Bus Manager

Tutorial

For Bus Manager 6.7

Release 2021-A – May 2021

dSPACE

How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	info@dspace.de
Web:	http://www.dspace.com

How to Contact dSPACE Support

If you encounter a problem when using dSPACE products, contact your local dSPACE representative:

- Local dSPACE companies and distributors: http://www.dspace.com/go/locations
- For countries not listed, contact dSPACE GmbH in Paderborn, Germany.
  Tel.: +49 5251 1638-941 or e-mail: support@dspace.de

You can also use the support request form: http://www.dspace.com/go/supportrequest. If you are logged on to mydSPACE, you are automatically identified and do not need to add your contact details manually.

If possible, always provide the relevant dSPACE License ID or the serial number of the CmContainer in your support request.

Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit http://www.dspace.com/go/patches for software updates and patches.

Important Notice

This publication contains proprietary information that is protected by copyright. All rights are reserved. The publication may be printed for personal or internal use provided all the proprietary markings are retained on all printed copies. In all other cases, the publication must not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of dSPACE GmbH.

© 2020 - 2021 by:
dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Germany

This publication and the contents hereof are subject to change without notice.

AUTERA, ConfigurationDesk, ControlDesk, MicroAutoBox, MicroLabBox, SCALEXIO, SIMPHERA, SYNECT, SystemDesk, TargetLink and VEOS are registered trademarks of dSPACE GmbH in the United States or other countries, or both. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.

Contents

About This Tutorial 7

Introduction to the Bus Manager 11

Basics on the Bus Manager ... 12

Introduction to the Tutorial 19

Introduction to the Example Used in the Tutorial ... 19
Working with the Tutorial ... 22
Overview of Use Scenarios and Workflows ... 24
Overview of Lessons ... 25

Workflows for Using the Bus Manager 27

Workflows for Configuring Bus Communication and Generating Bus Simulation Containers ... 28
Configuring Bus Communication for Generating Bus Simulation Containers with Existing Behavior Models ... 28
Configuring Bus Communication for Generating Bus Simulation Containers with a New Simulink Implementation Container ... 32
Configuring Bus Communication for Generating Bus Simulation Containers Without Behavior Models ... 38

Workflows for Configuring Bus Communication and Building Real-Time Applications ... 42
Configuring Bus Communication for Real-Time Applications and Using Existing Behavior Models ... 42
Configuring Bus Communication for Real-Time Applications and Generating a New Simulink Model ... 47
Configuring Bus Communication for Real-Time Applications Without Behavior Models ... 53

Working with the BusManagerTutorial Project 59

How to Open the BusManagerTutorial Project ... 59
How to Activate an Application in the BusManagerTutorial Project ... 61
How to Identify Accessible Licenses ... 62

Lesson 1: Adding Communication Matrices and Bus Configurations to the ConfigurationDesk Application 65

Overview of Lesson 1 65
Step1: How to Add a Communication Matrix to the ConfigurationDesk Application 66
Step 2: How to Check the Communication Matrix for Conflicts 68
Step 3: How to Add a Bus Configuration to the ConfigurationDesk Application 71
Result of Lesson 1 72

Lesson 2: Configuring Bus Communication for Simulation Purposes 75

Overview of Lesson 2 75
Step 1: How to Assign Communication Matrix Elements to the Simulated ECUs Part of a Bus Configuration 76
Step 2: How to Add Bus Configuration Features for Simulation Purposes 79
Step 3: How to Configure Function Ports of Bus Simulation Features 84
Step 4: How to Check the Configured Bus Communication for Conflicts and Resolve Conflicts 89
Result of Lesson 2 93

Lesson 3: Working with an Existing Behavior Model 97

Overview of Lesson 3 97
Step 1: How to Add a Behavior Model to the ConfigurationDesk Application 98
Step 2: How to Map Model Ports to Function Ports 100
Step 3: How to Explore Mapped Ports in a Graphical View 105
Result of Lesson 3 107

Lesson 4: Generating Bus Simulation Containers 109

Overview of Lesson 4 109
Step 1: How to Generate Bus Simulation Containers 110
Result of Lesson 4 114

Lesson 5: Building a Real-Time Application 115

Overview of Lesson 5 115

Step 1: How to Import a Hardware Topology 116
Step 2: How to Specify the Hardware Access for Bus Configurations. 118
Step 3: How to Build a Real-Time Application 121
Result of Lesson 5 123

Lesson 6: Configuring Bus Communication for Inspection Purposes 125

Overview of Lesson 6 125
Step 1: How to Assign Communication Matrix Elements to the Inspection Part of a Bus Configuration 126
Step 2: How to Add Bus Configuration Features for Inspection Purposes 128
Result of Lesson 6 130

Lesson 7: Configuring Bus Communication for Manipulation Purposes 133

Overview of Lesson 7 133
Step 1: How to Assign Communication Matrix Elements to the Manipulation Part of a Bus Configuration 135
Step 2: How to Add Bus Configuration Features for Manipulation Purposes 137
Step 3: How to Configure the Run-Time Behavior for Manipulating Bus Communication 139
Result of Lesson 7 142

Lesson 8: Working Without a Behavior Model 145

Overview of Lesson 8 145
Step 1: How to Configure Bus Configuration Elements for Access via Experiment Software and Automation Scripts 147
Step 2: How to Create an Application Process that Provides a Default Task 152
Step 3: How to Manually Assign a Bus Configuration to an Application Process 152
Result of Lesson 8 153

Lesson 9: Generating a Simulink Model and Synchronizing the Model Interfaces in ConfigurationDesk and Simulink 155

Overview of Lesson 9 155

Step 1: How to Generate a New Simulink Model Interface ... 157
Step 2: How to Add a Simulink Model to the ConfigurationDesk Application ... 159
Step 3: How to Propagate Changes to a Simulink Model ... 161
Step 4: How to Model Input Bus Signals in the Simulink Model ... 167
Step 5: How to Analyze the Simulink Model in the ConfigurationDesk Application ... 169
Step 6: How to Replace a Simulink Model with a Simulink Implementation Container ... 170
Result of Lesson 9 ... 172

Additional Lesson: Experimenting with ControlDesk 175

Overview of the Experimenting with ControlDesk Lesson ... 175
Building a Suitable Executable Application ... 176
Step 1: How to Create a Project and an Experiment in ControlDesk ... 179
Step 2: How to Prepare a Layout for Simulating LIN Communication ... 182
Step 3: How to Simulate LIN Communication ... 188
Step 4: How to Prepare a Layout for Simulating, Manipulating, and Inspecting CAN Communication ... 191
Step 5: How to Simulate, Manipulate, and Inspect CAN Communication ... 193
Result of the Experimenting with ControlDesk Lesson ... 197

Summary 199

Your Working Results ... 199

ConfigurationDesk Glossary 201

Index 227

About This Tutorial

Content

Introduces you to the basic steps of working with the Bus Manager.

Tip

Two variants of the Bus Manager are available, the Bus Manager in ConfigurationDesk and the Bus Manager (stand-alone). This tutorial is valid for both Bus Manager variants.

Target group

This tutorial is intended for engineers who use the Bus Manager to configure CAN and/or LIN communication for simulation, inspection, and/or manipulation purposes. The target group performs some or all of the following tasks:

- Set up bus communication based on communication matrices.
- Specify a model interface to exchange bus signals between the Bus Manager and a behavior model modeled in a modeling tool such as MATLAB®/Simulink®.
- Build real-time applications or generate bus simulation containers that include the configured bus communication.

Required knowledge

Basic knowledge of the CAN and LIN bus protocols is assumed. Additionally, depending on the lessons you work through, basic knowledge in the following areas is also assumed:

- SCALEXIO/MicroAutoBox III hardware
- VEOS Player/VEOS
- ControlDesk
- MATLAB®/Simulink®

Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
DANGER	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
WARNING	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
CAUTION	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
NOTICE	Indicates a hazard that, if not avoided, could result in property damage.
Note	Indicates important information that you should take into account to avoid malfunctions.
Tip	Indicates tips that can make your work easier.
	Indicates a link that refers to a definition in the glossary, which you can find at the end of the document unless stated otherwise.
	Precedes the document title in a link that refers to another document.

Naming conventions

dSPACE user documentation uses the following naming conventions:

%name% Names enclosed in percent signs refer to environment variables for file and path names.

< > Angle brackets contain wildcard characters or placeholders for variable file and path names, etc.

Special folders

Some software products use the following special folders:

Common Program Data folder A standard folder for application-specific configuration data that is used by all users.

```
%PROGRAMDATA%\dSPACE\<InstallationGUID>\<ProductName>
```

or

```
%PROGRAMDATA%\dSPACE\<ProductName>\<VersionNumber>
```

Documents folder A standard folder for user-specific documents.

```
%USERPROFILE%\Documents\dSPACE\<ProductName>\
<VersionNumber>
```

Local Program Data folder A standard folder for application-specific configuration data that is used by the current, non-roaming user.

```
%USERPROFILE%\AppData\Local\dSPACE\<InstallationGUID>\
<ProductName>
```

Accessing dSPACE Help and PDF Files

After you install and decrypt dSPACE software, the documentation for the installed products is available in dSPACE Help and as PDF files.

dSPACE Help (local) You can open your local installation of dSPACE Help:

- On its home page via Windows Start Menu
- On specific content using context-sensitive help via **F1**

dSPACE Help (Web) You can access the Web version of dSPACE Help at www.dspace.com/go/help.

To access the Web version, you must have a *mydSPACE* account.

PDF files You can access PDF files via the  icon in dSPACE Help. The PDF opens on the first page.

Introduction to the Bus Manager

Purpose of the Bus Manager

The Bus Manager lets you configure CAN and LIN communication for simulation, inspection, and manipulation purposes. You can work with multiple communication matrix files of various file formats and configure the communication of multiple communication clusters at the same time.

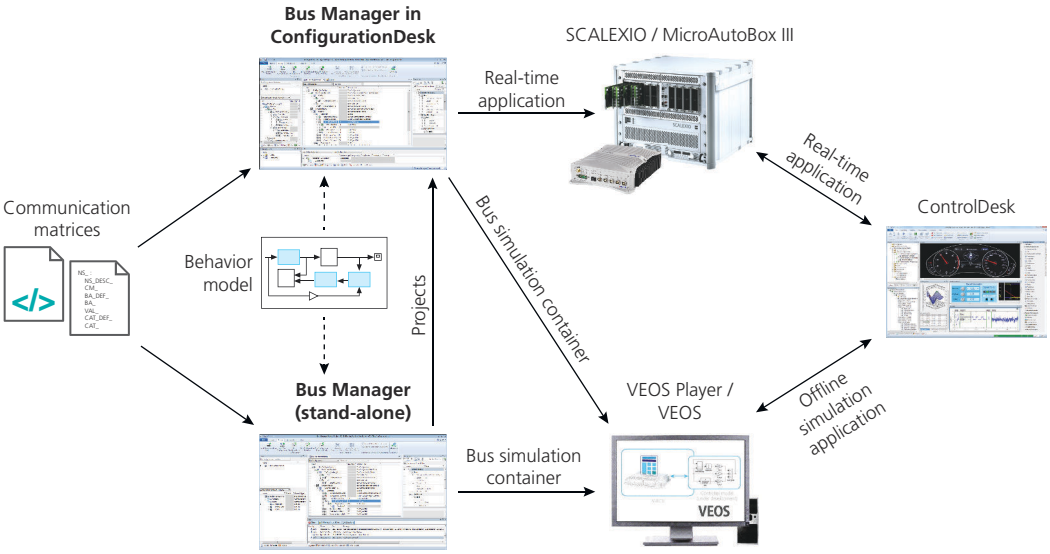
Note

It is assumed that you are familiar with the content of this section before you start working through the tutorial.

Basics on the Bus Manager

Variants of the Bus Manager

Two variants of the Bus Manager are available. The following illustration provides an overview of the variants and their fields of application.



Bus Manager in ConfigurationDesk The Bus Manager in ConfigurationDesk lets you configure bus communication and implement it directly in a real-time application for dSPACE SCALEXIO or MicroAutoBox III systems. Alternatively, it lets you generate bus simulation containers. Bus simulation containers can be used on various dSPACE simulation platforms. For example, you can use bus simulation containers in the VEOS Player to implement the configured bus communication in an offline simulation application for VEOS.

Bus Manager (stand-alone) The Bus Manager (stand-alone) lets you configure bus communication and generate bus simulation containers. To use the configured bus communication on a specific dSPACE simulation platform, you can import the bus simulation containers to the VEOS Player or ConfigurationDesk. The VEOS Player lets you implement the bus communication in an offline simulation application, ConfigurationDesk lets you implement the bus communication in a real-time application. You can also reuse projects you used with the Bus Manager (stand-alone) with the Bus Manager in ConfigurationDesk (e.g., to work with an already configured bus communication and implement it in a real-time application).

Use scenarios for simulating, inspecting, and manipulating bus communication

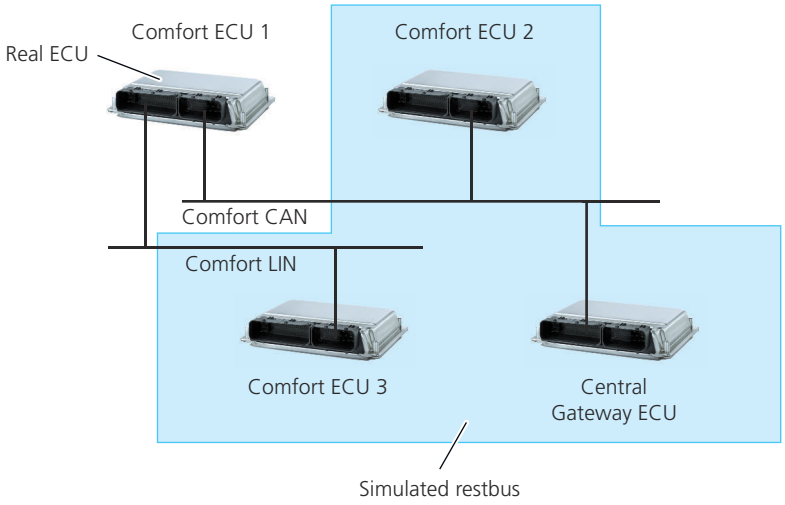
Use scenarios for simulating bus communication The Bus Manager lets you simulate the bus communication of ECUs. Typical use scenarios for this are the following:

- ECU simulation

  You can simulate individual ECUs, e.g., to test the behavior model of an ECU before the real ECU is available.

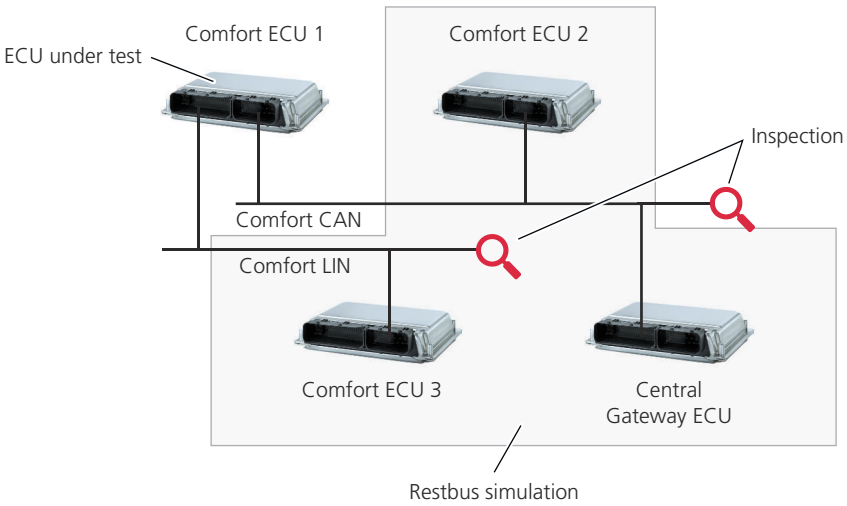
- Restbus simulation

  Restbus simulation lets you test one or more ECUs while simulating the other ECUs of the related communication clusters. For example, to test a real ECU, you can connect the real ECU to a simulator and simulate all the other ECUs that communicate with the real ECU, as shown in the following example.



Use scenario for inspecting bus communication The Bus Manager lets you inspect the bus communication of each communication cluster independently from the involved ECUs. Inspecting bus communication focuses on the bus communication that is already available on the bus (i.e., after it is transmitted) and that can be received by the bus members.

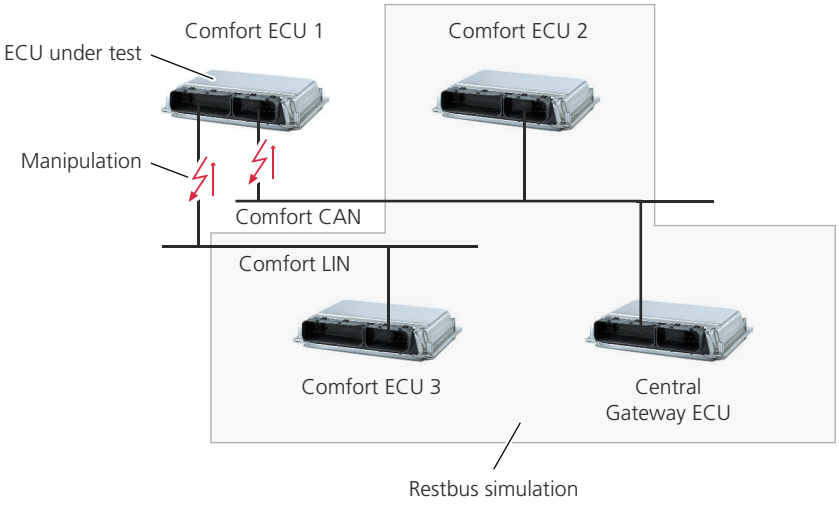
For example, you can inspect the bus communication that is exchanged between an ECU under test and simulated ECUs in a restbus simulation scenario, as shown in the following example.



Bus communication inspection is performed for each cluster and channel separately. This allows you, for example:

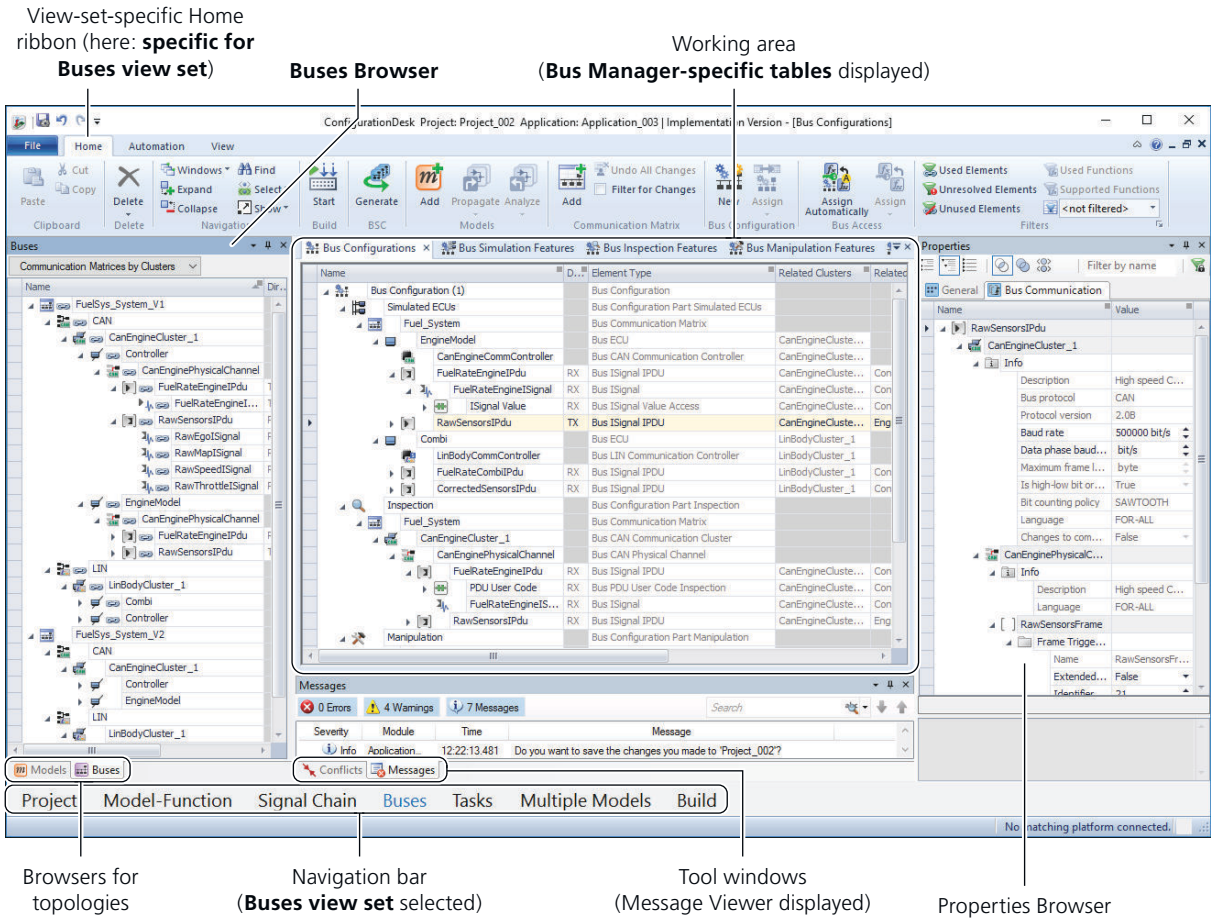
- To inspect the bus communication independently from the ECU development stage. For example, replacing a simulated ECU with a real ECU has no effect on your configurations.
- To reduce the effort for implementing bus communication in a ConfigurationDesk application. For example, to inspect whether a PDU is available on the bus, you have to configure the PDU only once for the inspection and not separately for all of its receiving ECUs.

Use scenario for manipulating bus communication The Bus Manager lets you manipulate bus communication before it is transmitted on the bus, for example, to test the response of an ECU under test in case it receives faulty bus communication. The manipulation of bus communication is based on communication clusters, i.e., it is independent from the sending ECUs. The following illustration is an example of bus communication manipulation in combination with a restbus simulation.



User interface

When you work with the Bus Manager, you use Bus Manager elements and standard ConfigurationDesk elements. In the following illustration, the Bus Manager elements are in bold letters.



Tip

If you use the Bus Manager (stand-alone), not all ConfigurationDesk elements are available, e.g., not all of the view sets displayed in the navigation bar.

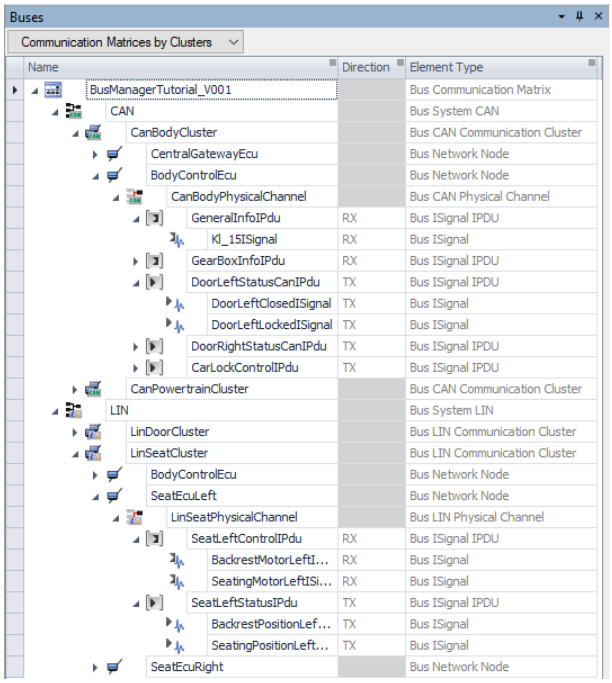
Working with communication matrices

To configure CAN and/or LIN communication with the Bus Manager, you need one or more communication matrices. Communication matrices can describe the bus communication of one communication cluster or a bus network consisting of different bus systems (e.g., CAN, LIN) and communication clusters.

The Bus Manager supports the following communication matrix file formats:

- AUTOSAR system description files
- FIBEX files
- DBC files
- LDF files

When you add communication matrices to the ConfigurationDesk application, the communication matrix elements are displayed hierarchically, as shown in the following example.

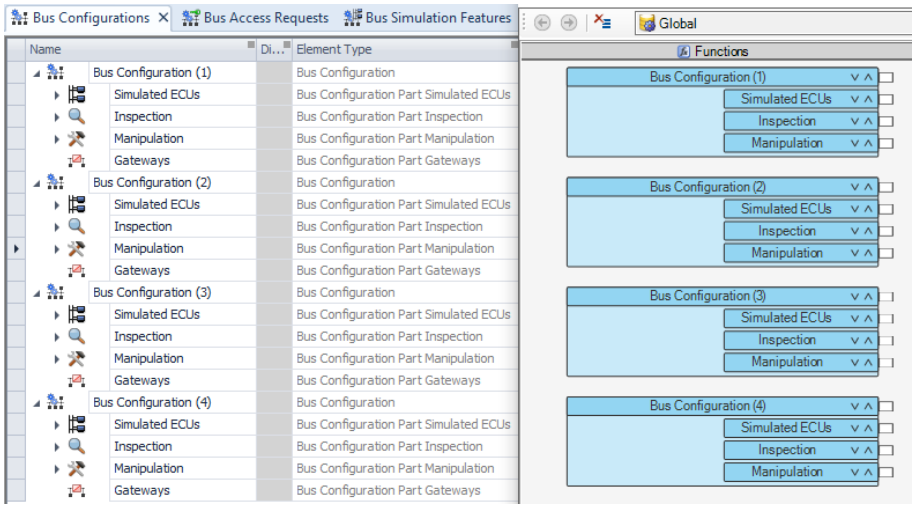


To configure bus communication, you must assign communication matrix elements to bus configurations.

Introduction to bus configurations

Bus configurations are the main Bus Manager element to configure bus communication. Each bus configuration provides specific parts that let you configure the bus communication independently for simulation, inspection, and manipulation purposes. Additionally, bus configurations let you specify gateways to exchange bus communication between two communication clusters.

You can work with multiple bus configurations at the same time, and access the bus configurations via specific tables and their **Bus Configuration** function blocks, as shown in the following example.



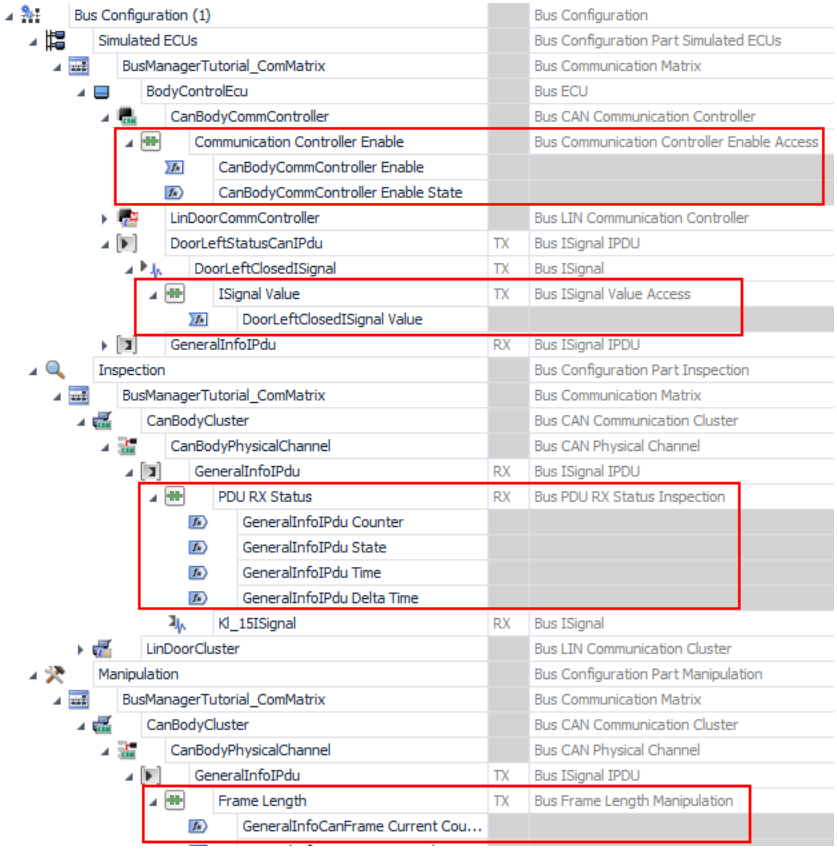
To each bus configuration, you can assign elements of one or more communication matrices and communication clusters, even of different communication matrix file formats (e.g., ARXML and DBC) and bus systems (i.e., CAN and LIN). When you do this, you can configure the bus communication by using bus configuration features.

Working with bus configuration features

The Bus Manager provides various bus configuration features that let you configure the bus communication. The bus configuration features are specific for the following:

- The bus configuration parts, i.e., Simulated ECUs, Inspection, Manipulation, and Gateways.
- The bus configuration elements, e.g., the bus configuration node, assigned ECUs, communication controllers, PDUs, or ISignals.

For example, bus configuration features let you enable and disable communication controllers, access ISignal values, inspect the status of received PDUs, or manipulate the length of CAN frames.



Each bus configuration feature applies only to the specific element to which it is added and not to other elements in the bus configuration.

Depending on the bus configuration feature, function ports are available. Function ports provide the interface for accessing data at run time. You can configure function ports to exchange data with a behavior model and/or access data via experiment software (e.g., ControlDesk).

Working with behavior models

The Bus Manager lets you work with and without behavior models. Behavior models can define the behavior of ECUs, for example. If you work with a behavior model, you can dynamically exchange data with the behavior model at run time, for example, between a real ECU that is connected to a simulator and ECUs that are simulated by the Bus Manager.

Introduction to the Tutorial

Purpose

To enable you to efficiently work with this tutorial. You are introduced to the example used and to the structure of the tutorial. Additionally, you will find tips on how to work with the tutorial.

Note

It is assumed that you are familiar with the content of this section before you start working through the tutorial.

Where to go from here

Information in this section

Introduction to the Example Used in the Tutorial....19
Working with the Tutorial....22
Overview of Use Scenarios and Workflows....24
Overview of Lessons....25

Introduction to the Example Used in the Tutorial

Introduction to the communication matrix

The example that is used in the tutorial is based on the `BusManager_Base_V1_0.arxml` communication matrix. This communication matrix specifies the communication of a bus network of a car, consisting of four communication clusters and eight ECUs.

Note

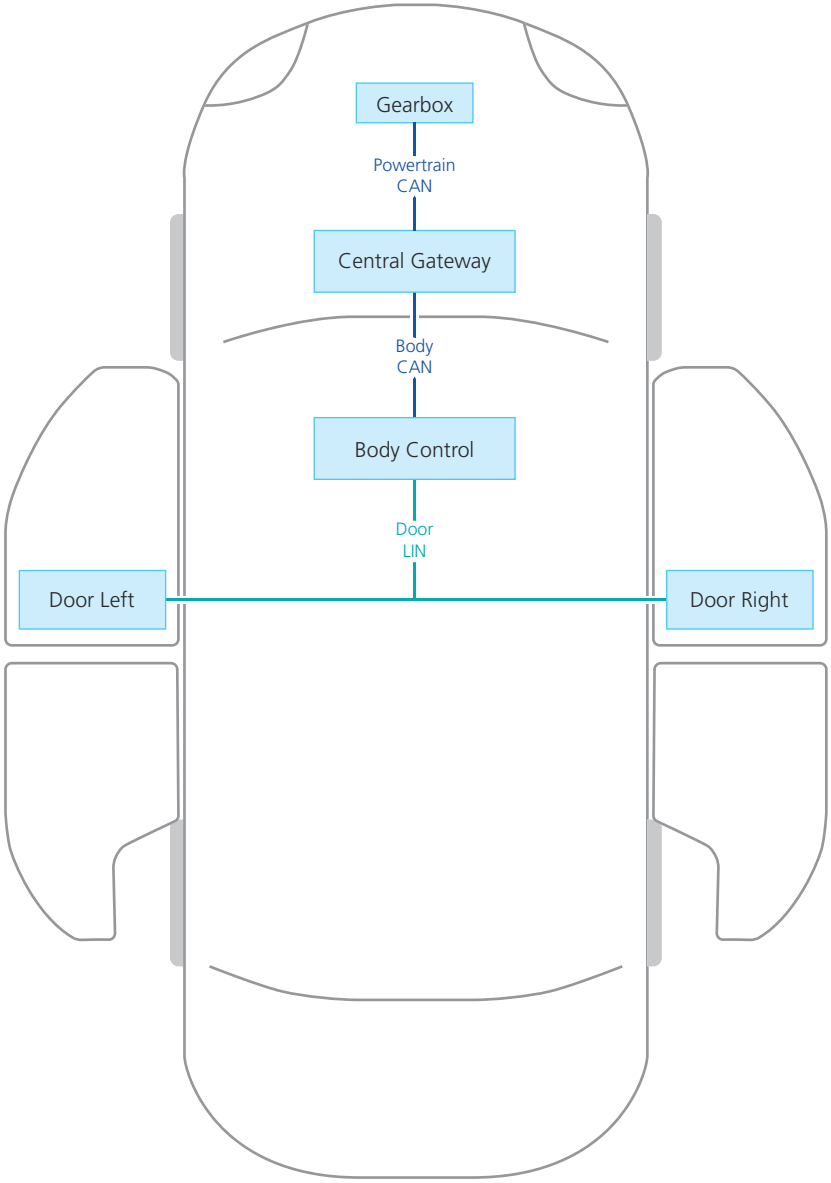
The `BusManager_Base_V1_0.arxml` communication matrix specifies only simplified bus communication for training purposes. It does not represent the complexity of real bus communication in a bus network of a car.

Example used in the tutorial

In the tutorial, you will work with three communication clusters and five ECUs:

Communication Cluster	Bus System	ECU
Body	CAN	▪ Central Gateway ▪ Body Control
Powertrain	CAN	▪ Central Gateway ▪ Gearbox
Door	LIN	▪ Body Control ▪ Door Left ▪ Door Right

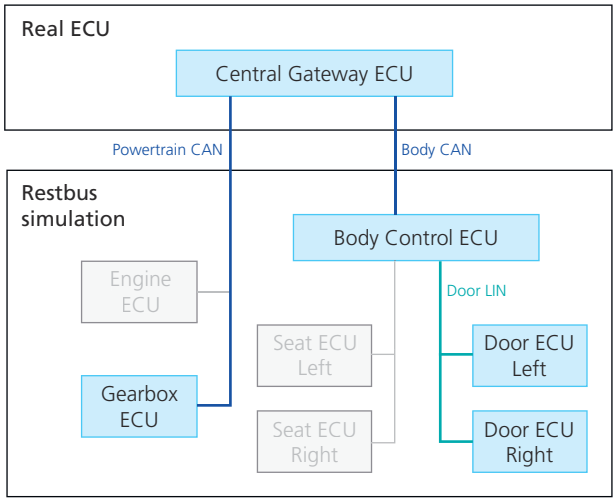
The following illustration shows the communication between the ECUs.



Use scenarios in this tutorial

You will configure bus communication for simulation, inspection, and manipulation purposes.

Restbus simulation In the tutorial, it is assumed that the Central Gateway ECU is available as a real ECU and is the device under test. To test the Central Gateway ECU, you will set up a restbus simulation, i.e., simulate ECUs of the bus network that communicate with the Central Gateway ECU. The following illustration shows this use scenario.



Tip

- The grayed-out elements are specified in the communication matrix, but you will not use them in the tutorial.
- To additionally reduce the complexity, you will configure only some PDUs and ISignals for the restbus simulation and not all of the PDUs/ISignals that are exchanged with the Central Gateway ECU.

Inspecting bus communication You will inspect the bus communication of the CAN Powertrain and CAN Body communication clusters, i.e., the bus communication that is exchanged between the Central Gateway ECU and the ECUs you configured for the restbus simulation.

Manipulating bus communication You will manipulate some PDUs and ISignals that are transmitted by the ECUs of the simulated restbus and received by the Central Gateway ECU, i.e., the Central Gateway ECU receives manipulated data. This lets you test the response of the Central Gateway ECU to faulty bus communication, for example.

Working with the Tutorial

Purpose of the tutorial

This tutorial shows you the basic steps required to configure CAN and LIN communication by using the Bus Manager, and implement the bus communication in bus simulation containers and/or a real-time application.

Workflows and lessons

The tutorial is structured by workflows and lessons as follows:

- Workflows for typical use scenarios

  There are various use scenarios for the Bus Manager. The tutorial introduces you to typical use scenarios and provides overviews of the required workflow steps.
- Lessons covering individual workflow steps

  The tutorial provides lessons that guide you through the individual workflows step by step. This results in the following:
  - Each workflow is divided into multiple lessons.
  - Not all lessons apply to each workflow.
  - The lessons that are relevant for your work depend on your use scenario.

Tip

- You can work through each lesson without having to complete other lessons beforehand.
- To work efficiently with the tutorial, use the workflows to select the lessons that are relevant for your use scenario. Refer to Overview of Use Scenarios and Workflows on page 24.

Required licenses

The Bus Manager is protected by several licenses, whereby the required licenses depend on your use scenario. Because the lessons of this tutorial cover different use scenarios, the required licenses for working through the tutorial depend on the lessons. For an overview of the lessons and their required licenses, refer to Overview of Lessons on page 25.

Tip

You can request evaluation licenses to complete this tutorial with the Bus Manager. For more information, contact dSPACE Support (www.dspace.com/go/supportrequest).

Tutorial project and applications

Working with the tutorial project and the applications This tutorial is based on the BusManagerTutorial project. For each tutorial lesson, the project provides a separate ConfigurationDesk application, which is the result of the related lesson.

Tip

- For lesson 1, two applications are available: Lesson_1_Start and Lesson_1_Result.
- For *Additional Lesson: Experimenting with ControlDesk*, no result application is available because you will finish this lesson in ControlDesk.

You can activate each application separately. This lets you use the applications as follows:

- You can use an application as the starting point of a related subsequent lesson. This lets you work through each lesson without having to complete other lessons beforehand.

  For an overview of the lessons and the suitable starting points, refer to Overview of Lessons on page 25.

  Tip

  The BusManagerTutorial project is backed up in a ZIP archive. You can use this backup if you made changes to applications in the tutorial project and you want to restore the initial states.

- You can compare your working results of a lesson with the lesson's result application.

For instructions on working with the tutorial project and applications, refer to Working with the BusManagerTutorial Project on page 59.

Location of the tutorial project and its backup The locations of the tutorial project and its backup are the following:

- The BusManagerTutorial project is available in the `Tutorials` subfolder of the Documents folder.
- The `BusManagerTutorial.zip` archive is available in the `<RCP and HIL installation folder>\Demos\<ProductName>\Tutorial` folder.

Overview of Use Scenarios and Workflows

Use scenarios and workflows The following table provides an overview of typical use scenarios for the Bus Manager and the descriptions of the related workflows in the tutorial.

Use Scenario	Related Workflow
Configuring bus communication for simulation, inspection, and/or manipulation purposes, and generating bus simulation containers:	
Using an existing behavior model	Configuring Bus Communication for Generating Bus Simulation Containers with Existing Behavior Models on page 28
Generating a new Simulink implementation container	Configuring Bus Communication for Generating Bus Simulation Containers with a New Simulink Implementation Container on page 32
Working without a behavior model	Configuring Bus Communication for Generating Bus Simulation Containers Without Behavior Models on page 38

Use Scenario	Related Workflow
Configuring bus communication for simulation, inspection, and/or manipulation purposes, and building a real-time application:	
Using an existing behavior model	Configuring Bus Communication for Real-Time Applications and Using Existing Behavior Models on page 42
Generating a new Simulink model	Configuring Bus Communication for Real-Time Applications and Generating a New Simulink Model on page 47
Working without a behavior model	Configuring Bus Communication for Real-Time Applications Without Behavior Models on page 53

Overview of Lessons

Overview

The following table provides an overview of the tutorial lessons, the applications that can be used as a starting point, and the licenses and additional tools that are required to work through each of the lessons.

Tip

- The Bus Manager license (BUS_MANAGER) is mandatory to use the Bus Manager. Therefore, this license is not explicitly listed in the following table.
- You can request evaluation licenses to complete this tutorial with the Bus Manager. For more information, contact dSPACE Support (www.dspace.com/go/supportrequest).
- For some lessons, you can use different applications as the starting point. The application in italics indicates the application that was used as the starting point to create the result application of the related lesson.

Lesson	Suitable Start Application	Required Licenses and Tools
Lesson 1: Adding Communication Matrices and Bus Configurations to the ConfigurationDesk Application on page 65	Lesson_1_Start	-
Lesson 2: Configuring Bus Communication for Simulation Purposes on page 75	Lesson_1_Result	▪ CAN module license (CFD_I_CAN) ▪ LIN module license (CFD_I_LIN)
Lesson 3: Working with an Existing Behavior Model on page 97	Lesson_2_Result	▪ CAN module license (CFD_I_CAN) ▪ LIN module license (CFD_I_LIN)
Lesson 4: Generating Bus Simulation Containers on page 109	Lesson_3_Result	▪ CAN module license (CFD_I_CAN) ▪ LIN module license (CFD_I_LIN)
Lesson 5: Building a Real-Time Application on page 115	▪ *Lesson_3_Result* ▪ Lesson_4_Result	▪ Bus Manager in ConfigurationDesk ▪ CAN module license (CFD_I_CAN) ▪ LIN module license (CFD_I_LIN)
Lesson 6: Configuring Bus Communication for Inspection Purposes on page 125	▪ Lesson_2_Result ▪ *Lesson_3_Result* ▪ Lesson_4_Result ▪ Lesson_5_Result	▪ CAN module license (CFD_I_CAN) ▪ LIN module license (CFD_I_LIN)

Lesson	Suitable Start Application	Required Licenses and Tools
Lesson 7: Configuring Bus Communication for Manipulation Purposes on page 133	▪ Lesson_2_Result ▪ Lesson_3_Result ▪ Lesson_4_Result ▪ Lesson_5_Result ▪ *Lesson_6_Result*	▪ CAN module license (CFD_I_CAN) ▪ LIN module license (CFD_I_LIN)
Lesson 8: Working Without a Behavior Model on page 145	Lesson_1_Result	▪ CAN module license (CFD_I_CAN) ▪ LIN module license (CFD_I_LIN)
Lesson 9: Generating a Simulink Model and Synchronizing the Model Interfaces in ConfigurationDesk and Simulink on page 155	Lesson_2_Result	▪ MATLAB/Simulink ▪ Model Interface Package for Simulink ▪ CAN module license (CFD_I_CAN) ▪ LIN module license (CFD_I_LIN)

Additional Lesson: Experimenting with ControlDesk on page 175	Lesson_ExpCD_Start (only required for building a real-time application)	▪ ControlDesk
		▪ ControlDesk Bus Navigator Module
		▪ For building and executing a real-time application:
		  ▪ ConfigurationDesk
		  ▪ CAN module license (CFD_I_CAN)
		  ▪ LIN module license (CFD_I_LIN)
		  ▪ SCALEXIO or MicroAutoBox III real-time hardware
		▪ For building and executing an offline simulation application:
		  ▪ VEOS Player
		  ▪ VEOS

Workflows for Using the Bus Manager

Where to go from here

Information in this section

Workflows for Configuring Bus Communication and Generating Bus Simulation Containers....28

Workflows for Configuring Bus Communication and Building Real-Time Applications....42

Workflows for Configuring Bus Communication and Generating Bus Simulation Containers

Introduction

You can use the Bus Manager to configure CAN and LIN communication for simulation, inspection, and manipulation purposes and generate bus simulation containers (BSC files). You can use BSC files in the VEOS Player to build an offline simulation application for VEOS, for example.

Where to go from here

Information in this section

Configuring Bus Communication for Generating Bus Simulation Containers with Existing Behavior Models....28

Configuring Bus Communication for Generating Bus Simulation Containers with a New Simulink Implementation Container....32

Configuring Bus Communication for Generating Bus Simulation Containers Without Behavior Models....38

Configuring Bus Communication for Generating Bus Simulation Containers with Existing Behavior Models

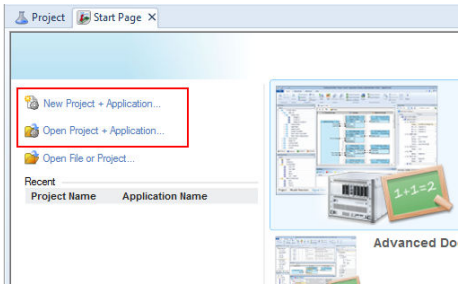
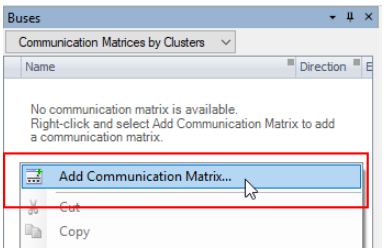
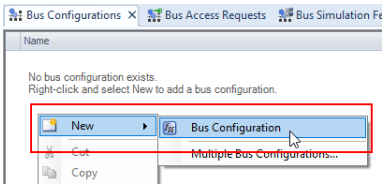
Introduction

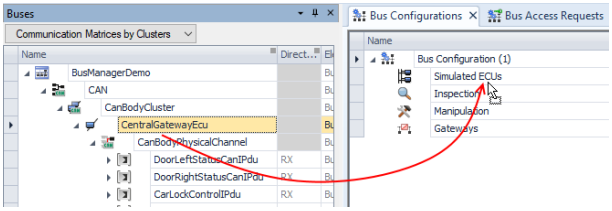
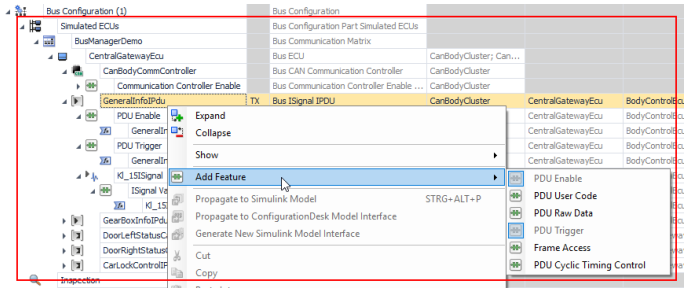
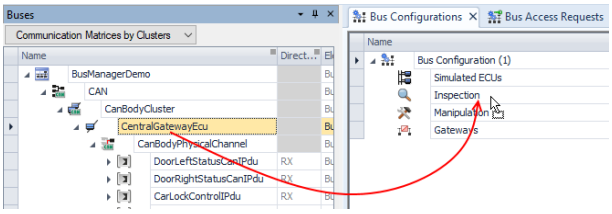
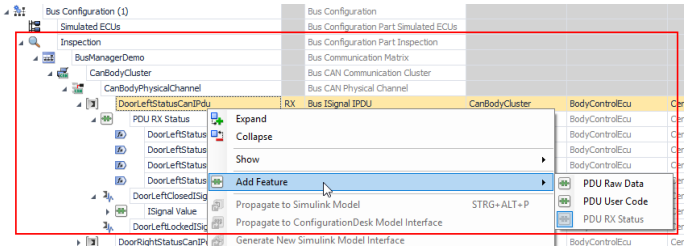
You need a behavior model if signal values must change dynamically during run time, e.g., to dynamically respond to signals received from an ECU under test. To generate bus simulation containers, the model must be a Simulink implementation container (SIC file).

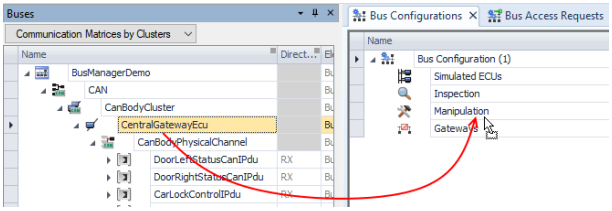
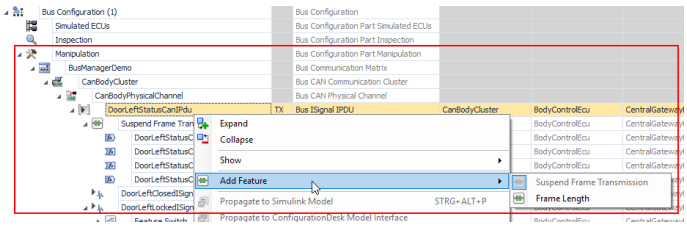
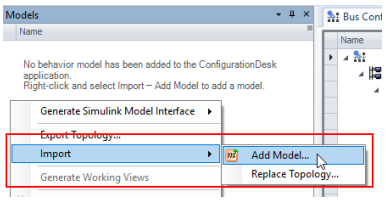
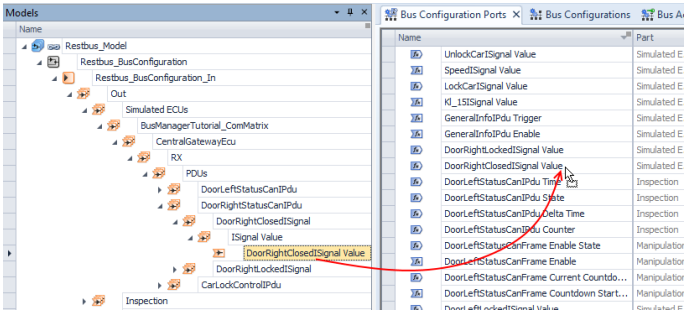
If you already have a suitable SIC file, you can add the file to the ConfigurationDesk application and map the bus configuration function ports that provide the required signals to the model. Then, you can generate bus simulation containers.

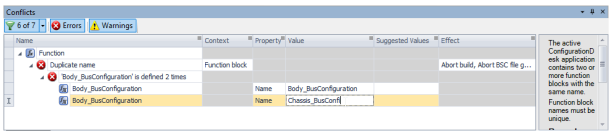
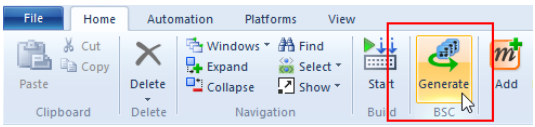
Workflow

The following table shows detailed workflow steps and indicates the related topics and lessons of this tutorial that guide you through the individual steps.

Step	Work	Related Topic/Lesson
1	Create a new or open an existing ConfigurationDesk project and application. 	How to Open the BusManagerTutorial Project on page 59
2	Add communication matrices and bus configurations to the ConfigurationDesk application.	Lesson 1: Adding Communication Matrices and Bus Configurations to the ConfigurationDesk Application on page 65
	1 Add a communication matrix. 	
	2 Add a bus configuration. 	
3	Assign communication matrix elements to bus configurations for simulation, inspection, and/or manipulation purposes.	
	Simulation	Lesson 2: Configuring Bus Communication for

Step	Work	Related Topic/Lesson
	1 Assign communication matrix elements to the Simulated ECUs part of a bus configuration. 	[Simulation Purposes](#) on page 75
	2 Configure bus communication for simulation by using bus configuration features. 	
	Inspection	[Lesson 6: Configuring Bus Communication for Inspection Purposes](#) on page 125
	1 Assign communication matrix elements to the Inspection part of a bus configuration. 	
	2 Configure bus communication for inspection by using bus configuration features. 	
	Manipulation	[Lesson 7: Configuring Bus Communication for](#)

Step	Work	Related Topic/Lesson
	1 Assign communication matrix elements to the Manipulation part of a bus configuration.  2 Configure bus communication for manipulation by using bus configuration features. 	Manipulation Purposes on page 133
4	Add a behavior model, i.e., a Simulink implementation container (SIC file), to the ConfigurationDesk application and map it to a bus configuration.	Lesson 3: Working with an Existing Behavior Model on page 97
	1 Add a Simulink implementation container (SIC file). 	
	2 Map model ports of the Simulink implementation container to ports of a bus configuration. 	

