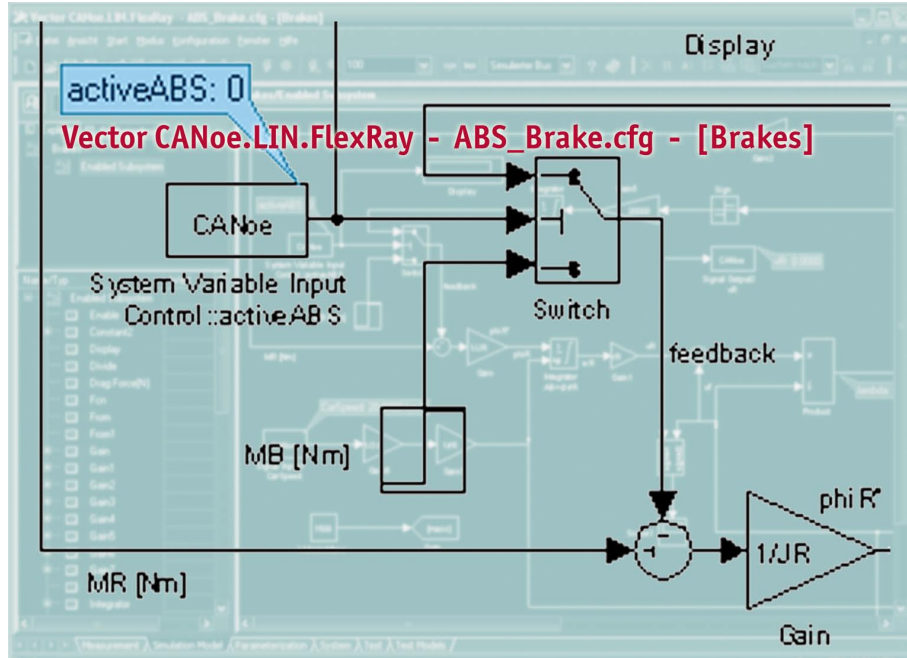


Model-Based Development of ECUs

Software Simulation with MATLAB/Simulink and CANoe



MATLAB/Simulink is a tool that is widely used in many engineering and scientific disciplines. In the automotive field, for example, it is used to model dynamic systems such as control systems, and to describe states and their transitions. Since these algorithms run on ECUs that communicate with one another, it is vital to provide access to the vehicle network over the course of the development process.

Access is provided directly from the model via Simulink blocks that communicate with the bus hardware. Nonetheless, a number of disadvantages are associated with this approach:

- > The model is always network-specific. Although the network and application content can be separated by intelligent structuring with subsystems, it is more advantageous to have a generic model that can very easily be adapted to different networks.
- > Additional protocol layers such as Network Management (NM), the Interaction Layer (IL) or Transport Protocols (TP), as well as a rest-of-bus simulation, can only be implemented with enormous effort in a Simulink model.

Easy access to a network can be obtained, for example, by using CANoe a simulation and testing software tool from Vector. This software is typically used to simulate the ECU network with all its nodes (remaining bus simulation). The communication layers of each node perform network-specific tasks, such as sending with the correct send type in CAN or scheduling in FlexRay. Other protocols, such as NM or TP, can easily be added in the form of OEM-specific components. The application layer of a node, the behavior described by a MATLAB/Simulink model, is placed over a pure signal interface (Figure 1). There is no more reference to the specific network. The model only writes signal values to its output and reads signal values at its input. Whether the signal is defined in a

CAN message or in a LIN or FlexRay frame is irrelevant from the perspective of the model. This abstraction on the signal level lets users re-use their existing models practically unchanged. All that needs to be done is to map the model's input and output ports to the relevant signals on the network.

Not only can signals be exchanged between CANoe and the Simulink model; system variables can be exchanged as well. These are

