



# AddOn MATLAB<sup>®</sup> Interface

## User Guide

Version 2.2

Vector Informatik GmbH, Ingersheimer Str. 24, D-70499 Stuttgart  
Tel. +49 711 80670-0, Fax +49 711 80670 555,  
Email [can@vector-informatik.de](mailto:can@vector-informatik.de), Internet <http://www.vector-informatik.de>

## Subsidiaries

### Vector CANtech, Inc.

Suite 550  
39500 Orchard Hill Place  
USA-Novi, Mi 48375

Tel.: +1 (248) 449 9290  
Fax: +1 (248) 449 9704

info@vector-cantech.com  
<http://www.vector-cantech.com>

### Vector France SAS

168, Boulevard Camélinat  
F-92240 Malakoff

Tel.: +33 (1) 4231 4000  
Fax: +33 (1) 4231 4009

information@vector-france.com  
<http://www.vector-france.com>

### Vector GB Limited

Rhodium  
Blythe Valley Park  
Solihull, Birmingham  
B90 8AS, UK

Tel.: +44 (0) 754 9001197

info@vector-gb.co.uk  
<http://www.vector-gb.co.uk>

### Vector Japan Co., Ltd.

Seafort Square Center Bld. 18F  
2-3-12, Higashi-shinagawa, Shinagawa-ku  
J-140-0002 Tokyo

Tel.: +81 3 5769 6970  
Fax: +81 3 5769 6975

info@vector-japan.co.jp  
<http://www.vector-japan.co.jp>

### Vector Korea IT Inc.

Daerung Post Tower III, 508  
Guro-dong, Guro-gu, 182-4  
Seoul, 152-790  
Republic of Korea

Tel.: +82(0)2 2028 0600  
Fax: +82(0)2 2028 0604

info@vector-korea.com  
<http://www.vector-korea.com/>

### VecScan AB

Theres Svenssons Gata 9  
SE-417 55 Göteborg

Tel.: +46 (31) 76476 00  
Fax: +46 (31) 76476 19

info@vecscan.com  
<http://www.vecscan.com/>

For Distributor Addresses please have a look on our website:

[www.vector-informatik.com](http://www.vector-informatik.com)

## International Quality Standard Certificate

The Quality/Process Management of Vector Informatik GmbH is being certified according to DIN EN ISO 9001:2000-12 (formerly DIN EN ISO 9001:1994-08) throughout since 1998-08-19.

## Typographic Conventions

<b>Note:</b>	Identifies important notes
•	Identifies enumerations (bullet items)
➔ '1.0 Introduction'	Identifies references to further chapters of this manual
<b>[OK]</b>	Notation for buttons in dialogs
<TAB>	Notation for keys on the computer keyboard
<Ctrl>+<Z>	Notation for keys of the computer keyboard which should be pressed simultaneously
<b>Add...</b> <b>File   File open...</b>	Notation for menu, command and dialog names
on message 0x100	Notation for MS-DOS syntax or program code

## Contents

<b>1</b>	<b>Introduction.....</b>	<b>1</b>
<b>2</b>	<b>Execution Modes .....</b>	<b>2</b>
2.1	Simulation Models .....	2
2.1.1	Offline Mode .....	2
2.1.2	Synchronized Mode.....	3
2.1.3	Hardware-In-The-Loop (HIL) Mode .....	3
2.2	Analysis Models .....	3
<b>3</b>	<b>Modeling Concepts.....</b>	<b>4</b>
3.1	Software Layers of an ECU.....	4
3.2	Interaction Layer .....	5
3.3	CANoe Interaction Layer .....	5
3.4	Application.....	6
3.5	Definition of Concepts .....	6
3.5.1	Middleware .....	6
3.5.2	Interaction Layer.....	6
3.5.3	Transport Layer .....	6
3.5.4	Network Management Layer.....	7
<b>4</b>	<b>Setup.....</b>	<b>8</b>
4.1	General .....	8
4.2	MATLAB / Compiler Versions .....	8
4.3	Visual C++ 2005 Express Edition.....	8
4.4	Verification.....	9
4.5	CANoe Interaction Layer .....	10
4.5.1	Database preparation .....	10
4.5.2	Configuration Setup from scratch .....	11
4.6	Customize a Matlab Simulink® Model for Interaction with CANoe .....	12
4.6.1	Simulink® Settings .....	12
4.7	Customize a Matlab Simulink® Model for Parameterization from CANoe.....	14
4.7.1	Simulink® Settings .....	14
4.7.2	CANoe Settings .....	16

4.8	Configuration and Test of CANoe Simulink® Blocks .....	18
4.8.1	Signal Block Configuration.....	18
4.8.2	Test of CANoe Blocks .....	20
4.9	Model Referencing .....	21
4.9.1	Overview.....	21
4.9.2	Use Case.....	22
4.9.3	Limitations .....	22
<b>5</b>	<b>Sample Configurations .....</b>	<b>24</b>
5.1	Offline Mode .....	24
5.2	Hardware-In-The-Loop (HIL) Mode .....	24
5.2.1	Signal Access Configuration .....	24
5.2.2	Sine Parameterization Configuration .....	25
5.3	Simple System Demo Configuration.....	26
5.4	StateFlowSystemDemo Sample Configuration.....	27
5.5	Trigger Sample Configuration.....	28
5.5.1	Triggering by Environment Input.....	29
5.5.2	Triggering by Environment Event.....	29
5.5.3	Using Different Targets .....	29
5.6	Analysis Configuration.....	29
<b>6</b>	<b>CANoe I/O Library.....</b>	<b>30</b>
6.1	Signal Input .....	31
6.2	Signal Output .....	31
6.3	Signal Block .....	31
6.4	Bus Block .....	33
6.5	Message Input.....	34
6.6	Message Output.....	35
6.7	Simulation Step .....	35
6.8	Analysis Step.....	36
6.9	CAPL Call.....	36
6.10	CAPL Handler .....	37
6.10.1	Environment Event .....	38
6.11	Environment Input .....	38
6.12	Data Environment Input.....	39

6.13 Environment Output .....	39
6.14 Data Environment Output .....	39
6.15 Environment Step .....	39
6.16 Environment Trigger .....	40
6.17 System Variable Input .....	41
6.18 System Variable Output .....	42
<b>7 CAPL Functions .....</b>	<b>43</b>
7.1 General .....	43
7.2 MODEL_init .....	43
7.3 MODEL_step .....	44
7.4 Parameterization Functions .....	44
7.4.1 MODEL_getParameter .....	45
7.4.2 MODEL_setParameter .....	45
7.4.3 MODEL_setScalarParameter .....	45
<b>8 Utilities .....</b>	<b>46</b>
8.1 Setup of signal and environment blocks .....	46
<b>9 Troubleshooting .....</b>	<b>47</b>
9.1 Unusable CANoe GUI during offline simulation .....	47
9.2 Unable to compile the <i>nlapml.cpp</i> source file .....	47
9.3 Spaces in MATLAB® installation path .....	47
9.4 Model Advisor Checks not available .....	48
9.5 High CPU load during simulation .....	48
<b>10 Index .....</b>	<b>49</b>

## 1 Introduction

This document describes the CANoe/MATLAB® interface. The CANoe/MATLAB® interface was designed to combine the strengths of both simulation tools.

CANoe is a total network simulation tool designed to both model nodes on a network and the network that binds them. Traditionally the functionality of these network nodes are modeled using the CAPL language provided with CANoe. The purpose of this interface is to extend CANoe's node modeling capability by adding the strength of the MATLAB®/ Simulink® environment.

Now models can be used to describe the functionality of a network node in CANoe. This allows CANoe to simulate nodes which are impossible or difficult to model using CAPL alone. Another feature would be to use a Simulink® model to generate an outside stimulus to an existing CAPL node.

---

**Note:** The Matlab Interface Version 2.2 requires CANoe 7.1 or newer. DLLs which are built using this interface are compatible to older versions as well.

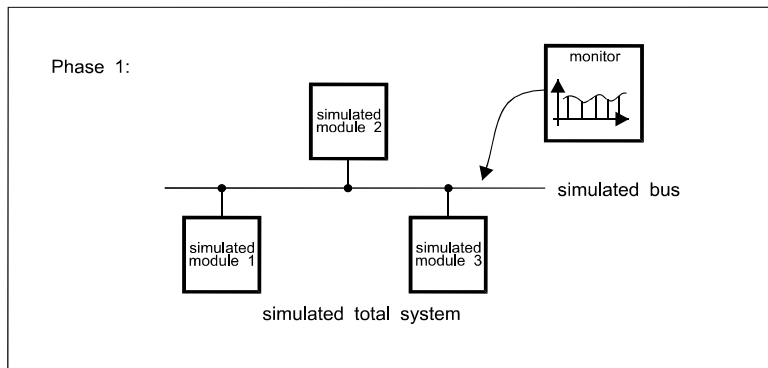
---

## 2 Execution Modes

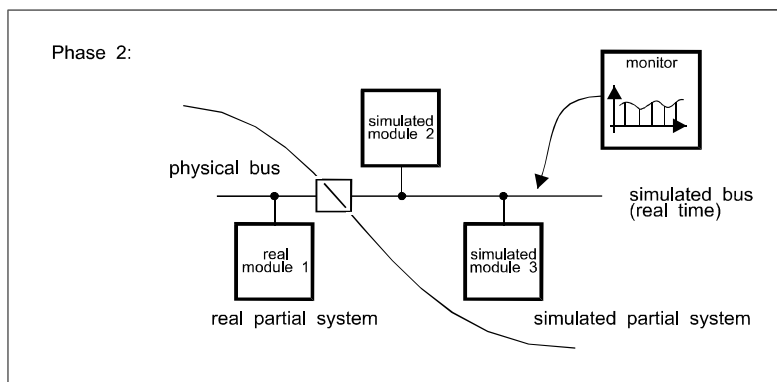
### 2.1 Simulation Models

Simulations utilizing this interface can be operated in three different modes:

1. Running a simulation in MATLAB®/ Simulink®.  
This is the offline simulation mode. It does not provide hardware access.



2. Running a simulation in MATLAB®/ Simulink® synchronized with CANoe.  
This is the synchronized simulation mode which provides real hardware access.



3. Running a simulation in the CANoe execution environment.  
This is the real-time or Hardware-In-The-Loop (HIL) simulation mode. It provides real hardware access and optimized runtime behavior.

#### 2.1.1 Offline Mode

In this mode the simulation is run in the MATLAB® Simulink® environment. CANoe is operated in Slave mode whereas MATLAB® Simulink® is the simulation Master. This is the mode for non-real-time simulation. The simulation is controlled from the MATLAB® Simulink® environment.

CANoe's Slave mode is a special simulation mode whereby CANoe is simulating the (CAN) bus and takes its measurement time base from outside (here: from Simulink®).



This is especially useful in early design stages. Debugging features of the MATLAB® environment can be used. Communication between MATLAB® Simulink® and CANoe takes place over Microsoft® DCOM for configuration and shared memory for data exchange.

You cannot perform real-time simulation with this kind of simulation.

### 2.1.2 Synchronized Mode

In this mode the simulation is run in the MATLAB® Simulink® environment. In contrast to the Offline Mode the CANoe time base is used in MATLAB® Simulink®. Therefore the simulation is run in almost real-time (typical resolution is about 1ms depending on the model). It is necessary that MATLAB® Simulink® computes the model faster than real-time in order to use this mode.

CANoe can either run in simulated mode or in real-time mode providing real hardware access. Debugging features of the MATLAB® environment might be used but synchronization will be lost as long as the model is paused.

This mode can be used for interaction with real (CAN) hardware devices. It is highly recommended using a multi core processor for best simulation results (please read 9.5: High CPU load during simulation). Communication between MATLAB® Simulink® and CANoe takes place over Microsoft® DCOM for configuration and shared memory for data exchange.

### 2.1.3 Hardware-In-The-Loop (HIL) Mode

In this mode the simulation is run in the CANoe execution environment. With the Simulink® Real-Time Workshop® you can target CANoe and produce a Windows® DLL which can be loaded in CANoe's simulation environment. One DLL must be generated per node. With this approach it is possible to test and verify your design with real (CAN) hardware in a networked environment. The simulation can be used as a simulation of the remainder of the bus in a real-time environment. Several ECUs can be simulated simultaneously within a single CANoe environment. This mode is recommended if it is necessary to run huge models and the remaining bus in real time. Requirements: Simulink® Real-Time Workshop®.

## 2.2 Analysis Models

The interface also supports models for analysis tasks like curve analysis or signal processing. In this case the model does not influence the simulation or measurement running in CANoe. Such a model cannot change the values of bus signals or environment variables. Instead the mathematical strength of Simulink® is used to monitor the results observed by CANoe. Analysis Models are simulated in two different ways:

1. CANoe starts the simulation using the COM Interface of MATLAB® Simulink®. This mode is comparable to the Synchronized Mode for simulation models except that the simulation is controlled by CANoe.
2. If the model is compiled using the Real-Time Workshop®, CANoe will load the model DLL when the measurement is started. This mode is comparable to the HIL Mode.

### 3 Modeling Concepts

It is important to understand how this interface communicates with CANoe in order to easily design a model.

It is easiest to think of the CANoe/MATLAB<sup>®</sup> interface as an extension or a complete replacement of existing CAPL code. The application of a CAPL node can be completely implemented with Simulink<sup>®</sup>.

It is possible to design an entire ECU in Simulink<sup>®</sup>. Examples of this can be found in Chapter 5 "Sample configurations".

For definition of concepts see "Definition of Concepts" in chapter 3.5.

Please continue with chapter 4 if you are familiar with Software Layers of an ECU.

#### 3.1 Software Layers of an ECU

CANoe's Simulation Setup lets you define the network nodes for a given (CAN) network. The functional behavior of these nodes can be described in CAPL (Communication Access Programming Language) which is a C style programming language. CAPL was designed to facilitate the modeling of a network node's basic logic (behavior). CAPL allows you to directly integrate the specification of the CAN communication matrix into the programming syntax.

#### Application Behavior

CAPL lets you define the basic application behavior of an ECU with special emphasis on its functional bus behavior.

#### Middleware

Apart from the application behavior of an ECU, there are parts of the software which are identical for each network node, such as interaction layers (IL), network management services (NM), diagnostic services and transport protocols (TP).

These layers are available as CANoe modeling libraries for all major OEMs.

NM and TP layers are available as CANoe DLLs; the IL is generated automatically as CAPL code. The IL and the modeling of the application are described in more detail in subsequent chapters.

