

ConfigurationDesk

# Real-Time Implementation Guide

For ConfigurationDesk 6.7

Release 2021-A – May 2021

## How to Contact dSPACE

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

## 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](mailto: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.

© 2008 - 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 Guide	19
Safety Precautions	23
General Warning .....	23
Safety Precautions for Configuration and Mapping .....	24
Introduction to ConfigurationDesk	27
Features of ConfigurationDesk.....	28
Overview of Main ConfigurationDesk Features.....	28
Development of ConfigurationDesk's Key Features.....	29
Compatibility Information.....	36
Compatibility of ConfigurationDesk 6.7.....	36
Required Licenses.....	39
Overview of Licenses.....	40
Details on Function Block Licenses.....	42
How to Show Accessible and Used Licenses.....	43
Details on SCALEXIO FIU Licenses.....	44
Notes on Using Floating Network Licenses.....	44
Migrating From Prior Versions of ConfigurationDesk.....	46
Basics on Migrating From Prior Versions of ConfigurationDesk.....	46
Starting ConfigurationDesk.....	48
Notes for Starting ConfigurationDesk.....	48
User Interface of ConfigurationDesk.....	50
Overview of the User Interface of ConfigurationDesk.....	50
Basics on Ribbons.....	54
Customizing View Sets.....	57
Using Keyboard Shortcuts.....	63
How to Move Panes to Different Positions.....	64
How to Customize the Quick Access Toolbar.....	65

## Typical Workflows for Beginners 69

Creating a Real-Time Application: From ECU to Model .....	69
Creating a Real-Time Application: Starting with a Simulink Behavior Model.....	77

## Managing ConfigurationDesk Projects and Applications 85

Introducing ConfigurationDesk Projects Applications.....	86
Handling ConfigurationDesk Projects and Applications.....	86
Managing ConfigurationDesk Projects.....	92
How to Create a Project.....	92
How to Open a Project.....	93
How to Specify a Project Root Folder.....	96
How to Back up and Transfer a Project.....	98
Managing ConfigurationDesk Applications.....	100
How to Add a ConfigurationDesk Application to a Project.....	100
How to Activate a ConfigurationDesk Application.....	101
How to Export and Import a ConfigurationDesk Application .....	102
How to Create a ControlDesk Experiment from a ConfigurationDesk Application.....	103
Managing Components of ConfigurationDesk Applications.....	105
Basics on Components of ConfigurationDesk Applications.....	105
How to Import a Device Topology.....	110
How to Import a Model Topology.....	112
How to Import a Hardware Topology.....	115
How to Import an External Cable Harness.....	117
How to Export Data of a Specific Application Component.....	119

## Accessing and Configuring Elements of a ConfigurationDesk Application 121

Accessing Elements.....	122
Basics on Accessing Elements.....	122
How to Select Elements of the Same Type.....	123
How to Show Elements in a Different Pane.....	125
How to Select and Show Assigned Elements.....	127
How to Find Elements of a ConfigurationDesk Application.....	129
Configuring Elements with the Properties Browser.....	131
Introduction to the Properties Browser.....	131

Configuring Properties in the Properties Browser.....	134
Expanding and Collapsing Categories in the Properties Browser.....	138
Changing the Intersection Modes in the Properties Browser.....	140
Using Filters in the Properties Browser.....	143
Use Scenarios for Modes and Filters in the Properties Browser.....	145
Using Tables to Access and Configure Elements.....	151
Introduction to Tables.....	151
Customizing Table Rows and Columns.....	154
Using Display Filters.....	158
Basics on Display Filters.....	158
Available Display Filters.....	160
How to Filter for Specific Signal Chain Elements.....	162
How to Filter for Signal Chain Elements from Associated Working Views.....	165
How to Filter Columns by Text and Regular Expressions.....	167
<b>Handling the Signal Chain in Working Views</b>	<b>169</b>
Using Working Views.....	170
Basics on Working Views.....	170
How to Create Working Views and Working View Groups.....	174
How to Open Working Views in the Signal Chain Browser.....	175
How to Open Working Views in the Model Communication Browser.....	178
How to Generate Working Views from Signal Chain Elements.....	180
How to Export and Import Elements from a Working View.....	182
Customizing the Display of Working Views.....	186
Vertical Alignment of Blocks.....	186
Ordering Blocks.....	187
Collapsing and Expanding Blocks.....	189
Changing the Background Color of Blocks.....	193
Increasing and Decreasing the Size of Displayed Signal Chain Elements.....	194
Highlighting Signal Chain Conflicts.....	194
Filtering Ports.....	197
Customizing the Display of the Working View Columns.....	198
<b>Managing Real-Time Hardware</b>	<b>201</b>
Handling Registered Hardware.....	202
Registering Hardware in ConfigurationDesk.....	202
Basics on Registering Real-Time Hardware.....	202

Basics on Connecting a ConfigurationDesk Application to a Hardware System.....	207
How to Register dSPACE Real-Time Hardware.....	208
How to Register Hardware for Multi-PU Applications and Add it to a ConfigurationDesk Application.....	211
Updating the Firmware of SCALEXIO or MicroAutoBox III Hardware.....	214
Basics on Updating SCALEXIO or MicroAutoBox III Firmware.....	214
How to Update or Repair SCALEXIO or MicroAutoBox III Firmware via ConfigurationDesk.....	219
Configuring Hardware Properties.....	221
How to Access Hardware Properties.....	222
How to Change a System Name.....	223
Configuring the I/O Ethernet Communication.....	225
How to Configure Internal Ethernet Switches.....	225
How to Configure I/O Ethernet Ports.....	227
Platform Access by Several Users Simultaneously.....	229
Synchronized Platform Management with Several dSPACE Products.....	229
Accessing the SCALEXIO or MicroAutoBox III System with Multiple dSPACE Products Simultaneously.....	232
Working with Hardware Topologies.....	236
Basics on Hardware Topologies.....	236
How to Create Hardware Topologies from Scratch.....	241
How to Extend Existing Hardware Topologies .....	243
How to Export a Hardware Topology Part.....	244
How to Merge Hardware Topologies.....	245
How to Establish a Network Connection in the Hardware Topology (SCADEXIO).....	246
How to Assign Boards to Specific Slots in an I/O Slot Unit.....	248
<b>Specifying the External Device Interface</b>	<b>251</b>
Basics on External Devices.....	252
Basics on Device Topologies.....	252
Basics on Device Pin Assignment.....	255
Basics on Device Blocks.....	257
Creating and Extending Device Topologies.....	260
How to Create and Extend Device Topologies via External Device Browser.....	260
How to Copy and Paste Device Topology Elements.....	263
How to Move Device Topology Elements.....	265

How to Merge Device Topologies.....	265
How to Export a Device Topology to a Microsoft Excel Sheet.....	267
Importing a Device Topology from a Microsoft Excel Sheet.....	268
Rules for Editing External Device Topologies.....	268
Configuring External Devices.....	272
Basics on Configuring External Devices.....	272
How to Rename Device Topology Elements.....	276
How to Assign Reference Ports.....	276
How to Assign Device Ports to External Device Pins.....	278
How to Add Custom Device Properties.....	281
How to Delete Custom Properties.....	282
How to Group Device Pins.....	283
Adding Device Topology Elements to the Signal Chain.....	287
Basics on Device Block Handling.....	287
How to Add Device Topology Elements to the Signal Chain.....	288
<b>Implementing I/O Functionality</b>	<b>291</b>
Basics on Implementing I/O Functionality.....	292
Basics on Instantiating Function Blocks.....	292
Basics on Function Blocks.....	297
Characteristics of Signal Ports.....	301
Characteristics of Function Ports.....	303
Characteristics of Event Ports.....	305
Adding Function Blocks to the Signal Chain.....	307
How to Add Function Blocks to the Signal Chain via Function Browser.....	307
How to Add Function Blocks to the Signal Chain via Device Ports.....	308
How to Add Function Blocks via Copy and Paste.....	309
Device Port Mapping.....	312
Methods for Device Port Mapping.....	312
Mapping Conflicts.....	315
Basics on Transferring Port Settings.....	315
How to Transfer Port Settings from Device Ports to Signal Ports.....	317
Configuring Function Blocks.....	318
Basics on Function Block Configuration.....	318
Categories of Configurable Parameters.....	320
Basics on Hardware Resources and Channel Types.....	321
How to Rename Function Blocks.....	324
How to View Circuit Diagrams of Hardware Resources.....	325

Implementing Function Triggers.....	328
Basics on Function Triggers.....	328
How to Avoid Multiple Function Triggers.....	332
<b>Adding Bus and Gigalink Communication to the Signal Chain</b>	<b>335</b>
Building the Signal Chain for CAN Bus Communication.....	336
Basics on Implementing CAN Communication.....	336
Recommended Workflow for Implementing CAN Communication.....	339
Building the Signal Chain for LIN Bus Communication.....	344
Basics on Implementing LIN Communication.....	344
Recommended Workflow for Implementing LIN Communication.....	347
Building the Signal Chain for FlexRay Communication.....	351
Basics on Implementing FlexRay Communication.....	351
Recommended Workflow for Implementing FlexRay Communication .....	356
Building the Signal Chain for Communication Using an Ethernet Interface.....	362
Implementing an Ethernet Interface.....	362
Building the Signal Chain for Gigalink Communication.....	363
Basics on Gigalink Communication.....	363
How to Build the Signal Chain for Gigalink Communication.....	366
<b>Adding FPGA Applications to the Signal Chain</b>	<b>369</b>
Steps to Implement an FPGA Application.....	370
<b>Specifying Failure Simulation in ConfigurationDesk</b>	<b>373</b>
Introduction to Failure Simulation.....	373
Specifying Failure Simulation Settings.....	374
Load Rejection and Signal Dropping.....	378
Providing the Specified Data to Experiment Software.....	381
<b>Handling Loads</b>	<b>385</b>
Basics on Using Internal or External Loads.....	385
Basics on Load Rejection.....	389
Details on Handling Internal Loads.....	392
Details on Handling External Loads.....	395
How to Provide Data of Internal Loads to the Hardware.....	397

## Assigning Hardware Resources to Function Blocks 399

Basics on Hardware Resource Assignment.....	399
Methods for Assigning Hardware Resources.....	404
Assignment Conflicts and Their Effects on Code Generation.....	409
How To Assign Hardware Resources Manually via Properties Browser.....	410
How to Assign Hardware Resources Manually via Drag & Drop.....	412
How to Assign Hardware Resources Automatically.....	414
How to Reuse the Hardware Resource Assignment of Missing Hardware.....	415

## Specifying the Model Interface 417

Basics on the Model Interface.....	418
Handling the Model Interface.....	418
Data Interchange Between ConfigurationDesk and Behavior Model.....	420
Specifying Options for Creating Model Port Blocks.....	421
Specifics on Handling the ConfigurationDesk Model Interface of Multimodel Applications.....	424
Basics on Model Port Blocks in ConfigurationDesk.....	426
Characteristics of Model Port Blocks.....	426
Characteristics of Model Ports.....	429
Adding Model Port Blocks to the Signal Chain.....	432
Basics on Adding Model Port Blocks to the Signal Chain.....	432
How to Add Model Port Blocks to the Signal Chain via Function Blocks.....	435
How to Add Model Port Blocks to the Signal Chain via the Model Browser.....	436
Model Port Mapping.....	438
Methods for Model Port Mapping.....	438
Rules for Model Port Mapping.....	443
Mapping States and Their Effects.....	445
Mapping Algorithm for Model Port Blocks With Structured Data Ports.....	446
Setting Up Model Communication.....	448
Basics on Model Communication.....	448
Configuring Data Snapshots.....	451
Basics on Function Modules (Advanced).....	454
Ensuring Data Consistency in Model Communication (Advanced).....	455
How to Create Communication Packages.....	459

How to Assign Data Import Blocks to Other Communication Packages.....	460
How to Change the Protocol of Communication Packages.....	461
Working with Model Port Blocks That Provide Variable-Size Signals.....	462
Model Communication Recommendations for Model Port Blocks with Structured Data Ports.....	464
Configuring Model Port Blocks.....	466
How to Access Model Port Block and Model Port Data.....	466
How to Rename Model Port Blocks.....	468
<b>Modeling Executable Applications and Tasks</b>	<b>471</b>
Introduction to Modeling Executable Applications and Tasks.....	472
Terms and Definitions for Building Executable Applications.....	472
Basics on Modeling Executable Applications.....	475
Basics on Tasks, Events, and Runnable Functions.....	480
Basics on Task Overruns.....	483
Basics on Modeling Tasks in ConfigurationDesk.....	486
Different Simulink Tasking Modes and Their Effects in ConfigurationDesk.....	488
How to Model an Executable Application Manually from Scratch.....	489
Using Multiple Model Implementations in the Same Application Process.....	491
Rules for Optimizing the Configuration of Application Processes.....	495
Replacing Model Implementations.....	498
Advanced: Effects of Replacing Components Used in an Application Process.....	500
Modeling Asynchronous Tasks.....	502
Basics on Modeling Asynchronous Tasks.....	502
How to Enable Event Generation for Function Blocks.....	504
How to Model Asynchronous Tasks for Models with Hardware-Triggered Runnable Function Blocks.....	505
How to Model Asynchronous Tasks and Create Suitable Runnable Function Blocks in One Step.....	508
How to Model Asynchronous Tasks for Models with Runnable Function Blocks (Manually).....	511
Configuring Tasks in ConfigurationDesk.....	513
Basics on Configuring Tasks.....	513
How to Configure Tasks in ConfigurationDesk.....	515
How to Configure the Start-Up Behavior of a Periodic Task.....	516
Configuring Delayed Tasks.....	517
Configuring Application Processes for Bus Monitoring.....	519

Using Application Processes Without Behavior Models.....	521
Introduction to Application Processes Without Behavior Models.....	521
How to Create Application Processes That Provide Default Tasks.....	523
Creating Multi-Processing-Unit Applications With ConfigurationDesk.....	525
Basics on Multi-Processing-Unit Applications.....	525
How to Create Processing Unit Applications.....	527
How to Assign Processing Units to Processing Unit Applications.....	528
<b>Working with Simulink Behavior Models</b>	<b>531</b>
Handling the Model Interfaces of ConfigurationDesk and Simulink Behavior Models.....	532
Handling MATLAB via ConfigurationDesk.....	532
Basics of Connecting ConfigurationDesk and Simulink Behavior Models.....	534
Effects of Changing Model Interfaces of Simulink Behavior Models.....	538
Synchronizing the Model Interfaces of ConfigurationDesk and Simulink.....	540
Transferring the ConfigurationDesk Model Interface to a New Simulink Interface Model.....	542
How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model.....	543
How to Initialize Simulink Behavior Models.....	546
Workflow for Creating a Multicore Real-Time Application Using Multiple Behavior Models.....	548
Workflow for Creating a Multi-PU Application Using Multiple Behavior Models.....	550
User-Friendly Connection of ConfigurationDesk and Simulink Models.....	553
Working with the Model-Function Mapping Browser.....	553
Creating Signal Chains via the Model-Function Mapping Browser.....	555
Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model.....	558
Analyzing Simulink Behavior Models.....	561
Deleting Model Port Blocks in ConfigurationDesk and in Simulink Simultaneously.....	564
Switching Between Model Port Blocks in ConfigurationDesk and in Simulink.....	566
Creating Multimodel Real-Time Applications With Models Separated From One Overall Model.....	569
Basics of Creating Multicore or Multi-PU Applications Using One Overall Model.....	569

Workflow for Creating a Multicore Real-Time Application Using One Overall Behavior Model.....	570
Workflow for Creating a Multi-PU Application Using One Overall Behavior Model.....	573
Working With MCD Files.....	576
Special Use Cases of Working With Simulink Behavior Models.....	578
Using Model Port Blocks to Handle Simulink Behavior Model Variants.....	578
Creating Model Port Blocks With Structured Data Ports for Bus Signals.....	579
Simplified Preparation of Model Interfaces for Model Communication.....	582
How to Create Inverse Model Port Blocks for Model Communication .....	583
<b>Working with Container Files as Behavior Models</b>	<b>585</b>
Adding Container Files to a ConfigurationDesk Application.....	586
Introduction to Working With Container Files as Behavior Models.....	586
How to Add Model Implementation Containers to a ConfigurationDesk Application.....	586
Working with Simulink Implementation Containers.....	588
Basics on Simulink Implementation Containers.....	588
Workflow for Integrating Simulink Implementation Containers in Executable Applications.....	590
Adding Simulink Implementation Containers to a ConfigurationDesk Application.....	591
Creating Precompiled SIC Files.....	592
Updating Simulink Implementation Containers in ConfigurationDesk .....	595
Working with Bus Simulation Containers.....	596
Adding Bus Simulation Containers to a ConfigurationDesk Application.....	596
Workflow for Integrating Bus Simulation Containers in Executable Applications.....	599
Updating Bus Simulation Containers in ConfigurationDesk.....	601
Creating Precompiled BSC Files.....	602
Build Results for Real-Time Applications Containing Bus Simulation Containers.....	605
Working with Functional Mock-up Units.....	607
Introduction to the FMU Support.....	607
Definition of the FMI Standard and FMUs.....	608
Preconditions for Using FMUs in ConfigurationDesk.....	609

Handling FMUs in a ConfigurationDesk Application.....	610
Workflow for Integrating FMUs in Executable Applications.....	611
Updating FMUs in ConfigurationDesk.....	615
Special Aspects of Runnable Functions Resulting from FMUs.....	616
Creating Precompiled FMUs.....	618
Variable Description File Entries for Variables Provided by FMUs.....	620
Special Aspects of the Simulation Behavior.....	623
Working with V-ECU Implementations.....	625
Handling V-ECU Implementations in ConfigurationDesk.....	625
Basics on V-ECU Implementations.....	625
Workflow for Integrating V-ECU Implementations in Executable Applications.....	629
Adding V-ECU Implementations to a ConfigurationDesk Application.....	632
Updating V-ECU Implementations Used in a ConfigurationDesk Application.....	633
Configuring Executable Applications Containing V-ECU Implementations.....	635
Configuring Properties of Tasks and Events.....	635
Delaying the Start of V-ECU Implementations.....	638
Special Aspects of V-ECU Implementations Containing CAN Controllers.....	639
Special Aspects of V-ECU Implementations Containing LIN Controllers.....	639
Details on the Build Process for Real-Time Applications Containing V-ECU Implementations.....	641
Configuring the Build Process for ConfigurationDesk Applications Containing V-ECU Implementations.....	642
Build Result Files of ConfigurationDesk Applications Containing V-ECU Implementations.....	644
Working with V-ECU Implementations That Provide Memory Segments.....	646
Working with V-ECU Implementations That Provide Virtual Bypassing Support.....	647

<b>ECU Interfacing with SCALEXIO or MicroAutoBox III Systems</b>	<b>649</b>
Introduction to ECU Interfacing with SCALEXIO or MicroAutoBox III Systems.....	650
Introduction to ECU Interfacing.....	650
Basics on ECU Interfacing with SCALEXIO or MicroAutoBox III Systems.....	652
Implementing ECU Interfacing in ConfigurationDesk.....	656
Typical Workflow for Implementing ECU Interfacing in ConfigurationDesk.....	656
How to Import ECU Interface Container (EIC) Files.....	658
Workflow for Modeling the Triggering Behavior in ECU Interfacing Scenarios via MATLAB/Simulink.....	659
Examples for Modeling the Triggering Behavior in ECU Interfacing Scenarios via MATLAB/Simulink.....	662
Notes on Model Interface Elements Generated for ECU Interface Configuration Function Blocks.....	667
Notes on Specifying CAN ECU Interfaces.....	667
Preconfigured Preprocessor Macros for ECU Interfacing.....	669
Updating ECU Interface Containers in ConfigurationDesk Applications.....	671
Basics on Updating ECU Interface Containers in ConfigurationDesk Applications.....	671
How to Update an ECU Interface Container in the ConfigurationDesk Application.....	674
<b>Resolving Conflicts</b>	<b>677</b>
Basics on Resolving Conflicts.....	677
Details on Filtering Conflicts.....	679
How to Resolve Conflicts via the Conflicts Viewer.....	681
<b>Calculating an External Cable Harness</b>	<b>683</b>
Basics on the External Cable Harness.....	683
How to (Re)Calculate the External Cable Harness.....	685
How to View the Wiring Information.....	687
How to Export Wiring Information for an External Cable Harness.....	689
Description of an Exported Cable Harness Microsoft Excel™ Sheet.....	690
Examples of Exported Wiring Information.....	693

## Building Real-Time Applications 697

Understanding the Build Process.....	698
Introducing the Build Process.....	698
Build Result Files of the Build Process.....	702
Configuring the Build Process.....	707
Specifying Options for the Build Process.....	707
Customizing the Build Process with User-Specific Files (For Simulink Behavior Models).....	713
How to Configure the Build Process.....	714
Starting the Build Process.....	716
Details on the Build Process.....	716
How to Start the Build Process.....	719
Information on Variable Description Files (TRC Files).....	722
Generating TRC Files.....	722
Basics on the Generation of TRC Files.....	722
Structure of a TRC File for SCALEXIO or MicroAutoBox III Systems.....	724
Available TRC File Variable Entries and Their Locations in the TRC File.....	724
Simulation and RTOS Group.....	726
Signal Chain Group.....	727
Diagnostics Group.....	729
XIL API/EESPort Group.....	730
Inclusion of Variables from the Simulink Model into the TRC File.....	731
Details on the Variable Groups of the Behavior Model.....	731
Methods of Including Variables of the Behavior Model in the TRC File.....	734
Adapting the Generation of the Variable Description File.....	736
Conditions for the Inclusion of Variables from the Simulink Model into the TRC File.....	740
Conditions for the Inclusion of Variables From Referenced Models.....	742

## Downloading and Executing Real-Time Applications 745

Basics on Downloading Real-Time Applications.....	745
How to Download a Real-Time Application.....	748
How to Start a Real-Time Application.....	751
Application States of a Real-Time Application.....	753
Handling Errors and Exceptions When Executing Real-Time Applications.....	755

Exporting Data for Documentation Purposes	757
How to Export the Configuration of an Application.....	757
Description of the Exported Configuration.....	758
How to Export Data for Documentation Purposes.....	760
Limitations	763
Limitations for Starting ConfigurationDesk.....	764
Limitations Concerning Hardware.....	764
Limitations Concerning Projects and Applications.....	765
Limitations Concerning the Signal Chain.....	765
Limitations Concerning Function Blocks.....	767
Limitations Concerning Multi-Processing-Unit Systems.....	767
Limitations Concerning Multiple Model Implementations Assigned to the Same Application Process.....	768
Limitations Concerning V-ECU Implementations.....	769
Limitations Concerning Simulink Implementation Containers.....	772
Limitations Concerning Functional Mock-up Units (FMUs).....	773
Limitations Concerning Bus Simulation Containers.....	774
Limitations Regarding Simulink Variables in the TRC File.....	776
Limitations for Ethernet Communication.....	777
Limitations for FlexRay Communication.....	778
Limitations for SENT Communication.....	778
Limitations for Gigalink Communication.....	778
Limitations Concerning ECU Interfacing with SCALEXIO or MicroAutoBox III.....	779
Limitations Concerning the Build and Download Process.....	779
Limitations Concerning MATLAB Compatibility.....	780
ConfigurationDesk Glossary	785
Appendix	811
File Types and Keyboard Shortcuts.....	812
File Types.....	812
Available Keyboard Shortcuts.....	814
Version-Specific Migration Steps.....	820
Migrating from ConfigurationDesk 6.5 to 6.6.....	820
Migrating from ConfigurationDesk 6.3 to 6.4.....	821
Migrating from ConfigurationDesk 6.1 to 6.2.....	822

Migrating from ConfigurationDesk 6.0 to 6.1.....	823
Migrating from ConfigurationDesk 5.7 to 6.0.....	823
Migrating from ConfigurationDesk 5.6 to 5.7.....	825
Migrating from ConfigurationDesk 5.5 to 5.6.....	825
Migrating from ConfigurationDesk 5.4 to 5.5.....	825
Migrating from ConfigurationDesk 5.3 to 5.4.....	826
Migrating from ConfigurationDesk 5.2 to 5.3.....	826
Migrating from ConfigurationDesk 5.1 to 5.2.....	827
Migrating from ConfigurationDesk 4.4 to 5.0.....	829
Migrating Projects from ConfigurationDesk 4.2 (on dSPACE Release 7.3) or Earlier to 4.3 or 4.4.....	830

<b>Index</b>	<b>835</b>
--------------	------------



# About This Guide

## Content

This guide introduces you to ConfigurationDesk and shows you how to implement real-time applications:

- Setting up projects and applications which hold all the data that is relevant for implementing real-time applications
- Implementing I/O functionality by using and configuring function blocks
- Specifying the model interface with model port blocks
- Triggering I/O functions and configuring tasks
- Calculating external wiring information
- Building and downloading real-time applications to SCALEXIO or MicroAutoBox III hardware

## Target group

This guide is primarily targeted at engineers who implement and build real-time applications.

The target group performs some or all of the following tasks:

- Implement I/O functionality
- Specify the model interface
- Implement I/O triggers and tasks
- Build and download real-time applications

## Required knowledge

The target group must have the following knowledge:

- Familiar with the basics of SCALEXIO or MicroAutoBox III hardware
- Familiar with Windows applications

## Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
 <b>DANGER</b>	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.

Symbol	Description
	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
	Indicates a hazard that, if not avoided, could result in property damage.
	Indicates important information that you should take into account to avoid malfunctions.
	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](http://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.



# Safety Precautions

## Where to go from here

## Information in this section

General Warning .....	23
Safety Precautions for Configuration and Mapping .....	24

## General Warning

### Danger potential

Using dSPACE software can be dangerous. You must observe the following safety instructions and the relevant instructions in the user documentation.

#### ⚠ WARNING

**Improper or negligent use can result in serious personal injury and/or property damage.**

Using the ConfigurationDesk software can have a direct effect on dSPACE systems and technical (electrical, hydraulic, mechanical) systems connected to them.

- **Only persons who are qualified to use dSPACE software, and who have been informed of the above dangers and possible consequences, are permitted to use it.**
- All applications where malfunctions or misoperation involve the danger of injury or death must be examined for potential hazards by the user, who must if necessary take additional measures for protection (for example, an emergency off switch).

<b>Liability</b>	<p>It is your responsibility to adhere to instructions and warnings. Any unskilled operation or other improper use of this product in violation of the respective safety instructions, warnings, or other instructions contained in the user documentation constitutes contributory negligence, which may lead to a limitation of liability by dSPACE GmbH, its representatives, agents and regional dSPACE companies, to the point of total exclusion, as the case may be. Any exclusion or limitation of liability according to other applicable regulations, individual agreements, and applicable general terms and conditions remain unaffected.</p>
<b>Data loss during operating system shutdown</b>	<p>The shutdown procedure of Microsoft Windows operating systems causes some required processes to be aborted although they are still being used by dSPACE software. To avoid data loss, the dSPACE software must be terminated manually before a PC shutdown is performed.</p>

## Safety Precautions for Configuration and Mapping

---

<b>Objective</b>	<p>To avoid risk of injury and/or damage to the hardware and to achieve safe and trouble-free operation, you have to observe the following rules.</p>
<b>Assigning device ports to device pins</b>	<p>You assign the device ports to device pins of your external device connector with the device pin assignment. The device pin assignment is required to calculate the external wiring information.</p> <div style="background-color: #f0f0f0; padding: 10px; border: 1px solid #ccc;"><p><b>NOTICE</b></p><p><b>If the device pin assignment is incorrect the wiring information that is calculated (for the external harness) will not match to the electrical requirements of the external devices (for example your ECU).</b></p><p>Risk of damage to the connected devices. Ensure that the device pin assignment matches the pin configuration of the device.</p></div>
<b>Mapping device ports to signal ports</b>	<p>The basis for the mapping rules between device ports and signal ports is the configuration of the device ports. The setting of the port type (inport, outport, bidirectional port, reference port, undefined port) plays a major role.</p>

**NOTICE**

**If the device port configuration is incorrect, and/or the device port mapping is incorrect, the wiring information that is calculated (for the external harness) will not match to the electrical requirements of the external devices (for example your ECU).**

Risk of damage to the connected devices and/or the dSPACE real-time hardware.