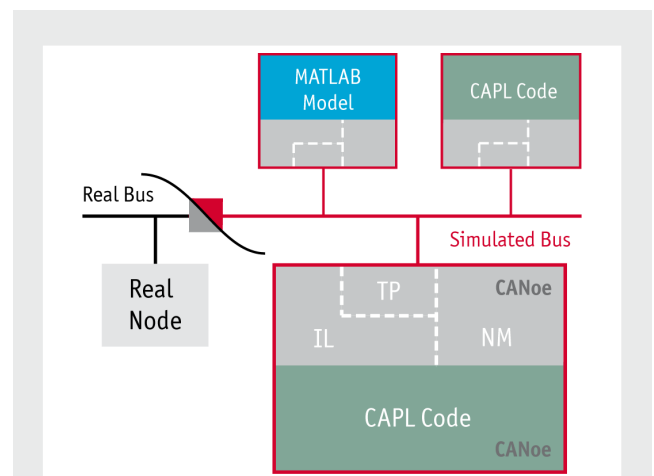


Figure 1: CANoe rest-of-bus simulation with simulated nodes

internal CANoe variables that can either be explicitly created by the user or automatically generated. Automatic generation could be implemented by linking digital/analog hardware to CANoe. Here, each port of the D/A hardware is mapped to a dedicated system variable. This lets the model interact directly with the connected D/A hardware. After installing the CANoe MATLAB integration package from Vector, a Simulink Blockset is available with the relevant blocks for exchanging data with CANoe. These blocks primarily consist of functions for writing and reading bus signals, system variables and environment variables. In addition, they contain blocks for calling CAPL functions. It is also possible to react to calls of CAPL functions, e.g. to control triggered subsystems in Simulink.

Offline Mode for Rapid Prototyping



This mode should be selected for very early development phases, when frequent changes still need to be made to the model, and the real existing network does not play an important role yet. The application developer can still test model behavior in relation to a fully simulated network. A model developed in this early phase can also be re-used in all later phases without needing to make any further modifications to the network.

Within Offline mode the simulation itself is started in the Simulink environment. CANoe works in Slave mode, and Simulink is the master for the simulation session. The simulation is computed based on the specific computer speed. All Simulink debug features can be used here, and there is no need to connect real hardware to CANoe.

Synchronized Mode for Early Development Phases



This mode was also designed for early development phases, in which the model has not yet attained its finished state. Compared to Offline mode, the Synchronized mode offers the additional option of integrating real hardware. Here too, the simulation is started in Simulink. Unlike Offline mode, the time from CANoe serves as the basis for the simulation time here, and the simulation is computed in real time. One limitation worth mentioning is that the model may not be so complex that it is not possible to quickly compute it as real time. That is because adapting the “Simulink time” to “CANoe time” slows execution speed in Simulink and conforms it to CANoe’s real time base. However, this slowing cannot be performed any longer if model computation requires too much effort. If the model does satisfy the condition mentioned above, full access to real hardware is assured in this operating mode. The debugging capabilities of Simulink can also be used. However, synchronization of the simulation time is lost whenever model execution is paused.

Online Mode or Hardware-In-The-Loop Mode



Model computation is executed entirely in the CANoe environment here. This involves generating a DLL from the Simulink model, which can be loaded and executed in CANoe. The Real-Time Workshop of MATLAB/Simulink controls code generation here. To generate code for CANoe, a special “Target makefile” was developed for the Real-Time Workshop environment that controls the generation process. The generated code is then compiled and linked using Microsoft Visual Studio compiler. This results in a Nodelayer DLL that implements a CANoe-internal API and can be added as a component to a node in the Simulation Setup. Other examples of such components are the OEM-specific Interaction Layer and components for network management or transport protocols. CANoe loads these components at the start of a measurement, and they are released at the end of the measurement. Therefore, when the model is recompiled, it is sufficient to end the measurement to integrate the changed components in the simulation.

Consequently, model execution takes place entirely within CANoe’s real-time environment. All events, such as actions on the network, timers and user inputs, are computed to precise cycles. The model is part of the CANoe configuration, and it is easy to transfer to other parties. Later it will be shown how the model can still be parameterized at a later time, i.e. without re-compiling. First, we will present a more precise description of how the model is executed in CANoe.

Model Computation in CANoe

To understand model computation better, it is important to understand the execution model in CANoe. Essentially, CANoe computes in an event-driven way. In this context, events are incoming network messages or timers. With each incoming event, the simulation time in CANoe is set to the time stamp of the given event. The time base is derived from high-resolution, non-Windows timers at the network interfaces. This is a way to achieve a high level of precision of the time stamps, even though CANoe is running under Windows. If a generated MATLAB/Simulink model is now added to the configuration in the form of a DLL, it must be computed periodically. As mentioned above, timers are also events that can set the simulation time of CANoe. Therefore a timer is set in the model, with exactly the time specified in the Solver settings of the Simulink model.

Parameterizing the Model

Users often wish to modify the model afterwards, i.e. without recompiling. Use cases can be sub-classified as follows:

- > In the model, certain starting conditions are defined at the beginning of the simulation, which may not change the actual

model logic, but may be entirely different in different test scenarios. Example: It might be necessary to modify the gain factors in control loops to test different control characteristics.