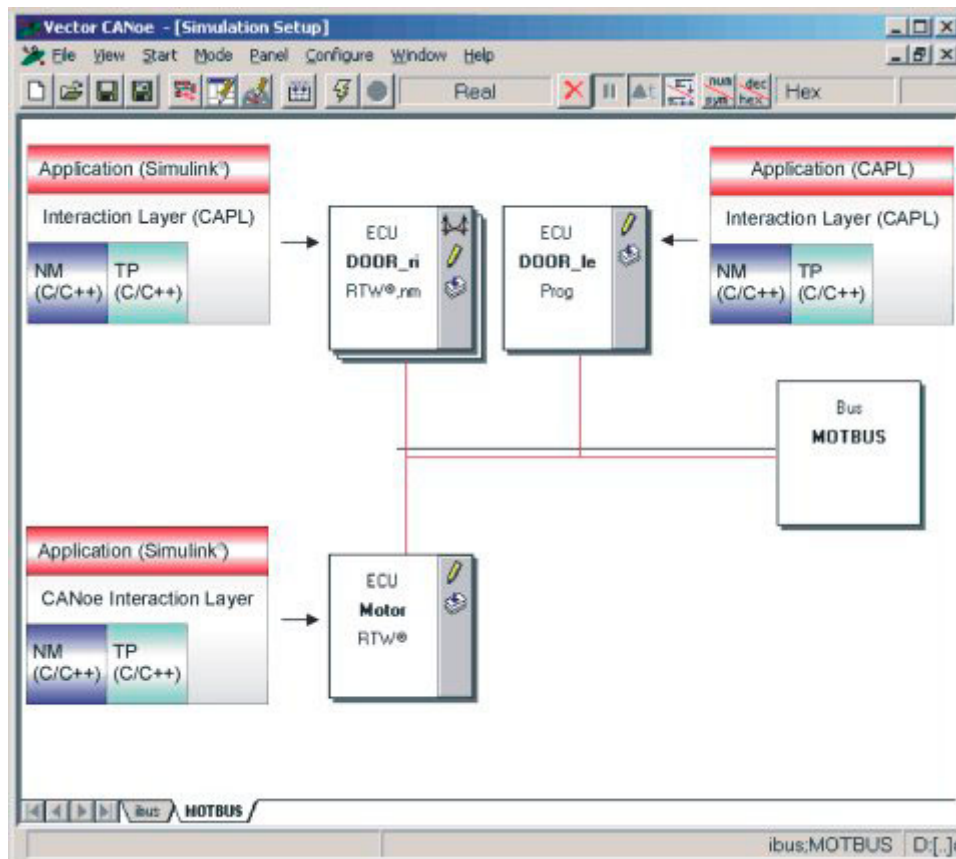


Figure 1: Software Layers

### 3.2 Interaction Layer

An important part of the software located between an ECU's application layer and lower-level functions is the Interaction Layer (IL) with its *Signal Interface*. It is important because applications deal with named signals (bus signals) instead of their networked representation in bits and bytes of the data stream. This Interaction Layer performs mapping between the signals and their network representations.

### 3.3 CANoe Interaction Layer

The Vector CANoe Interaction Layer (in short CANoeIL) provides a signal-oriented means of accessing the bus. The CANoeIL also performs mapping of signals to their send messages and controls the sending of these send messages as a function of the Send Model. Different Send Models are defined depending on the network type (OEM), and a special CANoeIL is provided to each of these Send Models.

It is possible to map model outputs and inputs directly to signals. This mapping causes the changed value of the model to be routed directly to the specific CANoeIL responsible for sending the value out and also the model's input is stimulated with the changed value.

### 3.4 Application

The application of an ECU has two major interfaces:

- The interface to bus signals
- The interface to peripheral I/Os like sensor and actuators. In CANoe the interface to I/Os is modeled using environment variables.

ECUs with simple application logic (e.g. some truth tables, simple conditional behavior, simple calculations) are usually modeled directly in CAPL.

ECUs with complex application behavior may be modeled with tools like MATLAB® Simulink® and Stateflow® using the signal input and output blocks as interfaces between the application and interaction layers. Environment variable input and output blocks are used as interfaces to the sensors and actuators. An additional advantage of this approach is that application models can be used to generate code for the real target, too.

### 3.5 Definition of Concepts

The following definitions and explanations are intended to describe the functionality of various components and their coordination with other modules/components.

#### 3.5.1 Middleware

Middleware is software that functions as a conversion or translation layer. It is also a consolidator and integrator. For decades custom-programmed middleware solutions have been developed to enable one application to communicate with another application that either runs on a different platform or comes from a different vendor or both. Today, a diverse group of products exists offering packaged middleware solutions.

#### 3.5.2 Interaction Layer

The Interaction Layer is a communication module responsible for separation of the low-level driver (a CAN Driver for example) which is dependent on the Data Link Layer and the application task which is independent of the underlying bus system.

Specifically the following items are handled by the Interaction Layer:

- Signal-based interface
- Sending of bus messages according to specified transmission types
- Monitoring of receiving messages
- Different notification types

The Interaction Layer is also responsible for transmitting and receiving all node-specific messages. Messages are sent according to predefined transmission types such as cyclic, event-triggered or a mixed mode specified in the network database.

#### 3.5.3 Transport Layer

Some information to be transmitted over the CAN bus does not fit into a single data frame because the data length exceeds 8 bytes. In such cases the sender has to split

up the data into several CAN messages with the same identifier. Additional information is necessary to re-assemble the data in the receiver.

This is performed by the Transport Layer:

- Segmentation and reassembly of data that are larger than the underlying Data Link Layer
- Flow control for single messages
- Error recognition

A Transport Layer is not only needed for diagnosis purposes but also for any large data which must be exchanged between different nodes, e.g. text information to be displayed on a dashboard.

### **3.5.4 Network Management Layer**

The availability of the CAN bus is handled by Network Management. Typical features include:

- Identifies the network configuration at start-up
- Monitors the network configuration while the bus system is running
- Synchronizes transition of all network nodes to bus sleep mode (power saving mode if CAN network is not needed)
- Controls peripheral hardware (CAN Controller and Bus Transceiver)
- Provides network-relevant status information
- Error recovery after bus-off

Each ECU is identified by a unique station number and has a special CAN message identifier for exchanging network-relevant information. This message contains identification of the transmitting node (encoded in the CAN identifier), the address of the receiving node as well as the message type and additional sleep flags.

## 4 Setup

This chapter describes the installation/setup process.

### 4.1 General

The CANoe/MATLAB® Interface requires the following minimum product versions:

- CANoe 7.1
- MATLAB® R12
- Visual C++® 6.0

CANoe/MATLAB® Interface files (canoe\\*.\*) are located in the following directory:  
\$(MATLABROOT)\rtw\c

<b>Note:</b>	After installation you should find the following path in your MATLAB® environment: \$(MATLABROOT)\rtw\c\canoe\devices
--------------	--

The sample files (CANoe configurations and Simulink® models) are located in the following directory:

\$(CANOE DEMOS)\Demo\_AddOn\Matlab

### 4.2 MATLAB / Compiler Versions

MATLAB® needs to be associated with your C compiler in order to generate the dll files for the HIL mode.

See your MATLAB®/RTW® documentation for a correct setup of the compiler environment. Typically the command 'mex – setup' must be run from the command window. The following compilers are currently supported:

Compiler	Supported since Matlab Version
Microsoft Visual C++® 6.0	R12
Microsoft Visual C++® .NET (7.1)	R14
Microsoft Visual C++® 2005 (8.0)	R2006a
Microsoft Visual C++® 2005 (8.0) Express Edition	R2007a
Microsoft Visual C++® 2008 (9.0) Professional and Express Edition	R2008b

### 4.3 Visual C++ 2005 Express Edition

The MATLAB® Interface can be used together with the free Visual C++ 2005 Express Edition. In this combination the download and installation of the latest Microsoft Windows Platform SDK is required (Microsoft® Windows Server® 2003 R2 Platform for example).

After installation a Windows environment variable named MSSdk must be set to the installation folder of the SDK. This can be done in the Windows Control Panel using the **System Properties** dialog. Choose the **Advanced** page and click on the **[Environment Variables]** button. In the **Environment Variables** dialog add a new system variable named MSSdk and set its value to the installation folder of the platform SDK.

#### 4.4 Verification

To verify proper installation open the Simulink<sup>®</sup> library browser. You should see an additional library: *Vector CANoe*.

When you open a model (or create a new one) select "RTW Options..." from the Tools menu. Under "Code generation" select the "Browse" button to open the "System Target File Browser". In this list you should see the following line:

*cn.tlc*                *Vector CANoe Real-Time Target DLL (Visual C++ Project Makefile)*

## 4.5 CANoe Interaction Layer

The CANoe Interaction Layer is used to access signals from Simulink® directly. Several preliminary steps have to be taken before the CANoe Interaction Layer can be utilized.

### 4.5.1 Database preparation

The database must contain several attributes to use it with the Vector Interaction Layer. You can import these attribute definitions from the Vector\_IL\_Basic template dbc file.

Database attributes to use the Vector Interaction Layer:

Object type	Database attribute	Description
Network	BusType	Defines the bus type, here 'CAN'
	NmBaseAddress	OSEK NM attribute
Node	NmStationAddress	OSEK NM attribute
Message	NmMessage	Identifies a NM message
	GenMsgStartDelayTime	Delay of cyclic sendings after the start of the Vector IL
	GenMsgSendType	Send type of the message
	GenMsgNrOfRepetition	Number of repetitions for send types 'OnWriteWithRepetition', 'OnChangeWithRepetition' and 'IfActiveWithRepetition'
	GenMsgILSupport	Defines the support of the Vector IL for this message
	GenMsgFastOnStart	Fast cyclic sending on start of the Vector IL
	GenMsgDelayTime	Minimum delay time between two sendings of the message
	GenMsgCycleTimeFast	Cycle time used by send types 'IfActive...' and for messages using attribute 'GenMsgFastOnStart'
	GenMsgCycleTime	Cycle time for send type 'cyclic'
Signal	NWM-WakeupAllowed	Allows a signal to wake up the network
	GenSigStartValue	Start value of the signal
	GenSigSendType	Send type of the signal
	GenSigInactiveValue	Inactive value of the signal – used by the send types 'IfActive...'

You can find more information about the Vector IL send types in the CANoe Online Help (*"CAPL Generator – Interaction Layer – Vector Interaction Layer"*)

Network management:

The communication between the network management and the Vector Interaction Layer doesn't need any CAPL code. If you don't want to use the OSEK NM please remove the entry "dmoseknm.dll" within the database attribute "NodeLayerModules".



All the greyed database attributes in the overview above can be removed from the database if there is no need to use the OSEK NM.

#### 4.5.2 Configuration Setup from scratch

Starting from a pre-configured dbc file you can automatically generate a CANoe configuration. Please take the following steps:

1. Prepare a new directory that shall contain the simulation configuration. There, create the folder 'candb' and copy your network database into this folder. The nodes will be generated to the folder "<Database-folder>/../Nodes". The panels will be generated to the folder "<Database-folder>/../Panels".
2. The exec32\ModelGeneration folder of your CANoe installation contains a Model Generation Wizard. A link can also be found in the CANoe start menu in the **Tools** folder. Start the wizard and select your network database. The output folder for the new configuration is set automatically. Choose **[Generate]** to create the configuration.

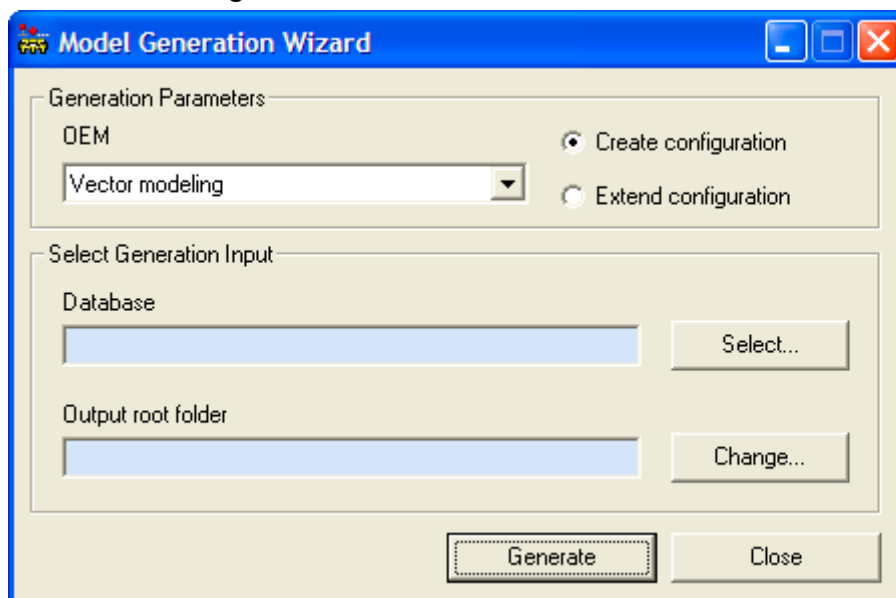


Figure 2: Model Generation Wizard

3. CANoe is started automatically. The Simulation creation runs about ½ up to 2 minutes, depending on your hardware configuration and the complexity of the model. You can track the status of the generation within the write window of CANoe.

---

**Note:** Please do not operate CANoe before the message "Vector Simulation creation finished." has occurred there.

---

#### Additional information for the automatic generation

- The original database will be saved as \*.bak file before the creation of the simulation model.

- The run of the automatic simulation creation requires the confirmation of the disclaimer of CANoe.

### Trouble-shooting of the automatic generation

If the automatic simulation creation fails, then please check the following topics:

- Start CANoe manually. Is it possible to start CANoe without the confirmation of dialogs, maybe error-dialogs?  
→ In this case please start CANoe manually, load a demo-configuration, and close CANoe.
- Is CANoe currently running? If yes, is the currently active configuration already saved?  
→ If the currently running configuration is not yet saved, then save it.
- Do you have multiple installations of CANoe onto your system?  
The generation must use the CANoe you have installed at last.

## 4.6 Customize a Matlab Simulink® Model for Interaction with CANoe

This chapter describes the steps which are necessary to run a Matlab model together with a CANoe configuration. It is recommendable to start CANoe and open the desired configuration at first. If a new configuration should be created consider using the Model Generation Wizard (→ '4.5.2 Configuration Setup from scratch'). The reason is that CAN signals or environment variables can be selected easily in the Simulink model if a CANdb is present.

It may be helpful to copy the Simulink® model file (\*.mdl) into the same folder as the CANoe configuration (\*.cfg). If you want to use the generated DLL, create a subfolder named Exec32 for example. The Real-Time Workshop® can be configured to copy the DLL into this folder.

### 4.6.1 Simulink® Settings

The CANoe interface for Matlab needs some settings of the Simulink® model which are described here. Choose **Simulation | Configuration Parameters...** to open the settings dialog.

#### Real-Time Workshop® Settings

The Real-Time Workshop® must be configured with the correct target for CANoe (cn.tlc). Otherwise the CANoe Matlab interface blocks can't be used.

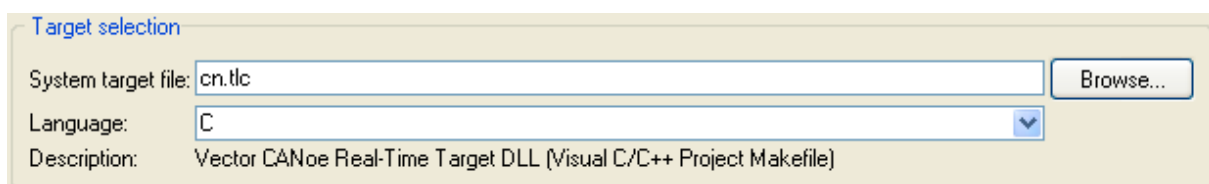


Figure 3: Target Selection

## Interface

If you want to parameterize the model from CANoe activate the C-API Interface for parameters. Please note that this setting is available with Matlab R14 and newer:

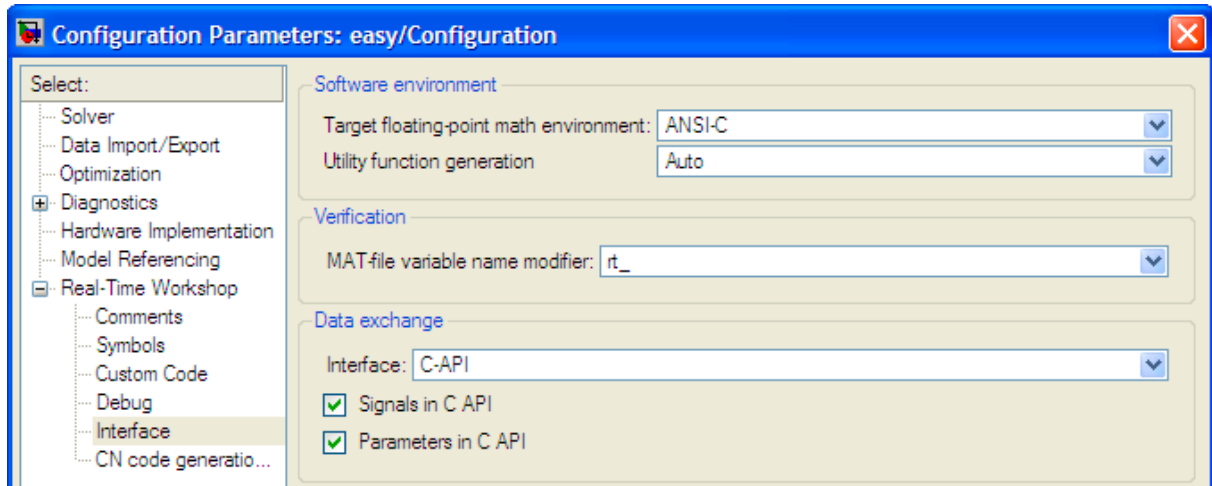


Figure 4: C-API Interface Settings

## CN code generation options

The CN target has some specific options for the code generation process. First of all the output directory can be set. If nothing is specified the DLL will be generated in the `..\MODEL_cn_rtw\release\` folder relative to the folder containing the Simulink® model. You can specify a subfolder of your configuration like Exec32 for example to generate the DLL here.

