

Software-in-the-Loop using virtual CAN buses: Current solutions and challenges

5. Tagung Simulation und Test für die Automobilelektronik

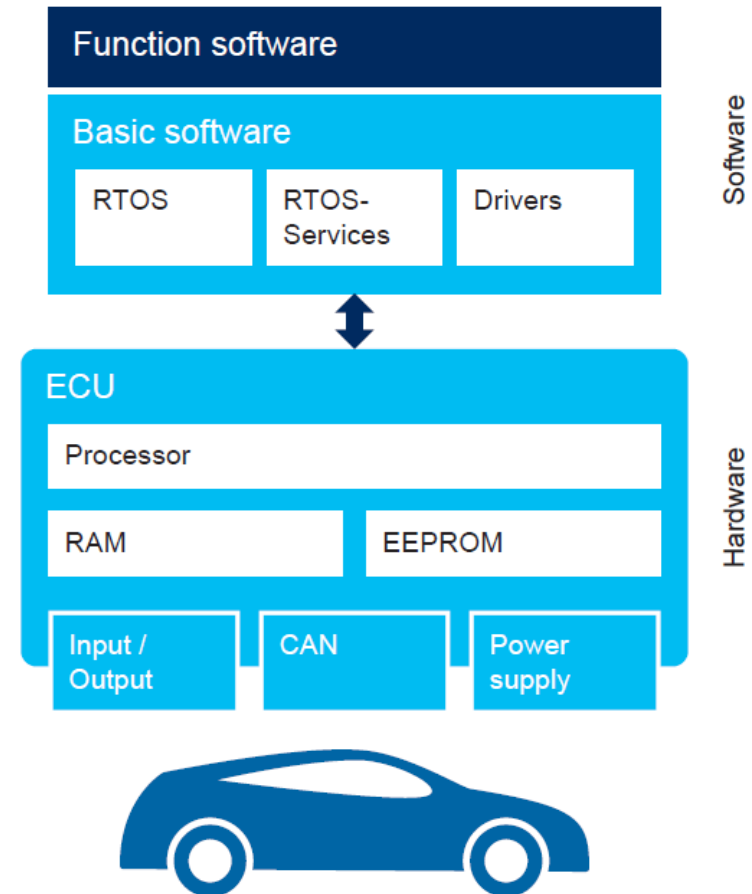
Dr. Th. Liebezeit¹, Dr. A. Junghanns², M. Bonin¹, R. Serway¹, Berlin, Mai 2012

Software-in-the-Loop using virtual CAN buses

Motivation

Motivation

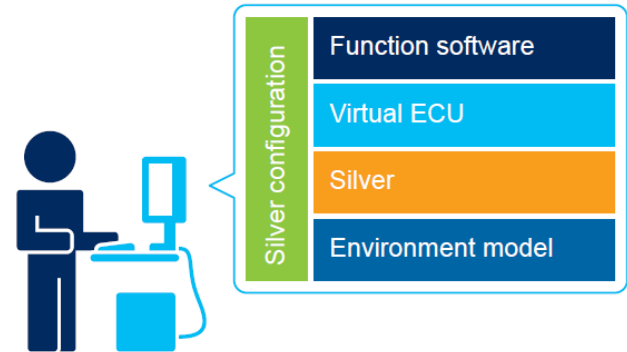
- Series Transmission Software development
 - Function software development
 - Basic software from ECU supplier
 - Different software variants
 - C-Code (Hand coded, auto code from TargetLink)
- Frontload development tasks
 - Debugging of series transmission function software
 - Functional behaviour testing
 - Fully utilized HiL systems