- Configure the device ports and particularly the port type so that they correspond as closely as possible to the characteristics of your device. Use the undefined port type only as an exception.
- Check the mapping lines. Ensure, that the mapping will not result in damage to the connected device because the electrical characteristics of the mapped ports do not match.

### Simulating failures for multiple signals simultaneously

Using the Failure Simulation module of your experiment software, you can simulate failures for multiple signals simultaneously. For example, you can simulate a situation where a short to GND for one signal and an open circuit for a different signal happen at the same time.

As a precondition in ConfigurationDesk, the Activation by FRU relay property must be set to Allowed for all signals which you want to use in a multiple failure scenario.

In multiple failure scenarios, the SCALEXIO system uses the electro-mechanical relays of the failure routing unit (FRU relays) to activate the failure directly by switching the relays under load instead of just switching the signal to the failplane.

**NOTICE**

#### Risk of increased wear and permanent damage

When you use the "Activation by FRU relay" feature, there is a risk of increased wear and permanent damage to the relays of the dSPACE failure simulation hardware. For details, refer to [Safety Precautions for Simulating Electrical Errors with a SCALEXIO System \(SCALEXIO – Hardware and Software Overview\)](#).

### Load compare check during download

If you download a real-time application to your hardware, ConfigurationDesk checks if the channel's internal load description which was used during the build process matches the internal load description stored on the hardware. If there is a deviation, then the application is possibly not compatible with the hardware system. ConfigurationDesk generates a warning message and you can abort the download.

You cannot disable this check.

**NOTICE**

**Ignoring the warning message can cause material damage.**

Risk of damage to the connected external device.

- Cancel the download of the real-time application if you are not sure about the consequences.
- Disable the affected channels of the internal loads if you do not need them.
- Ensure that any rejection of the affected internal loads cannot result in damage to the connected external device.  
If the two internal load descriptions conflict, load rejection of the affected internal load will always be enabled.

# Introduction to ConfigurationDesk

---

## Where to go from here

## Information in this section

Features of ConfigurationDesk.....	28
Compatibility Information.....	36
Required Licenses.....	39
Migrating From Prior Versions of ConfigurationDesk.....	46
Starting ConfigurationDesk.....	48
User Interface of ConfigurationDesk.....	50

# Features of ConfigurationDesk

## Where to go from here

## Information in this section

Overview of Main ConfigurationDesk Features.....	28
Development of ConfigurationDesk's Key Features.....	29

## Overview of Main ConfigurationDesk Features

### Overview of main features

ConfigurationDesk supports the implementation of real-time applications. Some of its main features are described in the following table:

Feature	Description	
	License-Protected Features	License-Free Features
Registering hardware systems	–	SCALEXIO and MicroAutoBox III hardware can be registered in ConfigurationDesk via IP address, MAC address, system name or serial number. An open project or application is not necessary for connecting and displaying hardware systems in ConfigurationDesk.
Creating and managing projects	ConfigurationDesk's Project Manager allows you to organize all the relevant project information, such as project and application-specific data.	You can open projects. Creating, editing, and saving projects is not possible.
Specifying a signal chain	The logical path of the signal from the ECU pin to the model and to the channel on the real-time hardware is shown in a signal chain which you can create and configure to match your requirements.	–
Implementing and configuring I/O functionality	You can implement and configure functions and add them to the signal chain.	–
Calculating the external wiring information	The wiring information of the external cable harness can be calculated in ConfigurationDesk and then exported to a human-readable file.	–
Modeling executable applications and tasks	ConfigurationDesk allows you to model executable applications (real-time applications) and the tasks used in them very flexibly to match your requirements.	–

Feature	Description	
	License-Protected Features	License-Free Features
Building real-time applications	ConfigurationDesk allows you to configure and start the build process for real-time applications (single-core, multicore real-time applications and multi-processing unit applications).	–
Downloading real-time applications	–	ConfigurationDesk's Platform Manager allows you to download, start, stop and unload single-core real-time applications, multicore real-time applications and multi-processing unit applications.
Conflicts	When you work with ConfigurationDesk, conflicts might arise that influence the results of the build process and thus the real-time application. ConfigurationDesk's Conflicts Viewer displays all conflicts and helps you to resolve them.	–
System messages	–	The Message Viewer in ConfigurationDesk displays all system messages in chronological order.

## Development of ConfigurationDesk's Key Features

**New features of the current version** For the description of new features of the current ConfigurationDesk version, refer to [ConfigurationDesk \(New Features and Migration\)](#).

**Overview of feature developments** The following table provides an overview of the development of key features in different areas of ConfigurationDesk.

ConfigurationDesk Version	New Supported Hardware	New and Enhanced I/O Function Block Types	Bus Manager	Other Features
6.6 dSPACE Release 2020-B	SCALEXIO: ▪ New Linux-based operating system for SCALEXIO	New: ▪ FuSa System Monitoring Enhanced: ▪ SENT In ▪ SENT Out ▪ Digital Encoder In ▪ Ethernet Switch ▪ Multiple function block types support	Support of AUTOSAR Classic Platform Release R19-11	<ul style="list-style-type: none"> <li>▪ New Linux-based operating system for SCALEXIO processing hardware</li> <li>▪ New V-ECU container format</li> <li>▪ V-ECU: Support of additional CAN and LIN features for MCAL</li> <li>▪ Support of int64/uint64 ports</li> </ul>

ConfigurationDesk Version	New Supported Hardware	New and Enhanced I/O Function Block Types	Bus Manager	Other Features
	<ul style="list-style-type: none"> <li>processing hardware</li> <li>▪ DS6336-PE Ethernet Board</li> <li>▪ DS6336-CS Ethernet Board</li> <li>▪ High Parallel Performance Real-Time PC</li> <li>▪ DS2502 IOCNET Link Board</li> </ul>	<ul style="list-style-type: none"> <li>the Differential out interface type</li> <li>▪ Custom function blocks: Including external artifacts in RTA</li> </ul>		<ul style="list-style-type: none"> <li>▪ New features for ECU interfacing with SCALEXIO and MicroAutoBox III</li> </ul>
6.5 dSPACE Release 2020-A	<ul style="list-style-type: none"> <li>SCALEXIO:</li> <li>▪ DS6651 Multi I/O Module</li> <li>▪ DS6342 CAN Board</li> <li>▪ Support of non-volatile memory access on SCALEXIO Processing Units</li> <li>▪ MicroAutoBox III:</li> <li>▪ DS1521 Bus Board</li> </ul>	<p>New:</p> <ul style="list-style-type: none"> <li>▪ Engine Control Setup</li> <li>▪ Engine Angular Pulse Out</li> <li>▪ Crank In</li> <li>▪ Cam In</li> <li>▪ Ignition Out</li> <li>▪ Injection Out</li> <li>▪ Knock In</li> <li>▪ FuSa Setup</li> <li>▪ FuSa Response Trigger</li> <li>▪ FuSa Challenge-Response Monitoring</li> <li>▪ PTP Master</li> <li>▪ PTP Slave</li> <li>▪ Domain Clock</li> </ul> <p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ Multiple function block types support new channel types</li> </ul>	<ul style="list-style-type: none"> <li>▪ Enhanced J1939 support</li> <li>▪ Filtering CAN frame gateways</li> <li>▪ Enhanced bus configuration features</li> <li>▪ Enhanced support of CAN and LIN communication</li> <li>▪ Generating bus simulation containers for offline simulation applications on Linux</li> <li>▪ Basic ARXML communication matrix</li> </ul>	<ul style="list-style-type: none"> <li>▪ Automatic configuration optimization in multimodel application processes</li> <li>▪ Support of variable-size signals for model communication</li> <li>▪ Support of V-ECU implementations on MicroAutoBox III</li> <li>▪ Support of V-ECU implementations with virtual bypassing</li> <li>▪ V-ECU: Basic support of CAN and LIN modules for MCAL</li> <li>▪ Generation and support of precompiled BSC files</li> <li>▪ Support of BSC files and bus configurations in the same application process</li> </ul>
6.4 dSPACE Release 2019-B	<ul style="list-style-type: none"> <li>▪ MicroAutoBox III</li> <li>▪ DS6121 Multi-I/O Board</li> <li>▪ DS6321 UART Board</li> </ul>	<p>New:</p> <ul style="list-style-type: none"> <li>▪ Block-Commutated PWM Out</li> <li>▪ Sine Encoder In</li> <li>▪ Hall Encoder In</li> <li>▪ Resolver In</li> <li>▪ EnDat Master</li> <li>▪ SSI Master</li> <li>▪ Ethernet Switch</li> <li>▪ Digital Pulse In</li> <li>▪ Voltage Signal Capture (ADC Type 4)</li> <li>▪ SPI Master</li> <li>▪ Non-Volatile Memory Access</li> </ul> <p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ Multiple function block types support new channel types</li> </ul>	<ul style="list-style-type: none"> <li>▪ Support of J1939</li> <li>▪ Improved import behavior for LDF files</li> <li>▪ New and enhanced bus configuration features</li> <li>▪ Adding bus configuration features to multi-selected, arbitrary elements</li> <li>▪ Editable names of communication cluster nodes and bus access requests</li> </ul>	<ul style="list-style-type: none"> <li>▪ Support of TargetLink-generated SIC files in multimodel application processes.</li> <li>▪ New V-ECU implementation license required</li> <li>▪ Support of V-ECU implementations that provide memory segments</li> <li>▪ Configurable XCP service for Simulink models and SIC files with A2L file fragments</li> <li>▪ New command for easily replacing model implementations</li> <li>▪ Support of BSC files in multimodel application processes</li> </ul>

ConfigurationDesk Version	New Supported Hardware	New and Enhanced I/O Function Block Types	Bus Manager	Other Features
6.3 dSPACE Release 2019-A	<ul style="list-style-type: none"> <li>▪ DS6601 FPGA Base Board</li> <li>▪ DS6602 FPGA Base Board</li> <li>▪ SCALEXIO AutoBox (8-slot)</li> <li>▪ Support of UART serial communication with the SCALEXIO Processing Unit and the DS6001 Processor Board</li> </ul>	<p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ Virtual Ethernet Setup</li> </ul>	<ul style="list-style-type: none"> <li>▪ Support of global time synchronization (GTS)</li> <li>▪ Exchanging CAN bus communication between two communication clusters (CAN gateway)</li> <li>▪ Verifying authentication information of received secured IPDUs</li> <li>▪ Enhanced AUTOSAR support</li> </ul>	<ul style="list-style-type: none"> <li>▪ Configuring bus monitoring for application processes</li> <li>▪ ECU interfacing: support of CAN ECU interfaces</li> </ul>
6.2 dSPACE Release 2018-B	<ul style="list-style-type: none"> <li>▪ DS6333-CS Automotive Ethernet Board</li> <li>▪ DS6333-PE Automotive Ethernet Board</li> <li>▪ DS6334-PE Ethernet Board</li> <li>▪ DS6335-CS Ethernet Board</li> <li>▪ SCALEXIO LabBox (8-slot)</li> </ul>	<p>New:</p> <ul style="list-style-type: none"> <li>▪ Virtual Ethernet Setup</li> </ul> <p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ CAN</li> <li>▪ LIN</li> <li>▪ SENT In</li> <li>▪ SENT Out</li> <li>▪ Ethernet Setup</li> <li>▪ Improved angular processing units (APU) for multiple function block types</li> </ul>	<ul style="list-style-type: none"> <li>▪ AUTOSAR 4.3.1 support</li> <li>▪ Support of secure onboard communication (SecOC)</li> <li>▪ Generating bus simulation containers without a mapped behavior model</li> <li>▪ Enhanced port interface of bus simulation containers</li> <li>▪ Support of user code</li> <li>▪ New and enhanced bus configuration features</li> <li>▪ Adding ISignal IPDUs and ISignals to communication matrices</li> <li>▪ Enhanced bus configuration tables</li> </ul>	<ul style="list-style-type: none"> <li>▪ Support of delayed tasks</li> <li>▪ Support of SIC files generated with TargetLink</li> </ul>
6.1 dSPACE Release 2018-A	DS6241 D/A Board	<p>New:</p> <ul style="list-style-type: none"> <li>▪ TCP</li> <li>▪ Angular Wavetable Voltage Out</li> <li>▪ Angular Wavetable Digital Out</li> <li>▪ Power On Signal In</li> <li>▪ System Shutdown</li> </ul> <p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ Enhanced use of trigger sources for</li> </ul>	<ul style="list-style-type: none"> <li>▪ FIBEX 4.1.2 support</li> <li>▪ Manipulating bus communication</li> <li>▪ New bus configuration features</li> <li>▪ Bus Manager Demo</li> </ul>	<ul style="list-style-type: none"> <li>▪ Support of multiple Simulink behavior models and SIC files assigned to the same application process</li> <li>▪ ECU interfacing: synchronous data access</li> </ul>

ConfigurationDesk Version	New Supported Hardware	New and Enhanced I/O Function Block Types	Bus Manager	Other Features
		<ul style="list-style-type: none"> <li>multiple function block types</li> <li>▪ Trigger In</li> <li>▪ Current In</li> <li>▪ Digital Pulse Capture</li> </ul>		
6.0 dSPACE Release 2017-B	<ul style="list-style-type: none"> <li>▪ DS6202 Digital I/O Board</li> <li>▪ DS6311 FlexRay Board</li> <li>▪ DS6341 CAN Board</li> <li>▪ DS6351 LIN Board</li> <li>▪ DS6551 IOCNET Link Board</li> </ul>	<p>New:</p> <ul style="list-style-type: none"> <li>▪ Digital Incremental Encoder In</li> <li>▪ Multi-Channel PWM Out</li> <li>▪ Digital Pulse Out</li> </ul> <p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ Wheelspeed Out</li> </ul>	<ul style="list-style-type: none"> <li>▪ AUTOSAR 4.3.0 support</li> <li>▪ New bus configuration features</li> <li>▪ New configurable communication matrix elements</li> <li>▪ New PDU elements to access supported PDU types</li> <li>▪ Enhanced bus configuration tables</li> </ul>	<ul style="list-style-type: none"> <li>▪ New licensing for dSPACE products</li> <li>▪ User interface consists of view sets for specific purposes</li> <li>▪ New Model-Function Mapping Browser</li> <li>▪ Runnable function blocks and event ports for modeling asynchronous tasks</li> <li>▪ Predefined hardware topologies available</li> <li>▪ Support of ECU calibration page handling and data access to individual ECU variables for ECU interfacing with SCALEXIO systems</li> </ul>
5.7 dSPACE Release 2017-A	<ul style="list-style-type: none"> <li>▪ DS6001 Processor Board</li> <li>▪ DS6221 A/D Board</li> <li>▪ DS6331-PE Ethernet Board</li> <li>▪ DS6332-CS Ethernet Board</li> </ul>	<p>New:</p> <ul style="list-style-type: none"> <li>▪ Trigger In</li> </ul> <p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ Voltage In</li> <li>▪ Voltage Signal Capture</li> </ul>	<ul style="list-style-type: none"> <li>▪ Inspecting bus communication</li> <li>▪ New supported PDU types</li> <li>▪ Support of end-to-end protected ISignal groups</li> <li>▪ New configurable communication matrix elements</li> <li>▪ New IPDU Enable bus configuration feature</li> <li>▪ Automatic assignment of bus accesses</li> </ul>	<ul style="list-style-type: none"> <li>▪ Support of new FPGA application types</li> <li>▪ Updating ECU Interface Container (EIC) files</li> <li>▪ Enhanced tasks</li> <li>▪ Download to flash memory (DS 6001)</li> </ul>
5.6 dSPACE Release 2016-B	–	<p>New:</p> <ul style="list-style-type: none"> <li>▪ Ethernet Setup</li> <li>▪ ECU Interface Configuration</li> </ul> <p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ SENT In</li> <li>▪ SENT Out</li> <li>▪ CAN</li> </ul>	<ul style="list-style-type: none"> <li>▪ Support of the SAE J2602 standard for LIN communication</li> <li>▪ Support of FBEX files based on FBEX 4.1.1 as communication matrices</li> </ul>	<ul style="list-style-type: none"> <li>▪ Enhanced tasks</li> <li>▪ Support of ECU interfacing</li> <li>▪ Adding bus simulation container files (BSC files) to your ConfigurationDesk application</li> </ul>

ConfigurationDesk Version	New Supported Hardware	New and Enhanced I/O Function Block Types	Bus Manager	Other Features
5.5 dSPACE Release 2016-A	<ul style="list-style-type: none"> <li>▪ DS6301 CAN/LIN Board</li> <li>▪ SCALEXIO LabBox</li> </ul>	<p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ Current Signal Capture</li> <li>▪ CAN</li> </ul>	<ul style="list-style-type: none"> <li>▪ Generation of bus simulation container (BSC) files, e.g., for simulating bus communication on VEOS</li> <li>▪ Configuring bus communication via bus configuration features</li> </ul>	<ul style="list-style-type: none"> <li>▪ Generation of working views based on project topologies (e.g., model topology, hardware topology)</li> <li>▪ Enhanced Properties Browser</li> <li>▪ Improved signal chain handling</li> <li>▪ Simplified generation of communication interfaces for behavior models (inverse model port blocks)</li> </ul>
5.4 dSPACE Release 2015-B	–	<p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ Current In</li> <li>▪ Voltage In</li> <li>▪ Current Signal Capture</li> <li>▪ Voltage Signal Capture</li> <li>▪ CAN</li> </ul>	<ul style="list-style-type: none"> <li>▪ Communication matrix modification</li> <li>▪ LIN Schedule Table function ports for LIN masters</li> </ul>	<ul style="list-style-type: none"> <li>▪ Support of enhanced trace file generation (Simulink Coder features concerning parameter handling introduced with MATLAB R2014a)</li> <li>▪ Support of precompiled SIC files</li> </ul>
5.3 dSPACE Release 2015-A	<ul style="list-style-type: none"> <li>▪ DS6101 Multi-I/O Board</li> <li>▪ DS6201 Digital I/O Board</li> </ul>	–	Bus Manager component introduced to implement bus communication in real-time applications for SCALEXIO systems	<ul style="list-style-type: none"> <li>▪ Product Model Interface Package for Simulink introduced</li> <li>▪ Adding Simulink implementation containers (SIC) to an executable application</li> </ul>
5.2 dSPACE Release 2014-B	–	<p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ SENT In</li> <li>▪ Lambda DCR</li> <li>▪ Lambda NCCR</li> <li>▪ Model access property for all function blocks</li> </ul>	–	<ul style="list-style-type: none"> <li>▪ New methods for modeling asynchronous tasks</li> <li>▪ Methods for using an application process without a behavior model</li> <li>▪ Improved user interface to make working with ConfigurationDesk applications easier and to provide convenient methods for common configuration tasks</li> <li>▪ Support of V-ECU implementations containing one or more LIN controllers</li> </ul>
5.1 dSPACE Release 2014-A	–	<p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ Injection/Ignition Current In</li> <li>▪ Injection/Ignition Voltage In</li> <li>▪ Power Switch</li> </ul>	–	<ul style="list-style-type: none"> <li>▪ Support of multi-processing-unit systems (multi-PU systems)</li> <li>▪ Support of Functional Mock-up Unit model implementations</li> </ul>

ConfigurationDesk Version	New Supported Hardware	New and Enhanced I/O Function Block Types	Bus Manager	Other Features
				<ul style="list-style-type: none"> <li>▪ Exporting and importing signal chain elements via CAFX file</li> <li>▪ Synchronized communication between SCALEXIO and PHS-bus applications via Gigalink</li> </ul>
5.0 dSPACE Release 2013-B	-	<p>New:</p> <ul style="list-style-type: none"> <li>▪ Waveform Current Sink</li> <li>▪ Waveform Voltage Out</li> <li>▪ Waveform Digital Out</li> <li>▪ Current Signal Capture</li> <li>▪ Voltage Signal Capture</li> </ul> <p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ CAN</li> <li>▪ Gigalink</li> <li>▪ SENT Out</li> </ul>	-	<ul style="list-style-type: none"> <li>▪ Ribbon user interface</li> <li>▪ Support of Simulink bus signals</li> <li>▪ Synchronized communication between SCALEXIO systems via Gigalink</li> </ul>
4.4 dSPACE Release 2013-A	<ul style="list-style-type: none"> <li>▪ DS2655 FPGA Base Module</li> <li>▪ DS2655M1 I/O Module</li> </ul>	<p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ Triggered Current In</li> <li>▪ Digital Pulse Capture</li> <li>▪ Wavetable Digital Out</li> <li>▪ Digital Incremental Encoder Out</li> </ul>	-	<ul style="list-style-type: none"> <li>▪ Integrating virtual ECUs (V-ECUs) in executable applications</li> <li>▪ Adding FPGA applications created with the RTI FPGA Programming Blockset to the signal chain</li> <li>▪ Failure simulation: Support of multiple failure scenarios</li> <li>▪ SYNECT server connection</li> </ul>
4.3 dSPACE Release 7.4	-	-	-	<ul style="list-style-type: none"> <li>▪ New software architecture</li> <li>▪ Properties Browser</li> <li>▪ Conflicts Viewer</li> <li>▪ Modeling executable applications and tasks</li> <li>▪ Support of Simulink SLX model format</li> </ul>
4.2 dSPACE Release 7.3	-	<p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ Power Simulation Control</li> </ul>	-	Implementing multimodel applications and building multicore real-time application that are executed in parallel on single processor cores
4.1 dSPACE Release 7.2	-	<p>New:</p> <ul style="list-style-type: none"> <li>▪ SENT Out</li> <li>▪ Ethernet custom function block types</li> </ul> <p>Enhanced:</p> <ul style="list-style-type: none"> <li>▪ SENT In</li> </ul>	-	Enhanced signal chain handling

ConfigurationDesk Version	New Supported Hardware	New and Enhanced I/O Function Block Types	Bus Manager	Other Features
		<ul style="list-style-type: none"><li>▪ Wavetable Voltage Out</li><li>▪ Wheelspeed Out</li></ul>		
4.0 dSPACE Release 7.1	ConfigurationDesk was introduced to implement real-time applications for the SCALEXIO system.			

# Compatibility Information

## Compatibility of ConfigurationDesk 6.7

### Compatibility in general

The products for working with dSPACE real-time hardware must be compatible. This is guaranteed only for products delivered with the same dSPACE Release. For more information, contact dSPACE.

### Supported SIC file versions

ConfigurationDesk 6.7 supports SIC file versions as listed below:

SIC Files Created With ...	MATLAB Release
dSPACE Release 2021-A: ▪ Model Interface Package for Simulink 4.5	R2021a, R2020b, R2020a, R2019b
dSPACE Release 2020-B: ▪ Model Interface Package for Simulink 4.4 ▪ TargetLink 5.1	R2020b, R2020a, R2019b, R2019a
dSPACE Release 2020-A: ▪ Model Interface Package for Simulink 4.3	R2020a, R2019b, R2019a, R2018b
dSPACE Release 2019-B: ▪ Model Interface Package for Simulink 4.2 ▪ TargetLink 5.0	R2019b, R2019a, R2018b, R2018a
dSPACE Release 2019-A: ▪ Model Interface Package for Simulink 4.1	R2017b
dSPACE Release 2018-B: ▪ Model Interface Package for Simulink 4.0	R2017a
dSPACE Release 2018-A: ▪ Model Interface Package for Simulink 3.6	R2016b
dSPACE Release 2017-B: ▪ Model Interface Package for Simulink 3.5	R2016a
dSPACE Release 2017-A: ▪ Model Interface Package for Simulink 3.4	R2015b

### Note

You can use only SIC files that were generated with the Model Interface Package for Simulink for the `dsrt.tlc` system target file to build real-time applications with ConfigurationDesk. SIC files generated for the `dsrt64.tlc` system target file can be used in ConfigurationDesk only to generate BSC files for VEOS running on a Linux operating system.

### Limitations for earlier SIC file versions in ConfigurationDesk

**scenarios** SIC files created with the Model Interface Package for Simulink version 3.4 ... 4.1 are not supported in the following ConfigurationDesk scenarios:

- In multimodel application processes.
- For building real-time applications that use Real-Time Testing.

### Limitations for Simulink behavior models underlying earlier SIC files

**versions** The following limitations apply to Simulink behavior models underlying SIC files created with the Model Interface Package for Simulink version 3.4 ... 4.1:

- The Simulink behavior model must not contain blocks from the following blocksets:
  - Blocks of ASM  
Real-time applications that contain such SIC files cannot be used with ModelDesk.
  - RTI FPGA Programming Blockset
  - MotionDesk Blockset  
Real-time applications that contain such SIC files cannot be used with MotionDesk.
  - Blocks of any dSPACE Solution.
- SIC files containing compiled objects that are not compatible with the target platform are not supported. In this case, it is not possible to use compiled objects, e.g., for IP protection scenarios.

### Supported V-ECU implementation container versions

The following table shows the tool versions that export supported V-ECU implementation containers, and the related container versions:

V-ECU Implementations Created With...	V-ECU Implementation Version
dSPACE Release 2021-A: ▪ SystemDesk 5.5	3.0
dSPACE Release 2020-B: ▪ SystemDesk 5.5 ▪ TargetLink 5.1	3.0 As of version 3.0, VECU is the new V-ECU implementation file format.
dSPACE Release 2020-A: ▪ SystemDesk 5.4	2.10
dSPACE Release 2019-B: ▪ SystemDesk 5.4 ▪ TargetLink 5.0	2.10

### Compatibility of FMUs

ConfigurationDesk supports Functional Mock-up Units (FMUs) that comply with the FMI 2.0 standard. You might be able to add an FMU based on other versions of the FMI standard to your ConfigurationDesk application. If you do, errors might occur during the build process. For detailed and up-to-date compatibility information on FMI support in ConfigurationDesk, refer to the following website: <http://www.dspace.com/go/FMI-Compatibility>.

**Compatible BSC files**

ConfigurationDesk supports BSC files of version 1.10.

**Compatible EIC files**

The following table shows the compatibility between ConfigurationDesk 6.7 and ECU interface container files (EIC files):

<b>EIC Files Created With ...</b>	<b>EIC Version</b>
dSPACE Release 2021-A (ECU Interface Manager 2.9)	4.0.0
dSPACE Release 2020-B (ECU Interface Manager 2.8)	4.0.0
dSPACE Release 2020-A (ECU Interface Manager 2.7)	4.0.0
dSPACE Release 2019-B (ECU Interface Manager 2.6)	4.0.0
dSPACE Release 2019-A (ECU Interface Manager 2.5)	3.0.0
dSPACE Release 2018-B (ECU Interface Manager 2.4)	3.0.0
dSPACE Release 2018-A (ECU Interface Manager 2.3)	2.0.0
dSPACE Release 2017-B (ECU Interface Manager 2.2)	1.0.0
dSPACE Release 2017-A (ECU Interface Manager 2.1)	1.0.0
dSPACE Release 2016-B (ECU Interface Manager 2.0p1)	1.0.0

**Note**

ECU interfacing with a MicroAutoBox III is supported only as of EIC file version 4.0.0.

**Firmware compatibility**

Firmware is downward-compatible. You cannot use the firmware from a dSPACE Release if the *firmware is of an older dSPACE Release* than the dSPACE Release with which the real-time application was built.

**Tip**

ConfigurationDesk's Platform Manager shows you if the firmware of registered hardware is incompatible and guides you through the update process. Refer to [How to Update or Repair SCALEXIO or MicroAutoBox III Firmware via ConfigurationDesk](#) on page 219.