- **Export CAPL-Function:** With this option activated two special CAPL functions are exported by the generated DLL. Refer to ➔ '7.0 CAPL-Functions' for more details.
- **Enable Parameterization from CANoe:** If this option is activated, values of model parameters will be accessible from CANoe. This option requires "Export CAPL Functions" and activation of the C-API for parameters.
- **Enable Simulink Signal Analysis:** Enables analysis of all named Simulink® signals. With CANoe 7.0 or newer system variables are created for each signal below the namespace MODEL::Signals. This option requires the activation of the C-API for signals.
- **Generate Visual C++ project file:** The build process first builds a makefile. If this option is selected Visual Studio is opened. The compilation process now must be finished within Visual Studio. This is useful to generate a DLL with debug info. If this option is not selected, the DLL is built directly by a command line call. This applies only to Visual Studio 6.0.
- **Perform Simulation Step at t(0) :** If the model does not contain blocks with states (e.g. the model contains only algebraic blocks) this option can be selected to get a value for the simulation step at t=0.

- **Use full signal qualification for compilation:** CANoe signal blocks can use unique signal names. This option ensures compatibility of the compiled model to older versions of CANoe by using full signal name qualification (see 4.8.1).
- **Check CANoe block configuration at compilation:** Uses the Model Advisor to check settings of all CANoe blocks (see 4.8.2).
- **Generate CANoe Model Viewer description file:** Generates information required for viewing the Matlab model within CANoe. This feature is only supported for Matlab R14 and newer.

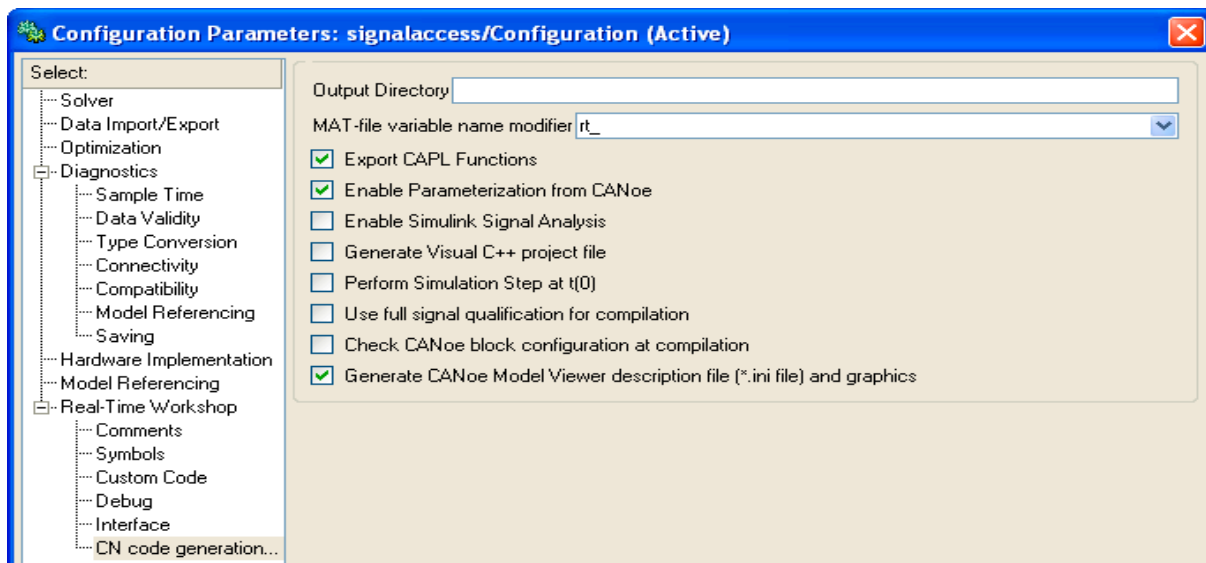


Figure 5: CN Code Generation Settings

**Note:** Only non-complex scalar or vector parameters are supported by the parameterization and signal analysis feature. Both make use of the C-API interface which is available since Matlab R14. Therefore these features are only available with R14 and newer.

## 4.7 Customize a Matlab Simulink® Model for Parameterization from CANoe

In this chapter all steps are described which are necessary for parameterization of a compiled model from CANoe. It is assumed that the model uses the CANoe target (cn.tlc).

### 4.7.1 Simulink® Settings

The CANoe interface for Matlab needs some settings of the Simulink® model which are described here. Choose **Simulation | Configuration Parameters...** to open the settings dialog.

## Optimization

The “Inline Parameters” option on the **Optimization** page defines which model parameters are accessible. If the option is deactivated, all parameters of every Simulink® block will be accessible. The number of parameters and their names might differ from the ones configurable in Simulink®. In most cases it won't be necessary to access all model parameters. Use workspace variables and assign them to block parameters. Activate the “Inline parameters” option and choose the **Configure...** button. Add all workspace variables which should be configurable after compilation. This is the recommended setting for optimal speed of the compiled model.

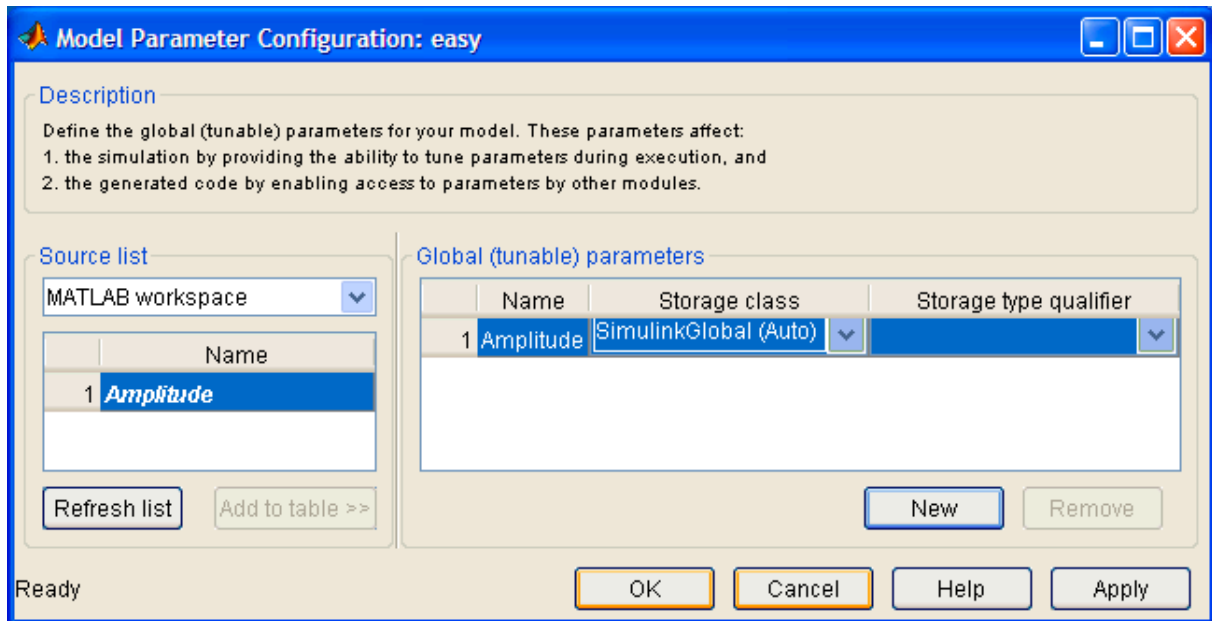


Figure 6: Model Parameter Configuration

## Real-Time Workshop® Settings

Select the C-API interface for data exchange on the “Interface” page:

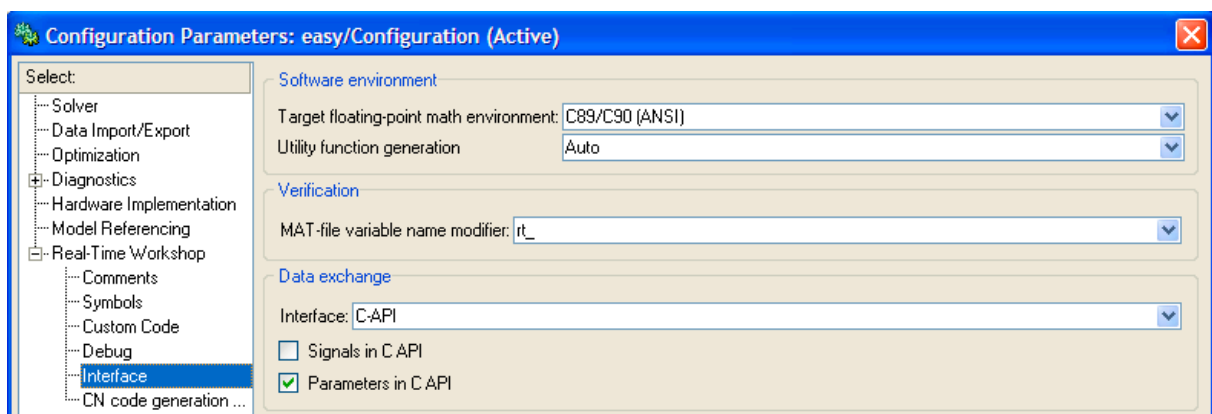


Figure 7: C-API Interface Settings

Enable the options “Export CAPL Functions” and “Enable Parameterization from CANoe” on the “CN code generation” page:

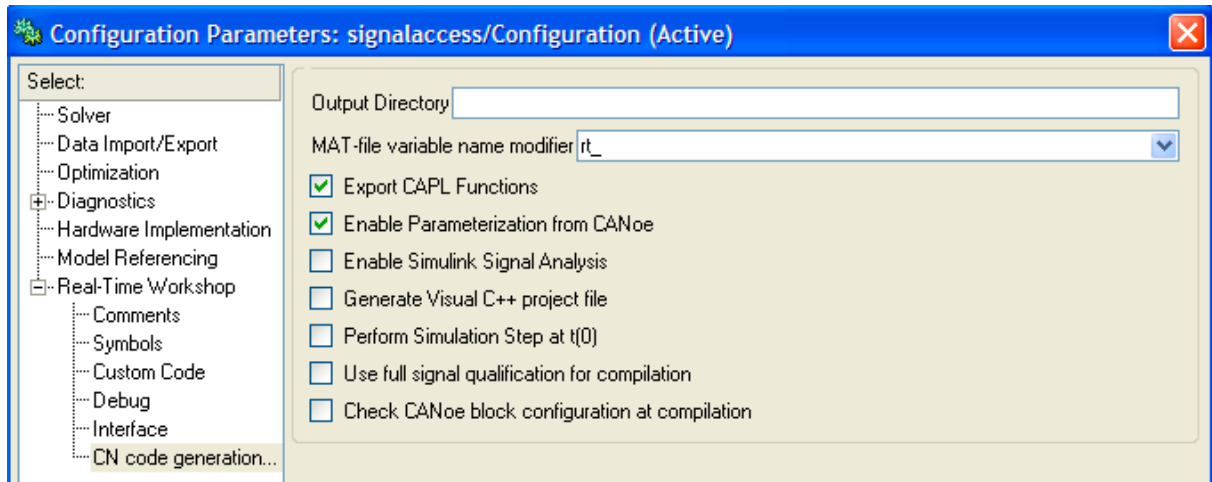


Figure 8: CN Code Generation Settings for Parameterization

Build the model with the new settings. The output folder will contain the model DLL and the parameterization files (named “model\_M.csv” and “model\_V.csv”).

**Note:** Only non-complex scalar or vector parameters are supported by the parameterization and signal analysis feature. Both make use of the C-API interface which is available since Matlab R14. Therefore these features are only available with R14 and newer.

#### 4.7.2 CANoe Settings

Using CANoe the model can be parameterized easily using system variables. The model DLL creates a system variable for each parameter as soon as it is added to the CANoe configuration. If you choose **Configuration | System Variables...** you will get an overview of all variables.

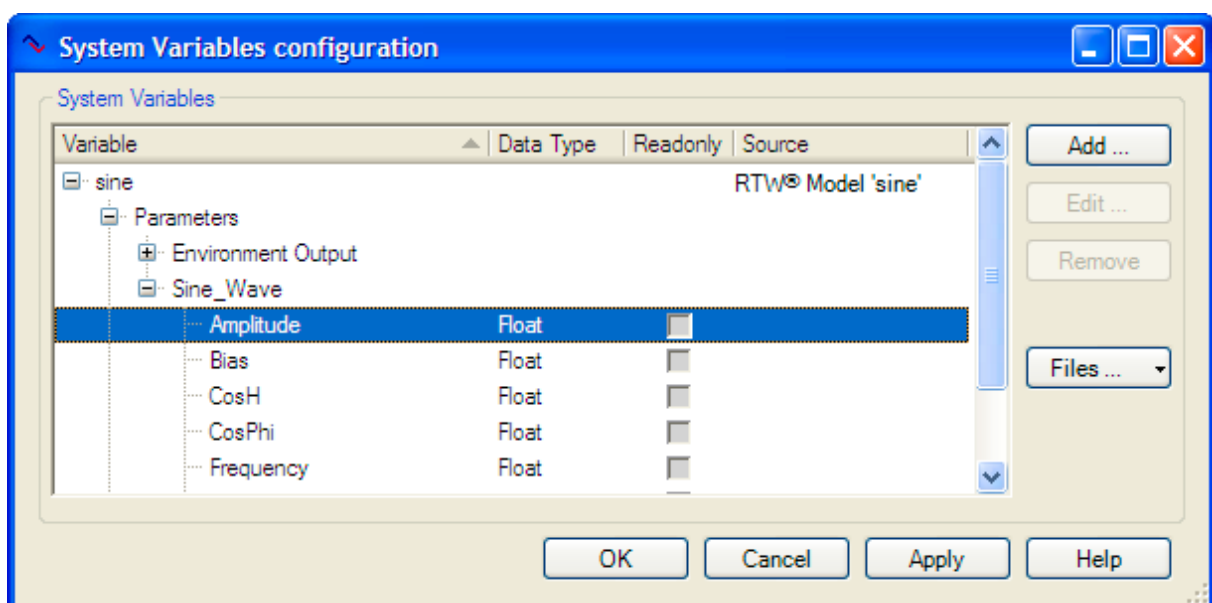
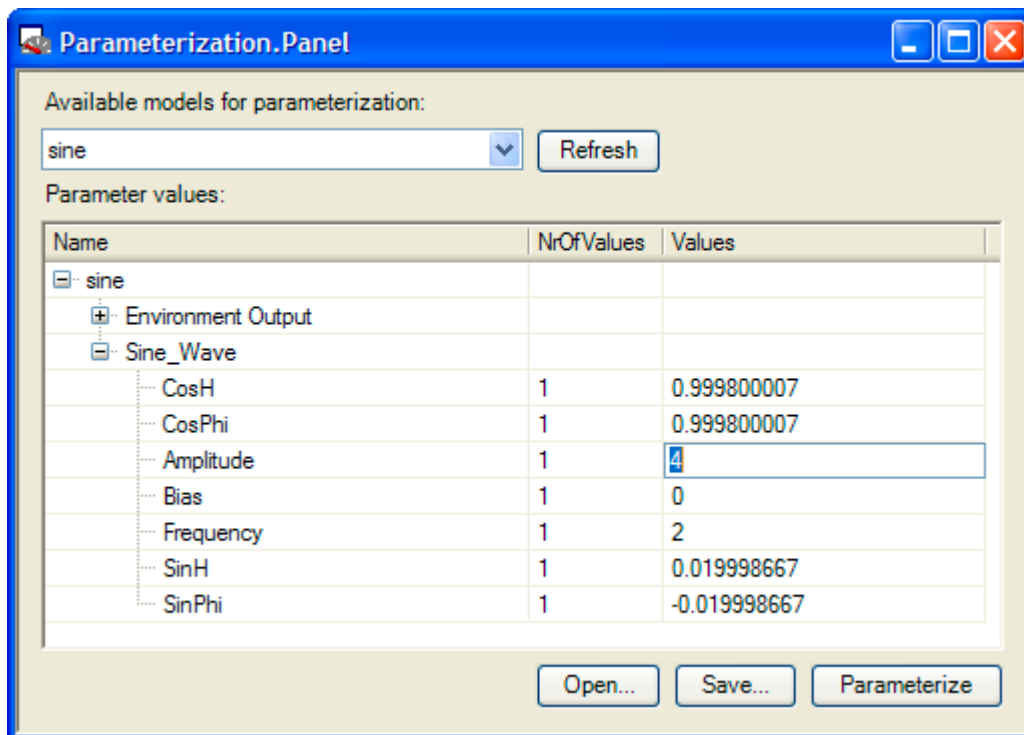