Step	Work	Related Topic/Lesson
5	Generate bus simulation containers.	Lesson 4: Generating Bus Simulation Containers on page 109
	1 Check the **Conflicts Viewer** for conflicts and resolve them, if necessary. 	
	2 Generate bus simulation containers (BSC files). 	

Configuring Bus Communication for Generating Bus Simulation Containers with a New Simulink Implementation Container

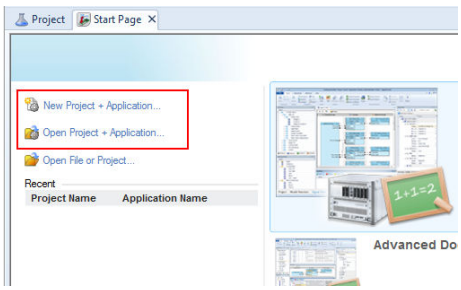
Introduction

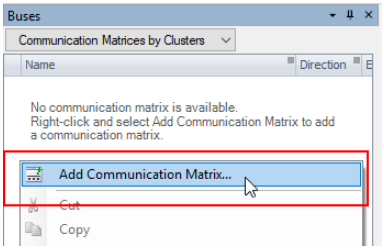
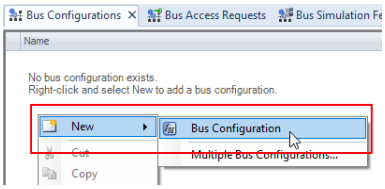
You need a behavior model if signal values must change dynamically during run time, e.g., to dynamically respond to signals received from an ECU under test. To generate bus simulation containers, the model must be a Simulink implementation container (SIC file).

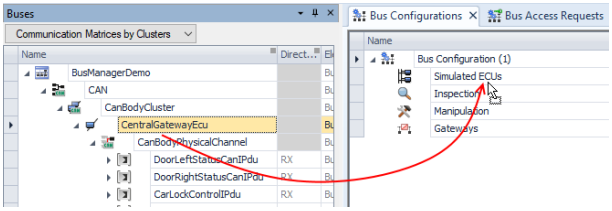
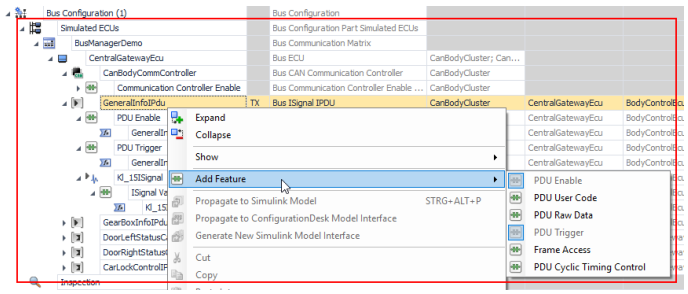
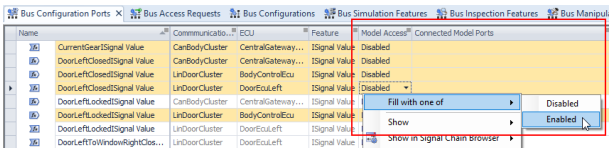
If MATLAB/Simulink and the Model Interface Package for Simulink are installed, you can generate a Simulink model interface for the configured bus communication. You can use the generated blocks as the basis for modeling the behavior in Simulink. After you synchronized the model interfaces of the ConfigurationDesk application and in Simulink, you can generate a SIC file from the Simulink model, add the file to the ConfigurationDesk application, and generate bus simulation containers.

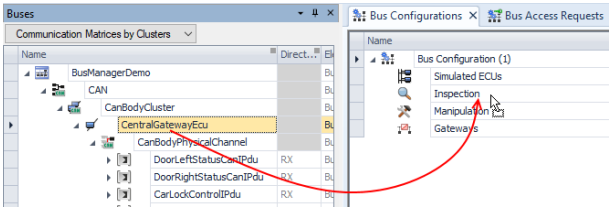
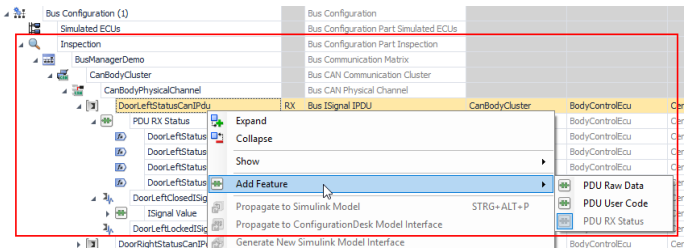
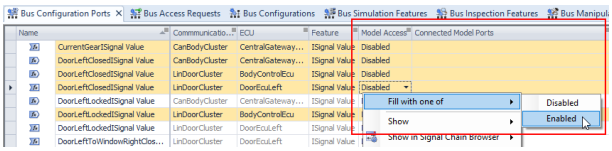
Workflow

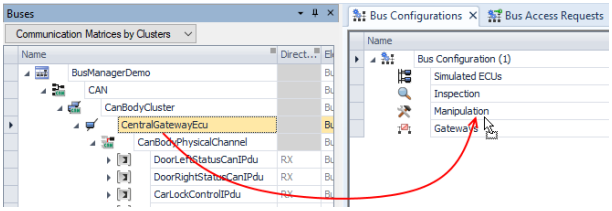
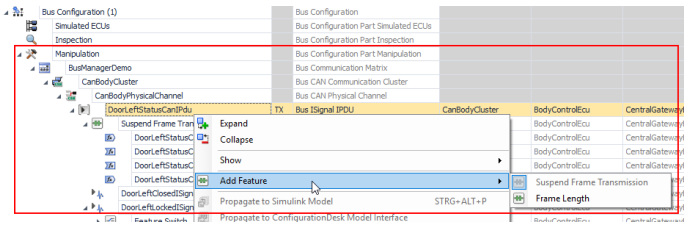
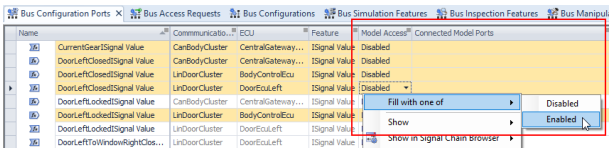
The following table shows detailed workflow steps and indicates the related topics and lessons of this tutorial that guide you through the individual steps.

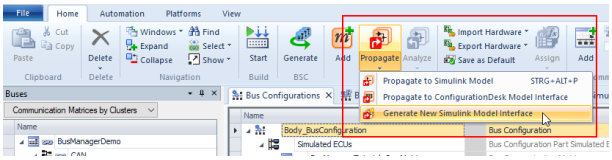
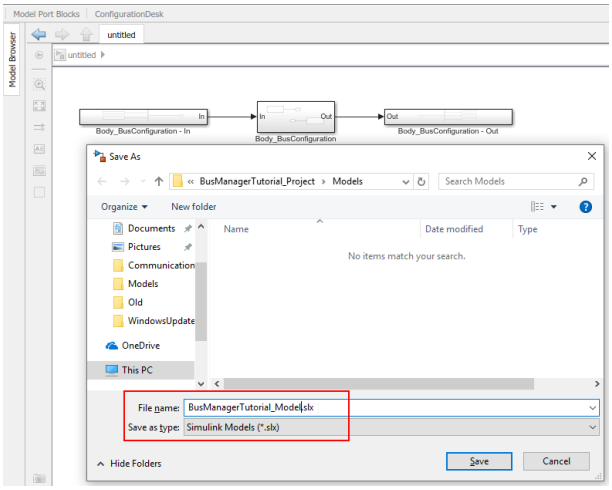
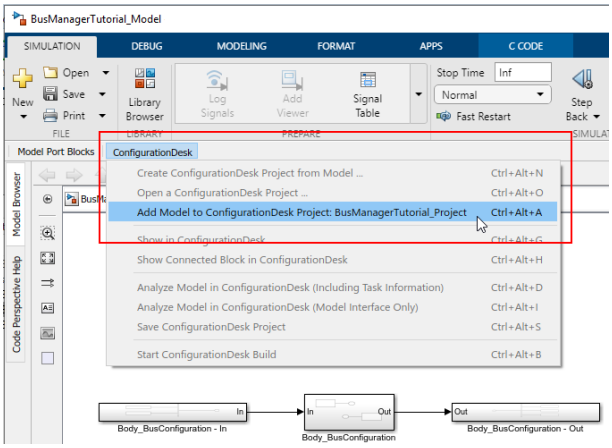
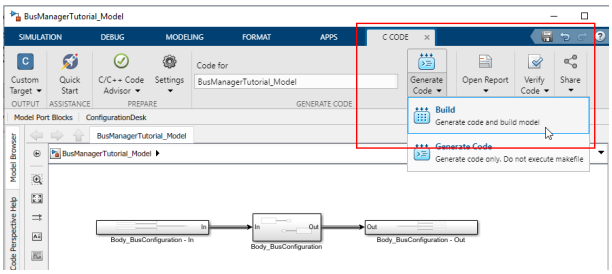
Step	Work	Related Topic/Lesson
1	Create a new or open an existing ConfigurationDesk project and application. 	How to Open the BusManagerTutorial Project on page 59

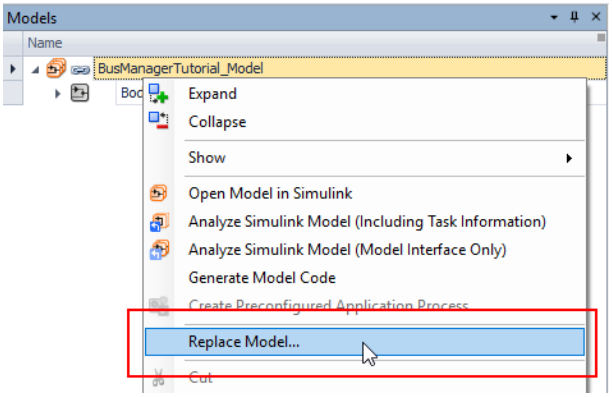
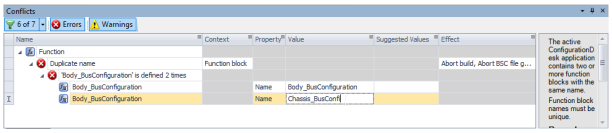
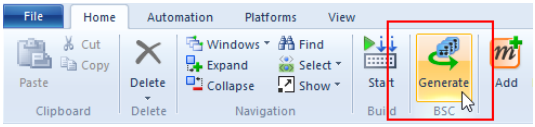
Step	Work	Related Topic/Lesson
2	Add communication matrices and bus configurations to the ConfigurationDesk application.	Lesson 1: Adding Communication Matrices and Bus Configurations to the ConfigurationDesk Application on page 65
	1 Add a communication matrix. 	
	2 Add a bus configuration. 	
3	Assign communication matrix elements to bus configurations for simulation, inspection, and/or manipulation purposes.	
	Simulation	Lesson 2: Configuring Bus Communication for

Step	Work	Related Topic/Lesson
	1 Assign communication matrix elements to the Simulated ECUs part of a bus configuration. 	Simulation Purposes on page 75
	2 Configure bus communication for simulation by using bus configuration features. 	
	3 Enable model access for the function ports whose signals you want to use in the Simulink implementation container. 	
	Inspection	Lesson 6: Configuring Bus Communication for

Step	Work	Related Topic/Lesson
	1 Assign communication matrix elements to the Inspection part of a bus configuration. 	Inspection Purposes on page 125
	2 Configure bus communication for inspection by using bus configuration features. 	
	3 Enable model access for the function ports whose signals you want to use in the Simulink implementation container. 	
	Manipulation	Lesson 7: Configuring Bus Communication for

Step	Work	Related Topic/Lesson
	1 Assign communication matrix elements to the Manipulation part of a bus configuration. 	[Manipulation Purposes](#) on page 133
	2 Configure bus communication for manipulation by using bus configuration features. 	
	3 Enable model access for the function ports whose signals you want to use in the Simulink implementation container. 	
4	Generate a Simulink implementation container (SIC file) and add it to the ConfigurationDesk application.	[Lesson 9: Generating a Simulink Model and Synchronizing the Model Interfaces in](#)

Step	Work	Related Topic/Lesson
	1 Generate a new Simulink model interface. 	ConfigurationDesk and Simulink on page 155
	2 Save the Simulink model. 	
	3 Add the Simulink model to the ConfigurationDesk application. 	
	4 Build a Simulink implementation container (SIC file) from the Simulink model. 	

Step	Work	Related Topic/Lesson
	5 In the ConfigurationDesk application, replace the Simulink model with the Simulink implementation container. 	
5	Generate bus simulation containers.	Lesson 4: Generating Bus Simulation Containers on page 109
	1 Check the **Conflicts Viewer** for conflicts and resolve them, if necessary. 	
	2 Generate bus simulation containers (BSC files). 	

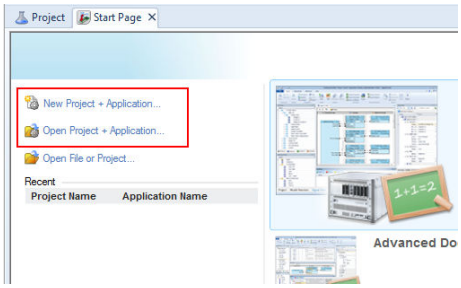
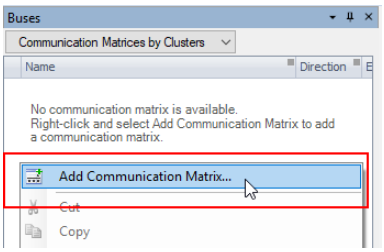
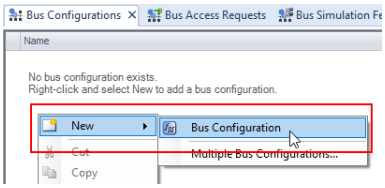
Configuring Bus Communication for Generating Bus Simulation Containers Without Behavior Models

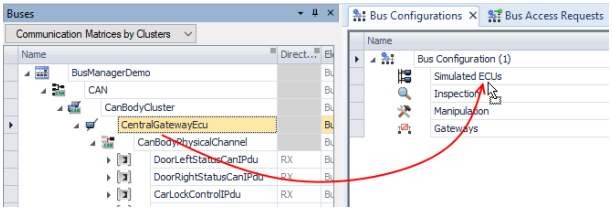
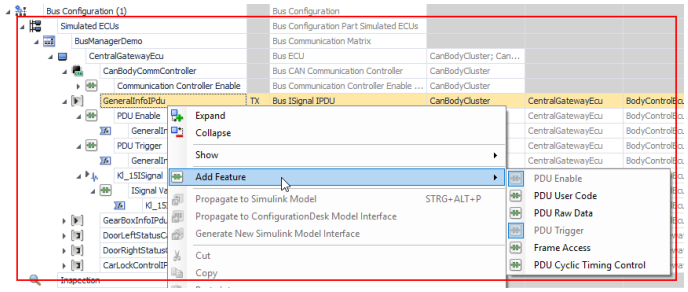
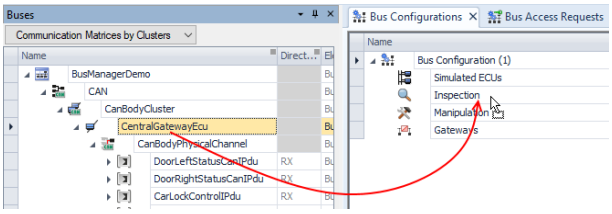
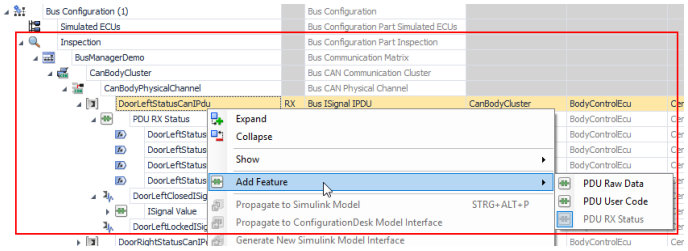
Introduction

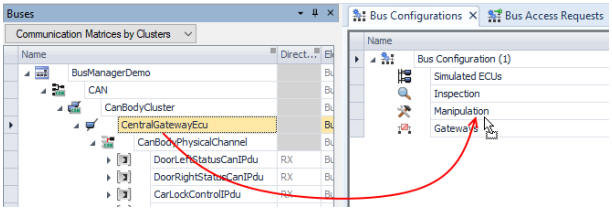
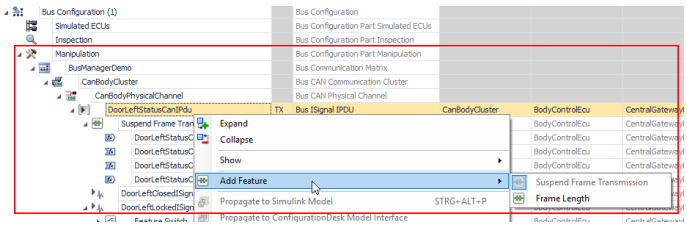
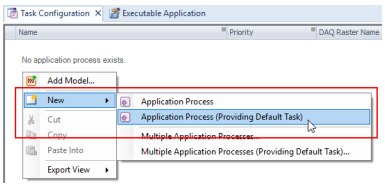
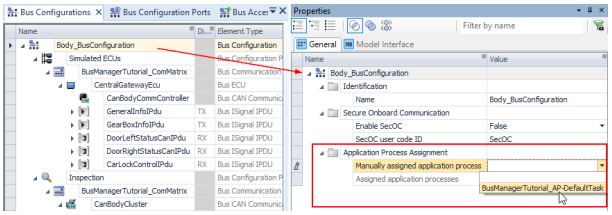
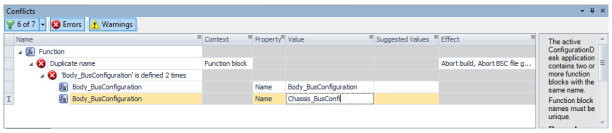
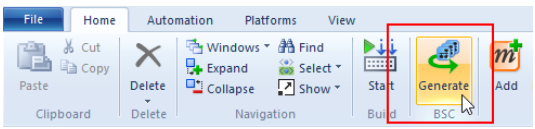
You can generate bus simulation containers without using a behavior model if signal values do not have to change dynamically during run time. For example, you do not need a behavior model if you only want to inspect the response of an ECU under test to manipulated static signal values.

Workflow

The following table shows detailed workflow steps and indicates the related topics and lessons of this tutorial that guide you through the individual steps.

Step	Work	Related Topic/Lesson
1	Create a new or open an existing ConfigurationDesk project and application. 	How to Open the BusManagerTutorial Project on page 59
2	Add communication matrices and bus configurations to the ConfigurationDesk application.	Lesson 1: Adding Communication Matrices and Bus Configurations to the ConfigurationDesk Application on page 65
	1 Add a communication matrix. 	
	2 Add a bus configuration. 	
3	Assign communication matrix elements to bus configurations for simulation, inspection, and/or manipulation purposes.	
	Simulation	Lesson 2: Configuring Bus Communication for

Step	Work	Related Topic/Lesson
	1 Assign communication matrix elements to the Simulated ECUs part of a bus configuration. 	Simulation Purposes on page 75
	2 Configure bus communication for simulation by using bus configuration features. 	
	Inspection	Lesson 6: Configuring Bus Communication for Inspection Purposes on page 125
	1 Assign communication matrix elements to the Inspection part of a bus configuration. 	
	2 Configure bus communication for inspection by using bus configuration features. 	
	Manipulation	Lesson 7: Configuring Bus Communication for

Step	Work	Related Topic/Lesson
	1 Assign communication matrix elements to the Manipulation part of a bus configuration.  2 Configure bus communication for manipulation by using bus configuration features. 	Manipulation Purposes on page 133
4	Create an application process that provides a default task and assign the bus configuration to the application process.	Lesson 8: Working Without a Behavior Model on page 145
	1 Create an application process that provides a default task. 	
	2 Assign the bus configuration to the application process. 	
5	Generate bus simulation containers.	Lesson 4: Generating Bus Simulation Containers on page 109
	1 Check the Conflicts Viewer for conflicts and resolve them, if necessary. 	
	2 Generate bus simulation containers (BSC files). 	

Workflows for Configuring Bus Communication and Building Real-Time Applications

Introduction

You can use the Bus Manager to configure CAN and LIN communication for simulation, inspection, and manipulation purposes, and implement the communication in real-time applications for SCALEXIO or MicroAutoBox III systems.

Where to go from here

Information in this section

Configuring Bus Communication for Real-Time Applications and Using Existing Behavior Models....42

Configuring Bus Communication for Real-Time Applications and Generating a New Simulink Model....47

Configuring Bus Communication for Real-Time Applications Without Behavior Models....53

Configuring Bus Communication for Real-Time Applications and Using Existing Behavior Models

Introduction

You need a behavior model if signal values must change dynamically during run time, e.g., to dynamically respond to signals received from an ECU under test.

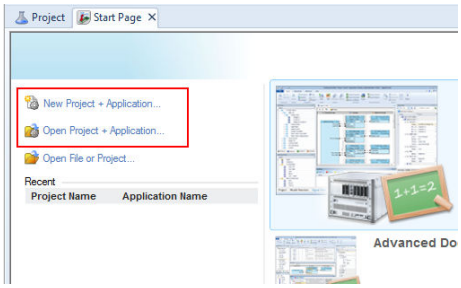
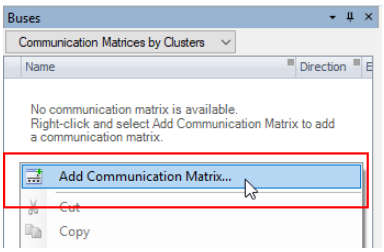
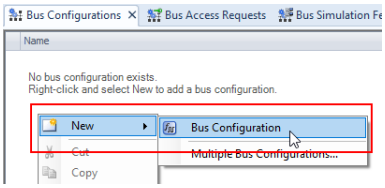
If you already have a suitable behavior model, you can add the model to the ConfigurationDesk application and map the bus configuration function ports that provide the required signals to the model. Then, you can implement the bus communication in a real-time application.

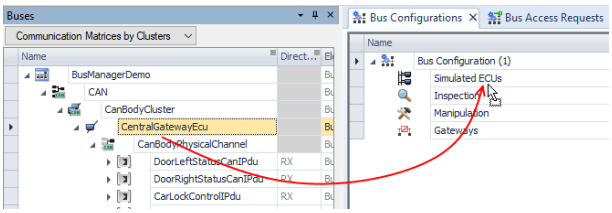
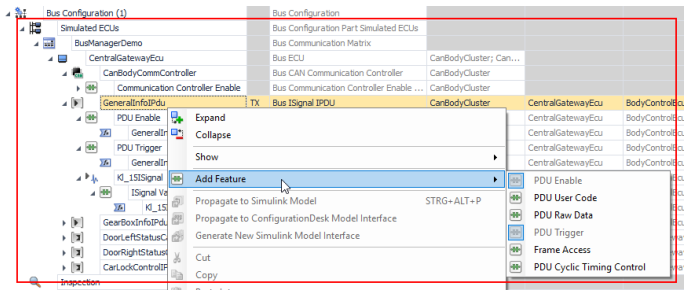
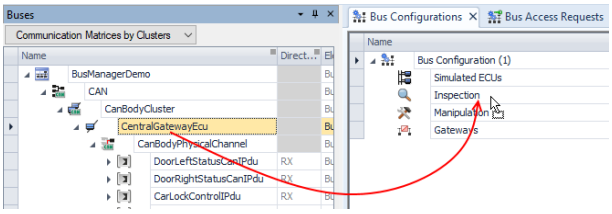
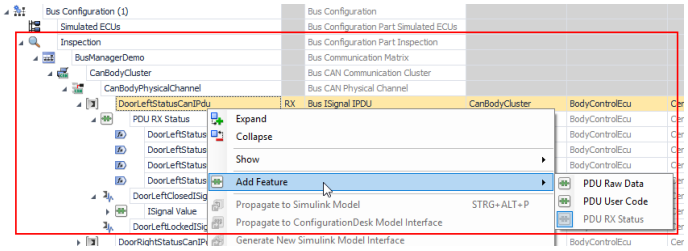
Note

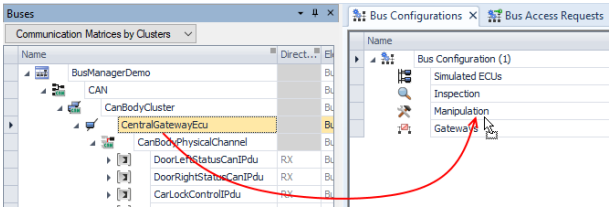
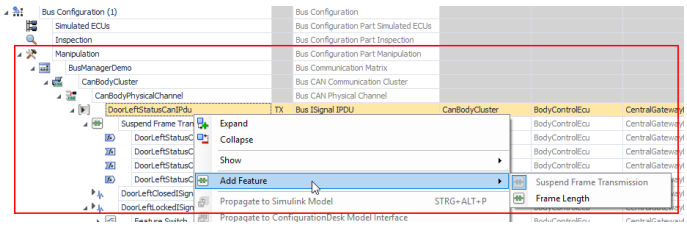
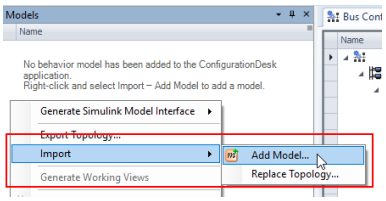
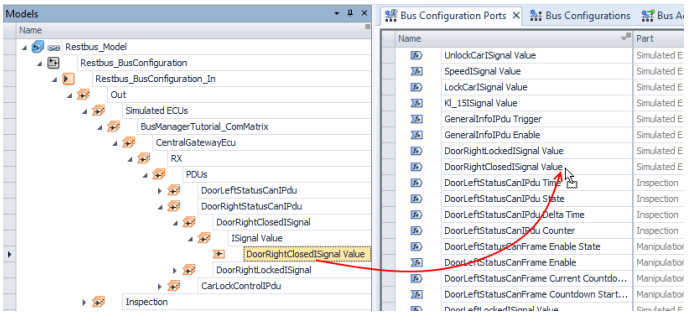
This use scenario applies only to the Bus Manager in ConfigurationDesk, not to the Bus Manager (stand-alone).

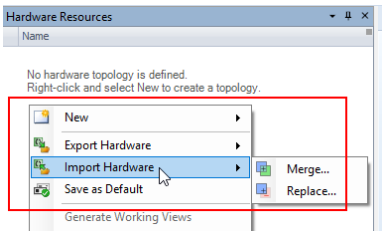
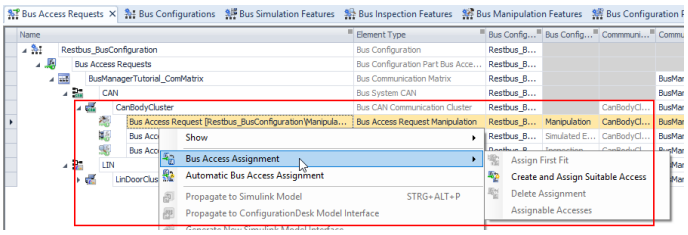
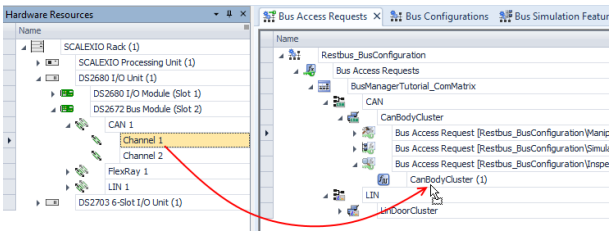
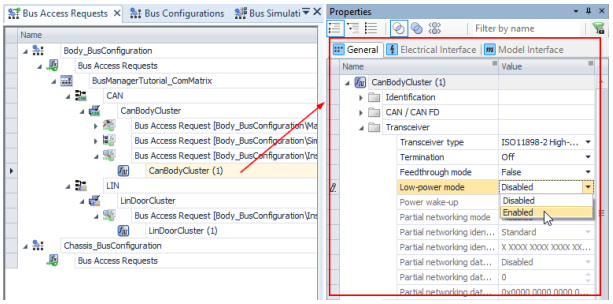
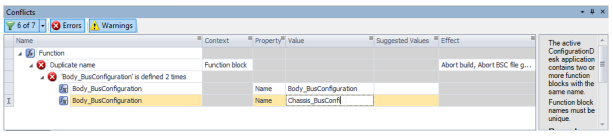
Workflow

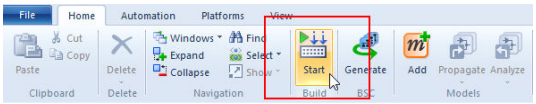
The following table shows detailed workflow steps and indicates the related topics and lessons of this tutorial that guide you through the individual steps.

Step	Work	Related Topic/Lesson
1	Create a new or open an existing ConfigurationDesk project and application. 	How to Open the BusManagerTutorial Project on page 59
2	Add communication matrices and bus configurations to the ConfigurationDesk application.	Lesson 1: Adding Communication Matrices and Bus Configurations to the ConfigurationDesk Application on page 65
	1 Add a communication matrix. 	
	2 Add a bus configuration. 	
3	Assign communication matrix elements to bus configurations for simulation, inspection, and/or manipulation purposes.	
	Simulation	Lesson 2: Configuring Bus Communication for

Step	Work	Related Topic/Lesson
	1 Assign communication matrix elements to the Simulated ECUs part of a bus configuration. 	Simulation Purposes on page 75
	2 Configure bus communication for simulation by using bus configuration features. 	
	Inspection	Lesson 6: Configuring Bus Communication for Inspection Purposes on page 125
	1 Assign communication matrix elements to the Inspection part of a bus configuration. 	
	2 Configure bus communication for inspection by using bus configuration features. 	
	Manipulation	Lesson 7: Configuring Bus Communication for

Step	Work	Related Topic/Lesson
	1 Assign communication matrix elements to the Manipulation part of a bus configuration.  2 Configure bus communication for manipulation by using bus configuration features. 	Manipulation Purposes on page 133
4	Add a behavior model to the ConfigurationDesk application and map it to a bus configuration. 1 Add a behavior model.  2 Map model ports of the behavior model to ports of a bus configuration. 	Lesson 3: Working with an Existing Behavior Model on page 97

Step	Work	Related Topic/Lesson
5	Specify access to real-time hardware and build a real-time application.	Lesson 5: Building a Real-Time Application on page 115
	1 Add a hardware topology to the ConfigurationDesk application. 	
	2 Specify the access to the bus for the bus configuration. To do this, assign the bus access requests of the bus configuration to bus accesses, i.e., to bus function blocks (CAN or LIN function blocks). 	
	3 Assign hardware resources to the bus function blocks. 	
	4 Configure the bus function blocks according to your requirements. 	
	5 Check the Conflicts Viewer for conflicts and resolve them, if necessary. 	

Step	Work	Related Topic/Lesson
	6 Start the build process. 	

Configuring Bus Communication for Real-Time Applications and Generating a New Simulink Model

Introduction

You need a behavior model if signal values must change dynamically during run time, e.g., to dynamically respond to signals received from an ECU under test.

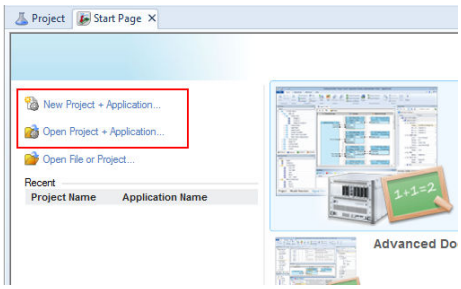
If MATLAB/Simulink and the Model Interface Package for Simulink are installed, you can generate a Simulink model interface for the configured bus communication. You can use the generated blocks as the basis for modeling the behavior in Simulink. After you synchronized the model interfaces of the ConfigurationDesk application and in Simulink, you can implement the bus communication in a real-time application.

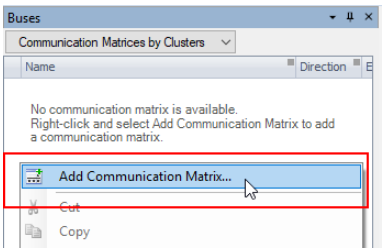
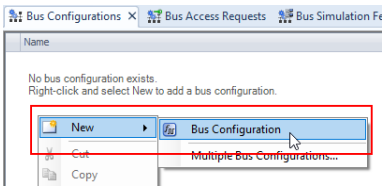
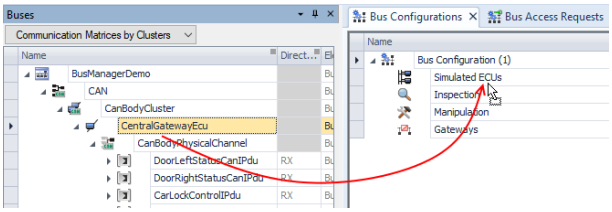
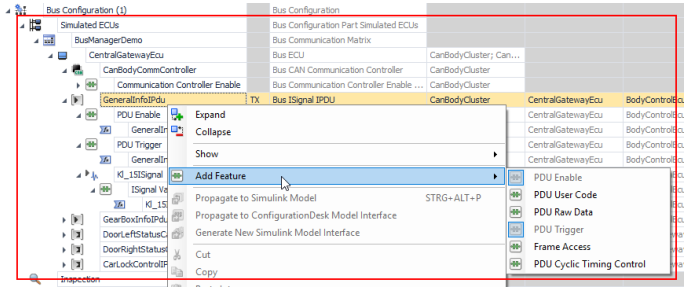
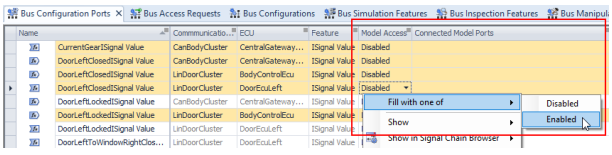
Note

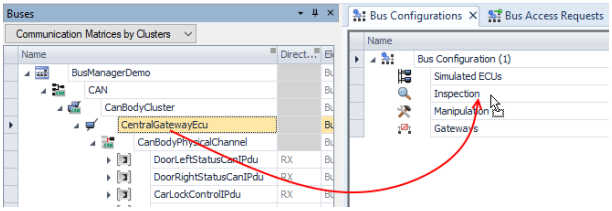
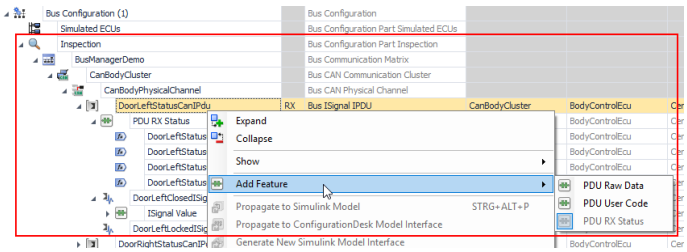
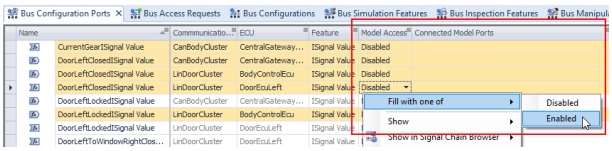
This use scenario applies only to the Bus Manager in ConfigurationDesk, not to the Bus Manager (stand-alone).

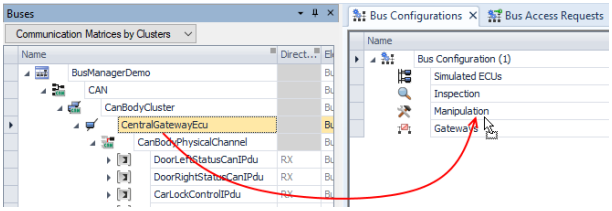
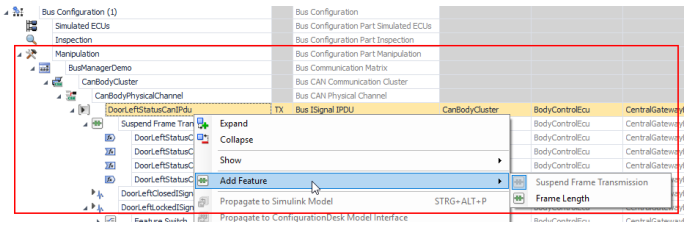
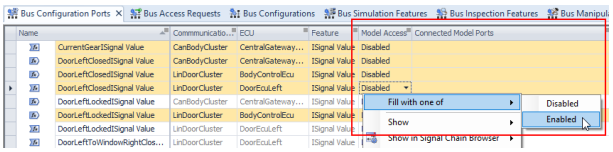
Workflow

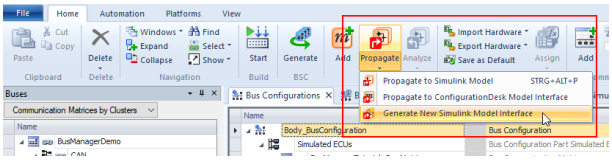
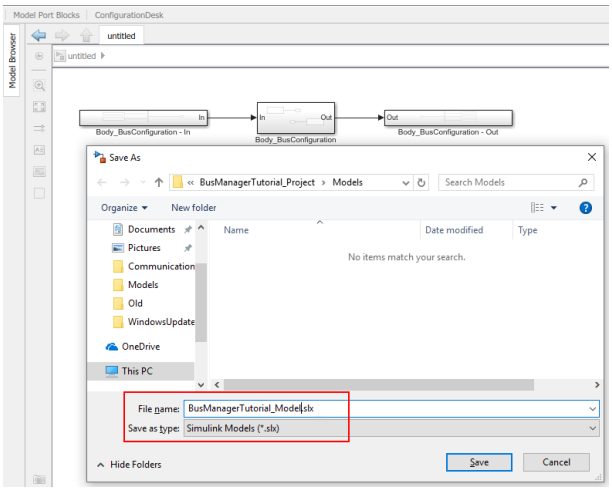
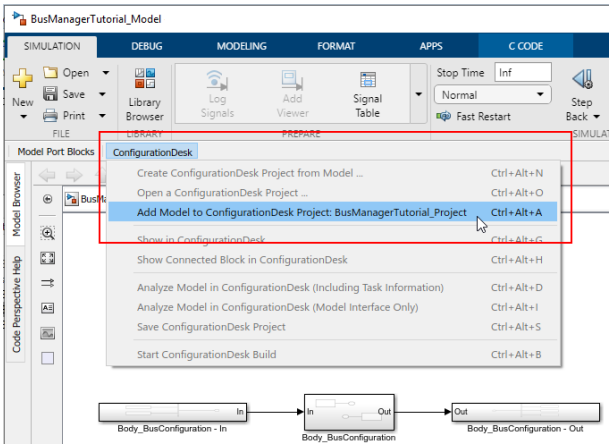
The following table shows detailed workflow steps and indicates the related topics and lessons of this tutorial that guide you through the individual steps.

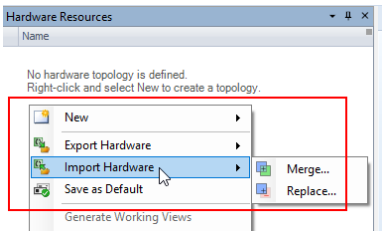
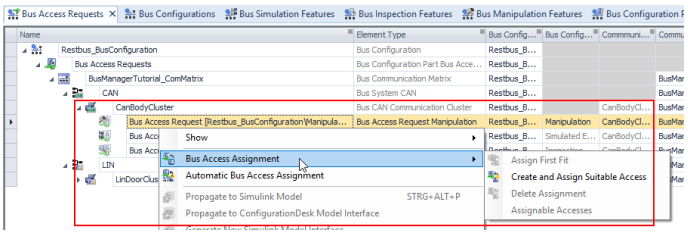
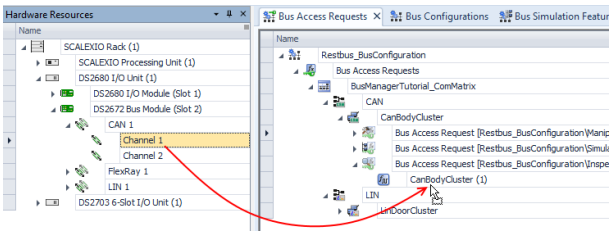
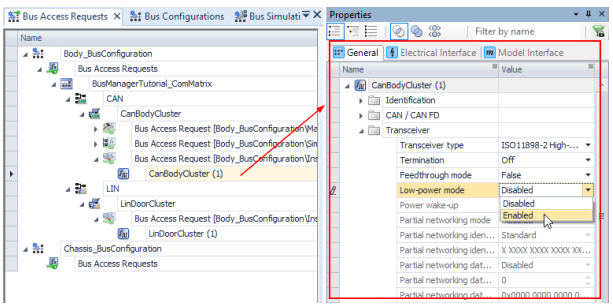
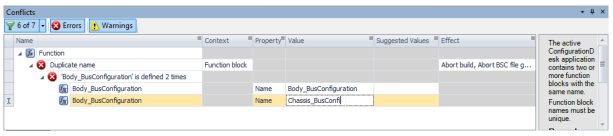
Step	Work	Related Topic/Lesson
1	Create a new or open an existing ConfigurationDesk project and application. 	How to Open the BusManagerTutorial Project on page 59
2	Add communication matrices and bus configurations to the ConfigurationDesk application.	Lesson 1: Adding Communication Matrices and Bus Configurations to

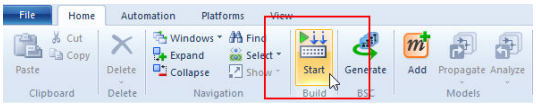
Step	Work	Related Topic/Lesson
	1 Add a communication matrix. 	the ConfigurationDesk Application on page 65
	2 Add a bus configuration. 	
3	Assign communication matrix elements to bus configurations for simulation, inspection, and/or manipulation purposes.	
	Simulation	Lesson 2: Configuring Bus Communication for Simulation Purposes on page 75
	1 Assign communication matrix elements to the Simulated ECUs part of a bus configuration. 	
	2 Configure bus communication for simulation by using bus configuration features. 	
	3 Enable model access for the function ports whose signals you want to use in the Simulink model. 	
	Inspection	Lesson 6: Configuring Bus Communication for

Step	Work	Related Topic/Lesson
	1 Assign communication matrix elements to the Inspection part of a bus configuration. 	Inspection Purposes on page 125
	2 Configure bus communication for inspection by using bus configuration features. 	
	3 Enable model access for the function ports whose signals you want to use in the Simulink model. 	
	Manipulation	Lesson 7: Configuring Bus Communication for

Step	Work	Related Topic/Lesson
	1 Assign communication matrix elements to the Manipulation part of a bus configuration. 	[Manipulation Purposes](#) on page 133
	2 Configure bus communication for manipulation by using bus configuration features. 	
	3 Enable model access for the function ports whose signals you want to use in the Simulink model. 	
4	Generate a Simulink model and synchronize the model interfaces of the ConfigurationDesk application and in Simulink.	[Lesson 9: Generating a Simulink Model and Synchronizing the Model Interfaces in](#)

Step	Work	Related Topic/Lesson
	1 Generate a new Simulink model interface. 	ConfigurationDesk and Simulink on page 155
	2 Save the Simulink model. 	
	3 Add the Simulink model to the ConfigurationDesk application. 	

Step	Work	Related Topic/Lesson
5	Specify access to real-time hardware and build a real-time application.	Lesson 5: Building a Real-Time Application on page 115
	1 Add a hardware topology to the ConfigurationDesk application. 	
	2 Specify the access to the bus for the bus configuration. To do this, assign the bus access requests of the bus configuration to bus accesses, i.e., to bus function blocks (CAN or LIN function blocks). 	
	3 Assign hardware resources to the bus function blocks. 	
	4 Configure the bus function blocks according to your requirements. 	
	5 Check the Conflicts Viewer for conflicts and resolve them, if necessary. 	

Step	Work	Related Topic/Lesson
	6 Start the build process. 	

Configuring Bus Communication for Real-Time Applications Without Behavior Models

Introduction

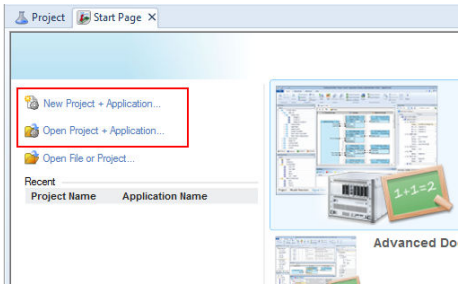
You can implement bus communication in a real-time application without using a behavior model if signal values do not have to change dynamically during run time. For example, you do not need a behavior model if you only want to inspect the response of an ECU under test to manipulated static signal values.

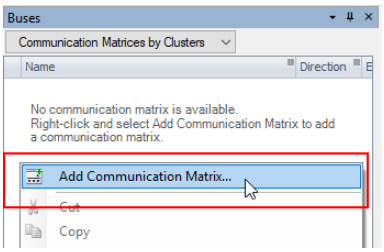
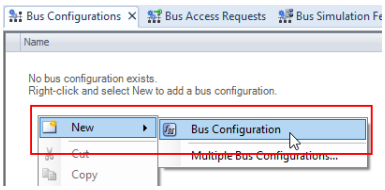
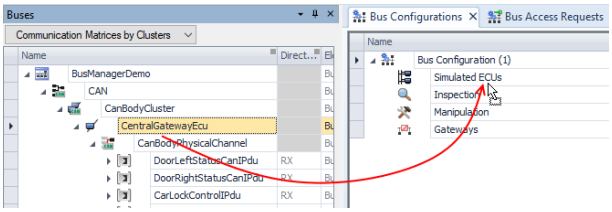
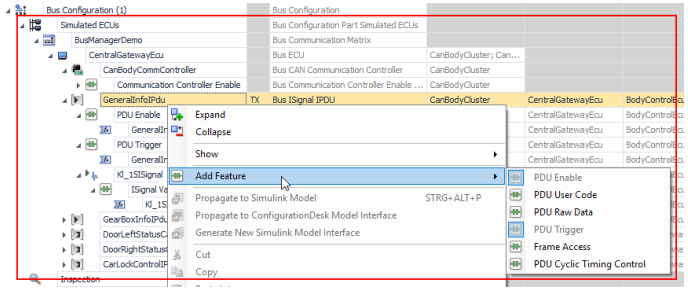
Note

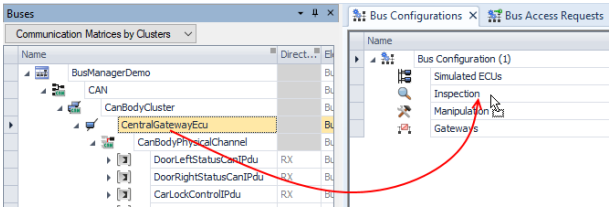
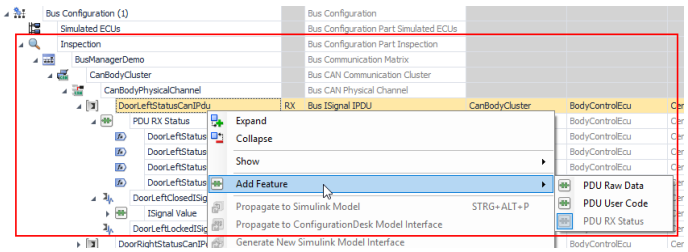
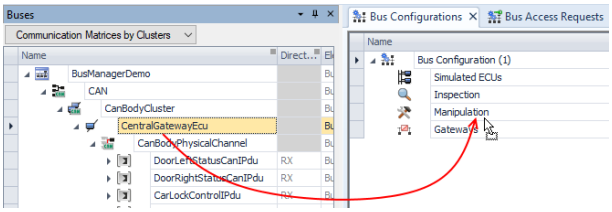
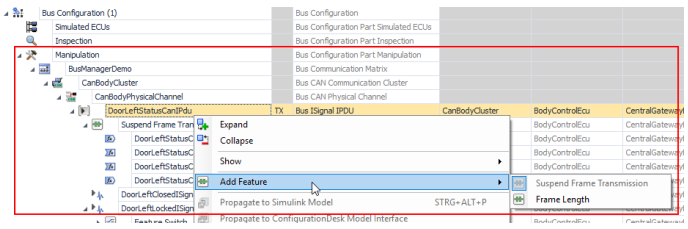
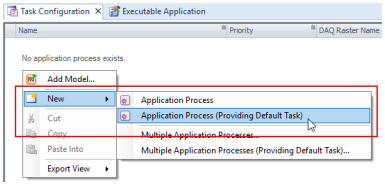
This use scenario applies only to the Bus Manager in ConfigurationDesk, not to the Bus Manager (stand-alone).

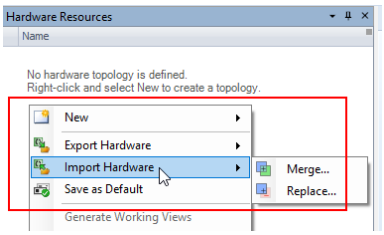
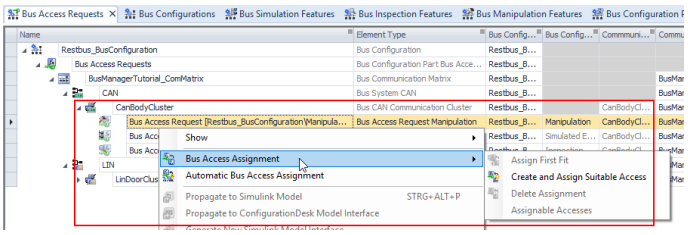
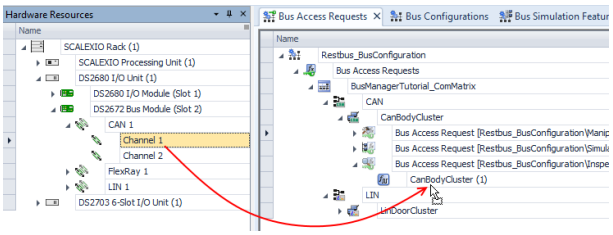
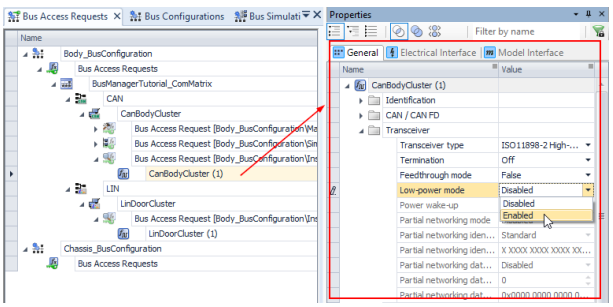
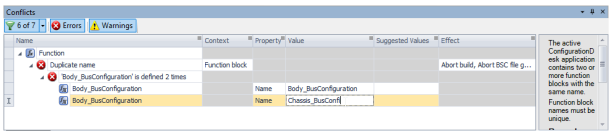
Workflow

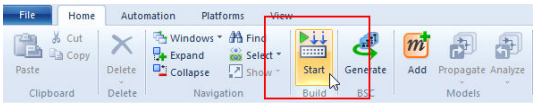
The following table shows detailed workflow steps and indicates the related topics and lessons of this tutorial that guide you through the individual steps.