# Required Licenses

<b>Objective</b>	ConfigurationDesk is license-protected.
------------------	---

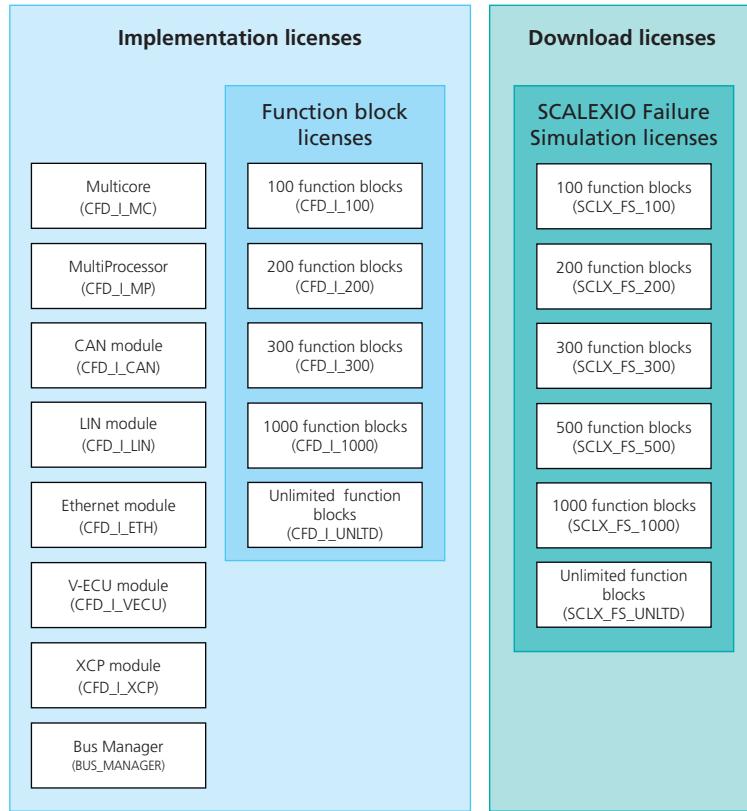
<b>Where to go from here</b>	<b>Information in this section</b>
------------------------------	------------------------------------

Overview of Licenses.....	40
Details on Function Block Licenses.....	42
How to Show Accessible and Used Licenses.....	43
Details on SCALEXIO FIU Licenses.....	44
Notes on Using Floating Network Licenses.....	44

## Overview of Licenses

### Overview

These are the available licenses for implementing a ConfigurationDesk application and downloading real-time applications:



**Implementation licenses** Implementation licenses are required for using the license-protected features of ConfigurationDesk. The following licenses for implementation tasks are available:

- A function block license (CFD\_I\_xxx) works as a basic license and is required to use the basic implementation features of ConfigurationDesk. They are available in different sizes for different numbers of instantiated function blocks in your ConfigurationDesk application. For details, refer to [Details on Function Block Licenses](#) on page 42.
- If you want to implement a multimodel application and to build a multicore real-time application, you additionally need the MC implementation license (CFD\_I\_MC).
- If you want to implement a multimodel application and to build a multi-processing-unit (multi-PU) application, you additionally need the MP implementation license (CFD\_I\_MP). Note, that the CFD\_I\_MC license is also required for multi-PU applications.
- If you want to implement CAN bus communication you additionally need the CAN module license (CFD\_I\_CAN).

This license is required in the following cases:

- You want to implement CAN communication via the RTI CAN MultiMessage Blockset.
- You add a V-ECU implementation to your applications that contains CAN bus communication.
- You use ConfigurationDesk's Bus Manager and you want to implement applications which contain at least one assigned CAN cluster.
- If you want to implement LIN bus communication you additionally need the LIN module license (CFD\_I\_LIN).

This license is required in the following cases:

- You want to implement LIN communication via the RTI LIN MultiMessage Blockset.
- You add a V-ECU implementation to your applications that contains LIN bus communication.
- You use ConfigurationDesk's Bus Manager and you want to implement applications which contain at least one assigned LIN cluster.
- If you want to implement a function block that provides access to an Ethernet hardware resource (for example, the Ethernet Setup function block), you need the Ethernet module license (CFD\_I\_ETH).
- If you want to work with V-ECU implementations in your ConfigurationDesk application, you need the V-ECU module license (CFD\_I\_VECU).
- If you want to configure an XCP service for accessing Simulink models or V-ECUs via A2L/XCP, you need the XCP module license (CFD\_I\_XCP).
- If you want to implement bus communication via ConfigurationDesk's Bus Manager you additionally need the Bus Manager license (BUS\_MANAGER).

**Download licenses** The download licenses must be available on the host PC with dSPACE software that you use to download and handle the real-time application. For example, this also can be a PC on which ControlDesk is installed. At least one SCALEXIO Failure Simulation license (SCALEXIO\_FIU\_xxx) is required to use failure simulation of a SCALEXIO system. For more information, refer to [Details on SCALEXIO FIU Licenses](#) on page 44.

## Enabling licenses

Two different license types (single-user license, floating network license) are available for working with ConfigurationDesk depending on your order:

- Single-user licenses: With single-user licenses, the dSPACE software is protected by a license mechanism based on a dongle. The dongle must be connected to your host PC to use the license-protected features of ConfigurationDesk.
- Floating network licenses: With floating network licenses, the dSPACE License Client automatically requests the license required by a particular action from the connected dSPACE License Server (automatic activation). If there is at least one instance of each required license available, you can continue working with the dSPACE software as usual. At the same time, the license is blocked for other clients.

<b>Invalid licenses</b>	A license becomes invalid when the dongle is removed from the PC or the required license is not available on the dSPACE License Server, for example. If you now try to carry out a license-protected action, an error message is displayed. However, you can save the ConfigurationDesk application.
-------------------------	--

## Details on Function Block Licenses

---

<b>Objective</b>	At least one function block license is required to implement and use function blocks in a ConfigurationDesk application.
------------------	--

<b>Not affected function block types</b>	Function block types providing bus communication are not protected by function block licenses. The corresponding instances can be implemented and used in a ConfigurationDesk application without having such a license. This applies to the following function blocks: <ul style="list-style-type: none"><li>▪ CAN function block</li><li>▪ LIN function block</li><li>▪ FlexRay function block</li><li>▪ Bus Configuration function block</li><li>▪ Ethernet Setup function block</li></ul>
--	---

### Note

However, you need other specific implementation licenses to implement and use the above listed function blocks in your ConfigurationDesk application. Refer to [Overview of Licenses](#) on page 40.

<b>Handling function block licenses</b>	The function block licenses are available in different sizes for different numbers of instantiated function blocks in your active ConfigurationDesk application. There are licenses that allow you to implement 100, 200, 300, 1000 or an unlimited number of function blocks.  ConfigurationDesk counts the instantiated function blocks in the signal chain of your active application at run time.
---	---

### Note

The size of the function block license affects only your active ConfigurationDesk application.

You can open an application that contains more function blocks than allowed by the function block license. In this case, using ConfigurationDesk is restricted: For example, if you try to add function blocks to the ConfigurationDesk application

or you try to start the build process, an error message is displayed. However, deleting elements of the application and saving the application is possible.

### Tip

If required, you can purchase larger function block licenses later.

## How to Show Accessible and Used Licenses

### Objective

ConfigurationDesk provides a license overview, which shows available licenses, accessible licenses and the licenses which are used in your active ConfigurationDesk application.

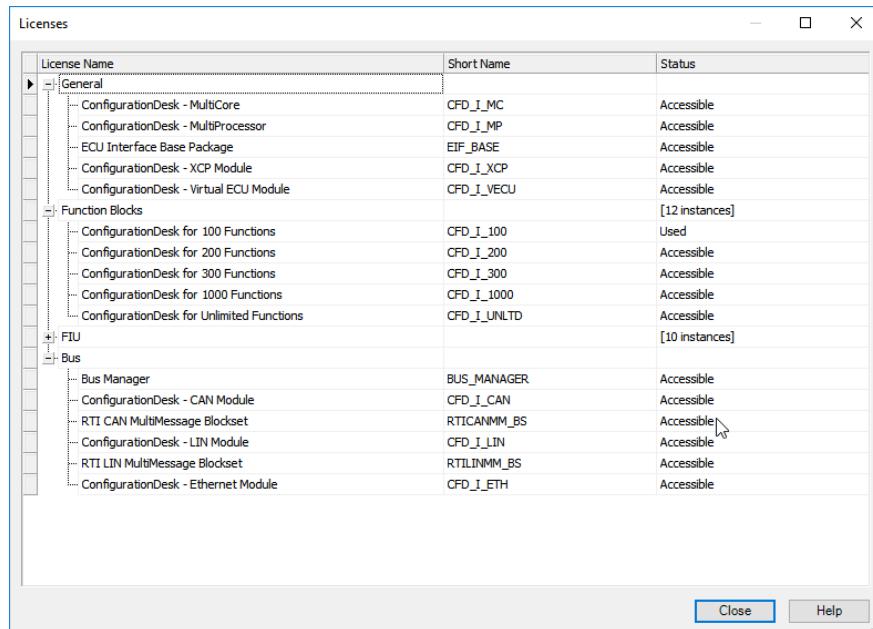
### Method

#### To display accessible and used licenses

- 1 On the File ribbon, click Licenses.

### Result

ConfigurationDesk opens the Licenses dialog.



- A license indicated as accessible can be used for your ConfigurationDesk projects.
- A license indicated as used, is currently used by your active ConfigurationDesk application. For function block licenses and the SCALEXIO FIU licenses, the number of instances is displayed.

## Details on SCALEXIO FIU Licenses

<b>Objective</b>	The Failure Simulation Units installed in the SCALEXIO simulator are license-protected. To enable the failure simulation feature of a SCALEXIO system, a SCALEXIO FIU license must be available on the host PC which you use for downloading and handling the real-time application.
<b>Notes on using SCALEXIO FIU licenses</b>	<p>The license is checked when the real-time application is downloaded. If no license is available, a warning is given and subsequent failure simulation on this downloaded real-time application is not possible. However, the real-time application is downloaded anyway and can be handled from within the host PC.</p> <p>The licenses are available in different sizes for different numbers of instantiated function blocks in your ConfigurationDesk application.</p>
<p><b>Note</b></p> <p>For FIU licenses only those function blocks are relevant, that are assigned to a hardware resource providing failure simulation support. For example, function blocks that are assigned to channel sets of the DS6101 and DS6201 boards are not counted, because these boards generally do not support failure simulation.</p>	

## Notes on Using Floating Network Licenses

<b>Objective</b>	When you use ConfigurationDesk with floating network licenses, the dSPACE License Client requests the required licenses from the connected dSPACE License Server.
<b>Details on the license behavior</b>	<p>One function block implementation license (CFD_I_xxx) is blocked when a license-protected action is performed in your active ConfigurationDesk application. All licenses are released immediately when ConfigurationDesk is closed again.</p> <p>The required function block license is calculated at run time from the number of instantiated function blocks in the signal chain of your active ConfigurationDesk application and blocked.</p> <p>If the dSPACE License Server provides function block licenses in different sizes, ConfigurationDesk blocks the smallest function block license that covers the number of instantiated function blocks in the signal chain. However, if you block</p>

a large function block license and you reduce the number of instantiated function blocks in your application, the larger license is not released.

# Migrating From Prior Versions of ConfigurationDesk

## Basics on Migrating From Prior Versions of ConfigurationDesk

### Opening projects created with previous ConfigurationDesk versions

You can open a project created with a previous ConfigurationDesk version higher than 4.2 in the same way as a project from the current ConfigurationDesk version. ConfigurationDesk automatically migrates the project when opening it.

#### Note

- As of dSPACE Release 2021-A, ConfigurationDesk supports the direct import only of projects last saved with one of the previous seven ConfigurationDesk versions.
- After you have migrated and saved a project, you can no longer open it with a previous ConfigurationDesk version.

### Version-specific migration steps

Starting with the ConfigurationDesk version that you migrate from you have to iterate the migration steps for each product version until the current version.

For the migration from the last ConfigurationDesk version (6.6) to the current ConfigurationDesk version (6.7), refer to [Migrating to ConfigurationDesk 6.7 \(New Features and Migration\)](#).

The following table provides an overview of migration steps from prior ConfigurationDesk versions.

Migrating From ConfigurationDesk Version ... to ...	Overview of Changes That Affect Migration	For Details, Refer to ...
6.5 to 6.6	<ul style="list-style-type: none"> <li>▪ Migrating to the Linux-based operating system on SCALEXIO</li> <li>▪ Project import restricted as of Release 2021-A</li> <li>▪ Changes to the I/O Ethernet communication</li> </ul>	<a href="#">Migrating from ConfigurationDesk 6.5 to 6.6 on page 820</a>
6.4 to 6.5	–	–
6.3 to 6.4	<ul style="list-style-type: none"> <li>▪ Changed naming scheme for bus access requests (available for Bus Manager elements)</li> </ul>	<a href="#">Migrating from ConfigurationDesk 6.3 to 6.4 on page 821</a>
6.2 to 6.3	–	–
6.1 to 6.2	<ul style="list-style-type: none"> <li>▪ Discontinuation of Python 2.7</li> <li>▪ Discontinuation of SCALEXIO Ethernet Solution</li> <li>▪ FPGA custom function blocks with APU functionality</li> </ul>	<a href="#">Migrating from ConfigurationDesk 6.1 to 6.2 on page 822</a>
6.0 to 6.1	<ul style="list-style-type: none"> <li>▪ Changes regarding the Bus Manager</li> </ul>	<a href="#">Migrating from ConfigurationDesk 6.0 to 6.1 on page 823</a>

Migrating From ConfigurationDesk Version ... to ...	Overview of Changes That Affect Migration	For Details, Refer to ...
5.7 to 6.0	<ul style="list-style-type: none"> <li>▪ Runnable functions from projects created with ConfigurationDesk 5.7 or earlier</li> <li>▪ Propagating changes to Simulink</li> <li>▪ ConfigurationDesk projects containing FlexRay communication</li> <li>▪ Changes in TRC file for Bus Manager elements</li> </ul>	<a href="#">Migrating from ConfigurationDesk 5.7 to 6.0 on page 823</a>
5.6 to 5.7	<ul style="list-style-type: none"> <li>▪ Changes in TRC file for Bus Manager elements</li> <li>▪ Discontinuation of SYNECT server connection</li> </ul>	<a href="#">Migrating from ConfigurationDesk 5.6 to 5.7 on page 825</a>
5.5 to 5.6	<ul style="list-style-type: none"> <li>▪ Inconsistency Ethernet adapters</li> </ul>	<a href="#">Migrating from ConfigurationDesk 5.5 to 5.6 on page 825</a>
5.4 to 5.5	<ul style="list-style-type: none"> <li>▪ Possible M script adjustments for automation</li> <li>▪ Hardware topology containing a DS2671 Bus Board</li> <li>▪ Discontinuation of platform management automation API version 1.0 as of dSPACE Release 2016-B</li> </ul>	<a href="#">Migrating from ConfigurationDesk 5.4 to 5.5 on page 825</a>
5.3 to 5.4	<ul style="list-style-type: none"> <li>▪ Changes in TRC file generation</li> </ul>	<a href="#">Migrating from ConfigurationDesk 5.3 to 5.4 on page 826</a>
5.2 to 5.3	<ul style="list-style-type: none"> <li>▪ New XML schema for custom function blocks</li> <li>▪ Using parallel build for referenced models</li> </ul>	<a href="#">Migrating from ConfigurationDesk 5.2 to 5.3 on page 826</a>
5.1 to 5.2	<ul style="list-style-type: none"> <li>▪ Pre-4.3 projects and component files no longer supported</li> <li>▪ Modifications concerning the TRC file generation</li> <li>▪ Handling of Bus Selector block changed</li> <li>▪ Generated code from Simulink Coder has changed</li> <li>▪ Changed custom function path in variable description file</li> </ul>	<a href="#">Migrating from ConfigurationDesk 5.1 to 5.2 on page 827</a>
5.0 to 5.1	–	–
4.4 to 5.0	<ul style="list-style-type: none"> <li>▪ Migrating a custom function block from ConfigurationDesk 4.4</li> <li>▪ Migration of SYNECT database</li> </ul>	<a href="#">Migrating from ConfigurationDesk 4.4 to 5.0 on page 829</a>
4.2 or earlier to 4.3 or 4.4	With ConfigurationDesk 4.3 a new software architecture was introduced.	<a href="#">Migrating Projects from ConfigurationDesk 4.2 (on dSPACE Release 7.3) or Earlier to 4.3 or 4.4 on page 830</a>

#### Migrating tool automation scripts

If you use scripts for tool automation with ConfigurationDesk's automation API there are also a number of version-specific migration steps to consider. Refer to [Version-Specific Migration Steps for ConfigurationDesk Tool Automation \(ConfigurationDesk Automating Tool Handling !\[\]\(d24c3affefeb42bd070edd596d3c9a41\_img.jpg\).](#)

# Starting ConfigurationDesk

---

<b>Objective</b>	ConfigurationDesk is started in the same way as other Windows-based software. However, there are some points to note.
------------------	---

## Notes for Starting ConfigurationDesk

---

<b>Prerequisite for starting</b>	If ConfigurationDesk is available in the Windows Start menu, you can start it as usual. If it is not available in the Start menu, the reason might be that ConfigurationDesk is not part of your active RCP and HIL installation on your host PC. In that case you have to activate the recommended installation first. For details, see below.
<b>Handling dSPACE installations</b>	You can have different dSPACE installations on your host PC. A dSPACE installation is all the dSPACE software products installed in the same installation folder: <ul style="list-style-type: none"><li>▪ If there is more than one RCP and HIL installation on your host PC, only one installation is the active one. You can access only the software belonging to the active installation.</li><li>▪ If there is only one RCP and HIL installation on your host PC, this is always the active installation.</li></ul> Installations are activated via dSPACE Installation Manager. You need administrator rights to do this. For detailed information, refer to <a href="#">Activating and Deactivating dSPACE Installations (Managing dSPACE Software Installations)</a> .
<b>Starting ConfigurationDesk</b>	You start ConfigurationDesk as follows: From the Start menu, select dSPACE RCP and HIL <dSPACE Release> – dSPACE ConfigurationDesk <x.y>.
<b>First steps</b>	When you have started ConfigurationDesk, you can: <ul style="list-style-type: none"><li>▪ Adapt the Screen Layout to your needs as described in <a href="#">Customizing View Sets</a> on page 57.</li><li>▪ Do a tutorial to learn how to work with ConfigurationDesk:<ul style="list-style-type: none"><li>▪ <a href="#">ConfigurationDesk Tutorial Starting with External Devices</a></li><li>▪ <a href="#">ConfigurationDesk Tutorial Starting with Simulink</a></li><li>▪ <a href="#">ConfigurationDesk Tutorial MicroAutoBox III</a></li></ul></li><li>▪ Start to work with one of the demo applications described in <a href="#">Overview of ConfigurationDesk Demo Projects (ConfigurationDesk Demo Projects)</a>.</li></ul>

- Make yourself familiar with the workflows in ConfigurationDesk as described in [Typical Workflows for Beginners](#) on page 69.

# User Interface of ConfigurationDesk

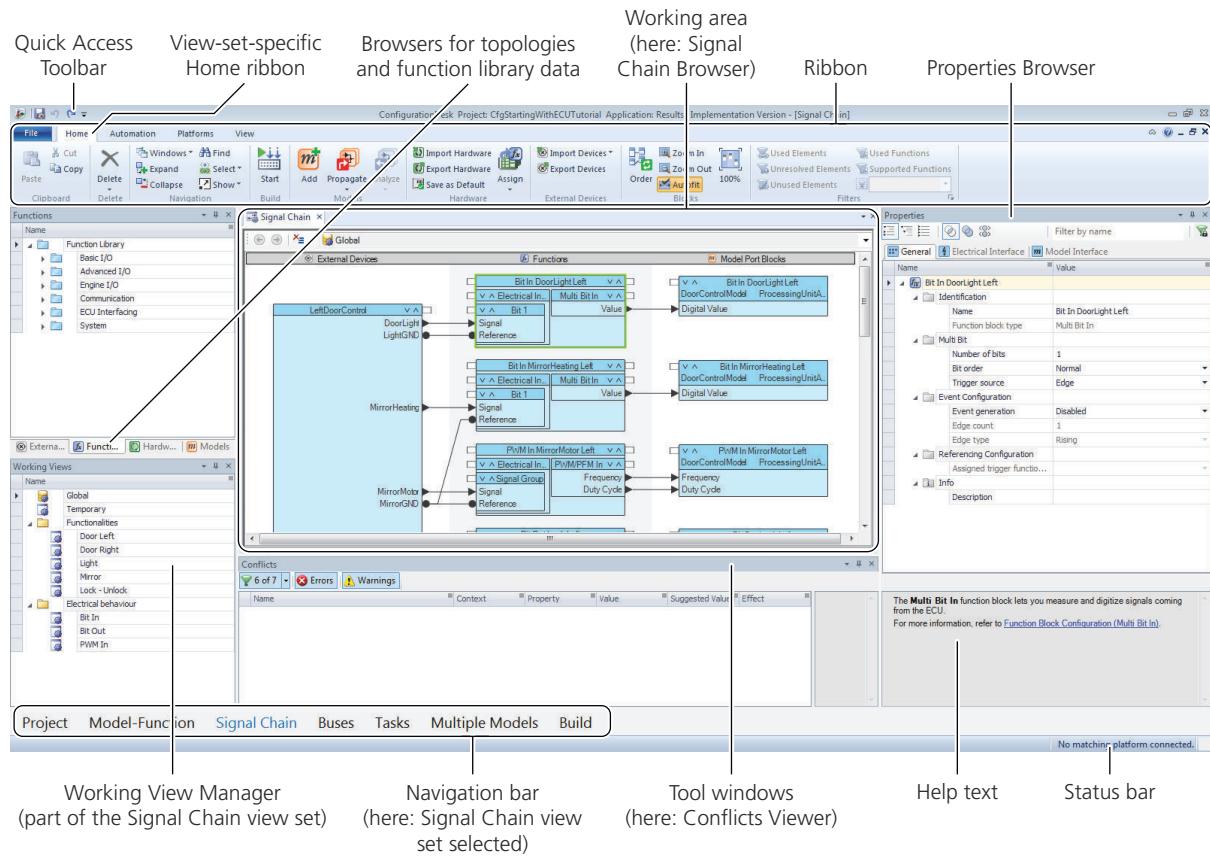
<b>Objective</b>	ConfigurationDesk's user interface consists of various elements. They can be customized to facilitate your work with ConfigurationDesk.
------------------	---

Where to go from here	Information in this section
	<a href="#">Overview of the User Interface of ConfigurationDesk.....</a> 50
	<a href="#">Basics on Ribbons.....</a> 54
	<a href="#">Customizing View Sets.....</a> 57
	<a href="#">Using Keyboard Shortcuts.....</a> 63
	<a href="#">How to Move Panes to Different Positions.....</a> 64
	<a href="#">How to Customize the Quick Access Toolbar.....</a> 65

## Overview of the User Interface of ConfigurationDesk

<b>Switching between view sets for specific purposes</b>	The user interface of ConfigurationDesk consists of different panes that are arranged in view sets. You can switch between view sets by using the navigation bar. The available panes of each view set serve a specific purpose. For example, the Project view set contains the Project Manager to perform project and application management tasks.  The order of view sets from left to right on the navigation bar represents the workflow for implementing a real-time application. The Project and Build view sets are the start and end points for all use scenarios. The other view sets are suitable for specific use scenarios.
--	--

The basic structure of view sets is similar. For example, the Signal Chain view set is structured as shown in the following illustration:



#### Permanent interface elements

Some interface elements are permanently available irrespective of the currently active view set.

**Ribbon and Quick Access Toolbar** The ribbon and the Quick Access Toolbar are always available at the top of the user interface. But for each view set, the Home ribbon contains specific commands suitable for the purpose of the view set. For more information on handling ribbons and the Quick Access Toolbar, refer to [Basics on Ribbons](#) on page 54.

**Status bar** The status bar displays the state of an active application in relation to registered hardware platforms. For details on the application states, refer to [Handling ConfigurationDesk Projects and Applications](#) on page 86.

## Available view sets and their purposes

View Set	Purpose	Panes <sup>1)</sup>	Further Information
Project	To manage ConfigurationDesk projects and applications. You need a project with application data to perform any task in ConfigurationDesk. For some project management tasks, you are automatically referred to the backstage view (File ribbon).	<ul style="list-style-type: none"> <li>▪ Project Manager</li> <li>▪ <i>If no project is currently open:</i> Start Page</li> </ul>	<a href="#">Managing ConfigurationDesk Projects and Applications on page 85</a>
Model-Function	To implement the connection between a behavior model and ConfigurationDesk's I/O functionality.	<ul style="list-style-type: none"> <li>▪ Model-Function Mapping Browser</li> <li>▪ Function Browser</li> <li>▪ Hardware Resource Browser</li> <li>▪ Properties Browser</li> <li>▪ Conflicts Viewer</li> </ul>	<a href="#">User-Friendly Connection of ConfigurationDesk and Simulink Models on page 553</a>
Signal Chain	To implement the complete logical signal chain between an external device and a behavior model.	<ul style="list-style-type: none"> <li>▪ Signal Chain Browser</li> <li>▪ Working View Manager</li> <li>▪ External Device Browser</li> <li>▪ Function Browser</li> <li>▪ Hardware Resource Browser</li> <li>▪ Model Browser</li> <li>▪ Properties Browser</li> <li>▪ Conflicts Viewer</li> </ul>	<a href="#">Handling the Signal Chain in Working Views on page 169</a>
Buses	To implement bus communication based on communication matrices.	<ul style="list-style-type: none"> <li>▪ Bus Configurations table</li> <li>▪ Bus Simulation Features table</li> <li>▪ Bus Inspection Features table</li> <li>▪ Bus Manipulation Features table</li> <li>▪ Bus Access Requests table</li> <li>▪ Bus Configuration Function Ports table</li> <li>▪ Buses Browser</li> <li>▪ Model Browser</li> <li>▪ Properties Browser</li> <li>▪ Conflicts Viewer</li> <li>▪ Message Viewer</li> </ul>	<a href="#">ConfigurationDesk Bus Manager Implementation Guide</a>
Tasks	To model and configure the tasks and application processes of your executable application.	<ul style="list-style-type: none"> <li>▪ Task Configuration table</li> <li>▪ Properties Browser</li> <li>▪ Conflicts Viewer</li> </ul>	<a href="#">Modeling Executable Applications and Tasks on page 471</a>
Multiple Models	To configure the communication between multiple behavior models.	<ul style="list-style-type: none"> <li>▪ Model Communication Browser</li> <li>▪ Model Communication Package table</li> <li>▪ Processing Resource Assignment table</li> <li>▪ Working View Manager</li> <li>▪ Model Browser</li> </ul>	<a href="#">Setting Up Model Communication on page 448</a>

View Set	Purpose	Panes <sup>1)</sup>	Further Information
		<ul style="list-style-type: none"> <li>▪ Properties Browser</li> <li>▪ Conflicts Viewer</li> </ul>	
Build	<ul style="list-style-type: none"> <li>▪ To configure the build process and build the real-time application.</li> <li>▪ To download the real-time application to dSPACE hardware.</li> </ul>	<ul style="list-style-type: none"> <li>▪ Build Configuration table</li> <li>▪ Build Log Viewer</li> <li>▪ Properties Browser</li> <li>▪ Platform Manager</li> <li>▪ Conflicts Viewer</li> </ul>	<ul style="list-style-type: none"> <li>▪ <a href="#">Building Real-Time Applications</a> on page 697</li> <li>▪ <a href="#">Downloading and Executing Real-Time Applications</a> on page 745</li> </ul>

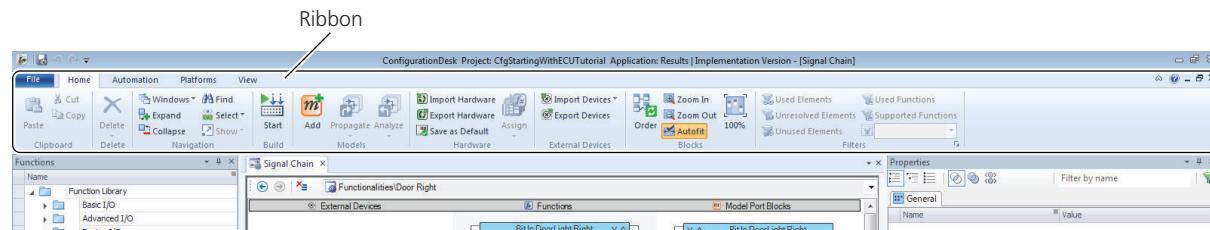
<sup>1)</sup> Default state after installing ConfigurationDesk.