- > The Real-Time Workshop and/or a Microsoft Compiler is unavailable, or the time required for frequent recompilation is too long, or the model file itself is unavailable. Here, users can still modify the models at runtime without a MATLAB/Simulink installation.

In such cases, it would be rather cumbersome to parameterize the model via separately generated DLLs – this would require continual, time-consuming reconfiguration of components at the nodes of the Simulation Setup. To properly handle these use cases, components generated for CANoe were made to be parameterizable after compiling. In code generation, system variables are generated for each parameter in the model, which CANoe then reads and writes. Model parameters are typically the properties of individual Simulink blocks. In the case of an Integrator block, this might be its initial state, or in the case of a Sinusoidal block its frequency, phase, offset or amplitude. A system variable, e.g. representing the gainfactor of a Simulink Gain block, can then be used in the analysis windows (Graphic, Data, Trace) as well as on panels, and in CAPL programs and tests. Another advantage is the ability to fine tune the model, since it is easy to vary internal parameters iteratively.

Viewing a Simulink Model in CANoe

MATLAB/Simulink models can be observed in CANoe, provided that they exist in the form of a generated component. A separate window is available for this at the specific node after it was configured accordingly (Figure 4). To modify the model at a later time, all generated internal model parameters may be moved to the CANoe analysis windows by drag-and-drop from the model display window. No MATLAB/Simulink license is required for this display.

Application example: Simulation of an active chassis control system with a FlexRay bus

The design of an active chassis control system, including a vehicle model, serves as a challenging example. The vehicle model should be able to simulate vehicle dynamics sufficiently realistically and serve as a platform for the chassis control system with active dampers. The goal here is to use the control system to attain the most comfortable driving behavior possible. The overall model is subdivided into two main blocks here:

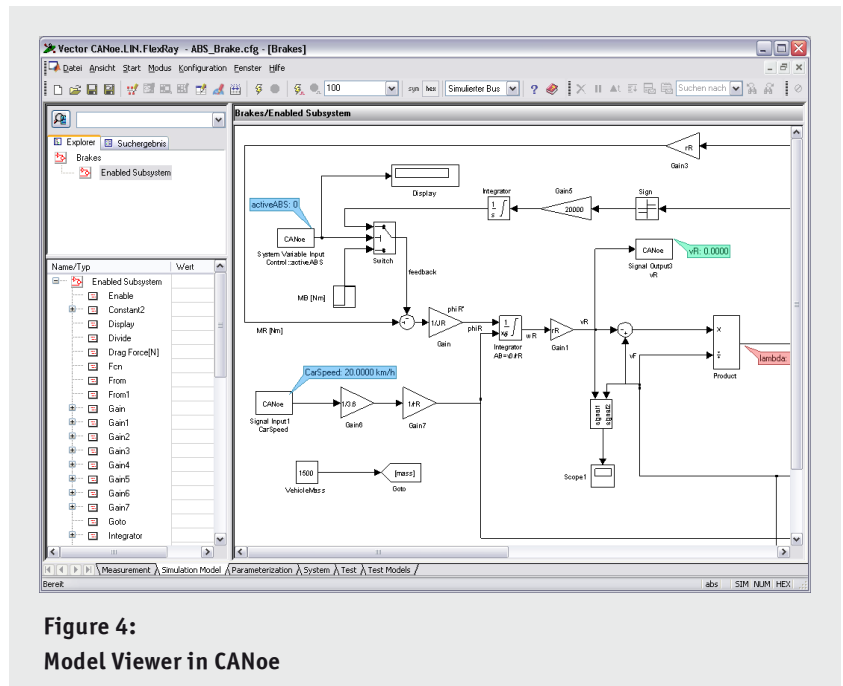


Figure 4:
Model Viewer in CANoe

- 1) Environment model (environment simulation for controllers), including:
 - > Multi-body model of the vehicle (mechanical vehicle model)
 - > Model of the powertrain (simplified engine model)
 - > Model of the brakes
 - > Four wheel models
 - > Road model
- 2) Chassis control system, including:
 - > Vehicle observer
 - > Main chassis controller
 - > Four subordinate power controllers for the active dampers

The environment model is subdivided into the vehicle model and the road model. The vehicle model is designed as a multi-body model with 15 degrees of freedom. The multi-body model is defined symbolically with the help of a computer algebra system, and motion equations are derived from this model (approx. 4,400 additions and approx. 20,800 multiplications). The vehicle model offers additional inputs for stimulation: Accelerator and brake pedal positions, engaged gear, steering angle, parking brake state and control target value (comfortable or sporty). These parameters are passed by system variables in CANoe. Driver inputs may be interactively set on a control panel with relevant controls. Suitable macro recordings/Enabled may also be used as driving profiles for automated playback. The road model is modeled by a lookup table containing the elevation of the road surface and surface normals. Similarly, the surface composition is described by a friction coefficient at all road locations.