Step	Work	Related Topic/Lesson
1	Create a new or open an existing ConfigurationDesk project and application. 	How to Open the BusManagerTutorial Project on page 59
2	Add communication matrices and bus configurations to the ConfigurationDesk application.	Lesson 1: Adding Communication Matrices and Bus Configurations to

Step	Work	Related Topic/Lesson
	1 Add a communication matrix. 	the ConfigurationDesk Application on page 65
	2 Add a bus configuration. 	
3	Assign communication matrix elements to bus configurations for simulation, inspection, and/or manipulation purposes.	
	Simulation	Lesson 2: Configuring Bus Communication for Simulation Purposes on page 75
	1 Assign communication matrix elements to the Simulated ECUs part of a bus configuration. 	
	2 Configure bus communication for simulation by using bus configuration features. 	
	Inspection	Lesson 6: Configuring Bus Communication for

Step	Work	Related Topic/Lesson
	1 Assign communication matrix elements to the Inspection part of a bus configuration.  2 Configure bus communication for inspection by using bus configuration features. 	Inspection Purposes on page 125
	Manipulation 1 Assign communication matrix elements to the Manipulation part of a bus configuration.  2 Configure bus communication for manipulation by using bus configuration features. 	Lesson 7: Configuring Bus Communication for Manipulation Purposes on page 133
4	Create an application process that provides a default task. 	Lesson 8: Working Without a Behavior Model on page 145

Step	Work	Related Topic/Lesson
5	Specify access to real-time hardware and build a real-time application.	Lesson 5: Building a Real-Time Application on page 115
	1 Add a hardware topology to the ConfigurationDesk application. 	
	2 Specify the access to the bus for the bus configuration. To do this, assign the bus access requests of the bus configuration to bus accesses, i.e., to bus function blocks (CAN or LIN function blocks). 	
	3 Assign hardware resources to the bus function blocks. 	
	4 Configure the bus function blocks according to your requirements. 	
	5 Check the Conflicts Viewer for conflicts and resolve them, if necessary. 	

Step	Work		Related Topic/Lesson
	6	Start the build process. 	

Working with the BusManagerTutorial Project

Where to go from here

Information in this section

How to Open the BusManagerTutorial Project....59
How to Activate an Application in the BusManagerTutorial Project....61
How to Identify Accessible Licenses....62

How to Open the BusManagerTutorial Project

Objective

To work through the Bus Manager tutorial, you must open the BusManagerTutorial project.

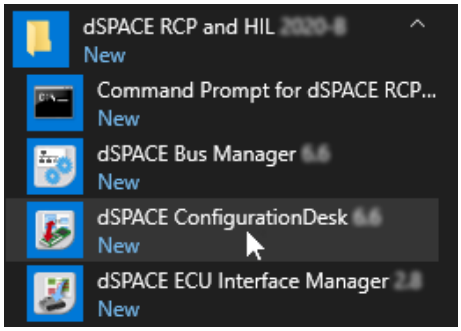
Method

To open the BusManagerTutorial project

1 Depending on your Bus Manager variant, choose one of the following:

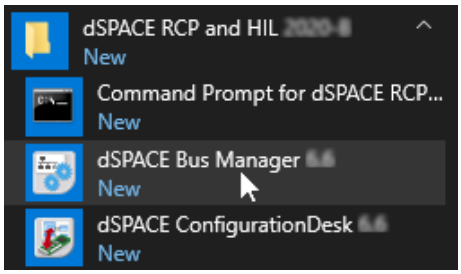
- Bus Manager in ConfigurationDesk:

  From the Start menu, choose dSPACE RCP and HIL <dSPACE Release>, and click dSPACE ConfigurationDesk <x.y>.



- Bus Manager (stand-alone):

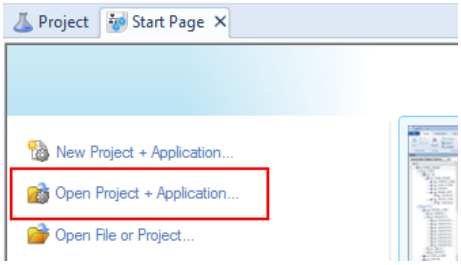
  From the Start menu, choose dSPACE RCP and HIL <dSPACE Release>, and click dSPACE Bus Manager <x.y>.



  <dSPACE Release> and *<x.y>* are placeholders for the dSPACE Release and the software version, because your host PC can have multiple installations and multiple items in the Start menu.

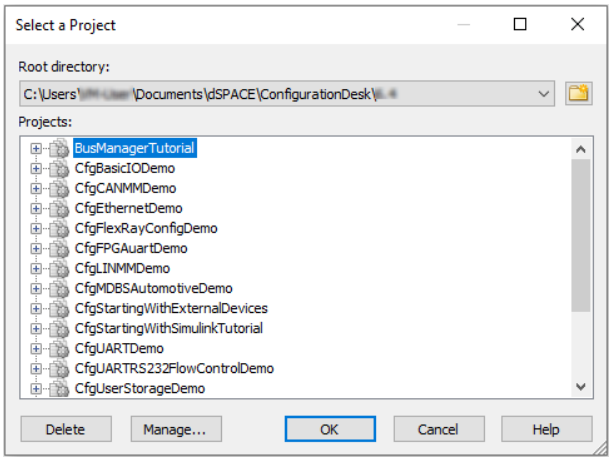
  ConfigurationDesk or the Bus Manager (stand-alone) opens.

2. On the Start Page, click Open Project + Application.



   The Select a Project dialog opens.

3. Select the BusManagerTutorial project and click OK.



Tip

If you work with the Bus Manager (stand-alone), only the BusManagerTutorial project is displayed in the dialog.

4 Only if you work with the Bus Manager (stand-alone): A **Project Manager** dialog opens, informing you that the project was saved with a different product. In the dialog, click **Yes**.

Result

You opened the **BusManagerTutorial** project. The **Lesson_1_Start** application is active. You can now start with lesson 1. Refer to Lesson 1: Adding Communication Matrices and Bus Configurations to the ConfigurationDesk Application on page 65.

If you want to start with another lesson, you must activate the result application of the related previous lesson. Refer to How to Activate an Application in the BusManagerTutorial Project on page 61.

How to Activate an Application in the BusManagerTutorial Project

Objective

To work with a specific application of the **BusManagerTutorial** project, you must activate it. The active application is displayed in bold letters.

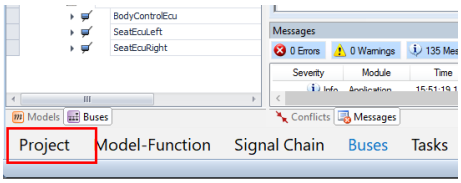
Precondition

The **BusManagerTutorial** project is open.

Method

To activate an application in the BusManagerTutorial project

1 Switch to the **Project** view set.



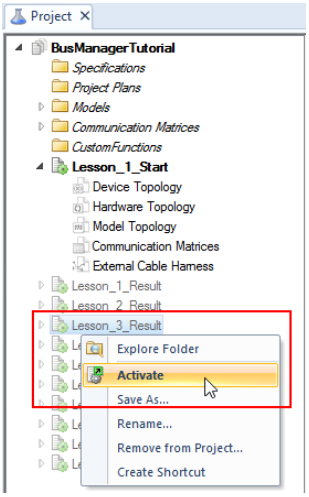
Tip

If you use the Bus Manager (stand-alone), not all of the displayed view sets are available.

The **Project Manager** is open.

2 In the **Project Manager**, right-click the inactive application you want to activate.

3 From the context menu, select Activate.



Result

You activated the selected application. The previously active application is automatically deactivated. You can now use the active application, for example, as the starting point for a related subsequent lesson.

Note

Depending on the activated application, specific licenses are required. These licenses must be available and accessible.

- For an overview of the required licenses, refer to Overview of Lessons on page 25.
- For an instruction to identify the accessible licenses, refer to How to Identify Accessible Licenses on page 62.

How to Identify Accessible Licenses

Objective

To work with an application of the BusManagerTutorial project, all of the required licenses must be available and accessible.

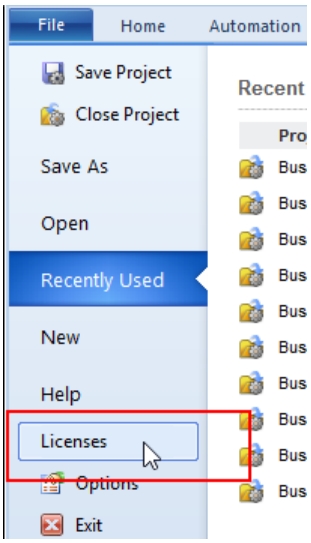
Precondition

The BusManagerTutorial project is open.

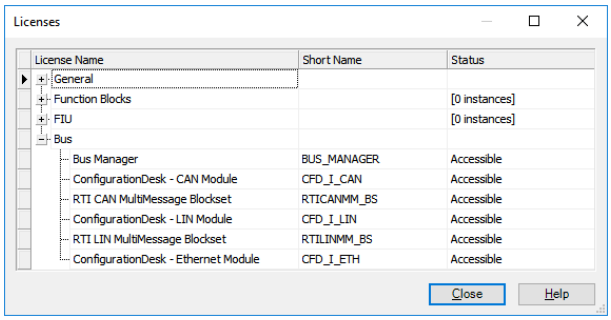
Method

To identify accessible licenses

1 On the File ribbon, click Licenses.



The Licenses dialog opens, displaying the accessible and used licenses.



Result

You opened the Licenses dialog, which lets you identify the accessible licenses.

To work with the Bus Manager, the Bus Manager license (BUS_MANAGER) is mandatory. The additional required licenses for working with the BusManagerTutorial project depend on the specific tutorial lesson. For an overview of the required licenses, refer to Overview of Lessons on page 25.

Tip

You can request evaluation licenses to complete this tutorial with the Bus Manager. For more information, contact dSPACE Support (www.dspace.com/go/supportrequest).

Lesson 1: Adding Communication Matrices and Bus Configurations to the ConfigurationDesk Application

Where to go from here

Information in this section

Overview of Lesson 1 65

Step1: How to Add a Communication Matrix to the ConfigurationDesk Application 66

Step 2: How to Check the Communication Matrix for Conflicts 68

Step 3: How to Add a Bus Configuration to the ConfigurationDesk Application 71

Result of Lesson 1 72

Overview of Lesson 1

Using the Bus Manager

The basis for using the Bus Manager is communication matrices and bus configurations.

What you will learn

In this lesson, you will learn how to add communication matrices and bus configurations to a ConfigurationDesk application and check communication matrices for conflicts.

Before you begin

Before you begin this lesson, the following precondition must be met:

The **Lesson_1_Start** application is active.

- This is the case if you opened the **BusManagerTutorial** project for the first time. Refer to How to Open the BusManagerTutorial Project on page 59.
- If you have to activate the application, refer to How to Activate an Application in the BusManagerTutorial Project on page 61.

Steps

This lesson contains the following steps:

- Step1: How to Add a Communication Matrix to the ConfigurationDesk Application on page 66
- Step 2: How to Check the Communication Matrix for Conflicts on page 68
- Step 3: How to Add a Bus Configuration to the ConfigurationDesk Application on page 71

Result and further information

The lesson concludes with a summary of the lesson content and with further information. Refer to Result of Lesson 1 on page 72.

Duration

Completing this lesson will take you about 15 minutes.

Step1: How to Add a Communication Matrix to the ConfigurationDesk Application

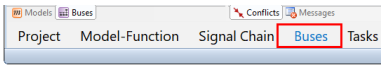
Objective

To configure bus communication with the Bus Manager, you must add one or more communication matrices to the ConfigurationDesk application.

Method

To add a communication matrix to the ConfigurationDesk application

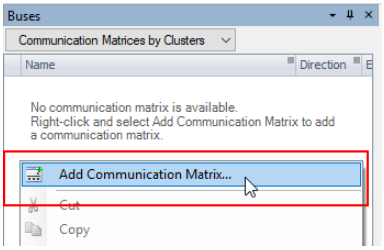
1 Switch to the **Buses** view set.



Tip

If you use the Bus Manager (stand-alone), not all of the displayed view sets are available.

2 Right-click in the **Buses Browser** and select **Add Communication Matrix** from the context menu.



The **Add Communication Matrix** dialog opens.

3 In the dialog, navigate to the **Communication Matrices** folder of the **BusManagerTutorial** project. The path to the folder depends on your Bus Manager variant:

- Bus Manager in ConfigurationDesk:

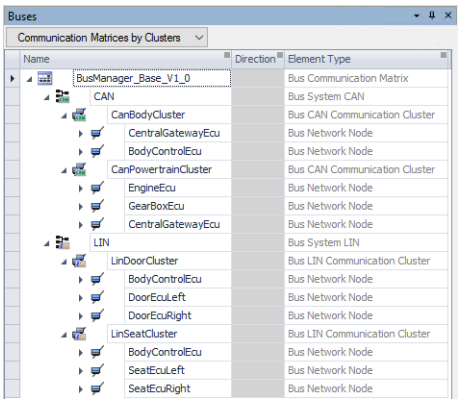
  `%USERPROFILE%\Documents\dSPACE\ConfigurationDesk\<VersionNumber>\Tutorials\BusManagerTutorial\Communication Matrices`

- Bus Manager (stand-alone):

  `%USERPROFILE%\Documents\dSPACE\BusManager\<VersionNumber>\Tutorials\BusManagerTutorial\Communication Matrices`

4 Select `BusManager_Base_V1_0.arxml` and click **Open**.

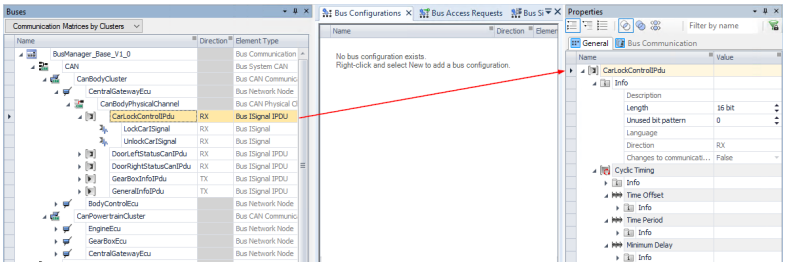
The communication matrix is added to the ConfigurationDesk application and displayed in the **Buses Browser**.



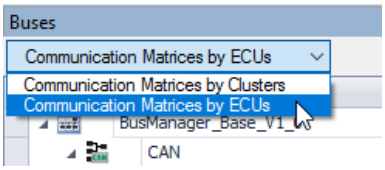
The **Buses Browser** displays the communication matrix elements sorted by communication clusters. The displayed network nodes let you access the bus communication of each ECU for only one cluster.

5 In the **Buses Browser**, explore the displayed communication matrix elements:

- Click the expand arrow (▸) of the elements to display their lower-level elements.
- Select elements to display their properties in the **Properties Browser**.



6 In the Buses Browser, select Communication Matrices by ECUs.



The Buses Browser displays the communication matrix elements sorted by ECUs. The displayed ECU nodes let you access the complete bus communication of each ECU, regardless of the involved communication clusters.

Result

You added the BusManager_Base_V1_0.arxml communication matrix from the Communication Matrices folder of the BusManagerTutorial project to the ConfigurationDesk application. Additionally, you explored the communication matrix elements and the different views of the elements in the Buses Browser.

Tip

You can back up ConfigurationDesk projects. When you do this, the communication matrices that are located in the Communication Matrices folder are automatically included in the generated ZIP archive.

What's next

In the next step, you will check the communication matrix for conflicts.

Step 2: How to Check the Communication Matrix for Conflicts

Objective

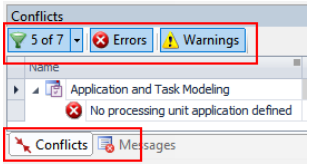
To ensure that there are no severe faults in the communication matrix, you should check it for conflicts after you added it to the ConfigurationDesk application.

Method

To check the communication matrix for conflicts

1 Select the Conflicts Viewer.

The Conflicts Viewer displays all conflicts of a ConfigurationDesk application. You can enable and disable the evaluation of conflicts via filters. Selected filters are highlighted in blue.

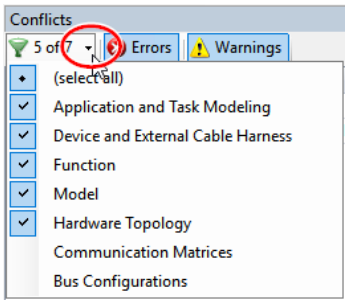


Tip

The selection of filters is stored locally on the host PC and applies to all ConfigurationDesk applications. Therefore, the selected filters in your ConfigurationDesk application might differ from the selected filters displayed above.

2 Make sure that the Errors and Warnings filters are selected: If a filter is not highlighted in blue, click the filter to select it.

3 Click the expand arrow of the context set filter.



The context sets are displayed. All conflicts of a ConfigurationDesk application are assigned to specific context sets. Conflicts of communication matrices are assigned to the Communication Matrices context set. Conflicts are evaluated only if the related context set is selected. Selected context sets are indicated by a check mark.

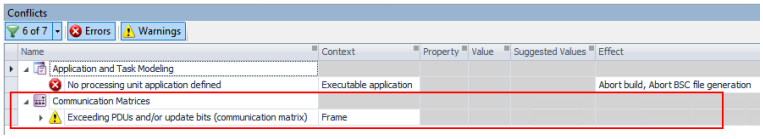
Tip

- The selection of the context sets is stored locally on the host PC.
- If you work with the Bus Manager (stand-alone), not all of the context sets that are displayed above are available.

4 Select the Communication Matrices context set.

5 Click the expand arrow to close the context set filter.

The **Communication Matrices** context set is selected. One communication matrix conflict is displayed.



Communication matrix conflicts have no effects on building a real-time application or generating bus simulation containers. In most cases, you therefore do not have to resolve them. However, if you assign conflicting communication matrix elements to a bus configuration, bus configuration conflicts occur which might affect the build process or the generation of bus simulation containers.

Independently of the assigned elements, the following communication matrix conflicts might result in unintended behavior at run time:

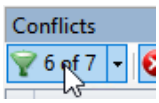
- Failed consistency checks during import of communication matrix
- Failed schema check during import of communication matrix

If these conflicts occur, it is recommended to resolve them before you assign elements to bus configurations. Because the displayed communication matrix conflict is not one of these two conflicts, you do not have to resolve it.

Tip

- When you select the communication matrix node in the **Buses Browser**, the **Properties Browser** displays the **Consistency state** and **Schema validation state** properties. The property values indicate whether the communication matrix successfully passed the consistency and schema checks during the import.
- You will resolve the communication matrix conflict in lesson 2.
- If the **Application and Task Modeling** context set is selected, the **Conflicts Viewer** displays a conflict that affects the build process and the generation of bus simulation containers. You will automatically resolve this conflict by configuring the bus communication in the following lessons.

6 Click the context set filter.



All context sets are cleared. The evaluation of all conflicts is disabled and the **Conflicts Viewer** displays no conflicts.

Result

You checked the communication matrix for conflicts. Then, you disabled the evaluation of all conflicts.

Note

The evaluation of conflicts can significantly influence the performance of the ConfigurationDesk application. Depending on the complexity of the bus communication, this especially applies to Bus Manager-related conflicts. It is therefore recommended to enable the conflict evaluation only when and as long as required.

What's next

In the next step, you will add a bus configuration to the ConfigurationDesk application.

Step 3: How to Add a Bus Configuration to the ConfigurationDesk Application

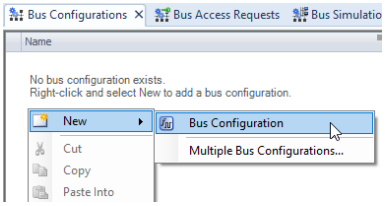
Objective

To configure bus communication for simulation, inspection, and manipulation purposes, you must add one or more bus configurations to the ConfigurationDesk application.

Method

To add a bus configuration to the ConfigurationDesk application

1 Right-click in an empty area of the Bus Configurations table and select New - Bus Configuration from the context menu.

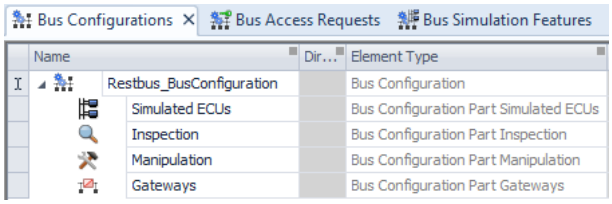


A new bus configuration is added to the ConfigurationDesk application.

2 In the Name column of the Bus Configurations table, select Bus Configuration (1) and change the name to Restbus_BusConfiguration.

Result

You added a new bus configuration to the ConfigurationDesk application and changed its default name. The Bus Configurations table displays the bus configuration and its Simulated ECUs, Inspection, Manipulation, and Gateways parts.



What's next

With this step, you finished the lesson. For a summary of the lesson and for more information, refer to Result of Lesson 1 on page 72.

Result of Lesson 1

Result

In this lesson, you learned the basic principles of adding communication matrices and bus configurations to ConfigurationDesk applications, and checking communication matrices for conflicts:

- The **Buses Browser** provides different views of communication matrices and their elements.
- You can access the properties of selected elements in the **Properties Browser**.
- Each bus configuration provides bus configuration parts.
- To easily identify bus configurations, you can specify user-defined names for bus configurations.
- The **Conflicts Viewer** displays all conflicts of the ConfigurationDesk application and lets you enable and disable the evaluation of conflicts. For optimum performance, it is recommended to enable the conflict evaluation only as required.
- In general, communication matrix conflicts have no effect on building a real-time application or generating bus simulation containers. However, conflicts that result from failed schema and/or consistency checks might result in unintended behavior at run time.

Prepared result application

The **Lesson_1_Result** application provides the result of this lesson. You can activate the application to compare it with your own results, for example. For more information, refer to Tutorial project and applications on page 23.

Further information

- For more information on working with communication matrices, refer to:
  - Bus Manager in ConfigurationDesk: Working with Communication Matrices (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Working with Communication Matrices (Bus Manager (Stand-Alone) Implementation Guide 🕮)

- For more information on bus configurations, refer to:
  - Bus Manager in ConfigurationDesk: Basics on Bus Configurations (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Basics on Bus Configurations (Bus Manager (Stand-Alone) Implementation Guide 🕮)
- For more information on Bus Manager-related conflicts, refer to:
  - Bus Manager in ConfigurationDesk: Basics on Bus Manager-Related Conflicts (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Basics on Bus Manager-Related Conflicts (Bus Manager (Stand-Alone) Implementation Guide 🕮)
- For more information on working with common user interface elements such as ribbons and view sets, refer to User Interface of ConfigurationDesk (ConfigurationDesk Real-Time Implementation Guide 🕮).

Where to go from here

You can now assign communication matrix elements to the bus configuration and configure bus communication for simulation purposes. Refer to Lesson 2: Configuring Bus Communication for Simulation Purposes on page 75.

Lesson 2: Configuring Bus Communication for Simulation Purposes

Where to go from here

Information in this section

Overview of Lesson 2 75

Step 1: How to Assign Communication Matrix Elements to the Simulated ECUs Part of a Bus Configuration 76

Step 2: How to Add Bus Configuration Features for Simulation Purposes 79

Step 3: How to Configure Function Ports of Bus Simulation Features 84

Step 4: How to Check the Configured Bus Communication for Conflicts and Resolve Conflicts 89

Result of Lesson 2 93

Overview of Lesson 2

Simulating bus communication

To simulate bus communication, you must assign communication matrix elements to the **Simulated ECUs** part of a bus configuration. By using bus configuration features, you can configure the bus communication to be simulated.

In this lesson, you will configure a restbus simulation. For more information, refer to Use scenarios in this tutorial on page 21.

What you will learn

In this lesson, you will learn how to assign communication matrix elements to the **Simulated ECUs** part of a bus configuration and configure the assigned elements by using bus configuration features. Additionally, you will check the configured bus communication for conflicts and resolve conflicts.

Before you begin

Before you begin this lesson, the following preconditions must be met:

- The following licenses are accessible:
  - CAN module license (CFD_I_CAN)
  - LIN module license (CFD_I_LIN)

  Refer to How to Identify Accessible Licenses on page 62.
- The result of lesson 1 is available and the related application is active.

  If you did not work through lesson 1, you can activate the **Lesson_1_Result** application. Refer to How to Activate an Application in the BusManagerTutorial Project on page 61.

Steps

This lesson contains the following steps:

- Step 1: How to Assign Communication Matrix Elements to the Simulated ECUs Part of a Bus Configuration on page 76
- Step 2: How to Add Bus Configuration Features for Simulation Purposes on page 79
- Step 3: How to Configure Function Ports of Bus Simulation Features on page 84
- Step 4: How to Check the Configured Bus Communication for Conflicts and Resolve Conflicts on page 89

Result and further information

The lesson concludes with a summary of the lesson content and with further information. Refer to Result of Lesson 2 on page 93.

Duration

Completing this lesson will take you about 40 minutes.

Step 1: How to Assign Communication Matrix Elements to the Simulated ECUs Part of a Bus Configuration

Objective

To simulate bus communication, you must assign communication matrix elements to the **Simulated ECUs** part of a bus configuration.

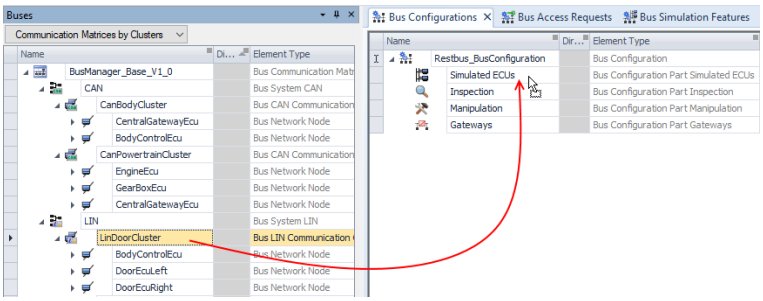
Part 1

To assign a communication cluster to the Simulated ECUs part

1 Switch to the Buses view set, if necessary.

2 In the Buses Browser, select Communication Matrices by Clusters.

3 Drag the LinDoorCluster node to the Simulated ECUs part of the Restbus_BusConfiguration.

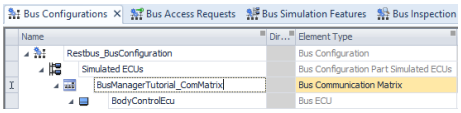


All the network nodes of the LinDoorCluster and their lower-level elements are assigned to the Simulated ECUs part of the bus configuration. In the Buses Browser, the assigned elements are marked by a chain symbol ( ).

Tip

If you dragged the LinDoorCluster to a different bus configuration part by mistake, you can undo the assignment, e.g., by pressing the **Ctrl+Z** keys.

4 In the Name column of the Bus Configurations table, change the name of the communication matrix node to BusManagerTutorial_ComMatrix.

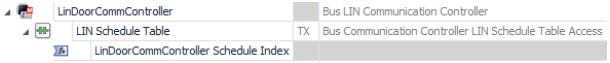


Tip

- It is recommended to use only names without date and version information, especially if you work with automation scripts later on.
- Renaming the communication matrix node in the bus configuration applies only to this bus configuration. It does not apply to other bus configurations or the communication matrix file name.

5 In the Bus Configurations table, expand the BodyControlEcu and DoorEcuLeft nodes to display their assigned lower-level elements.
The BodyControlEcu is the LIN master of the LinDoorCluster, which is indicated by the symbol at the communication controller. The DoorEcuLeft is a LIN slave ( ) of this cluster.

6 Completely expand the LinDoorCommController node.



When you assign a LIN master to the Simulated ECUs part, the LIN Schedule Table bus configuration feature is automatically added to the LIN master communication controller. The feature provides a Schedule Index function port. Function ports let you access data at run time.

Note

If the LIN master is simulated by the Bus Manager, LIN communication is disabled by default. The LIN Schedule Table feature is mandatory to enable LIN communication. To enable LIN communication, you must configure the Schedule Index function port.

Tip

You will configure the Schedule Index function port in step 3 of this lesson.

7 Collapse the DoorEcuLeft node.

Interim result

You assigned all elements of the LinDoorCluster to the Simulated ECUs part and renamed the communication matrix node in the bus configuration. By doing this, you removed the version information from the node name. You also explored the differences between LIN masters and LIN slaves.

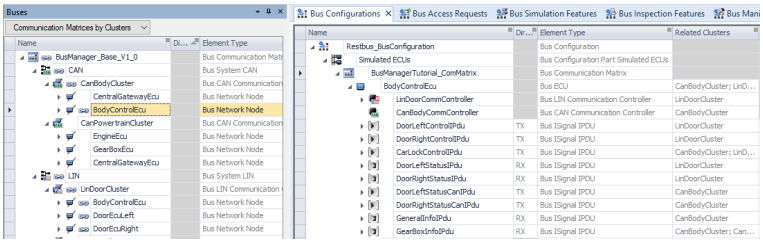
You can now assign further elements to the Simulated ECUs part. Continue with the next part.

Part 2

To assign further elements to the Simulated ECUs part

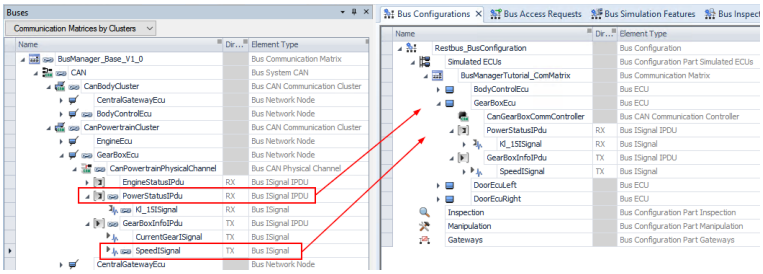
1 In the Buses Browser, select the BodyControlEcu network node of the CanBodyCluster and drag it to the Simulated ECUs part.

All elements of the BodyControlEcu that are relevant for the bus communication via the CanBodyCluster are assigned to the BodyControlEcu node in the bus configuration.



2 Collapse the BodyControlEcu node.

3 From the CanPowertrainCluster, assign the following elements of the GearBoxEcu to the Simulated ECUs part:
   - PowerStatusIPdu
   - SpeedISignal of the GearBoxInfoIPdu



   The selected elements and related higher-level and lower-level elements are assigned to the Simulated ECUs part.

4 In the Bus Configurations table, completely expand the GearBoxEcu node to display all assigned elements.

   For the SpeedISignal and Kl_15ISignal, the ISignal Value feature is available. The ISignal Value feature is automatically added to all ISignals that are assigned to the Simulated ECUs part. This applies regardless of whether you assign the ISignal itself or a related higher-level element. The ISignal Value feature provides a Value function port. You can configure the function port to access the value of the related ISignal at run time, for example.

Result

You assigned different communication matrix elements to the Simulated ECUs part of the bus configuration. Related higher-level and lower-level elements are assigned automatically. You renamed the communication matrix node in the bus configuration, and explored the assigned elements and the bus configuration features that are automatically added to specific elements.

What's next

In the next step, you will manually add further bus configuration features to bus configuration elements to configure bus communication for simulation purposes.

Step 2: How to Add Bus Configuration Features for Simulation Purposes

Objective

To configure bus communication for simulation purposes, you must add bus configuration features to elements that are assigned to the Simulated ECUs part of a bus configuration.

Depending on the bus configuration element, you can add available bus simulation features in the Bus Configurations and Bus Simulation Features tables.

Part 1

To add bus simulation features in the Bus Configurations table

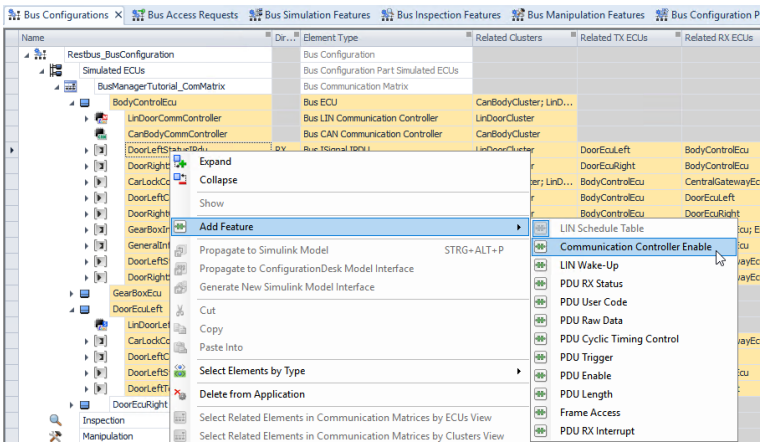
1 In the Bus Configurations table, do the following:

- Collapse the GearBoxEcu node.
- Expand the BodyControlEcu and DoorEcuLeft nodes.

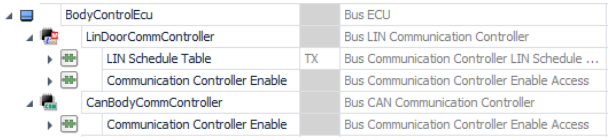
The assigned communication controllers and PDUs of the BodyControlEcu and DoorEcuLeft are displayed.

2 Select both ECUs, including all their displayed elements: Click the BodyControlEcu node, press and hold the **Shift** key, and click the last PDU of DoorEcuLeft.

3 Right-click any of the highlighted elements and select Add Feature - Communication Controller Enable from the context menu.



The Communication Controller Enable feature is added to all communication controllers of both ECUs. In the Bus Configurations table, a Communication Controller Enable feature node ( ) is added to each controller, as shown in the following example.



Tip

- In the context menu, the LIN Schedule Table feature is grayed out because it has already been added to the LIN master communication controller.
- By selecting the ECUs as described above, you might have also selected the GearBoxEcu node. However, the communication controller of the GearBoxEcu is not selected because the GearBoxEcu node is collapsed. Therefore, the Communication Controller Enable feature is not added to this communication controller.

4 Expand a Communication Controller Enable feature node.

The Communication Controller Enable feature provides two function ports. You can configure the function ports to enable or disable the related communication controller at run time, and provide the enable state of the communication controller to a behavior model, for example.

Tip

You will configure function ports in step 3 of this lesson.

Interim result

You added the Communication Controller Enable feature to the communication controllers of the BodyControlEcu and DoorEcuLeft, and explored the function ports that are available for the Communication Controller Enable feature. You can now add further bus simulation features. Continue with the next part.

Part 2

To add bus simulation features in the Bus Simulation Features table

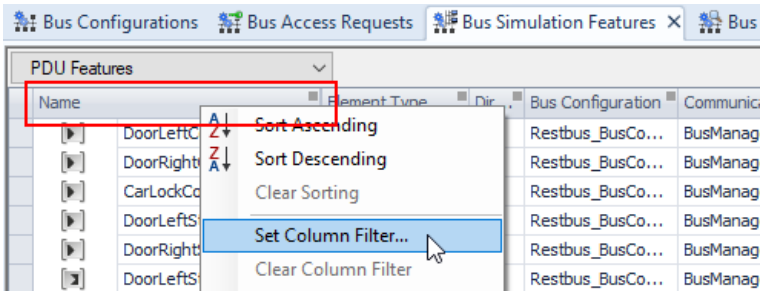
1 Select the Bus Simulation Features table.

The PDU Features subview is active, displaying all PDUs that are assigned to the Simulated ECUs part of bus configurations.

Tip

- The Bus Simulation Features table lets you add bus simulation features only to PDUs and ISignals. For this purpose, it provides two subviews, the PDU Features and Signal Features subviews.
- If you closed the table by mistake, you can open it via Windows on the Home ribbon.

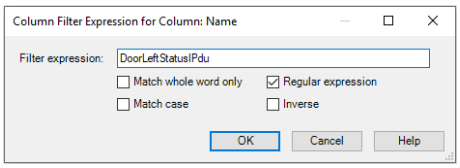
2 In the PDU Features subview, right-click the Name column header and select Set Column Filter from the context menu.



The Column Filter Expressions for Column: Name dialog opens.

3 In the dialog, type **`DoorLeftStatusIPdu`** in the Filter expression edit field.

4 Make sure that only the **Regular expression** checkbox is selected.

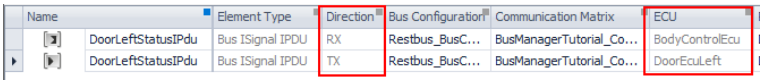


5 Click **OK**.

The dialog closes. The **Name** column is filtered for PDUs whose name contains *DoorLeftStatusIPdu*, i.e., two PDUs:

- One RX PDU of the **BodyControlEcu**.
- One TX PDU of the **DoorEcuLeft**.

The column header indicates the active column filter by a blue square.



Note

The filter is active until you clear it explicitly. This applies even when you close ConfigurationDesk or the Bus Manager (stand-alone).

Tip

- The filter applies to both subviews of the table.
- The order of table columns can be customized. The order is stored locally on the host PC and applies to all ConfigurationDesk applications. Therefore, the order of table columns in your ConfigurationDesk application might differ from the column order displayed in this tutorial.

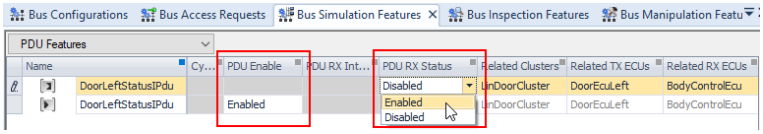
6 Scroll to the right until the **PDU Enable** and **PDU RX Status** columns are displayed.

The **PDU Enable** feature is available only for TX PDUs, the **PDU RX Status** feature only for RX PDUs. Because of this, each feature can be enabled only for one of the displayed PDUs.

Tip

If a column is not available, right-click any column header and select **Column Chooser** from the context menu. From the **Column Chooser**, drag the required column to the table.

7 In the **PDU Enable** and **PDU RX Status** columns, select **Enabled** from the list of the available cell.



The PDU Enable feature is added to the TX DoorLeftStatusIPdu and the PDU RX Status feature is added to the RX DoorLeftStatusIPdu. However, no feature nodes are displayed in this table.

8 Select both DoorLeftStatusIPdu rows.

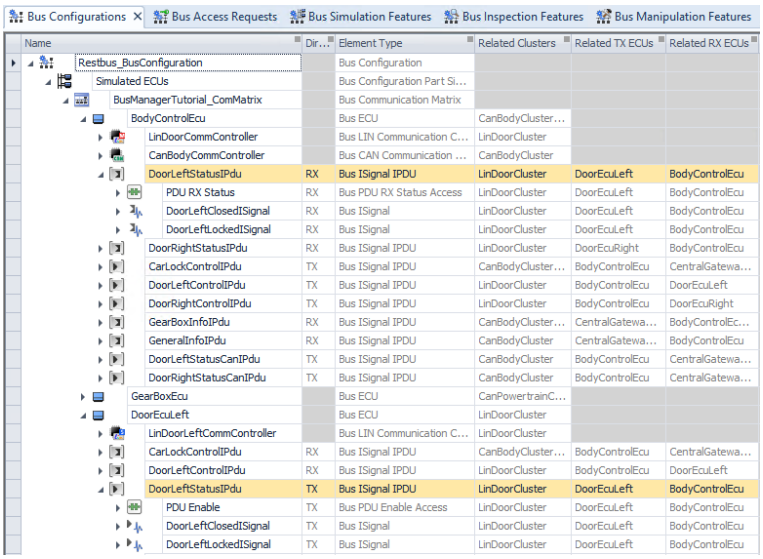
9 Right-click any of the selected cells and select Show - Show in Bus Configurations Table from the context menu.

The Bus Configurations table is displayed, and both DoorLeftStatusIPdu nodes are highlighted in the table.

10 Expand both PDU nodes.

The lower-level elements of the PDUs are displayed, i.e.:

- RX DoorLeftStatusIPdu: The PDU RX Status feature node and the ISignal nodes
- TX DoorLeftStatusIPdu: The PDU Enable feature node and the ISignal nodes



The DoorLeftStatusIPdu is transmitted by the DoorEcuLeft and received by the BodyControlEcu. The PDU Enable feature lets you enable and disable the transmission of the PDU. The PDU RX Status feature lets you observe the status of the received PDU, e.g., whether the PDU is received by the BodyControlEcu.

11 Select the Bus Simulation Features table.

12 Right-click the Name column header and select Set Column Filter from the context menu.

13 In the Column Filter Expressions for Column: Name dialog, specify **`DoorRightStatusIPdu`** as the Filter expression and click OK.

An RX and a TX DoorRightStatusIPdu are displayed.

14 For the PDUs, set the available PDU Enable and PDU RX Status feature to Enabled.

15 Right-click the Name column header and select Clear Column Filter from the context menu.

The specified column filter is cleared and all PDUs that are assigned to the **Simulated ECUs** part are displayed.

Result

You added bus configuration features to elements that are assigned to the **Simulated ECUs** part of the bus configuration. You added the features in the **Bus Configurations** and **Bus Simulation Features** table for multiple selected elements and by using column filters.

Tip

In the **Bus Configurations** table, you can add available bus configuration features to any element type via the context menu.

What's next

In the next step, you will configure some function ports of the added bus simulation features.

Step 3: How to Configure Function Ports of Bus Simulation Features

Objective

To access bus simulation features at run time and to configure their initial behavior, you must configure the available function ports.

Depending on the configured settings, you can access the data of bus simulation features via experiment software (e.g., ControlDesk) and automation scripts, and/or exchange data with a behavior model at run time.

Part 1

To specify an initial LIN schedule table

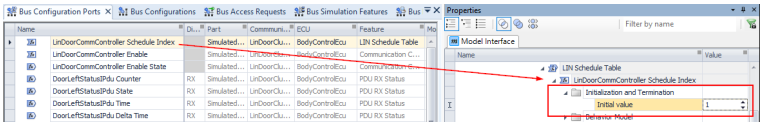
1. Select the **Bus Configuration Ports** table.

   The **Bus Configuration Ports** table displays all the ports that are available for bus configurations of the active ConfigurationDesk application.

2. In the **Bus Configuration Ports** table, select the **LinDoorCommController Schedule Index** function port.

   The **Properties Browser** displays the properties of the function port.

3. In the **Properties Browser**, set **Initial value** to 1.



For each LIN master, multiple schedule tables can be specified in the communication matrix. Via the **Initial Value** property of the Schedule Index function port, you can specify the schedule table that is initially active at run time.

Interim result

You specified the first schedule table of the LIN master as the initial schedule table. This enables LIN communication by default. You can now configure to access bus simulation features via experiment software and automation scripts. Continue with the next part.

Part 2

To enable access for experiment software and automation scripts

1 In the Bus Configuration Ports table, right-click the Feature column header and select Set Column Filter from the context menu.

The Column Filter Expressions for Column: Feature dialog opens, displaying the filter settings that were specified last.

Tip

If you open the filter dialog for an unfiltered column, the dialog displays the last specified column filter, regardless of the table or browser for which the column filter was specified.

2 In the dialog, specify **`ISignal Value`** as the Filter expression.

3 Select Inverse.

4 Click OK.

The dialog closes. The Bus Configuration Ports table displays all ports that are available for bus configuration features except for the ISignal Value feature. The Feature column header indicates the active column filter by a blue square.

Note

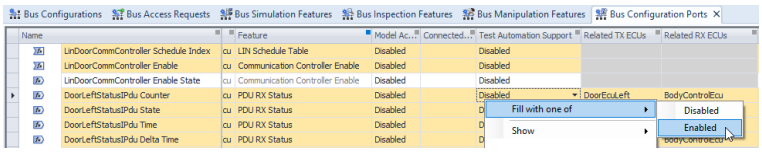
The filter is active until you clear it explicitly. This applies even when you close ConfigurationDesk or the Bus Manager (stand-alone).

5 Press the **`Ctrl+A`** keys to select all the displayed function ports.

6 Press and hold the **`Ctrl`** key and click the LinDoorCommController Enable State function port to deselect the function port.

7 Scroll to the right until the Test Automation Support column is displayed.

8 In the Test Automation Support column, right-click one of the highlighted cells and select Fill with one of - Enabled from the context menu.



Test automation support is enabled for the selected ports. This lets you access the values of the function ports via experiment software (e.g., ControlDesk) and automation scripts at run time.

Interim result

You enabled test automation support for selected function ports to access the function port values via experiment software and automation scripts at run time. You can now enable the access to a behavior model. Continue with the next part.

Part 3

To enable the behavior model access

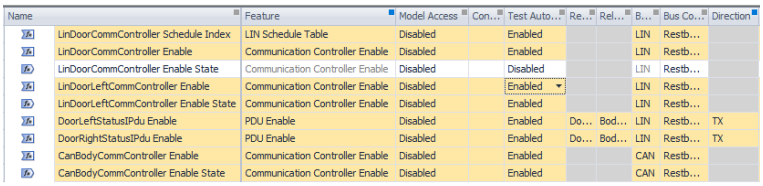
1 In the Bus Configuration Ports table, scroll to the right until the Direction column is displayed.

2 Right-click the Direction column header and select Set Column Filter from the context menu.

The Column Filter Expressions for Column: Direction dialog opens, displaying the previously specified filter settings.

3 In the dialog, specify **RX** as the Filter expression and click OK.

The dialog closes and the Bus Configuration Ports table is filtered by two column filters, i.e., by the filters you specified for the Feature column and for the Direction column. Because the Inverse checkbox was selected for both filters, all function ports of the ISignal Value feature and all RX function ports are hidden.



Name	Feature	Model Access	Con...	Test Auto...	Re...	Rel...	B...	Bus Co...	Direction
LinDoorCommController Schedule Index	LIN Schedule Table	Disabled		Enabled			LIN	Restb...	
LinDoorCommController Enable	Communication Controller Enable	Disabled		Enabled			LIN	Restb...	
LinDoorCommController Enable State	Communication Controller Enable	Disabled		Disabled			LIN	Restb...	
LinDoorLeftCommController Enable	Communication Controller Enable	Disabled		Enabled			LIN	Restb...	
LinDoorLeftCommController Enable State	Communication Controller Enable	Disabled		Enabled			LIN	Restb...	
DoorLeftStatusIPdu Enable	PDU Enable	Disabled		Enabled	Do...	Bod...	LIN	Restb...	TX
DoorRightStatusIPdu Enable	PDU Enable	Disabled		Enabled	Do...	Bod...	LIN	Restb...	TX
CanBodyCommController Enable	Communication Controller Enable	Disabled		Enabled			CAN	Restb...	
CanBodyCommController Enable State	Communication Controller Enable	Disabled		Enabled			CAN	Restb...	

Note

Even though the function ports are hidden, they are still selected. Because of this, any changes you make in the Bus Configuration Ports table will also affect the hidden ports.

4 In the Bus Configuration Ports table, right-click an empty area without selecting a command from the context menu.

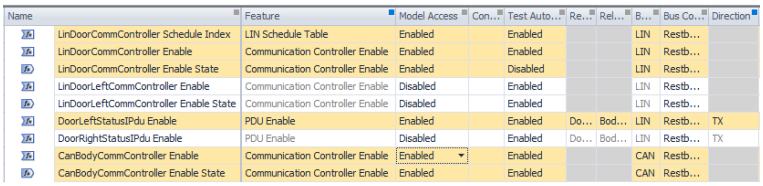
All function ports, including the hidden function ports, are deselected.

5 Select all the displayed function ports except for the following:

- LinDoorLeftCommController Enable
- LinDoorLeftCommController Enable State
- DoorRightStatusIPdu Enable

6 In the Model Access column, right-click one of the highlighted cells and select Fill with one of - Enabled from the context menu.

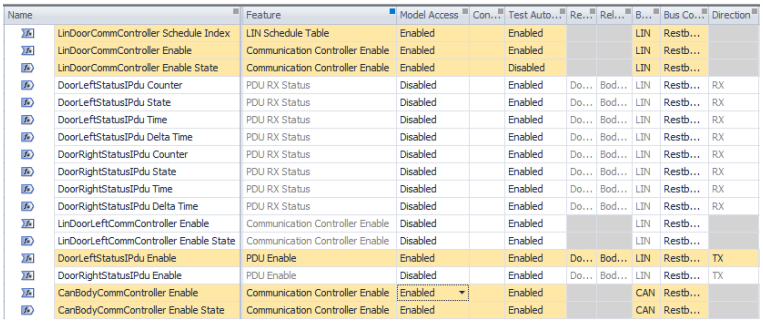
Model access is enabled for the selected function ports. This lets you map the function ports to model ports and exchange data with a behavior model at run time.



Name	Feature	Model Access	Con...	Test Auto...	Re...	Rel...	B...	Bus Co...	Direction
LinDoorCommController Schedule Index	LIN Schedule Table	Enabled		Enabled			LIN	Restb...	
LinDoorCommController Enable	Communication Controller Enable	Enabled		Enabled			LIN	Restb...	
LinDoorCommController Enable State	Communication Controller Enable	Enabled		Disabled			LIN	Restb...	
LinDoorLeftCommController Enable	Communication Controller Enable	Disabled		Enabled			LIN	Restb...	
LinDoorLeftCommController Enable State	Communication Controller Enable	Disabled		Enabled			LIN	Restb...	
DoorLeftStatusIPdu Enable	PDU Enable	Enabled		Enabled	Do...	Bod...	LIN	Restb...	TX
DoorRightStatusIPdu Enable	PDU Enable	Disabled		Enabled	Do...	Bod...	LIN	Restb...	TX
CanBodyCommController Enable	Communication Controller Enable	Enabled		Enabled			CAN	Restb...	
CanBodyCommController Enable State	Communication Controller Enable	Enabled		Enabled			CAN	Restb...	

7 Right-click the Direction column header and select Clear Column Filter from the context menu.

The specified column filter is cleared and the function ports are displayed regardless of their direction. For the RX function ports, model access is disabled.