<b>Common panes available in most view sets</b>	<p>Some panes are available and required in most view sets.</p> <p><b>Properties Browser</b> Lets you configure the properties of selected elements. Refer to <a href="#">Configuring Elements with the Properties Browser</a> on page 131.</p> <p><b>Conflicts Viewer</b> Displays configuration conflicts in your ConfigurationDesk application. You can resolve most of the conflicts here. Refer to <a href="#">Resolving Conflicts</a> on page 677.</p>	
<b>Additional panes for specific purposes</b>	<p>Some panes are part of few or no view sets but may still be required for specific purposes.</p> <p><b>Message Viewer</b> Provides a history of all the info, advice, error, and warning messages, and all the questions that occur when you work with the product. Refer to <a href="#">Message Viewer (ConfigurationDesk User Interface Reference)</a>.</p> <p><b>Find Results Viewer</b> Displays the results of searches you performed by using the Find command. Refer to <a href="#">How to Find Elements of a ConfigurationDesk Application</a> on page 129.</p> <p><b>Interpreter</b> Handles Python commands and scripts for automating ConfigurationDesk. Refer to <a href="#">ConfigurationDesk Automating Tool Handling</a>.</p>	
<b>Customizing view sets</b>	<p>If the available view sets do not meet your requirements, you can customize them or create additional view sets with the panes of your choice. Refer to <a href="#">Customizing View Sets</a> on page 57.</p>	
<b>Related topics</b>	<p>Basics</p> <table border="1" style="margin-top: 10px;"> <tr> <td style="padding: 5px;"> <a href="#">Basics on Ribbons</a>.....54  <a href="#">Customizing View Sets</a>.....57                 </td></tr> </table>	<a href="#">Basics on Ribbons</a> .....54 <a href="#">Customizing View Sets</a> .....57
<a href="#">Basics on Ribbons</a> .....54 <a href="#">Customizing View Sets</a> .....57		

## Basics on Ribbons

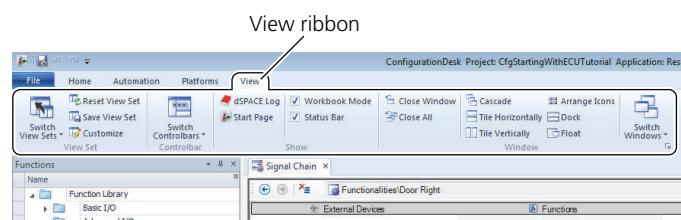
### Ribbon

The ribbon organizes and groups commands to make them easily available. The ribbon is located at the top of the user interface, see the following example.



Depending on the current context or the selected elements, some ribbon commands are grayed out and not available.

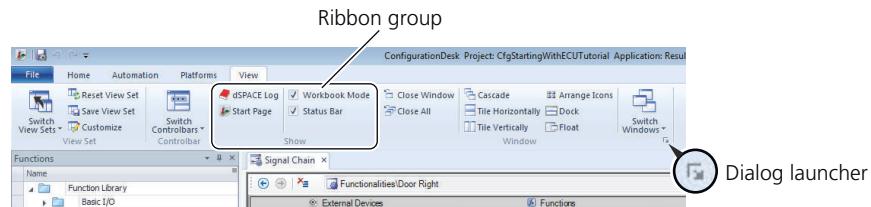
The ribbon consists of several tabbed ribbons, see the following example of a View ribbon.



### Ribbon group

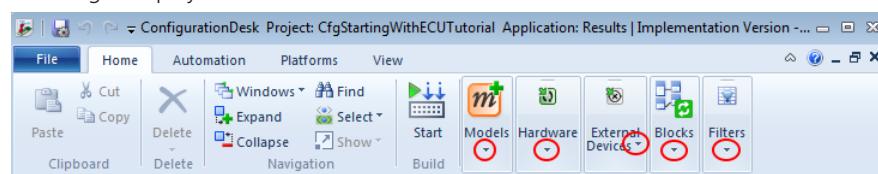
A ribbon group is a part of a tabbed ribbon.

A ribbon group displays a set of related commands.



**Dialog launcher** A dialog launcher is an optional element of a ribbon group that lets you open a dialog related to that ribbon group.

**Collapsed ribbon groups** If you minimize the ConfigurationDesk window, some ribbon groups might be collapsed so that the different commands are no longer displayed. Small arrows indicate that the commands are available.



Click a ribbon group to make its commands available again.

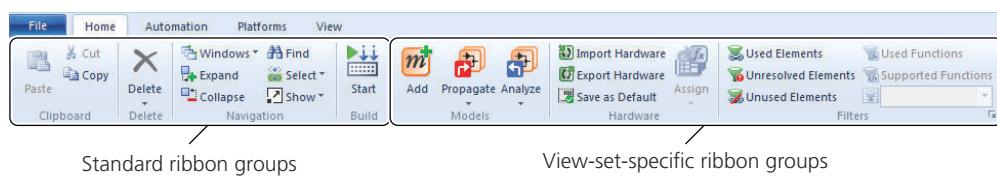


### View-set-specific Home ribbon

For each view set, the Home ribbon contains specific commands suitable for the purpose of the view set. The Home ribbon consists of two parts:

- To the left, there are a number of standard ribbon groups that are available in each view set.
- To the right, there are a number of specific ribbon groups that are available in some view sets or only in one view set.

The following illustration shows the view-set-specific Home ribbon in the Model-Function view set:



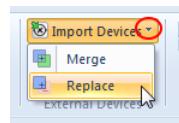
#### Note

Some ribbon groups on the view-set-specific Home ribbon are associated with a specific pane and are removed from the Home ribbon if you close this pane. For example, the External Devices ribbon group in the Signal Chain view set is removed from the Home ribbon if you close the External Device Browser.

### Submenus and split buttons

Most ribbon commands are executed simply by clicking them, but there are some specific types of ribbon commands such as submenus and split buttons.

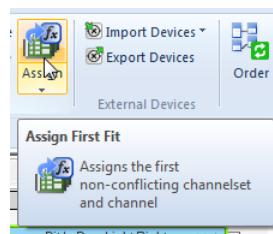
**Submenu** Some ribbon commands open a submenu from which you can select multiple commands. This is indicated by a small arrow to the right of the command.



**Split button** A split button consists of two parts: The main icon part of the button executes a default command from the submenu that is available via the arrow part.



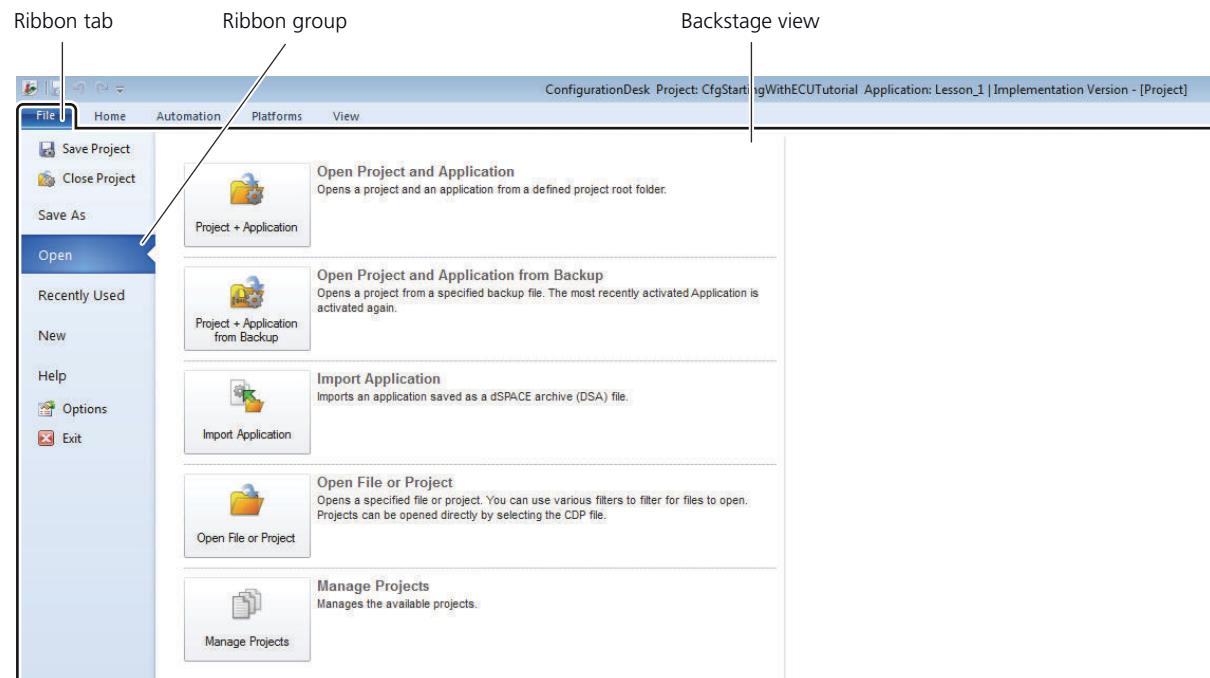
The split button's tooltip shows you the default command.



## Backstage view

Clicking the File ribbon tab opens the backstage view. ConfigurationDesk's backstage view lets you access several project management-related commands and ribbon groups as well as global program options and access to help commands.

The following illustration shows the Open ribbon group of the backstage view as an example.



## Quick Access Toolbar

The Quick Access Toolbar is an easy way to call frequently used commands. It is always available irrespective of the currently active view set.

Quick Access Toolbar



You can customize the Quick Access Toolbar to provide the commands you use most frequently. For instructions, refer to [How to Customize the Quick Access Toolbar](#) on page 65.

## Ribbon navigation

You can navigate the ribbon via mouse and via keyboard.

**Mouse navigation** In addition to clicking ribbon tabs, you can switch between the ribbons of a view set using the mouse scroll wheel.

**Keyboard navigation** If you want to navigate the ribbon via keyboard, press the **Alt** key. Each command in the Quick Access Toolbar and each ribbon tab then is marked by an access key.

The following illustration shows a ribbon after pressing **Alt** as an example.



If you then press one of the ribbon tab access keys, each command in the selected ribbon is also marked by an access key.

You can remove the access keys by pressing **Alt** again.

## Related topics

### Basics

[Overview of the User Interface of ConfigurationDesk](#).....50

### HowTos

[How to Customize the Quick Access Toolbar](#).....65

## Customizing View Sets

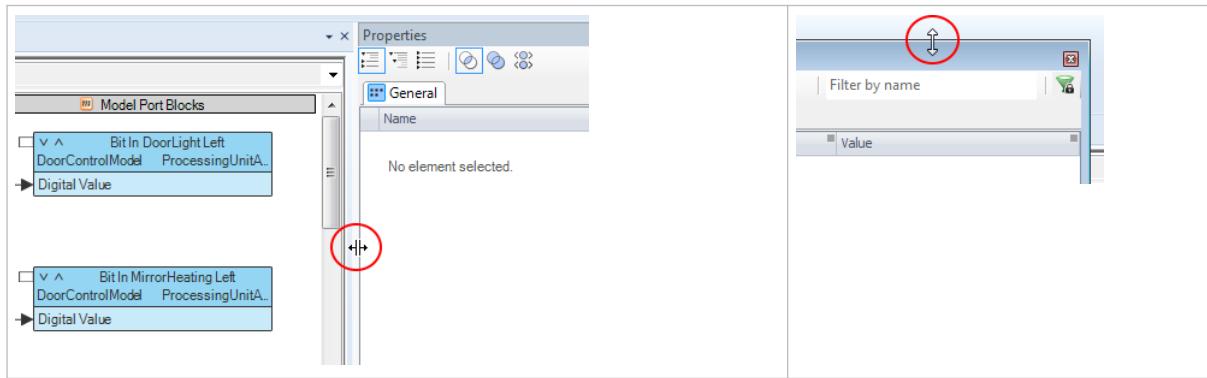
### Adjusting the position and size of panes

There are several ways to change the position and size of panes.

**Moving panes to different positions** You can move a pane to a different position in a view set after changing its docking state. ConfigurationDesk provides various commands, such as the Floating command, to change the docking state of each pane.

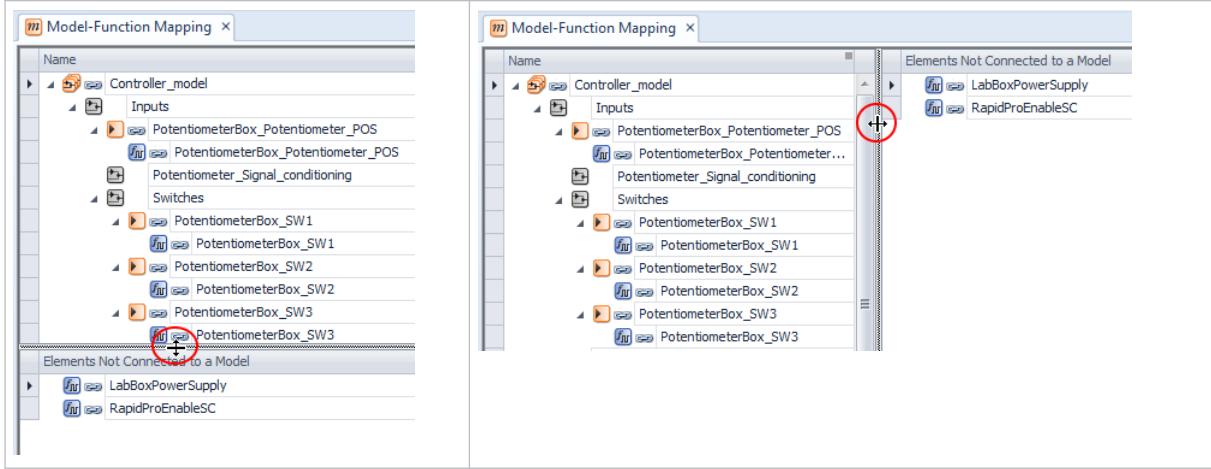
For instructions, refer to [How to Move Panes to Different Positions](#) on page 64.

**Adjusting the height and width** You can adjust the height and width of panes by dragging the double-headed arrow into which the pointer turns when you move it over a split line between docked panes or a border of a floating pane:

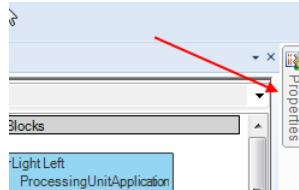


**Panes with multiple areas** Some panes, such as the Model-Function Mapping Browser, consist of areas whose size and position you can change in

the pane. Use the double-headed arrow to change the size of areas. Double-click the split line to change the arrangement of areas.



**Auto-hiding unused panes** You can automatically hide docked panes that you are currently not using by clicking the icon in the title bar. A tab representing the pane is created.



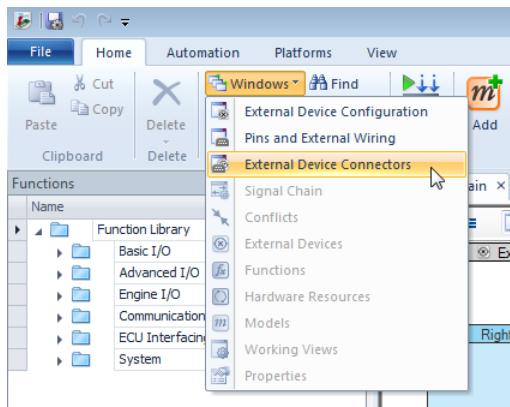
Move the pointer to the tab to temporarily make the pane available again. To disable auto-hide, click the tab or click the icon in the title bar.

**Additional commands for panes in the working area** The View ribbon offers some additional commands for panes in the working area. For descriptions of these commands, refer to [Window Handling \(ConfigurationDesk User Interface Reference\)](#).

## Opening and closing panes

**Opening view-set-specific panes** On the Home ribbon, you can select a pane from the Navigation – Windows list.

The list contains the panes that are open in the current view set by default, plus additional panes that are associated with the view set's purpose. Panes that are already open in the current view set are grayed out and cannot be selected.

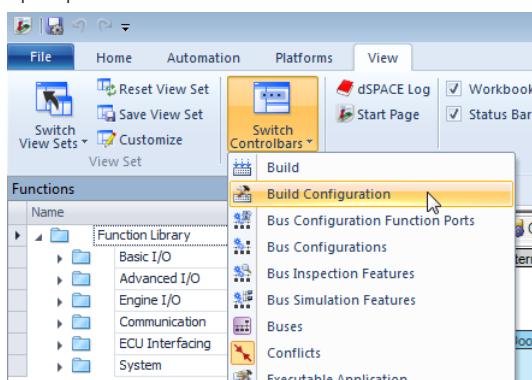


### Tip

This is also the easiest method to reopen a pane of the view set that you accidentally closed.

**Opening any pane** On the View ribbon, you can select a pane from the Controlbar – Switch Controlbars list.

The list contains all panes that are available in ConfigurationDesk. The icons of panes that are already open in the current view set are highlighted. Selecting an open pane closes it.



**Tip**

- Some ribbon groups on the view-set-specific Home ribbon are associated with a specific pane. For example, the External Devices ribbon group is added to the Home ribbon if you open the External Device Browser.
- You can also open selected elements in a specific pane using the Navigation – Show list on the Home ribbon or various Show in ... commands in different context menus. For instructions, refer to [How to Show Elements in a Different Pane](#) on page 125.

**Closing panes** To close a pane, click the  icon. The icon's design might vary depending on the docking state or the position of the pane.

---

**Customizing tables**

Some elements of a ConfigurationDesk application are managed in tables. Tables can be opened and customized like other panes. There are some specific customizing features for table rows and columns. Refer to [Customizing Table Rows and Columns](#) on page 154.

---

**Saving and resetting changes to view sets**

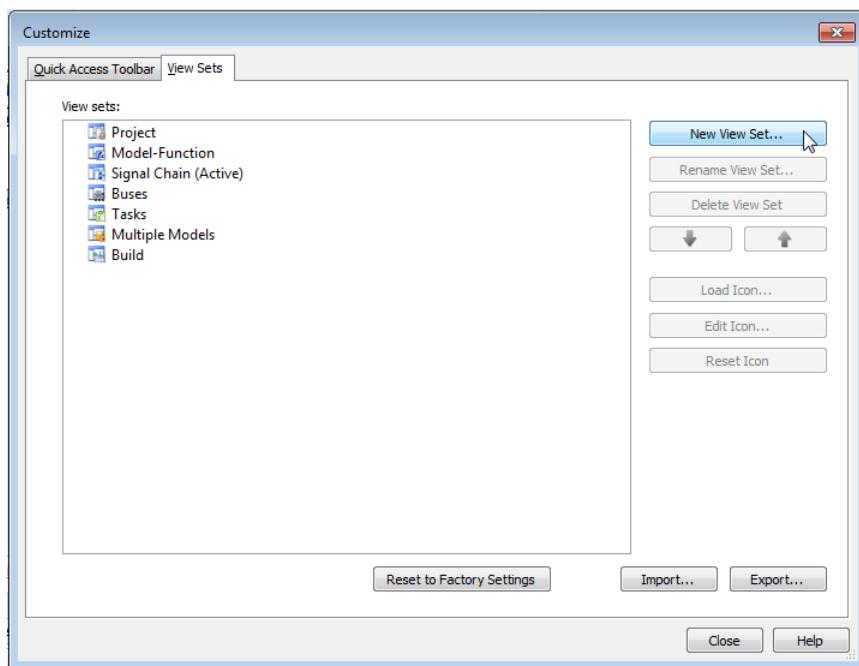
Every change you apply to a view set is automatically saved. This means that ConfigurationDesk keeps the current selection of panes, their position, and their size for a view set if you switch the view set, activate a different ConfigurationDesk application, open a different project, or close and restart ConfigurationDesk.

If you are no longer satisfied with the current customized state of a view set and do not want to manually undo all the changes, you can use the View Set – Reset View Set command on the View ribbon to reset the current view set to its default state. Initially, the default state is equal to the factory settings. If you want to make the current state of the current view set its new default state, use the Save View Set command.

---

**Creating user-defined view sets**

If using and customizing the default view sets does not satisfy your requirements, you can create user-defined view sets on the View Sets page of the Customize dialog. To open the Customize dialog, you can use the View Set – Customize command on the View ribbon, for example.



Click **New View Set** and provide a name for the user-defined view set. You can change the view set's position on the navigation bar by using the up and down arrow buttons. The **Load Icon** or **Edit Icon** buttons let you provide or create a custom icon for the user-defined view set.

The newly created user-defined view set is based on the currently active view set. To change it according to your requirements, you can use all the previously described methods of customization.

**Tip**

Use the **Save View Set** command as soon as you finished customizing your user-defined view set. This allows you to use the **Reset View Set** command to restore the finished state later.

---

**Restoring the factory settings**

To discard all the changes that you made to view sets, you can restore the factory settings. Use the **Reset to Factory Settings** command from the **View Sets** page of the **Customize** dialog to make the user interface look like it did the first time you started ConfigurationDesk.

**Note**

All the user-defined view sets that you created are deleted.

## Exporting and importing view set files

In the Customize dialog, you can export and import view set files using the Export and Import button. This allows you to back up or share user-defined view sets. The exported VSET file contains all view sets and their settings from the current ConfigurationDesk installation.

### Note

To import a view set, the exporting and the importing ConfigurationDesk installation must have the same configurations (such as the available licenses).

## Related topics

### Basics

[Customizing Table Rows and Columns.....](#) 154

### HowTos

[How to Move Panes to Different Positions.....](#) 64

[How to Show Elements in a Different Pane.....](#) 125

## Using Keyboard Shortcuts

### Introduction

You can access many of ConfigurationDesk's panes, dialogs, or commands by using keyboard shortcuts.

### Using standard shortcuts to speed up your work

ConfigurationDesk supports standard shortcuts to speed up your work, such as **Ctrl+Z** and **Ctrl+Y** to undo and redo operations, or **Ctrl+C** and **Ctrl+V** to copy and paste elements.

### Available keyboard shortcuts

**Opening a list of available keyboard shortcuts** To open a list of available keyboard shortcuts, use the Keyboard Help command on the File – Help ribbon, or press **Ctrl+F1** while working in ConfigurationDesk.

**Showing keyboard shortcuts in tooltips** To show the keyboard shortcut for a command in its tooltip (if available), enable the Show shortcut keys in tooltips option on the Views page of the ConfigurationDesk Options dialog.

**Using ribbon shortcuts** You can use the keyboard to navigate ConfigurationDesk's ribbons after pressing the **Alt** key. For details, refer to [Basics on Ribbons](#) on page 54.

**Related topics****Basics**

Available Keyboard Shortcuts..... 814

## How to Move Panes to Different Positions

**Objective**

To move a pane to a different position in a view set.

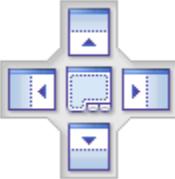
**Method****To move a pane to a different position**

- 1 Right-click the title bar of the pane whose position you want to change. If you want to move a tabbed pane in the working area, you must right-click its tab instead of the title bar.
- 2 Select Floating. For a tabbed pane in the working area, select Float. The docking state of the pane is changed to *floating*. Click the pane's title bar and drag it to a new position. ConfigurationDesk displays *docking stickers* that you can use to specify the new position.

Docking Sticker	Description
	The pane is docked to the top of the view set.
	The pane is docked to the bottom of the view set.
	The pane is docked to the left of the view set.
	The pane is docked to the right of the view set.
	The pane is docked to the top, bottom, left, or right of the working area.

**Tip**

To dock the pane in the working area, use the *Float in Working Area* command.

Docking Sticker	Description
	The pane is docked above, below, to the left, or to the right of a specific pane. If you drag the mouse onto the middle docking sticker, the pane is docked as a new tabbed pane.

- 3 Drag the pane onto a docking sticker. A blue frame indicates its new position.
- 4 Release the left mouse button.

ConfigurationDesk docks the pane in the new position and creates a tab for it, if necessary.

#### Tip

- If you release the mouse button anywhere except on a docking sticker, the docking state of the pane remains floating. In the floating state, you can drag the pane to any position on your screen or even to a second screen.
- If you prefer a pane to constantly remain in the floating state, you can disable the Allow Docking command from the title bar's context menu of the floating pane. This causes the pane to be no longer dockable and no docking stickers will appear when you drag it.
- In the ConfigurationDesk Options dialog, you can configure that panes can be changed to the floating state by dragging their title bar. Refer to [Views Page \(ConfigurationDesk User Interface Reference\)](#).
- If you want to change the order of pane tabs, you can drag them to new positions.
- The docking state and position of each pane is saved for the current view set. For details, refer to [Customizing View Sets](#) on page 57.

---

#### Result

You moved a pane to a different position in a view set.

---

#### Related topics

##### Basics

[Customizing View Sets](#) ..... 57

## How to Customize the Quick Access Toolbar

---

#### Objective

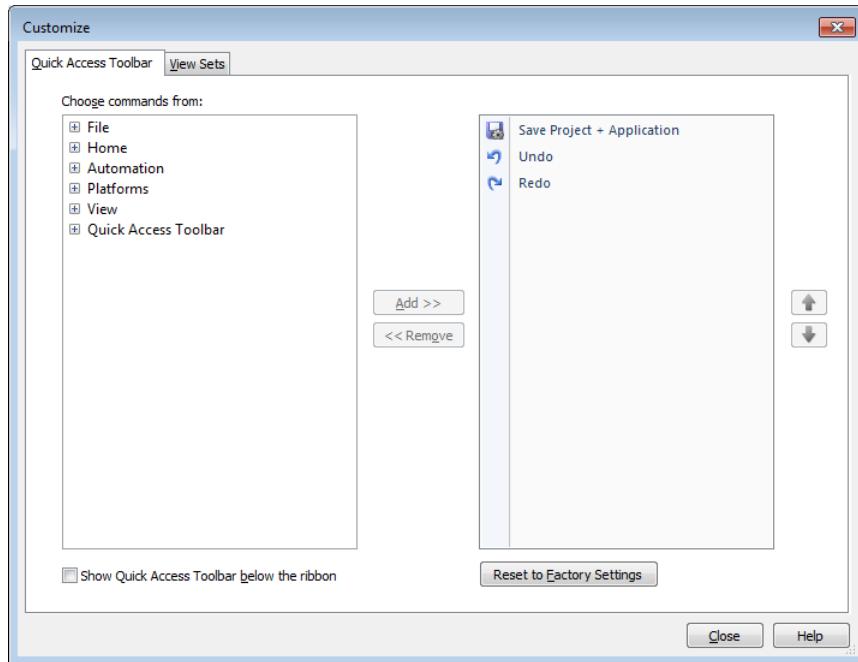
You can customize the Quick Access Toolbar to display the commands that you use frequently and you can specify its position.

## Method

### To customize the Quick Access Toolbar

- 1 On the Quick Access Toolbar, click  – More Commands.

The Quick Access Toolbar page of the Customize dialog is displayed.



- 2 On the Quick Access Toolbar page you can add and remove commands, reset the toolbar and specify the position of the Quick Access Toolbar.
  - To add a command to the Quick Access Toolbar, open the Choose commands from list and select a ribbon. Then select a command from the list on the left and click Add.

**Tip**

To add a command to the Quick Access Toolbar, you can also select Add to Quick Access Toolbar in the context menu of a command in a ribbon.

- To remove a command from the Quick Access Toolbar, select the command in the list on the right and click Remove.

**Tip**

To remove a command from the Quick Access Toolbar, you can also select Remove from Quick Access Toolbar in the context menu of a command icon in the Quick Access Toolbar.

- To reset the Quick Access Toolbar to the factory default, click Reset to Factory Settings.
- To show the Quick Access Toolbar below the ribbon, select Show Quick Access Toolbar below the ribbon.

**3** Click Close to save the changes.

---

**Result**

You have customized the Quick Access Toolbar.