Figure 9: CANoe system variables dialog

The model can be parameterized by setting the values of these system variables. In the following sample CAPL code the amplitude of a sine block in the “Sine Parameterization” configuration will be changed:

```
@sysvar::sine::Parameters::Sine_Wave::Amplitude = 3;
```

In addition there is a .NET panel included to the Matlab Interface located in \$(MATLABROOT)\rtw\bin\canoe\parameterization\Parameterization.dll. It allows modifications of all system variables belonging to a model DLL. You can add the panel to your CANoe configuration using the dialog “Panel Configuration” from the **Configuration** menu.



The panel searches for system variables used for model parameterization if it is opened in CANoe or if the **[Refresh]** button is pressed. It is possible to update the currently loaded parameters with values of a value file (\*.V.csv) by pressing **[Open...]**. The button **[Save...]** allows writing parameter values back into a value file. If you press **[Parameterize]** all system variable values are written and the model uses the new values if possible.

**Note:** Some parameters might not be configurable during measurement. This depends on the implementation of a Simulink® block. Therefore it is recommended to change the system variables in the On PreStart section of a CAPL program or to restart the measurement after using the **[Parameterize]** button of the parameterization panel in order to make sure that all parameters are taken over completely.

You can also customize any other .NET project to read a value file and create the system variables. The parameterization panel uses the assembly ParamAccess.dll (also located in \$(MATLABROOT)\rtw\bin\canoe\parameterization) which must be included as reference. It contains the following classes:

Class name	Description
CANoeOperations	<p>Get name space names of all models which are currently loaded in CANoe:</p> <pre>static string[] GetModelNames()</pre> <p>Get and set parameters from CANoe:</p> <pre>static void GetParametersFromCANoe(string namespaceName, ModelParameters params, ParamChangeNotification changeNotification)</pre> <pre>static void ParameterizeCANoe(ModelParameters params)</pre>
FileOperations	<p>Reads and writes the parameterization files:</p> <pre>static void ReadParameters(string valueFilePath, ModelParameters params)</pre> <pre>static void WriteValueFile(string valueFilePath, ModelParameters params)</pre> <p>Please note that the model parameters must be loaded via <code>GetParametersFromCANoe</code> at first.</p>
ModelParameters	Contains all parameter values of a model. Note that the property "LastError" returns a string if any of the methods above failed. The property is null if the method succeeded.
ParamData	Contains the name and values of a single parameter.
ParamChangeNotification	<p>Delegate for a function called whenever a parameter changes its value:</p> <pre>void ParamChangeNotification(ParamData changedParam)</pre>

## 4.8 Configuration and Test of CANoe Simulink® Blocks

**Note:** This chapter only applies to Matlab R14 and newer.

### 4.8.1 Signal Block Configuration

CANoe signals can normally be identified by four parameters:

1. Database name
2. Node name (the node which sends the message)
3. Message name (the message containing the signal)
4. Signal name

By double clicking CANoe signal blocks a selection dialog appears containing all signals of the current CANoe configuration. The dialog also sets the parameters of the signal block so that it uses the currently selected signal.



The parameters 1. to 3. are optional. If the signal name is unique, database, node and message name are set to a space character (' '). If two messages contain signals with the same name, an additional message name will be set to resolve the ambiguity. This also applies to the node and database name. The use of unique signal names simplifies the development of models in early development stages where database and node name might change frequently.

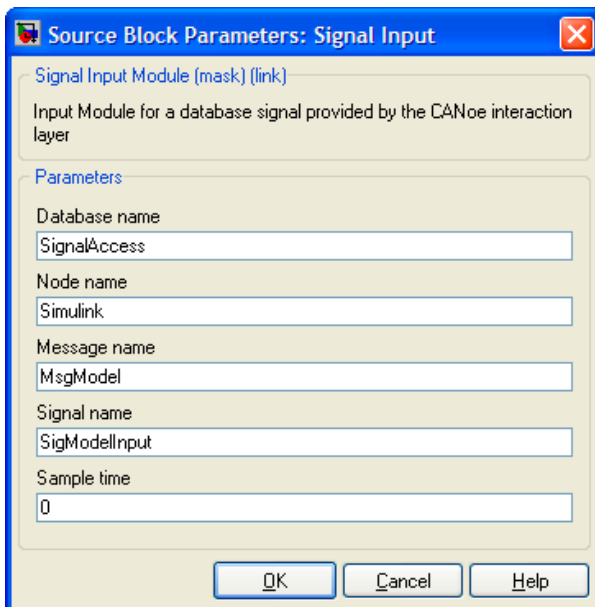


Figure 10: Full signal qualification

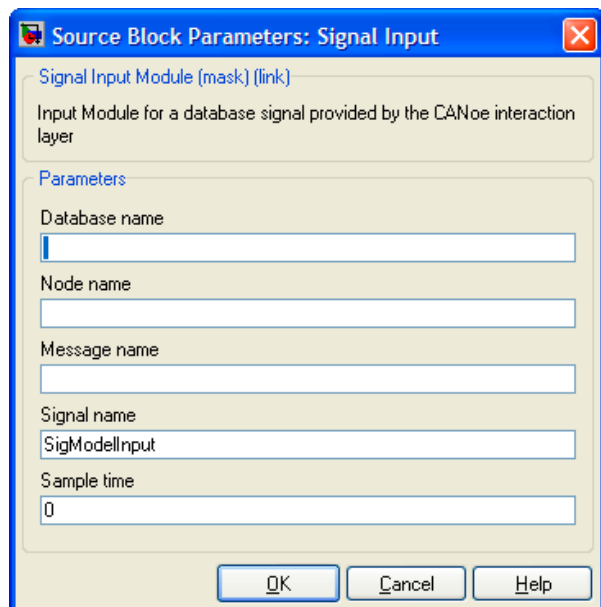


Figure 11: Unique signal name

**Note:** The use of unique signal names is not compatible with older versions of CANoe and of the CANoe Matlab® Interface. Therefore the following functions are provided to change the signal block settings to full signal qualification and back.

### VSignalQualification.m

This m-function allows switching signal blocks or a whole model to full signal qualification. Please make sure that CANoe has opened a configuration containing all databases used by the model. You can manually call the function from the Matlab® prompt:

```
VSignalQualification('model_name', 1); %full qualification  
VSignalQualification('model_name', 0); %use unique names
```

### Code Generation

Signal names must be fully qualified if the compiled model should work with versions of CANoe older than 7.0 SP3. This can easily be achieved by using the m-function described above or by checking the CN code generation option "Use full signal qualification for compilation":

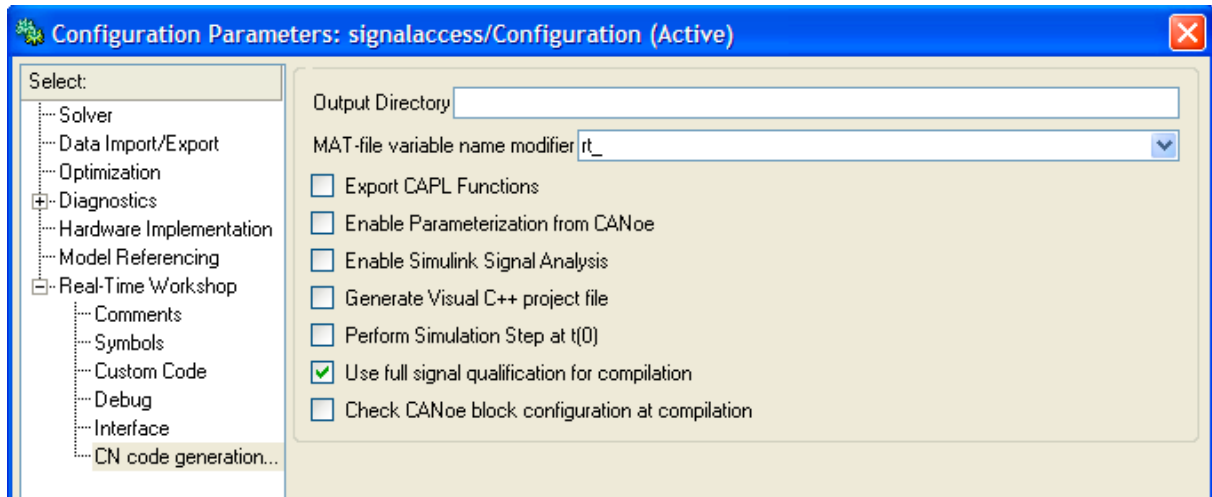


Figure 12: Signal Qualification Option

The VSignalQualification function is called twice during the compilation process if the option is enabled. It automatically switches to full signal qualification before the code is generated and sets back the unique names at the end.

#### 4.8.2 Test of CANoe Blocks

The CANoe Matlab® Interface provides custom checks for the Simulink® Model Advisor. Please make sure that CANoe has opened a configuration containing all databases used by the model before running these checks. The Model Advisor will report all CANoe blocks which are not configured correctly.

By following the links of erroneous blocks in the Model Advisor report, Simulink® opens the correct sub system and highlights the block. Using the Model Advisor it is very easy to find all CANoe blocks that won't work properly with the current CANoe configuration.

The Model Advisor checks can be run automatically during compilation if the CN code generation option "Check CANoe block configuration at compilation" is enabled.

**Note:** With Matlab® R14 this option just opens the Model Advisor and doesn't interrupt the compilation process if errors are detected. Using R2006a and newer the compilation process is stopped if the Model Advisor detects any erroneous CANoe block and opens the report. If all blocks are configured correctly the compilation is resumed.

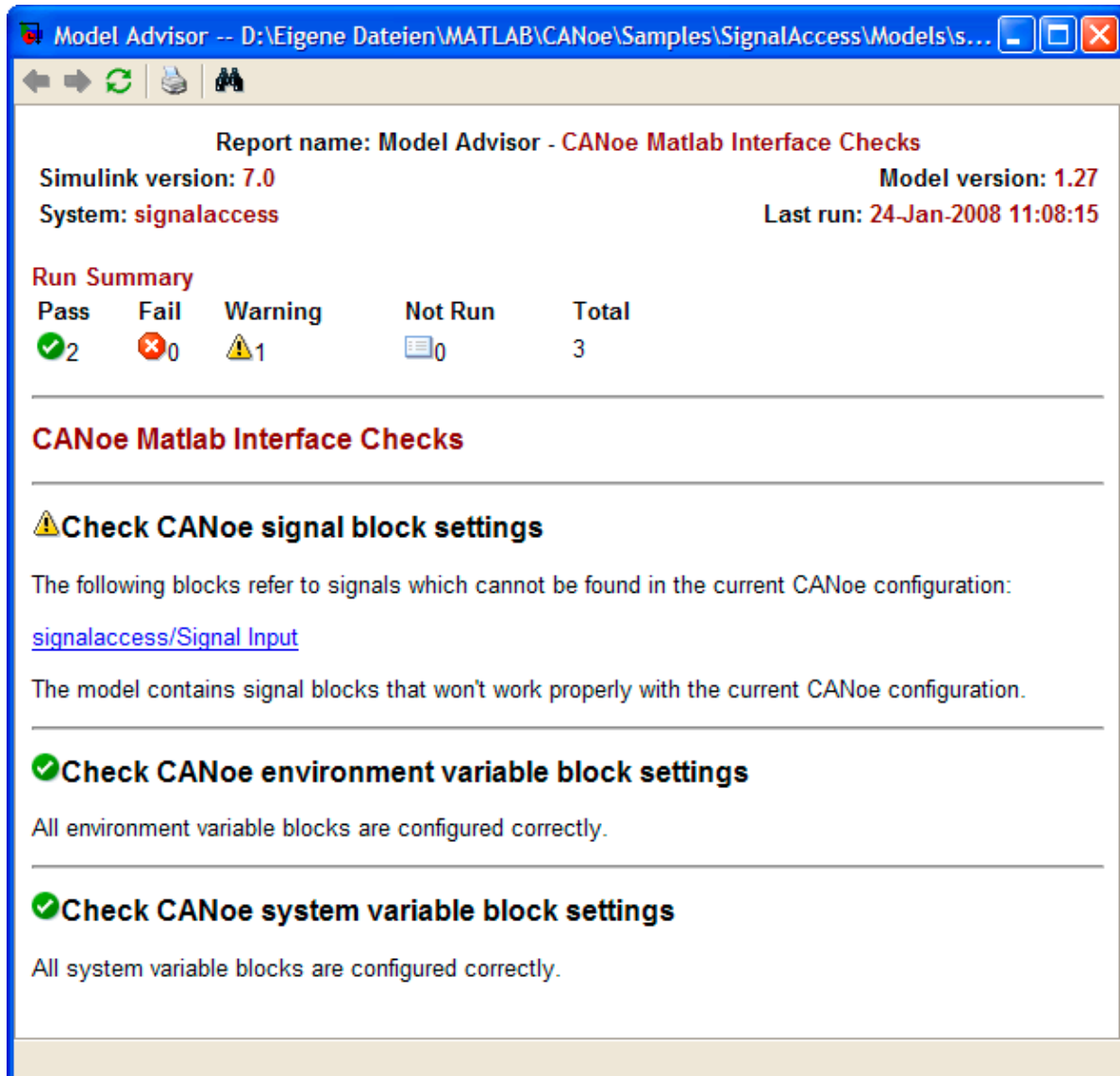


Figure 13: Model Advisor Report

## 4.9 Model Referencing

### 4.9.1 Overview

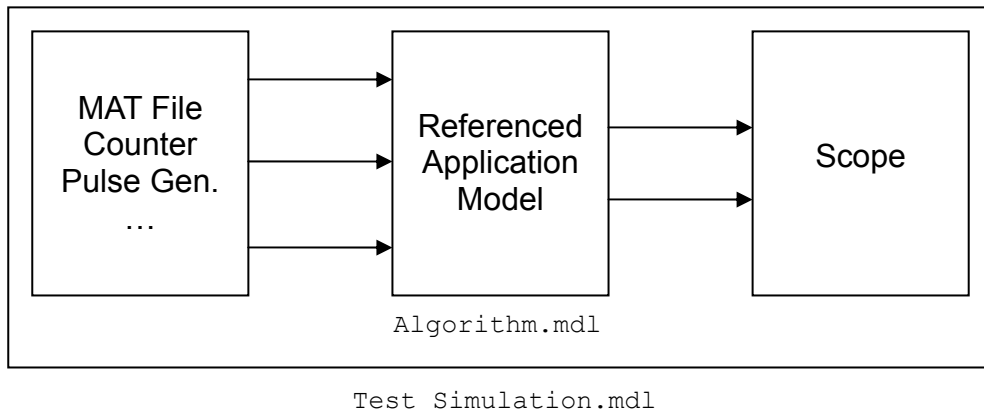
Using MATLAB® Version R14 or newer it is possible to use model references. Every model may contain one or more instances of a referenced model which in turn may also contain references to other models. The model block which realizes the referencing can be found in the Simulink® library.