Name	Feature	Model Access	Con...	Test Auto...	Re...	Rel...	B...	Bus Co...	Direction
LinDoorCommController Schedule Index	LIN Schedule Table	Enabled		Enabled			LIN	Restb...	
LinDoorCommController Enable	Communication Controller Enable	Enabled		Enabled			LIN	Restb...	
LinDoorCommController Enable State	Communication Controller Enable	Enabled		Disabled			LIN	Restb...	
DoorLeftStatusIPdu Counter	PDU RX Status	Disabled		Enabled	Do...	Bod...	LIN	Restb...	RX
DoorLeftStatusIPdu State	PDU RX Status	Disabled		Enabled	Do...	Bod...	LIN	Restb...	RX
DoorLeftStatusIPdu Time	PDU RX Status	Disabled		Enabled	Do...	Bod...	LIN	Restb...	RX
DoorLeftStatusIPdu Delta Time	PDU RX Status	Disabled		Enabled	Do...	Bod...	LIN	Restb...	RX
DoorRightStatusIPdu Counter	PDU RX Status	Disabled		Enabled	Do...	Bod...	LIN	Restb...	RX
DoorRightStatusIPdu State	PDU RX Status	Disabled		Enabled	Do...	Bod...	LIN	Restb...	RX
DoorRightStatusIPdu Time	PDU RX Status	Disabled		Enabled	Do...	Bod...	LIN	Restb...	RX
DoorRightStatusIPdu Delta Time	PDU RX Status	Disabled		Enabled	Do...	Bod...	LIN	Restb...	RX
LinDoorLeftCommController Enable	Communication Controller Enable	Disabled		Enabled			LIN	Restb...	
LinDoorLeftCommController Enable State	Communication Controller Enable	Disabled		Enabled			LIN	Restb...	
DoorLeftStatusIPdu Enable	PDU Enable	Enabled		Enabled	Do...	Bod...	LIN	Restb...	TX
DoorRightStatusIPdu Enable	PDU Enable	Disabled		Enabled	Do...	Bod...	LIN	Restb...	TX
CanBodyCommController Enable	Communication Controller Enable	Enabled		Enabled			CAN	Restb...	
CanBodyCommController Enable State	Communication Controller Enable	Enabled		Enabled			CAN	Restb...	

8 Right-click an empty area without selecting a command from the context menu to deselect all function ports.

Interim result

You enabled the access to a behavior model for the selected function ports of some bus configuration features. You can now enable test automation support and model access for further function ports. Continue with the next part.

Part 4

To enable test automation support and model access for further function ports

1 In the Bus Configuration Ports table, right-click the Feature column header and select Set Column Filter from the context menu.

The Column Filter Expressions for Column: Feature dialog opens, displaying the currently active filter of the Feature column.

2 In the dialog, clear Inverse and click OK.

The dialog closes. The Bus Configuration Ports table displays only function ports that are available for the ISignal Value feature.

3 Click the Name column header.

The function ports are sorted alphabetically in ascending order. The sorting order is indicated by an arrow (▲) in the column header.

4 Select the following function ports:

- All DoorLeftLockedISignal Value function ports
- All DoorRightLockedISignal Value function ports
- All LockCarISignal Value function ports
- All SpeedISignal Value function ports

5 For the selected function ports, set Model Access to Enabled.

Name	Feature	Model Access	Con...	Test Auto...
DoorLeftLockedISignal Value	ISignal Value	Enabled		Disabled
DoorLeftLockedISignal Value	ISignal Value	Enabled		Disabled
DoorLeftLockedISignal Value	ISignal Value	Enabled		Disabled
DoorLeftToWindowRightCloseISignal Value	ISignal Value	Disabled		Disabled
DoorLeftToWindowRightCloseISignal Value	ISignal Value	Disabled		Disabled
DoorLeftToWindowRightOpenISignal Value	ISignal Value	Disabled		Disabled
DoorLeftToWindowRightOpenISignal Value	ISignal Value	Disabled		Disabled
DoorRightClosedISignal Value	ISignal Value	Disabled		Disabled
DoorRightClosedISignal Value	ISignal Value	Disabled		Disabled
DoorRightClosedISignal Value	ISignal Value	Disabled		Disabled
DoorRightLockedISignal Value	ISignal Value	Enabled		Disabled
DoorRightLockedISignal Value	ISignal Value	Enabled		Disabled
DoorRightLockedISignal Value	ISignal Value	Enabled		Disabled
Kl_15ISignal Value	ISignal Value	Disabled		Disabled
Kl_15ISignal Value	ISignal Value	Disabled		Disabled
LockCarISignal Value	ISignal Value	Enabled		Disabled
LockCarISignal Value	ISignal Value	Enabled		Disabled
LockCarISignal Value	ISignal Value	Enabled		Disabled
SpeedISignal Value	ISignal Value	Enabled		Disabled
SpeedISignal Value	ISignal Value	Enabled		Disabled

6 In addition to the function ports that are already selected, select the following function ports:

- All DoorLeftClosedISignal Value function ports
- All DoorRightClosedISignal Value function ports
- All UnlockCarISignal Value function ports

7 For the selected function ports, set Test Automation Support to Enabled.

Name	ECU	Feature	Model Access	Con...	Test Automation Support
DoorLeftClosedISignal Value	BodyControlEcu	ISignal Value	Disabled		Enabled
DoorLeftClosedISignal Value	DoorEcuLeft	ISignal Value	Disabled		Enabled
DoorLeftClosedISignal Value	BodyControlEcu	ISignal Value	Disabled		Enabled
DoorLeftLockedISignal Value	BodyControlEcu	ISignal Value	Enabled		Enabled
DoorLeftLockedISignal Value	DoorEcuLeft	ISignal Value	Enabled		Enabled
DoorLeftLockedISignal Value	BodyControlEcu	ISignal Value	Enabled		Enabled
DoorLeftToWindowRightCloseISignal Value	DoorEcuLeft	ISignal Value	Disabled		Disabled
DoorLeftToWindowRightCloseISignal Value	DoorEcuRight	ISignal Value	Disabled		Disabled
DoorLeftToWindowRightOpenISignal Value	DoorEcuLeft	ISignal Value	Disabled		Disabled
DoorLeftToWindowRightOpenISignal Value	DoorEcuRight	ISignal Value	Disabled		Disabled
DoorRightClosedISignal Value	BodyControlEcu	ISignal Value	Disabled		Enabled
DoorRightClosedISignal Value	DoorEcuRight	ISignal Value	Disabled		Enabled
DoorRightClosedISignal Value	BodyControlEcu	ISignal Value	Disabled		Enabled
DoorRightLockedISignal Value	BodyControlEcu	ISignal Value	Enabled		Enabled
DoorRightLockedISignal Value	DoorEcuRight	ISignal Value	Enabled		Enabled
DoorRightLockedISignal Value	BodyControlEcu	ISignal Value	Enabled		Enabled
Kl_15ISignal Value	BodyControlEcu	ISignal Value	Disabled		Disabled
Kl_15ISignal Value	GearBoxEcu	ISignal Value	Disabled		Disabled
LockCarISignal Value	BodyControlEcu	ISignal Value	Enabled		Enabled
LockCarISignal Value	DoorEcuLeft	ISignal Value	Enabled		Enabled
LockCarISignal Value	DoorEcuRight	ISignal Value	Enabled		Enabled
SpeedISignal Value	BodyControlEcu	ISignal Value	Enabled		Enabled
SpeedISignal Value	GearBoxEcu	ISignal Value	Enabled		Enabled
UnlockCarISignal Value	BodyControlEcu	ISignal Value	Disabled		Enabled
UnlockCarISignal Value	DoorEcuLeft	ISignal Value	Disabled		Enabled
UnlockCarISignal Value	DoorEcuRight	ISignal Value	Disabled		Enabled

8 Right-click the Feature column header and select Clear Column Filter from the context menu.

The specified column filter is cleared and all function ports of the bus configuration are displayed.

9 Right-click the Name column header and select Clear Sorting from the context menu.

The function ports are sorted in the original order.

Result

You enabled LIN communication by specifying an initial schedule table. Additionally, you configured the run-time access for bus simulation features by enabling test automation support and/or model access for some of the available function ports. This lets you access the function port values via experiment software (e.g., ControlDesk) and automation scripts, and/or exchange data with a behavior model at run time.

Note

For optimum performance, it is recommended to enable model access and test automation support only for those function ports that you want to access via a behavior model or experiment software and automation scripts, respectively.

What's next

In the next step, you will check the configured bus communication for conflicts and resolve conflicts.

Step 4: How to Check the Configured Bus Communication for Conflicts and Resolve Conflicts

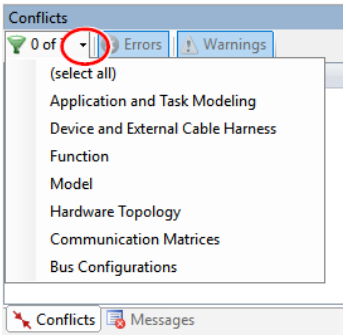
Objective

To efficiently configure bus communication in the ConfigurationDesk application, you should regularly check the configured bus communication for conflicts and resolve conflicts.

Part 1

To check the configured bus communication for conflicts

1 In the Conflicts Viewer, click the expand arrow of the context set filter.



The context sets are displayed. If you worked through lesson 1, all context sets are cleared as in the illustration above.

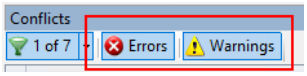
Tip

- The selection of context sets is stored locally on the host PC and applies to all ConfigurationDesk applications. If you did not work through lesson 1, the selected context sets in your ConfigurationDesk application might therefore differ.
- If you work with the Bus Manager (stand-alone), not all of the context sets are available.

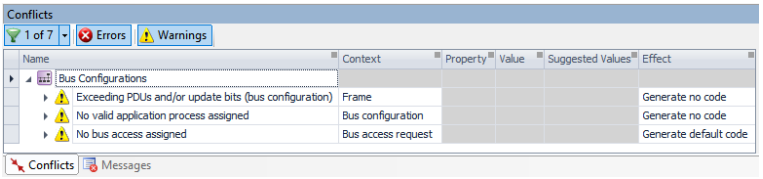
2 Select the **Bus Configurations** context set.

3 Clear all other context sets, if required.

4 Make sure that the **Errors** and **Warnings** filters are selected.



The evaluation of bus configuration conflicts is enabled and the **Conflicts Viewer** displays conflicts.



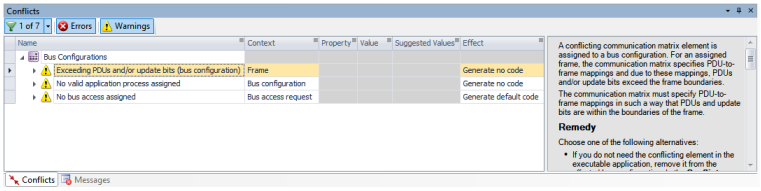
The **Exceeding PDUs and/or update bits (bus configuration)** conflict results from a frame for which the communication matrix specifies conflicting settings. The other conflicts result from the configuration state of the bus communication in the bus configuration.

Tip

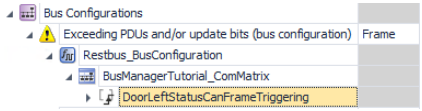
- You will resolve the conflicts of the **Bus configuration** and **Bus access request** contexts by configuring the bus communication in the following lessons.
- If you work with the Bus Manager (stand-alone), no conflict with **Bus access request** context is displayed. This is because the Bus Manager (stand-alone) displays only conflicts that are relevant for the fields of application in which the Bus Manager (stand-alone) can be used.

5 Select the **Exceeding PDUs and/or update bits (bus configuration)** conflict.

The Conflicts Viewer displays a description of the conflict causes and provides remedies for resolving the conflict.



6 Expand the Exceeding PDUs and/or update bits (bus configuration) node to the DoorLeftStatusCanFrameTriggerig node.



7 Right-click the DoorLeftStatusCanFrameTriggerig node and select Select Related Elements in Bus Configurations Table from the context menu.

In the Bus Configurations Table, the DoorLeftStatusCanIPdu is highlighted. This PDU is affected by the frame with the conflicting setting.

Tip

The Bus Communication page of the Properties Browser displays the frame properties.

Interim result

You checked the configured bus communication for conflicts. You explored the conflict cause of the Exceeding PDUs and/or update bits (bus configuration) conflict and the affected elements. You can now resolve this conflict. Continue with the next part.

Part 2

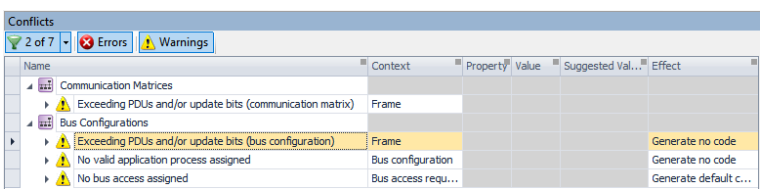
To resolve a bus configuration conflict

1 Click the expand arrow of the context set filter.

2 In the context set filter, select the Communication Matrices context set.

3 Close the context set filter.

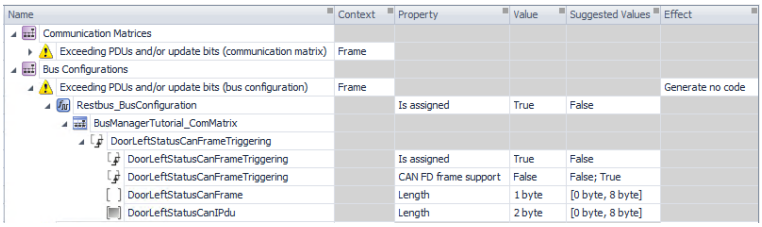
The evaluation of communication matrix conflicts is enabled. In addition to the bus configuration conflicts, the Conflicts Viewer displays a communication matrix conflict.



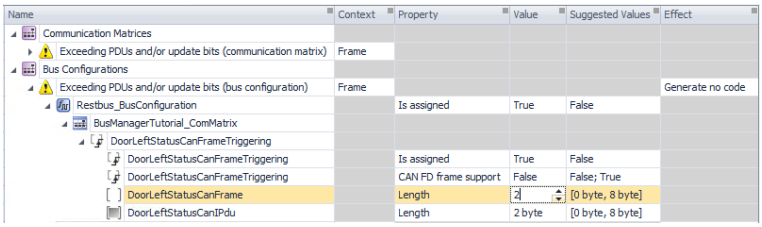
The Exceeding PDUs and/or update bits conflicts of the communication matrix and the bus configuration result from the same conflicting element. However, only the bus configuration conflict affects the build process and the generation of bus simulation containers: If you do not resolve this conflict, no code is generated for the entire bus configuration when you start the build process or generate bus simulation containers.

4 Completely expand the Exceeding PDUs and/or update bits (bus configuration) node.

The Conflicts Viewer displays different options that let you resolve the conflict.



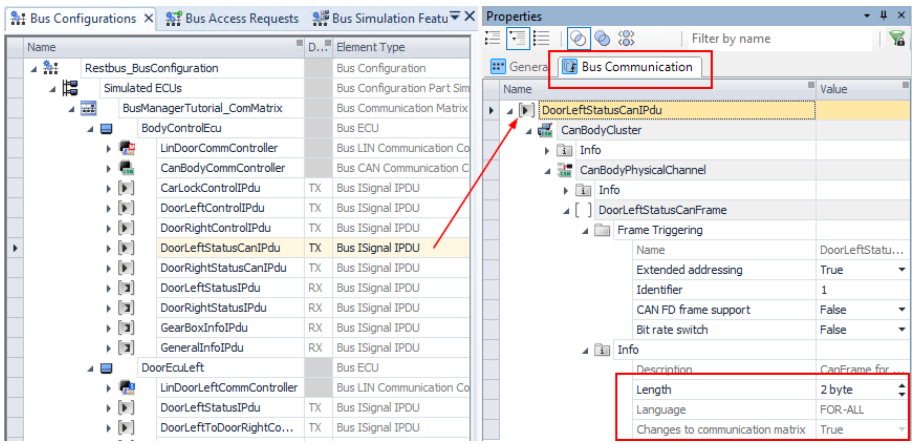
5 For the DoorLeftStatusCanFrame, specify 2 as the value of the Length property.



By this, you change the frame length from 1 byte to 2 bytes, which matches the 2 bytes of the related PDU. This change applies to the frame in the bus configuration and the communication matrix. Therefore, both Exceeding PDUs and/or update bits conflicts are resolved.

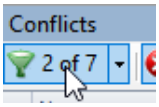
6 In the Bus Configurations Table, select the DoorLeftStatusCanIPdu of the BodyControlEcu.

7 In the Properties Browser, select the Bus Communication page.



The **Properties Browser** displays the frame length, which is now 2 bytes. The **Changes to communication matrix** property indicates that the communication matrix was changed. However, this change applies only to the communication matrix in the ConfigurationDesk application. The original communication matrix file remains unchanged.

8 In the **Conflicts Viewer**, click the context set filter.



The evaluation of all ConfigurationDesk conflicts is disabled and the **Conflicts Viewer** displays no conflicts.

Result

You checked the configured bus communication for conflicts and resolved a conflict of an assigned conflicting communication matrix element. Because you corrected the conflicting setting in the communication matrix, you also resolved the related communication matrix conflict. Finally, you disabled the evaluation of all conflicts.

Note

The evaluation of conflicts can significantly influence the performance of the ConfigurationDesk application. Depending on the complexity of the bus communication, this especially applies to Bus Manager-related conflicts. It is therefore recommended to enable the conflict evaluation only when and as long as required.

What's next

With this step, you finished the lesson. For a summary of the lesson and for more information, refer to Result of Lesson 2 on page 93.

Result of Lesson 2

Result

In this lesson, you learned the basic principles of configuring bus communication for simulation purposes:

- To simulate bus communication, you must assign communication matrix elements to the **Simulated ECUs** part of bus configurations.
- To configure the bus communication, you can add bus simulation features to assigned elements.
- Depending on the specific bus simulation feature, function ports are available. To access data at run time, you must configure function ports:
  - To exchange data with a behavior model, you must enable model access.
  - To access data via experiment software or automation scripts, you must enable test automation support.

- The **LIN Schedule Table** feature is automatically added to assigned LIN masters. This feature is mandatory to enable LIN communication. Additionally, the **ISignal Value** feature is automatically added to assigned ISignals.
- You should regularly check the configured bus communication for conflicts and resolve conflicts. The **Conflicts Viewer** provides information on the conflict causes and possible remedies, and lets you resolve many conflicts.
- You can resolve some conflicts by correcting conflicting communication matrix elements. When you do this, you change the communication matrix in the ConfigurationDesk application. The original communication matrix file remains unchanged.

Prepared result application

The **Lesson_2_Result** application provides the result of this lesson. You can activate the application to compare it with your own results, for example. For more information, refer to Tutorial project and applications on page 23.

Further information

- For more information on assigning communication matrix elements to bus configurations, refer to:
  - Bus Manager in ConfigurationDesk: Assigning Communication Matrix Elements to Bus Configurations (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Assigning Communication Matrix Elements to Bus Configurations (Bus Manager (Stand-Alone) Implementation Guide 🕮)
- For more information on working with bus configuration features, refer to:
  - Bus Manager in ConfigurationDesk: Basics on Working with Bus Configuration Features (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Basics on Working with Bus Configuration Features (Bus Manager (Stand-Alone) Implementation Guide 🕮)
- For more information on working with LIN schedule tables, refer to:
  - Bus Manager in ConfigurationDesk: Working with LIN Schedule Tables (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Working with LIN Schedule Tables (Bus Manager (Stand-Alone) Implementation Guide 🕮)
- For more information on Bus Manager-related conflicts, refer to:
  - Bus Manager in ConfigurationDesk: Handling Bus Manager-Related Conflicts (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Handling Bus Manager-Related Conflicts (Bus Manager (Stand-Alone) Implementation Guide 🕮)
- For more information on using tables in ConfigurationDesk applications, refer to Using Tables to Access and Configure Elements (ConfigurationDesk Real-Time Implementation Guide 🕮).
- For more information on using keyboard shortcuts in ConfigurationDesk applications, refer to Using Keyboard Shortcuts (ConfigurationDesk Real-Time Implementation Guide 🕮).

Where to go from here

Depending on your use scenario, you can now do the following:

- Add an existing behavior model to the ConfigrationDesk application and map the model to the bus configuration. Refer to Lesson 3: Working with an Existing Behavior Model on page 97.
- Generate a new Simulink model. Refer to Lesson 9: Generating a Simulink Model and Synchronizing the Model Interfaces in ConfigurationDesk and Simulink on page 155.
- Configure bus communication for inspection purposes. Refer to Lesson 6: Configuring Bus Communication for Inspection Purposes on page 125.
- Configure bus communication for manipulation purposes. Refer to Lesson 7: Configuring Bus Communication for Manipulation Purposes on page 133.

Lesson 3: Working with an Existing Behavior Model

Where to go from here

Information in this section

Overview of Lesson 3 97
Step 1: How to Add a Behavior Model to the ConfigurationDesk Application 98
Step 2: How to Map Model Ports to Function Ports 100
Step 3: How to Explore Mapped Ports in a Graphical View 105
Result of Lesson 3 107

Overview of Lesson 3

Working with existing behavior models

When you configure bus communication by using the Bus Manager, you can exchange data with a behavior model, for example, to provide dynamically calculated signal values to an ECU under test.

If you already have a behavior model, you can add the behavior model to the ConfigurationDesk application. To exchange data between a bus configuration and the behavior model, you must map model ports of the behavior model to function ports of the bus configuration.

What you will learn

In this lesson, you will learn how to add a behavior model to the ConfigurationDesk application and to map model ports of the behavior model to function ports of a bus configuration.

Before you begin

Before you begin this lesson, the following preconditions must be met:

- The following licenses are accessible:
  - CAN module license (CFD_I_CAN)
  - LIN module license (CFD_I_LIN)

  Refer to How to Identify Accessible Licenses on page 62.
- The result of lesson 2 is available and the related application is active.

  If you did not work through lesson 2, you can activate the Lesson_2_Result application. Refer to How to Activate an Application in the BusManagerTutorial Project on page 61.

Steps

This lesson contains the following steps:

- Step 1: How to Add a Behavior Model to the ConfigurationDesk Application on page 98
- Step 2: How to Map Model Ports to Function Ports on page 100
- Step 3: How to Explore Mapped Ports in a Graphical View on page 105

Result and further information

The lesson concludes with a summary of the lesson content and with further information. Refer to Result of Lesson 3 on page 107.

Duration

Completing this lesson will take you about 25 minutes.

Step 1: How to Add a Behavior Model to the ConfigurationDesk Application

Objective

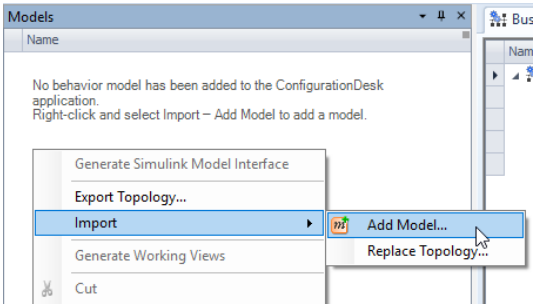
To exchange data with a behavior model, you must add the behavior model to the ConfigurationDesk application.

Method

To add a behavior model to the ConfigurationDesk application

1. Switch to the Buses view set, if necessary.
2. Select the Model Browser.

3 Right-click an empty area in the **Model Browser** and select **Import - Add Model** from the context menu.



The **Add Model** dialog opens.

4 In the dialog, click **Add Model - Add model from file**.

A standard **Open** dialog opens, displaying the folder you opened last in the ConfigurationDesk project.

Tip

If you worked through lesson 1 and 2 before you started with this lesson, the **Communication Matrices** folder of the **BusManagerTutorial** project is displayed.

5 In the dialog, navigate to the **Models** folder of the **BusManagerTutorial** project. The path to the folder depends on your Bus Manager variant:

- Bus Manager in ConfigurationDesk:

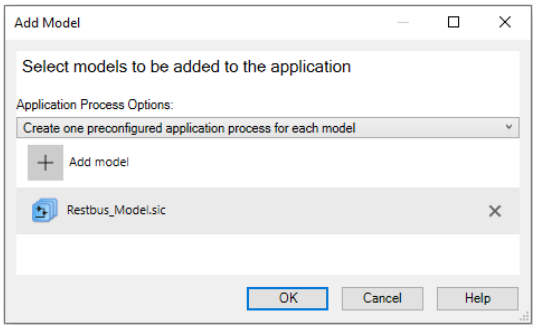
  `%USERPROFILE%\Documents\dSPACE\ConfigurationDesk\<VersionNumber>\Tutorials\BusManagerTutorial\Models`

- Bus Manager (stand-alone):

  `%USERPROFILE%\Documents\dSPACE\BusManager\<VersionNumber>\Tutorials\BusManagerTutorial\Models`

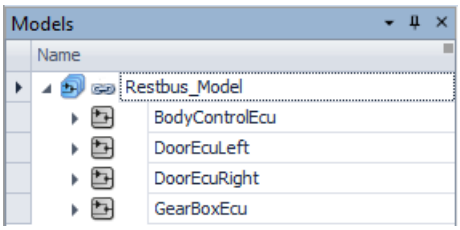
6 Select `Restbus_Model.sic` and click **Open**.

The **Open** dialog closes and the **Restbus_Model.sic** is displayed in the **Add Model** dialog.



7 In the Add Model dialog, click OK.

The dialog closes and the behavior model is added to the Model Browser.



Result

You added the Restbus_Model behavior model to the ConfigurationDesk application. This model is an SIC file, i.e., a Simulink implementation container. Because you kept the default settings in the Add Model dialog, a preconfigured application process is created for the Restbus_Model.

Tip

- Application processes are structuring elements that contain tasks. Tasks are pieces of code whose execution is controlled by the simulation platform, e.g., by the real-time operating system (RTOS). Tasks are required during the execution of executable applications.
- To include a bus configuration in a real-time application or a bus simulation container, it must be assigned to exactly one application process. When you map model ports of the Restbus_Model to function ports of the Restbus_BusConfiguration, the bus configuration is automatically assigned to the created application process.

What's next

In the next step, you will map model ports of the Restbus_Model to function ports of the Restbus_BusConfiguration.

Step 2: How to Map Model Ports to Function Ports

Objective

To exchange data with a behavior model, you must map model ports of a behavior model to function ports of a bus configuration. You can map the ports via the Model Browser and the Bus Configuration Ports table and by using various filter options.

Part 1

To map model ports to function ports of the GearBoxEcu

1 In the Model Browser, right-click the GearBoxEcu node and select Expand from the context menu.

The model port blocks and model ports of the GearBoxEcu subsystem are displayed.

2 Select the Bus Configuration Ports table.

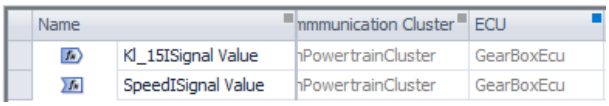
3 In the table, right-click the ECU column header and select Set Column Filter from the context menu.

The Column Filter Expressions for Column: ECU dialog opens.

4 In the dialog, specify **gear** as the Filter expression.

5 Make sure that only the Regular expression checkbox is selected and click OK.

The Bus Configuration Ports table displays only function ports that are available for the GearBoxEcu. The column header indicates the active column filter by a blue square.

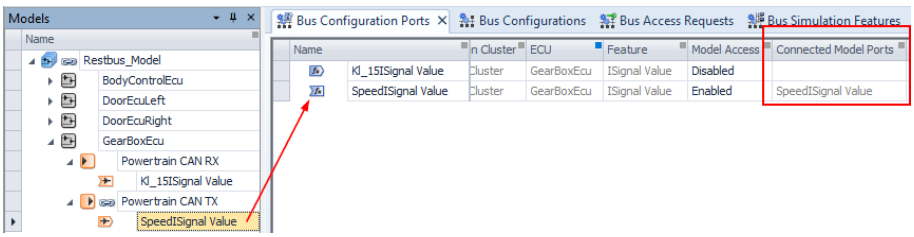


Note

The filter is active until you clear it explicitly. This applies even when you close ConfigurationDesk or the Bus Manager (stand-alone).

6 In the Model Browser, select the SpeedISignal Value model port and drag it to the SpeedISignal Value function port in the Bus Configuration Ports table.

The SpeedISignal Value ports are mapped. The Bus Configuration Ports table displays the mapped model port in the Connected Model Ports column.



Tip

In the Model Browser, the Powertrain CAN TX model port block is marked with a chain symbol. However, this does not indicate that its model port is mapped to a function port. This symbol only indicates that the model port block is used in the signal chain.

7 Map the Kl_15ISignal Value model port to the Kl_15ISignal Value function port.

Tip

This automatically enables the model access of the Kl_15ISignal Value function port.

Interim result

You mapped ports of the GearBoxEcu. You can now map ports of the DoorEcuRight. Continue with the next part.

Part 2

To map model ports to function ports of the DoorEcuRight

1 Right-click an empty area in the Model Browser and select Filter for Unconnected Ports from the context menu.

The mapped model ports and related higher-level elements are hidden. Because all model ports of the GearBoxEcu subsystem are mapped, the GearBoxEcu node is hidden completely.

Note

Selected port filters are active until you clear them explicitly. This applies even when you close ConfigurationDesk or the Bus Manager (stand-alone). Active port filters are indicated only in the context menu.

2 Right-click the DoorEcuRight node and select Expand from the context menu.

3 In the Bus Configuration Ports table, right-click the ECU column and select Set Column Filter from the context menu.

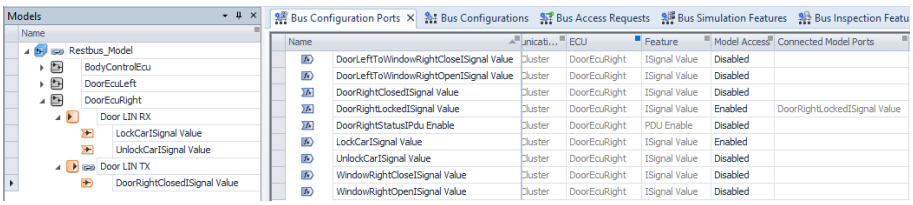
4 In the Column Filter Expressions for Column: ECU dialog, specify **`right`** as the Filter expression and click OK.

All function ports that are available for the DoorEcuRight are displayed.

5 In the Bus Configuration Ports table, click the Name column to sort the displayed function ports by their names.

6 Map the DoorRightLockedISignal Value model port to the DoorRightLockedISignal Value function port.

Because Filter for Unconnected Ports is active, the model port is hidden in the Model Browser. The Bus Configuration Ports table displays the mapped model port in the Connected Model Ports column.



Tip

If you mapped ports incorrectly, you can undo the mapping by pressing the `Ctrl+Z` keys.

7 Map the remaining model ports of the DoorEcuRight to the function ports with identical names.

After you finish, the DoorEcuRight node is hidden in the Model Browser and the Bus Configuration Ports table displays the four mapped ports.

Interim result

You mapped ports of the DoorEcuRight. You can now map ports of the DoorEcuLeft. Continue with the next part.

Part 3

To map model ports to function ports of the DoorEcuLeft

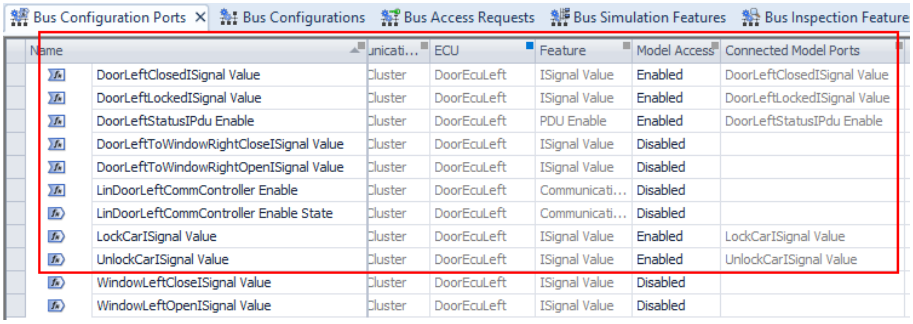
1 In the Model Browser, expand the DoorEcuLeft node to its lowest level.

2 In the Bus Configuration Ports table, set the column filter of the ECU column to `left`.

All function ports that are available for the DoorEcuLeft are displayed.

3 Map the model ports of the DoorEcuLeft to the function ports with identical names.

4 Check the mapped ports in the Bus Configuration Ports table.



Name	Communicati...	ECU	Feature	Model Access	Connected Model Ports
DoorLeftClosedISignal Value	Cluster	DoorEcuLeft	ISignal Value	Enabled	DoorLeftClosedISignal Value
DoorLeftLockedISignal Value	Cluster	DoorEcuLeft	ISignal Value	Enabled	DoorLeftLockedISignal Value
DoorLeftStatusIPdu Enable	Cluster	DoorEcuLeft	PDU Enable	Enabled	DoorLeftStatusIPdu Enable
DoorLeftToWindowRightCloseISignal Value	Cluster	DoorEcuLeft	ISignal Value	Disabled	
DoorLeftToWindowRightOpenISignal Value	Cluster	DoorEcuLeft	ISignal Value	Disabled	
LinDoorLeftCommController Enable	Cluster	DoorEcuLeft	Communicati...	Disabled	
LinDoorLeftCommController Enable State	Cluster	DoorEcuLeft	Communicati...	Disabled	
LockCarISignal Value	Cluster	DoorEcuLeft	ISignal Value	Enabled	LockCarISignal Value
UnlockCarISignal Value	Cluster	DoorEcuLeft	ISignal Value	Enabled	UnlockCarISignal Value
WindowLeftCloseISignal Value	Cluster	DoorEcuLeft	ISignal Value	Disabled	
WindowLeftOpenISignal Value	Cluster	DoorEcuLeft	ISignal Value	Disabled	

Interim result

You mapped ports of the DoorEcuLeft. You can now map ports of the BodyControlEcu. Continue with the next part.

Part 4

To map model ports to function ports of the BodyControlEcu

1 In the Model Browser, expand the BodyControlEcu node to its lowest level.

2 In the Bus Configuration Ports table, specify the following column filters:

Column	Filter Expression
ECU	body
Model Access	enabled

3 In the Bus Configuration Ports table, right-click an empty area and select Filter for Inports from the context menu.

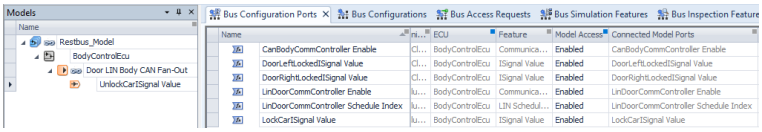
4 In the Model Browser, right-click an empty area and select Filter for Outports.

5 Map the the displayed model ports to function ports with the same name.

Tip

For the UnlockCarISignal Value model port, no suitable function port is available. This is because the related function port does not match the active filters of the Bus Configuration Ports table. Therefore, you cannot map the port yet.

After you finish, all displayed function ports are mapped to model ports.



Interim result

You mapped all function ports of the BodyControlEcu that match the specified filters. You can now change the filter settings to map the remaining unmapped model ports of the Restbus_Model. Continue with the next part.

Part 5

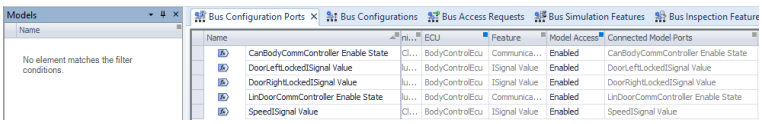
To map the remaining unmapped model ports of the Restbus_Model

1 Change the selected port filters as follows:

Browser/Table	Clear Filter	Select Filter
Model Browser	Filter for Outports	Filter for Inports
Bus Configuration Ports table	Filter for Inports	Filter for Outports

2 Map the displayed model ports to function ports with the same name.

After you finish, the Model Browser does not display any elements and the Bus Configuration Ports table displays the mapped ports.



3 Clear the following port filters:

- Model Browser: Clear Filter for Inports.
- Bus Configuration Ports table: Clear Filter for Outports.

The Model Browser displays the remaining unmapped model port, i.e., the UnlockCarISignal Value model port.

4 In the Bus Configuration Ports table, right-click the Model Access column and select Set Column Filter from the context menu.

5 In the Column Filter Expressions for Column: Model Access dialog, select Inverse and click OK.

All function ports of the BodyControlEcu with disabled model access are displayed.

6 Map the UnlockCarISignal Value model port to the UnlockCarISignal Value function port.

This enables the model access of the UnlockCarISignal Value function port. Therefore, the port does not match the active Model Access column filter and is hidden.

7 In the Bus Configuration Ports table, right-click any column header and select Clear All Column Filters from the context menu.

8 In the Model Browser, clear the Filter for Unconnected Ports.

Result

You mapped all the model ports that are available for the Restbus_Model to suitable function ports.

What's next

In the next step, you will explore the mapping of the ports in a graphical view.

Step 3: How to Explore Mapped Ports in a Graphical View

Objective

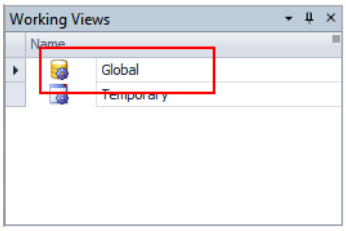
To explore mapped ports in a graphical view, you can select the Signal Chain Browser. The Signal Chain Browser displays working views that provide a block-based view on the elements that are used in the signal chain.

Method

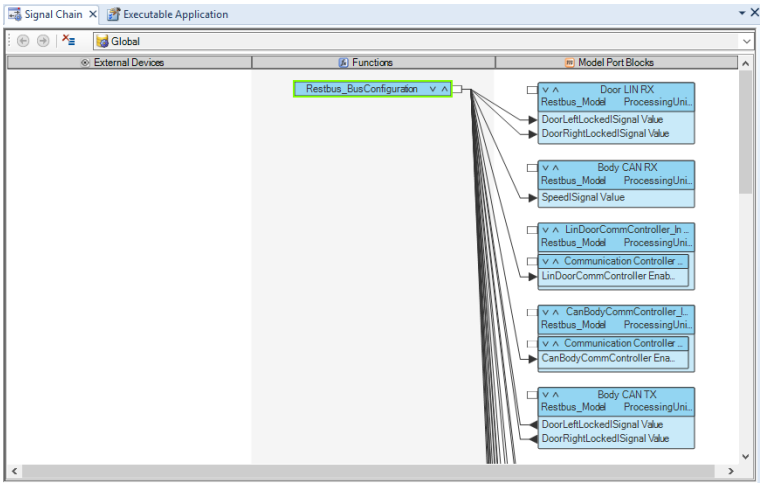
To explore mapped ports in a graphical view

1 Switch to the Signal Chain view set.

2 If the Signal Chain Browser does not display the Global working view, open the Global working view: In the Working View Manager, double-click the Global working view.

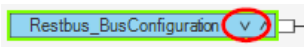


The Signal Chain Browser displays the Global working view. The Global working view displays the model port blocks of the behavior model and the Restbus_BusConfiguration function block. The Restbus_BusConfiguration function block might be collapsed, as shown in the following example.



3 If the Restbus_BusConfiguration function block is collapsed, do the following:

1. Click the expand arrow of the Restbus_BusConfiguration function block.



The Restbus_BusConfiguration function block is expanded but the mapping lines are still drawn to the parent port of the function block.

2. Click the expand arrow of the Restbus_BusConfiguration function block once more.

The mapping lines are drawn to the function ports of the function block.

4 On the Home ribbon, click  .

The order of the displayed blocks is rearranged so that as many mapping lines as possible are horizontal.

5 Scroll in the Signal Chain Browser to explore the mapping of function ports and model ports.

Tip

If you place the cursor on a signal chain element, e.g., on a function port, a tooltip displays the name of the element.

What's next

With this step, you finished the lesson. For a summary of the lesson and for more information, refer to Result of Lesson 3 on page 107.

Result of Lesson 3

Result

In this lesson, you learned the basic principles of working with an existing behavior model:

- When you add a behavior model to the ConfigurationDesk application, a preconfigured application process can be created automatically. When you map a bus configuration to this model, the bus configuration is automatically assigned to the application process.
- You must manually map model ports of the behavior model to function ports of a bus configuration. To do this, you can drag model ports from the **Model Browser** to function ports in the **Bus Configuration Ports** table.
- The **Model Browser** and the **Bus Configuration Ports** table provide various filter options that can support you by mapping the ports.
- Working views provide a graphical representation of the port mapping.

Prepared result application

The **Lesson_3_Result** application provides the result of this lesson. You can activate the application to compare it with your own results, for example. For more information, refer to Tutorial project and applications on page 23.

Further information

- For more information on working with behavior models in ConfigurationDesk applications, refer to Specifying the Model Interface (ConfigurationDesk Real-Time Implementation Guide 🕮).
- For more information on application processes and other elements of executable applications, refer to Introduction to Modeling Executable Applications and Tasks (ConfigurationDesk Real-Time Implementation Guide 🕮).
- For more information on working views, refer to Handling the Signal Chain in Working Views (ConfigurationDesk Real-Time Implementation Guide 🕮).

Where to go from here

Depending on your use scenario, you can now do the following:

- Generate bus simulation containers. Refer to Lesson 4: Generating Bus Simulation Containers on page 109.
- Build a real-time application. Refer to Lesson 5: Building a Real-Time Application on page 115.
- Configure bus communication for inspection purposes. Refer to Lesson 6: Configuring Bus Communication for Inspection Purposes on page 125.
- Configure bus communication for manipulation purposes. Refer to Lesson 7: Configuring Bus Communication for Manipulation Purposes on page 133.

Lesson 4: Generating Bus Simulation Containers

Where to go from here

Information in this section

Overview of Lesson 4 109
Step 1: How to Generate Bus Simulation Containers 110
Result of Lesson 4 114

Overview of Lesson 4

Generating bus simulation containers

The Bus Manager lets you generate bus simulation containers. Bus simulation containers let you use the bus communication you configured by using the Bus Manager on various simulation platforms, e.g., on VEOS or SCALEXIO systems. Additionally, you can use bus simulation containers to archive the configured bus communication or to exchange it between different users, for example.

When you generate bus simulation containers, one BSC file is generated for each application process that fulfills the following conditions:

- The application process contains no or only one behavior model. If the application process contains a behavior model, it must be a Simulink implementation container (SIC file).
- The application process contains at least one bus configuration that does not have any conflict with **Generate no code** effect.
- For the application process, there are no conflicts with **Abort BSC file generation** effect.

Tip

Application processes are structuring elements that contain tasks. Tasks are pieces of code whose execution is controlled by the simulation platform, e.g., by the real-time operating system (RTOS). Tasks are required during the execution of executable applications.

What you will learn

In this lesson, you will learn how to generate bus simulation containers.

Before you begin

Before you begin this lesson, the following preconditions must be met:

- The following licenses are accessible:
  - CAN module license (CFD_I_CAN)
  - LIN module license (CFD_I_LIN)

  Refer to How to Identify Accessible Licenses on page 62.
- The result of lesson 3 is available and the related application is active.

  If you did not work through lesson 3, you can activate the **Lesson_3_Result** application. Refer to How to Activate an Application in the BusManagerTutorial Project on page 61.

Steps

This lesson contains the following steps:

- Step 1: How to Generate Bus Simulation Containers on page 110

Result and further information

The lesson concludes with a summary of the lesson content and with further information. Refer to Result of Lesson 4 on page 114.

Duration

Completing this lesson will take you about 10 minutes.

Step 1: How to Generate Bus Simulation Containers

Objective

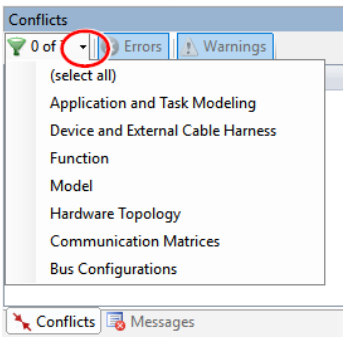
To generate bus simulation containers, first check the **Conflicts Viewer** for conflicts that affect the generation of bus simulation containers and/or the code generation for bus configurations. If no such conflicts exist, you can generate bus simulation containers in one step for all application processes that fulfill the preconditions for generating bus simulation containers.

Part 1

To check the Conflicts Viewer for conflicts

1 Switch to the Buses view set, if necessary.

2 In the Conflicts Viewer, click the expand arrow of the context set filter.



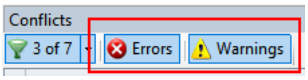
The context sets are displayed. In the illustration above, no context sets are selected.

Tip

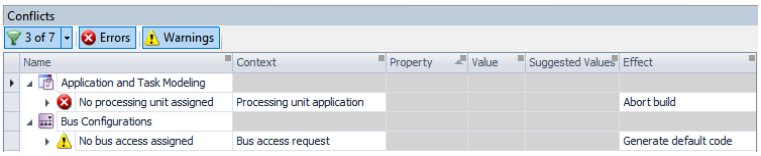
- The selection of context sets is stored locally on the host PC and applies to all ConfigurationDesk applications. Therefore, the selected context sets in your ConfigurationDesk application might differ.
- If you work with the Bus Manager (stand-alone), not all of the context sets that are displayed above are available.

3 Make sure that at least the Application and Task Modeling, Model, and Bus Configurations context sets are selected.

4 Make sure that the Errors and Warnings filters are selected.



The Conflicts Viewer displays all conflicts of the selected context sets and filters.

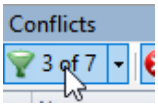


The displayed conflicts affect neither the generation of bus simulation containers nor the code that will be generated for the bus configuration and included in bus simulation containers. Therefore, you do not have to resolve them before generating bus simulation containers.

Tip

If you work with the Bus Manager (stand-alone), the Conflicts Viewer displays no conflicts. This is because the Bus Manager (stand-alone) displays only conflicts that are relevant for the fields of application in which the Bus Manager (stand-alone) can be used.

5 Click the context set filter.



The evaluation of all ConfigurationDesk conflicts is disabled and the Conflicts Viewer displays no conflicts.

Note

For optimum performance, it is recommended to enable the evaluation of conflicts only when and as required.

Interim result

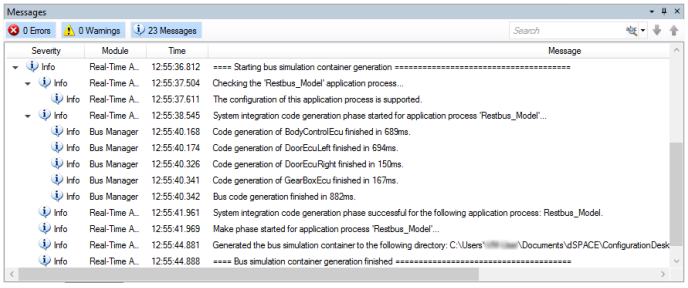
You checked the Conflicts Viewer for conflicts that affect bus simulation containers and/or their included bus configurations. Because no such conflicts exist, you can now generate bus simulation containers. Continue with the next part.

Part 2

To generate bus simulation containers

1 On the Home ribbon, click  .

The generation of bus simulation containers starts. The Message Viewer displays the progress of the generation process in chronological order.

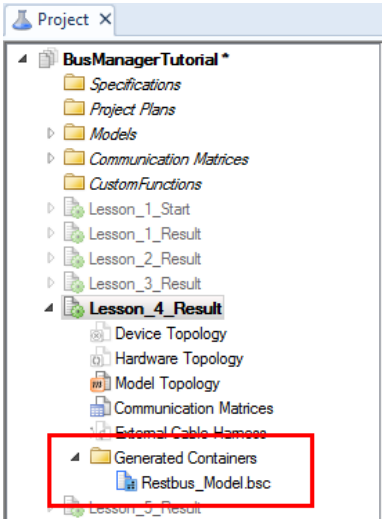


Tip

During the generation process, each available application process is checked to determine whether its configuration is supported for generating bus simulation containers. If an application process is not supported, e.g., because it contains a behavior model which is no Simulink implementation container, the Message Viewer informs you that no BSC file will be generated for this application process.

2 After the generation of bus simulation containers has finished, select the Project view set.

The Project Manager displays the BusManagerTutorial project and the active application. For this application, a Generated Containers folder is available. The folder contains the Restbus_Model.bsc file.



Tip

The BSC file is named after the application process that was automatically created when you added the Restbus_Model in lesson 3.

Result

You generated a bus simulation container for the application process to which the Restbus_BusConfigration is assigned.

What's next

With this step, you finished the lesson. For a summary of the lesson and for more information, refer to Result of Lesson 4 on page 114.

Result of Lesson 4

Result

In this lesson, you learned the basic principles of generating bus simulation containers:

- Conflicts can affect the generation of bus simulation containers and/or the code that is generated for bus configurations. The **Conflicts Viewer** can support you in identifying and resolving such conflicts.
- For each application process that fulfills the preconditions for generating bus simulation containers, one BSC file is generated.
- The generated BSC files are stored to the **Generated Containers** folder that is added to the active ConfigurationDesk application when you generate bus simulation containers for the first time.
- The generated BSC files are named after the related application processes.

Prepared result application

The **Lesson_4_Result** application provides the result of this lesson. You can activate the application to compare it with your own results, for example. For more information, refer to Tutorial project and applications on page 23.

Further information

- For more information on generating bus simulation containers, refer to:
  - Bus Manager in ConfigurationDesk: Generating Bus Simulation Containers (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Generating Bus Simulation Containers (Bus Manager (Stand-Alone) Implementation Guide 🕮)
- For more information on the **Conflicts Viewer** and Bus Manager-related conflicts, refer to:
  - Bus Manager in ConfigurationDesk: Handling Bus Manager-Related Conflicts (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Handling Bus Manager-Related Conflicts (Bus Manager (Stand-Alone) Implementation Guide 🕮)

Where to go from here

You can now continue with a lesson that is relevant for your use scenario. Refer to Workflows for Using the Bus Manager on page 27.

Lesson 5: Building a Real-Time Application

Where to go from here

Information in this section

Overview of Lesson 5 115
Step 1: How to Import a Hardware Topology 116
Step 2: How to Specify the Hardware Access for Bus Configurations 118
Step 3: How to Build a Real-Time Application 121
Result of Lesson 5 123

Overview of Lesson 5

Including bus communication in real-time applications

When you work with the Bus Manager in ConfigurationDesk, you can directly implement the configured bus communication in a real-time application. To do this, you must specify the bus access.

Each communication cluster that is assigned to a bus configuration part (e.g., **Simulated ECUs**) requires access to a bus. This request is represented by a bus access request. Therefore, a bus configuration can provide multiple bus access requests. The bus access requests contain the requirements for the bus channels that are used for the bus communication at run time.

To implement the bus communication of a bus configuration in a real-time application, you must also specify the hardware access for all of its bus access requests. For this purpose, you need the following:

- A hardware topology that provides suitable bus channels
- Suitable bus function blocks, i.e., **CAN** and/or **LIN** function blocks

What you will learn

In this lesson, you will learn how to import a hardware topology, specify the hardware access for bus configurations, and build a real-time application.

Before you begin

Before you begin this lesson, the following preconditions must be met:

- You work with the Bus Manager in ConfigurationDesk.
- The following licenses are accessible:
  - CAN module license (CFD_I_CAN)
  - LIN module license (CFD_I_LIN)

  Refer to How to Identify Accessible Licenses on page 62.
- The result of lesson 3 is available and the related application is active.

  If you did not work through lesson 3, you can activate the Lesson_3_Result application. Refer to How to Activate an Application in the BusManagerTutorial Project on page 61.

  Alternatively, you can use other applications as the starting point. For an overview of the suitable applications, refer to Overview of Lessons on page 25.

Steps

This lesson contains the following steps:

- Step 1: How to Import a Hardware Topology on page 116
- Step 2: How to Specify the Hardware Access for Bus Configurations on page 118
- Step 3: How to Build a Real-Time Application on page 121

Result and further information

The lesson concludes with a summary of the lesson content and with further information. Refer to Result of Lesson 5 on page 123.

Duration

Completing this lesson will take you about 15 minutes.

Step 1: How to Import a Hardware Topology

Objective

To connect a ConfigurationDesk application to real-time hardware, you need a hardware topology. To load a real-time application to real-time hardware, the hardware topology must match the real-time hardware.

