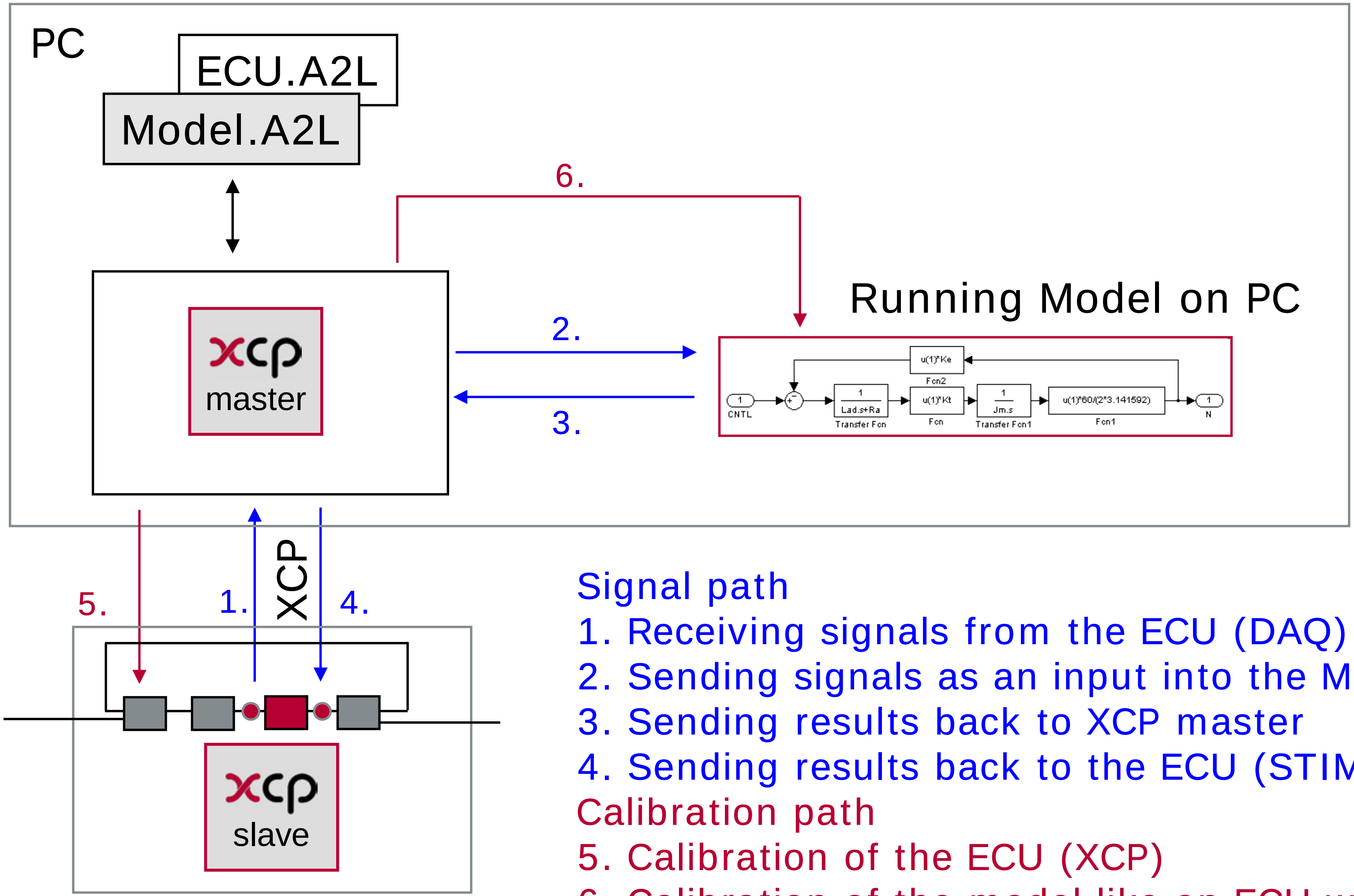
Rapid Prototyping with Bypassing and Stimulation

Bypassing can be realized by making use of synchronous data acquisition and synchronous data stimulation simultaneously.



1. Synchronous Data Acquisition (DAQ)
2. Synchronous Data Stimulation (STIM)

Rapid Prototyping with Bypassing and Stimulation



Signal path

1. Receiving signals from the ECU (DAQ)
2. Sending signals as an input into the Model
3. Sending results back to XCP master
4. Sending results back to the ECU (STIM)