Chassis control consists of a LQ controller with observer. The controller computes target forces for the active force elements in the

wheel suspensions. Essentially, two control system goals are set here:

- > Reduction in body accelerations. This results in greater comfort for the car driver
- > Maintenance of constant wheel contact zone (minimizing variations in wheel contact forces). Results for the car driver: Improved safety and vehicle dynamics.

The observer reconstructs the non-measurable vehicle states by applying a simplified linear vehicle model (with seven degrees of freedom). The chassis control system needs measurement parameter values of the angular accelerations of the car body's roll and pitch from two gyro sensors, as well as the car body's lifting acceleration from an accelerometer. The vehicle simulation makes these parameters available, and they are also passed to the chassis controller via system variables in CANoe. The force controller regulates forces of the active force elements, targeting values specified by the chassis control system. This might be designed as a simple PI control system for current regulation, for example.

In CANoe, six simulation nodes are defined (Figure 5): These are the "Environment simulation" node (Vehicle model), the "Chassis controller" node and four "Force control" nodes. The six nodes exchange data via a FlexRay bus on the four target forces for the force actuators and four deflections of the wheel suspensions. Periodic transmission of these signals over a bus represents discrete sampling with a dead time in the feedback control loop. Due to its indeterminate nature and its size, this often has a negative effect on control quality. Nonetheless, these signals can be transmitted

very reliably and without jitter using a FlexRay bus. It is also possible to transmit these signals at a high cycle rate (2.5 to 5 ms). This significantly improves control quality of the overall control system compared to a CAN bus.

Concluding Remarks

CANoe/Matlab integration enables simultaneous use of Simulink to model complex application behavior and integrate the relevant in-vehicle network within the same development process. Users can work in the familiar MATLAB/Simulink environment during development and do not need to concern themselves with network-specific details.

Solver

A Solver specifies the mathematical approximation method used to compute time-dependent variables in a model. Simulink provides Solver algorithms with variable step-width or fixed step-width. Solvers with variable step-width optimize the algorithm according to how quickly the data changes in the model. In an extreme case, if the Solver finds discontinuities then certain simulation time points are re-computed with a smaller step-width. On the other hand, large step-widths are used for times over which the model data does not vary greatly; this accelerates the simulation time. A Solver with a fixed step-width always computes with the specified step-width value. This type of Solver does not detect discontinuities.