Note

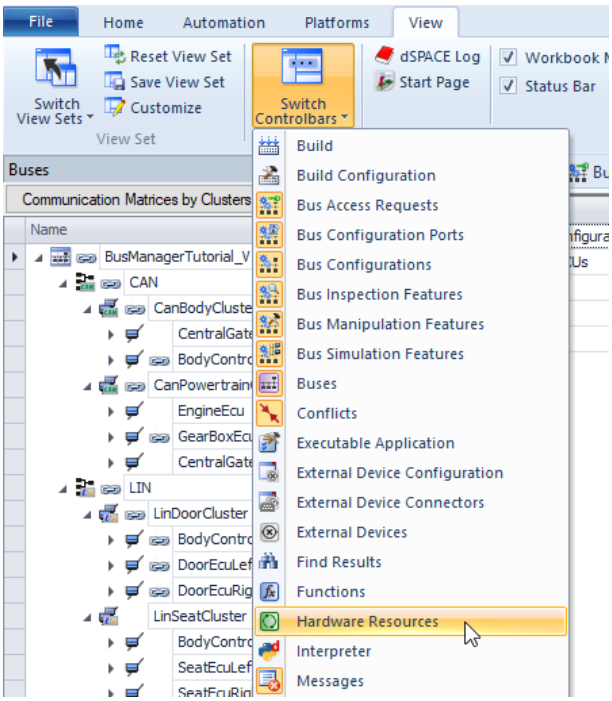
In this step, you will work with a predefined hardware topology that might not match your real-time hardware. Therefore, it is unlikely that you can use the real-time application of this lesson on your real-time hardware.

Method

To import a hardware topology

1 Switch to the Buses view set, if necessary.

2 On the View ribbon, click Switch Controlbars - Hardware Resources.

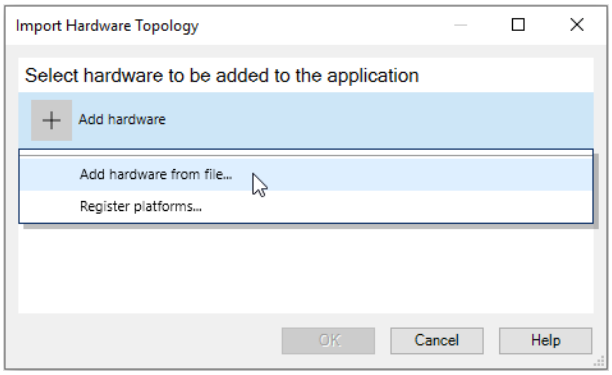


The Hardware Resource Browser is displayed.

3 Right-click an empty area in the Hardware Resource Browser and select Import Hardware - Replace from the context menu.

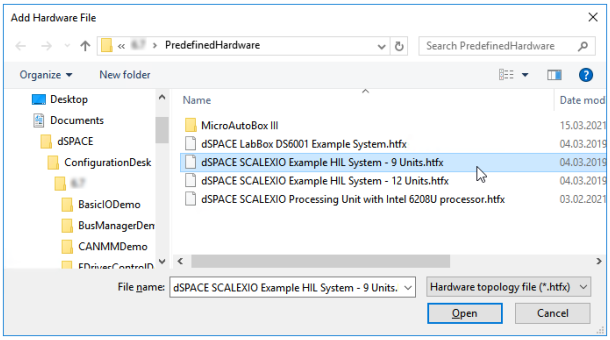
The Import Hardware Topology dialog opens.

4 In the dialog, click Add hardware - Add hardware from file.



The Add Hardware File dialog opens, displaying the PredefinedHardware folder of your Documents folder.

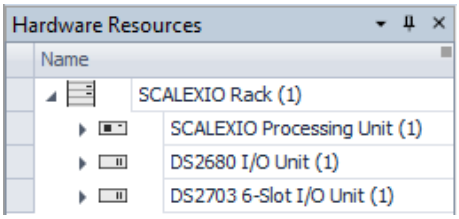
5 In the dialog, select dSPACE SCALEXIO Example HIL System - 9 Units.htfx and click Open.



The dialog closes. The Import Hardware Topology dialog displays the selected HTFX file.

6 In the dialog, click OK.

The HTFX file is imported to the ConfigurationDesk application. This might take a while. When finished, the Hardware Resource Browser displays the hardware topology.



Result

You imported a hardware topology to the ConfigurationDesk application.

What's next

In the next step, you will specify the hardware access for the Restbus_BusConfiguration.

Step 2: How to Specify the Hardware Access for Bus Configurations

Objective

To specify the hardware access for bus configurations, you must assign the bus access requests to bus accesses, i.e., to bus function blocks (CAN, LIN). To access bus channels of real-time hardware, you must assign hardware resources to the bus function blocks.

You can manually and/or automatically assign bus access requests to bus accesses. In this step, you will automatically assign all the bus access requests of the Restbus_BusConfiguration to bus accesses.

Part 1

To automatically assign bus access requests to bus accesses

1 Select the Bus Access Requests table.

The Bus Access Requests table displays the bus access requests of all bus configurations.

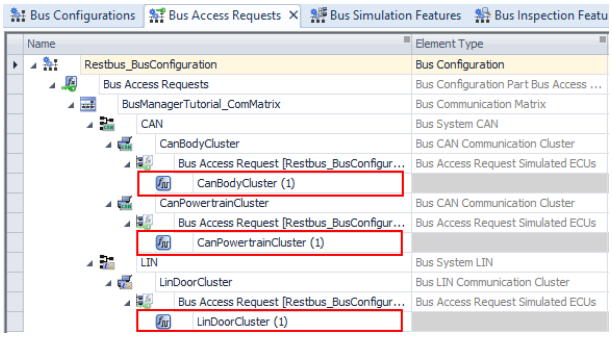
2 In the Bus Access Requests table, select the Restbus_BusConfiguration.

3 On the Home ribbon, click  .

All the bus access requests of the Restbus_BusConfiguration are automatically assigned to bus accesses, i.e., to suitable bus function blocks (CAN, LIN).

4 Completely expand the CAN and LIN nodes.

The lowest level of the table displays the bus function blocks that are assigned to the bus access requests of the bus configuration. The function blocks were newly added to the signal chain and they are named after the related communication cluster.



Interim result

You automatically assigned the bus access requests of the Restbus_BusConfiguration to bus accesses. You can now assign hardware resources to the bus function blocks. Continue with the next part.

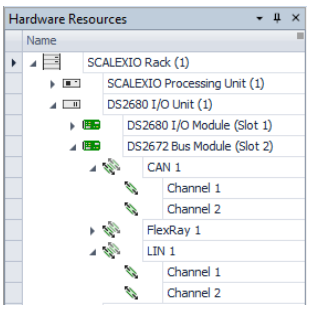
Part 2

To assign hardware resources to bus function blocks

1 In the Hardware Resource Browser, expand the DS2680 I/O Unit.

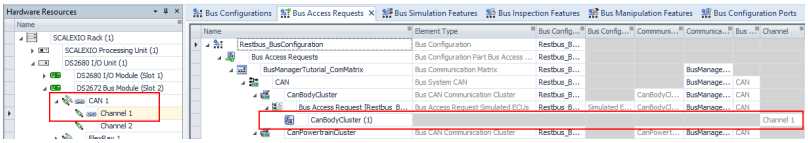
The modules of the DS2680 I/O Unit are displayed.

2 Expand the DS2672 Bus Module to the channels of the CAN 1 and LIN 1 channel types.



3 Select Channel 1 of the CAN 1 channel type and drag it to the CanBodyCluster (1) node in the Bus Access Requests table.

In the Hardware Resource Browser, the assignment of Channel 1 is indicated by a chain symbol. In the Bus Access Requests table, the Channel column displays the assigned channel for the CanBodyCluster (1) function block.



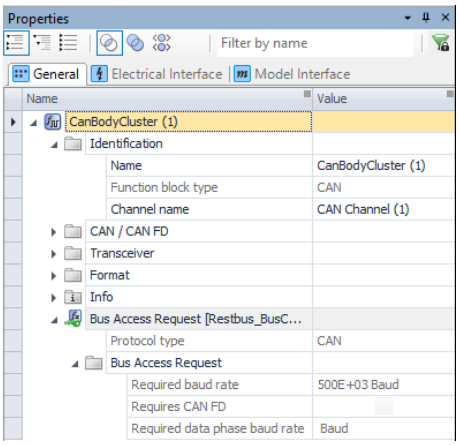
4 Assign hardware resources to the remaining bus function blocks as follows:

Channel Type	Channel	Bus Function Block
CAN 1	Channel 2	CanPowertrainCluster (1)
LIN 1	Channel 1	LinDoorCluster (1)

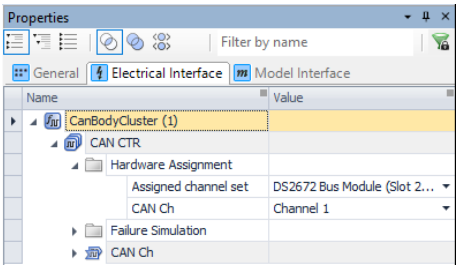
5 In the Bus Access Requests table, select the CanBodyCluster (1) node.

The Properties Browser displays the properties of the CAN function block:

- The General page displays the function block type and the requirements of the assigned bus access request.



- The Electrical Interface page displays the assigned hardware resource.



Result

You specified the hardware access by automatically assigning all the bus access requests of the Restbus_BusConfiguration to suitable bus accesses, i.e., CAN and LIN function blocks. Then, you assigned hardware resources to the function blocks.

What's next

In the next step, you will build a real-time application.

Step 3: How to Build a Real-Time Application

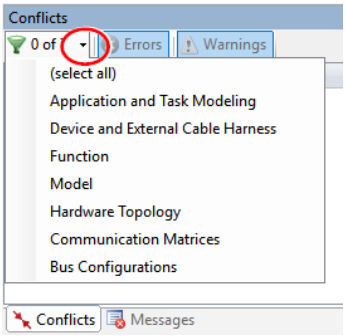
Objective

To build a real-time application, first check the Conflicts Viewer for conflicts that affect the build process and/or the code generation for bus configurations. If no such conflicts exist, you can start the build process.

Part 1

To check the Conflicts Viewer for conflicts

1 In the Conflicts Viewer, click the expand arrow of the context set filter.



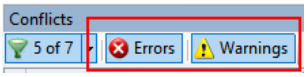
The context sets are displayed. In the illustration above, no context sets are selected.

Tip

The selection of context sets is stored locally on the host PC and applies to all ConfigurationDesk applications. Therefore, the selected context sets in your ConfigurationDesk application might differ.

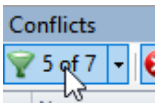
2 Make sure that at least the Application and Task Modeling, Function, Model, Hardware Topology, and Bus Configurations context sets are selected.

3 Make sure that the Errors and Warnings filters are selected.



The Conflicts Viewer displays all conflicts of the selected context sets and filters. However, by specifying the hardware access, you automatically resolved the last conflicts of the active ConfigurationDesk application. Therefore, the Conflicts Viewer does not display any conflicts.

4 Click the context set filter.



The evaluation of all ConfigurationDesk conflicts is disabled.

Note

For optimum performance, it is recommended to enable the evaluation of conflicts only when and as long it is required.

Interim result

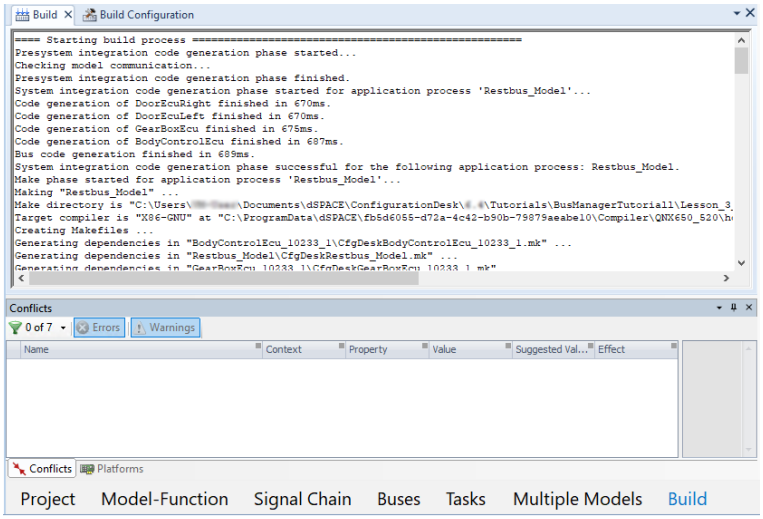
You checked the Conflicts Viewer for conflicts that affect the build process and/or the code generation for bus configurations. Because no such conflicts exist, you can now build a real-time application. Continue with the next part.

Part 2

To build a real-time application

1 On the Home ribbon, click  .

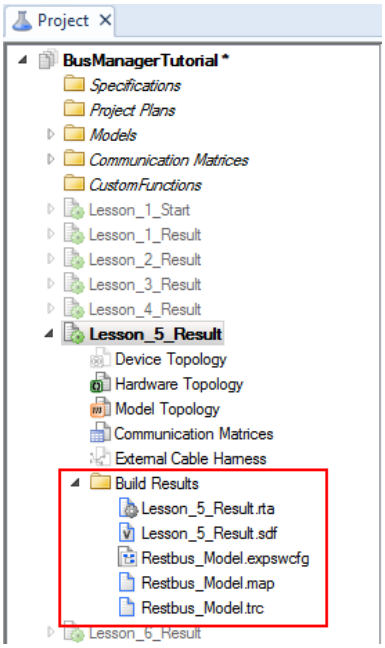
The build process starts. The Build view set is selected and the Build Log Viewer displays the progress of the build process in chronological order.



The Build Log Viewer informs you whether the build process finished successfully.

2 After the build process has finished, select the Project view set.

The Project Manager displays the BusManagerTutorial project and the active application. For this application, a Build Results folder is available, containing the files that were generated during the build process.



Result

You built a real-time application that includes the bus communication configured in the Restbus_BusConfiguration.

What's next

With this step, you finished the lesson. For a summary of the lesson and for more information, refer to Result of Lesson 5 on page 123.

Result of Lesson 5

Result

In this lesson, you learned the basic principles of building a real-time application that includes bus configurations:

- To include bus configurations in a real-time application, you must assign the bus access requests to bus accesses, i.e., to bus function blocks (CAN, LIN).
- To specify the access to real-time hardware, you must assign suitable hardware resources to the bus function blocks. For this purpose, you need a hardware topology.
- Conflicts can affect the build process and/or the code that is generated for bus configurations. The Conflicts Viewer can support you in identifying and resolving such conflicts.
- The files that are generated during the build process are stored to the Build Results folder that is added to the active ConfigurationDesk application when you start the build process for the first time.

Prepared result application

The Lesson_5_Result application provides the result of this lesson. You can activate the application to compare it with your own results, for example. For more information, refer to Tutorial project and applications on page 23.

Further information

- For more information on bus access requests, refer to Handling Bus Access Requests (ConfigurationDesk Bus Manager Implementation Guide 🕮).
- For more information on the **Conflicts Viewer** and Bus Manager-related conflicts, refer to Handling Bus Manager-Related Conflicts (ConfigurationDesk Bus Manager Implementation Guide 🕮).
- For information on the **CAN** and **LIN** function blocks, e.g., on their configuration features, refer to CAN (ConfigurationDesk I/O Function Implementation Guide 🕮) and LIN (ConfigurationDesk I/O Function Implementation Guide 🕮), respectively.
- For more information on hardware topologies, refer to Working with Hardware Topologies (ConfigurationDesk Real-Time Implementation Guide 🕮).
- For information on handling available real-time hardware in ConfigurationDesk, refer to Handling Registered Hardware (ConfigurationDesk Real-Time Implementation Guide 🕮).
- For more information on building real-time applications, refer to Building Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 🕮).
- For more information on the generated files, refer to Build Result Files of the Build Process (ConfigurationDesk Real-Time Implementation Guide 🕮).

Where to go from here

You can now continue with a lesson that is relevant for your use scenario. Refer to Workflows for Using the Bus Manager on page 27.

Lesson 6: Configuring Bus Communication for Inspection Purposes

Where to go from here

Information in this section

Overview of Lesson 6 125

Step 1: How to Assign Communication Matrix Elements to the Inspection Part of a Bus Configuration 126

Step 2: How to Add Bus Configuration Features for Inspection Purposes 128

Result of Lesson 6 130

Overview of Lesson 6

Inspecting bus communication

To inspect bus communication, you must assign communication matrix elements to the Inspection part of a bus configuration and use bus configuration features to specify the data you want to inspect.

In this lesson, you will inspect the bus communication that is exchanged between the CentralGatewayEcu and the restbus that was simulated in lesson 2. For more information, refer to Use scenarios in this tutorial on page 21.

What you will learn

In this lesson, you will learn how to assign communication matrix elements to the Inspection part of a bus configuration and configure assigned elements by using bus configuration features.

Before you begin

Before you begin this lesson, the following preconditions must be met:

- The following licenses are accessible:
  - CAN module license (CFD_I_CAN)
  - LIN module license (CFD_I_LIN)

  Refer to How to Identify Accessible Licenses on page 62.
- The result of lesson 3 is available and the related application is active.

  If you did not work through lesson 3, you can activate the Lesson_3_Results application. Refer to How to Activate an Application in the BusManagerTutorial Project on page 61.

  Alternatively, you can use other applications as the starting point. For an overview of the suitable applications, refer to Overview of Lessons on page 25.

Steps

This lesson contains the following steps:

- Step 1: How to Assign Communication Matrix Elements to the Inspection Part of a Bus Configuration on page 126
- Step 2: How to Add Bus Configuration Features for Inspection Purposes on page 128

Result and further information

The lesson concludes with a summary of the lesson content and with further information. Refer to Result of Lesson 6 on page 130.

Duration

Completing this lesson will take you about 15 minutes.

Step 1: How to Assign Communication Matrix Elements to the Inspection Part of a Bus Configuration

Objective

To inspect bus communication, you must assign communication matrix elements to the Inspection part of a bus configuration.

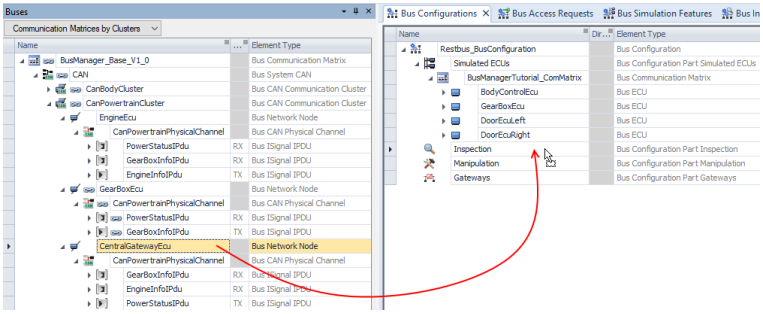
Method

To assign communication matrix elements to the Inspection part

1. Switch to the Buses view set, if necessary.
2. Select the Bus Configurations table.

   The Restbus_BusConfiguration, including its parts, is displayed.
3. In the Buses Browser, select Communication Matrices by Clusters.
4. In the Buses Browser, expand the network nodes of the CanPowertrainCluster to their PDUs.

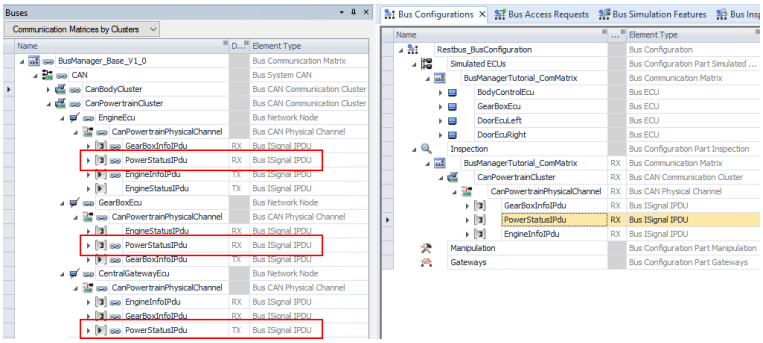
5 Drag the CentralGatewayEcu node of the CanPowertrainCluster to the Inspection part of the Restbus_BusConfiguration.



All the PDUs and ISignals that are transmitted and received by the CentralGatewayEcu via the CanPowertrainCluster are assigned to the Inspection part.

The inspection of bus communication focuses on the bus communication that is exchanged via a communication cluster, i.e., the bus communication that can be received by bus members. However, the involved specific sending and receiving ECUs are out of focus. Because of this, the elements are assigned in the context of the communication cluster and their direction is RX.

For example, the PowerStatusIPdu is transmitted by the CentralGatewayECU and received by the GearBoxEcu and EngineEcu. To inspect this PDU on the CanPowertrainCluster, it is assigned to the Inspection part once as an RX PDU. Nevertheless, in the Buses Browser the PDU is marked by the chain symbol ( ) for all the involved network nodes.



Tip

For assigned elements, the Related TX ECUs and Related RX ECUs columns of the Bus Configurations table display the sending and receiving ECUs as specified in the communication matrix.

6 In the Buses Browser, select the CentralGatewayEcu node of the CanBodyCluster and drag it to the Inspection part of the Restbus_BusConfiguration.

All the PDUs and ISignals that are exchanged with the CentralGatewayEcu via the CanBodyCluster are assigned to the Inspection part.

Result

You assigned the PDUs and ISignals that are exchanged with the CentralGatewayEcu via the CanPowertrainCluster and CanBodyCluster to the Inspection part of the bus configuration.

The name of the communication matrix node that is available for the Inspection part is derived from the communication matrix node that is available for the Simulated ECUs part. Therefore, it was not necessary to change the name of the communication matrix node in this step.

Tip

The names of communication matrix nodes that are available for the same communication matrix in a bus configuration are automatically synchronized.

What's next

In the next step, you will add bus configuration features to elements of the Inspection part to configure bus communication for inspection purposes.

Step 2: How to Add Bus Configuration Features for Inspection Purposes

Objective

To configure bus communication for inspection purposes, you must add bus configuration features to elements that are assigned to the Inspection part of a bus configuration.

Part 1

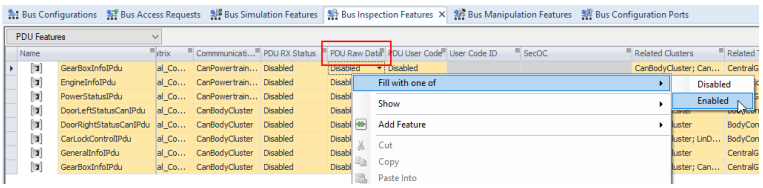
To add bus inspection features to PDUs

1. Select the Bus Inspection Features table.

   The PDU Features subview is active, displaying all PDUs that are assigned to the Inspection part of bus configurations.

2. Select all displayed PDUs.

3. In the PDU Raw Data column, right-click one of the highlighted cells and select Fill with one of - Enabled from the context menu.



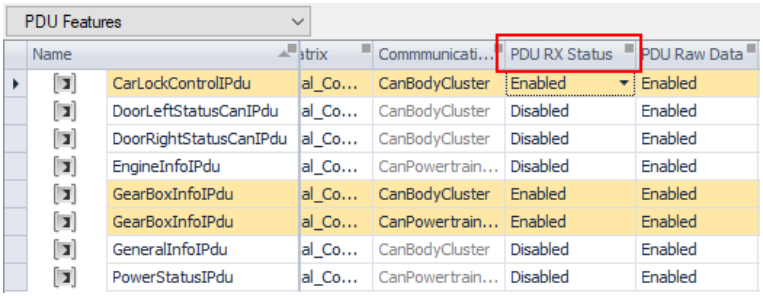
   The PDU Raw Data feature is added to all the PDUs.

4. Click the Name column header to sort the PDUs alphabetically.

   The sorting order is indicated by an arrow (▲) in the column header.

5 Deselect the PDUs except for all CarLockControlIPdu and GearBoxInfoIPdu PDUs.

6 For the selected PDUs, set PDU RX Status to Enabled.



The PDU RX Status feature is added to the selected PDUs.

7 Right-click any of the highlighted cells and select Show - Show in Bus Configurations Table from the context menu.

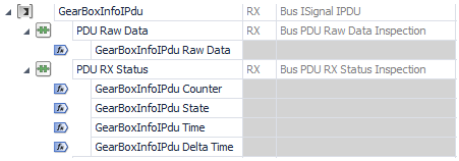
The Bus Configurations table is displayed, and the PDUs are highlighted in the table.

8 Expand the PDU nodes.

The lower-level elements of the PDUs are displayed, i.e., their PDU Raw Data and PDU RX Status feature nodes and their ISignals.

9 Expand a PDU Raw Data and a PDU RX Status feature node of a PDU.

The function ports of both features are displayed: One function port for the PDU Raw Data feature, four function ports for the PDU RX Status feature.



At run time, the function ports provide the raw data and status information of the inspected PDUs, e.g., the number of received PDU instances or the delta time between two consecutively received PDU instances.

Tip

For function ports of all bus inspection features, test automation support is enabled by default. This lets you access the function port values via experiment software and automation scripts without any additional configuration steps.

Interim result

You added bus inspection features to PDUs. You can now add bus inspection features to ISignals. Continue with the next part.

Part 2

To add bus inspection features to ISignals

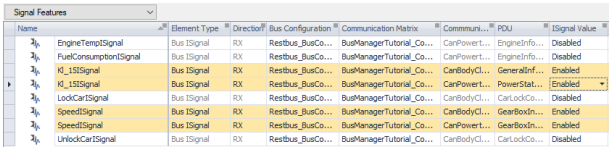
1 In the Bus Inspection Features table, select the Signal Features subview.

All ISignals that are assigned to the **Inspection** part of bus configurations are displayed.

2 Click the **Name** column header to sort the ISignals alphabetically.

3 Select all **Kl_15ISignal** and **SpeedISignal** ISignals.

4 For the selected ISignals, set **ISignal Value** to **Enabled**.



The **ISignal Value** feature is added to the selected ISignals.

Tip

The **ISignal Value** feature is also available as a bus simulation feature. In contrast to the simulation feature, you must manually add the **ISignal Value** inspection feature to assigned ISignals.

Result

You added bus configuration features to PDUs and ISignals that are assigned to the **Inspection** part of the bus configuration.

Tip

After you configured a reasonable set of bus communication, it is recommended to check the configured bus communication for conflicts. However, this step is skipped in this lesson.

What's next

With this step, you finished the lesson. For a summary of the lesson and for more information, refer to Result of Lesson 6 on page 130.

Result of Lesson 6

Result

In this lesson, you learned the basic principles of configuring bus communication for inspection purposes:

- To inspect bus communication, you must assign communication matrix elements to the **Inspection** part of bus configurations. The elements are assigned in the context of the related communication clusters and their direction is RX.
- To specify the data of the bus communication that is to be inspected, you can add bus inspection features to assigned elements.

- Depending on the specific bus inspection feature, function ports are available. For the function ports, test automation support is enabled by default, i.e., you can access the function port data via experiment software and automation scripts by default.

Prepared result application

The Lesson_6_Result application provides the result of this lesson. You can activate the application to compare it with your own results, for example. For more information, refer to Tutorial project and applications on page 23.

Further information

- For more information on assigning communication matrix elements to bus configurations, refer to:
  - Bus Manager in ConfigurationDesk: Assigning Communication Matrix Elements to Bus Configurations (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Assigning Communication Matrix Elements to Bus Configurations (Bus Manager (Stand-Alone) Implementation Guide 🕮)
- For more information on working with bus configuration features, refer to:
  - Bus Manager in ConfigurationDesk: Basics on Working with Bus Configuration Features (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Basics on Working with Bus Configuration Features (Bus Manager (Stand-Alone) Implementation Guide 🕮)

Where to go from here

You can now configure bus communication for manipulation purposes. Refer to Lesson 7: Configuring Bus Communication for Manipulation Purposes on page 133.

You can also continue with another lesson that is relevant for your use scenario. Refer to Workflows for Using the Bus Manager on page 27.

Lesson 7: Configuring Bus Communication for Manipulation Purposes

Where to go from here

Information in this section

Overview of Lesson 7 133

Step 1: How to Assign Communication Matrix Elements to the Manipulation Part of a Bus Configuration 135

Step 2: How to Add Bus Configuration Features for Manipulation Purposes 137

Step 3: How to Configure the Run-Time Behavior for Manipulating Bus Communication 139

Result of Lesson 7 142

Overview of Lesson 7

Manipulating bus communication

To manipulate bus communication, you must assign communication matrix elements to the **Manipulation** part of a bus configuration and use bus configuration features to configure the manipulation behavior. For example, you can use bus configuration features to manipulate the length of frames or overwrite ISignal values.

In this lesson, you will manipulate bus communication before it is transmitted to the **CentralGatewayEcu**. For more information, refer to Use scenarios in this tutorial on page 21.

Run-time behavior of manipulating bus communication

In general, three enable states can be distinguished for manipulating bus communication at run time:

Enable State	Description
Disabled	Manipulation is disabled and the original data is transmitted.
Permanently enabled	The affected bus configuration element is permanently manipulated until the enable state is changed explicitly.
Temporarily enabled	The affected bus configuration element is manipulated for a defined number of times. After the number of times has elapsed, the enable state switches to disabled.

Depending on whether bus manipulation features can be added to PDUs or ISignals, you can specify the run-time behavior as follows:

Bus Manipulation Feature Added to ...	Specifying the Run-Time Behavior
PDU	You can specify the run-time behavior, i.e., the enable state, separately for each feature. If multiple features are added to one PDU, the features can therefore have different enable states.
ISignal	You can specify the run-time behavior only once per ISignal, regardless of whether one or more manipulation features are added to the ISignal. This is because for ISignals only one manipulation feature can be active at a time. Therefore, a feature switch is available for each ISignal that lets you enable and disable the manipulation of the ISignal, activate a manipulation feature, and select whether it manipulates the ISignal permanently or temporarily.

You can specify the initial run-time behavior and change the specified setting at run time.

What you will learn

In this lesson, you will learn how to assign communication matrix elements to the **Manipulation** part of a bus configuration and configure the assigned elements by using bus configuration features.

Before you begin

Before you begin this lesson, the following preconditions must be met:

- The following licenses are accessible:
  - CAN module license (CFD_I_CAN)
  - LIN module license (CFD_I_LIN)

  Refer to How to Identify Accessible Licenses on page 62.
- The result of lesson 6 is available and the related application is active.

  If you did not work through lesson 6, you can activate the **Lesson_6_Results** application. Refer to How to Activate an Application in the BusManagerTutorial Project on page 61.

  Alternatively, you can use other applications as the starting point. For an overview of the suitable applications, refer to Overview of Lessons on page 25.

Steps

This lesson contains the following steps:

- Step 1: How to Assign Communication Matrix Elements to the Manipulation Part of a Bus Configuration on page 135
- Step 2: How to Add Bus Configuration Features for Manipulation Purposes on page 137
- Step 3: How to Configure the Run-Time Behavior for Manipulating Bus Communication on page 139

Result and further information

The lesson concludes with a summary of the lesson content and with further information. Refer to Result of Lesson 7 on page 142.

Duration

Completing this lesson will take you about 20 minutes.

Step 1: How to Assign Communication Matrix Elements to the Manipulation Part of a Bus Configuration

Objective

To manipulate bus communication, you must assign communication matrix elements to the Manipulation part of a bus configuration.

Method

To assign communication matrix elements to the Manipulation part

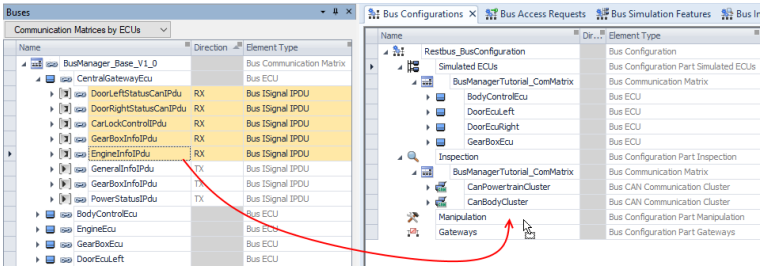
1. Switch to the Buses view set, if necessary.
2. Select the Bus Configurations table.
   The Restbus_BusConfiguration, including its parts, is displayed.
3. In the Buses Browser, select Communication Matrices by ECUs.
   The communication matrix elements are sorted by ECUs.
4. In the Buses Browser, expand the CentralGatewayEcu node.
   The PDUs that are transmitted and received by the CentralGatewayEcu are displayed

   Tip

   All the PDUs are already used in the bus configuration. Therefore, the chain symbol ( ) is displayed for the PDUs.

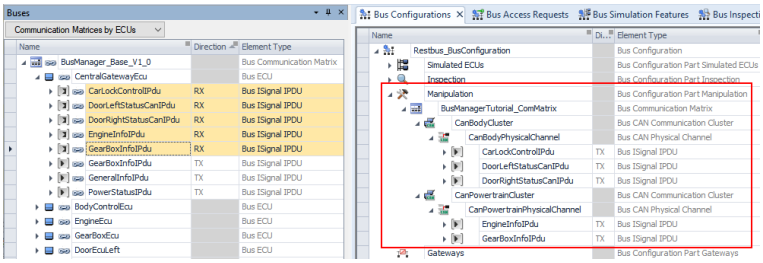
5. Click the Direction column to sort the PDUs by their direction.
   The sorting order is indicated by an arrow ( ) in the column header.

6 Select all the PDUs with the RX direction and drag them to the Manipulation part of the Restbus_BusConfiguration.



All the PDUs and ISignals that are received by the CentralGatewayEcu are assigned to the Manipulation part.

Manipulating bus communication applies to the bus communication that is transmitted via a communication cluster, regardless of the involved sending and receiving ECUs. Therefore, the elements are assigned in the context of the communication cluster and their direction is TX.



Tip

For assigned elements, the Related TX ECUs and Related RX ECUs columns of the Bus Configurations table display the sending and receiving ECUs as specified in the communication matrix.

Result

You assigned the PDUs and ISignals that are received by the CentralGatewayEcu to the Manipulation part of the bus configuration. This lets you manipulate the PDUs and ISignals before they are transmitted on the bus and received by the CentralGatewayEcu.

The name of the communication matrix node that is available for the Manipulation part is derived from the communication matrix nodes that are available for the Simulated ECUs and Inspection parts. Therefore, it was not necessary to change the name of the communication matrix node in this step.

Tip

The names of communication matrix nodes that are available for the same communication matrix in a bus configuration are automatically synchronized.

What's next

In the next step, you will add bus configuration features to elements of the Manipulation part to configure bus communication for manipulation purposes.

Step 2: How to Add Bus Configuration Features for Manipulation Purposes

Objective

To configure bus communication for manipulation purposes, you must add bus configuration features to elements that are assigned to the Manipulation part of a bus configuration.

Part 1

To add bus manipulation features to PDUs

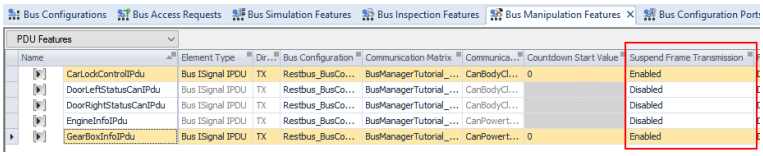
1 Select the Bus Manipulation Features table.

The PDU Features subview is active, displaying all PDUs that are assigned to the Manipulation part of bus configurations.

2 Click the Name column header to sort the PDUs alphabetically.

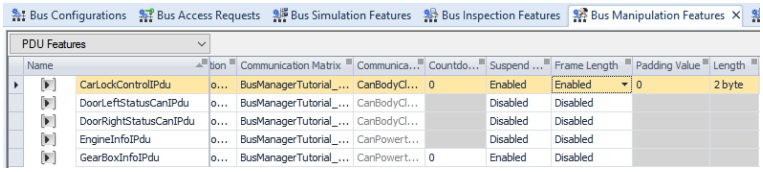
The sorting order is indicated by an arrow (▲) in the column header.

3 Select the CarLockControlIPdu and GearBoxInfoIPdu and set Suspend Frame Transmission to Enabled.



The Suspend Frame Transmission feature is added to both PDUs. Because this is the first manipulation feature that is added to the PDUs, the Countdown Start Value column is enabled for both PDUs. This lets you specify an initial value for the number of times to manipulate the PDUs at run time.

4 For the CarLockControlIPdu, set Frame Length to Enabled.



The Frame Length feature is added to the PDU. This enables the Padding Value and Length columns for the PDU, which let you configure the feature-specific manipulation behavior of the Frame Length feature.

Interim result

You added bus manipulation features to PDUs. You can now add bus manipulation features to ISignals. Continue with the next part.

Part 2

To add bus manipulation features to ISignals

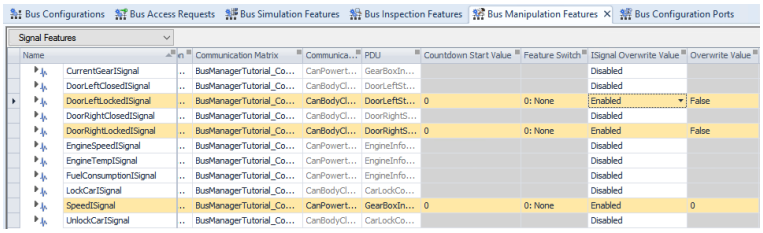
1 In the Bus Manipulation Features table, select the Signal Features subview.

All ISignals that are assigned to the Manipulation part of bus configurations are displayed.

2 Click the Name column header to sort the ISignals alphabetically.

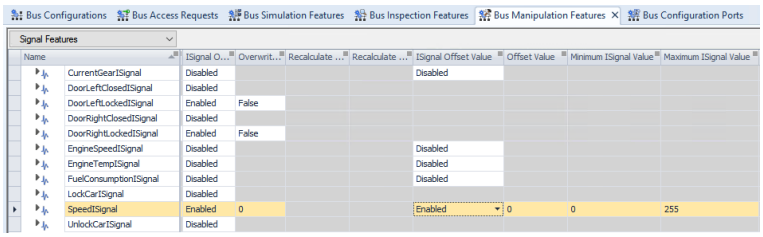
3 Select the DoorLeftLockedISignal, DoorRightLockedISignal, and SpeedISignal.

4 For the selected ISignals, set ISignal Overwrite Value to Enabled.



The ISignal Overwrite Value feature is added to the selected ISignals. This enables the Countdown Start Value, Feature Switch, and Overwrite Value columns for the ISignals. The Countdown Start Value and Feature Switch columns let you specify the initial run-time behavior for manipulating the ISignals. The Overwrite Value column lets you specify an initial value to overwrite the original ISignal value at run time.

5 For the SpeedISignal, set ISignal Offset Value to Enabled.



The ISignal Offset Value feature is added to the ISignal. This enables the Offset Value, Minimum ISignal Value, and Maximum ISignal Value columns for the ISignal, which let you configure the feature-specific manipulation behavior of the ISignal Offset Value feature.

Result

You added bus configuration features to PDUs and ISignals that are assigned to the Manipulation part of the bus configuration.

What's next

In the next step, you will configure the run-time behavior for manipulating the PDUs and ISignals.

Step 3: How to Configure the Run-Time Behavior for Manipulating Bus Communication

Objective

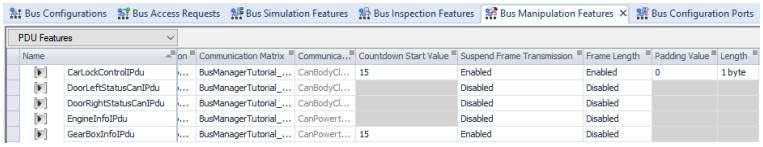
To configure the run-time behavior for manipulating bus communication, you can specify feature-specific and common manipulation settings for each PDU and ISignal to which you added bus manipulation features.

Part 1

To configure the run-time behavior for PDU manipulation

1. In the Bus Manipulation Features table, select the PDU Features subview.
2. Click the Name column header to sort the PDUs alphabetically.
3. For the CarLockControlIPdu and GearBoxInfoIPdu, set Countdown Start Value to 15.

   The specified value applies to all bus manipulation features that are added to each of the PDUs.
4. For the CarLockControlIPdu, set Length to 1.



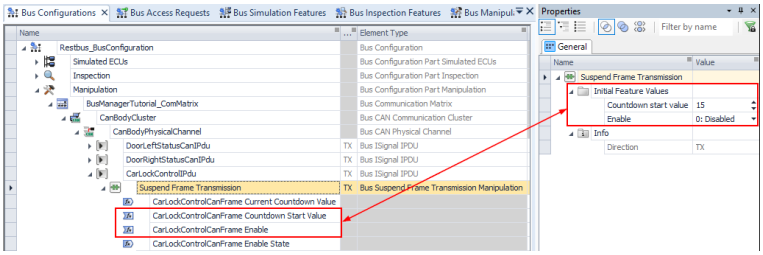
   This value applies only to the Frame Length feature: When the feature is active at run time, the payload length of the frame that transmits the PDU is set to 1 byte by default.
5. Right-click the CarLockControlIPdu and select Show - Show in Bus Configurations Table from the context menu.

   The Bus Configurations table is displayed and the PDU is highlighted in the table.
6. Expand the CarLockControlIPdu node to display its feature nodes and ISignals.
7. Expand the Suspend Frame Transmission feature node to display the function ports that are available for the feature.

   The displayed function ports are common manipulation function ports. They are available for all bus manipulation features that can be added to PDUs.
8. Select the Suspend Frame Transmission feature node.

   The Properties Browser displays the properties of the Suspend Frame Transmission feature node.

   The values of the properties that are available for the Initial Feature Values category can be accessed via their related function ports at run time. The value of the Countdown start value property is derived from the value you specified in the Bus Manipulation Features table.



9 In the Properties Browser, set Enable to 1: Permanently enabled.

By default, the Suspend Frame Transmission feature is permanently enabled at run time, i.e., the frame is not transmitted. At run time, you can switch the enable state via the Enable function port.

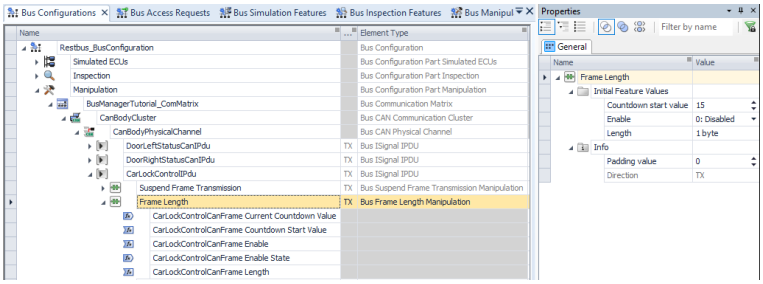
Tip

- You can specify the initial enable state of bus manipulation features that are available for PDUs only via the feature node or the Enable function port. You cannot specify the initial enable state via the Bus Manipulation Features table.
- For function ports of all bus manipulation features, test automation support is enabled by default. This lets you access the function port values via experiment software and automation scripts without any additional configuration steps.

10 In the Bus Configurations table, expand the Frame Length feature node to display its function ports.

11 Select the Frame Length feature node.

The Properties Browser displays the properties of the Frame Length feature node.



In addition to the common Countdown start value and Enable properties, the Initial Feature Values category provides a Length property. The property value is the value you specified in the Bus Manipulation Features table. At run time, you can change the specified value via the Length function port that is available for the Frame Length feature.

Additionally, a Padding value property is available. If you extend the length of a frame, the added bytes are filled with the specified padding value at run time. You can also access this property via the Bus Manipulation Features table. However, you cannot change the specified padding value at run time. Therefore, the property is not part of the Initial Feature Values category and no function port is available.

12 In the Properties Browser, set the Countdown start value to 10.

The specified value applies only to the Frame Length feature. This lets you specify different countdown start values for the bus manipulation features that are added to a PDUs.

Interim result

You configured the run-time behavior for PDU manipulation. You can now configure the run-time behavior for ISignal manipulation. Continue with the next part.

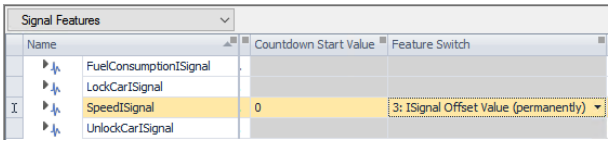
Part 2

To configure the run-time behavior for ISignal manipulation

1 In the Bus Manipulation Features table, select the Signal Features subview.

2 Click the Name column header to sort the ISignals alphabetically.

3 Select the SpeedISignal and set Feature Switch to 3: ISignal Offset Value (permanently).



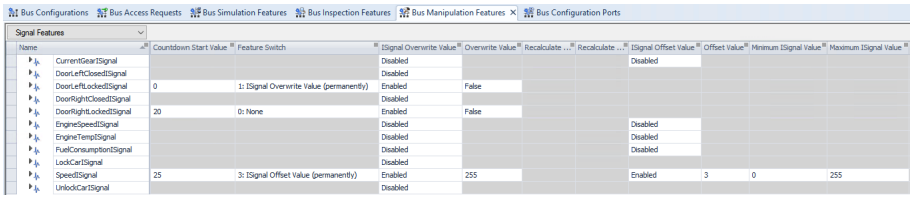
When you add bus manipulation features to ISignals, a feature switch is available for each ISignal. The feature switch lets you specify the initially active bus manipulation feature and its enable state, i.e., permanently or temporarily enabled. You can change the specified initial setting at run time.

4 In the Bus Manipulation Features table, specify further settings as follows:

ISignal	Setting	Value
SpeedISignal	Countdown Start Value	25
	Overwrite Value	255
	Offset Value	3
DoorLeftLockedISignal	Feature Switch	1: ISignal Overwrite Value (permanently)
DoorRightLockedISignal	Countdown Start Value	20

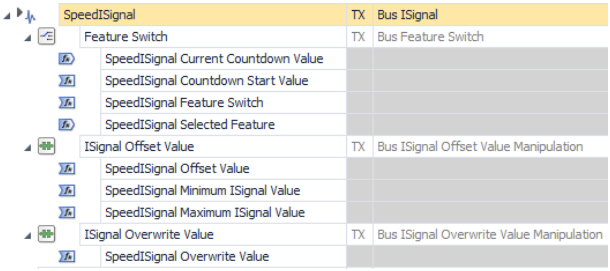
The specified overwrite value and offset value are feature-specific settings and apply only to the ISignal Overwrite Value and ISignal Offset Value feature, respectively. When the related feature is active at run time, the original ISignal value is manipulated by the specified value by default.

After you finish, the following settings are displayed in the Bus Manipulation Features table.



5 Right-click the SpeedISignal and select Show - Show in Bus Configurations Table from the context menu.

6 In the Bus Configurations table, expand the SpeedISignal node and its lower-level nodes.



The feature switch provides the common manipulation function ports. Therefore, these function ports are available only once for the ISignal and not for each of the added bus manipulation features. The bus manipulation features provide only feature-specific function ports.

The feature switch node and the feature nodes provide properties that let you specify common manipulation settings and feature-specific manipulation settings, respectively. These settings are identical to the settings that are available in the Bus Manipulation Features table. You can use the function ports to change the specified settings at run time.

Result

You configured the run-time behavior for PDU and ISignal manipulation.

Tip

After you configured a reasonable set of bus communication, it is recommended to check the configured bus communication for conflicts. However, this step is skipped in this lesson.

What's next

With this step, you finished the lesson. For a summary of the lesson and for more information, refer to Result of Lesson 7 on page 142.

Result of Lesson 7

Result

In this lesson, you learned the basic principles of configuring bus communication for manipulation purposes:

- To manipulate bus communication, you must assign communication matrix elements to the Manipulation part of bus configurations. The elements are assigned in the context of the related communication clusters and their direction is TX.

- To manipulate assigned elements at run time, you must add bus manipulation features to the elements.
- To configure the run-time behavior for manipulation, bus manipulation features provide feature-specific and common manipulation settings. For the settings that can be changed during run time, function ports are available. For the function ports, test automation support is enabled by default, i.e., you can access the function port data via experiment software and automation scripts by default.
- For ISignals, only one bus manipulation feature can be active at a time. Therefore, a feature switch is available for each ISignal that is configured for manipulation. The feature switch lets you specify the active bus manipulation feature.

Prepared result application

The Lesson_7_Result application provides the result of this lesson. You can activate the application to compare it with your own results, for example. For more information, refer to Tutorial project and applications on page 23.

Further information

- For more information on assigning communication matrix elements to bus configurations, refer to:
  - Bus Manager in ConfigurationDesk: Assigning Communication Matrix Elements to Bus Configurations (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Assigning Communication Matrix Elements to Bus Configurations (Bus Manager (Stand-Alone) Implementation Guide 🕮)
- For more information on working with bus configuration features, refer to:
  - Bus Manager in ConfigurationDesk: Basics on Working with Bus Configuration Features (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Basics on Working with Bus Configuration Features (Bus Manager (Stand-Alone) Implementation Guide 🕮)
- For more information on manipulating bus communication, refer to:
  - Bus Manager in ConfigurationDesk: Basics on Bus Manipulation Features (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Basics on Bus Manipulation Features (Bus Manager (Stand-Alone) Implementation Guide 🕮)

Where to go from here

You can now continue with a lesson that is relevant for your use scenario. Refer to Workflows for Using the Bus Manager on page 27.

You can also continue with an experiment in ControlDesk to simulate, manipulate, and inspect the configured bus communication. Refer to Additional Lesson: Experimenting with ControlDesk on page 175.

Lesson 8: Working Without a Behavior Model

Where to go from here

Information in this section

Overview of Lesson 8 145

Step 1: How to Configure Bus Configuration Elements for Access via Experiment Software and Automation Scripts 147

Step 2: How to Create an Application Process that Provides a Default Task 152

Step 3: How to Manually Assign a Bus Configuration to an Application Process 152

Result of Lesson 8 153

Overview of Lesson 8

Working without a behavior model

When you configure bus communication with the Bus Manager, you can work without a behavior model, for example, if you only want to access the configured bus communication via experiment software, such as ControlDesk.

When you work without a behavior model, you must manually create at least one application process and task, and assign bus configurations to this application process.

Tip

Application processes are structuring elements that contain tasks. Tasks are pieces of code whose execution is controlled by the simulation platform, e.g., by the real-time operating system (RTOS). Tasks are required during the execution of executable applications.

When you manually create an application process, you can assign a bus configuration to this application process in one of the following ways:

- Specify the access to real-time hardware

  Use this method if you work with hardware topologies.

  Tip

  This method is recommended if you build a real-time application later on.

- Assign the bus configuration to the application process manually

  Use this method if you work without hardware topologies.

  Tip

  This method is recommended if you generate bus simulation containers later on. If you use the Bus Manager (stand-alone), this is the only way to assign a bus configuration to an application process when working without a behavior model.

Note

Each bus configuration must be assigned to exactly one application process. To prevent conflicting configurations, it is therefore recommended to use only one of the methods of assigning a bus configuration to an application process.

In this lesson, you will configure bus communication for simulation purposes and access via experiment software and automation scripts only.