This feature allows a modular development of Simulink models and might be used to separate CANoe input/output blocks from application blocks (see next chapter). The CANoe MATLAB® Interface supports Model Referencing but some restrictions apply.

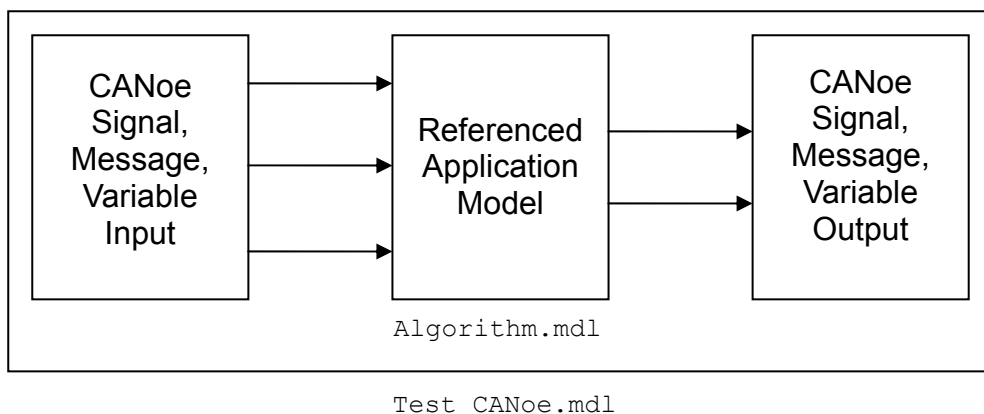
### 4.9.2 Use Case

The main focus of Mode Referencing is reusing code without the need to copy it. Several models can reference the same sub model and might use several instances of it. One aspect of this technique is that it allows separating the application code from code used for communication. Example:

A new algorithm is to be implemented for an ECU and it uses the CAN bus to exchange data with other ECU. In the first development phase the algorithm might be developed with a fixed set of test stimuli as inputs and scopes to validate the outputs:



For the next development step communication with the CAN bus is required. The CANoe MATLAB® Interface can be utilized to test the algorithm with either simulated bus communication or up to create a real-time prototype that communicates with real bus interfaces (→ '2 Execution Modes'). No modification of the algorithm itself is necessary if it is referenced by a model that uses the CANoe I/O library blocks:



In the last step the algorithm can be reference by a model which targets the real ECU hardware. Instead of the CANoe input and output blocks the ECU's communication interfaces must be used instead.

### 4.9.3 Limitations

Among the various points described in the MATLAB® help (see "Simulink Model Referencing Limitations") the following limitations apply when using Model Referencing with the CANoe MATLAB® Interface:

- The CANoe target file (CN.tlc) must be selected for all models (parent and sub models).
- The CN code generation options must not differ between the models.
- The parameterization feature must not be used.
- Models referencing other models need the option “Inline parameters” in the configuration settings to be enabled.
- CANoe I/O blocks must not be used in referenced models.

## 5 Sample Configurations

This chapter shows how to use the Matlab Interface in the context of various  
➔ '2.0 Execution Modes'.

### 5.1 Offline Mode

The sine sample configuration is provided for showing the usage of CANoe in Offline Mode. Within this mode CANoe's timing is controlled from Simulink.

The sine sample configuration shows the usage of the "Environment Out" and "Simulation Step" block. The "Environment Out" block maps the sine's output to the environment variable EnvFloat.

The "Simulation Step" block is required when running in Offline Mode. It is used to control CANoe's timing from Simulink®. This block may be left in models intended for Hardware-In-The-Loop mode but has no function; RTW will not generate any code for it.

- Open the CANoe configuration  
`$(MATLABROOT)\work\samples\Offline\Sine\sine.cfg`
- Start the measurement by clicking on the Start Simulation button within Simulink.

A sine wave should display in the CANoe Graphics Window. Notice that it runs for 100 seconds (simulation time) before stopping on its own. This stop time is defined in the Simulink® model under **simulation parameters | solver | stop time**. Offline mode does not run in real time and this configuration should run much faster since it is so basic.

### 5.2 Hardware-In-The-Loop (HIL) Mode

Two sample configurations are provided for showing the usage of CANoe as the execution environment for Simulink® models.

For using the model for the HIL Mode, Simulink® Real-Time Workshop® must compile the model into a DLL that CANoe can run. The generated DLL must then be added to the CANoe configuration.

#### 5.2.1 Signal Access Configuration

This CANoe configuration shows the usage of the CANoe "Signal Input" and "Signal Output" blocks for directly accessing signals. A sine wave is displayed in the CANoe Graphics Window.

Enter a value in the Simulink panel for signal SigModelInput in CANoe. This value is used as a multiplier for the amplitude of sine wave 2 in the model. The output of sine wave 2 is mapped to the signal SigResult and should now also be displayed in the CANoe Graphics Window.

- Open the Simulink® model  
`$(MATLABROOT)\work\samples\  
HIL\SignalAccess\SignalAccess.mdl`  
in MATLAB®, specify the Output Directory (Exec32) and build the DLL.

To execute the DLL within CANoe you must add it to the configuration. Since CANoe 5.1 you can add the DLL to a network node:

- Right click a network node in simulation setup and select **Configuration**.
- Open the “Modules” tab and choose **Add**.
- Select the DLL and choose **OK**.

Alternatively you can add the DLL via CANdb.

- To do so, open the database of your configuration within the CANdb Editor. The DLL will be attached to a node in the Simulation Setup.
- Select the list of nodes in the CANdb Editor. Here you can attach the DLL to an existing node or create a new one if desired.
- Select the node in the list and choose **Edit Node** in its context menu.
- Select the **Attributes** tab.

A dialog appears which lists all attributes defined for the selected node. Verify that there is a **NodeLayerModules** entry in the list. If there is no entry you must first define this kind of attribute in your database. This requires closing all opened dialogs and selecting **Attribute Definitions** from the **View** main menu. Select **Edit | New...** and define the following:

Name: NodeLayerModules  
Object Type: Node  
Value type: String  
Leave the "Default" editing box empty.  
Close the dialog.

Now you can edit this (node) attribute as described above.

Select the **NodeLayerModules** attribute in the attribute list of the desired node and edit its value.

- Type the name of the generated DLL.  
If the name of your Simulink model is "mymodel.mdl" you would type "mymodel.dll".
- Close all dialogs and save your database changes.

Start the measurement to view the model in action.

### 5.2.2 Sine Parameterization Configuration

---

**Note:** This sample configuration will only work with Matlab R14 and newer.

---

This CANoe configuration shows how Simulink® models can be parameterized after compilation with the Real-Time Workshop®. It is similar to the Sine configuration. The difference is that you can manipulate the parameters of the “Sine Wave” block in the Hardware-In-The-Loop (HIL) Mode. To do so, the following options must be set in Simulink **Simulation Parameters | Real-Time Workshop**.

- Choose the **Interface** page and select “C-API” for data exchange interface.
- Activate the “Parameters in C API” option.
- Choose the **CN code generation options** page and activate the options “Export CAPL Functions” and “Enable Parameterization from CANoe”.

The DLL will provide system variables and CAPL functions for accessing the parameters of the model (→ '7 CAPL Functions'). See the configuration comment of the sample for usage of these functions.

### 5.3 Simple System Demo Configuration

This configuration is derived from the CAN System Demo configuration that is supplied with CANoe. In addition to the original configuration the generated code of two Simulink® models has been added to the Simulation Setup.

The first model – engine.mdl – implements a simple vehicle speed control system. Vehicle and engine speed are the input variables and the speed is controlled as a function of the pedal position and gear. You will find this model in:

```
$(MATLABROOT)\work\samples\SimpleSystemDemo\Models\engine\engine.mdl
```

Open the model in MATLAB®. The model contains 5 CANoe specific blocks:

- Simulation step
- Actual speed
- Engine speed
- Pedal position
- Gear

The first block – Simulation step – is only relevant to the simulation in Offline and Synchronized Mode. The others are two Environment Input and Output blocks respectively. They read and write directly to the corresponding CANoe environment variables. Double click on the Environment blocks to see the actual controlled variable names.

The second model – door\_ri.mdl – lowers the window of the right door automatically to the lowest position when the user presses the “window down” button longer than a specified time. You will find the model in:

```
$(MATLABROOT)\work\samples\SimpleSystemDemo\Models\door_ri\door_ri.mdl
```

Generate code for the models as described in the “Signal Access” example above. For your convenience you will find all prebuilt DLLs in:

```
$(MATLABROOT)\work\samples\SimpleSystemDemo\Exec32
```

After you have generated code ...

- Start CANoe and open the configuration:  
\$(MATLABROOT)\work\samples\SimpleSystemDemo\SimpleSystemDemo.cfg
- Select **Compile all nodes** from the **Configure** menu.  
(**Configure | Compile all nodes**)  
To verify that CANoe has detected the two DLLs select **View | Simulation Setup** to have the Simulation Setup window pop up.  
The *Engine* and *DOOR\_ri* node should display a *RTW*® string in its second description line.
- Start the measurement to view the models in action.



- Change to the Engine Desktop and try the **[Brake]** button. Changes are shown immediately in the CANoe Graphics Window.
- Change to the Door Desktop and play with the window up and down buttons of the doors and watch the effects.

#### 5.4 StateFlowSystemDemo Sample Configuration

This configuration is derived from the CAN System Demo configuration that is supplied with CANoe. In addition to the original configuration the generated code of the Simulink® demo model “sf\_car.mdl” has been added. The original System Demo models the behavior of a car engine with CAPL. This part was replaced by the “Automatic Transmission Control” demo that comes with Simulink®. The model also shows the usage of a Stateflow subsystem.

You will find this model in:

```
$(MATLABROOT)\work\samples\  
StateFlowSystemDemo\Models\sf_car.mdl
```

It contains seven CANoe specific signal input/output blocks:

- BreakForce
- EngineRunning
- PedalPosition
- EngSpeed
- Gear
- CarSpeed
- Simulation step

The Simulation step block is only relevant to the Simulink® Simulation.

When you build the model a folder (sf\_car\_cn\_rtw) containing several files generated during the code building process is made in the current MATLAB® working directory. To specify a destination for the DLL file generated you must first specify the compiler's output directory. Select the **RTW Options...** item from the **Tools** menu and click the **[Options...]** button. In the editing box labeled **Output Directory** specify your directory:

```
$(MATLABROOT)\work\samples\StateFlowSystemDemo\Models\Exec32
```

<b>Note:</b>	If the directory path contains spaces (e.g.: C:\Program files\CANoe) you must specify the MS-DOS name (usually: C:\Progra~1\CANoe). Use the Explorer to find the MS-DOS name.
--------------	---

If nothing is specified the DLL will be generated in the `..\sf_car_cn_rtw\release\` directory and must be copied manually to `$(MATLABROOT)\work\samples\StateFlowSystemDemo\Models\Exec32` directory.

Additional settings in the Simulation Parameters dialog:

1. **Export CAPL-Function:** With this option activated two special CAPL functions are exported by the generated DLL. Refer to → '7.0 CAPL-Functions' for more details.
2. **Enable Parameterization from CANoe:** If this option is activated values of all model parameters are accessible from CANoe. Refer to → '5.2.2 Sine Parameterization Configuration' for an example.
3. **Enable Simulink Signal Analysis:** Enables analysis of all named Simulink® signals. With CANoe 7.0 or newer system variables are created for each signal below the namespace MODEL::Signals. This option requires the activation of the C-API for signals.
4. **Generate Visual C++ project file:** The build process first builds a makefile. If this option is selected Visual Studio is opened. The compilation process now must be finished within Visual Studio. This is useful to generate a DLL with debug info. If this option is not selected, the DLL is built directly by a command line call.
5. **Perform Simulation Step at t(0) :** If the model does not contain blocks with states (e.g. the model contains only algebraic blocks) this option can be selected to get a value for the simulation step at t=0.

After all settings are done...

- Close all dialogs and save your changes.
- Select **RTW Build** from the **Tools** menu.
- The DLL will be linked into your `Exec32` directory.
- Start CANoe and open the configuration:  
`$(MATLABROOT)\work\samples\  
StateFlowSystemDemo\StateFlowSystemDemo.cfg`
- Select **Compile all nodes** from the **Configure** menu.  
(**Configure | Compile all nodes**)  
To verify that CANoe has detected the DLL select **View | Simulation Setup** to have the Simulation Setup window pop up.  
The *Engine* node should display a *RTW®* string in its second description line although this depends on the order the nodelayer modules are defined in the CANdb.
- Start the measurement to view the models in action
- There is a replay block that controls the measurement. 20 seconds after measurement start, the pedal position moves to 100%. Now you can see the model's behavior.

## 5.5 Trigger Sample Configuration

This example shows how to use CANoe events to trigger Simulink® subsystems. Furthermore it introduces the concept of using a Simulink® library for the actual model and targeting of different hardware and/or software environments.

### 5.5.1 Triggering by Environment Input

The Simulink® model *trigsub1.mdl* uses an Environment Input block to trigger the subsystem. The subsystem trigger is configured as an “either edge” trigger.

### 5.5.2 Triggering by Environment Event

The Simulink® model *trigsub2.mdl* uses an Environment Event block to trigger the subsystem. The subsystem trigger is configured as a “function-call” trigger. In this simple example the actual value of the Environment Event block variable is not used and it is therefore connected to a terminator.

### 5.5.3 Using Different Targets

The Simulink® models *trigsub3.mdl* and *trigsub4.mdl* use a subsystem which is defined in the *trigsub.mdl* Simulink® library. The *trigsub3.mdl* model targets CANoe whereas the *trigsub4.mdl* model targets a VxWorks environment. Using a Simulink® library for the actual model (here: *Triggered Subsystem*) makes it easy for you to target different environments while providing a single point of maintenance for your application model.

## 5.6 Analysis Configuration

This configuration shows how MATLAB Simulink models can be used for analysis tasks. In Online mode the CAPL node Traffic outputs a message to the CAN bus which contains a signal named Random. The Simulink model MovingAverage uses the signal Random as input value and calculates a moving average as well as the minimum and maximum values. The calculation results are stored into system variables. It also performs a range check of the input signal using the system variables "UpperBound" and "LowerBound".