Software-in-the-Loop using virtual CAN buses

Motivation

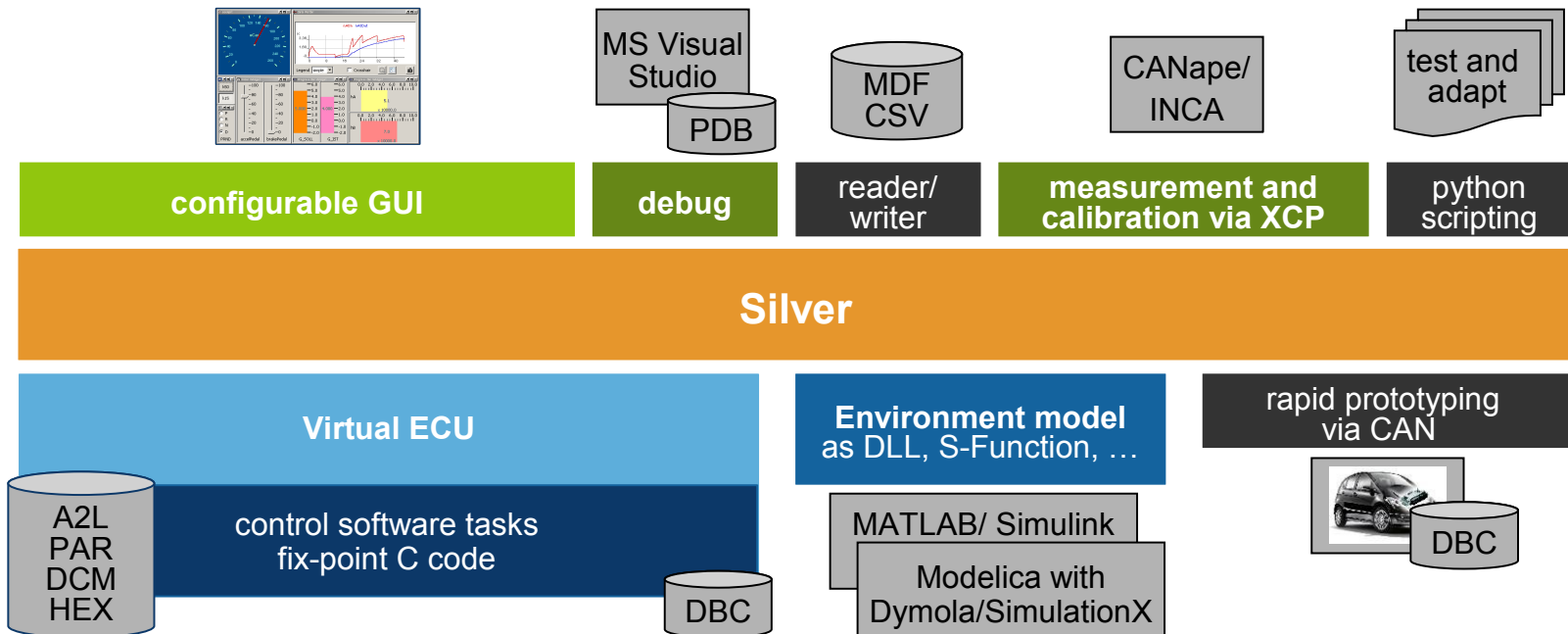
- Software-in-the-Loop (SiL)
 - Integral step in the development process
 - All-time deployable by developer
 - Closed-loop simulation on Developer PC
 - Enables convenient debugging
 - Faster change-analysis-change cycles
- Controller Area Network (CAN)
 - most commonly used inter-ECU communication
- **Objective**
 - Use CAN in simulation



Software-in-the-Loop using virtual CAN buses

Silver

- Silver from QTronic GmbH
 - Software-in-the-Loop (SiL) simulation environment
 - All relevant automotive standard formats supported (A2L, PAR, DBC)
- IAV has already experience with Silver for 2 years



Software-in-the-Loop using virtual CAN buses

IAV Software-in-the-Loop setup

- **Virtual ECU**

- Hardware and BIOS software
 - IO interface
 - Timing of tasks
 - BIOS functionality
 - Non-volatile memory