---

**Related topics****Basics**

Basics on Ribbons.....	54
------------------------	----



# Typical Workflows for Beginners

---

## Where to go from here

## Information in this section

Creating a Real-Time Application: From ECU to Model .....	69
Creating a Real-Time Application: Starting with a Simulink Behavior Model.....	77

## Creating a Real-Time Application: From ECU to Model

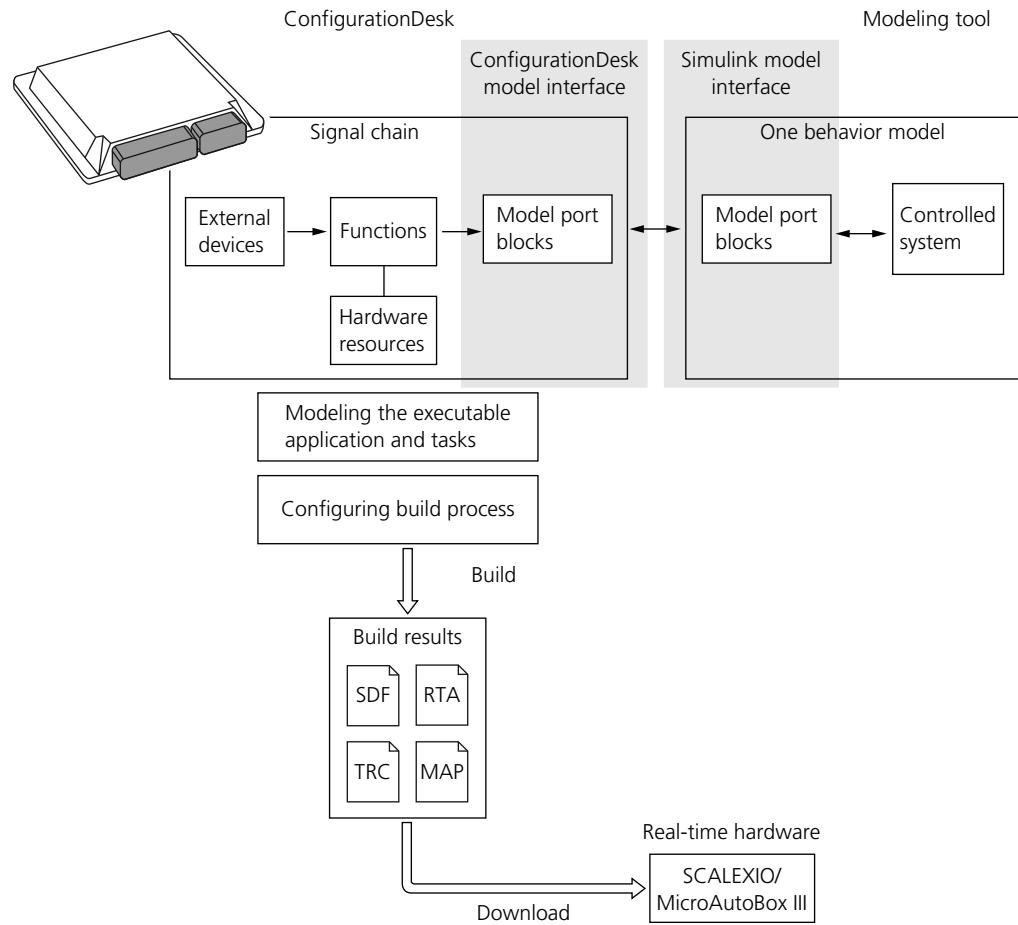
---

### Objective

"From ECU to Model" is a workflow whose starting point is the representation of your ECU in ConfigurationDesk. This workflow is typically used in implementing real-time applications for hardware-in-the-loop simulation scenarios.

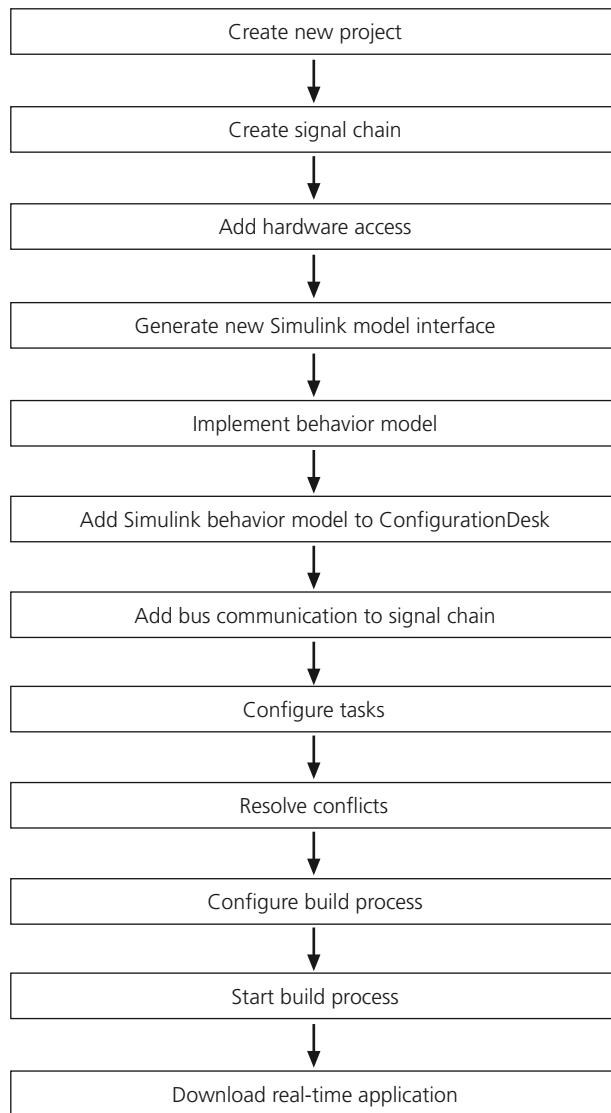
## Overview

The following illustration gives you an overview of the use scenario.



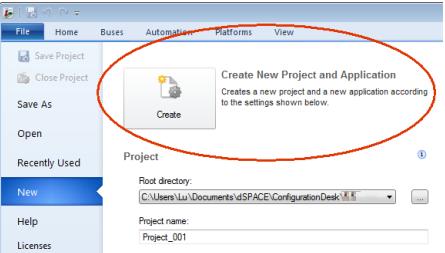
**Workflow**

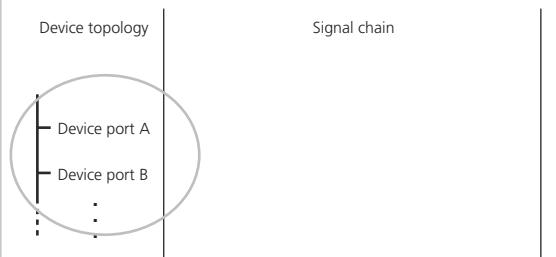
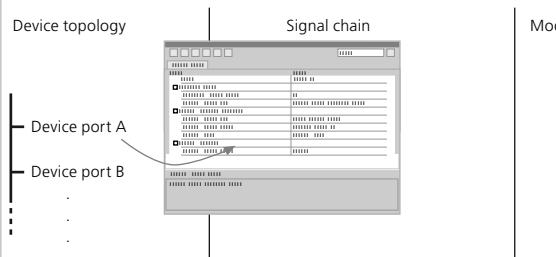
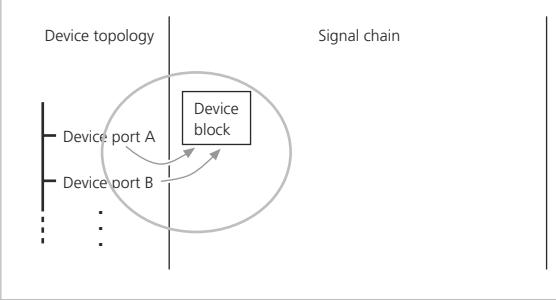
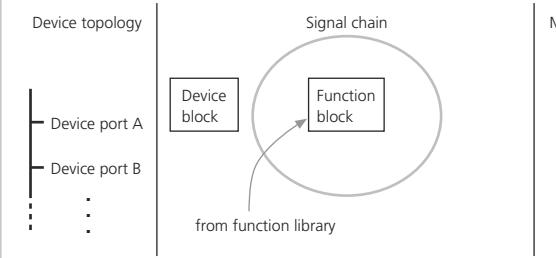
The workflow shows the steps of a standard workflow for implementing a ConfigurationDesk application with one behavior model.

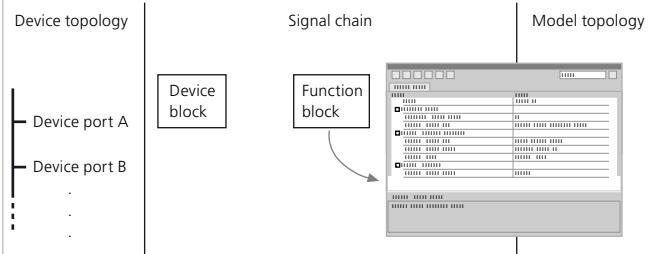
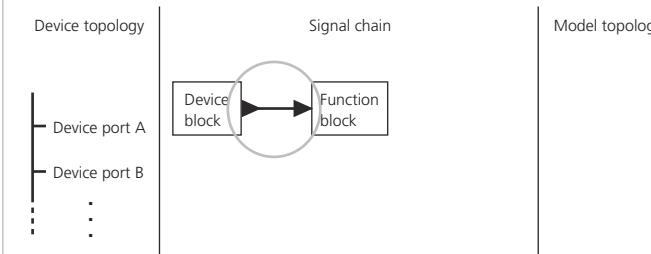
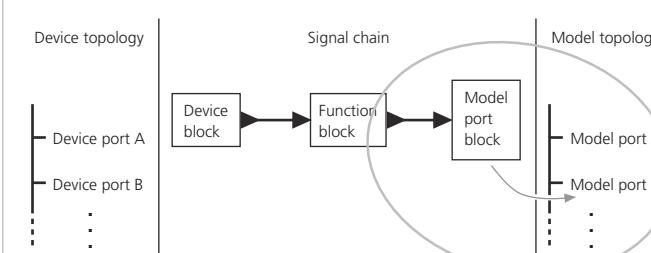
**Detailed steps**

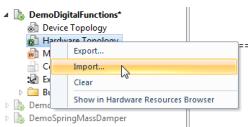
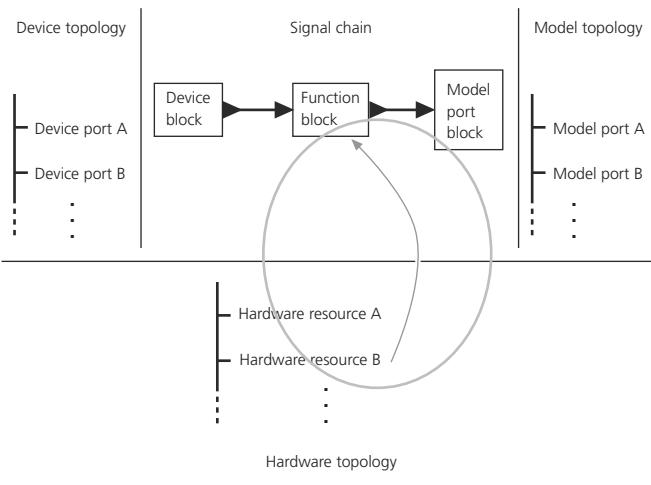
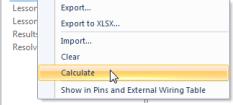
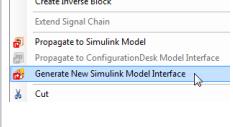
The following table shows detailed workflow steps and indicates the corresponding chapters, which give you detailed information (basic information, instructions etc.).

Step	Work	Refer to ...
1	Create a new project and a ConfigurationDesk application.	Managing ConfigurationDesk Projects

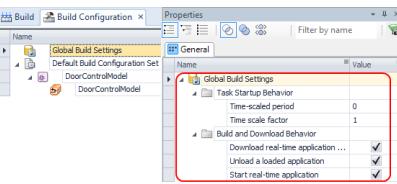
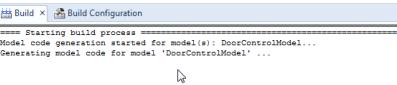
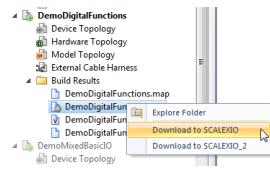
Step	Work	Refer to ...
	 <p>The screenshot shows the ConfigurationDesk application interface. The 'File' menu is open, and the 'New' option is selected. A red oval highlights the 'Create' button in the 'Create New Project and Application' dialog. The dialog also displays the root directory as 'C:\Users\lu\Documents\dSPACE\ConfigurationDesk' and the project name as 'Project_001'.</p>	<p>and Applications on page 85</p>

Step	Work	Refer to ...
2	Create a signal chain in ConfigurationDesk.	
1	Create a device topology.	<a href="#">Creating and Extending Device Topologies on page 260</a>
	 <p>Device topology</p> <p>Signal chain</p> <p>Model topology</p> <p>Device port A</p> <p>Device port B</p>	
2	Configure device ports.	<a href="#">Configuring External Devices on page 272</a>
	 <p>Device topology</p> <p>Signal chain</p> <p>Model topology</p> <p>Device port A</p> <p>Device port B</p>	
3	Add device blocks to the signal chain.	<a href="#">Adding Device Topology Elements to the Signal Chain on page 287</a>
	 <p>Device topology</p> <p>Signal chain</p> <p>Model topology</p> <p>Device port A</p> <p>Device port B</p> <p>Device block</p>	
4	Add function blocks to the signal chain.	<a href="#">Adding Function Blocks to the Signal Chain on page 307</a>
	 <p>Device topology</p> <p>Signal chain</p> <p>Model topology</p> <p>Device port A</p> <p>Device port B</p> <p>Function block</p> <p>from function library</p>	

Step	Work	Refer to ...
5	<p>Configure the function block properties.</p> 	<a href="#">Configuring Function Blocks</a> on page 318
6	<p>Map device blocks to function blocks (= device port mapping).</p> 	<a href="#">Device Port Mapping</a> on page 312
7	<p>Add model port blocks to the signal chain.</p> 	<a href="#">Adding Model Port Blocks to the Signal Chain</a> on page 432

Step	Work	Refer to ...
3	Add hardware access.	
1	Add a hardware topology to the ConfigurationDesk application.	<a href="#">How to Import a Hardware Topology on page 115</a>
		
2	Assign hardware resources to the function blocks.	<a href="#">Assigning Hardware Resources to Function Blocks on page 399</a>
		
3	Calculate wiring information for the cable harness and export it for external usage.	<a href="#">How to (Re)Calculate the External Cable Harness on page 685</a>
		
4	Generate the Simulink model interface and implement the behavior model.	
1	Generate new Simulink model interface.	<a href="#">How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model on page 543</a>
		
2	Implement the behavior model:	
	<ul style="list-style-type: none"> <li>▪ Model the control algorithm. Use Simulink Blocksets and Toolboxes from MathWorks®.</li> <li>▪ Specify the model interface of the behavior model.</li> <li>▪ Configure tasks.</li> <li>▪ Implement bus communication (CAN, LIN, FlexRay) via specific dSPACE RTI blocksets (if necessary).</li> </ul>	<a href="#">Introduction to the Model Interface Package for Simulink (Model Interface Package for Simulink - Modeling Guide)</a>

Step	Work	Refer to ...
3	Add the behavior model to the ConfigurationDesk application.	<a href="#">How to Import a Model Topology on page 112</a>
5	Add bus communication to the signal chain (if necessary).	
	<p>1 Add specific function blocks for bus communication (CAN, LIN) to the signal chain.</p> <p>2 Map these function blocks to device blocks and specific model port blocks (= Configuration Port blocks).</p>	<a href="#">Adding Bus and Gigalink Communication to the Signal Chain on page 335</a>
6	Configure tasks.	<a href="#">Configuring Tasks in ConfigurationDesk on page 513</a>
7	Resolve conflicts via the Conflicts Viewer.	<a href="#">Resolving Conflicts on page 677</a>

Step	Work	Refer to ...
8	Configure and start the build process to build the real-time application.	
1	Configure the build process. 	<a href="#">Configuring the Build Process on page 707</a>
2	Start the build process. 	<a href="#">Starting the Build Process on page 716</a>
9	Download the real-time application to a platform. 	<a href="#">Downloading and Executing Real-Time Applications on page 745</a>

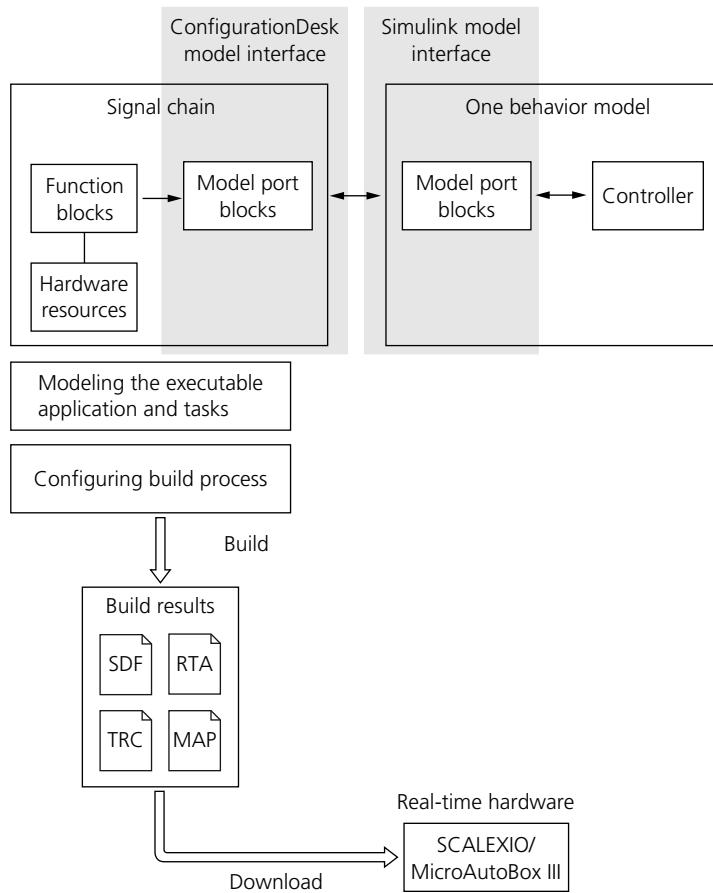
## Creating a Real-Time Application: Starting with a Simulink Behavior Model

### Objective

"Starting with a Simulink behavior model" is a workflow whose starting point is one behavior model in Simulink. This workflow is typically used in implementing real-time applications for rapid prototyping scenarios.

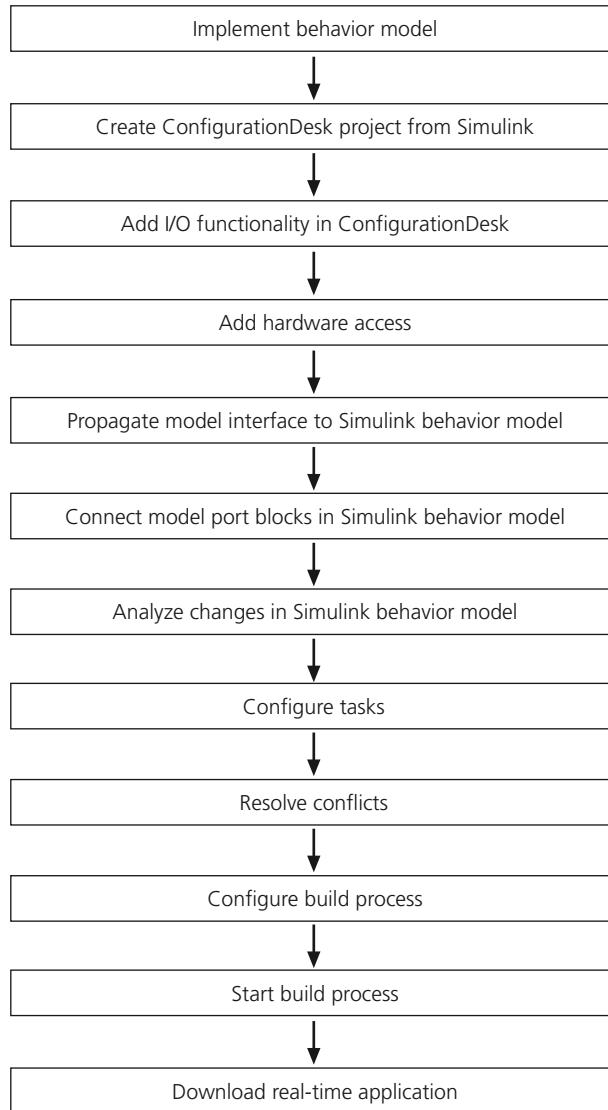
## Overview

The following illustration gives you an overview of the use scenario. ConfigurationDesk Modeling tool



**Workflow**

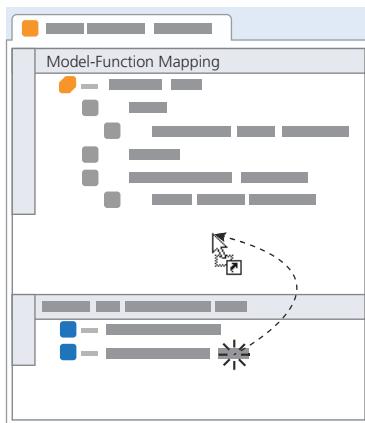
The workflow shows the steps of a standard workflow for implementing a ConfigurationDesk application with one behavior model.

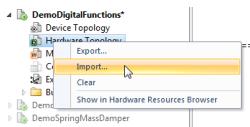
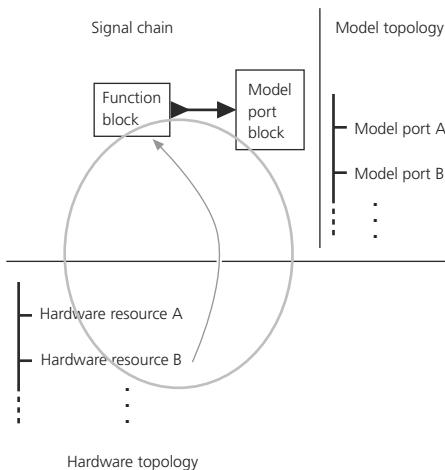
**Detailed steps**

The following table shows detailed workflow steps and indicates the corresponding chapters, which give you detailed information (basic information, instructions etc.).

Step	Work	Refer to ...
1	Implement the behavior model: <ul style="list-style-type: none"> <li>▪ Model the control algorithm. Use Simulink Blocksets and Toolboxes from MathWorks®.</li> <li>▪ Implement bus communication (CAN, LIN, FlexRay) via specific dSPACE RTI blocksets (if necessary).</li> </ul>	<a href="#">Introduction to the Model Interface Package for Simulink (Model Interface Package for Simulink - Modeling Guide)</a>

Step	Work	Refer to ...
2	<p>Create a new ConfigurationDesk project.</p> <p>1 Create ConfigurationDesk project from the Simulink behavior model.</p> <p>2 Add the behavior model to the ConfigurationDesk application.</p>	<a href="#">Remote Access to ConfigurationDesk (Model Interface Package for Simulink - Modeling Guide)</a>
		<a href="#">Managing ConfigurationDesk Projects and Applications on page 85</a>

Step	Work	Refer to ...
3	Add I/O functionality to the ConfigurationDesk application.	
1	Create signal chain using the Model-Function Mapping Browser.	<a href="#">Creating Signal Chains via the Model-Function Mapping Browser on page 555</a>
2	<p>Configure the function block properties.</p> 	<a href="#">Configuring Function Blocks on page 318</a>

Step	Work	Refer to ...
4	<p>Add hardware access.</p> <p>1 Add a hardware topology to the ConfigurationDesk application.</p>  <p>2 Assign hardware resources to the function blocks.</p> 	<a href="#">How to Import a Hardware Topology on page 115</a> <a href="#">Assigning Hardware Resources to Function Blocks on page 399</a>
5	Propagate changes from ConfigurationDesk to the Simulink behavior model.	<a href="#">Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model on page 558</a>
6	Connect new model port blocks in Simulink behavior model.	<a href="#">Introduction to the Model Interface Package for Simulink (Model Interface Package for Simulink - Modeling Guide)</a>

Step	Work	Refer to ...
7	Analyze changes in Simulink behavior model from ConfigurationDesk.	<a href="#">Analyzing Simulink Behavior Models on page 561</a>
8	Configure tasks.	<a href="#">Configuring Tasks in ConfigurationDesk on page 513</a>
9	Resolve conflicts via the Conflicts Viewer.	<a href="#">Resolving Conflicts on page 677</a>
10	Configure and start the build process to build the real-time application.	
1	Configure the build process.	<a href="#">Configuring the Build Process on page 707</a>
2	Start the build process.	<a href="#">Starting the Build Process on page 716</a>
11	Download the real-time application to a platform.	<a href="#">Downloading and Executing Real-Time Applications on page 745</a>



# Managing ConfigurationDesk Projects and Applications

---

**Objective** Managing projects and ConfigurationDesk applications is the basis for performing tasks in ConfigurationDesk.

---

Where to go from here	Information in this section
	<a href="#">Introducing ConfigurationDesk Projects Applications.....</a> 86
	<a href="#">Managing ConfigurationDesk Projects.....</a> 92
	<a href="#">Managing ConfigurationDesk Applications.....</a> 100
	<a href="#">Managing Components of ConfigurationDesk Applications.....</a> 105

# Introducing ConfigurationDesk Projects Applications

---

<b>Objective</b>	To perform your tasks in ConfigurationDesk, you need a project and an application.
------------------	--

## Handling ConfigurationDesk Projects and Applications

---

<b>Definition of projects and applications</b>	To structure your work in ConfigurationDesk you need two key elements: projects and applications.
--	---

**Projects** A project centralizes all the relevant items of your work in ConfigurationDesk, for example:

- One or more applications that belong together.
- Project-specific documents such as project plans or specifications.

A project thus functions as a container for applications and all project-specific documents.

**Applications** An application is the basis for carrying out implementation work in ConfigurationDesk. It must always be included in a project, i.e., it cannot exist separately. One or more applications can be contained in a project, each representing a specific implementation. An application can contain the following components:

- Device topology
- Hardware topology
- Model topology
- Communication matrices
- External cable harness
- Build results (after a successful build process has finished)
- Generated containers (after bus simulation containers have been generated)

**Note**

You can work with only one application at a time, and that application must be activated. This application is called active application.

The following illustration visualizes the interaction between projects and applications:

**Project 1**

Project-specific files

**Application 1.1**

- Device Topology
- Hardware Topology
- Model Topology
- Communication Matrices
- External Cable Harness
- Build Results
- Generated Containers

**Application 1.2**

- Device Topology
- Hardware Topology
- Model Topology
- Communication Matrices
- External Cable Harness
- Build Results
- Generated Containers

**Project 2**

Project-specific files

**Application 2.1**

- Device Topology
- Hardware Topology
- Model Topology
- Communication Matrices
- External Cable Harness
- Build Results
- Generated Containers

**Application 2.2**

- Device Topology
- Hardware Topology
- Model Topology
- Communication Matrices
- External Cable Harness
- Build Results
- Generated Containers

**Application 2.3**

- Device Topology
- Hardware Topology
- Model Topology
- Communication Matrices
- External Cable Harness
- Build Results
- Generated Containers

**Project root folders for grouping projects**

Before projects can be created, at least one project root folder must be specified. Project root folders are folders 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.

**Default project root folder** ConfigurationDesk uses the *Documents folder* (refer to [Documents folder](#) on page 20) as the default project root folder unless you specify a different one.

**Specifying further project root folders** You can specify further project root folders in addition to the default project root folder. This allows you to specify different destination directories for your projects, and to group projects.

For instructions, refer to [How to Specify a Project Root Folder](#) on page 96.

### Note

To avoid errors, adhere to the following rules:

- Do not specify a ConfigurationDesk project folder as a project root folder.
- Do not specify a project root folder in a project root folder.

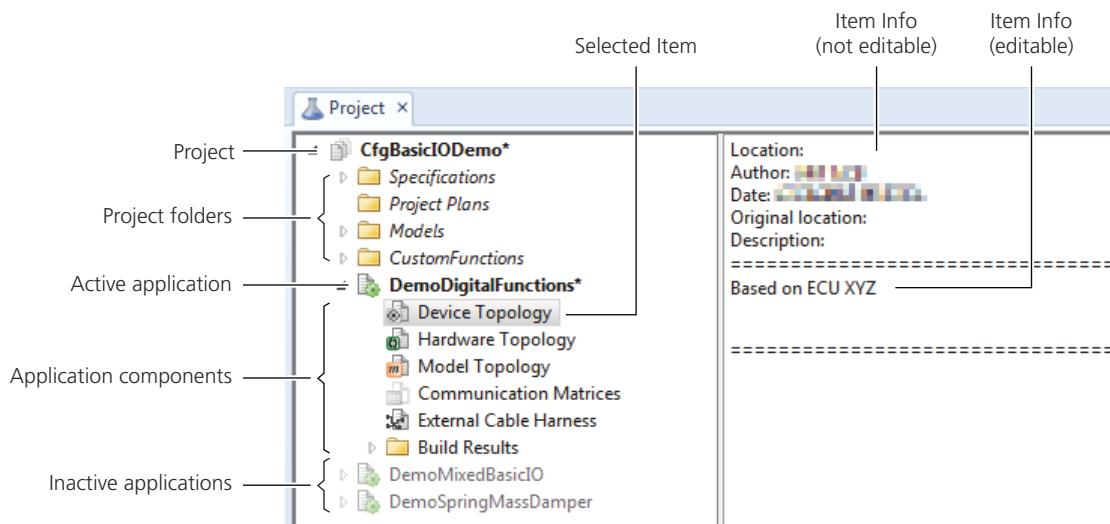
## Using the Project view set

ConfigurationDesk offers the Project view set to handle your project management tasks. It displays the Start Page and the Project Manager. The Start Page is closed after you opened a project.