What you will learn

In this lesson, you will learn how to access bus configuration elements via experiment software and automation scripts, create an application process that provides a default task, and assign a bus configuration to this application process manually.

Before you begin

Before you begin this lesson, the following preconditions must be met:

- The following licenses are accessible:
  - CAN module license (CFD_I_CAN)
  - LIN module license (CFD_I_LIN)

  Refer to How to Identify Accessible Licenses on page 62.
- The result of lesson 1 is available and the related application is active.

  If you did not work through lesson 1, you can activate the Lesson_1_Results application. Refer to How to Activate an Application in the BusManagerTutorial Project on page 61.

Steps

This lesson contains the following steps:

- Step 1: How to Configure Bus Configuration Elements for Access via Experiment Software and Automation Scripts on page 147
- Step 2: How to Create an Application Process that Provides a Default Task on page 152
- Step 3: How to Manually Assign a Bus Configuration to an Application Process on page 152

Result and further information

The lesson concludes with a summary of the lesson content and with further information. Refer to Result of Lesson 8 on page 153.

Duration

Completing this lesson will take you about 15 minutes.

Step 1: How to Configure Bus Configuration Elements for Access via Experiment Software and Automation Scripts

Objective

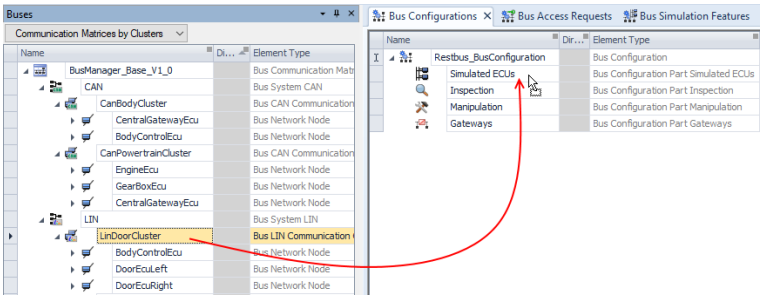
To access bus configuration elements via experiment software or automation scripts, you must do the following:

- Assign communication matrix elements to a bus configuration.
- Add bus configuration features that provide function ports to elements in the bus configuration.
- Enable test automation support for function ports.

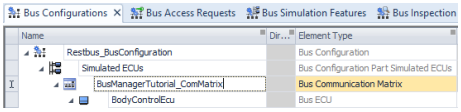
Part 1

To assign communication matrix elements to a bus configuration

1. Switch to the Buses view set, if necessary.
2. In the Buses Browser, select Communication Matrices by Clusters.
3. Drag the LinDoorCluster node to the Simulated ECUs part of the Restbus_BusConfiguration in the Bus Configurations table.



4 In the Name column of the Bus Configurations table, change the name of the communication matrix node to BusManagerTutorial_ComMatrix.

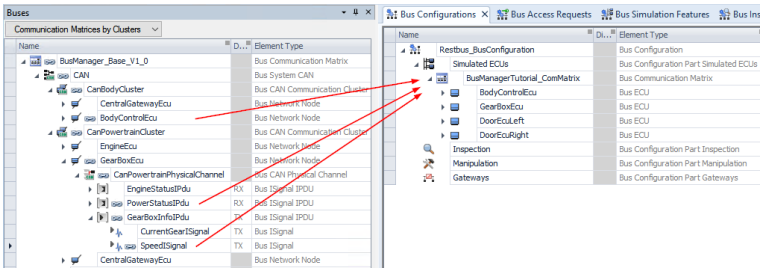


Tip

- It is recommended to use only names without date and version information, especially if you work with automation scripts later on.
- Renaming the communication matrix node in the bus configuration applies only to this bus configuration. It does not apply to other bus configurations or the communication matrix file name.

5 Assign further communication matrix elements to the bus configuration:

- From the CanBodyCluster, assign the BodyControlEcu to the Simulated ECUs part.
- From the CanPowertrainCluster, assign the following elements of the GearBoxEcu to the Simulated ECUs part:
  - PowerStatusIPdu
  - SpeedISignal of the GearBoxInfoIPdu



Interim result

You added communication matrix elements to the Simulated ECUs part of the Restbus_BusConfiguration. To the assigned elements, the ISignal Value and LIN Schedule Table feature are added automatically. You can now enable test automation support for the function ports of these features. Continue with the next part.

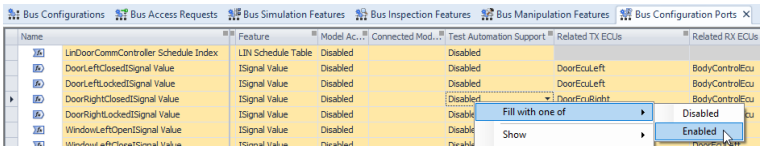
Part 2

To enable test automation support for function ports

1 Select the Bus Configuration Ports table.

2 Select all displayed function ports, e.g., by pressing **Ctrl+A**.

3 Right-click a cell of the Test Automation Support column and select Fill with one of - Enabled from the context menu.



Interim result

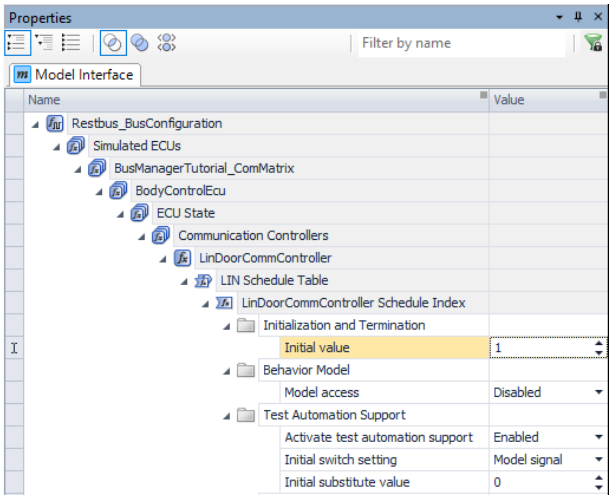
You enabled test automation support for all the available function ports. You can now enable LIN communication by default. Continue with the next part.

Part 3

To enable LIN communication by default

1 Select the LinDoorCommController Schedule Index function port.

2 In the Properties Browser, set the Initial value to 1.



This selects the first LIN schedule table of the LIN master communication controller and enables LIN communication by default.

Tip

By default, the Initial switch setting of function inports is set to Model signal. If function ports are not mapped to ports of a behavior model, the specified Initial value is used as the function port's default value. Therefore, it is not necessary to change the default value of the Initial switch setting if you work without a behavior model.

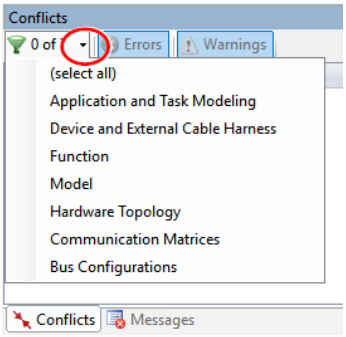
Interim result

You enabled LIN communication by default. You can now check the configured bus communication for conflicts and resolve conflicts, if required. Continue with the next part.

Part 4

To check the configured bus communication for conflicts

1 In the Conflicts Viewer, click the expand arrow of the context set filter.



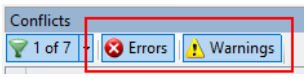
The context sets are displayed. In the illustration above, no context sets are selected.

Tip

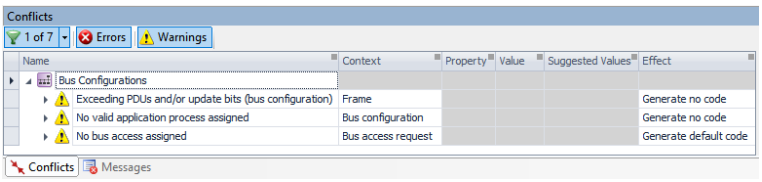
- The selection of context sets is stored locally on the host PC and applies to all ConfigurationDesk applications. Therefore, the selected context sets in your ConfigurationDesk application might differ.
- If you work with the Bus Manager (stand-alone), not all of the context sets that are displayed above are available.

2 Select the Bus Configurations context set and make sure that all other context sets are cleared.

3 Make sure that the Errors and Warnings filters are selected.



The Conflicts Viewer displays all conflicts of the Bus Configurations context set.



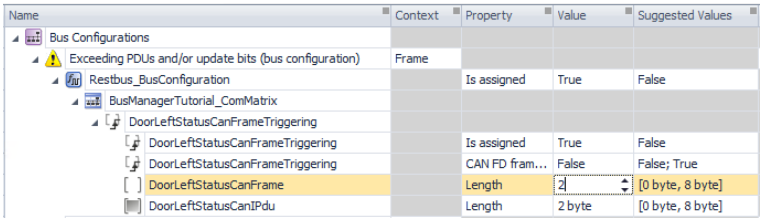
The Exceeding PDUs and/or update bits (bus configuration) conflict results from a frame for which the communication matrix specifies conflicting settings. The other conflicts result from the configuration state of the bus communication in the bus configuration.

Tip

If you work with the Bus Manager (stand-alone), the No bus access assigned conflict is not displayed. This is because the Bus Manager (stand-alone) displays only conflicts that are relevant for the fields of application in which the Bus Manager (stand-alone) can be used.

4 Completely expand the Exceeding PDUs and/or update bits (bus configuration) node.

5 For the DoorLeftStatusCanFrame, specify 2 as the value of the Length property.

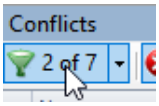


By this, you change the frame length from 1 byte to 2 bytes, which matches the 2 bytes of the related PDU. This change applies to the bus configuration and the communication matrix in the ConfigurationDesk application, and resolves the conflict.

Tip

- You will automatically resolve the No valid application process assigned conflict in step 3 of this lessen.
- In the context of this lesson, you can ignore the No bus access assigned conflict.

6 In the Conflicts Viewer, click the context set filter.



The evaluation of all ConfigurationDesk conflicts is disabled and the Conflicts Viewer displays no conflicts.

Result

You configured bus communication for simulation purposes and access via experiment software and automation scripts.

Note

For optimum performance, note the following:

- Do not enable model access for function ports if you work without a behavior model.
- It is recommended to enable the evaluation of conflicts only when and as long it is required.

What's next

In the next step, you will create an application process that provides a default task.

Step 2: How to Create an Application Process that Provides a Default Task

Objective

To work without a behavior model, you must manually create at least one application process and task. You can do this in one step by creating an application process that provides a default task.

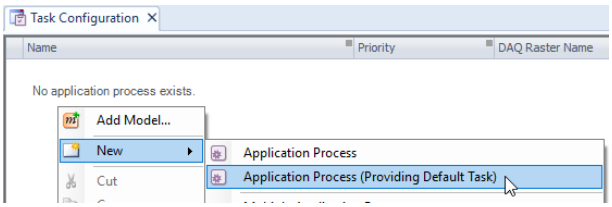
Method

To create an application process that provides a default task

1 Switch to the Tasks view set.

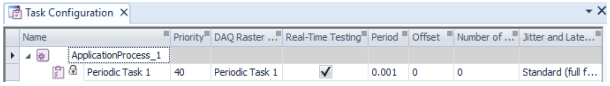
The Task Configuration table is displayed.

2 Right-click an empty area in the Task Configuration table and select New - Application Process (Providing Default Task) from the context menu.



Result

You created an application process and a default task, i.e., a periodic task. The application process and the task are created with default names and default settings.



The default settings are sufficient in most use cases.

What's next

In the next step, you will manually assign the Restbus_BusConfiguration to the application process.

Step 3: How to Manually Assign a Bus Configuration to an Application Process

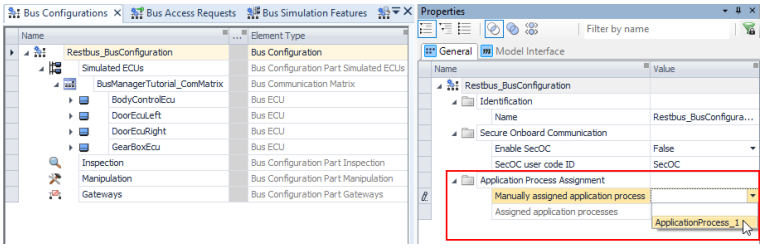
Objective

If you work without a behavior model and without hardware topologies, you must assign bus configurations to application processes manually.

Method

To manually assign a bus configuration to an application process

1. Switch to the Buses view set.
2. In the Conflicts Viewer, select the Bus Configurations context set.
   The Conflicts Viewer displays the No valid application process assigned conflict, which is relevant in the context of this step.
3. In the Bus Configurations table, select the Restbus_BusConfiguration node.
4. In the Properties Browser, select ApplicationProcess_1 from the Manually assigned application process list.



   The Restbus_BusConfiguration is assigned to the ApplicationProcess_1. This resolves the No valid application process assigned conflict.
5. In the Conflicts Viewer, click the context set filter to disable the evaluation of conflicts.

Result

You manually assigned the Restbus_BusConfiguration to the ApplicationProcess_1.

What's next

With this step, you finished the lesson. For a summary of the lesson and for more information, refer to Result of Lesson 8 on page 153.

Result of Lesson 8

Result

In this lesson, you learned the basic principles of working without a behavior model:

- You must manually create at least one application process and task. You can do this in one step by creating an application process that provides a default task.
- If you work without hardware topologies, you must also assign bus configurations to application processes manually.
- If you work with hardware topologies, it is recommended not to assign bus configurations to application processes manually. Instead, bus configurations are assigned to application processes automatically when you specify the hardware access.

Prepared result application

The Lesson_8_Result application provides the result of this lesson. You can activate the application to compare it with your own results, for example. For more information, refer to Tutorial project and applications on page 23.

Further information

- For more information on working without behavior models, refer to:
  - Bus Manager in ConfigurationDesk: Working Without Behavior Models (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Working Without Behavior Models (Bus Manager (Stand-Alone) Implementation Guide 🕮)
- For more information on configuring function ports, refer to:
  - Bus Manager in ConfigurationDesk: Configuring Function Ports for Bus Configuration Features (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Configuring Function Ports for Bus Configuration Features (Bus Manager (Stand-Alone) Implementation Guide 🕮)
- For more information on Bus Manager-related conflicts, refer to:
  - Bus Manager in ConfigurationDesk: Handling Bus Manager-Related Conflicts (ConfigurationDesk Bus Manager Implementation Guide 🕮)
  - Bus Manager (stand-alone): Handling Bus Manager-Related Conflicts (Bus Manager (Stand-Alone) Implementation Guide 🕮)

Where to go from here

You can now continue with a lesson that is relevant for your use scenario. Refer to Workflows for Using the Bus Manager on page 27.

Lesson 9: Generating a Simulink Model and Synchronizing the Model Interfaces in ConfigurationDesk and Simulink

Where to go from here

Information in this section

Overview of Lesson 9 155
Step 1: How to Generate a New Simulink Model Interface 157
Step 2: How to Add a Simulink Model to the ConfigurationDesk Application 159
Step 3: How to Propagate Changes to a Simulink Model 161
Step 4: How to Model Input Bus Signals in the Simulink Model 167
Step 5: How to Analyze the Simulink Model in the ConfigurationDesk Application 169
Step 6: How to Replace a Simulink Model with a Simulink Implementation Container 170
Result of Lesson 9 172

Overview of Lesson 9

Generating a Simulink model interface

If you want to exchange configured bus communication with a behavior model and you do not have a suitable model yet, you can generate a Simulink model interface from within the ConfigurationDesk application. When you do this,

model port blocks are generated for function ports whose model access is enabled, and the model ports are mapped to the function ports.

The model port blocks are available in the ConfigurationDesk application and in a Simulink model. Additionally, a model block periphery can be generated in the Simulink model, i.e., subsystems providing bus creators, bus selectors, constant blocks, etc.

For example, you can use the generated Simulink model interface as follows:

- As a starting point to model a behavior model.
- To copy the generated blocks to an existing behavior model.

In this lesson, you will generate a Simulink model interface for the simulated restbus communication that was configured in lesson 2.

Synchronizing the model interfaces of the ConfigurationDesk application and in Simulink

When you add a Simulink model to the ConfigurationDesk application, you can synchronize the model interfaces if you make changes in the ConfigurationDesk application or the Simulink model. The synchronization applies only to model port blocks, not to other blocks that might be available in the Simulink model.

Replacing Simulink models in the ConfigurationDesk application

You can generate a Simulink implementation container (SIC file) from a Simulink model. If the Simulink model is already available in the ConfigurationDesk application, you can easily replace the model in the ConfigurationDesk application with the generated container. When you do this, the mapping of model ports to function ports remains unchanged.

Tip

For example, this lets you generate bus simulation containers without any additional manual port mapping.

What you will learn

In this lesson, you will learn how to generate a Simulink model interface including a model block periphery, and synchronize the model interfaces in the ConfigurationDesk application and in Simulink. Additionally, you will generate a Simulink implementation container from the Simulink model, and replace the Simulink model in the ConfigurationDesk application with the generated container.

Before you begin

Before you begin this lesson, the following preconditions must be met:

- MATLAB/Simulink and the Model Interface Package for Simulink are installed.

  MATLAB must be connected to the dSPACE installation. If multiple MATLAB installations are connected to the dSPACE installation, one MATLAB installation must be specified as the preferred connection. For more information, refer to Connecting dSPACE Installations to MATLAB (Managing dSPACE Software Installations).

- The following licenses are accessible:
  - CAN module license (CFD_I_CAN)
  - LIN module license (CFD_I_LIN)

  Refer to How to Identify Accessible Licenses on page 62.
- The result of lesson 2 is available and the related application is active.

  If you did not work through lesson 2, you can activate the **Lesson_2_Results** application. Refer to How to Activate an Application in the BusManagerTutorial Project on page 61.

Steps

This lesson contains the following steps:

- Step 1: How to Generate a New Simulink Model Interface on page 157
- Step 2: How to Add a Simulink Model to the ConfigurationDesk Application on page 159
- Step 3: How to Propagate Changes to a Simulink Model on page 161
- Step 4: How to Model Input Bus Signals in the Simulink Model on page 167
- Step 5: How to Analyze the Simulink Model in the ConfigurationDesk Application on page 169
- Step 6: How to Replace a Simulink Model with a Simulink Implementation Container on page 170

Result and further information

The lesson concludes with a summary of the lesson content and with further information. Refer to Result of Lesson 9 on page 172.

Duration

Completing this lesson will take you about 40 minutes.

Step 1: How to Generate a New Simulink Model Interface

Objective

To exchange data with a Simulink model when no suitable model is available yet, you can generate a Simulink model interface for function ports whose model access is enabled.

Tip

When you configured the bus communication for simulation purposes in lesson 2, you already enabled model access for function ports. Therefore, you do not have to enable model access in this step.

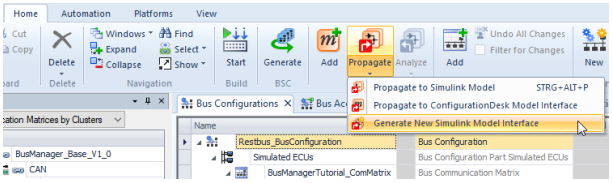
Method

To generate a new Simulink model interface

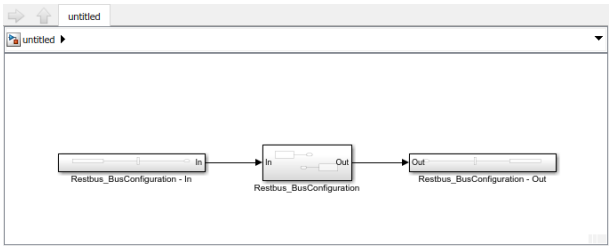
1 Switch to the Buses view set, if necessary.

2 In the Bus Configurations table, select the Restbus_BusConfiguration node.

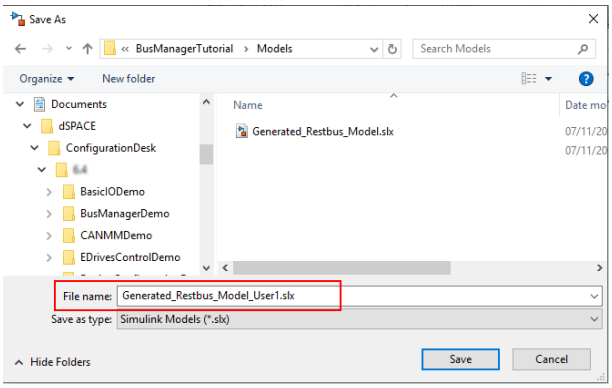
3 On the Home ribbon, click Propagate - Generate New Simulink Model Interface.



If not already open, MATLAB opens and a new Simulink model is generated. This might take a while. The generated model looks as follows:



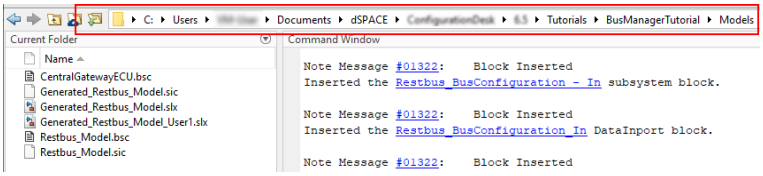
4 Save the Simulink model in the Models folder of the BusManagerTutorial project as `Generated_Restbus_Model_User1.slx`.



Note

The Generated_Restbus_Model.slx model that is already available in the Models folder is used for the result application of this lesson. Do not overwrite this model.

5 In MATLAB, set the working directory to the Models folder of the BusManagerTutorial project.



Result

You generated a Simulink model interface for the function ports of the Restbus_BusConfiguration whose model access is enabled. On the root level of the generated Simulink model interface, three subsystems are available:

Subsystem	Description
Restbus_BusConfiguration - In	Contains the model block periphery that provides data to the Restbus_BusConfiguration subsystem.
Restbus_BusConfiguration	Contains two model port blocks: ▪ Restbus_BusConfiguration_Out This block is a Data Outport block. It receives the data of the Restbus_BusConfiguration - In subsystem and provides it to function inports of the Restbus_BusConfiguration in the ConfigurationDesk application, e.g., to TX ISignal Value function ports. ▪ Restbus_BusConfiguration_In This block is a Data Inport block. It receives data from function outports of the Restbus_BusConfiguration in the ConfigurationDesk application, e.g., from RX ISignal Value function ports. This data is provided to the Restbus_BusConfiguration - Out subsystem.
Restbus_BusConfiguration - Out	Contains the model block periphery that receives data from the Restbus_BusConfiguration subsystem.

What's next

In the next step, you will add the generated Simulink model to the ConfigurationDesk application.

Step 2: How to Add a Simulink Model to the ConfigurationDesk Application

Objective

To make the Simulink model available in ConfigurationDesk or the Bus Manager (stand-alone), you must add it to the ConfigurationDesk application.

Methods in this step

Note

If you use the Bus Manager in ConfigurationDesk, you can add the Simulink model via the ConfigurationDesk menu of the model. This is not possible if you use the Bus Manager (stand-alone).

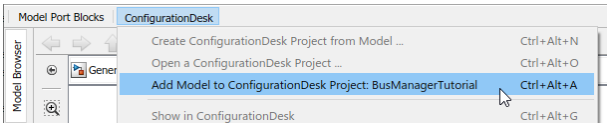
Depending on your Bus Manager variant, select one of the following methods:

- If you use the Bus Manager in ConfigurationDesk, select Method 1 on page 160.
- If you use the Bus Manager (stand-alone), select Method 2 on page 160.

Method 1

To add a Simulink model when using the Bus Manager in ConfigurationDesk

1 In the Simulink Editor, click ConfigurationDesk - Add Model to ConfigurationDesk Project: BusManagerTutorial.



The Simulink model is analyzed and added to the ConfigurationDesk application. In ConfigurationDesk, the Model-Function view set is selected and the Model-Function Mapping Browser displays the Simulink model and the mapped Restbus_BusConfiguration.

2 Switch to the Buses view set.

3 Select the Bus Configuration Ports table.

All function ports for which model access is enabled are mapped to model ports. The Connected Model Ports column displays the mapped model ports.

Skip the following method and continue with the result of this step. Refer to Result on page 161.

Method 2

To add a Simulink model when using the Bus Manager (stand-alone)

1 Switch to the Bus Manager (stand-alone).

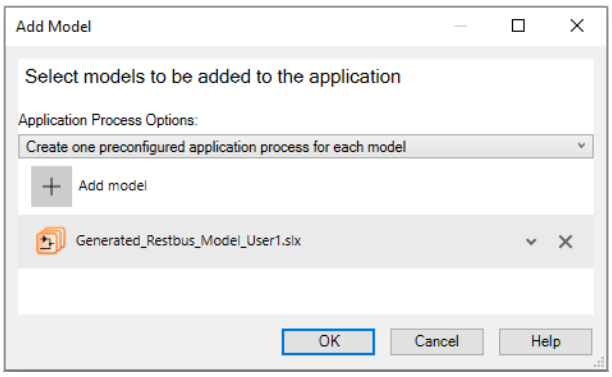
2 On the Home ribbon, click  .

The Add Model dialog opens.

3 In the dialog, click Add model.

A submenu opens, displaying the open Simulink model.

4 In the submenu, select Generated_Restbus_Model_User1.slx.
The submenu closes and the dialog displays the model.



5 In the dialog, click OK.
The Simulink model is analyzed and added to the ConfigurationDesk application. The Model Browser displays the Simulink model.

6 Select the Bus Configuration Ports table.
All function ports for which model access is enabled are mapped to model ports. The Connected Model Ports column displays the mapped model ports.

Result

You added the Generated_Restbus_Model_User1.slx model to the ConfigurationDesk application.

What's next

In the next step, you will enable model access for further function ports of the Restbus_BusConfiguration and propagate the changes to the Simulink model.

Step 3: How to Propagate Changes to a Simulink Model

Objective

To synchronize the model interfaces of the ConfigurationDesk application and a Simulink model, you can propagate changes you make in the ConfigurationDesk application to the Simulink model and vice versa.

Part 1

To enable model access for function ports of the Restbus_BusConfiguration

1 In the Bus Configuration Ports table, right-click the Related TX ECUs column header and select Set Column Filter from the context menu.
The Column Filter Expressions for Column: Related TX ECUs dialog opens.

2 In the dialog, specify `left` as the Filter expression.

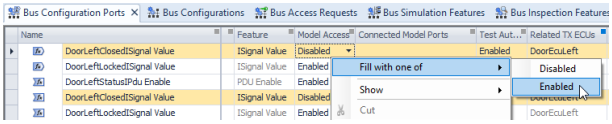
3 Make sure that only the **Regular expression** checkbox is selected and click **OK**.

The **Bus Configuration Ports** table displays only the function ports of elements that are transmitted by the **DoorEcuLeft**.

Note

The filter is active until you clear it explicitly. This applies even when you close ConfigurationDesk or the Bus Manager (stand-alone).

4 Select both **DoorLeftClosedISignal Value** function ports and set **Model Access** to **Enabled**.



Interim result

You enabled model access for a TX and an RX function port. You can now adapt the model port blocks of the ConfigurationDesk application and in the Simulink model accordingly. Continue with the next part.

Part 2

To adapt the model port blocks

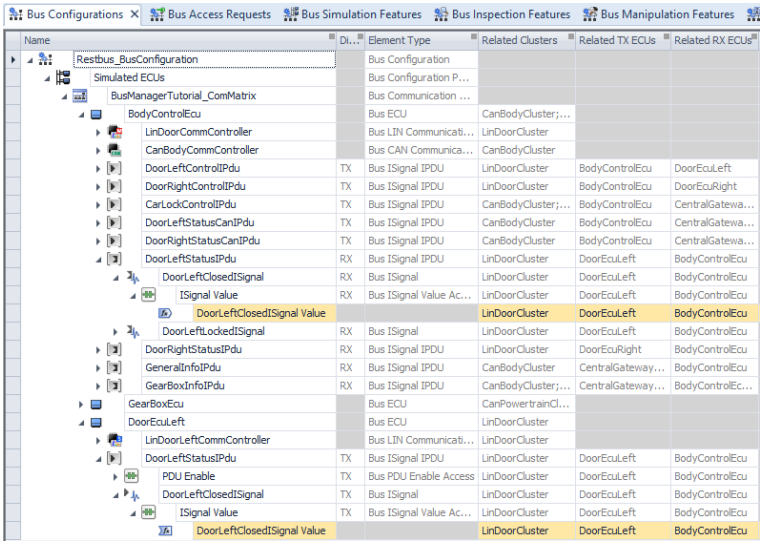
1 In the **Bus Configuration Ports** table, make sure that both **DoorLeftClosedISignal Value** function ports are still selected.

2 In the **Bus Configuration Ports** table, right-click the **Related TX ECUs** column header and select **Clear Column Filter** from the context menu.

All function ports are displayed and the **DoorLeftClosedISignal Value** function ports are still highlighted.

3 Right-click one of the highlighted function ports and select **Show - Show in Bus Configurations Table** from the context menu.

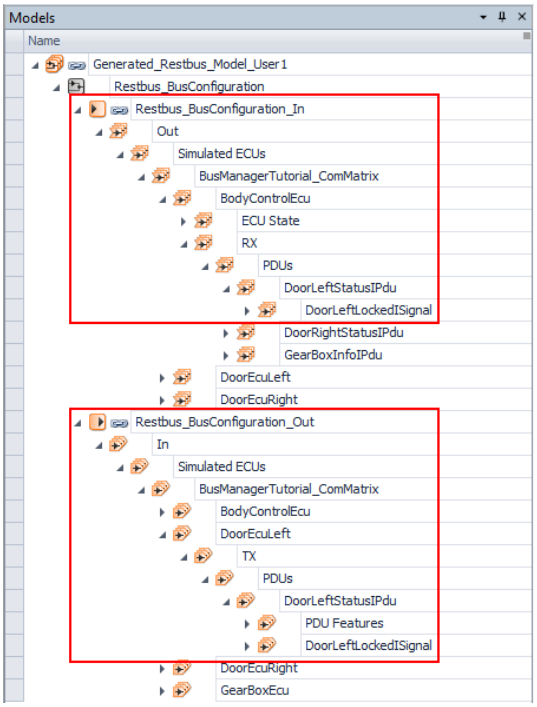
Both function ports are highlighted in the Bus Configurations table.



4 Select the Model Browser.

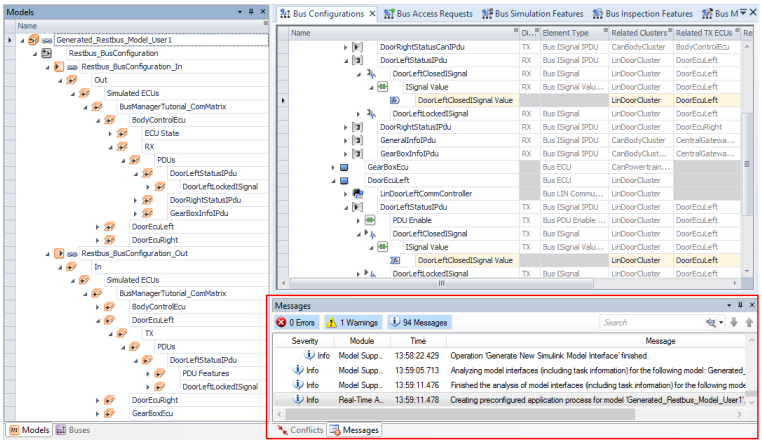
5 In the Model Browser, expand the Restbus_BusConfiguration_In and Restbus_BusConfiguration_Out model port blocks as follows:

- Restbus_BusConfiguration_In - Out - Simulated ECUs - BusManagerTutorial_ComMatrix - BodyControlEcu - RX - PDUs - DoorLeftStatusIPdu
- Restbus_BusConfiguration_Out - In - Simulated ECUs - BusManagerTutorial_ComMatrix - DoorEcuLeft - TX - PDUs - DoorLeftStatusIPdu



For both DoorLeftStatusIPdu nodes, the DoorLeftLockedISignal node is displayed.

6 Select the Message Viewer.

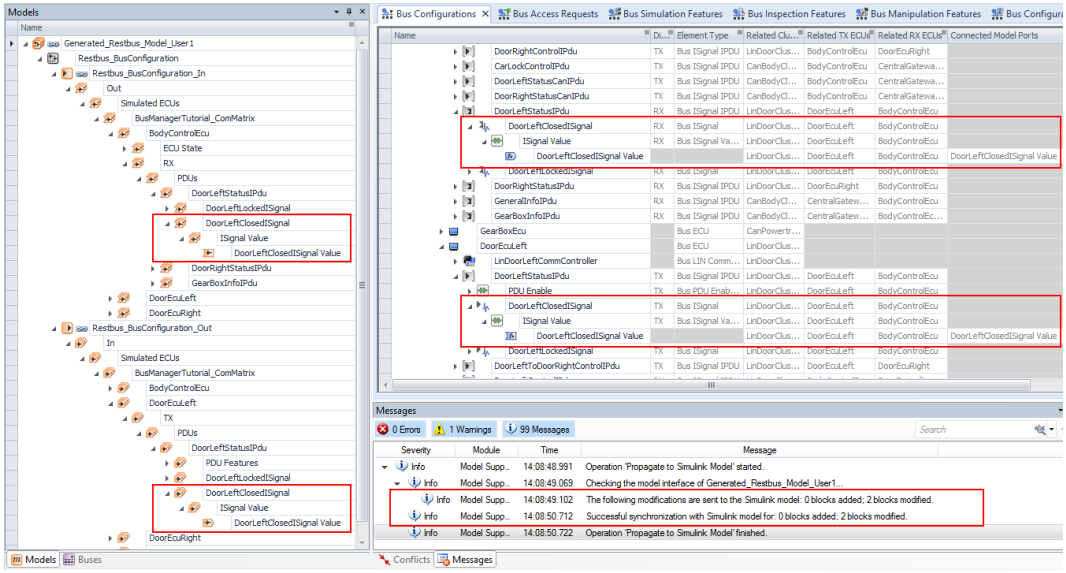


7 In the Bus Configurations table, select the Restbus_BusConfiguration node.

8 On the Home ribbon, click  .

For the DoorLeftClosedISignal Value function ports, suitable model ports are created and mapped to the function ports. The model ports are added to the model port blocks:

- In the Model Browser, two new DoorLeftClosedISignal nodes are available that contain the model ports on their lowest level.
- The Connected Model Ports column of the Bus Configurations table displays the model ports that are mapped to the DoorLeftClosedISignal Value function ports.
- The Message Viewer provides information on the changes that are made in the Simulink model.



Interim result

You adapted the model port blocks of the ConfigurationDesk application and the Simulink model to the changes you made for the function ports of the Restbus_BusConfiguration. You can now explore the effects in the Simulink model. Continue with the next part.

Part 3

To explore the effects of propagation in the Simulink model

1 Select the Generated_Restbus_Model_User1 Simulink model.

2 In the Simulink model, open the Restbus_BusConfiguration subsystem.

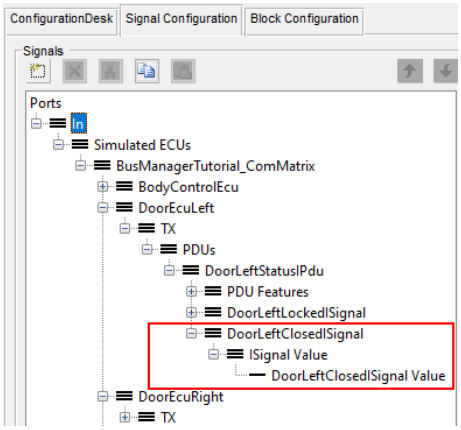
3 In the subsystem, double-click the Restbus_BusConfiguration_Out model port block.

The block dialog opens.

4 In the block dialog, select the Signal Configuration tab.

The In port of the Restbus_BusConfiguration_Out model port block is displayed. This port is a structured data port which contains further hierarchically structured ports and signals.

For the DoorLeftStatusIPdu port of the DoorEcuLeft port, the DoorLeftClosedISignal port is available. This port and its lower-level elements were added to the block by the propagate operation.



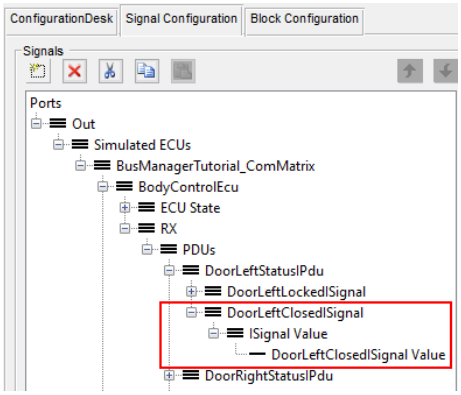
The value of the DoorLeftClosedISignal Value signal is provided to the TX DoorLeftClosedISignal Value function port in the ConfigurationDesk application.

5 Close the dialog.

6 Open the block dialog of the Restbus_BusConfiguration_In model port block.

7 In the block dialog, select the Signal Configuration tab.

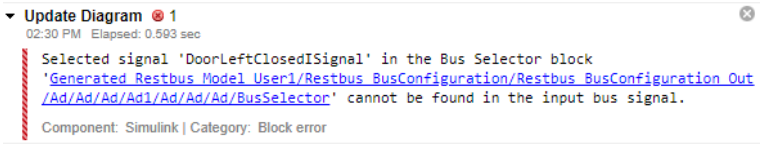
For the Restbus_BusConfiguration_In model port block, the DoorLeftClosedISignal Value signal is added to the RX DoorLeftStatusIPdu of the BodyControlEcu. This signal receives the value from the RX DoorLeftClosedISignal Value function port of the ConfigurationDesk application.



8 Close the dialog.

9 In the Simulink model, press **Ctrl+D** to execute the Update Diagram functionality.

The Diagnostic Viewer opens, displaying an error for the Restbus_BusConfiguration_Out model port block: For the DoorLeftClosedISignal, no input bus signal is available.



The Restbus_BusConfiguration_Out model port block receives data from the model block periphery of the Restbus_BusConfiguration - In subsystem. Because the model block periphery is not affected by propagating changes to Simulink, it is not updated automatically.

To prevent errors in Simulink, you must at least model the required input bus signals manually.

10 Close the Diagnostic Viewer.

Result

You enabled model access for further function ports of the Restbus_BusConfiguration and propagated the changes to Simulink. This automatically updates the model port blocks in the ConfigurationDesk application and in Simulink. However, the model block periphery in the Simulink model is not updated.

What's next

In the next step, you will model the required input bus signal in the Simulink model.

Step 4: How to Model Input Bus Signals in the Simulink Model

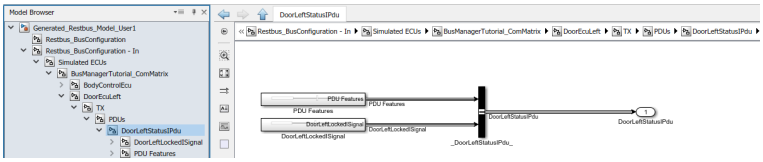
Objective

To provide the required input bus signal to the updated Restbus_BusConfiguration_Out model port block in the Simulink model, you must model the input bus signal manually.

Method

To model the DoorLeftClosedISignal input bus signal

1 In the Simulink model, navigate to the following subsystem: Restbus_BusConfiguration - In - Simulated ECUs - BusManagerTutorial_ComMatrix - DoorEcuLeft - TX - PDUs - DoorLeftStatusIPdu.

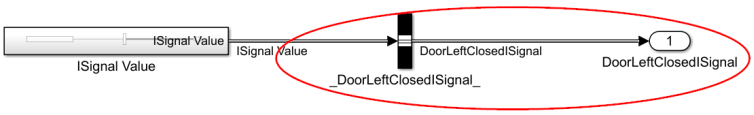


2 Copy the DoorLeftLockedISignal subsystem and paste it on the same level, e.g., by using the **Crtl+C** and **Ctrl+V** shortcuts.

3 Rename the pasted subsystem to DoorLeftClosedISignal.

4 Open the DoorLeftClosedISignal subsystem.

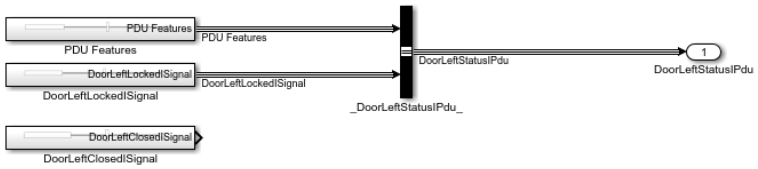
5 Rename all _DoorLeft*Locked*ISignal_ and DoorLeft*Locked*ISignal elements to _DoorLeft*Closed*ISignal_ and DoorLeft*Closed*ISignal, respectively.



6 Open the ISignal Value subsystem.

7 Rename the Const_DoorLeft*Locked*ISignal Value block to Const_DoorLeft*Closed*ISignal Value and the DoorLeft*Locked*ISignal Value mapping line to DoorLeft*Closed*ISignal Value.

8 Navigate to the upper-level DoorLeftStatusIPdu subsystem.



9 Double-click the _DoorLeftStatusIPdu_ bus creator.

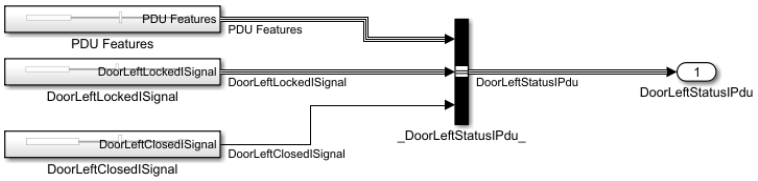
The Block Parameters:_DoorLeftStatusIPdu_ dialog opens.

10 In the dialog, type 3 in the Number of inputs field and click OK.

An inport is added to the bus creator.

11 Map the DoorLeftClosedISignal port to the inport of the bus creator.

12 Rename the mapping line to DoorLeftClosedISignal.



13 Save the model.

Result

You modeled the DoorLeftClosedISignal input bus signal in the Restbus_BusConfiguration - In subsystem. This signal is required by the Restbus_BusConfiguration_Out model port block.

What's next

In the next step, you will analyze the Simulink model in the ConfigurationDesk application.

Step 5: How to Analyze the Simulink Model in the ConfigurationDesk Application

Objective

To make changes in the Simulink model available in ConfigurationDesk or the Bus Manager (stand-alone), you must analyze the model in the ConfigurationDesk application.

Methods in this step

Note

If you use the Bus Manager in ConfigurationDesk, you can analyze the Simulink model via the ConfigurationDesk menu of the model. This is not possible if you use the Bus Manager (stand-alone).

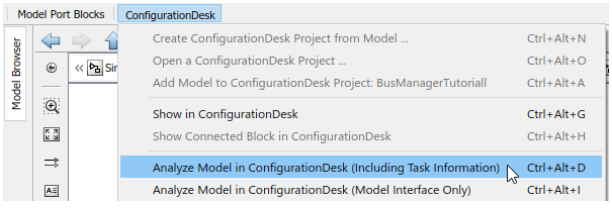
Depending on your Bus Manager variant, select one of the following methods:

- If you use the Bus Manager in ConfigurationDesk, select Method 1 on page 169.
- If you use the Bus Manager (stand-alone), select Method 2 on page 170.

Method 1

To analyze the Simulink model when using the Bus Manager in ConfigurationDesk

1 In the Simulink Editor, click ConfigurationDesk - Analyze Model in ConfigurationDesk (Including Task Information).



The Simulink model is analyzed in the ConfigurationDesk application. When the analysis is finished, the Finished Model Analysis dialog opens.

2 In the dialog, click Ok.

Skip the following method and continue with the result of this step. Refer to Result on page 170.

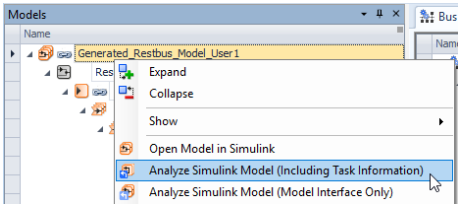
Method 2

To analyze the Simulink model when using the Bus Manager (stand-alone)

1 Switch to the Bus Manager (stand-alone).

2 Select the Model Browser.

3 In the Model Browser, right-click the Generated_Restbus_Model_User1 node and select Analyze Simulink Model (Including Task Information) from the context menu.



The model is analyzed and the Message Viewer provides information on the analysis process.

Result

You analyzed the Simulink model including its task information in the ConfigurationDesk application. This synchronizes the model interfaces.

Tip

During the model analysis including task information, the Simulink Update Diagram functionality is automatically executed and the model is checked for errors.

What's next

In the next step, you will generate a Simulink implementation container from the Simulink model and replace the model with the container in the ConfigurationDesk application.

Step 6: How to Replace a Simulink Model with a Simulink Implementation Container

Objective

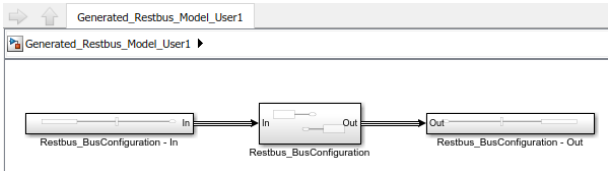
When you generate a Simulink implementation container from a Simulink model that is already available in the ConfigurationDesk application, you can replace the model with the container in the ConfigurationDesk application.

Part 1

To generate a Simulink implementation container

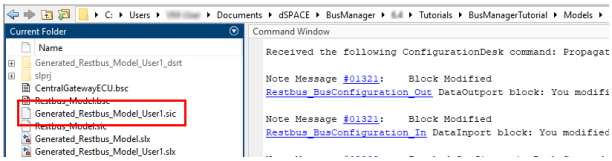
1 Select the Generated_Restbus_Model_User1 Simulink model.

2 In the model, navigate to the root level.



3 Press `Ctrl+B`.

A Simulink implementation container is generated. The related `Generated_Restbus_Model_User1.sic` file is saved to the current working directory of MATLAB, i.e., in the Models folder of the BusManagerTutorial project.



4 Close the Generated_Restbus_Model_User1 Simulink model.

Interim result

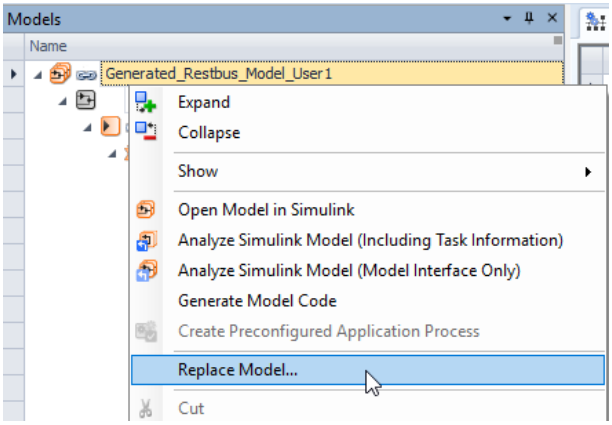
You generated a Simulink implementation container from the Simulink model. You can now replace the Simulink model in the ConfigurationDesk application. Continue with the next part.

Part 2

To replace the Simulink model in the ConfigurationDesk application

1 In the ConfigurationDesk application, select the Model Browser.

2 Right-click the Generated_Restbus_Model_User1 node and select Replace Model from the context menu.



The Replace Model dialog opens.

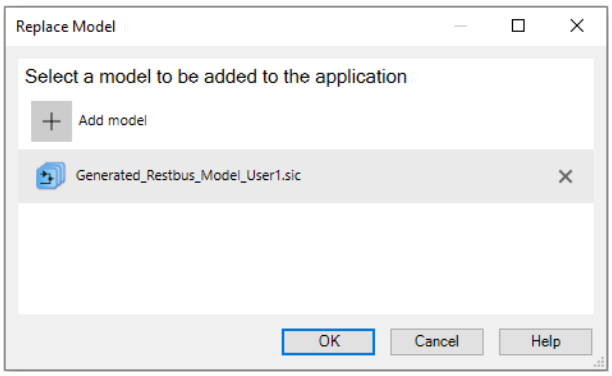
3 In the dialog, click **Add model - Add model from file**.

A standard **Open** dialog opens.

4 In the dialog, navigate to the **Models** folder of the **BusManagerTutorial** project.

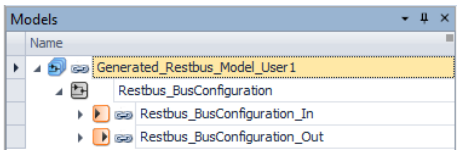
5 Select the **Generated_Restbus_Model_User1.sic** file and click **Open**.

The **Open** dialog closes and the **Replace Model** dialog displays the **Generated_Restbus_Model_User1.sic** file.



6 In the dialog, click **OK**.

The dialog closes and the Simulink model is replaced with the Simulink implementation container.



7 Select the **Bus Configuration Ports** table.

The mapping of function ports to model ports has not changed. The **Connected Model Ports** column displays the mapped model ports.

Result

You replaced the Simulink model with the Simulink implementation container you generated from the Simulink model. The mapping of model ports to function ports remains unchanged.

What's next

With this step, you finished the lesson. For a summary of the lesson and for more information, refer to Result of Lesson 9 on page 172.

Result of Lesson 9

Result

In this lesson, you learned the basic principles of generating a Simulink model, synchronizing the model interfaces in ConfigurationDesk/the Bus Manager

(stand-alone) and Simulink, and replacing a Simulink model with its related Simulink implementation container in the ConfigurationDesk application:

- For function ports with enabled model access, you can generate a Simulink model interface from within the ConfigurationDesk application.
- To exchange data between the ConfigurationDesk application and the Simulink model, model port blocks are generated. Model port blocks are available in the ConfigurationDesk application and in the Simulink model.
- Additionally, a model block periphery can be generated in the Simulink model which provides subsystems with bus creators, bus selectors, constant blocks, etc. This periphery can be used to exchange bus signals between Simulink and the ConfigurationDesk application, for example.
- When you make changes in the ConfigurationDesk application or the Simulink model, you can synchronize the model interfaces.
- Synchronizing the model interfaces applies only to model port blocks. Other blocks in the Simulink model are not affected. Because of this, you might have to adapt parts of the Simulink model manually to the changes of the model port blocks.
- You can generate Simulink implementation containers from Simulink models. When you do this for a Simulink model that is available in the ConfigurationDesk application, you can replace the model in the ConfigurationDesk application with the container. In this case, the mapping of model ports to function ports remains unchanged.

Prepared result application

The Lesson_9_Result application provides the result of this lesson. You can activate the application to compare it with your own results, for example. For more information, refer to Tutorial project and applications on page 23.

Further information

- For more information on specifying the model interface in the ConfigurationDesk application, refer to Specifying the Model Interface (ConfigurationDesk Real-Time Implementation Guide 🕮).
- For more information on working with Simulink models in the ConfigurationDesk application, refer to Working with Simulink Behavior Models (ConfigurationDesk Real-Time Implementation Guide 🕮).

Where to go from here

You can now continue with a lesson that is relevant for your use scenario. Refer to Workflows for Using the Bus Manager on page 27.