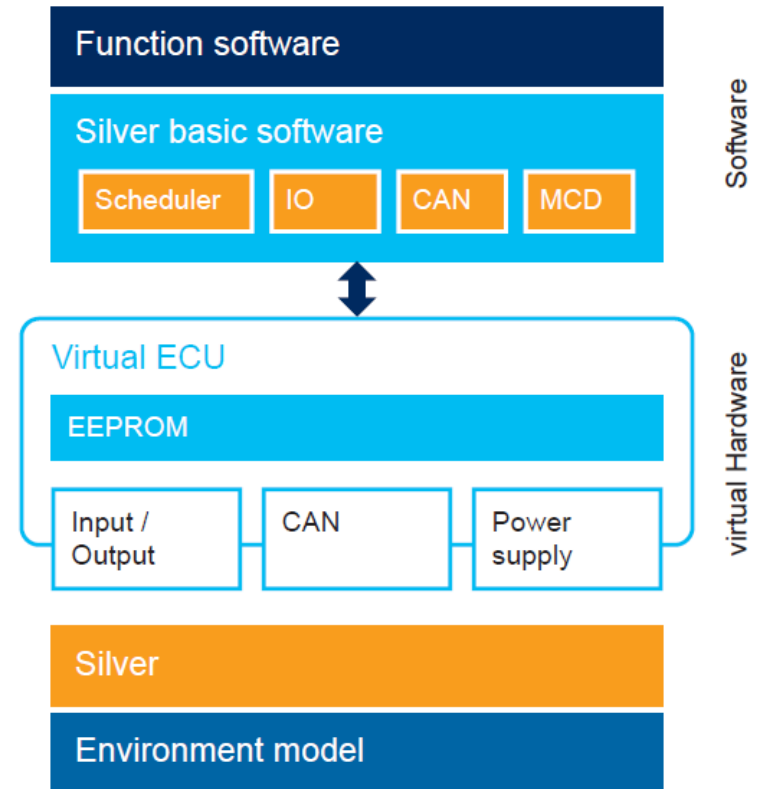
- uses C-Silver-API

- **Environment model**

- Longitudinal vehicle dynamics, detailed transmission model, and CAN rest bus
- Reuse of HiL models

- **Silver configuration**

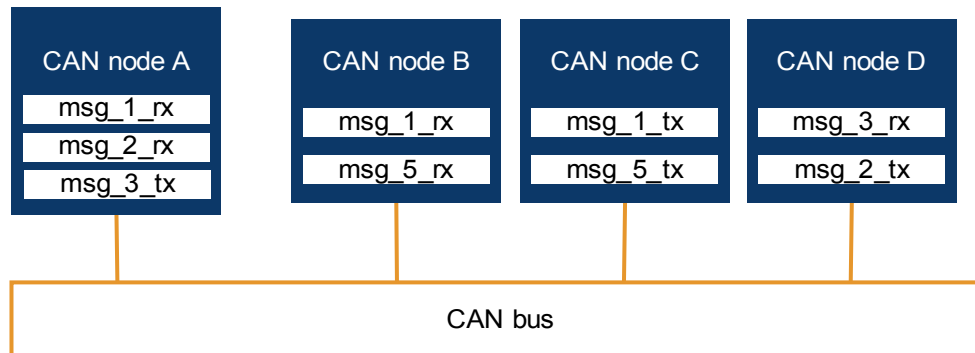
- Graphical user interface
- PAR file flashing, Access A2L variables



Software-in-the-Loop using virtual CAN buses

CAN Basics

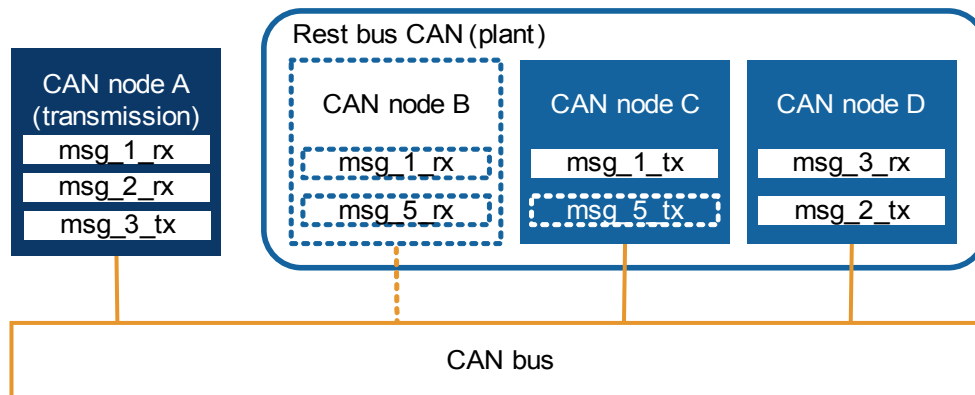
- Bus: One transmitter - multiple receivers
- Node: typically one ECU, transmits or receives messages
- Message: up-to 8 Byte data, cyclic or event-based, priority
- Signal: packed into a message, scaled by gain/offset (1-64 bit)
- DBC File: specifies CAN bus, especially message structure and timing