**Start Page** The Start Page provides:

- Commands for opening an existing project or create a new one.
- Access to recently opened projects and applications.
- Access to product documentation.

**Project Manager** In the Project Manager, you can manage the hierarchical project structure of project folders, applications, and application components.



Some Project Manager commands are also available on the Home ribbon.

If a project contains unsaved data, ConfigurationDesk displays an asterisk next to the project name in the Project Manager. An asterisk next to an application indicates that the application contains unsaved data. Asterisks disappear when you save the project or application.

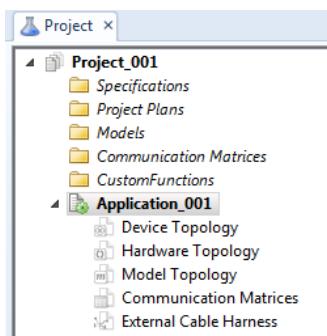
**Backstage view** Many project management commands are available in the backstage view. In most cases you are automatically directed to the relevant backstage view commands when you use a command in the Project Manager, on the Start Page, or on the Home ribbon. For example, using the New Application command on the Home ribbon automatically directs you to the

New ribbon group of the backstage view where you can specify and create an application.

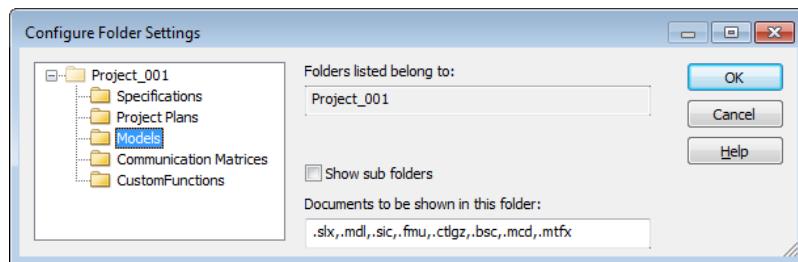
### Handling external documents in project folders

ConfigurationDesk lets you include external documents in a project, such as specifications or model files which are necessary for implementing your real-time application. They are displayed in project folders in the Project Manager. External documents cannot be modified with ConfigurationDesk, but you can open them in associated external programs via double-click and they are included in project backups and exported applications.

Every new project contains the Specifications, Project Plans, Models, Communication Matrices, and CustomFunctions folders.



Selecting Configure from the context menu of a project folder opens a dialog for you to configure the file extensions of the documents to be displayed in the folder, for example. The following illustration shows the default settings for the Models folder:



#### Tip

Enter `.*` in the Documents to be shown in this folder field to display all file types.

You can create new project or application folders by right-clicking the project item or the active application in the Project Manager and selecting Add Folder from the context menu.

A project folder in ConfigurationDesk represents an actual subfolder of the project folder in the *Documents folder*. Use the Explore Folder command to open the folder and add or remove files. Select Refresh from the context menu of a project or application to see the changes in an open project.

**Application states in ConfigurationDesk**

Each active ConfigurationDesk application has one of two application states, depending on the hardware topology and registered platforms:

State	Status Bar Visualization	Description
Matching platform connected	 (green)	The hardware topology of the active application matches the hardware topology of an accessible hardware system displayed in the Platform Manager.
No matching platform connected	 (white)	The hardware topology of the active application does not match the hardware topology of any accessible hardware system displayed in the Platform Manager or no registered platform is available.

For more details, refer to [Basics on Connecting a ConfigurationDesk Application to a Hardware System](#) on page 207.

**Support of path and file names**

To be supported by the build process, path and file names of projects and applications can contain the following characters only:

- A-Z, a-z, 0-9
- Underscores \_
- Minus -
- Dots .
- Parentheses ()

**Note**

Restrictions for path names:

- A path must not contain dots or whitespaces without other characters.
- White spaces and dots must not be the first or last character in a path name.

Restrictions for file names:

- Dots must not be the first or last character in a file name.
- A minus must not be the first character in a file name.

**Demo projects**

ConfigurationDesk includes a number of demo projects. They are available in the *Documents folder*, so you can open them like a normal project.

The demo projects are also available in ZIP archives, which reside in the <RCP and HIL installation folder>\Demos\ConfigurationDesk folder after installation of the dSPACE software.

---

Related topics

Basics

Basics on Connecting a ConfigurationDesk Application to a Hardware System.....	207
Managing Components of ConfigurationDesk Applications.....	105
Managing ConfigurationDesk Applications.....	100

HowTos

How to Back up and Transfer a Project.....	98
How to Create a Project.....	92
How to Open a Project.....	93
How to Specify a Project Root Folder.....	96

# Managing ConfigurationDesk Projects

<b>Objective</b>	Projects are key elements in ConfigurationDesk. They hold all the items that are relevant for working with ConfigurationDesk.
------------------	---

Where to go from here	Information in this section
	<a href="#">How to Create a Project.....</a> 92
	<a href="#">How to Open a Project.....</a> 93
	<a href="#">How to Specify a Project Root Folder.....</a> 96
	<a href="#">How to Back up and Transfer a Project.....</a> 98

## How to Create a Project

<b>Objective</b>	To be able to work with ConfigurationDesk applications, you must create a project first.
------------------	--

<b>Preconditions</b>	No other project must be open.
----------------------	--------------------------------

Method	<b>To create a project</b>
	<p><b>1</b> Switch to the Project view set if necessary.</p> <p><b>2</b> On the Home ribbon, click Project – New Project + Application. ConfigurationDesk switches to the New ribbon group of the File ribbon.</p> <p><b>3</b> From the Root directory list, select a project root directory. If you want to specify a new project root directory, click  . For detailed instructions on specifying project root directories, refer to <a href="#">How to Specify a Project Root Folder</a> on page 96.</p> <p><b>4</b> In the Project name field, change the default name. The name you enter can contain letters, digits, underscores, minus signs, parentheses, and dots. The name must not contain a dot or whitespace as the first or last character.</p> <p><b>5</b> In the Application name field, change the default name of the automatically added application. To create a project without an application, clear the Application name field.</p> <p><b>6</b> Optional: Add a model topology or hardware topology to the application. You can do this later.</p>

For details on the available model topology and hardware topology settings, refer to:

- [How to Import a Model Topology](#) on page 112
- [How to Import a Hardware Topology](#) on page 115

**7 Click Create.**

ConfigurationDesk creates and opens the new project, activates the added application, and opens it in the **Model–Function** view set.

<b>Result</b>	You created a new project.
---------------	----------------------------

<b>Next steps</b>	<ul style="list-style-type: none"> <li>▪ You can start to implement and configure I/O functionality via function blocks. Refer to <a href="#">Implementing I/O Functionality</a> on page 291.</li> <li>▪ You can add external documents, applications, and components to the new project. Refer to:           <ul style="list-style-type: none"> <li>▪ <a href="#">Handling ConfigurationDesk Projects and Applications</a> on page 86</li> <li>▪ <a href="#">Managing ConfigurationDesk Applications</a> on page 100</li> <li>▪ <a href="#">Managing Components of ConfigurationDesk Applications</a> on page 105</li> </ul> </li> </ul>
-------------------	---

<b>Related topics</b>	Basics
Handling ConfigurationDesk Projects and Applications.....86	

## How to Open a Project

<b>Objective</b>	To work with an existing project, you must first open it.
------------------	---

<b>Possible methods</b>	<p>Usually, you open projects that are stored in the project root folder. But you can also open projects that are stored outside of it, for example, for quick access without changing the root folder or moving files to it.</p> <ul style="list-style-type: none"> <li>▪ To open a project that is stored in a project root folder, refer to <a href="#">Method 1</a> on page 94.</li> <li>▪ To open a project that is stored outside of the project root folder, refer to <a href="#">Method 2</a> on page 95.</li> </ul>
-------------------------	--

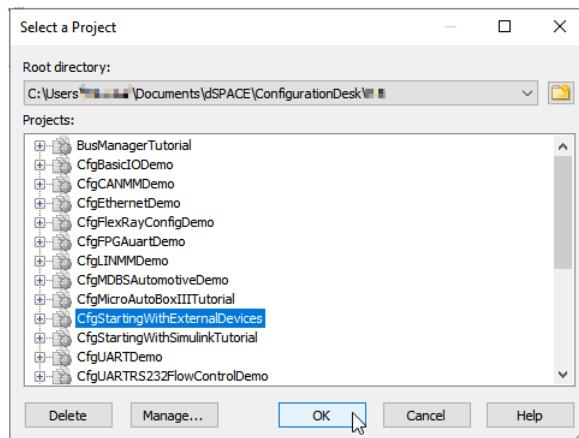
<b>Opening projects created with previous ConfigurationDesk versions</b>	You can open a project created with a previous ConfigurationDesk version higher than 4.2 in the same way as a project from the current ConfigurationDesk version. ConfigurationDesk automatically migrates the project when opening it.
--	---

**Note**

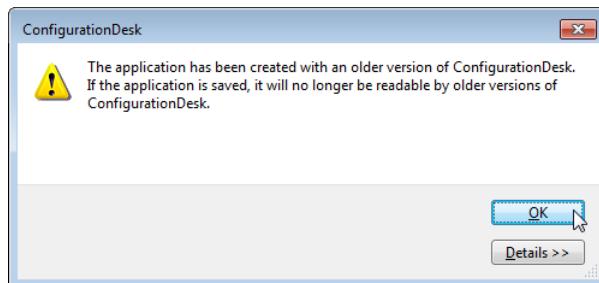
- As of dSPACE Release 2021-A, ConfigurationDesk supports the direct import only of projects last saved with one of the previous seven ConfigurationDesk versions.
- After you have migrated and saved a project, you can no longer open it with a previous ConfigurationDesk version.

**Method 1****To open a project that is stored in a project root folder**

- 1 Switch to the Project view set if necessary.
  - 2 On the File ribbon, click Open – Open Project.
- ConfigurationDesk opens the Select a Project dialog.



- 3 From the Root directory list, select the project root folder containing the project you want to open, or click to define a new one.
- 4 In the Projects field, select the project you want to open. You can also expand the project and select an application that you want to be active after opening the project.
- 5 Click OK.
- 6 If you select a project created with a previous ConfigurationDesk version, ConfigurationDesk opens the following dialog:



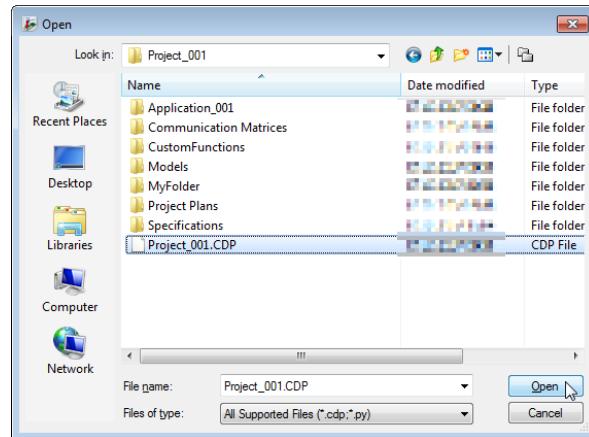
Click OK to start the migration.

**Method 2****To open a project that is stored outside of a project root folder**

- 1 On the File ribbon, click Open – Open File or Project.

ConfigurationDesk opens a standard Open dialog for you to browse for a project (CDP) file.

- 2 Select a CDP file and click Open.



ConfigurationDesk opens the selected project. The path to the selected CDP file is not added to the list of project root folders.

**Note**

There are some restrictions for working with projects outside a project root folder regarding the tool automation.

**Result**

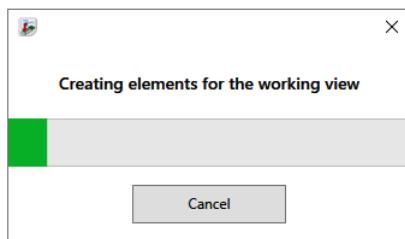
You opened a ConfigurationDesk project.

ConfigurationDesk:

- Closes the current project (if applicable). You are prompted to save any unsaved changes.
- Opens and loads the previously active application in this project or the application you selected in the Select a Project dialog.

**Tip**

- You can also open a project by selecting it from the Recently Used – Recent Projects and Applications list on the File ribbon or from the Recent list on the Start Page if no project is open.
- If you want to open a project and its most recently used ConfigurationDesk application directly after starting ConfigurationDesk, you must enable the Automatically load the most recently used application on startup checkbox on the Project page of the ConfigurationDesk Options dialog.
- You can create a desktop shortcut for each ConfigurationDesk application via its context menu. This allows you to open ConfigurationDesk and quickly load the project containing a specific application.
- In the Select a Project dialog, you can use the Delete button to delete the selected project(s). You can also right-click the project you want to remove and click Delete.
- If a large working view with many signal chain elements is open in the activated application, a progress bar is displayed while the working view is prepared.



Click Cancel to stop the opening process. The working view is not opened. You can open it later from the Working View Manager.

**Related topics**

**Basics**

[Handling ConfigurationDesk Projects and Applications.....](#) 86

## How to Specify a Project Root Folder

**Objective**

ConfigurationDesk allows you to specify different project root folders. This enables you to group projects on your file system.

**Grouping projects in root folders**

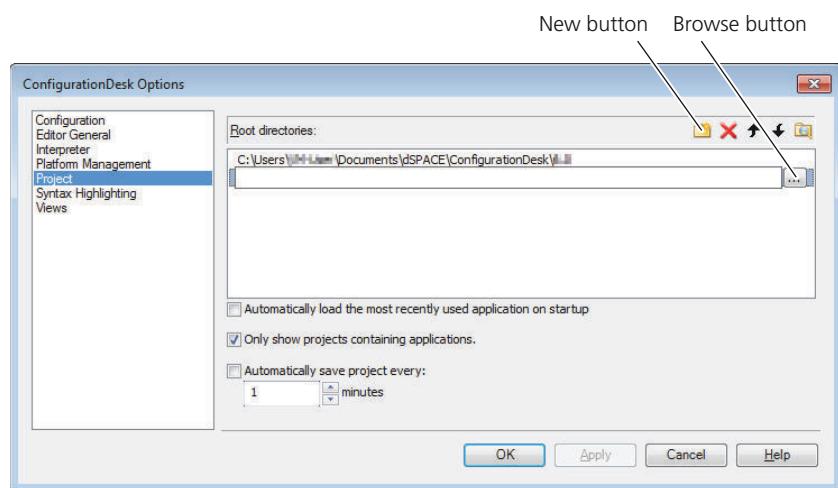
Each ConfigurationDesk project is related to a project root folder. This is a physical folder on your file system. ConfigurationDesk creates a folder structure beneath the project root folder and saves all the files of a project to it.

**Default project root folder**

ConfigurationDesk uses the *Documents folder* as the default project root folder unless you specify a different one.

**Method****To specify a project root folder**

- 1 On the File ribbon, click Options to open the ConfigurationDesk Options dialog.
- 2 On the Project page of the ConfigurationDesk Options dialog, click  , then click the Browse button.



- 3 In the Browse for Folder dialog, select the new project root folder and click OK.

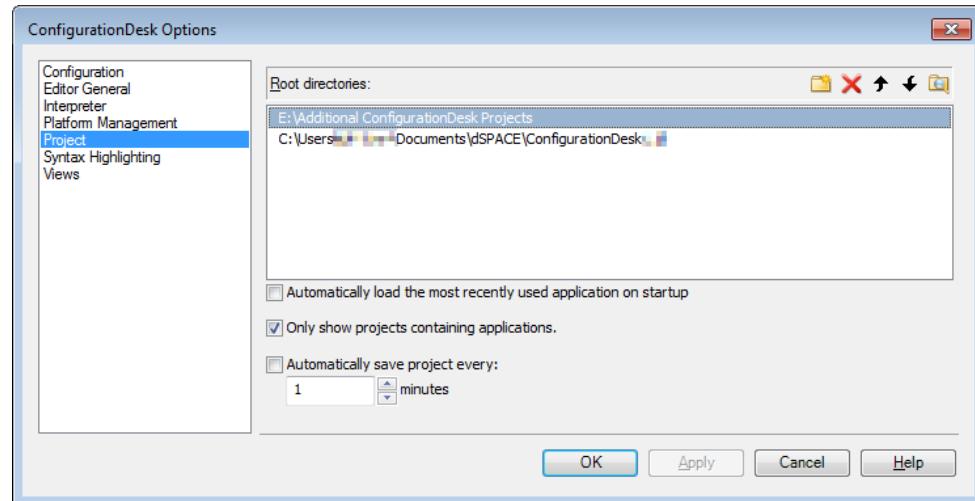
The path name can contain letters, digits, underscores, minus, parentheses, and dots. The name must not contain a dot or whitespace as the first or last character.

**Note**

To avoid errors, adhere to the following rules:

- Do not specify a ConfigurationDesk project folder as a project root folder.
- Do not specify a project root folder in a project root folder.

The ConfigurationDesk Options dialog displays the new entry:



**4** Click OK.

The new project root folder now is at the top of the list. Whenever you need to specify a project root folder, the entry at the top of this list is preselected. Use the arrow symbols to change the order of project root folders.

**5** Click OK or Apply to confirm your settings.

---

**Result**

You specified a new project root folder.

**Tip**

You can modify a project root folder by double-clicking and then editing its list entry.

---

**Related topics**

**Basics**

[Handling ConfigurationDesk Projects and Applications.....](#) 86

## How to Back up and Transfer a Project

---

**Objective**

Backing up a project allows you to save and transfer the entire contents of a project, including external documents and the build results, in one ZIP archive.

---

**Contents of the backup**

Backup ZIP archives contain only files that are part of the project tree.

**Note**

Folders and their contents (documents and further subfolders) are included in a backup ZIP archive only if they are displayed in the Project Manager. To include subfolders, you have to enable the Show subfolders option in the Configure Folder Settings dialog. To include documents, you have to specify a comma-separated list with file name extensions. Refer to [Handling ConfigurationDesk Projects and Applications](#) on page 86.

---

**Method****To back up and transfer a project**

- 1 On the File ribbon, click Save As – Backup Project. A standard Save As dialog opens.
- 2 Save the project as a ZIP archive.  
You can keep the ZIP archive as a backup or transfer its contents to a different ConfigurationDesk PC. The files are archived with relative paths. If a ZIP archive contains a file from a different file system, the absolute path is stored.
- 3 To load the contents of a project backup, click Open – Open Project and Application from Backup on the File ribbon. A standard Open dialog opens.
- 4 Select a ZIP archive. You can also specify the project root folder the project is to be extracted to.
- 5 Click Open to extract all the files of the project backup to the specified project root folder.

---

**Result**

You have backed up and transferred the contents of a project.

---

**Related topics****Basics**

[Handling ConfigurationDesk Projects and Applications](#).....86

# Managing ConfigurationDesk Applications

<b>Objective</b>	ConfigurationDesk applications are necessary for carrying out implementation actions in ConfigurationDesk. Applications are defined and managed within projects.
------------------	--

Where to go from here	Information in this section
	<p><a href="#">How to Add a ConfigurationDesk Application to a Project</a>..... 100</p> <p><a href="#">How to Activate a ConfigurationDesk Application</a>..... 101</p> <p><a href="#">How to Export and Import a ConfigurationDesk Application</a> ..... 102</p> <p><a href="#">How to Create a ControlDesk Experiment from a ConfigurationDesk Application</a>..... 103</p>

## How to Add a ConfigurationDesk Application to a Project

<b>Objective</b>	To carry out specific implementation actions, you can add additional ConfigurationDesk applications to a project.
<b>Preconditions</b>	A project must be open.
<b>Method</b>	<p><b>To add a ConfigurationDesk application to a project</b></p> <p><b>1</b> Switch to the Project view set if necessary.</p> <p><b>2</b> On the Home ribbon, click Project – New Application. ConfigurationDesk switches to the New ribbon group of the File ribbon.</p> <p><b>3</b> In the Application name field, change the default name. The name can contain letters, digits, underscores, minus, parentheses, and dots. The name you enter must not contain a dot or whitespace as the first or last character.</p> <p><b>4</b> Optional: Add a model topology or hardware topology to the application. You can do this later.</p> <p>For details on the available model topology and hardware topology settings, refer to:</p> <ul style="list-style-type: none"><li>▪ <a href="#">How to Import a Model Topology</a> on page 112</li><li>▪ <a href="#">How to Import a Hardware Topology</a> on page 115</li></ul>

**5** Click Create Application.

ConfigurationDesk creates and activates the new application.

**Result**

You added a ConfigurationDesk application to a project.

**Next steps**

You can add external documents and components to the application. Refer to:

- [Handling ConfigurationDesk Projects and Applications](#) on page 86
- [Managing Components of ConfigurationDesk Applications](#) on page 105

**Related topics****Basics**

[Handling ConfigurationDesk Projects and Applications.....](#) 86

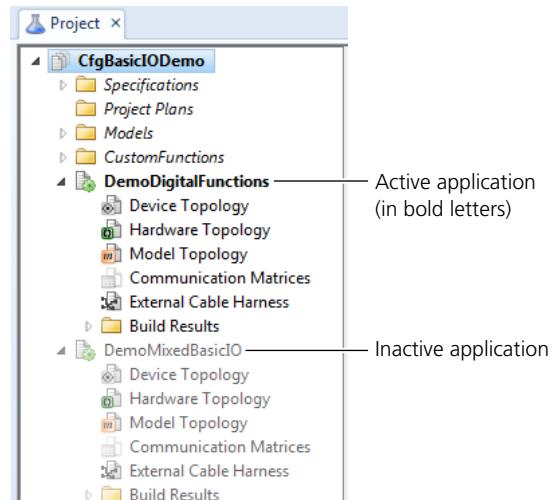
## How to Activate a ConfigurationDesk Application

**Objective**

You cannot work with a ConfigurationDesk application which is inactive. You must first activate it.

**Method****To activate a ConfigurationDesk application**

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

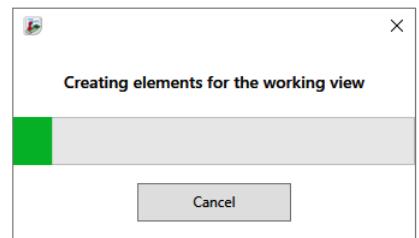


- 2 From the context menu, select Activate.

ConfigurationDesk activates the selected application, and deactivates the other. The latest file data of the active application is loaded.

**Tip**

If a large working view with many signal chain elements is open in the activated application, a progress bar is displayed while the working view is prepared.



Click Cancel to stop the opening process. The working view is not opened. You can open it later from the Working View Manager.

---

**Result**

You activated a ConfigurationDesk application.

---

**Related topics**

Basics

Handling ConfigurationDesk Projects and Applications.....86

## How to Export and Import a ConfigurationDesk Application

---

**Objective**

Exporting a ConfigurationDesk application allows you to save its contents (including external documents and build results) in a dSPACE archive file (DSA file) and transfer it to another project.

**Tip**

You can also use the export/import function to:

- Make a copy of an application in the same project.
- Make an application available to others, for example, if they do not have MATLAB, but need to configure and build the application. After your colleagues have configured the exported application and transferred it back to you, you can build it for them.

---

<b>Precondition</b>	A ConfigurationDesk application must already be defined and activated.		
<b>Method</b>	<p><b>To export and import a ConfigurationDesk application</b></p> <ol style="list-style-type: none"> <li>1 In the Project Manager, right-click the active application.</li> <li>2 From the context menu, select Export. A standard Save As dialog opens. You can keep the generated DSA file as a backup or transfer its contents to a different ConfigurationDesk project.</li> <li>3 To load the contents of an exported application, right-click a project in the Project Manager.</li> <li>4 From the context menu, select Import Application. A standard Open dialog opens.</li> <li>5 Select a DSA file.</li> <li>6 Click Open to extract all the files of the exported application.</li> </ol>		
<b>Result</b>	You have exported and imported a ConfigurationDesk application.		
<b>Related topics</b>	<p>Basics</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px; width: 80%;">Handling ConfigurationDesk Projects and Applications.....</td> <td style="padding: 5px; width: 20%; text-align: right;">86</td> </tr> </table>	Handling ConfigurationDesk Projects and Applications.....	86
Handling ConfigurationDesk Projects and Applications.....	86		

---

## How to Create a ControlDesk Experiment from a ConfigurationDesk Application

---

<b>Objective</b>	ConfigurationDesk lets you create a ControlDesk project and experiment from the currently active ConfigurationDesk application based on the build results.
	<p><b>Note</b></p> <p>Currently this feature supports only ConfigurationDesk applications that contain a single application process, i.e., the SDF file references one variable description file. If the SDF file references multiple variable description files, e.g., in a multicore or multi-processing-unit application, the created project will be incomplete. For these applications we recommend that you still create the project and experiment in ControlDesk.</p>

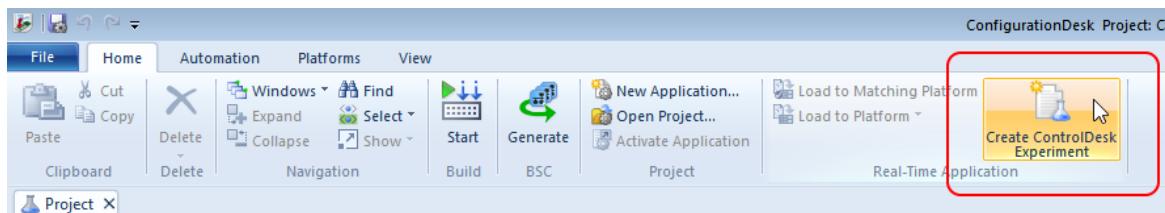
---

**Preconditions**

- The dSPACE experiment software ControlDesk must be installed on the same system.
- A ConfigurationDesk project must be open, and one of its applications must be active.
- The active ConfigurationDesk application must contain build results.
- If you want the platform name in the ControlDesk experiment to match your registered platform, make sure that the platform is the only registered platform in ConfigurationDesk and that the application state is Matching platform connected.

**Method****To create a ControlDesk experiment from a ConfigurationDesk application**

- 1 Switch to the Project view set if necessary.
- 2 On the Home – Real-Time Application ribbon, click Create ControlDesk Experiment.



- A new ControlDesk project named after the ConfigurationDesk project is created in the ControlDesk project root directory and opened in ControlDesk.
- An experiment named after the ConfigurationDesk application is created and activated in the ControlDesk project.
- The experiment contains a layout with a Numeric Input instrument connected to the Task Turnaround Time variable of the Periodic Task 1 in the variable description file.

**Note**

- If a matching platform is registered and connected to the ConfigurationDesk application, the platform in the ControlDesk experiment is named after it. Otherwise, a default name is used.
- If a ControlDesk project with the name of the ConfigurationDesk project already exists, it is opened and the experiment is added to it.
- If an existing ControlDesk project with the same name already contains an experiment with the name of the ConfigurationDesk application, it is not overwritten. However, missing information such as the new name of a matching platform is added.

**Result**

You created a ControlDesk experiment from a ConfigurationDesk application.