Additional Lesson: Experimenting with ControlDesk

Where to go from here

Information in this section

Overview of the Experimenting with ControlDesk Lesson 175
Building a Suitable Executable Application 176
Step 1: How to Create a Project and an Experiment in ControlDesk 179
Step 2: How to Prepare a Layout for Simulating LIN Communication 182
Step 3: How to Simulate LIN Communication 188
Step 4: How to Prepare a Layout for Simulating, Manipulating, and Inspecting CAN Communication 191
Step 5: How to Simulate, Manipulate, and Inspect CAN Communication 193
Result of the Experimenting with ControlDesk Lesson 197

Overview of the Experimenting with ControlDesk Lesson

Experimenting with ControlDesk

After you included bus communication in an executable application, you can load the application to a simulation platform and access the bus communication at run time by using ControlDesk. The ControlDesk Bus Navigator provides a hardware-related view of the configured bus communication and lets you create bus-specific instruments. You can use the bus-specific instruments and common ControlDesk instruments to simulate, manipulate, and inspect the bus communication in real time.

What you will learn

In this lesson, you will learn how to simulate, manipulate, and inspect bus communication by using bus-specific and common ControlDesk instruments.

Before you begin

Before you begin this lesson, the following preconditions must be met:

- ControlDesk is installed and decrypted.
- The ControlDesk Bus Navigator Module is available.
- You have a suitable simulation platform, i.e., a SCALEXIO or MicroAutoBox III system, or VEOS.
- The simulation platform is registered in ControlDesk. Refer to How to Register a Platform (ControlDesk Platform Management 🕮)
- You built a suitable executable application and loaded the application to the simulation platform. Refer to Building a Suitable Executable Application on page 176.

Steps

This lesson contains the following steps:

- Step 1: How to Create a Project and an Experiment in ControlDesk on page 179
- Step 2: How to Prepare a Layout for Simulating LIN Communication on page 182
- Step 3: How to Simulate LIN Communication on page 188
- Step 4: How to Prepare a Layout for Simulating, Manipulating, and Inspecting CAN Communication on page 191
- Step 5: How to Simulate, Manipulate, and Inspect CAN Communication on page 193

Result and further information

The lesson concludes with a summary of the lesson content and with further information. Refer to Result of the Experimenting with ControlDesk Lesson on page 197.

Duration

Completing this lesson will take you about 60 minutes.

Building a Suitable Executable Application

Overview

To work through this lesson, you must build a suitable executable application. The requirements for the application depend on your simulation platform.

Simulation Platform	Required Application	Refer to
▪ SCALEXIO ▪ MicroAutoBox III	Real-time application	Building a suitable real-time application on page 177

Simulation Platform	Required Application	Refer to
VEOS	Offline simulation application	Building a suitable offline simulation application on page 179

Building a suitable real-time application

Preconditions concerning real-time hardware Your real-time hardware must meet the following requirements:

- 1 LIN channel
- 4 CAN channels
- The LIN VBAT and LIN GND pins of the LIN channel are connected to an external 12 V power supply.
- The CAN channels are physically connected as follows:
  - Channel 1 is connected to channel 2.
  - Channel 3 is connected to channel 4.

To establish the physical connection of the CAN channels, you need information on the mapping of CAN signals to bus lines of real-time hardware. The following table provides an overview of further hardware-related information.

Real-Time Hardware		Refer to
SCALEXIO	DS2671 Bus Board	Signal Mapping of the DS2671 Bus Board (SCALEXIO Hardware Installation and Configuration 🕮)
	DS2672 Bus Module	Signal Mapping of the DS2672 Bus Module (SCALEXIO Hardware Installation and Configuration 🕮)
	DS6301 CAN/LIN Board	Signal Mapping of the DS6301 CAN/LIN Board (SCALEXIO Hardware Installation and Configuration 🕮)
	DS6341 CAN Board	Signal Mapping of the DS6341 CAN Board (SCALEXIO Hardware Installation and Configuration 🕮)

Real-Time Hardware		Refer to
MicroAutoBox III	DS1511/DS1511B1 Multi-I/O Board	DS1511 ZIF I/O Connector Pinout (MicroAutoBox III Hardware Installation and Configuration 🕮)
	DS1513 Multi-I/O Board	DS1513 ZIF I/O Connector Pinout (MicroAutoBox III Hardware Installation and Configuration 🕮)
	DS4342 CAN FD Interface Module	DS1514 ZIF I/O Connector Pinout (DS4342) (MicroAutoBox III Hardware Installation and Configuration 🕮)

Required licenses To build a suitable real-time application, the following licenses must be accessible:

- CAN module license (CFD_I_CAN)
- LIN module license (CFD_I_LIN)

Refer to How to Identify Accessible Licenses on page 62.

Building the real-time application To build the real-time application, you must do the following:

1. Activate the Lesson_ExpCD_Start application.

   Refer to How to Activate an Application in the BusManagerTutorial Project on page 61.
2. Create or import a hardware topology that matches your real-time hardware.

   Tip

   You can automatically create a matching hardware topology by registering real-time hardware in ConfigurationDesk or by scanning hardware that is already registered. Refer to Basics on Hardware Topologies (ConfigurationDesk Real-Time Implementation Guide 🕮).
3. Assign hardware resources to the CAN and LIN function blocks that are used in the signal chain. Assign the physically connected CAN channels to the CAN function blocks as follows:

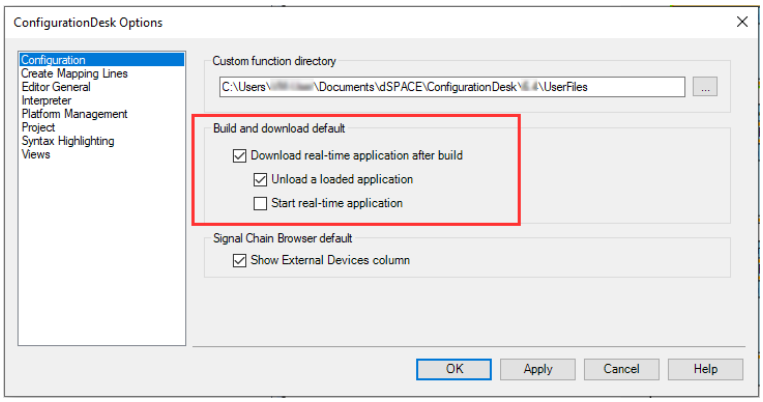
CAN Channel	CAN Function Block
Channel 1	CANBodyCluster_CentralGatewayECU
Channel 2	CANBodyCluster_Restbus
Channel 3	CANPowertrainCluster_CentralGatewayECU
Channel 4	CANPowertrainCluster_Restbus

4. Configure the CAN and LIN function blocks according to the requirements of your real-time hardware (e.g., the transceiver type).

   Refer to:

   - Hardware Dependencies (CAN) (ConfigurationDesk I/O Function Implementation Guide 🕮)
   - Hardware Dependencies (LIN) (ConfigurationDesk I/O Function Implementation Guide 🕮)

5. On the File ribbon, open the Options dialog. Make sure that the following build and download settings are specified on the Configuration page.



6. Build the real-time application.

Building a suitable offline simulation application

Preconditions The VEOS Player must be installed and decrypted.

Building the offline simulation application To build the offline simulation application, you must do the following:

1. In the VEOS Player, click New on the Home ribbon to create an empty OSA file.
2. Import the CentralGatewayECU.bsc and Restbus_Model.bsc files to the VEOS Player. The BSC files are located in the Models folder of the BusManagerTutorial project.

   Refer to How to Import Bus Simulation Containers (VEOS Manual 🕮).

   Tip

   The communication controllers are automatically connected to communication clusters.

3. Save the OSA file to the VEOS Documents folder. Name the file Lesson_ExpCD_Start.osa.
4. Load the OSA file in VEOS without starting the offline simulation application.

   How to Load and Run an Offline Simulation Application (VEOS Manual 🕮).

Step 1: How to Create a Project and an Experiment in ControlDesk

Objective To start working with ControlDesk, you need a project and an experiment.

Part 1

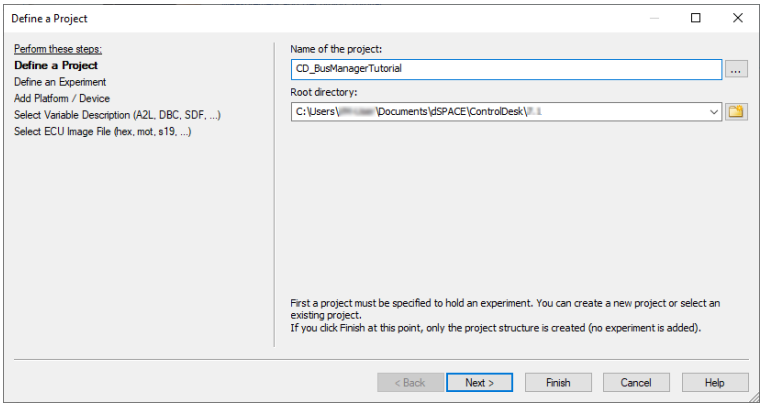
To create a project and an experiment in ControlDesk

1 Start ControlDesk.

2 On the Start Page, click New Project + Experiment.

A dialog opens, displaying the Define a Project page.

3 On the Define a Project page, name the project CD_BusManagerTutorial.



4 Click Next.

The Define an Experiment page opens.

5 On the Define an Experiment page, name the experiment Experiment_BusManagerTutorial.

6 Click Next.

The Add Platform / Device page opens.

7 On the Add Platform / Device page, select the registered platform, i.e., MicroAutoBox III, SCALEXIO, or VEOS.

8 Click Next.

The Select Variable Description (A2L, DBC, SDF, ...) page opens.

9 On the Select Variable Description (A2L, DBC, SDF, ...) page, click Import from file.

The Select Variable Description (A2L, DBC, SDF, ...) dialog opens.

10 In the dialog, navigate to the folder where you saved the real-time application or offline simulation application, i.e.:

- Real-time application: `<ConfigurationDesk documents folder>/Tutorials/BusManagerTutorial/Lesson_ExpCD_Start/Build Results`
- Offline simulation application: `<VEOS documents folder>`

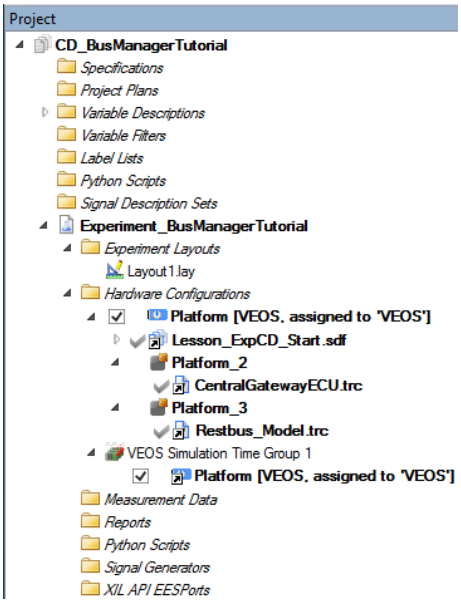
In the folder, a Lesson_ExpCD_Start.sdf file is available.

11 Select the Lesson_ExpCD_Start.sdf file and click Open.

The dialog closes and the SDF file is displayed on the Select Variable Description (A2L, DBC, SDF, ...) page.

12 Click Finish.

The dialog closes and the project and experiment are created. The Project Manager displays the experiment and the selected platform, as shown in the following example.



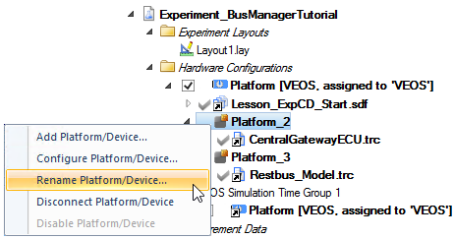
Interim result

You created a project and an experiment in ControlDesk. For an easier identification in the following steps of this lesson, you can now rename the platforms. Continue with the next part.

Part 2

To rename the platforms

1 Right-click the platform to which the CentralGatewayECU.trc file is loaded and select Rename Platform/Device from the context menu.



The Rename Platform/Device dialog opens.

2 In the dialog, specify **`CentralGateway_Platform`** as the platform name.

3 Click OK.

The dialog closes and the platform is renamed.

4 Rename the platform to which the Restbus_Model.trc file is loaded to **`Restbus_Platform`**.

Result

You created an experiment in ControlDesk and renamed the used platforms.

What's next

In the next step, you will prepare a layout for simulating LIN communication.

Step 2: How to Prepare a Layout for Simulating LIN Communication

Objective

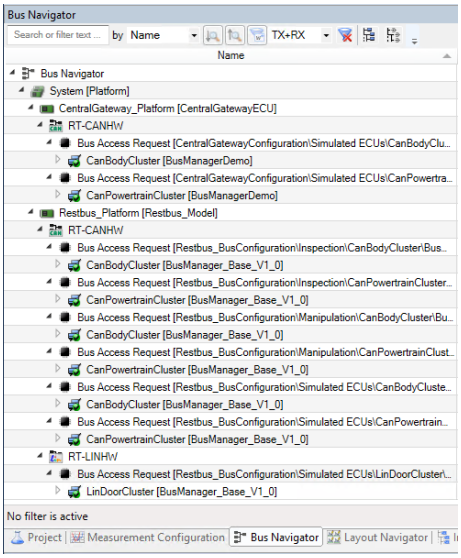
To handle bus communication in a ControlDesk experiment, you can use common and bus-specific instruments which are arranged on one or more layouts.

Part 1

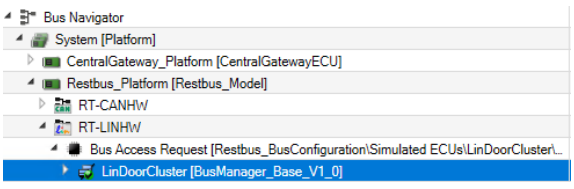
To generate a layout providing a Bus Instrument (TX status type)

1 Select the Bus Navigator.

The Bus Navigator displays a hardware-related view on the bus communication that is accessed via the Lesson_ExpCD_Start.sdf file, starting with the platforms to which the bus communication of the CentralGatewayECU and Restbus_Model bus simulation containers is loaded.



2 Navigate to the LinDoorCluster [BusManager_Base_V1_0] node of Restbus_Platform [<xxx>].

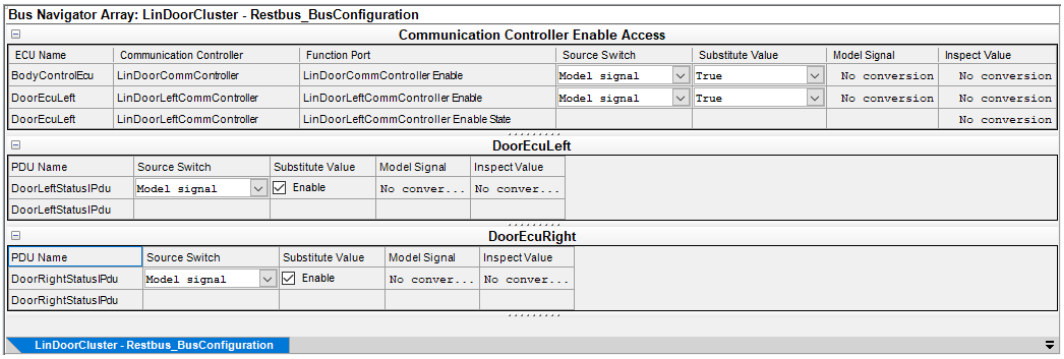


Tip

<xxx> is a placeholder because the displayed value depends on the simulation platform.

3 Right-click the LinDoorCluster [BusManager_Base_V1_0] node and select Generate TX Status Layout from the context menu.

A new layout is generated that provides a Bus Instrument (TX status type). The instrument looks as follows:



- The Communication Controller Enable Access region is available for the Communication Controller Enable feature you added to the BodyControlEcu and the DoorEcuLeft.
- The DoorEcuLeft and DoorEcuRight regions are available for the bus simulation features you added to the TX PDUs of the ECUs, i.e., the PDU Enable feature of the TX DoorLeftStatusIPdu and TX DoorRightStatusIPdu, respectively.
- The regions display all function ports of the related bus simulation features for which you enabled test automation support.
- For function inports, the instrument provides elements that let you access and change the function port values. The values of function outports are read-only.

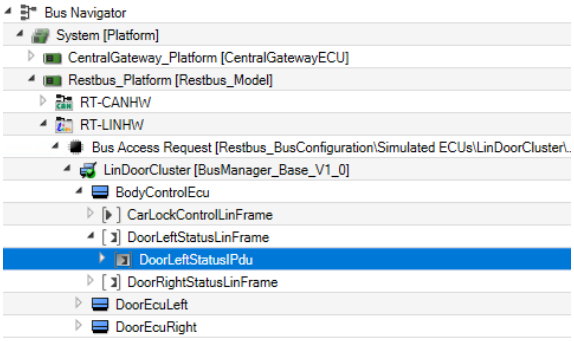
Interim result

You generated a layout that provides a Bus Instrument (TX status type). You can now add further instruments to the layout. Continue with the next part.

Part 2

To add further instruments to the layout

1 In the Bus Navigator, select the following PDU: LinDoorCluster [BusManager_Base_V1_0] - BodyControlEcu - RX DoorLeftStatusLinFrame - RX DoorLeftStatusIPdu.

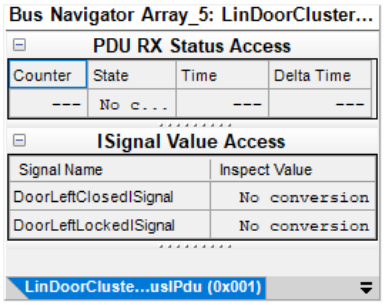


2 Drag the RX DoorLeftStatusIPdu to an empty area of the layout.

A list opens.

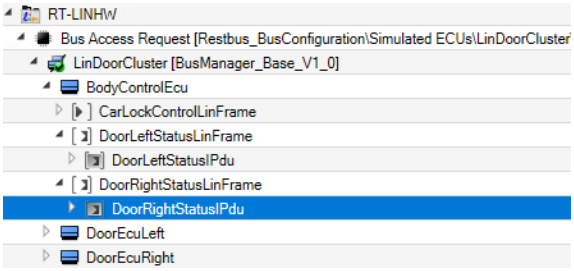
3 In the list, select Bus Navigator Rx Instrument.

A Bus Instrument (RX type) is added to the layout.



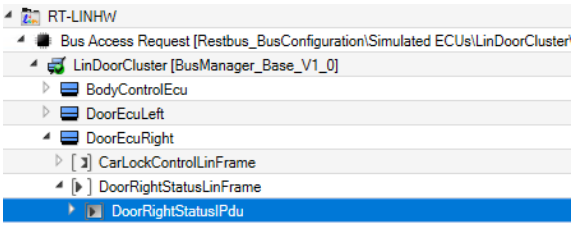
The regions of the instrument are derived from the bus simulation features you added to the RX DoorLeftStatusIPdu and its ISignals.

4 In the Bus Navigator, select the RX DoorRightStatusIPdu that is available for the RX DoorRightStatusLinFrame of the BodyControlEcu.



5 Add a Bus Instrument (RX type) for the RX DoorRightStatusIPdu to the layout.

6 In the Bus Navigator, select the following PDU: LinDoorCluster [BusManager_Base_V1_0] - DoorEcuRight - TX DoorRightStatusLinFrame - TX DoorRightStatusIPdu.

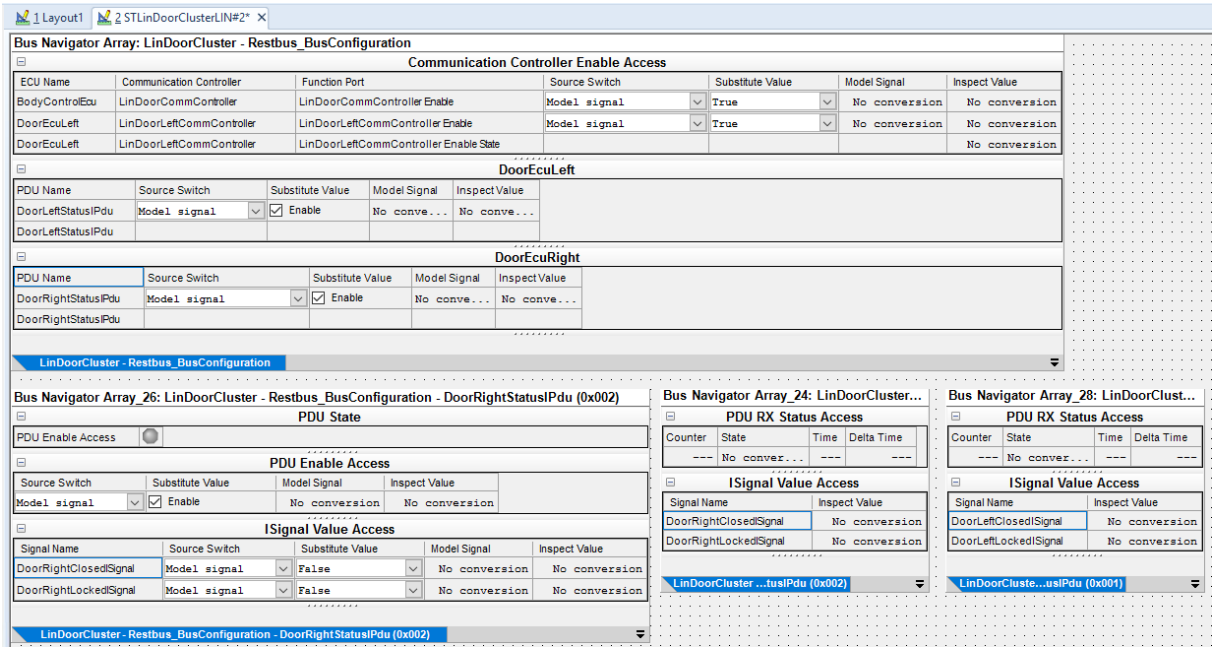


7 Drag the TX DoorRightStatusIPdu to an empty area of the layout and select Bus Navigator Tx Instrument from the list.

A Bus Instrument (TX type) is added to the layout.

8 Arrange the instruments of the PDUs below the Bus Instrument (TX status type) from left to right as follows:

TX DoorRightStatusIPdu - RX DoorRightStatusIPdu - RX DoorLeftStatusIPdu.



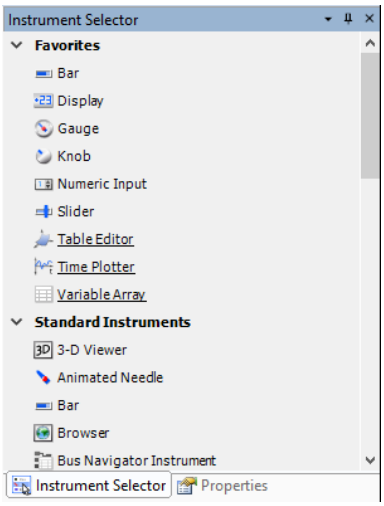
Interim result

You added instruments for the RX and TX DoorRightStatusIPdu and the RX DoorLeftStatusIPdu to the layout. You can now add an instrument to the layout to access the variables of the LIN Schedule Table feature of the LIN master communication controller. Continue with the next part.

Part 3

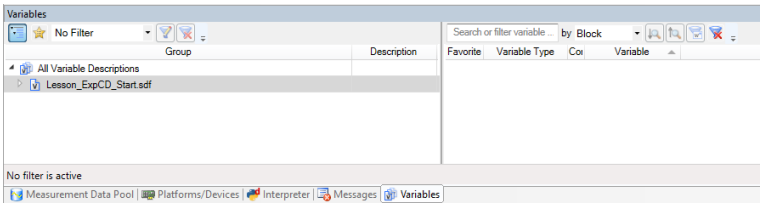
To add an instrument for accessing the variables of the LIN Schedule Table feature.

1 Select the Instrument Selector.



2 From the Instrument Selector, drag a Variable Array instrument to an empty area of the layout.

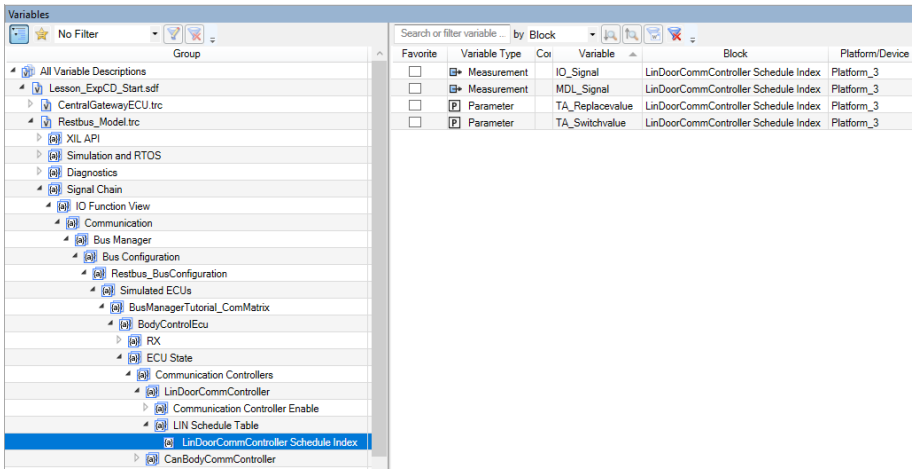
3 Select the Variables controlbar.



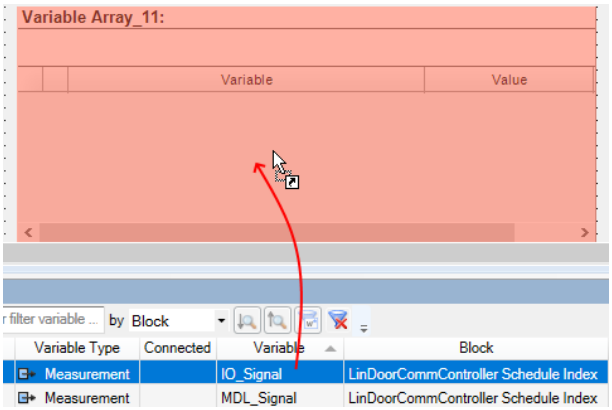
4 In the Variables controlbar, navigate to the following block: Lesson_ExpDC_Start.sdf - Restbus_Model.trc - Signal Chain - IO Function View - Communication - Bus Manager - Bus Configuration - Restbus_BusConfiguration - Simulated ECUs - BusManagerTutorial_ComMatrix - BodyControlEcu - ECU State - Communication Controllers - LinDoorCommController - LIN Schedule Table - LinDoorCommController Schedule Index.

5 Select the LinDoorCommController Schedule Index block.

The variables of the LinDoorCommController Schedule Index block are displayed. These variables let you access the value of the Schedule Index function port.



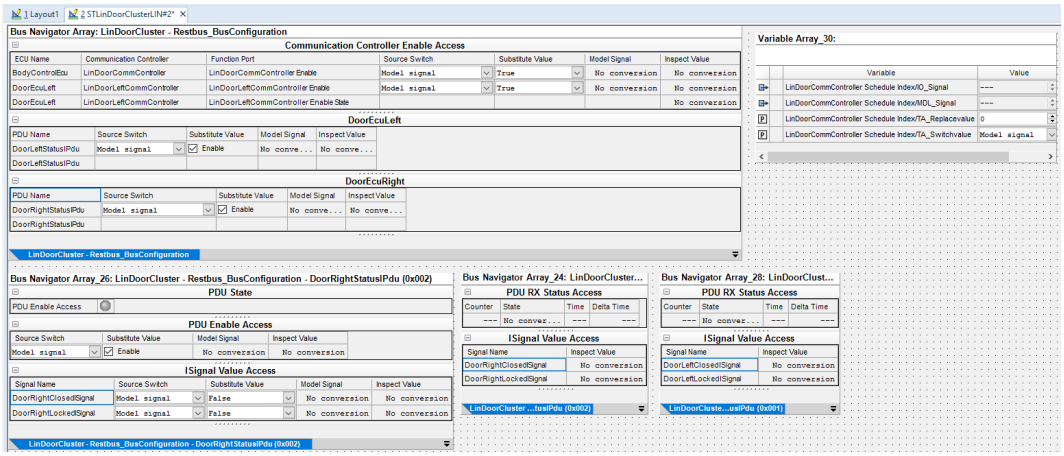
6 Drag the IO_Signal variable to the Variable Array instrument.



The IO_Signal variable is connected to the Variable Array instrument. In the Variables controlbar, a chain symbol indicates the connection state.

7 Drag the MDL_Signal, TA_Replacevalue, and TA_Switchvalue variables to the Variable Array instrument.

8 Arrange the Variable Array instrument as follows:



Result

You prepared a ControlDesk layout for simulating LIN communication.

What's next

In the next step, you will simulate LIN communication.

Step 3: How to Simulate LIN Communication

Objective

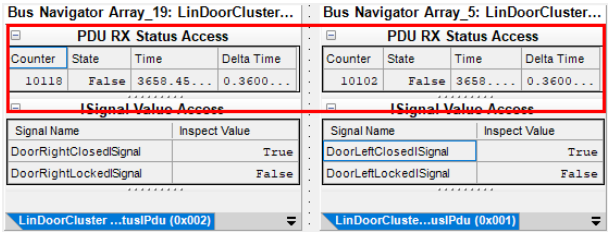
When you start online calibration, you can simulate the LIN communication that you configured with the Bus Manager. You can use ControlDesk instruments to access the LIN communication.

Method

To simulate LIN communication

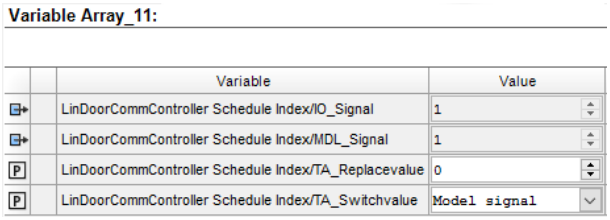
1 On the Home ribbon, click .

Online calibration starts and the LIN communication is active. The Counter, Time, and Delta Time of the PDU RX Status Access regions in both Bus Instrument (RX type) are running. This indicates that the DoorRightStatusIPdu and DoorLeftStatusIPdu are received by the BodyControlEcu.



The variables you connected to the Variable Array instrument let you access the value of the Schedule Index function port:

- The TA_Switchvalue variable indicates that the model signal is selected for the Schedule Index function port. This value is provided by the behavior model.
- The MDL_Signal variable indicates that the model signal value is 1. This selects the first schedule table of the LIN master communication controller and enables the LIN communication on the LinDoorCluster.
- The IO_Signal variable indicates the current value of the Schedule Index function port.



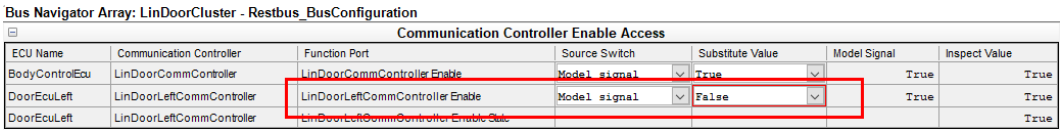
2 Set the value of the TA_Switchvalue variable to Substitute value.

The source of the Schedule Index function port is switched to the substitute value, i.e., to the value of the TA_Replacevalue variable. The IO_Signal variable displays this value as the current function port value.

Because this value is set to 0, no schedule table is selected and the entire LIN communication on the LinDoorCluster is disabled. As a result, the Counter, Time, and Delta Time in both Bus Instrument (RX type) stop.

3 Set the value of the TA_Switchvalue variable to Model signal to enable the LIN communication again.

4 In the Communication Controller Enable Access region of the Bus Instrument (TX status type), set the Substitute Value of the LinDoorLeftCommController Enable function port to False.


Bus Navigator Array: LinDoorCluster - Restbus_BusConfiguration

Communication Controller Enable Access

ECU Name	Communication Controller	Function Port	Source Switch	Substitute Value	Model Signal	Inspect Value
BodyControlEcu	LinDoorCommController	LinDoorCommController Enable	Model signal	True	True	True
DoorEcuLeft	LinDoorLeftCommController	LinDoorLeftCommController Enable	Model signal	False	True	True
DoorEcuLeft	LinDoorLeftCommController	LinDoorLeftCommController Enable Stat...				True

The substitute value of the function port is set to False. However, the displayed Inspect Value indicates that this is not the current function port value.

5 For the LinDoorLeftCommController Enable function port, set the Source Switch to Substitute value.

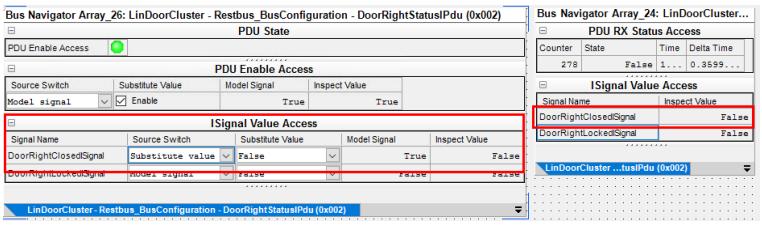
The source of the function port value is switched from the model signal to the substitute value, i.e., the current function port value is False. This disables the communication controller of the DoorEcuLeft and the LIN communication of the ECU stops. Therefore, the DoorLeftStatusIPdu is not transmitted and it cannot be received by the BodyControlEcu. Nevertheless, the transmission of the DoorRightStatusIPdu is not affected.

6 For the LinDoorCommController Enable function port, set the Substitute Value to True.

The communication controller of the DoorEcuLeft is enabled. The ECU transmits the DoorLeftStatusIPdu, and the BodyControlEcu receives the PDU.

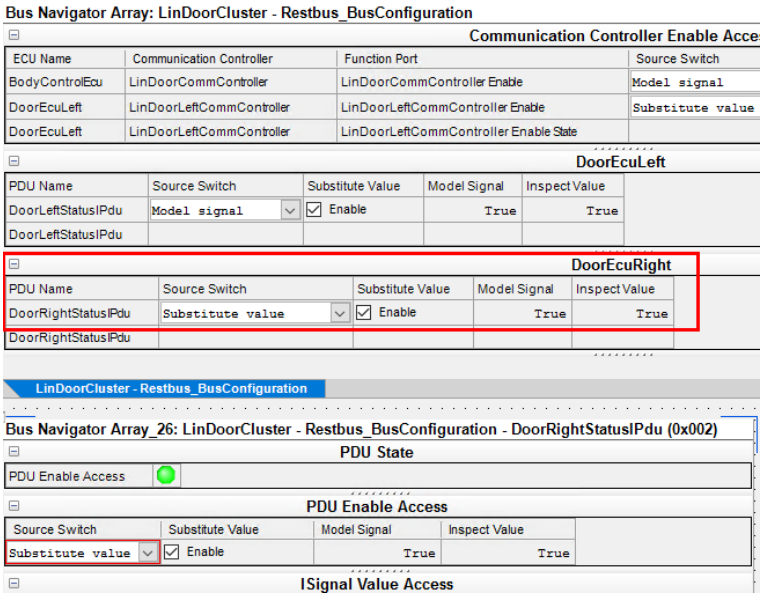
7 In the ISignal Value Access region of the Bus Instrument (TX type), set the Source Switch of the DoorRightClosedISignal value to Substitute value.

The substitute value of the ISignal is transmitted and received by the BodyControlEcu.



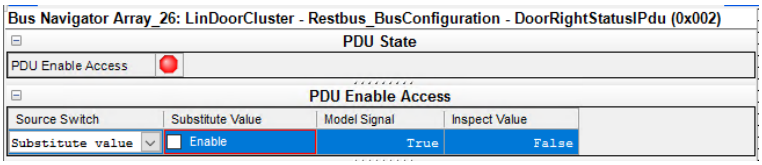
8 In the PDU Enable Access region of the Bus Instrument (TX type), set the Source Switch to Substitute value.

In the DoorEcuRight region of the Bus Instrument (TX status type), the source switch of the DoorRightStatusIPdu changes accordingly because it is the same source switch.



9 Clear the Enable checkbox.

The transmission of the DoorRightStatusIPdu is disabled. In the PDU State region of the Bus Instrument (TX type), the color of the PDU Enable Access LED changes from green to red.



10 On the Home ribbon, click  .

Online calibration stops.

Result

You simulated the LIN communication that you configured with the Bus Manager and accessed the communication via instruments.

What's next

In the next step, you will prepare a layout to simulate, manipulate, and inspect CAN communication.

Step 4: How to Prepare a Layout for Simulating, Manipulating, and Inspecting CAN Communication

Objective

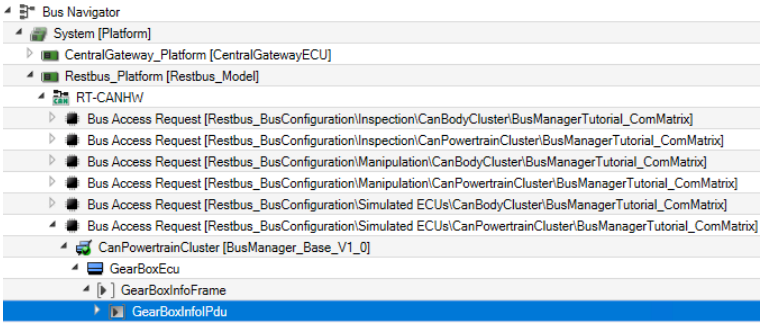
To handle bus communication in a ControlDesk experiment, you can use common and bus-specific instruments which are arranged on one or more layouts.

Method

To prepare a layout for simulating, manipulating, and inspecting CAN communication

1 Select the Bus Navigator.

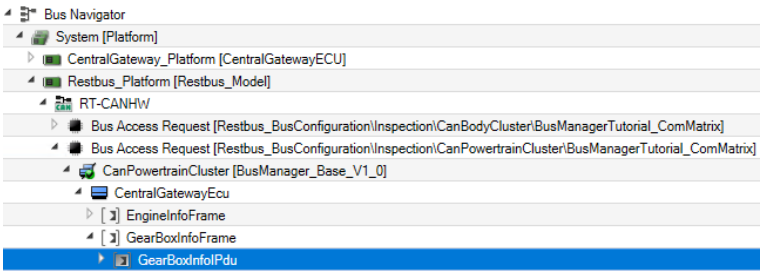
2 Navigate to the following TX IPDU: Restbus_Platform [<xxx>] - RT-CANHW - Bus Access Request [Restbus_BusConfiguration\Simulated ECUs\CanPowertrainCluster\BusManagerTutorial_ComMatrix] - CanPowertrainCluster [BusManager_Base_V1_0] - GearBoxEcu - GearBoxInfoFrame - GearBoxInfoIPdu.



3 Right-click the GearBoxInfoIPdu and select Generate TX Layout from the context menu.

A new layout is generated and a Bus Instrument (TX type) is added to the layout.

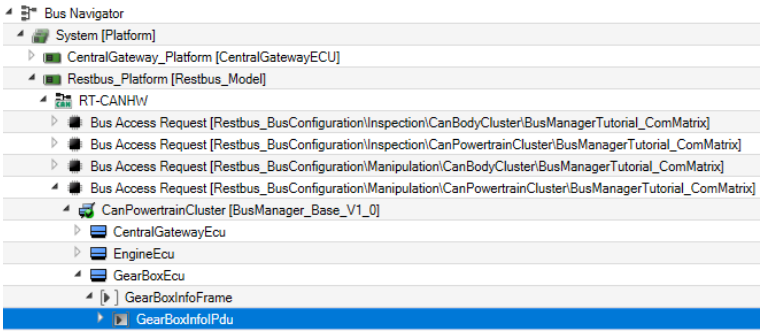
4 Navigate to the following RX IPDU: Restbus_Platform [<xxx>] - RT-CANHW - Bus Access Request [Restbus_BusConfiguration\Inspection\CanPowertrainCluster\BusManagerTutorial_ComMatrix] - CanPowertrainCluster [BusManager_Base_V1_0] - CentralGatewayEcu - GearBoxInfoFrame - GearBoxInfoIPdu.



5 Drag the RX GearBoxInfoIPdu to the layout and select Bus Navigator Inspection Instrument from the list.

A Bus Instrument (Inspection type) is added to the layout.

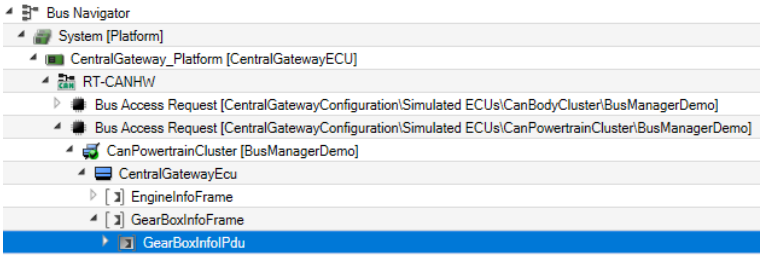
6 Navigate to the following TX IPDU: Restbus_Platform [<xxx>] - RT-CANHW - Bus Access Request [Restbus_BusConfiguration\Manipulation\CanPowertrainCluster\BusManagerTutorial_ComMatrix] - CanPowertrainCluster [BusManager_Base_V1_0] - GearBoxEcu - GearBoxInfoFrame - GearBoxInfoIPdu.



7 Drag the TX GearBoxInfoIPdu to the layout and select Bus Navigator Manipulation Instrument from the list.

A Bus Instrument (Manipulation type) is added to the layout.

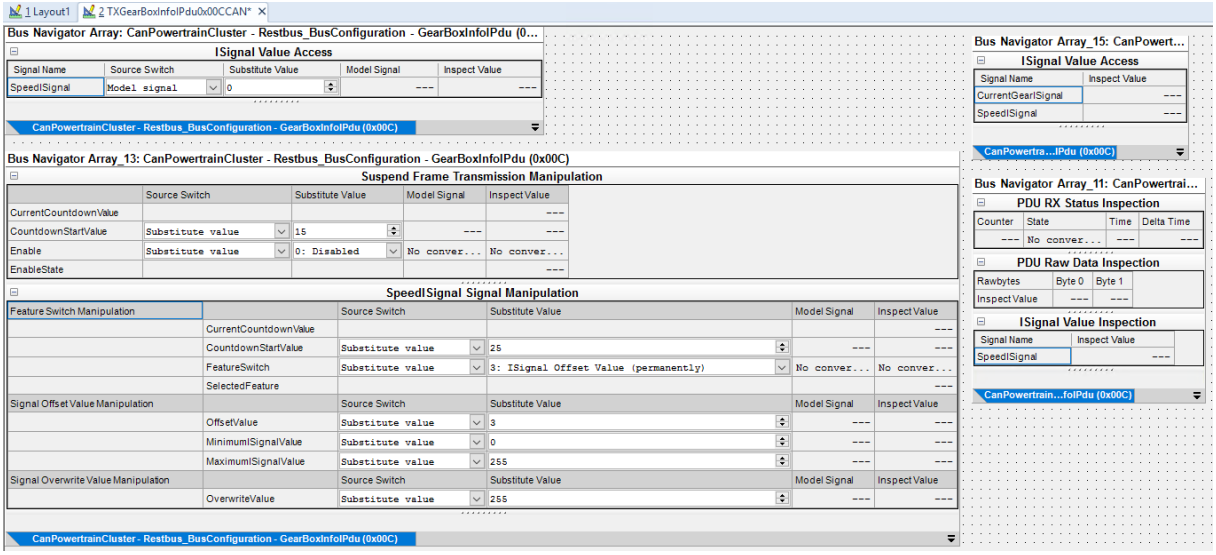
8 Navigate to the following RX IPDU: CentralGateway_Platform [<xxx>] - RT-CANHW - Bus Access Request [CentralGatewayConfiguration\Simulated ECUs\CanPowertrainCluster\BusManagerDemo] - CanPowertrainCluster [BusManagerDemo] - CentralGatewayEcu - GearBoxInfoFrame - GearBoxInfoIPdu.



9 Drag the RX GearBoxInfoIPdu to the layout and select Bus Navigator Rx Instrument from the list.

A Bus Instrument (RX type) is added to the layout.

10 Arrange the instruments as follows:



Result

You prepared a layout for simulating, manipulating, and inspecting CAN communication.

What's next

In the next step, you will simulate and manipulate CAN communication, and inspect the effects on communication.

Step 5: How to Simulate, Manipulate, and Inspect CAN Communication

Objective

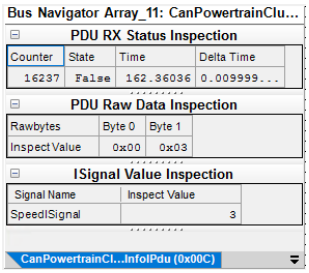
When you start online calibration, you can simulate, manipulate, and inspect the CAN communication that you configured with the Bus Manager. You can use ControlDesk instruments to access the CAN communication.

Method

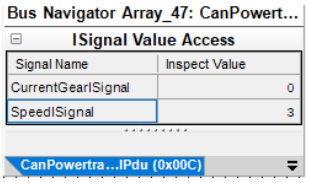
To simulate, manipulate, and inspect CAN communication

1 On the Home ribbon, click  .

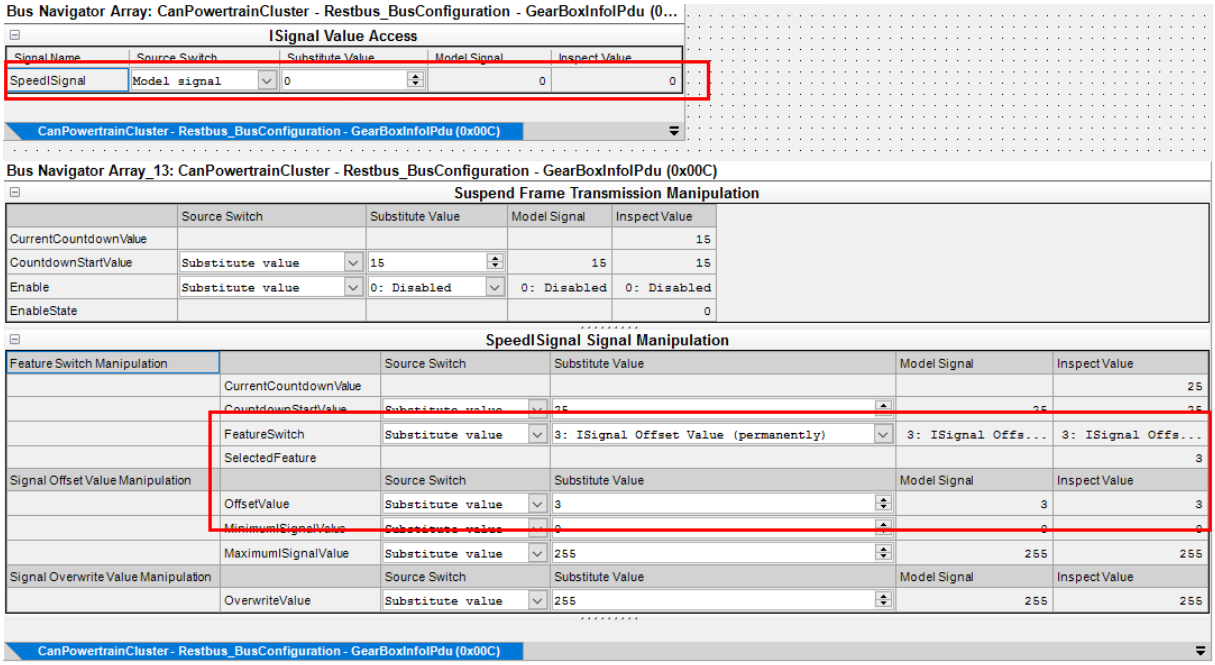
Online calibration starts and the CAN communication is active. The Counter, Time, and Delta Time of the PDU RX Status Inspection region in the Bus Instrument (Inspection type) are running. This indicates that the GearBoxInfoIPdu is transmitted on the bus and can be received by the bus members. The ISignal Value Inspection region indicates that the value of the SpeedISignal is 3.



The recipient of the IPDU is the CentralGatewayECU. The Bus Instrument (RX type) indicates that the ECU receives the SpeedISignal with the inspected value.

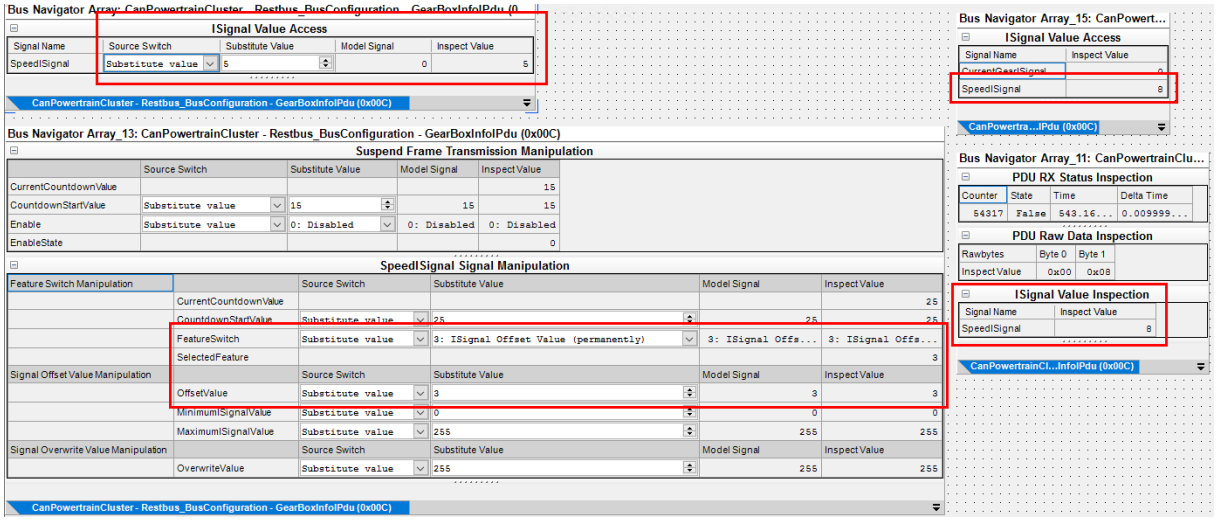


The value of the SpeedISignal is already manipulated: The Bus Instrument (TX type) indicates that the ISignal is simulated with its model signal, which is 0. But before transmission, an offset value of 3 is added to the ISignal value because the ISignal's manipulation feature switch is set to 3: ISignal Offset Value (permanently) by default.



2 In the ISignal Value Access region of the Bus Instrument (TX type), specify the following settings:
   - Substitute Value: 5
   - Source Switch: Substitute value

The ISignal value is simulated with a value of 5. Because this value is manipulated with the offset value, the transmitted and inspected ISignal value is 8. This value is received by the CentralGatewayEcu.