Software-in-the-Loop using virtual CAN buses

CAN Basics: rest bus

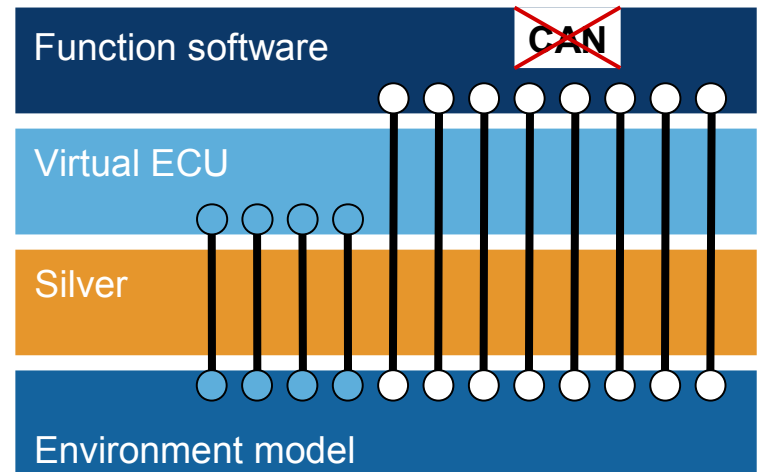
- SiL simulation focused on one ECU
 - DBC defines whole bus
 - Not all nodes/ messages are needed in simulation
 - Emulation filters by
 - node names
 - black listing or white listing of messages



Software-in-the-Loop using virtual CAN buses

Signals in common SiL

- Common SiL signal rooting (without CAN)
 - Virtual ECU/ Function software (Silver C API)
 - Remove CAN code
 - define **Silver I/O** for code variable (gain and offset manually)
 - Model (Silver Simulink block set)
 - define **Silver I/O** for Simulink signal
- Silver
 - detects Silver I/O signals by name
 - copies information automatically at begin/ end of simulation step
 - connection data type: double
- Summary
 - Function software CAN code is bypassed
 - Typically many signals to be set up



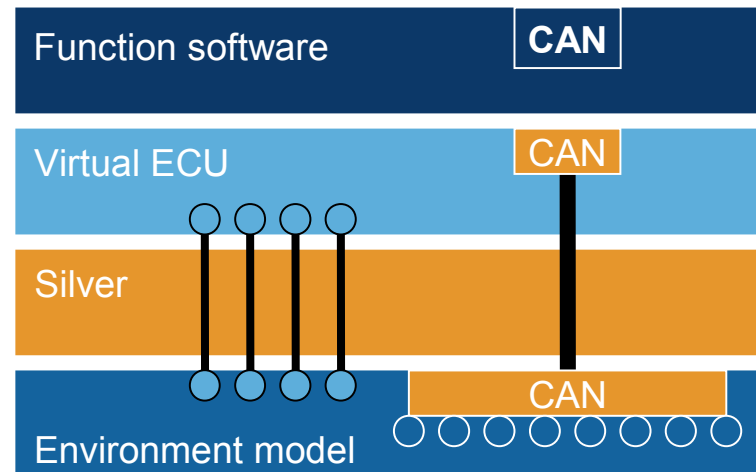
Legend:

- Pin Silver I/O
- CAN Silver I/O

Software-in-the-Loop using virtual CAN buses

Signals in SiL with CAN (1)