# Managing Components of ConfigurationDesk Applications

<b>Objective</b>	ConfigurationDesk applications consist of different components. Each component provides ConfigurationDesk with component-specific information. This information can, for example, comprise data on the interface to the behavior model or to the external devices.
------------------	--

Where to go from here	Information in this section
	<p>Basics on Components of ConfigurationDesk Applications..... 105</p> <p>How to Import a Device Topology..... 110</p> <p>How to Import a Model Topology..... 112</p> <p>How to Import a Hardware Topology..... 115</p> <p>How to Import an External Cable Harness..... 117</p> <p>How to Export Data of a Specific Application Component..... 119</p>

## Basics on Components of ConfigurationDesk Applications

<b>Components of a ConfigurationDesk application</b>	A ConfigurationDesk application contains the following kinds of components:  <b>Device topology</b> Contains information on the interface of the external devices, such as the ECUs to be tested. The information includes details of the available device pins and their characteristics. The device topology can be imported from an existing DTFX file or from an XLSX file, or you can create it yourself in ConfigurationDesk. DTFX is the exchange format for device topologies. You can export and import DTFX files. For more information on the external device interface, refer to <a href="#">Specifying the External Device Interface</a> on page 251.  <b>Hardware topology</b> 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 types and slot numbers. A hardware topology can be added to an application by scanning registered hardware, importing an existing HTFX file or creating a hardware topology from scratch. HTFX is the exchange format for hardware topologies. You can export and import HTFX files.
--	--

For more information on handling hardware in ConfigurationDesk, refer to [Managing Real-Time Hardware](#) on page 201.

**Model topology** Contains information on the interface to the behavior model, such as the implemented model port blocks and their subsystems. You can add different file types, such as SLX or FMU files, to the ConfigurationDesk application. MTFX is the exchange format for model topologies. You can export and import MTFX files. For details on the different file types, refer to [Adding model topology information](#) on page 108.

For more information on the model interface, refer to [Specifying the Model Interface](#) on page 417.

**Communication matrices** Contains information on the communication matrices that are imported and therefore available in the active application. You need communication matrices when you implement bus communication with the Bus Manager.

For more information on working with communication matrices, refer to [Working with Communication Matrices \(ConfigurationDesk Bus Manager Implementation Guide\)](#).

**External cable harness** 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, such as the ECUs to be tested. ECHX is the exchange format for external cable harnesses. You can export and import ECHX files.

For more information on the external cable harness, refer to [Calculating an External Cable Harness](#) on page 683.

**Build results** This folder displays the files which are generated during a successful build process. These files are:

- TRC file(s): Contain the description of all the signals and parameters of the generated real-time application.
- MAP file(s): Map symbolic names to physical addresses.
- RTA file: Is the executable file for the real-time application.
- SDF file: Provides information about CPU name(s), object file(s) and variable description file(s).
- CANCFG file: Contains configuration data (in XML format) for CAN communication
- LINCFG file: Contains configuration data (in XML format) for LIN communication
- FLXCFG file: Contains configuration data (in XML format) for FlexRay communication
- A2L file:
  - Contains model-specific variables provided by the A2L fragment of the related V-ECU implementation container
  - Contains variables for I/O functions and test automation
  - Contains DAQ raster names and the XCP service port number
  - Contains information required to perform virtual bypassing via the RTI Bypass Blockset (only for application processes containing a V-ECU implementation configured for virtual bypassing).

- EXPSWCFG file: Contains configuration data (in XML format) for automotive fieldbus communication.
- HEX file (initial data file, generated only for V-ECU implementations that support memory pages): Contains information about the initial state of calibration variables.

You can remove the build results files from the active ConfigurationDesk application by using the Clear Build Results command from the context menu of the application or the Build Results folder. They are also removed from your file system.

For more information on the build process, refer to [Building Real-Time Applications](#) on page 697.

**Generated Containers** This folder displays generated bus simulation container (BSC) files. For more information, refer to [Basics on Bus Simulation Containers \(ConfigurationDesk Bus Manager Implementation Guide\)](#).

## Handling of application components

Each component can be contained only once in an application.

ConfigurationDesk allows you to import, replace, remove, and export any of them except for the build results and generated containers. Communication matrices can only be removed.

Only device and hardware topologies can be created from scratch and merged during import.

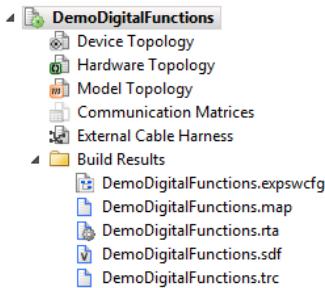
If you remove a device, hardware or model topology from an application, its content is deleted except for elements which are still used in the signal chain. This also applies to used pins of the external cable harness. These elements are marked as unresolved. They disappear when they are removed from the signal chain. Information on unresolved elements is displayed as a conflict.

### Note

When you remove communication matrices via the Project Manager, you delete all the communication matrices and their elements that are used in the signal chain. The bus communication that is implemented in the signal chain is lost completely. To prevent unintended data loss, remove selected communication matrices via the Buses Browser.

## Display of application components

The Project Manager displays the components of ConfigurationDesk applications under the application to which they belong. The following illustration shows an application with different kinds of components:



The Build Results are displayed after you have executed at least one successful build process. The External Cable Harness has to be calculated, otherwise it is grayed out. Empty topologies are also grayed out.

## Adding model topology information

The model topology lets you choose between different file types to add to the active ConfigurationDesk application:

- MDL file or SLX file

The Simulink model files contains the behavior model. To work with MDL file or SLX file data in ConfigurationDesk, you must first perform a model analysis. The model analysis process creates a model topology from the Simulink model file.

- MTFX file

The model topology file is a replacement format and contains information on the interface to the behavior model, such as the implemented model port blocks and their subsystems. It can be exported and imported from/to a model topology. Each MTFX file originates from a model analysis of an MDL or SLX file.

- MCD file

The model communication description file is used to add several behavior models that were separated from an overall model to implement a multimodel application.

The MCD file contains information on the separated models and information on the connections between them. The file is generated with the modeling tool. In MATLAB/Simulink this is done with the Model Separation Setup block, which resides in the Model Interface Blockset from dSPACE.

- SIC file

An SIC file is a Simulink implementation container file containing the model code of a Simulink behavior model. An SIC file is generated from a Simulink behavior model using the Model Interface Package for Simulink. It can also be generated with TargetLink.

- BSC file

A BSC file is a bus simulation container file that contains the configured bus communication of one application process. A BSC file is generated with the Bus Manager.

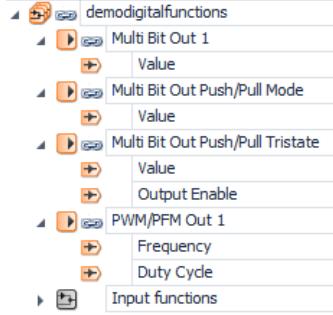
- VECU files are ZIP files containing a V-ECU implementation. VECU files can contain data packages for different platforms. VECU files are exported from TargetLink or SystemDesk. You can add a V-ECU implementation based on a VECU file to the model topology in the same way as adding a Simulink model based on an MDL file.

- FMU file

FMU files are used to add model implementations of the Functional Mock-up Unit type (FMUs) to the model topology. FMUs are based on the Functional Mock-up Interface standard (FMI standard). The FMI standard defines an interface standard for model exchange and co-simulation. It is supported by different tools, for example, Dymola, MapleSim, and SimulationX. An FMU implements a model using the interfaces defined by the FMI standard.

Depending on which file type you add to the application, different effects occur in the model topology:

- If you add an MTFX file or an MDL (or SLX) and perform model analysis on it, the model topology is displayed in a hierarchical structure like this:



- If you add an MDL (or SLX) file and do not perform model analysis on it at the same time, only the model item is displayed:

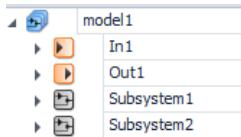


After you perform a model analysis, the model topology is displayed in a hierarchical structure.

- If you add an MCD file and do not perform model analysis on it at the same time, the separated models are displayed without their interfaces. Only blocks which are connected between one separated model and another are displayed as unresolved blocks.

After you perform a model analysis, the model topology is updated with the complete model interface data of your behavior models.

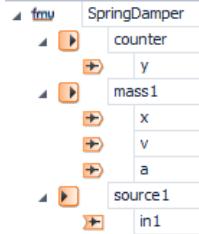
- If you add an SIC file, the model topology is displayed like this:



- If you add a VECU file, the model topology is displayed like this:



- If you add an FMU file, the model topology is displayed like this:



## Related topics

### Basics

[Handling ConfigurationDesk Projects and Applications](#).....86

### HowTos

How to Export Data of a Specific Application Component.....	119
How to Import a Device Topology.....	110
How to Import a Hardware Topology.....	115
How to Import a Model Topology.....	112
How to Import an External Cable Harness.....	117

## How to Import a Device Topology

### Objective

For example, if you want to calculate the wiring information of your application, ConfigurationDesk requires a device topology. The device topology contains information on the interface to the external devices, such as the ECUs to be tested.

### Precondition

A device topology stored in a DTFX or XLSX file must be available.

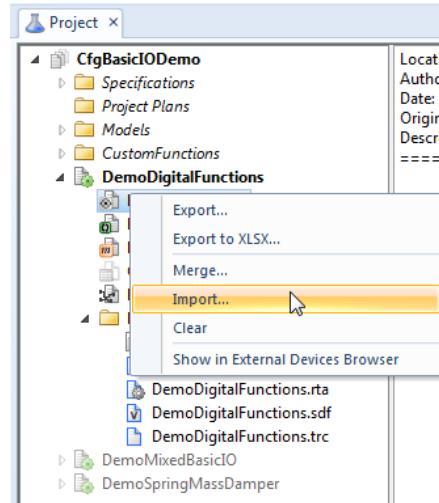
#### Note

An external device topology stored in an XLSX file must comply with editing rules. Rows that do not comply with the editing rules are not added. ConfigurationDesk does not check whether all the rows of a table are imported.

For more information, refer to [Rules for Editing External Device Topologies](#) on page 268.

**Method****To import a device topology**

- 1 In the Project Manager, right-click the device topology of the currently active application and select Import from the context menu.



A standard Open dialog opens.

- 2 Choose the file format you want to import:
  - You can add a DTFX file containing a device topology to the application.
  - You can add a Microsoft Excel™ file (XLSX) containing a device topology to the application.
- 3 Select the file containing the device topology information and click Open. ConfigurationDesk adds the device topology information to the active application. The device topology is displayed in a hierarchical structure in the External Device Browser.

**Note**

Existing device topology elements are removed. If they were used in the signal chain, they are still displayed and marked as unresolved.

**Result**

You imported a device topology.

**Tip**

You can also create and extend a device topology in ConfigurationDesk or merge multiple device topologies. Refer to [Creating and Extending Device Topologies](#) on page 260.

**Related topics****Basics**

Basics on Components of ConfigurationDesk Applications.....	105
Specifying the External Device Interface.....	251

## How to Import a Model Topology

**Objective**

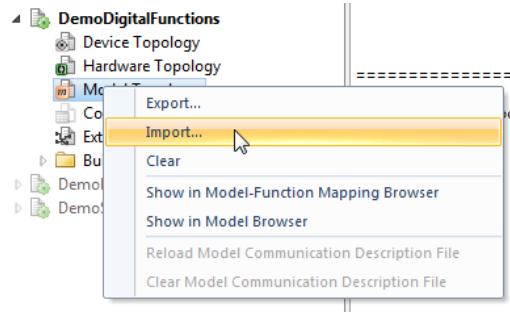
To implement a real-time application, ConfigurationDesk requires a model topology. It contains information about the interface to the behavior model, such as the implemented model port blocks including their subsystems.

**Different file types**

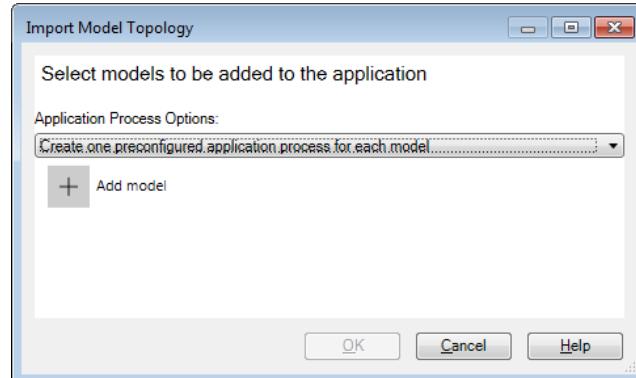
If you want to import a model topology to an application, you can select between different file types. For details, refer to [Basics on Components of ConfigurationDesk Applications](#) on page 105.

**Method****To import a model topology**

- In the Project Manager, right-click the model topology of the currently active application and select **Import** from the context menu.



An Import Model Topology dialog opens.



**2** Select one of the Application Process Options.

The Application Process Options let you specify the assignment of the selected model implementations to the created application processes. You can select one of the following options:

- Create one preconfigured application process for each model (default).  
If you select this option, ConfigurationDesk creates a separate application process for each model implementation that is to be added to the active ConfigurationDesk application.
- Create one application process for all models and optimize configuration.  
If you select this option, ConfigurationDesk creates one application process and assigns all the selected model implementations to it. ConfigurationDesk optimizes the configuration of the new application process by grouping runnable functions in tasks, and specifying the task priorities.
- Create no application process.  
If you select this option, ConfigurationDesk adds the selected model implementations to the ConfigurationDesk application without creating application processes.

**Note**

For the Create one preconfigured application process for each model option and the Create one application process for all models and optimize configuration option the following applies:

- For Simulink models, these options are available only if you select the Analyze model (including task information) checkbox. If the model analysis fails, the preconfigured application process is not created.
- These options are not available if the executable application contains more than one processing unit application.

**3** Click Add model.**Tip**

If you already opened behavior models in Simulink, you can select them directly from the Models open in Simulink list.

**4** Click Add model from file.

A standard Open dialog opens for you to select the file type and the file you want to add to the ConfigurationDesk application.

- 5 Select one of the following file types and the file you want to add to the ConfigurationDesk application:
  - Simulink model (\*.mdl; \*.slx)

**Tip**

- It may be useful to save the referenced Simulink model file directly to a project folder that is configured to display MDL / SLX files. For more information, refer to [Handling ConfigurationDesk Projects and Applications](#) on page 86. You can use the folder to save data belonging to your project, such as model files which are necessary for implementing your real-time application.
- You can find a Models folder with an appropriate configuration in the project folder per default.

- Simulink implementation container file (\*.sic)
- Bus simulation container file (\*.bsc)
- Model communication description file (\*.mcd)
- Model topology file (\*.mtfx)
- V-ECU implementation container (\*.vecu)
- Functional Mock-up Unit (\*.fmu)

- 6 Click Open.

The selected file is added to the Selected model(s) list. Depending on the selected file type, the following options are available after clicking :

- Analyze model (including task information) (only available for MDL, SLX, MTFX, and MCD files)
- Model initialization command (only available for Simulink models and MCD files)

The Analyze model (including task information) checkbox is selected by default. You can clear it if required.

- 7 In the Import Model Topology dialog, enter a model initialization command, if required.

**Tip**

Model initialization commands are executed before the behavior model is opened, for example, to prepare data in the MATLAB Base Workspace that is linked to the behavior model. You can also define a model initialization command later. For details and instructions, refer to [How to Initialize Simulink Behavior Models](#) on page 546.

- 8 Repeat steps 3 - 7 for all the files you want to add to the ConfigurationDesk application. You can select multiple files from the same folder at once.

To remove file from the list of models to be added, click .

- 9 Click OK.

- ConfigurationDesk adds the selected files to the active application. Depending on the file types you added to the application, different effects

occur in the Project Manager and in the model topology. For details, refer to [Basics on Components of ConfigurationDesk Applications](#) on page 105.

- ConfigurationDesk creates a preconfigured application process for the selected models according to the Application Process Options.

#### Note

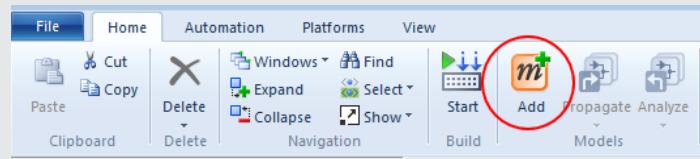
Existing model topology elements are removed. If they were used in the signal chain, they are still displayed and marked as unresolved.

#### Result

You imported a model topology.

#### Tip

You can also add model files without removing existing model topology elements by using the Add command on the Home ribbon of most view sets.



#### Related topics

##### Basics

<a href="#">Basics on Components of ConfigurationDesk Applications</a> .....	105
<a href="#">Specifying the Model Interface</a> .....	417

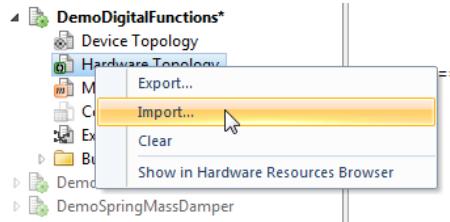
## How to Import a Hardware Topology

#### Objective

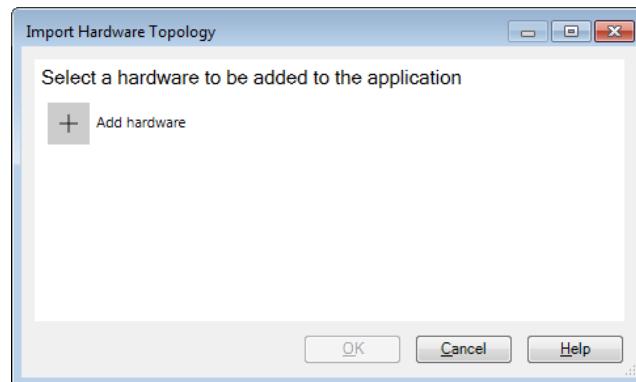
To implement a real-time application, ConfigurationDesk requires a hardware topology. The hardware topology contains information on a specific hardware system which can be used with ConfigurationDesk. It provides information on the components of the system, such as boards, channel types, and slot numbers.

**Method****To import a hardware topology**

- 1 In the Project Manager, right-click the hardware topology of the currently active application and select Import from the context menu.



An Import Hardware Topology dialog opens.



- 2 Click Add hardware.
- 3 Select a hardware platform or file to be used for creating a hardware topology:
  - Registered platforms are available in the Registered Hardware list.
  - A Default Hardware based on a platform or a file is also available if you previously saved one via the Save as Default command.
  - Add hardware from file lets you select and import an existing hardware topology file (HTFX file).

**Tip**

- ConfigurationDesk offers a number of predefined hardware topologies that you can add to your ConfigurationDesk application. The topologies are stored in the following folder, which is opened by default:  
`<Documents folder>\PredefinedHardware`
- You can click the down arrow next to the added hardware to make the Save as Default command available. Using the command makes the hardware the default hardware for future hardware topology imports.
- Register platforms opens the Register Platforms dialog, where you can scan for and register a dSPACE real-time hardware platform. Afterwards, the platform is available in the Registered Hardware list.

**4** Click OK.

- ConfigurationDesk adds the hardware topology information to the active application. The hardware topology is displayed in a hierarchical structure in the Hardware Resource Browser.
- ConfigurationDesk automatically performs an identity check: It checks whether the hardware topology of any hardware system displayed in the Platform Manager is identical to the hardware topology of your active application. For more information, refer to [Basics on Connecting a ConfigurationDesk Application to a Hardware System](#) on page 207.

**Note**

Existing hardware topology elements are removed. If they were used in the signal chain, they are still displayed and marked as unresolved.

**Result**

You imported a hardware topology.

**Tip**

You can also create and extend the hardware topology in ConfigurationDesk or merge hardware topologies. Refer to [Working with Hardware Topologies](#) on page 236.

**Related topics****Basics**

Basics on Components of ConfigurationDesk Applications.....	105
Managing Real-Time Hardware.....	201

## How to Import an External Cable Harness

**Objective**

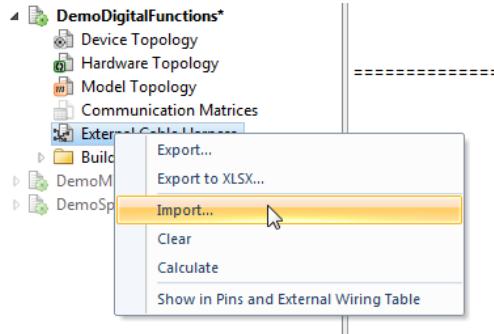
If you want to use a specific cable harness for your application, ConfigurationDesk requires information on it. The external cable harness, stored as an ECHX file, contains data on the wiring of pins. Data on physical characteristics is not included in the ECHX file. The external cable harness is the connection between the I/O connectors of the real-time hardware and the devices, such as the ECUs to be tested.

**Precondition**

An external cable harness stored in an ECHX file must be available.

**Method****To import an external cable harness**

- 1 In the Project Manager, right-click the external cable harness of the currently active application and select Import from the context menu.



A standard Open dialog opens.

- 2 Select the ECHX file containing the wiring information and click Open. ConfigurationDesk adds the external cable harness representing the description of a physically existing cable harness to the active application.

**Note**

Existing wiring information is overwritten. If pins were used in the signal chain, they are still displayed and marked as unresolved.

**Result**

You imported an external cable harness.

**Tip**

You can also calculate the external cable harness on the basis of the signal chain configuration. Refer to [Calculating an External Cable Harness](#) on page 683.

**Related topics****Basics**

Basics on Components of ConfigurationDesk Applications.....	105
Basics on Device Pin Assignment.....	255
Calculating an External Cable Harness.....	683

## How to Export Data of a Specific Application Component

### Objective

ConfigurationDesk allows you to export the data of each application component into a specific internal file format to reuse it in other ConfigurationDesk projects and applications.

### Extensions of file types

The following table shows the file name extensions of the file types for the different application components:

Application Component	File Name Extension
Device topology	DTFX
Hardware topology	HTFX
Model topology	MTFX
External cable harness	ECHX

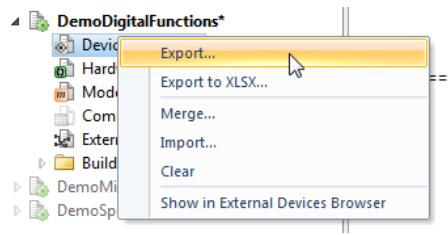
### Precondition

The ConfigurationDesk application which contains the application component to be exported is active.

### Method

#### To export data of a specific application component

- 1 In the Project Manager, right-click the application component whose data you want to export.
- 2 Select Export from the context menu.



A component-specific Export <Component> dialog with preset file format opens.

- 3 Specify a file name and location and click Save.

**Result**

You exported the data of a specific application component.

**Tip**

- You can export the device topology and the external cable harness into the Microsoft Excel™ file format (XLSX file), too. Refer to [How to Export a Device Topology to a Microsoft Excel Sheet](#) on page 267 or [How to Export Wiring Information for an External Cable Harness](#) on page 689.
- You can export a selected system or rack and all its subelements from the hardware topology. Refer to [How to Export a Hardware Topology Part](#) on page 244.

---

**Related topics**

**Basics**

<a href="#">Basics on Components of ConfigurationDesk Applications</a> .....	105
--	-----

**HowTos**

<a href="#">How to Import a Device Topology</a> .....	110
<a href="#">How to Import a Hardware Topology</a> .....	115
<a href="#">How to Import a Model Topology</a> .....	112
<a href="#">How to Import an External Cable Harness</a> .....	117

# Accessing and Configuring Elements of a ConfigurationDesk Application

---

**Objective** ConfigurationDesk offers several convenient methods for selecting, displaying, and configuring elements and their properties.

---

Where to go from here	Information in this section
	<a href="#">Accessing Elements.....</a> 122
	<a href="#">Configuring Elements with the Properties Browser.....</a> 131
	<a href="#">Using Tables to Access and Configure Elements.....</a> 151
	<a href="#">Using Display Filters.....</a> 158

# Accessing Elements

<b>Objective</b>	You can use various features to find, show, and select one or more elements of a ConfigurationDesk application.
------------------	---

Where to go from here	Information in this section
	<a href="#">Basics on Accessing Elements.....</a> 122
	<a href="#">How to Select Elements of the Same Type.....</a> 123
	<a href="#">How to Show Elements in a Different Pane.....</a> 125
	<a href="#">How to Select and Show Assigned Elements.....</a> 127
	<a href="#">How to Find Elements of a ConfigurationDesk Application.....</a> 129

## Basics on Accessing Elements

<b>Selecting an element</b>	You can select an element of a ConfigurationDesk application by clicking it. This has the following effects: <ul style="list-style-type: none"><li>▪ The element is selected in each pane that contains the element. This includes panes in inactive view sets.</li><li>▪ The element's properties are accessible in the Properties Browser.</li></ul>
-----------------------------	--

<b>Selecting multiple elements</b>	The effects of selecting multiple elements are the same as for selecting single elements.
------------------------------------	---

There are different methods for selecting multiple elements:

**Multiselect with Ctrl key** Press and hold the **Ctrl** key while clicking different elements to select them.

### Tip

You can multiselect all types of elements together. To avoid accidentally including a previously selected element in your selection, make sure to click the first element *before* pressing **Ctrl** and adding other elements to your selection.

**Selecting multiple elements of the same type** You can select multiple elements of the same type, such as all function blocks in a working view or all channels in the hardware topology. For instructions, refer to [How to Select Elements of the Same Type](#) on page 123.

**Multiselect area in a working view** In a working view, you can select multiple elements by holding the left mouse button and drawing a box around them.

**Selecting multiple blocks in a working view column** In a working view, you can select multiple blocks in a column by clicking a block, pressing **Shift**, and afterwards clicking a different block. This way, you select both blocks and all blocks between them in the column.

**Multiselect hierarchy elements with Shift key** In a hierarchy of elements, you can select an element and then press the **Shift** key while clicking a different element to select both elements and all elements between them.

#### Showing elements in a different pane

You can show selected elements in a specific different pane using, for example, the Navigation – Show list on the Home ribbon. For instructions, refer to [How to Show Elements in a Different Pane](#) on page 125.

#### Selecting and showing assigned elements

You can select and show elements that are assigned to each other, for example, hardware channels that are assigned to function blocks. For instructions, refer to [How to Select and Show Assigned Elements](#) on page 127.

#### Finding elements

You can find elements of a ConfigurationDesk application by entering a search string. For instructions, refer to [How to Find Elements of a ConfigurationDesk Application](#) on page 129.

#### Related topics

##### HowTos

How to Find Elements of a ConfigurationDesk Application.....	129
How to Select and Show Assigned Elements.....	127
How to Select Elements of the Same Type.....	123
How to Show Elements in a Different Pane.....	125

## How to Select Elements of the Same Type

#### Objective

You can select multiple elements of the same type, for example, to set common properties to the same value.

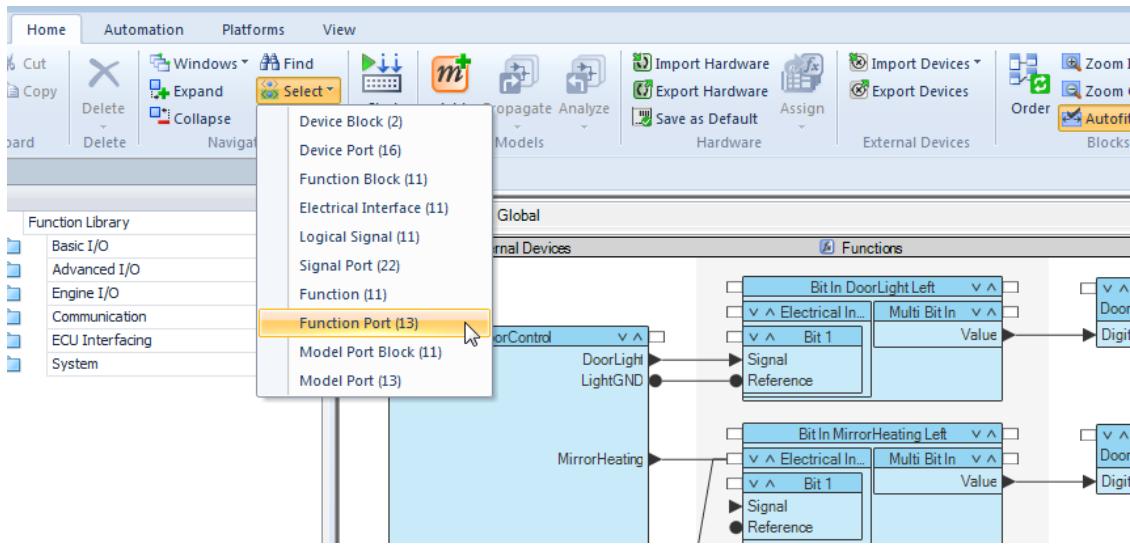
**Elements of the same type**

Elements of the same type are, for example, elements that are used at the same position in the signal chain (e.g., device ports, function blocks, channels).