The Simulink® model is compiled as DLL. If no DLL is provided, CANoe starts the simulation of an analysis model in Simulink directly via COM. Analysis models do not affect the simulation. They can only modify system variables which exist in the measurement setup of CANoe. Analysis models can be used in the Offline mode, too.

## 6 CANoe I/O Library

The CANoe I/O library is a collection of S-Functions blocks.

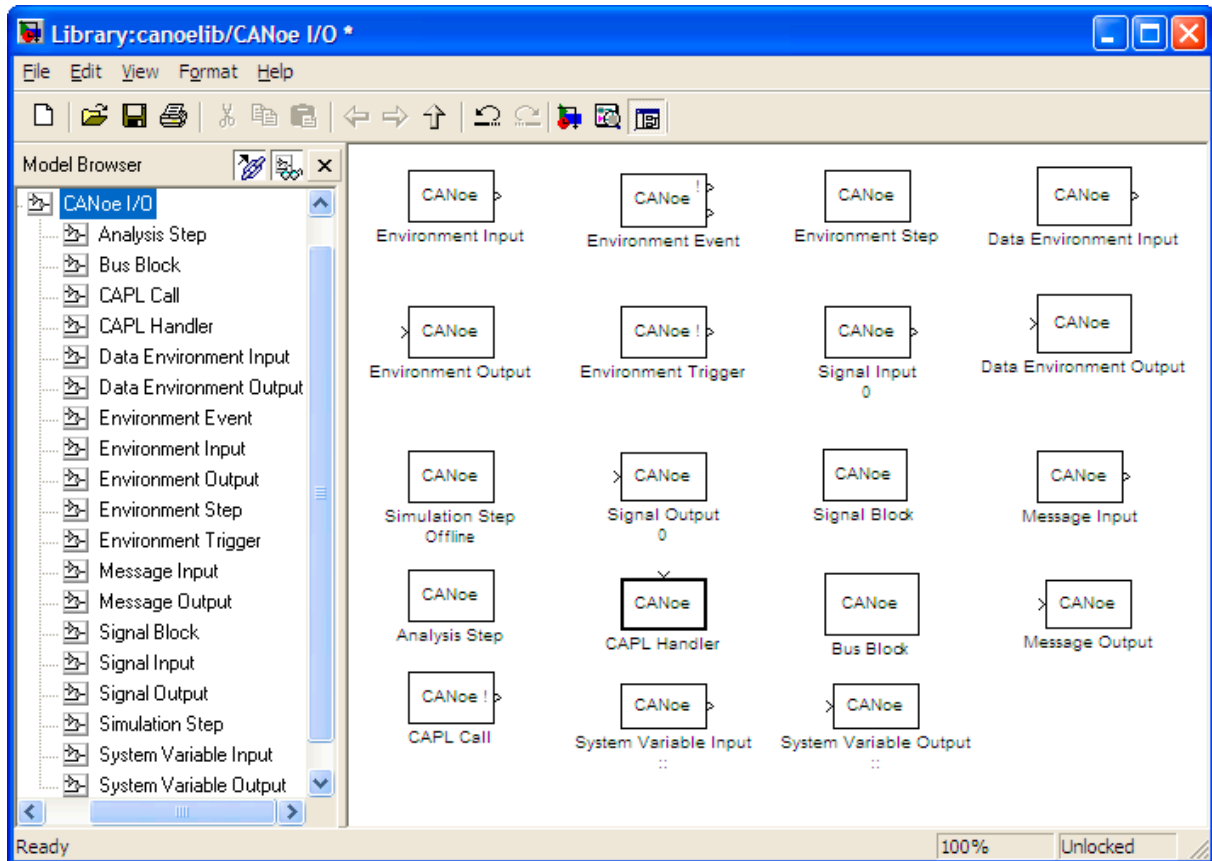


Figure 14: CANoe I/O library

The library provides the following features:

- Data I/O
- Trigger sources
- Event sinks
- Model execution control
- CANoe simulation control

## 6.1 Signal Input

The *Signal Input* block is the basic input block for a database signal. It is configured with the signal name, message name, node name and database name. It contains the additional parameter “Sample Time”. According to Simulink® semantics:

0 = continuous sample time

-1 = inherited sample time

Continuous sample time is set as default. If the *Signal Input* block is used in a triggered subsystem for example, the sample time must be set to “inherited” because only constant blocks or blocks with an inherited sample time are allowed in triggered subsystems.

The access to signals is provided through the Vector Interaction Layer. Please refer to 4.5 CANoe Interaction Layer for information about preliminary steps.

For further information on the CANoe Interaction Layer see CANoe online help.

Compatibility:	HIL	Offline/Synchronized
	X	X

## 6.2 Signal Output

The *Signal Output* block is the basic output block for a database signal. It is configured with the signal name, message name, node name and database name.

The access to signals is provided through the Vector CANoe Interaction Layer. Please refer to 4.5 CANoe Interaction Layer for information about preliminary steps.

For further information on the CANoe Interaction Layer see CANoe online help.

Compatibility:	HIL	Offline/Synchronized	
	X	X	(Simulation models only)

## 6.3 Signal Block

This block allows the user to collect a number of *Signal Input* and *Signal Output* block into one subsystem and configure them by one step. This block consists of an empty subsystem which can be configured by a Matlab® GUI.

Add Signal Output...	Adds <i>Signal Output</i> blocks to the subsystem
Add Signal Input...	Adds <i>Signal Input</i> blocks to the subsystem
Delete	Removes the selected signal

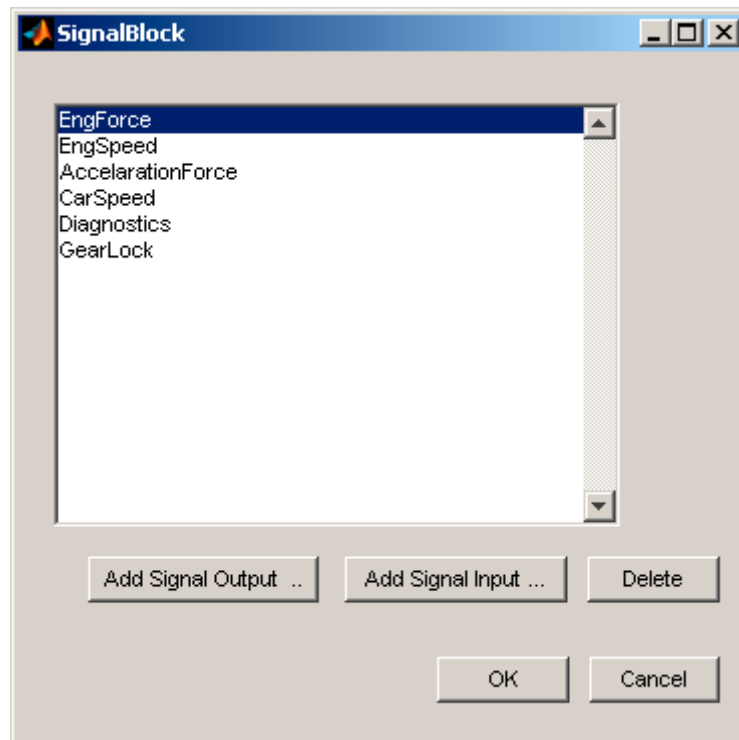


Figure 15: Configuration of the Signal Block

The example above results in the following *Signal Block*:

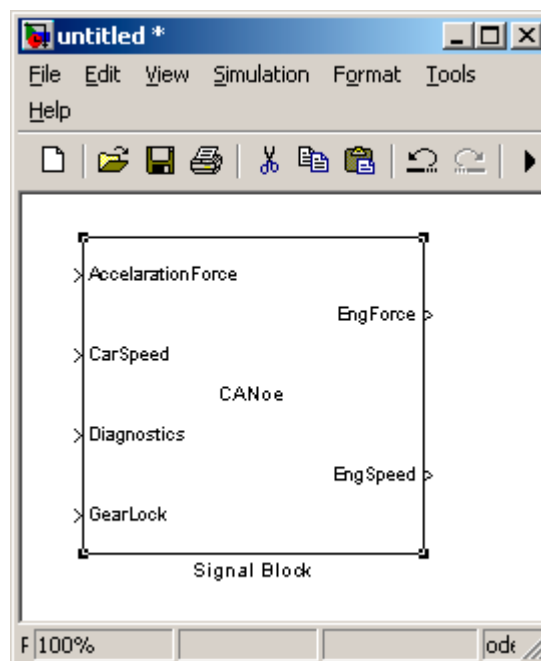


Figure 16: Configured Signal Block

Under the hood Signal Input and Signal Output blocks are created and configured with database name, node name, message name and signal name.

Compatibility:	HIL X	Offline/Synchronized X	Matlab R14 and later
----------------	----------	---------------------------	----------------------

## 6.4 Bus Block

This block allows the user to connect to a Simulink® signal bus. This eases the handling of many input and/or output signal. A Matlab® GUI supports the assignment of a Simulink® bus signal to a specific CANoe signal.

The bus block is a subsystem containing a Simulink® bus creator and/or bus selector. By the usage of these blocks the bus signals are mapped to *Signal Input* and/or *Signal Output* blocks within the subsystem.

Add Signal Output...	Adds <i>Signal Output</i> blocks to the subsystem
Add Signal Input...	Adds <i>Signal Input</i> blocks to the subsystem
Delete All	Removes all signals

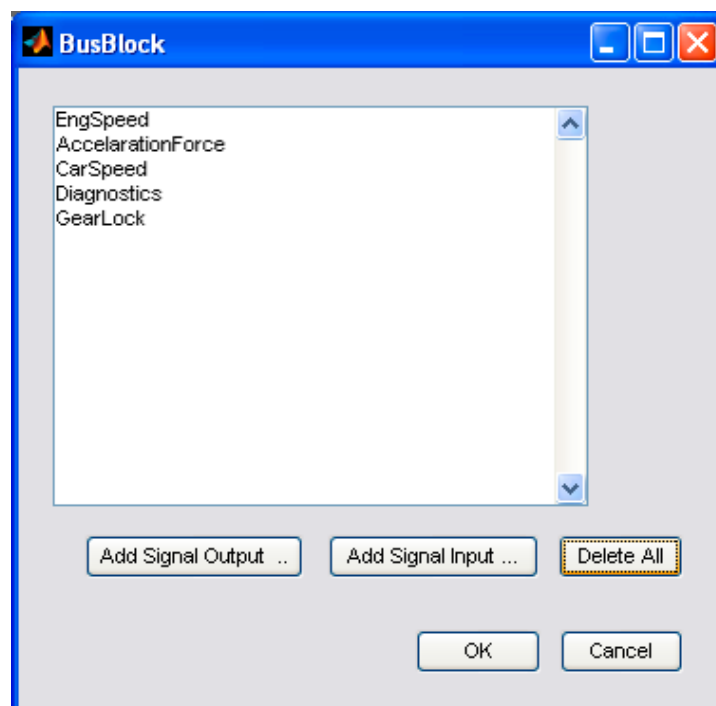


Figure 17: Configuration of a Bus Block

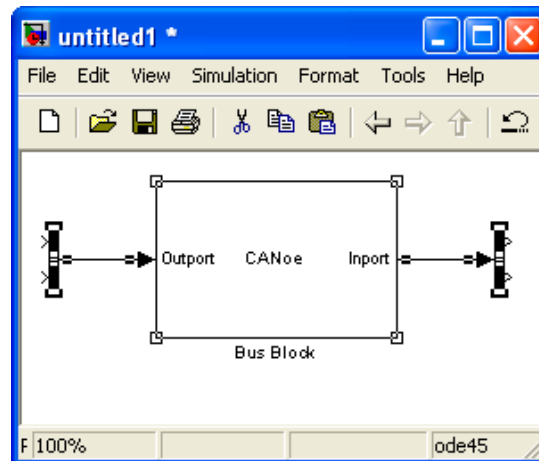


Figure 18: Configured Bus Block

Compatibility:	HIL X	Offline/Synchronized X	Matlab R14 and later
----------------	----------	---------------------------	----------------------

## 6.5 Message Input

The *Message Input* block serves as input block for all signals of a database message. The values of the signals are output as a vector from the block. Double click the block to select a message. The block will be configured with the message name, node name and database name. It contains the additional parameter “Sample Time” accessible from the dialog “Mask Parameters”. According to Simulink® semantics:

0 = continuous sample time  
-1 = inherited sample time

Continuous sample time is set as default. If the *Message Input* block is used in a triggered subsystem for example, the sample time must be set to “inherited” because only constant blocks or blocks with an inherited sample time are allowed in triggered subsystems.

The access to the signals of a message is provided through the Vector Interaction Layer. Please refer to 4.5 CANoe Interaction Layer for information about preliminary steps.

For further information on the CANoe Interaction Layer see CANoe online help.

Compatibility:	HIL X	Offline/Synchronized X	Matlab R14 and later
----------------	----------	---------------------------	----------------------



## 6.6 Message Output

The *Message Output* block serves as output block for all signals of a database message. It receives the values of the signals as a vector. Double click the block to select a message. The block will be configured with the message name, node name and database name.

The access to signals of the message is provided through the Vector Interaction Layer. Please refer to 4.5 CANoe Interaction Layer for information about preliminary steps.

For further information on the CANoe Interaction Layer see CANoe online help.

Compatibility:	HIL X	Offline/Synchronized X	Matlab R14 and later (Simulation models only)
----------------	----------	---------------------------	--

## 6.7 Simulation Step

The Simulation Step block is used for synchronization and data exchange between CANoe and Simulink®. You must place this block in your model if you wish to run a simulation within Simulink® (Offline and Synchronized mode). Please read 9.5 when using Synchronized mode.

When used within RTW® for HIL mode this block performs nothing.

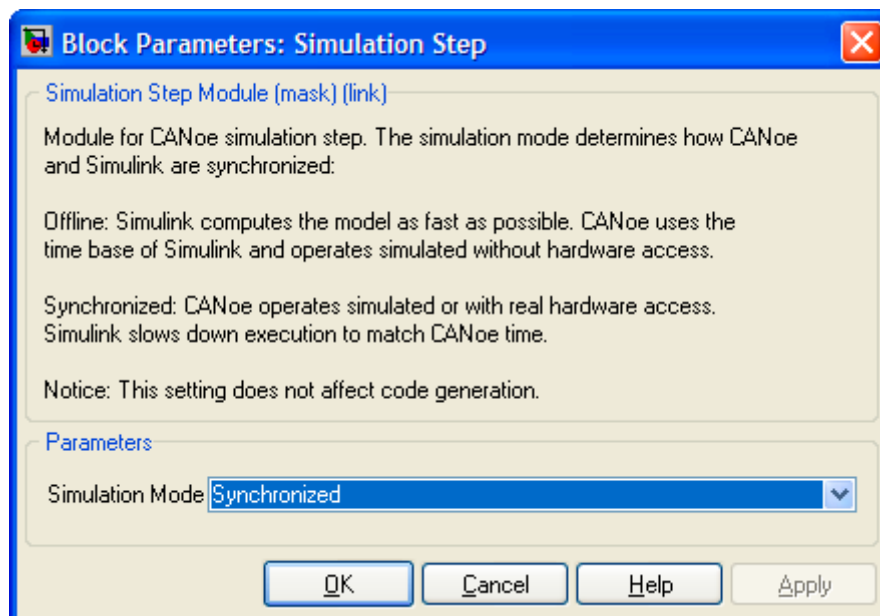


Figure 19: Simulation Step Configuration

Compatibility:	HIL (Not Required)	Offline/Synchronized X (Required)	(Simulation models only)
----------------	-----------------------	--------------------------------------	--------------------------

## 6.8 Analysis Step

The Analysis Step block is used for synchronization and data exchange between CANoe and Simulink®. You must place this block in your model if the model should be used for analysis purposes. CANoe will start the simulation in Simulink® when the model is configured in the **Simulink Analysis Model** settings and the measurement is started.