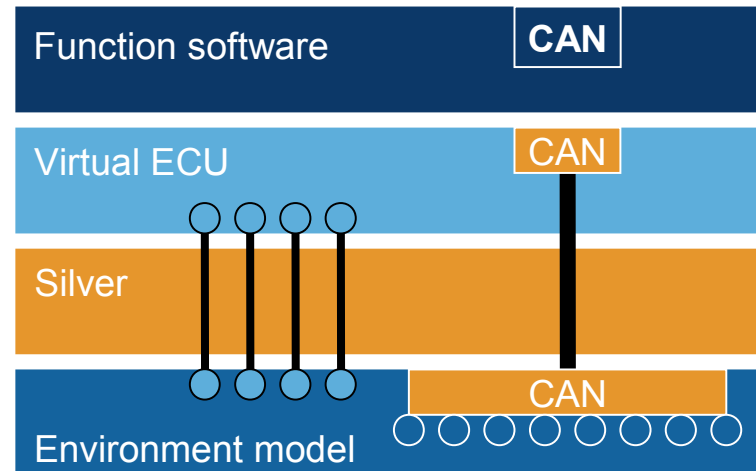
- SiL with CAN
 - Virtual ECU (Silver C API)
 - define **Silver I/O** for ECU pins
 - define **CAN emulation** (by DBC node) to access message data
 - Model (Silver Simulink block set)
 - define **Silver I/O** for ECU pins
 - define **CAN emulation** (by DBC nodes) and connect signals to CAN blocks
- Silver
 - Silver I/O are handled as before
 - CAN emulation (by DBC usage)
 - messages are identified by ID, multiple buses possible
 - Silver copies information automatically with DBC timing
 - connection data type: from DBC, automatically scaled



Software-in-the-Loop using virtual CAN buses

Signals in SiL with CAN (2)

- Summary
 - Function software CAN code in simulation, can be debugged and tested
 - Reduced communication setup effort
 - Less manual definitions
 - Automatic scaling from/ to CAN message
 - Use of DBC work product
 - Car comparable communication behaviour



Software-in-the-Loop using virtual CAN buses

Using Silver CAN emulation: Code

- Silver API 2.0 defines C functions for:
 - Configure CAN bus or busses by DBC file or by single messages
 - Start/ stop CAN emulation
 - Check new message received
 - Transmit/ receive message (8 byte data)
 - Transmit/ receive signal/ variable (from a message, raw or scaled)
 - Manipulate messages for counter/ CRC (call-back dll)

Virtual ECU

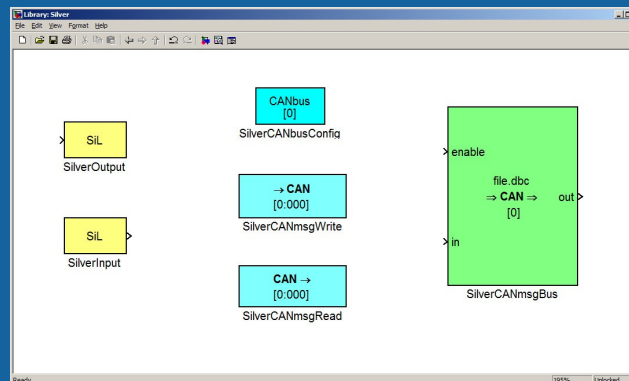
```
void SBS_USER_get_module_interface20 (void *sbs, int argc, char **argv) {
    // first comes the definition of tasks
    ...
    // second is the Silver input/ output variable definition
    ...
    // now we want to create a can bus using a dbc file, node and ignore file
    SBS_CONF_AddDBC (
        sbs,          // sbs handle
        1,           // can bus ID
        "trans.dbc", // file name
        "Transmission", // node
        "ignore.txt", // ignore file name
        0x0,         // flags, z.B. SBS_DBC_ENFORCE_RANGES
        0x0,         // channel mask
        NULL         // modified signals, e.g. message counter and CRCs
    );
    return DLL_OK;
}
```

Software-in-the-Loop using virtual CAN buses

Using Silver CAN emulation: Simulink

- Silver CAN block set
 - Bus message setup (SilverCANmsgBus)
 - Configure one CAN bus by DBC and node names
 - Enable/ disable node by Simulink input bus
 - Transmit messages from Simulink input bus
 - Output received messages to Simulink output bus
 - Single message setup (SilverCANbusConfig, SilverCANbusRead, SilverCANbusWrite)