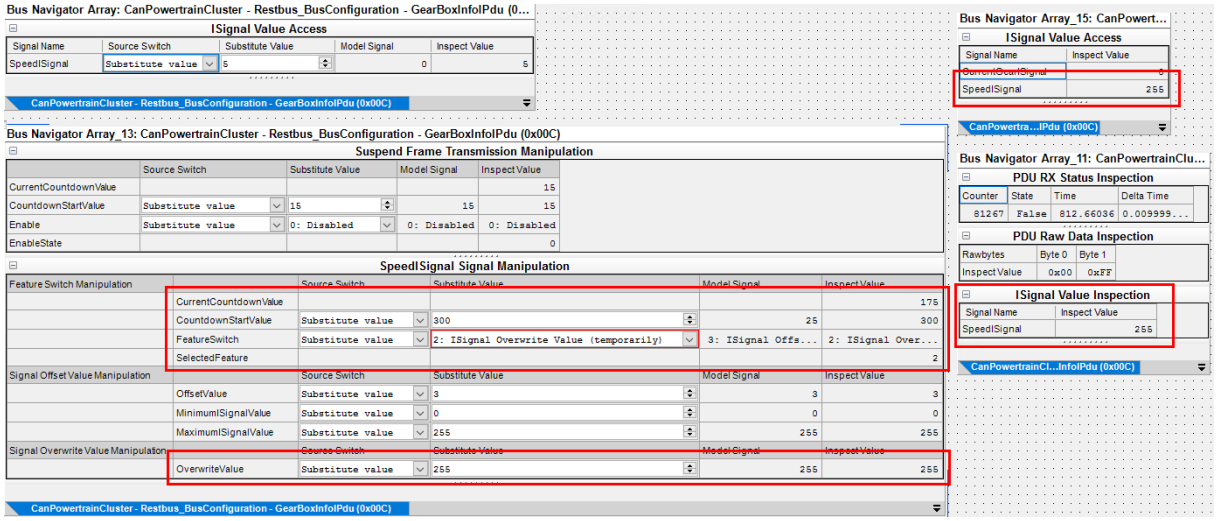
3 In the SpeedISignal Signal Manipulation region of the Bus Instrument (Manipulation type), set the Substitute Value of the FeatureSwitch to 0: None.

The ISignal value is not manipulated and the SpeedISignal is transmitted with its simulated substitute value, i.e., 5.

4 In the SpeedISignal Signal Manipulation region, set the Substitute Value of the CountdownStartValue to 300.

5 Set the Substitute Value of the FeatureSwitch to 2: ISignal Overwrite Value (temporarily).

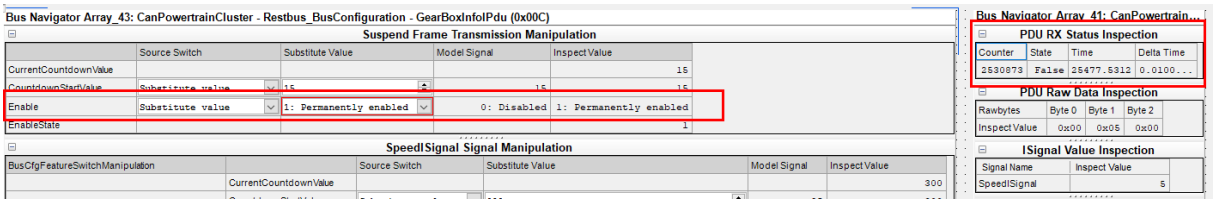
For 300 sampling steps, the SpeedISignal is manipulated by the ISignal Overwrite Value feature, i.e., the simulated ISignal value is overwritten with 255. The Inspect Value of the CurrentCountdownValue indicates the remaining number of sampling steps for manipulating the ISignal.



After the number of sampling steps has elapsed, the FeatureSwitch is automatically set to 0: None and the ISignal is transmitted with its simulated value.

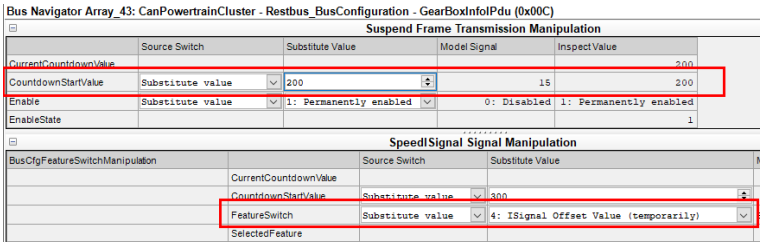
6 In the Suspend Frame Transmission Manipulation region, set the Substitute Value of Enable to 1: Permanently enabled.

The Suspend Frame Transmission feature is permanently enabled, i.e., the GearBoxInfoFrame and its included GearBoxInfoIPdu are not transmitted on the bus. Therefore, the Counter, Time, and Delta Time in the PDU RX Status Inspection region stop.



7 In the Bus Instrument (Manipulation type), specify the following settings:

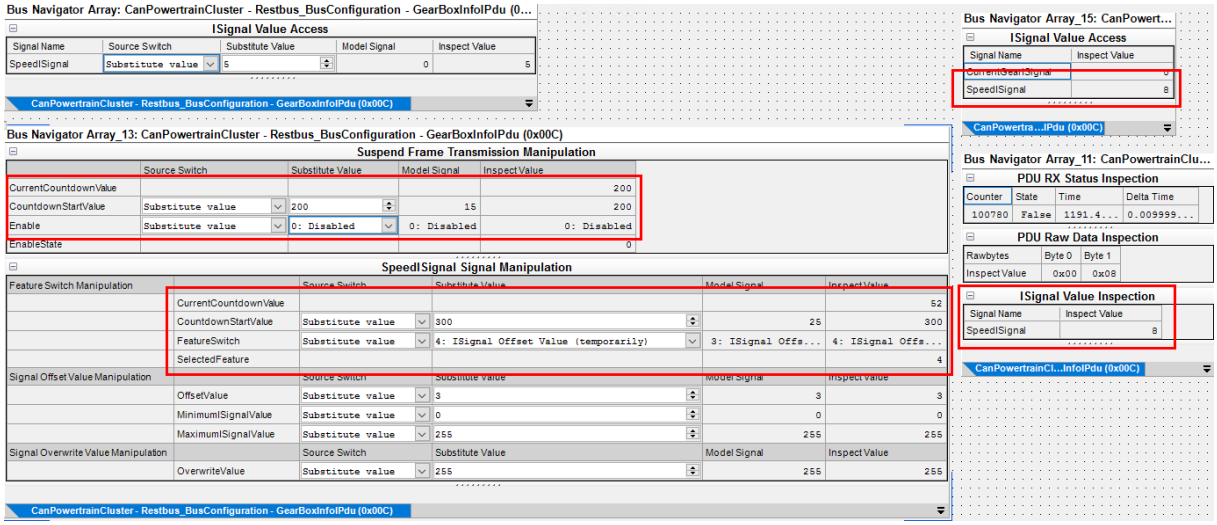
Region	Element	Substitute Value
SpeedISignal Signal Manipulation	FeatureSwitch	4: ISignal Offset Value (temporarily)
Suspend Frame Transmission Manipulation	CountdownStartValue	200



8 In the Suspend Frame Transmission Manipulation region, set the Substitute Value of Enable to 2: Temporarily enabled.

For 200 sampling steps, the transmission of the GearBoxInfoFrame is disabled and the GearBoxInfoIPdu is not transmitted on the bus. After the number of sampling steps has elapsed, Enable is automatically set to 0: Disabled, i.e., the frame manipulation is disabled. The IPDU including its ISignals is transmitted on the bus.

Now, the SpeedISignal is manipulated by the ISignal Offset Value feature for 300 sampling steps, i.e., the transmitted ISignal value is 8.



After the number of sampling steps has elapsed, the FeatureSwitch is automatically set to 0: None and the ISignal is transmitted with its simulated value.

9 On the Home ribbon, click C.
Online calibration stops.

Result

You simulated, manipulated, and inspected the CAN communication that you configured with the Bus Manager and accessed the communication via instruments.

What's next

With this step, you finished the lesson. For a summary of the lesson and for more information, refer to Result of the Experimenting with ControlDesk Lesson on page 197.

Result of the Experimenting with ControlDesk Lesson

Result

In this lesson, you learned the basic principles of experimenting with ControlDesk:

- The Bus Navigator provides a hardware-related access to the bus communication you configured with the Bus Manager.
- To access elements of the bus communication in ControlDesk, you must have configured the access in the ConfigurationDesk application, i.e., you must have added bus configuration features to elements and enabled test automation support for function ports.

- You can use bus-specific instruments and common ControlDesk instruments to access elements of the bus communication, e.g., PDUs.
- There are different types of bus instruments, e.g., to access simulated RX and TX PDUs, manipulate PDUs, or inspect PDUs on the bus. Each bus-specific instrument is tailor-made, i.e., its available elements are derived from the configured bus communication.

Prepared result application

For this lesson, no result application is available because you finished your work in ControlDesk.

Further information

For more information on working with the Bus Navigator, refer to ControlDesk Bus Navigator 🕮.

Where to go from here

You have finished the optional lesson on experimenting with ControlDesk.

- To continue with a lesson that is relevant for your use scenario, refer to Workflows for Using the Bus Manager on page 27.
- For a summary of your working results, refer to Summary on page 199.

Summary

Your Working Results

What you learned

You learned:

- To configure bus communication for simulation, inspection, and manipulation purposes by assigning communication matrix elements to the **Simulated ECUs**, **Inspection**, and **Manipulation** part of a bus configuration.
- To specify the run-time behavior by adding bus configuration features to assigned communication matrix elements, and configuring the bus configuration features.
- To check communication matrices and the configured bus communication for conflicts and to resolve conflicts.
- To work with and without a behavior model, and to generate a Simulink model interface for the configured bus communication.
- To build a real-time application and generate bus simulation containers.
- To experiment with the configured bus communication in ControlDesk.

Where to go from here

You can continue working in different ways:

- For more information on the Bus Manager, refer to:
  - Bus Manager in ConfigurationDesk: ConfigurationDesk Bus Manager Implementation Guide 🕮
  - Bus Manager (stand-alone): Bus Manager (Stand-Alone) Implementation Guide 🕮
- For more information on ConfigurationDesk, refer to:
  - ConfigurationDesk Getting Started 🕮
  - ConfigurationDesk Real-Time Implementation Guide 🕮

ConfigurationDesk Glossary

Introduction

The glossary briefly explains the most important expressions and naming conventions used in the ConfigurationDesk documentation.

Where to go from here

Information in this section

A....202
B....202
C....205
D....208
E....209
F....211
G....212
H....212
I....213
L....214
M....215
N....218
O....218
P....218
R....220
S....221
T....222
U....223

V....224
W....224
X....225

A

Application There are two types of applications in ConfigurationDesk:

- A part of a ConfigurationDesk project: ConfigurationDesk application.
- An application that can be executed on dSPACE real-time hardware: real-time application.

Application process A component of a processing unit application. An application process contains one or more tasks.

Application process component A component of an application process. The following application process components are available in the **Components** subfolder of an application process:

- Behavior models that are assigned to the application process, including their predefined tasks, runnable functions, and events.
- Function blocks that are assigned to the application process.

AutomationDesk A dSPACE software product for creating and managing any kind of automation tasks. Within the dSPACE tool chain, it is mainly used for automating tests on dSPACE hardware.

AUTOSAR system description file An AUTOSAR XML (ARXML) file that describes a system according to AUTOSAR. A system is a combination of a hardware topology, a software architecture, a network communication, and information on the mappings between these elements. The described network communication usually consists of more than one bus system (e.g., CAN, LIN, FlexRay).

B

Basic PDU A general term used in the documentation to address all the PDUs the Bus Manager supports, except for container IPDUs, multiplexed IPDUs, and secured IPDUs. Basic PDUs are represented by the or symbol in

tables and browsers. The Bus Manager provides the same functionalities for all basic PDUs, such as ISignal IPDUs or NMPDUs.

Behavior model A model that contains the control algorithm for a controller (function prototyping system) or the algorithm of the controlled system (hardware-in-the-loop system). It does not contain I/O functionality nor access to the hardware. Behavior models can be modeled, for example, in MATLAB/Simulink by using Simulink Blocksets and Toolboxes from the MathWorks®.

You can add Simulink behavior models to a ConfigurationDesk application. You can also add code container files containing a behavior model such as Functional Mock-up Units, or Simulink implementation containers to a ConfigurationDesk application.

Bidirectional signal port A signal port that is independent of a data direction or current flow. This port is used, for example, to implement bus communication.

BSC file A bus simulation container file that is generated with the Bus Manager and contains the configured bus communication of one application process.

Build Configuration table A pane that lets you create build configuration sets and configure build settings, for example, build options, or the build and download behavior.

Build Log Viewer A pane that displays messages and warnings during the build process.

Build process A process that generates an executable real-time application based on your ConfigurationDesk application that can be run on a SCALEXIO system or MicroAutoBox III system. The build process can be controlled and configured via the Build Log Viewer. If the build process is successfully finished, the build result files (build results) are added to the ConfigurationDesk application.

Build results The files that are created during the build process. Build results are named after the ConfigurationDesk application and the application process from which they originate. You can access the build results in the Project Manager.

Bus access The representation of a run-time communication cluster. By assigning one or more bus access requests to a bus access, you specify which communication clusters form one run-time communication cluster.

In ConfigurationDesk, you can use a bus function block (**CAN**, **LIN**) to implement a bus access. The hardware resource assignment of the bus function block specifies the bus channel that is used for the bus communication.

Bus access request The representation of a request regarding the bus access. There are two sources for bus access requests:

- At least one element of a communication cluster is assigned to the **Simulated ECUs**, **Inspection**, or **Manipulation** part of a bus configuration. The related bus access requests contain the requirements for the bus channels that are to be used for the cluster's bus communication.

- A frame gateway is added to the **Gateways** part of a bus configuration. Each frame gateway provides two bus access requests that are required to specify the bus channels for exchanging bus communication.

Bus access requests are automatically included in BSC files. To build a real-time application, each bus access request must be assigned to a bus access.

Bus Access Requests table A pane that lets you access bus access requests of a ConfigurationDesk application and assign them to bus accesses.

Bus configuration A Bus Manager element that implements bus communication in a ConfigurationDesk application and lets you configure it for simulation, inspection, and/or manipulation purposes. The required bus communication elements must be specified in a communication matrix and assigned to the bus configuration. Additionally, a bus configuration lets you specify gateways for exchanging bus communication between communication clusters. A bus configuration can be accessed via specific tables and its related Bus Configuration function block.

Bus Configuration Function Ports table A pane that lets you access and configure function ports of bus configurations.

Bus Configurations table A pane that lets you access and configure bus configurations of a ConfigurationDesk application.

Bus Inspection Features table A pane that lets you access and configure bus configuration features of a ConfigurationDesk application for inspection purposes.

Bus Manager

- Bus Manager in ConfigurationDesk

  A ConfigurationDesk component that lets you configure bus communication and implement it in real-time applications or generate bus simulation containers.
- Bus Manager (stand-alone)

  A dSPACE software product based on ConfigurationDesk that lets you configure bus communication and generate bus simulation containers.

Bus Manipulation Features table A pane that lets you access and configure bus configuration features of a ConfigurationDesk application for manipulation purposes.

Bus simulation container A container that contains bus communication configured with the Bus Manager. Bus simulation container (BSC) files can be used in the VEOS Player and in ConfigurationDesk. In the VEOS Player, they let you implement the bus communication in an offline simulation application.

In ConfigurationDesk, they let you implement the bus communication in a real-time application independently from the Bus Manager.

Bus Simulation Features table A pane that lets you access and configure bus configuration features of a ConfigurationDesk application for simulation purposes.

Buses Browser A pane that lets you display and manage the communication matrices of a ConfigurationDesk application. For example, you can access communication matrix elements and assign them to bus configurations. This pane is available only if you work with the Bus Manager.

C

Cable harness A bundle of cables that provides the connection between the I/O connectors of the real-time hardware and the external devices, such as the ECUs to be tested. In ConfigurationDesk, it is represented by an external cable harness component.

CAFX file A ConfigurationDesk application fragment file that contains signal chain elements that were exported from a user-defined working view or the Temporary working view of a ConfigurationDesk application. This includes the elements' configuration and the mapping lines between them.

CDL file A ConfigurationDesk application file that contains links to all the documents related to an application.

Channel multiplication A feature that allows you to enhance the max. current or max. voltage of a single hardware channel by combining several channels. ConfigurationDesk uses a user-defined value to calculate the number of hardware channels needed. Depending on the function block type, channel multiplication is provided either for current enhancement (two or more channels are connected in parallel) or for voltage enhancement (two or more channels are connected in series).

Channel request A channel assignment required by a function block. ConfigurationDesk determines the type(s) and number of channels required for a function block according to the assigned channel set, the function block features, the block configuration and the required physical ranges. ConfigurationDesk provides a set of suitable and available hardware resources for each channel request. This set is produced according to the hardware topology added to the active ConfigurationDesk application. You have to assign each channel request to a specific channel of the hardware topology.

Channel set A number of channels of the same channel type located on the same I/O board or I/O unit. Channels in a channel set can be combined, for example, to provide a signal with channel multiplication.

Channel type A term to indicate all the hardware resources (channels) in the hardware system that provide exactly the same characteristics. Examples for

channel type names: **Flexible In 1**, **Digital Out 3**, **Analog In 1**. An I/O board in a hardware system can have channel sets of several channel types. Channel sets of one channel type can be available on different I/O boards.

Cluster Communication cluster.

Common Program Data folder A standard folder for application-specific configuration data that is used by all users.

`%PROGRAMDATA%\dSPACE\<InstallationGUID>\<ProductName>`

or

`%PROGRAMDATA%\dSPACE\<ProductName>\<VersionNumber>`

Communication cluster A communication network of network nodes that are connected to the same physical channels and share the same bus protocol and address range.

Communication matrix A file that defines the communication of a bus network. It can describe the bus communication of one communication cluster or a bus network consisting of different bus systems and clusters. Files of various file formats can be used as a communication matrix: For example, AUTOSAR system description files, DBC files, LDF files, and FIBEX files.

Communication package A package that bundles Data Inport blocks which are connected to Data Outport blocks. Hence, it also bundles the signals that are received by these blocks. If Data Inport blocks are executed within the same task and belong to the same communication package, their data inports are read simultaneously. If Data Outport blocks that are connected to the Data Inport blocks are executed in the same task, their output signals are sent simultaneously in one data package. Thus, communication packages guarantee simultaneous signal updates within a running task (data snapshot).

Configuration port A port that lets you create the signal chain for the bus communication implemented in a Simulink behavior model. The following configuration ports are available:

- The configuration port of a Configuration Port block.
- The **Configuration** port of a **CAN**, **LIN**, or **FlexRay** function block.

To create the signal chain for bus communication, the configuration port of a **Configuration Port** block must be mapped to the **Configuration** port of a **CAN**, **LIN**, or **FlexRay** function block.

Configuration Port block A model port block that is created in ConfigurationDesk during model analysis for each of the following blocks found in the Simulink behavior model:

- **RTICANMM ControllerSetup** block
- **RTILINMM ControllerSetup** block
- **FLEXRAYCONFIG UPDATE** block

Configuration Port blocks are also created for bus simulation containers. A Configuration Port block provides a configuration port that must be mapped

to the Configuration port of a **CAN**, **LIN**, or **FlexRay** function block to create the signal chain for bus communication.

ConfigurationDesk application A part of a ConfigurationDesk project that represents a specific implementation. You can work with only one application at a time, and that application must be activated.

An application can contain:

- Device topology
- Hardware topology
- Model topology
- Communication matrices
- External cable harness
- Build results (after a successful build process has finished)

You can also add folders with application-specific files to an application.

ConfigurationDesk model interface The part of the model interface that is available in ConfigurationDesk. This specific term is used to explicitly distinguish between the model interface in ConfigurationDesk and the model interface in Simulink.

Conflict A result of conflicting configuration settings that is displayed in the Conflicts Viewer. ConfigurationDesk allows flexible configuration without strict constraints. This lets you work more freely, but it can lead to conflicting configuration settings. ConfigurationDesk automatically detects conflicts and provides the **Conflicts Viewer** to display and help resolve them. Before you build a real-time application, you have to resolve at least the most severe conflicts (e.g., errors that abort the build process) to get proper build results.

Conflicts Viewer A pane that displays the configuration conflicts that exist in the active ConfigurationDesk application. You can resolve most of the conflicts directly in the **Conflicts Viewer**.

Container IPDU A term according to AUTOSAR. An IPDU that contains one or more other IPDUs (i.e., contained IPDUs). When a container IPDU is mapped to a frame, all its contained IPDUs are included in that frame as well.

ControlDesk A dSPACE software product for managing, instrumenting and executing experiments for ECU development. ControlDesk also supports calibration, measurement and diagnostics access to ECUs via standardized protocols such as CCP, XCP, and ODX.

CTLGZ file A ZIP file that contains a V-ECU implementation. CTLGZ files are exported by TargetLink or SystemDesk. You can add a V-ECU implementation based on a CTLGZ file to the model topology just like adding a Simulink model based on an SLX file.

Cycle time restriction A value of a runnable function that indicates the sample time the runnable function requires to achieve correct results. The cycle time restriction is indicated by the **Period** property of the runnable function in the Properties Browser.

D

Data inport A port that supplies data from ConfigurationDesk's function outports to the behavior model.

In a multimodel application, data inports also can be used to provide data from a data outport associated to another behavior model (model communication).

Data outport A port that supplies data from behavior model signals to ConfigurationDesk's function inports.

In a multimodel application, data outports also can be used to supply data to a data inport associated to another behavior model (model communication).

DBC file A Data Base Container file that describes CAN or LIN bus systems. Because the DBC file format was primarily developed to describe CAN networks, it does not support definitions of LIN masters and schedules.

Device block A graphical representation of devices from the device topology in the signal chain. It can be mapped to function blocks via device ports.

Device connector A structural element that lets you group device pins in a hierarchy in the External Device Connectors table to represent the structure of the real connector of your external device.

Device pin A representation of a connector pin of your external device. Device ports are assigned to device pins. ConfigurationDesk can use the device pin assignment together with the hardware resource assignment and the device port mapping to calculate the external cable harness.

Device port An element of a device topology that represents the signal of an external device in ConfigurationDesk.

Device port group A structural element of a device topology that can contain device ports and other device port groups.

Device topology A component of a ConfigurationDesk application that represents external devices in ConfigurationDesk. You can create a device topology from scratch or easily extend an existing device topology. You can also merge device topologies to extend one. To edit or create device topologies independently of ConfigurationDesk, you can export and import DTFX and XLSX files.

Documents folder A standard folder for user-specific documents.

`%USERPROFILE%\Documents\dSPACE\<ProductName>\<VersionNumber>`

DSA file A dSPACE archive file that contains a ConfigurationDesk application and all the files belonging to it as one unit. It can later be imported to another ConfigurationDesk project.

dSPACE Help The dSPACE online help that contains all the relevant user documentation for dSPACE products. Via the **F1** key or the **Help** button in the dSPACE software you get context-sensitive help on the currently active context.

dSPACE Log A collection of errors, warnings, information, questions, and advice issued by all dSPACE products and connected systems over more than one session.

DTFX file A device topology export file that contains information on the interface to the external devices, such as the ECUs to be tested. The information includes details of the available device ports, their characteristics, and the assigned pins.

E

ECHX file An external cable harness file that contains the wiring information for the external cable harness. The external cable harness is the connection between the I/O connectors of the real-time hardware and the devices to be tested, for example, ECUs.

ECU Abbreviation of *electronic control unit*.

An embedded computer system that consists of at least one CPU and associated peripherals. An ECU contains communication controllers and communication connectors, and usually communicates with other ECUs of a bus network. An ECU can be member of multiple bus systems and communication clusters.

ECU application An application that is executed on an ECU. In ECU interfacing scenarios, parts of the ECU application can be accessed (e.g., by a real-time application) for development and testing purposes.

ECU function A function of an ECU application that is executed on the ECU. In ECU interfacing scenarios, an ECU function can be accessed by functions that are part of a real-time application, for example.

ECU Interface Manager A dSPACE software product for preparing ECU applications for ECU interfacing. The ECU Interface Manager can generate ECU interface container (EIC) files to be used in ConfigurationDesk.

ECU interfacing A generic term for methods and tools to read and/or write individual ECU functions and variables of an ECU application. In ECU interfacing scenarios, you can access ECU functions and variables for development and testing purposes while the ECU application is executed on the ECU. For example, you can perform ECU interfacing with SCALEXIO systems or MicroAutoBox III systems to access individual ECU functions by a real-time application.

EIC file An ECU interface container file that is generated with the ECU Interface Manager and describes an ECU application that is configured for ECU interfacing. You can import EIC files to ConfigurationDesk to perform ECU interfacing with SCALEXIO systems or MicroAutoBox III systems.

Electrical interface unit A segment of a function block that provides the interface to the external devices and to the real-time hardware (via hardware

resource assignment). Each electrical interface unit of a function block usually needs a channel set to be assigned to it.

Event A component of a ConfigurationDesk application that triggers the execution of a task. The following event types are available:

- Timer event
- I/O event
- Software event

Event port An element of a function block. The event port can be mapped to a runnable function port for modeling an asynchronous task.

Executable application The generic term for real-time applications and offline simulation applications. In ConfigurationDesk, an executable application is always a real-time application since ConfigurationDesk does not support offline simulation applications.

Executable application component A component of an executable application. The following components can be part of an executable application:

- Imported behavior models including predefined tasks, runnable functions, and events. You can assign these behavior models to application processes via drag & drop or by selecting the **Assign Model** command from the context menu of the relevant application process.
- Function blocks added to your ConfigurationDesk application including associated I/O events. Function blocks are assigned to application processes via their model port mapping.

Executable Application table A pane that lets you model executable applications (i.e., real-time applications) and the tasks used in them.

EXPSWCFG file An experiment software configuration file that contains configuration data for automotive fieldbus communication. It is created during the build process and contains the data in XML format.

External cable harness A component of a ConfigurationDesk application that contains the wiring information for the external cable harness (also known as cable harness). It contains only the logical connections and no additional information such as cable length, cable diameters, dimensions or the arrangement of connection points, etc. It can be calculated by ConfigurationDesk or imported from a file so that you can use an existing cable harness and do not have to build a new one. The wiring information can be exported to an ECHX file or XLSX file.

External device A device that is connected to the dSPACE hardware, such as an ECU or external load. The external device topology is the basis for using external devices in the signal chain of a ConfigurationDesk application.

External Device Browser A pane that lets you display and manage the device topology of your active ConfigurationDesk application.

External Device Configuration table A pane that lets you access and configure the most important properties of device topology elements via table.

External Device Connectors table A pane that lets you specify the representation of the physical connectors of your external device including the device pin assignment.

F

FIBEX file An XML file according the ASAM MCD-2 NET standard (also known as Field Bus Exchange Format) defined by ASAM. The file can describe more than one bus system (e.g., CAN, LIN, FlexRay). It is used for data exchange between different tools that work with message-oriented bus communication.

Find Results Viewer A pane that displays the results of searches you performed via the Find command.

FMU file A Functional Mock-up Unit file that describes and implements the functionality of a model. It is an archive file with the file name extension FMU. The FMU file contains:

- The functionality defined as a set of C functions provided either in source or in binary form.
- The model description file (`modelDescription.xml`) with the description of the interface data.
- Additional resources needed for simulation.

You can add an FMU file to the model topology just like adding a Simulink model based on an SLX file.

Frame A piece of information of a bus communication. It contains an arbitrary number of non-overlapping PDUs and the data length code (DLC). CAN frames and LIN frames can contain only one PDU. To exchange a frame via bus channels, a frame triggering is needed.

Frame triggering An instance of a frame that is exchanged via a bus channel. It includes transmission information of the frame (e.g., timings, ID, sender, receiver). The requirements regarding the frame triggerings depend on the bus system (CAN, LIN, FlexRay).

Function block A graphical representation in the signal chain that is instantiated from a function block type. A function block provides the I/O functionality and the connection to the real-time hardware. It serves as a container for functions, for electrical interface units and their logical signals. The function block's ports (function ports and/or signal ports), provide the interfaces to the neighboring blocks in the signal chain.

Function block type A software plug-in that provides a specific I/O functionality. Every function block type has unique features which are different from other function block types.

To use a function block type in your ConfigurationDesk application, you have to create an instance of it. This instance is called a function block. Instances of function block types can be used multiple times in a ConfigurationDesk

application. The types and their instantiated function blocks are displayed in the function library of the Function Browser.

Function Browser A pane that displays the function library in a hierarchical tree structure. Function block types are grouped in function classes. Instantiated function blocks are added below the corresponding function block type.

Function inport A function port that inputs the values from the behavior model to the function block to be processed by the function.

Function library A collection of function block types that allows access to the I/O functionality in ConfigurationDesk. The I/O functionality is based on function block types. The function library provides a structured tree view on the available function block types. It is displayed in the Function Browser.

Function outport A function port that outputs the value of a function to be used in the behavior model.

Function port An element of a function block that provides the interface to the behavior model via model port blocks.

Functional Mock-up Unit An archive file that describes and implements the functionality of a model based on the Functional Mock-up Interface (FMI) standard.

G

Global working view The default working view that always contains all signal chain elements.

H

Hardware resource A hardware element (normally a channel on an I/O board or I/O unit) which is required to execute a function block. A hardware resource can be localized unambiguously in a hardware system. Every hardware resource has specific characteristics. A function block therefore needs a hardware resource that matches the requirements of its functionality. This means that not every function block can be executed on every hardware resource. There could be limitations on a function block's features and/or the physical ranges.

Hardware resource assignment An action that assigns the electrical interface unit of a function block to one or more hardware resources. Function blocks can be assigned to any hardware resource which is suitable for

the functionality and available in the hardware topology of your ConfigurationDesk application.

Hardware Resource Browser A pane that lets you display and manage all the hardware components of the hardware topology that is contained in your active ConfigurationDesk application in a hierarchical structure.

Hardware topology A component of a ConfigurationDesk application that contains information on a specific hardware system which can be used with ConfigurationDesk. It provides information on the components of the system, such as channel type and slot numbers. It can be scanned automatically from a registered platform, created in ConfigurationDesk's Hardware Resource Browser from scratch, or imported from an HTFX file.

HTFX file A file containing the hardware topology after an explicit export. It provides information on the components of the system and also on the channel properties, such as board and channel types and slot numbers.

I/O event An asynchronous event triggered by I/O functions. You can use I/O events to trigger tasks in your application process asynchronously. You can assign the events to the tasks via drag & drop, via the **Properties Browser** if you have selected a task, or via the **Assign Event** command from the context menu of the relevant task.

Interface model A temporary Simulink model that contains blocks from the Model Interface Blockset. ConfigurationDesk initiates the creation of an interface model in Simulink. You can copy the blocks with their identities from the interface model and paste them into an existing Simulink behavior model.

Interpreter A pane that lets you run Python scripts and execute line-based commands.

Inverse model port block A model port block that has the same configuration (same name, same port groups, and port names) but the inverse data direction as the original model port block from which it was created.

IOCNET Abbreviation of I/O carrier network.

A dSPACE proprietary protocol for internal communication in a SCALEXIO system between the real-time processors and I/O units. The IOCNET lets you connect more than 100 I/O nodes and place the parts of your SCALEXIO system long distances apart.

IPDU Abbreviation of interaction layer protocol data unit.

A term according to AUTOSAR. An IPDU contains the communication data that is routed from the interaction layer to a lower communication layer and vice

versa. An IPDU can be implemented, for example, as an ISignal IPDU, multiplexed IPDU, or container IPDU.

ISignal A term according to AUTOSAR. A signal of the interaction layer that contains communication data as a coded signal value. To transmit the communication data on a bus, ISignals are instantiated and included in ISignal IPDUs.

ISignal IPDU A term according to AUTOSAR. An IPDU whose communication data is arranged in ISignals. ISignal IPDUs allow the exchange of ISignals between different network nodes.

L

LDF file A LIN description file that describes networks of the LIN bus system according to the LIN standard.

LIN master A member of a LIN communication cluster that is responsible for the timing of LIN bus communication. A LIN master provides one LIN master task and one LIN slave task. The LIN master task transmits frame headers on the bus, and provides LIN schedule tables and LIN collision resolver tables. The LIN slave task transmits frame responses on the bus. A LIN cluster must contain exactly one LIN master.

LIN schedule table A table defined for a LIN master that contains the transmission sequence of frame headers on a LIN bus. For each LIN master, several LIN schedule tables can be defined.

LIN slave A member of a LIN communication cluster that provides only a LIN slave task. The LIN slave task transmits frame responses on the bus when they are triggered by a frame header. The frame header is sent by a LIN master. A LIN cluster can contain several LIN slaves.

Local Program Data folder A standard folder for application-specific configuration data that is used by the current, non-roaming user.

`%USERPROFILE%\AppData\Local\dSPACE\<InstallationGUID>\<ProductName>`

Logical signal An element of a function block that combines all the signal ports which belong together to provide the functionality of the signal. Each logical signal causes one or more channel requests. Channel requests are available after you have assigned a channel set to the logical signal.

Logical signal chain A term that describes the logical path of a signal between an external device and the behavior model. The main elements of the logical signal chain are represented by different graphical blocks (device blocks, function blocks and model port blocks). Every block has ports to provide the mapping to neighboring blocks.

In the documentation, usually the short form 'signal chain' is used instead.

M

MAP file A file that maps symbolic names to physical addresses.

Mapping line A graphical representation of a connection between two ports in the signal chain. You can draw mapping lines in a working view.

MCD file A model communication description file that is used to implement a multimodel application. It lets you add several behavior models that were separated from an overall model to the model topology.

The MCD file contains information on the separated models and information on the connections between them. The file is generated with the Model Separation Setup Block in MATLAB/Simulink. The block resides in the Model Interface Blockset (dsmpblib) from dSPACE.

MDL file A Simulink model file that contains the behavior model. You can add an MDL file to your ConfigurationDesk application.

As of MATLAB® R2012a, the file name extension for the Simulink model file has been changed from MDL to SLX by The MathWorks®.

Message Viewer A pane that displays a history of all error and warning messages that occur during work with ConfigurationDesk.

Model analysis A process that analyzes the model to determine the interface of a behavior model. You can select one of the following commands:

- Analyze Simulink Model (Model Interface Only)

  Analyzes the interface of a behavior model. The model topology of your active ConfigurationDesk application is updated with the properties of the analyzed behavior model.
- Analyze Simulink Model (Including Task Information)

  Analyzes the model interface and the elements of the behavior model that are relevant for the task configuration. The task configuration in ConfigurationDesk is then updated accordingly.

Model Browser A pane that lets you display and access the model topology of an active ConfigurationDesk application. The **Model Browser** provides access to all the model port blocks available in the behavior models which are linked to a ConfigurationDesk application. The model elements are displayed in a hierarchy, starting with the model roots. Below the model root, all the subsystems containing model port blocks are displayed as well as the associated model port blocks.

Model communication The exchange of signal data between the models within a multimodel application. To set up model communication, you must use a mapping line to connect a data outport (sending model) to a data inport

(receiving model). The best way to set up model communication is using the Model Communication Browser.

Model Communication Browser A pane that lets you open and browse working views like the **Signal Chain Browser**, but shows only the Data Outport and Data Inport blocks and the mapping lines between them.

Model Communication Package table A pane that lets you create and configure model communication packages which are used for model communication in multimodel applications.

Model implementation An implementation of a behavior model. It can consist of source code files, precompiled objects or libraries, variable description files and a description of the model's interface. Specific model implementation types are, for example, model implementation containers, such as Functional Mock-up Units or Simulink implementation containers.

Model implementation container A file archive that contains a model implementation. Examples are FMUs, SIC files, and VECU files.

Model interface An interface that connects ConfigurationDesk with a behavior model. This interface is part of the signal chain and is implemented via model port blocks. The model port blocks in ConfigurationDesk can provide the interface to:

- Model port blocks (from the Model Interface Package for Simulink) in a Simulink behavior model. In this case, the model interface is also called ConfigurationDesk model interface to distinguish it from the Simulink model interface available in the Simulink behavior model.
- Different types of model implementations based on code container files, e.g., Simulink implementation containers, Functional Mock-up Units, and V-ECU implementations.

Model Interface Package for Simulink A dSPACE software product that lets you specify the interface of a behavior model that you can directly use in ConfigurationDesk. You can also create a code container file (SIC file) that contains the model code of a Simulink behavior model. The SIC file can be used in ConfigurationDesk and VEOS Player.

Model port An element of a model port block. Model ports provide the interface to the function ports and to other model ports (in multimodel applications).

These are the types of model ports:

- Data inport
- Data outport
- Runnable function port
- Configuration port

Model port block A graphical representation of the ConfigurationDesk model interface in the signal chain. Model port blocks contain model ports that can be mapped to function blocks to provide the data flow between the function blocks in ConfigurationDesk and the behavior model. The model ports can also be mapped to the model ports of other model port blocks with

data inports or data outports to set up model communication. Model port blocks are available in different types and can provide different port types:

- Data port blocks with data inports and data outports
- Runnable Function blocks with runnable function ports
- Configuration Port blocks with configuration ports. Configuration Port blocks are created during model analysis for each of the following blocks found in the Simulink behavior model:
  - RTICANMM ControllerSetup block
  - RTILINMM ControllerSetup block
  - FLEXRAYCONFIG UPDATE block

  Configuration Port blocks are also created for bus simulation containers.

Model Separation Setup Block A block that is contained in the Model Interface Package for Simulink. It is used to separate individual models from an overall model in MATLAB/Simulink. Additionally, model separation generates a model communication description file (MCD file) that contains information on the separated models and their connections. You can use this MCD file in ConfigurationDesk.

Model topology A component of a ConfigurationDesk application that contains information on the subsystems and the associated model port blocks of all the behavior models that have been added to a ConfigurationDesk application.

Model-Function Mapping Browser A pane that lets you create and update signal chains for Simulink behavior models. It directly connects them to I/O functionality in ConfigurationDesk.

MTFX file A file containing a model topology when explicitly exported. The file contains information on the interface to the behavior model, such as the implemented model port blocks including their subsystems and where they are used in the model.

Multicore real-time application A real-time application that is executed on several cores of one PU of the real-time hardware.

Multimodel application A real-time application that executes several behavior models in parallel on dSPACE real-time hardware (SCALEXIO or MicroAutoBox III).

Multiplexed IPDU A term according to AUTOSAR. An IPDU that consists of one dynamic part, a selector field, and one optional static part. Multiplexing is used to transport varying ISignal IPDUs via the same bytes of a multiplexed IPDU.

- The dynamic part is one ISignal IPDU that is selected for transmission at run time. Several ISignal IPDUs can be specified as dynamic part alternatives. One of these alternatives is selected for transmission.

- The selector field value indicates which ISignal IPDU is transmitted in the dynamic part during run time. For each selector field value, there is one corresponding ISignal IPDU of the dynamic part alternatives. The selector field value is evaluated by the receiver of the multiplexed IPDU.
- The static part is one ISignal IPDU that is always transmitted.

Multi-PU application Abbreviation of multi-processing-unit application. A multi-PU application is a real-time application that is partitioned into several processing unit applications. Each processing unit application is executed on a separate PU of the real-time hardware. The processing units are connected via IOCNET and can be accessed from the same host PC.

N

Navigation bar An element of ConfigurationDesk's user interface that lets you switch between view sets.

Network node A term that describes the bus communication of an ECU for only one communication cluster.

O

Offline simulation A purely PC-based simulation scenario without a connection to a physical system, i.e., neither simulator hardware nor ECU hardware prototypes are needed. Offline simulations are independent from real time and can run on VEOS.

Offline simulation application An application that runs on VEOS to perform offline simulation. An offline simulation application can be built with the VEOS Player and the resulting OSA file can be downloaded to VEOS.

OSA file An offline simulation application file that is built with the VEOS Player and can be downloaded to VEOS to perform offline simulation.

P

Parent port A port that you can use to map multiple function ports and model ports. All child ports with the same name are mapped. ConfigurationDesk enforces the mapping rules and allows only mapping lines which agree with them.

PDU Abbreviation of protocol data unit.

A term according to AUTOSAR. A PDU transports data (e.g., control information or communication data) via one or multiple network layers according to the AUTOSAR layered architecture. Depending on the involved layers and the function of a PDU, various PDU types can be distinguished, e.g., ISignal IPDUs, multiplexed IPDUs, and NMPDUs.

Physical signal chain A term that describes the electrical wiring of external devices (ECU and loads) to the I/O boards of the real-time hardware. The physical signal chain includes the external cable harness, the pinouts of the connectors and the internal cable harness.

Pins and External Wiring table A pane that lets you access the external wiring information

Platform A dSPACE real-time hardware system that can be registered and displayed in the Platform Manager.

Platform Manager A pane that lets you handle registered hardware platforms. You can download, start, and stop real-time applications via the Platform Manager. You can also update the firmware of your SCALEXIO system or MicroAutoBox III system.

Preconfigured application process An application process that was created via the Create preconfigured application process command. If you use the command, ConfigurationDesk creates new tasks for each runnable function provided by the model which is not assigned to a predefined task. ConfigurationDesk assigns the corresponding runnable function and (for periodic tasks) timer events to the new tasks. The tasks are preconfigured (e.g., DAQ raster name, period).

Processing Resource Assignment table A pane that lets you configure and inspect the processing resources in an executable application. This table is useful especially for multi-processing-unit applications.

Processing unit application A component of an executable application. A processing unit application contains one or more application processes.

Project A container for ConfigurationDesk applications and all project-specific documents. You must define a project or open an existing one to work with ConfigurationDesk. Projects are stored in a project root folder.

Project Manager A pane that provides access to ConfigurationDesk projects and applications and all the files they contain.

Project root folder A folder on your file system to which ConfigurationDesk saves all project-relevant data, such as the applications and documents of a project. Several projects can use the same project root folder. ConfigurationDesk uses the Documents folder as the default project root

folder. You can specify further project root folders. Each can be made the default project root folder.

Properties Browser A pane that lets you access the properties of selected elements.

PU Abbreviation of processing unit.

A hardware assembly that consists of a motherboard or a dSPACE processor board, possibly additional interface hardware for connecting I/O boards, and an enclosure, i.e., a SCALEXIO Real-Time PC.

R

Real-time application An application that can be executed in real time on dSPACE real-time hardware. The real-time application is the result of a build process. It can be downloaded to real-time hardware via an RTA file. There are different types of real-time applications:

- Single-core real-time application.
- Multicore real-time application.
- Multi-PU application.

Restbus simulation A simulation method to test real ECUs by connecting them to a simulator that simulates the other ECUs in the communication clusters.

RTA file A real-time application file. An RTA file is an executable object file for processor boards. It is created during the build process. After the build process it can be downloaded to the real-time hardware.

Runnable function A function that is called by a task to compute results. A model implementation provides a runnable function for its base rate task. This runnable function can be executed in a task that is triggered by a timer event. In addition, a Simulink behavior model provides a runnable function for each **Hardware-Triggered Runnable Function** block contained in the Simulink behavior model. This runnable function is suitable for being executed in an asynchronous task.

Runnable Function block A type of model port block. A **Runnable Function** block provides a runnable function port that can be mapped to an event port of a function block for modeling an asynchronous task.

Runnable function port An element of a Runnable Function block. The runnable function port can be mapped to an event port of a function block for modeling an asynchronous task.

RX Communication data that is received by a bus member.

S

SCALEXIO system A dSPACE hardware-in-the-loop (HIL) system consisting of one or more real-time industry PCs (PUs), I/O boards, and I/O units. They communicate with each other via the IOCNET. The system simulates the environment to test an ECU. It provides the sensor signals for the ECU, measures the signals of the ECU, and provides the power (battery voltage) for the ECU and a bus interface for restbus simulation.

SDF file A system description file that contains information on the CPU name(s), the corresponding object file(s) to be downloaded and the corresponding variable description file(s). It is created during the build process.

Secured IPDU A term according to AUTOSAR. An IPDU that secures the payload of another PDU (i.e., authentic IPDU) for secure onboard communication (SecOC). The secured IPDU contains the authentication information that is used to secure the authentic IPDU's payload. If the secured IPDU is configured as a cryptographic IPDU, the secured IPDU and the authentic IPDU are mapped to different frames. If the secured IPDU is not configured as a cryptographic IPDU, the authentic IPDU is directly included in the secured IPDU.

SIC file A Simulink implementation container file that contains the model code of a Simulink behavior model. The SIC file can be used in ConfigurationDesk and in VEOS Player.

Signal chain A term used in the documentation as a short form for logical signal chain. Do not confuse it with the physical signal chain.

Signal Chain Browser A pane that lets you open and browse working views such as the Global working view or user-defined working views.

Signal inport A signal port that represents an electrical connection point of a function block which provides signal measurement (= input) functionality.

Signal outport A signal port that represents an electrical connection point of a function block which provides signal generation (= output) functionality.

Signal port An element of a function block that provides the interface to external devices (e.g., ECUs) via device blocks. It represents an electrical connection point of a function block.

Signal reference port A signal port that represents a connection point for the reference potential of inports, outports and bidirectional ports. For example: With differential signals, this is a reference signal, with single-ended signals, it is the ground signal (GND).

Simulink implementation container A container that contains the model code of a Simulink behavior model. A Simulink implementation container is generated from a Simulink behavior model by using the Model Interface Package

for Simulink. The file name extension of a Simulink implementation container is SIC.

Simulink model interface The part of the model interface that is available in the connected Simulink behavior model.

Single-core real-time application An executable application that is executed on only one core of the real-time hardware.

Single-PU system Abbreviation of single-processing-unit system.

A system consisting of exactly one PU and the directly connected I/O units and I/O routers.

SLX file A Simulink model file that contains the behavior model. You can add an SLX file to your ConfigurationDesk application.

As of MATLAB® R2012a, the file name extension for the Simulink model file has been changed from MDL to SLX by The MathWorks®.

Software event An event that is activated from within a task to trigger another subordinate task. Consider the following example: A multi-tasking Simulink behavior model has a base rate task with a sample time = 1 ms and a periodic task with a sample time = 4 ms. In this case, the periodic task is triggered on every fourth execution of the base rate task via a software event. Software events are available in ConfigurationDesk after model analysis.

Source Code Editor A Python editor that lets you open and edit Python scripts that you open from or create in a ConfigurationDesk project in a window in the working area. You cannot run a Python script in a **Source Code Editor** window. To run a Python script you can use the **Run Script** command in the Interpreter or on the **Automation** ribbon or the **Run** context menu command in the Project Manager.

Structured data port A hierarchically structured port of a Data Inport block or a Data Outport block. Each structured data port consists of more structured and/or unstructured data ports. The structured data ports can consist of signals with different data types (single, double, int8, int16, int32, int64, uint8, uint16, uint32, uint64, Boolean).

SystemDesk A dSPACE software product for development of distributed automotive electrics/electronics systems according to the AUTOSAR approach.

SystemDesk is able to provide a V-ECU implementation container (as a VECU file) to be used in ConfigurationDesk.

T

Table A type of pane that offers access to a specific subset of elements and properties of the active ConfigurationDesk application in rows and columns.

TargetLink A dSPACE software product for production code generation. It lets you generate highly efficient C code for microcontrollers in electronic control

units (ECUs). It also helps you implement control systems that have been modeled graphically in a Simulink/Stateflow diagram on a production ECU.

TargetLink is able to provide a V-ECU implementation container (as a VECU file) or a Simulink implementation container (SIC file) to be used in ConfigurationDesk.

Task A piece of code whose execution is controlled by a real-time operating system (RTOS). A task is usually triggered by an event, and executes one or more runnable functions. In a ConfigurationDesk application, there are predefined tasks that are provided by executable application components. In addition, you can create user-defined tasks that are triggered, for example, by I/O events. Regardless of the type of task, you can configure the tasks by specifying the priority, the DAQ raster name, etc.

Task Configuration table A pane that lets you configure the tasks of an executable application.

Temporary working view A working view that can be used for drafting a signal chain segment, like a notepad.

Timer event A periodic event with a sample rate and an optional offset.

Topology A hierarchy that serves as a repository for creating a signal chain. All the elements of a topology can be used in the signal chain, but not every element needs to be used. You can export each topology from and import it to a ConfigurationDesk application. Therefore a topology can be used in several applications.

A ConfigurationDesk application can contain the following topologies:

- Device topology
- Hardware topology
- Model topology

TRC file A variable description file that contains all variables (signals and parameters) which can be accessed via the experiment software. It is created during the build process.

TX Communication data that is transmitted by a bus member.

U

User function An external function or program that is added to the Automation – User Functions ribbon group for quick and easy access during work with ConfigurationDesk.

V

VECU file A ZIP file that contains a V-ECU implementation. A VECU file can contain data packages for different platforms. VECU files are exported by TargetLink or SystemDesk. You can add a V-ECU implementation based on a VECU file to the model topology in the same way as adding a Simulink model based on an SLX file.

VEOS The dSPACE software product for performing offline simulation. VEOS is a PC-based simulation platform which allows offline simulation independently from real time.

VEOS Player A software running on the host PC for building offline simulation applications. Offline simulation applications can be downloaded to VEOS to perform offline simulation. ConfigurationDesk lets you generate a bus simulation container (BSC file) via the Bus Manager. You can then import the BSC file into the VEOS Player.

View set A specific arrangement of some of ConfigurationDesk's panes. You can switch between view sets by using the navigation bar. ConfigurationDesk provides preconfigured view sets for specific purposes. You can customize existing view sets and create user-defined view sets.

VSET file A file that contains all view sets and their settings from the current ConfigurationDesk installation. A VSET file can be exported and imported via the **View Sets** page of the **Customize** dialog.

W

Working area The central area of ConfigurationDesk's user interface.

Working view A view of the signal chain elements (blocks, ports, mappings, etc.) used in the active ConfigurationDesk application. A working view can be opened in the Signal Chain Browser or the Model Communication Browser. ConfigurationDesk provides two default working views: The Global working view and the Temporary working view. In the Working View Manager, you can also create user-defined working views that let you focus on specific signal chain segments according to your requirements.

Working View Manager A pane that lets you manage the working views of the active ConfigurationDesk application. You can use the **Working View Manager** for creating, renaming, and deleting working views, and also to open a working view in the Signal Chain Browser or the Model Communication Browser.

X

XLSX file A Microsoft Excel™ file format that is used for the following purposes:

- Creating or configuring a device topology outside of ConfigurationDesk.
- Exporting the wiring information for the external cable harness.
- Exporting the configuration data of the currently active ConfigurationDesk application for documentation purposes.

B

behavior model
  generating Simulink model interface 155
  introduction 18
  mapping ports 97
  replacing 155
  synchronizing model interfaces 155
  working without 145
bus configuration features
  for inspection 125
  for manipulation 133
  for simulation 75
  introduction 17
bus configurations 16
Bus Manager variants 12
bus simulation containers
  generating 109
  workflows 28

C

Common Program Data folder 8, 206
communication matrix
  conflicts 68
  in the ConfigurationDesk application 66
  in this tutorial 19
  introduction 15
conflicts
  communication matrix 68
ControlDesk 175

D

Documents folder 8, 208

E

example
  of this tutorial 19

F

feature switch 139

I

inspecting bus communication
  configuring 125
  in this tutorial 21
  use scenario 12

L

lessons
  overview 25
Local Program Data folder 8, 214

M

manipulating bus communication
  configuring 133
  feature switch 139
  in this tutorial 21
  use scenario 12

O

overview
  lessons 25
  workflows 24

R

real-time application
  building 115
  workflows 42

S

simulating bus communication
  configuring 75
  in this tutorial 21
  use scenario 12

T

tutorial
  example 19
  licenses and tools 25
  project and applications 23
  workflows and lessons 22

W

workflows
  overview 24