Compatibility:	HIL	Offline/Synchronized	
	X (Required)	X (Required)	(Analysis models only)

Please note that no outputs except the System Variable Output blocks can be used together with this block.

## 6.9 CAPL Call

The *CAPL Call* block can be used to trigger a Simulink® subsystem from CAPL.

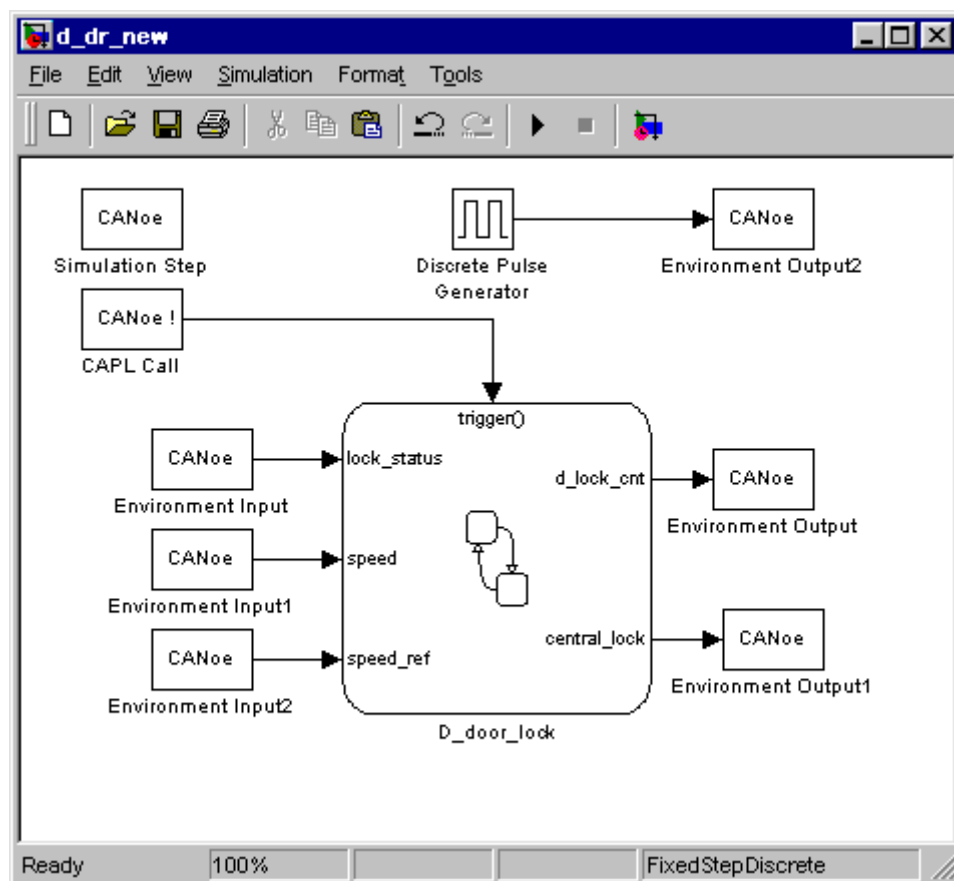


Figure 20: CAPL Call block

The *CAPL Call* block is configured with a function name which is exported by the RTW® generated CANoe DLL. Function parameters are not supported.

Compatibility:	HIL X	Offline/Synchronized —	(Simulation models only)
----------------	----------	---------------------------	--------------------------

## 6.10 CAPL Handler

The *CAPL Handler* block can be used to handle a Simulink® event in CAPL.

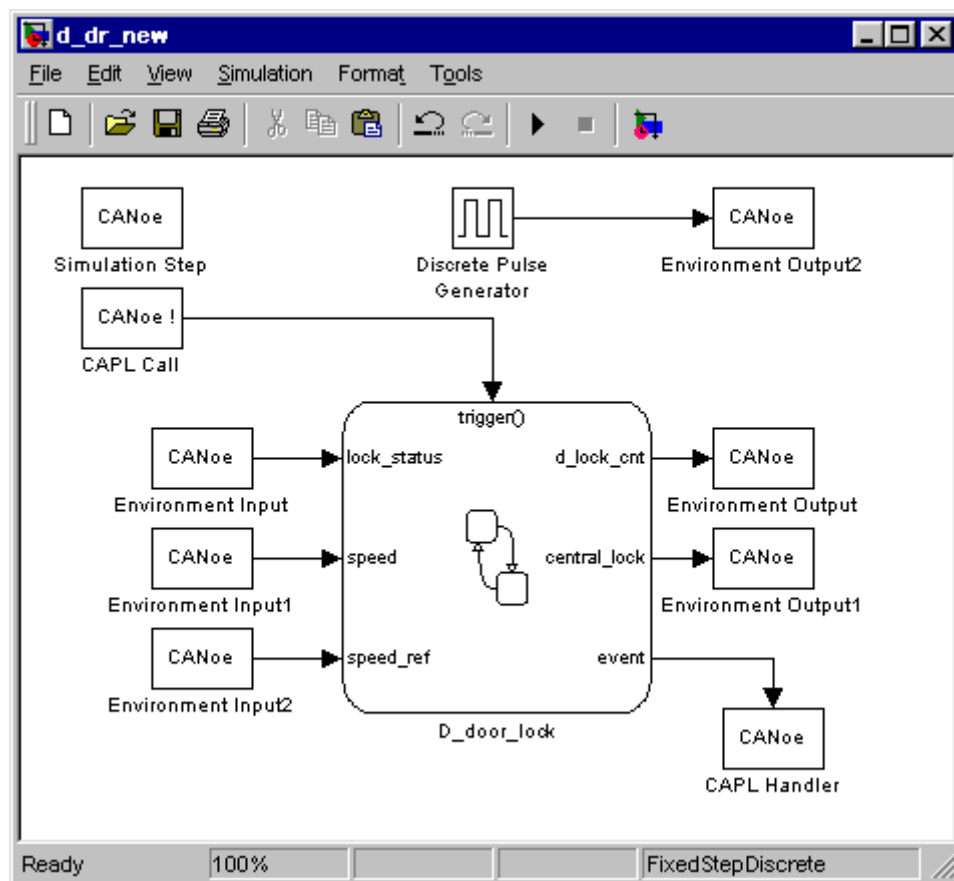


Figure 21: CAPL Handler block

The *CAPL Handler* block is configured with a function name. If the attached CAPL file implements this function it will be called whenever the Simulink® event fires. Function parameters are not supported.

Compatibility:	HIL X	Offline/Synchronized X	(Simulation models only)
----------------	----------	---------------------------	--------------------------

### 6.10.1 Environment Event

The *Environment Event* block is a combination of an *Environment Input* block and a function-call trigger source. You can use this block to trigger Simulink® subsystems.

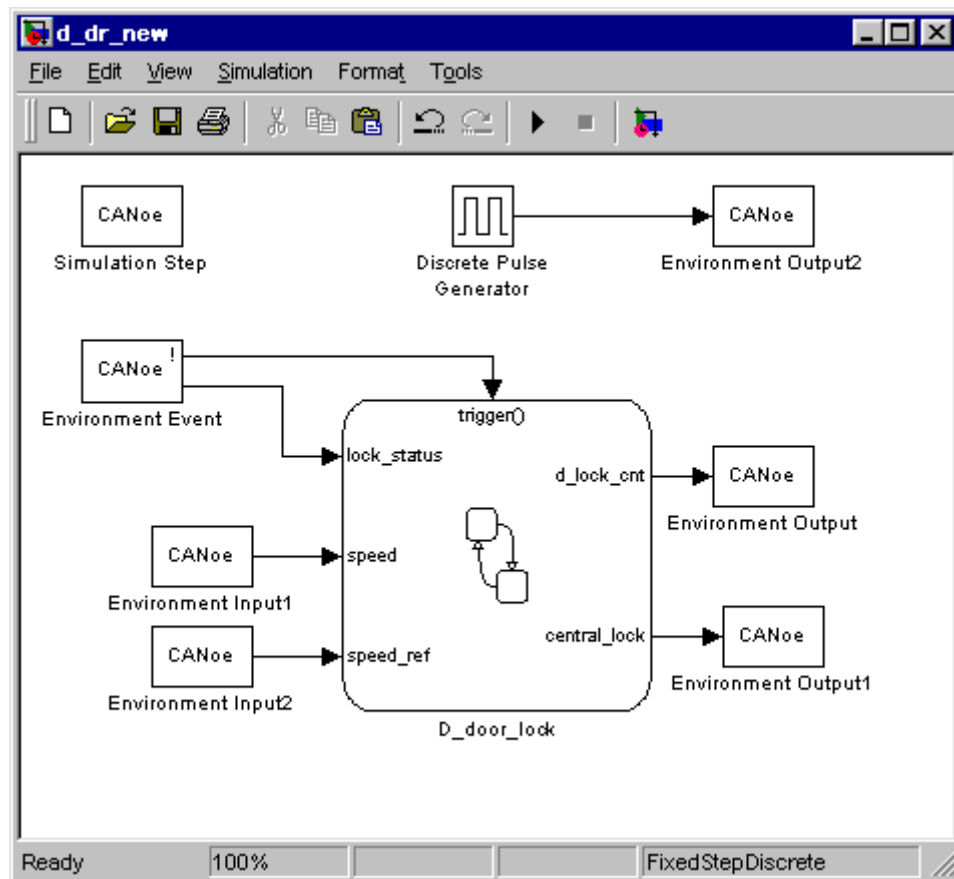


Figure 22: Environment Event block

The *Environment Event* block is configured with an environment variable name. The trigger port fires whenever the input variable changes.

Compatibility:	HIL	Offline/Synchronized
	X	X

### 6.11 Environment Input

The *Environment Input* block is the basic input block for a CANoe environment variable of type integer or float. It is configured with an environment variable name. It contains the additional parameter “Sample Time”. According to Simulink® semantics:

- 0 = continuous sample time
- 1 = inherited sample time

Continuous sample time is set as default. If the *Environment Input* block is used in a triggered subsystem for example, the sample time must be set to “inherited” because only constant blocks or blocks with an inherited sample time are allowed in triggered subsystems.

Compatibility:	HIL	Offline/Synchronized
	X	X

## 6.12 Data Environment Input

The *Data Environment Input* block is used for an environment variable of type data. The bytes of the environment variable are output as a vector from the block.

Compatibility:	HIL	Offline/Synchronized
	X	X

## 6.13 Environment Output

The *Environment Output* block is the basic output block for a CANoe environment variable of type integer or float. It is configured with an environment variable name.

Compatibility:	HIL	Offline/Synchronized	
	X	X	(Simulation models only)

## 6.14 Data Environment Output

The *Data Environment Output* block is used for an environment variable of type data. It receives the bytes of the environment variable as a vector.

Compatibility:	HIL	Offline/Synchronized	
	X	X	(Simulation models only)

## 6.15 Environment Step

The *Environment Step* block is used to designate a Simulink® model that is executed only when one of its CANoe environment inputs changes. This is particularly useful for models whose outputs are a direct function of its inputs and only need to be updated when the inputs are changed.

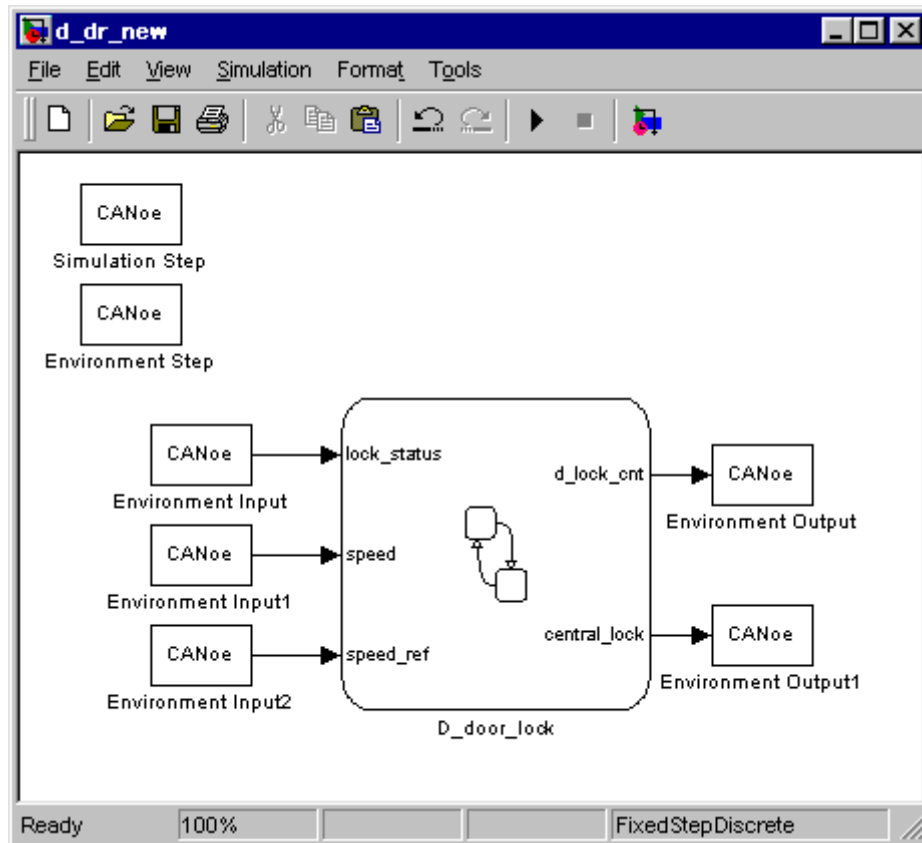


Figure 23: Environment Step block

**Note:** The Environment Step block should only be used if the Simulink® model does not contain any continuous or discrete blocks.

Compatibility:	HIL	Offline/Synchronized
	X	—

## 6.16 Environment Trigger

The *Environment Trigger* block can be used to trigger Simulink® subsystems on environment input variable changes.

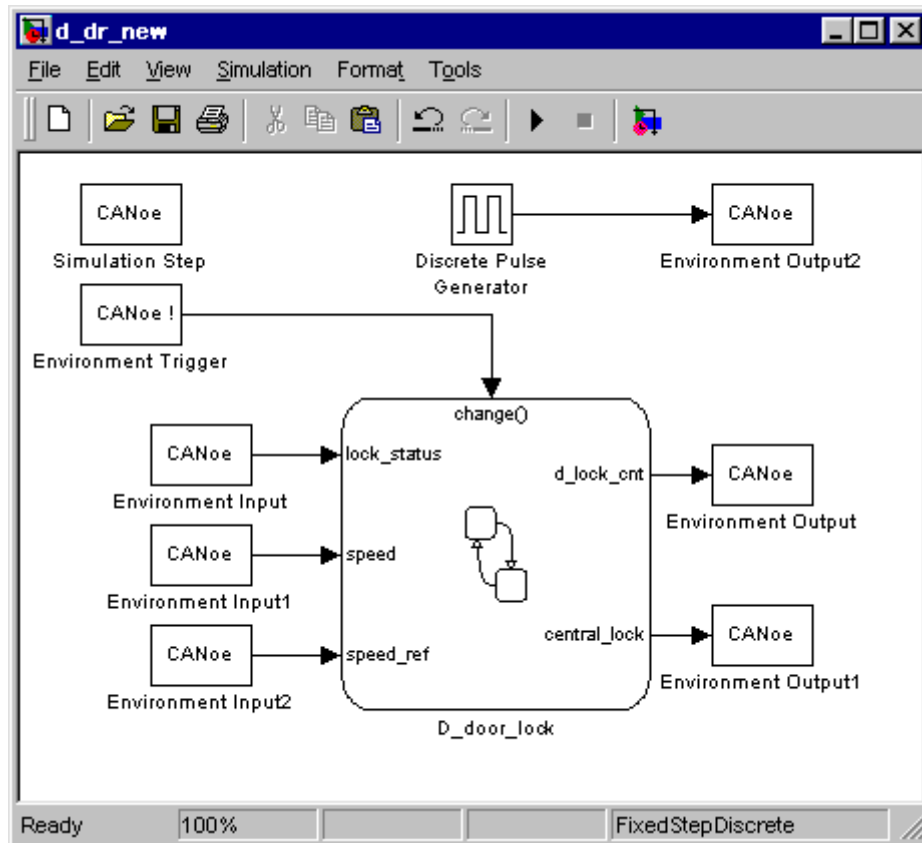


Figure 24: Environment Trigger block

The *Environment Trigger* block fires a function call trigger whenever one of the environment input variables changes.