Environment model

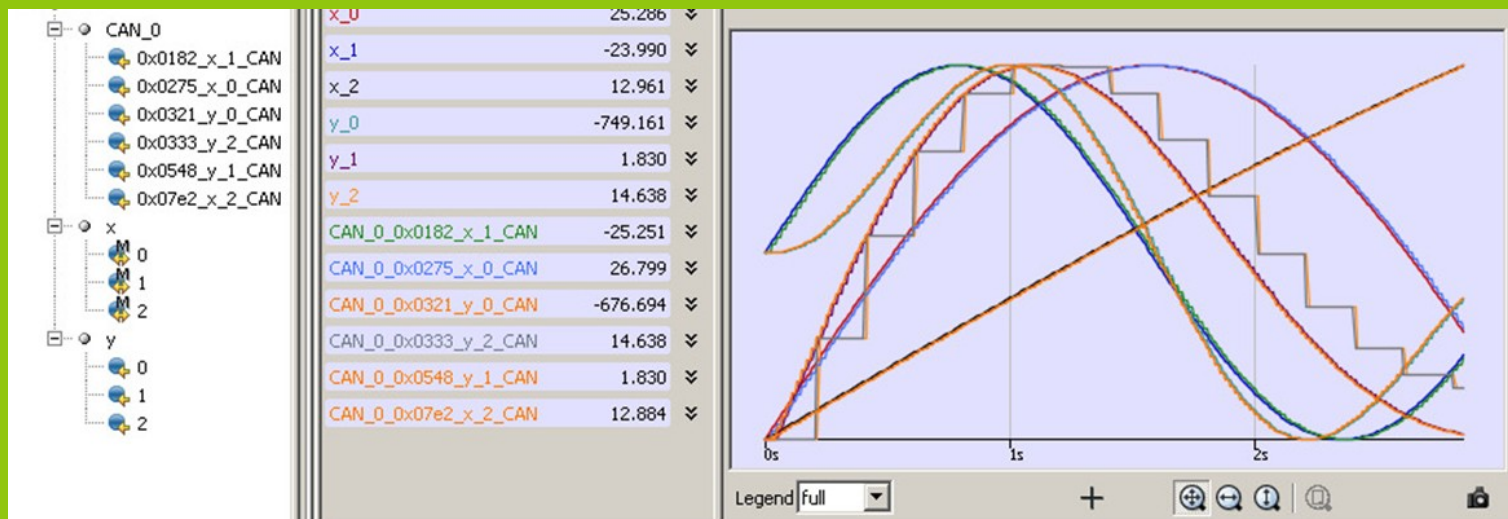


Software-in-the-Loop using virtual CAN buses

Using Silver CAN emulation: User GUI

- See connection state of CAN messages
- Easy access to CAN signals for plotting/ debugging

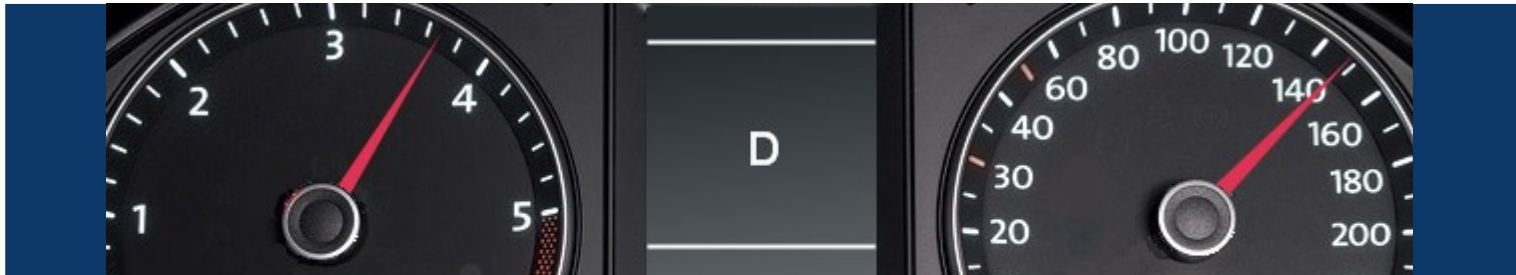
Silver GUI



Software-in-the-Loop using virtual CAN buses

Summary

- Network communication is an important part of system design
- Using CAN emulation in SiL simulations
 - drastically reduces the effort for defining SiL communication setup
 - improving the consistency of definitions
 - adds additional simulation aspects (signal scale, communication timing)
 - enables tests of the CAN-related software parts (which are up to now bypassed)



Thank you!

Dr. Thomas Liebezeit
IAV GmbH

Carnotstraße 1, 10587 Berlin
Telefon +49 30 39978-9021

thomas.liebezeit@iav.de

www.iav.com