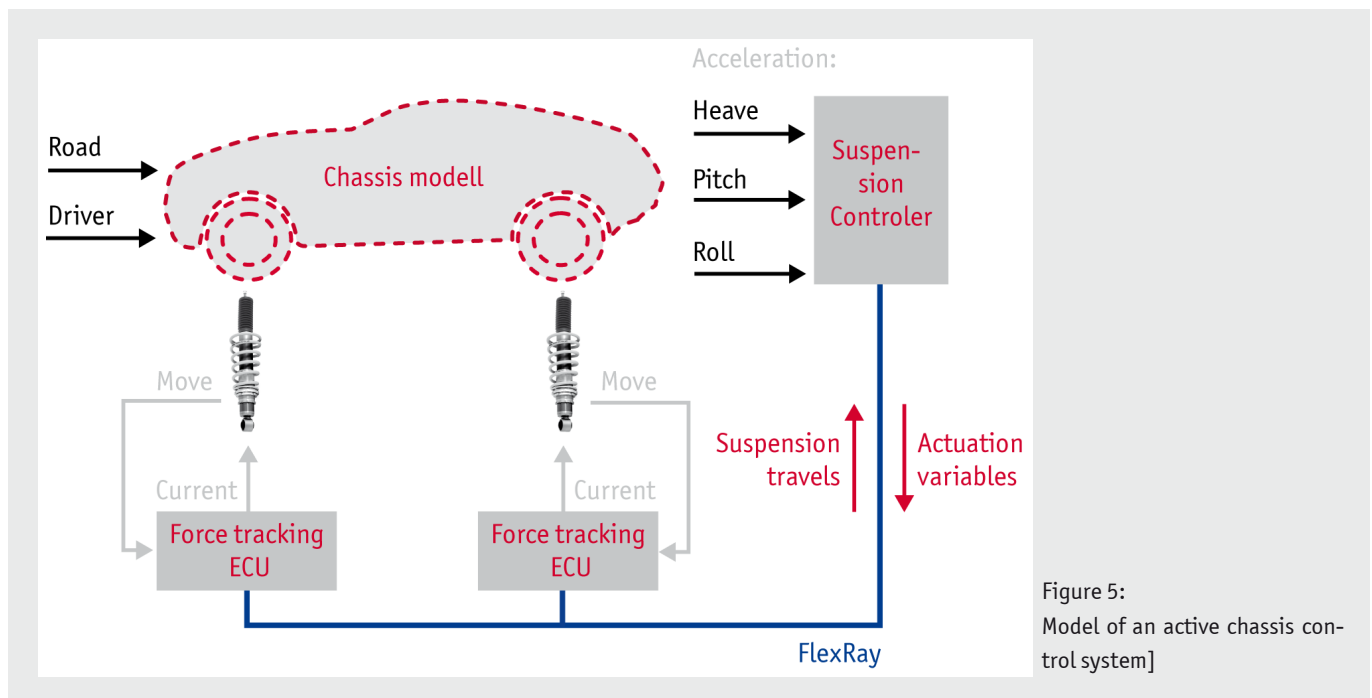


Figure 5: Model of an active chassis control system]

Selecting a Solver

Due to the interface to CANoe, and in consideration of later code generation, the user should choose Solvers with a fixed step-width, because Solvers with variable step-width exhibit the following limitations:

- > Solvers with variable step-width cannot be used for code generation. With these Solvers, it is impossible to predict when the next simulation step needs to be computed and how long it will take. This fact violates the real-time behavior that is sought on typical target platforms. This limitation applies in general and for all target platforms.
- > Since the Solver has no knowledge of the change in data at input and output blocks – unless they are non-deterministic bus signals – a Solver cannot optimize with variable step-width here.

Three different operating modes are available for code generation with MATLAB integration in CANoe: In “Offline mode” and “Synchronized mode” both software tools are active; the simulation is started from Simulink. In “Online mode” or “HIL mode”, however, code for CANoe is generated from the model.

Blockset

The individual blocks of the blockset are implemented as subsystems. They consist of a so-called S-Function, which implements the specific functionality. For the most part, an S-Function is user-specific code that implements an API provided by MATLAB/Simulink. This makes it rather easy for developers to extend MATLAB/Simulink functional features user-specifically.

Translation of a German publication in Elektronik automotive, March/2010

Links:

Homepage Vector: www.vector.com
 Product Information CANoe Matlab Integration: www.vector.com/vi_canoe_simulation_en.html



Jochen Neuffer

studied Information Technology at the University of Applied Science in Esslingen. Since 2002, he has been working at Vector Informatik GmbH where he is a Product Management Engineer in the Tools for Networks and Distributed Systems area.



Carsten Böke

studied computer science at the University of Paderborn. From 1995 to 2004 he was a scientific assistant at the Heinz Nixdorf Institute, working in the Parallel Systems Design area. Since 2004, he has been working as a Senior Software Development Engineer at Vector Informatik GmbH where he develops tools for bus analysis and bus simulation of FlexRay systems.

>> Your Contact:

Germany and all countries, not named below

Vector Informatik GmbH, Stuttgart, Germany, www.vector.com

France, Belgium, Luxembourg

Vector France, Paris, France, www.vector-france.com

Sweden, Denmark, Norway, Finland, Iceland

VecScan AB, Göteborg, Sweden, www.vector-scandinavia.com

Great Britain

Vector GB Ltd., Birmingham, United Kingdom, www.vector-gb.co.uk

USA, Canada, Mexico

Vector CANtech, Inc., Detroit, USA, www.vector-cantech.com

Japan

Vector Japan Co., Ltd., Tokyo, Japan, www.vector-japan.co.jp

Korea

Vector Korea IT Inc., Seoul, Republic of Korea, www.vector.kr

India

Vector Informatik India Prv. Ltd., Pune, India, www.vector.in

China

Vector Informatik GmbH Shanghai Representative Office, Shanghai, China, www.vector-china.com

E-Mail Contact

info@vector.com