Compatibility:	HIL	Offline/Synchronized
	X	—

### 6.17 System Variable Input

The *System Variable Input* block is the basic input block for CANoe system variables. It is configured with the name space name, system variable name and an array index for accessing only one value of an array system variable. It contains the additional parameter “Sample Time”. According to Simulink® semantics:

- 0 = continuous sample time
- 1 = inherited sample time

Continuous sample time is set as default. If the *System Variable Input* block is used in a triggered subsystem for example, the sample time must be set to “inherited” because only constant blocks or blocks with an inherited sample time are allowed in triggered subsystems.

Compatibility:	HIL X	Offline/Synchronized X	Matlab R14 and later
----------------	----------	---------------------------	----------------------

### 6.18 System Variable Output

The *System Variable Output* block is the basic output block for a CANoe system variable. It is configured with the name space name, system variable name and an array index for accessing only one value of an array system variable.

Compatibility:	HIL X	Offline/Synchronized X	Matlab R14 and later
----------------	----------	---------------------------	----------------------



## 7 CAPL Functions

### 7.1 General

The RTW<sup>®</sup> generated DLL contains two additional functions which can be called by CAPL. These functions may be used to control the behavior of the generated CANoe/MATLAB<sup>®</sup> Interface and the execution of the Simulink<sup>®</sup> model. The function names are in the form "MODEL\_fct" where MODEL is the name of the Simulink<sup>®</sup> model.

**Note:** The CANoe/MATLAB<sup>®</sup> Interface uses meaningful default values to set up and control Simulink<sup>®</sup> model execution. These defaults are determined during the generation process by the RTW<sup>®</sup>. If you use the CAPL functions described herein you can override these defaults and take control of model execution.

Build option "Export CAPL Functions":

- The CAPL functions described herein are only exported if this option is "on". Default value is "off".
- Long model filenames (> 44 characters) are only supported if this option is "off".

### 7.2 MODEL\_init

#### Function:

```
LONG MODEL_init( DWORD enableTimer, DWORD enableEvents);
```

#### Parameters:

- `DWORD enableTimer`  
Enables or disables the internal model execution timer
- `DWORD enableEvents`  
Enables or disables the event triggered model execution

#### Description:

The `MODEL_init` function is used to initialize the CANoe runtime environment for the generated MODEL DLL. This effectively overrides the default execution settings which were compiled into the MODEL DLL.

#### **DWORD enableTimer**

Per default the generated MODEL is executed by the CANoe runtime environment according to its fixed simulation step size. If the user sets the fixed step size to "auto", Simulink<sup>®</sup> calculates the required minimum fixed step size based on the model contents. However, if the user specifies a fixed step size Simulink<sup>®</sup> checks this value against the model contents during simulation startup and generates an error message if the step size violates any timing restrictions. If the MODEL is built into a

DLL with the RTW, the fixed step size is written into the generated C source code (See generated "MdlInitializeSampleTimes" function in MODEL.c). The CANoe/ MATLAB® Interface reads this generated fixed step size value during measurement startup to set up a timer used to execute the MODEL at the specified sample time, thus providing exactly the same execution environment as in Simulink®. If the user wants to execute the MODEL non-cyclically at arbitrary sample times (for example with the MODEL\_step function) the user must disable the CANoe/ MATLAB® Interface timer. This is done by setting this parameter to zero.

**DWORD enableEvents**

In addition (or instead) of using the MODEL\_step function to execute the MODEL a simulation step may be performed on each environment variable change. To activate this feature this parameter must be set to a nonzero value.

**Return:**

Zero on success, nonzero otherwise.

### 7.3 MODEL\_step

**Function:**

```
LONG MODEL_step();
```

**Description:**

The MODEL\_step function is used to execute a simulation step.

**Return:**

Zero on success, nonzero otherwise.

### 7.4 Parameterization Functions

If the options "Export CAPL Functions" and "Enable Parameterization from CANoe" are activated the RTW® generated DLL will contain the following three functions for accessing parameter values. The return values of all functions are listed in the table below:

Return value	Meaning
0	Success
3	No model data available (internal error)
4	Wrong parameter type: nrOfValues does not match or unsupported parameter value type (complex numbers for example).
5	Wrong index: No parameter exists with the given index.

**Note:** Some parameters might not be configurable during measurement. This depends on the implementation of a Simulink® block. If the `MODEL_set` functions are used in the PreStart section of a CAPL program, all parameters will be configurable and will use the new values during measurement.

Parameterization requires the C-API for data exchange which is available since Matlab R14.

### 7.4.1 MODEL\_getParameter

**Function:**

```
LONG MODEL_getParameter( DWORD index, FLOAT values[],  
                        DWORD nrOfValues)
```

**Description:**

The `MODEL_getParameter` function retrieves the values of a model parameter specified by `index` and copies them into the `values` array. `nrOfValues` must match the number of values the parameter has. The size of the `values` array must not be smaller than `nrOfValues`.

### 7.4.2 MODEL\_setParameter

**Function:**

```
LONG MODEL_setParameter( DWORD index, FLOAT values[],  
                        DWORD nrOfValues)
```

**Description:**

The `MODEL_setParameter` function copies the values from the `values` array to a model parameter specified by `index`. `nrOfValues` must match the number of values the parameter has. The size of the `values` array must not be smaller than `nrOfValues`.

### 7.4.3 MODEL\_setScalarParameter

**Function:**

```
LONG MODEL_setScalarParameter( DWORD index, FLOAT value)
```

**Description:**

The `MODEL_setScalarParameter` function sets a scalar model parameter specified by `index` to the given value.

## 8 Utilities

### 8.1 Setup of signal and environment blocks

**Note:** The setup procedure described here **only** applies to **Matlab R14 or later**

To specify database name, node name, message name and signal name of the signal input and signal output block you can choose from a list rather than typing the names into the block mask parameters. To specify an environment variable name for a block you can also choose from a list of available variables.

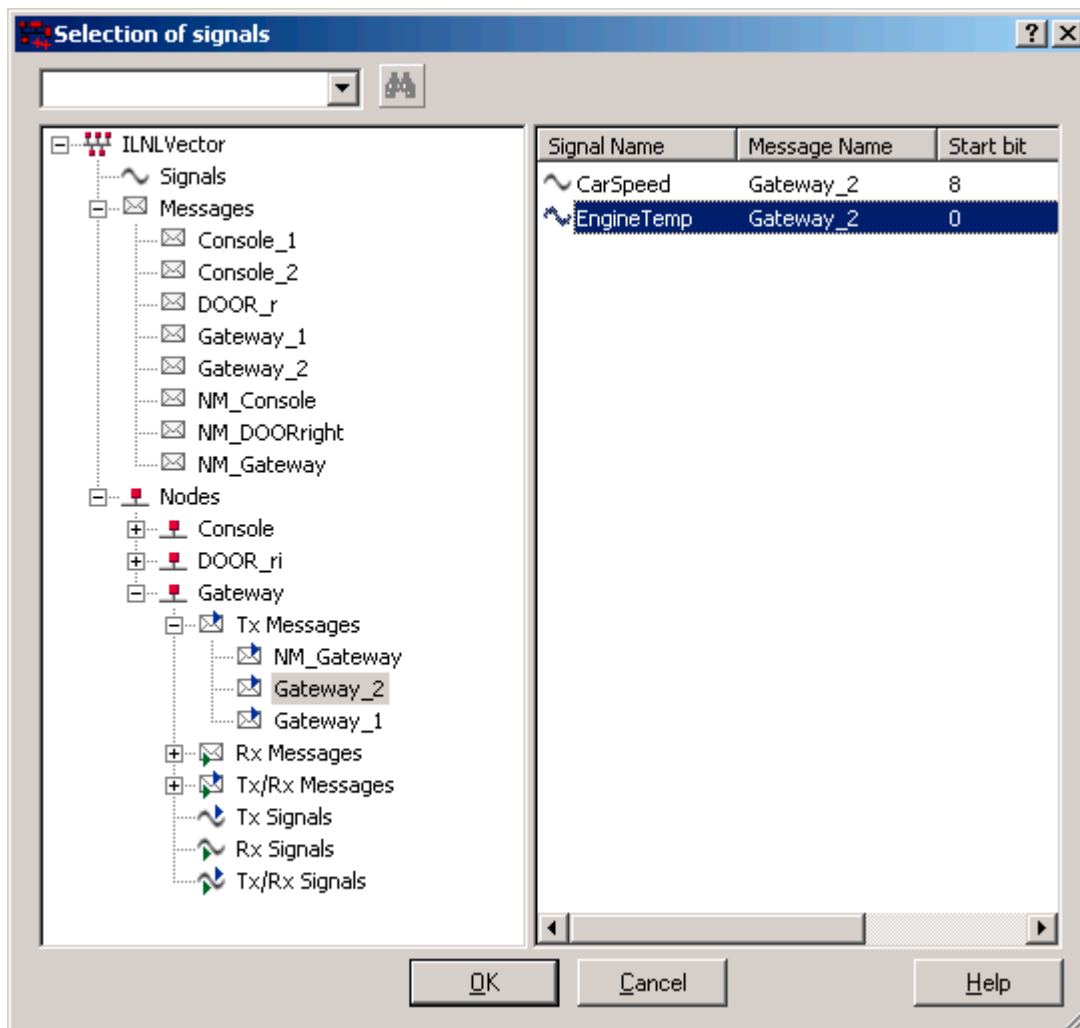


Figure 25: Symbol Selection dialog

A double-click on the signal input / signal output block will bring up the dialog. You can still edit the names by choosing “Mask parameters...” from the context menu.

## 9 Troubleshooting

### 9.1 Unusable CANoe GUI during offline simulation

It can happen that the CANoe user interface doesn't update correctly or isn't usable at all when operation in offline mode. If so, open the CAN.INI configuration file and verify that the following entry is set to zero:

```
[System]
GUIThreadPriority=0
```

The default setting for this entry is -3 which lowers the priority of the CANoe GUI.

### 9.2 Unable to compile the *nlapml.cpp* source file.

If the compilation of the source file fails with the following error:

**error C2117: 'test\_for\_long\_filename\_init' : array bounds overflow**

you've activated the export of CAPL functions in the code generation options and the file name of your model is longer than 44 characters. If you **really** need the CAPL functions save your model under a shorter name. If you don't need the additional CAPL functions, deactivate the export in the CN code generation options.

### 9.3 Spaces in MATLAB® installation path

Due to problems with spaces in the MATLAB® installation path Mathworks® issued a patch for the code generation process which applies to MATLAB® 7.0 R14 SP2. If you are using MATLAB® 7.0 R14 SP2 or R2007a and have installed it in a directory which contains spaces you cannot generate code unless you modify your target makefile. In this case please add the following lines to your target makefile.

```
( $(MATLABROOT)\rtw\c\canoe\cn_msdl1.tmf) :

line 101:

ALT_MATLAB_ROOT          = |>ALT_MATLAB_ROOT<|
ALT_MATLAB_BIN            = |>ALT_MATLAB_BIN<|

!if "$(MATLAB_ROOT)" != "$(ALT_MATLAB_ROOT)"
MATLAB_ROOT = $(ALT_MATLAB_ROOT)
!endif
!if "$(MATLAB_BIN)" != "$(ALT_MATLAB_BIN)"
MATLAB_BIN = $(ALT_MATLAB_BIN)
!endif
```

## 9.4 Model Advisor Checks not available

After installation of the CANoe Matlab® Interface the Simulink® Model Advisor should contain a new folder named “CANoe Matlab Interface Checks” below the “By Product section”. The CANoe checks are created by a file named `sl_customization.m` located in the folder `$(My Documents)\Matlab\CANoe\Samples`. Please make sure that this folder is included in the Matlab® path list (menu “File” – “Set Path...”). Matlab® updates all customizations if the command `slcustomize` is called. Please make sure that you have the Simulink® Verification and Validation Toolbox installed, which supports custom checks for the Model Advisor.

## 9.5 High CPU load during simulation

If the simulation is run in Synchronized mode, best synchronization results are achieved by using a “busy wait” function in Simulink®. The advantage is that the simulation remains active and is just delayed by the Simulation Step block. The block continuously checks the current time of CANoe until the next simulation step must be computed.

Therefore it is recommended to use a multi core processor. In this case Simulink® will cause up to 100% load at one of the CPU’s cores while CANoe can be computed by the other core. Using a single core processor CANoe and Simulink® have to share the capacity of the CPU possibly resulting in less exact simulation results.

In Offline mode it not critical how fast Simulink® can compute the model since CANoe interrupts its simulation and resumes correctly synchronized with Simulink®. Since the simulation is run as fast as possible, it will also use most of the CPU capacity available.

## 10 Index

### A

actuator ..... 6  
Attributes ..... 10, 25

### C

CANdb attributes ..... 11  
CANoe ..... 1, 3, 47  
CANoe I/O Library ..... 30  
CANoe Interaction Layer ..... 5, 10, 31, 34, 35  
CANoeIL ..... 5  
CAPL Call ..... 36  
CAPL Handler ..... 37

### D

Data Environment Input ..... 39  
Data Environment Output ..... 39

### E

ECU ..... 4, 6, 7  
Enable Parameterization from CANoe .... 13, 28  
Environment Event ..... 38  
Environment Input ..... 38  
Environment Output ..... 39  
Environment Step ..... 39  
Environment Trigger ..... 40  
Export CAPL-Function ..... 13, 28

### G

Generate Visual C++ project file ..... 13, 28

### H

HIL ..... 24  
HIL Mode ..... 3

### I

Interacion Layer ..... 6  
Interaction Layer ..... 5

### M

Message Input ..... 34  
Message Output ..... 35  
Middleware ..... 4, 6  
Model Generation Wizard ..... 11  
MODEL\_getParameter ..... 45  
MODEL\_init ..... 43  
MODEL\_setParameter ..... 45  
MODEL\_setScalarParameter ..... 45  
MODEL\_step ..... 44

### N

Network management ..... 10  
Network Management Layer ..... 7  
NodeLayerModules ..... 25

### O

Offline ..... 24  
Offline Mode ..... 2  
OSEK ..... 10

### P

Parameterization ..... 14  
Perform Simulation Step at t(0) ..... 13, 28

### R

RTW Options ..... 27

### S

sensor ..... 6  
Setup ..... 8  
Signal Access ..... 24  
Signal Input ..... 31  
Signal Output ..... 31  
Simple System Demo ..... 26  
Simulation Setup ..... 26, 27  
Simulation Step ..... 24  
Sine Parameterization ..... 25, 28

Symbol Selection dialog .....	46
System Target File Browser.....	9
System Variable Input.....	41
System Variable Output.....	42

---

## T

Transport Layer.....	6
----------------------	---