Calibration path

5. Calibration of the ECU (XCP)
6. Calibration of the model like an ECU with XCP

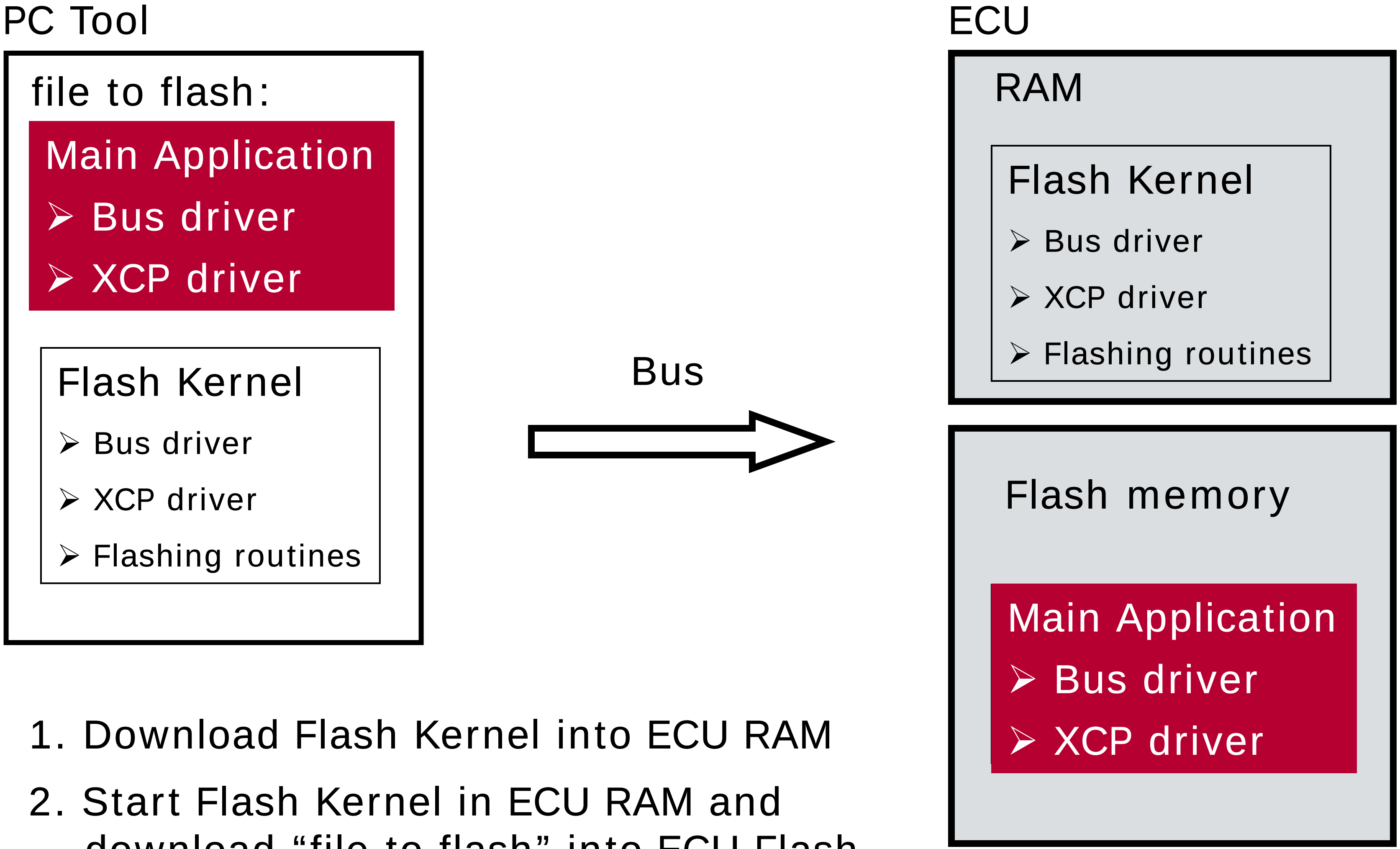
Flashing

Data stored in the flash memory can only be re-programmed via special flash routines that are necessary inside the ECU.

- Solution 1:
  The routines are integrated in the flash, permanently
  → Flash memory is wasted
  → It's a security problem in the released product

- Solution 2:
  A flash kernel is loaded via PC tool into the micro controller's RAM via XCP whenever the flash memory has to be reprogrammed. The flash kernel contains the needed flash routines, its own minimized bus and XCP driver to communicate via the bus interface with the PC tool.

Flashing



PC Tool

file to flash:

Main Application
- ➢ Bus driver
- ➢ XCP driver

Flash Kernel
- ➢ Bus driver
- ➢ XCP driver
- ➢ Flashing routines

Bus

ECU

RAM

Flash Kernel
- ➢ Bus driver
- ➢ XCP driver
- ➢ Flashing routines

Flash memory

Main Application
- ➢ Bus driver
- ➢ XCP driver

1. Download Flash Kernel into ECU RAM
2. Start Flash Kernel in ECU RAM and download “file to flash” into ECU Flash Memory

Future Prospects

- XCP on LIN proposal for a new working group is on the way
- Functional addressing: In the course of distributed functions we need a solution to address a parameter of a function, regardless were the function and parameter is.
- Synchronous calibration: To realize synchronous calibration in more than one ECU at the same time, we need an answer to address different slaves at the same time. A kind of broadcast addressing may be the right way.

Summary

- Tool- and vendor-independent communication platform (like TCP/IP on Ethernet for office communication and internet)
- No need for usage of expensive proprietary protocols
- Event synchronized measurement and stimulation
- Small scalable code for any bandwidth
- Useable for every purpose: from function development and modeling to production car application
- Approved in many customer projects
- Vector offers the most sophisticated XCP master tool (CANape) and XCP source code on the market. Available since 2002.

For more information about Vector and our products please visit

www.vector.com

Author:
Andreas Patzer
Vector Germany