**Method****To select elements of the same type**

- 1 Click a selection of elements, a higher-level element in a hierarchy of elements, or a free area in a pane containing elements.

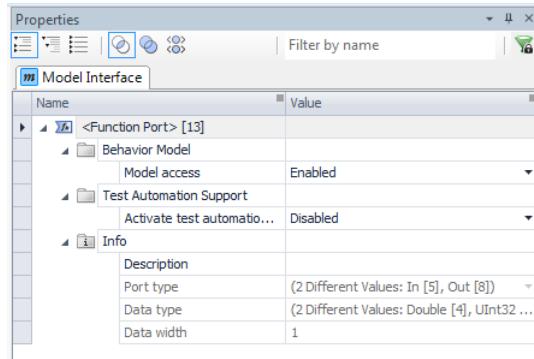
On the Home ribbon, the Navigation – Select submenu offers the element types from the selection, subhierarchy, or pane. The number of elements of the given type is indicated in parentheses. The following illustration shows the available element types in a working view containing signal chain elements.

**Tip**

A Select Elements by Type submenu is also available in several context menus.

**2** Select an available type from the Select submenu.

You can now access the properties of your selection in the Properties Browser, for example.

**Result**

You selected elements of a specific type.

**Next steps**

You can now set common properties to the same value for your selection.

Refer to:

- [Configuring Properties in the Properties Browser](#) on page 134
- [Use Scenarios for Modes and Filters in the Properties Browser](#) on page 145

**Related topics****Basics**

[Basics on Accessing Elements.....](#) 122

## How to Show Elements in a Different Pane

**Objective**

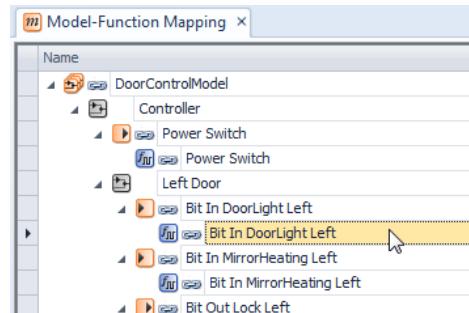
Most elements of a ConfigurationDesk application can be shown in multiple panes for different purposes.

**Method**

**To show elements in a different pane**

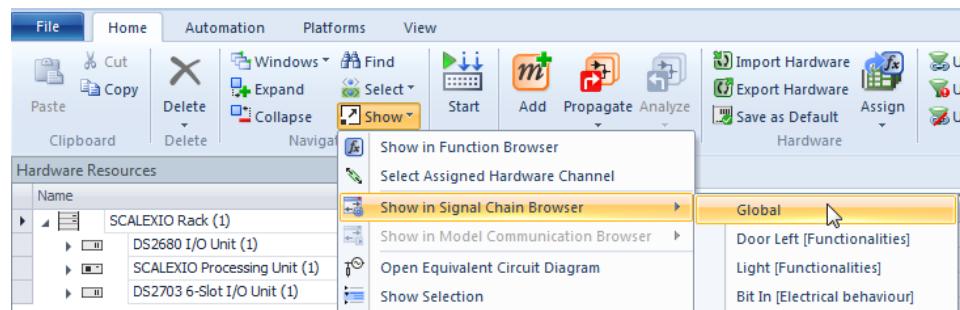
- 1 Select one or more elements in a pane.

For example, select a function block in the Model-Function Mapping Browser.

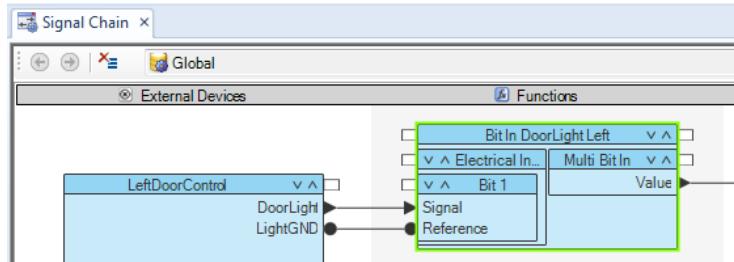


- 2 On the Home ribbon, select one of the Show in commands from the Navigation – Show submenu. The available commands depend on the types of elements you selected.

For example, select Show in Signal Chain Browser – Global for the selected function block.



ConfigurationDesk shows your selection in the desired pane. If necessary, ConfigurationDesk opens the pane and switches to a different view set.



## Result

You have shown selected elements in a different pane.

## Related topics

### Basics

[Basics on Accessing Elements.....](#) 122

## How to Select and Show Assigned Elements

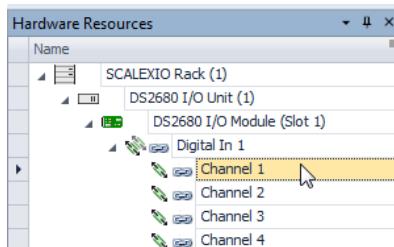
### Objective

To select and show elements that are assigned to each other.

### Method

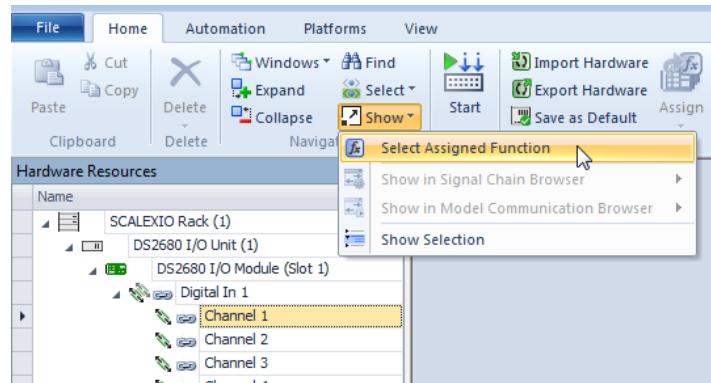
#### To select and show assigned elements

- In a pane, select one or more elements that are assigned to different elements.  
For example, in the Hardware Resource Browser, select a hardware channel that is assigned to a function block.

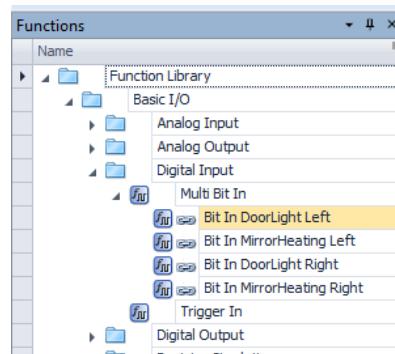


- On the Home ribbon, select one of the Select commands from the Navigation – Show submenu. The available commands depend on the types of elements you selected.

For example, select Select Assigned Function for the hardware channel.



ConfigurationDesk selects and shows you the assigned elements in the most common pane for the respective element type. If necessary, ConfigurationDesk opens the pane and switches to a different view set.



### Tip

There are always commands available to select and show in both directions. For the example used here, the Select Assigned Hardware Channel command lets you select hardware channels from function blocks.

## Result

You have selected and shown assigned elements.

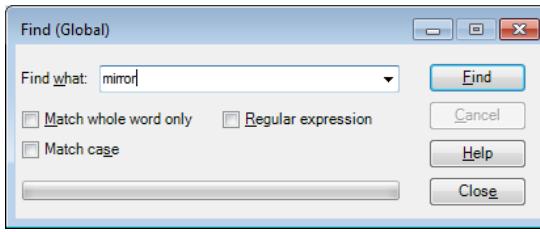
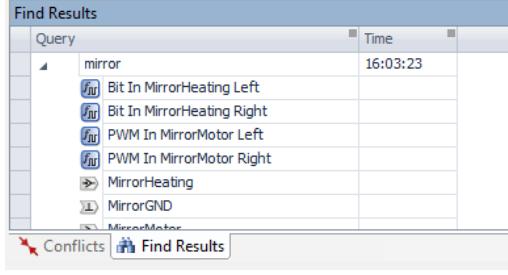
## Related topics

### Basics

Basics on Accessing Elements.....	122
-----------------------------------	-----

## How to Find Elements of a ConfigurationDesk Application

<b>Objective</b>	To find elements of a ConfigurationDesk application by entering a search string.
<b>Basics on the find functionality</b>	<ul style="list-style-type: none"> <li>▪ The name of every element in the active ConfigurationDesk application whose properties can be accessed in the Properties Browser is included in the search. This includes, for example, device topology elements, tasks, or build configuration sets. It does not include, for example, mapping lines.</li> <li>▪ There are no character limitations. However, searching for an empty string will return no results.</li> <li>▪ If you are not sure about the exact name of the element you are searching for, you can search for known fragments or single characters.</li> <li>▪ You can use the following functionalities to reduce the number of matches: <ul style="list-style-type: none"> <li>▪ You can limit the results to matches where no letter or numeral precedes or follows the search string (<b>Match whole word only</b>) and/or to matches which have the same case as the search string (<b>Match case</b>).</li> <li>▪ You can use regular expressions to limit the number of matches (<b>Regular expression</b>). The regular expression can also be case sensitive (<b>Match case</b>).</li> </ul> </li> <li>▪ If the element name contains spaces, they must be included in the search string.</li> </ul>

<b>Method</b>	<b>To find an element of a ConfigurationDesk application</b>
	<ol style="list-style-type: none"> <li>1 On the Home ribbon, click Navigation – Find. The Find (Global) dialog opens.</li> <li>2 In the Find what field, enter your search string.</li> </ol>  <ol style="list-style-type: none"> <li>3 Click Find.</li> </ol> <p>The results are displayed in the Find Results Viewer, which is opened if necessary.</p> 

The results are displayed in two columns:

- The first column displays the results as a tree. The top node contains the search string. You can collapse and expand the results. The list is sorted by element type, which you can identify by its symbol.
- The second column displays the time at which the search was done.

The Remove and Clear context menu commands let you remove individual search results or clear all search results from the Find Results Viewer.

---

## Result

You found elements of the active ConfigurationDesk application and the result is displayed in the Find Results Viewer.

### Tip

You can also open the Find (Global) dialog via the keyboard shortcut **Ctrl+F**.

---

## Next steps

- In the Find Results Viewer, you can select elements and access their properties in the Properties Browser. Refer to [Configuring Properties in the Properties Browser](#) on page 134.
- You can show elements in specific panes or select and show elements assigned to them. Refer to:
  - [How to Show Elements in a Different Pane](#) on page 125
  - [How to Select and Show Assigned Elements](#) on page 127
- From the Find Results Viewer, you can perform drag & drop operations such as instantiating a function block by dragging a function block type to a working view.

---

## Related topics

### Basics

Basics on Accessing Elements.....	122
-----------------------------------	-----

# Configuring Elements with the Properties Browser

<b>Objective</b>	In the Properties Browser, you can edit and view all the properties of selected elements in the active ConfigurationDesk application.
------------------	---

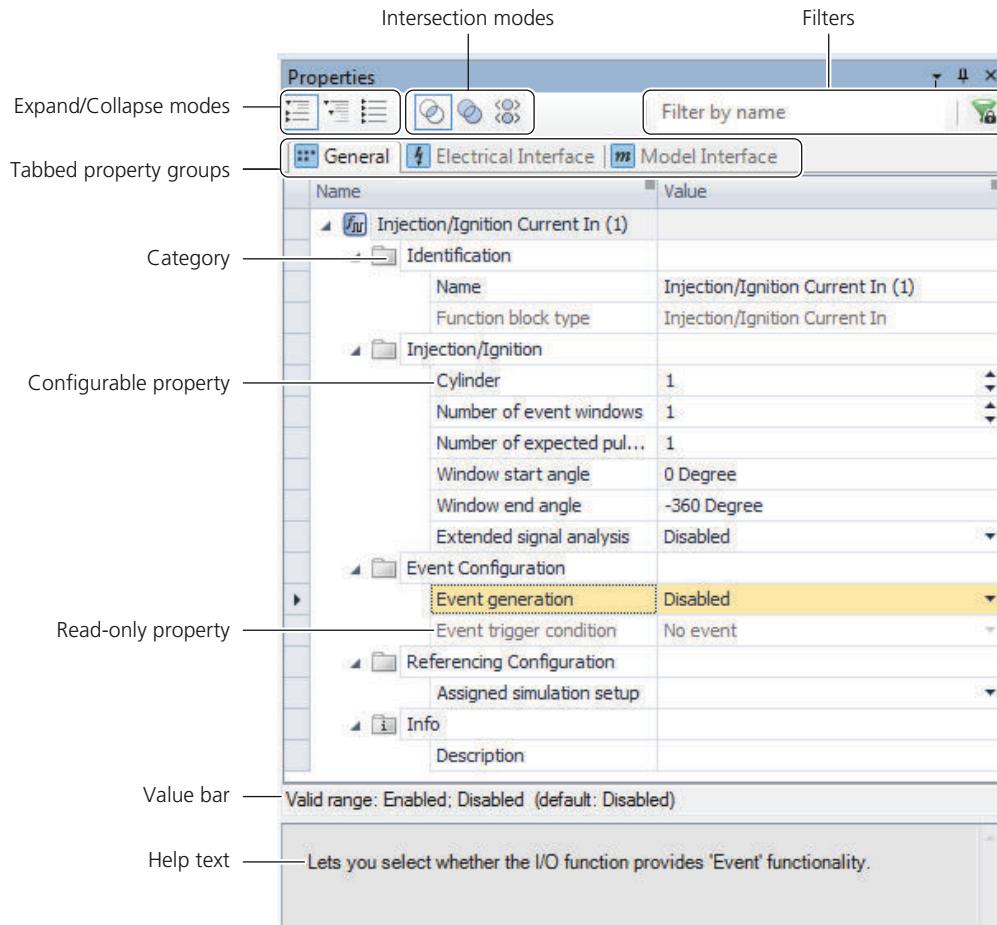
<b>Where to go from here</b>	<b>Information in this section</b>												
	<table><tr><td>Introduction to the Properties Browser.....</td><td>131</td></tr><tr><td>Configuring Properties in the Properties Browser.....</td><td>134</td></tr><tr><td>Expanding and Collapsing Categories in the Properties Browser.....</td><td>138</td></tr><tr><td>Changing the Intersection Modes in the Properties Browser.....</td><td>140</td></tr><tr><td>Using Filters in the Properties Browser.....</td><td>143</td></tr><tr><td>Use Scenarios for Modes and Filters in the Properties Browser.....</td><td>145</td></tr></table>	Introduction to the Properties Browser.....	131	Configuring Properties in the Properties Browser.....	134	Expanding and Collapsing Categories in the Properties Browser.....	138	Changing the Intersection Modes in the Properties Browser.....	140	Using Filters in the Properties Browser.....	143	Use Scenarios for Modes and Filters in the Properties Browser.....	145
Introduction to the Properties Browser.....	131												
Configuring Properties in the Properties Browser.....	134												
Expanding and Collapsing Categories in the Properties Browser.....	138												
Changing the Intersection Modes in the Properties Browser.....	140												
Using Filters in the Properties Browser.....	143												
Use Scenarios for Modes and Filters in the Properties Browser.....	145												

## Introduction to the Properties Browser

<b>Element-specific access to properties</b>	The properties you can access in the Properties Browser depend on the elements that are selected. For example, if a device port is selected, the Properties Browser contains its properties, such as the Port type. If an element has subelements, their properties are also accessible.
--	--

## Overview of the Properties Browser

The following illustration is an example that shows the elements of the Properties Browser and the different property types.



**Expand/Collapse modes** The active expand/collapse mode determines which parts of the hierarchy of elements, categories, and properties are expanded or collapsed. Refer to [Expanding and Collapsing Categories in the Properties Browser](#) on page 138.

**Intersection modes** The active intersection mode determines how properties and elements are grouped. Refer to [Changing the Intersection Modes in the Properties Browser](#) on page 140.

**Filters** You can reduce the number of displayed contents by using different filters. Refer to [Using Filters in the Properties Browser](#) on page 143.

**Tabbed property groups** Depending on the element types you selected, properties might be grouped on different pages, indicated by tabs.

For example, elements and their properties belonging to the electrical interface or the model interface of the signal chain are displayed on the Electrical Interface or Model Interface pages, respectively.

ConfigurationDesk opens the last selected page if you switch to a selection where it is available.

**Category** The property categories help you navigate to the desired property by grouping related properties.

**Configurable property** You can enter or select a value for the property. Press **Enter** or click anywhere outside the property value field to confirm.

**Read-only property** Read-only properties are displayed for information purposes and their values cannot be changed.

Some properties are read-only because their features are disabled by another property or not supported by assigned hardware resources. For example, the Event trigger condition in the illustration above is configurable only if the Event generation property is set to Enabled.

**Value bar** If you select a property, the value bar shows the valid range or the selectable values for it. For some properties, a default value is also shown.

**Help text** Provides information on the selected element or property.

**Customizable table rows and columns** The Name and Value columns and their rows are customizable like the columns of other ConfigurationDesk tables. This means, for example, that you can sort properties by name in ascending order within their categories. For more information, refer to [Customizing Table Rows and Columns](#) on page 154.

### Tip

Use the Clear Sorting context menu command of the column header to remove the sorting of a column.

## Related topics

### Basics

Accessing Elements.....	122
Changing the Intersection Modes in the Properties Browser.....	140
Configuring Properties in the Properties Browser.....	134
Expanding and Collapsing Categories in the Properties Browser.....	138
Using Filters in the Properties Browser.....	143

### Examples

Use Scenarios for Modes and Filters in the Properties Browser.....	145
--	-----

## Configuring Properties in the Properties Browser

---

### Configuring properties with different value types

Selected elements can have properties with different value types that must be configured in different ways.

The value bar provides the valid range of values for all properties. For vector values, the value bar also provides the vector width.

Properties can have one of the following value types:

**String or numeric values** To configure:

1. Click the edit field of the property.
2. Type the string or numerical value into the property's edit field.  
If you need to enter a vector of strings or numeric values, separate them by semicolons.
3. Press **Enter** or click anywhere outside the edit field to confirm.

**List of allowed values** To configure:

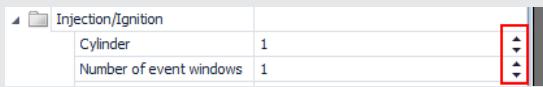
1. Click the edit field of the property.  
A list of valid values appears.
2. Select the desired value from the list.

**Checkboxes** To configure:

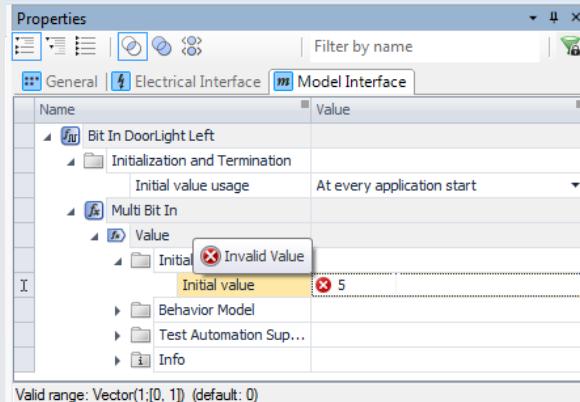
- Select the checkbox to enable the property.  
For some properties, you can select multiple checkboxes from a list.

**Tip**

- Units such as A or V are automatically added to entered values.
- Numeric values without units can also be increased or decreased by using the up and down arrows to the right of the edit field:



- If you enter an invalid value, the Properties Browser notifies you:



To remove the notification and unlock the Properties Browser without making any changes, press **Esc**.

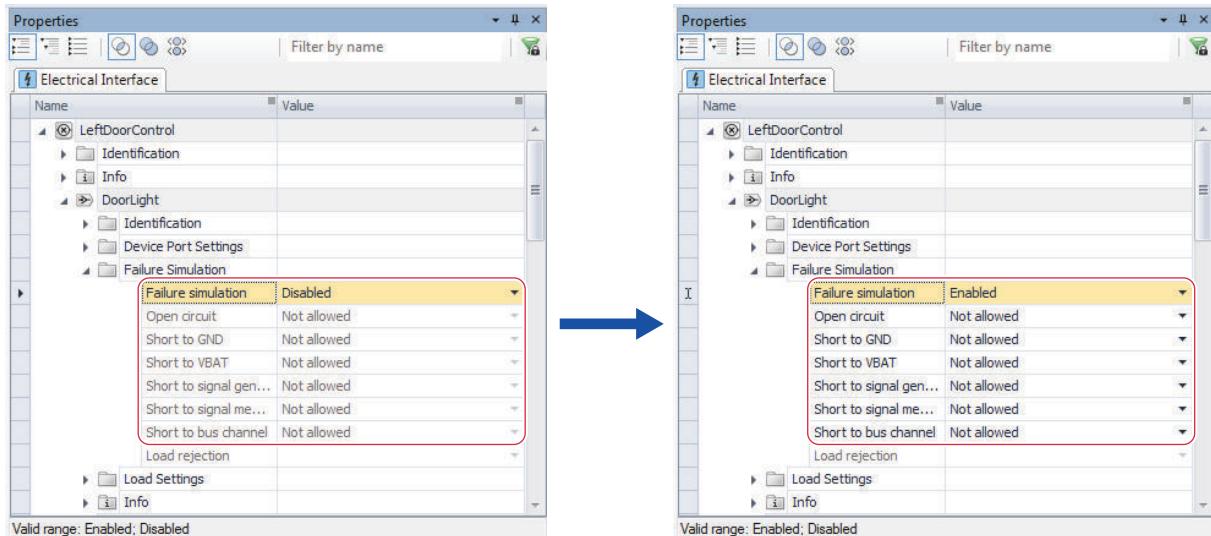
- For some string vectors, you can alternatively enter the numerical values shown in the help text to set one of the available strings.

### Dependencies between property settings

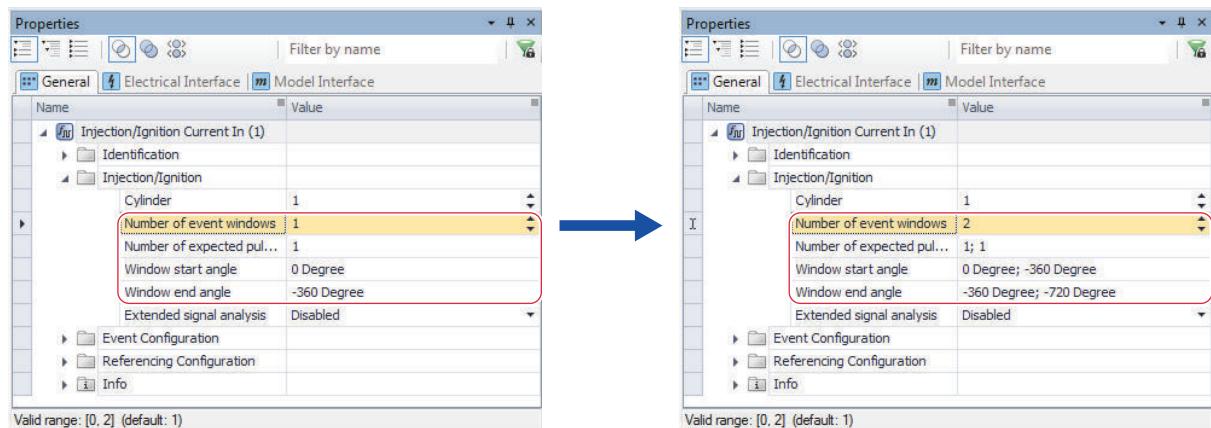
The availability, configurability, value type, or value range of a property might depend on the configuration of one or more other properties.

Examples:

**Making read-only properties configurable** For example, failure classes of a device port are read-only if the Failure simulation property of the device port is set to Disabled. Changing the property value to Enabled makes the failure classes configurable.

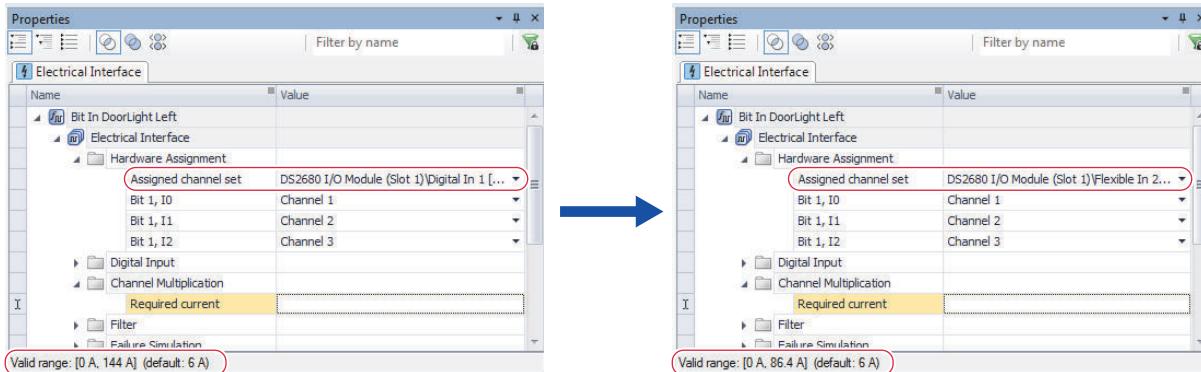


**Turning a numeric property into a vector** For example, many properties of the Injection/Ignition Current In function block depend on the Number of event windows property. Changing the property value to 2 turns multiple properties into vectors.



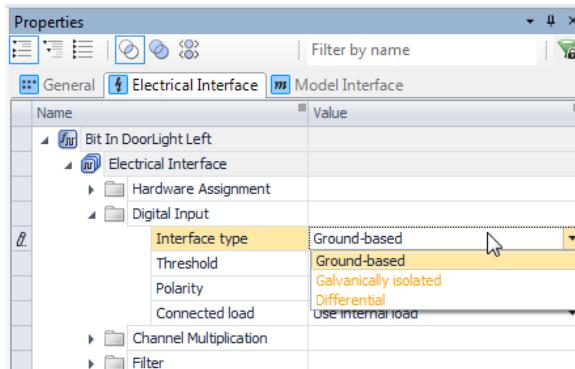
**Value range defined by hardware resources** For example, the value range of the Required current property from the Channel Multiplication category, that is available via the electrical interface of function blocks of several types,

depends on the assigned channel set and channels in the **Hardware Assignment** category.



### Highlighting configuration conflicts

In some property value lists, settings that cause configuration conflicts are displayed in yellow or red.



For more information on conflicts, refer to [Resolving Conflicts](#) on page 677.

### Configuring common properties for multiple elements

You can select multiple elements and set common properties to the same value at once.

#### Tip

The display of common properties depends on the active intersection mode. Refer to [Changing the Intersection Modes in the Properties Browser](#) on page 140.

If you selected multiple elements, the values of common properties are displayed in one of the following ways:

- If the property value is identical for the selected elements, it is displayed as usual.

	<Device Port> [3]
	Identification
	Device Port Settings
Port type	Out

- If there are two different values for the selected elements, both values are listed. For each value, the number of occurrences is shown in parentheses.

Physical attributes (2 Different Values: Unspecified [2], Voltage [1])

- If there are three or more different values for the selected elements, the values are not displayed.

Pins (Multiple Different Values)

- If a property is read-only for some of the selected elements, it is displayed as read-only for all elements of the selection and cannot be configured.

Change the value of the common property to set the new value for all selected elements at once.

## Related topics

### Basics

Accessing Elements.....	122
Changing the Intersection Modes in the Properties Browser.....	140
Expanding and Collapsing Categories in the Properties Browser.....	138
Introduction to the Properties Browser.....	131
Using Filters in the Properties Browser.....	143

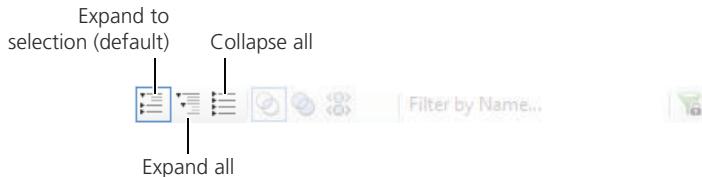
### Examples

Use Scenarios for Modes and Filters in the Properties Browser.....	145
--	-----

## Expanding and Collapsing Categories in the Properties Browser

### Selecting an expand/collapse mode

You can select one of three different expand/collapse modes to influence how the element/category hierarchy of selected elements is displayed in the Properties Browser. The active mode is indicated by a blue frame:

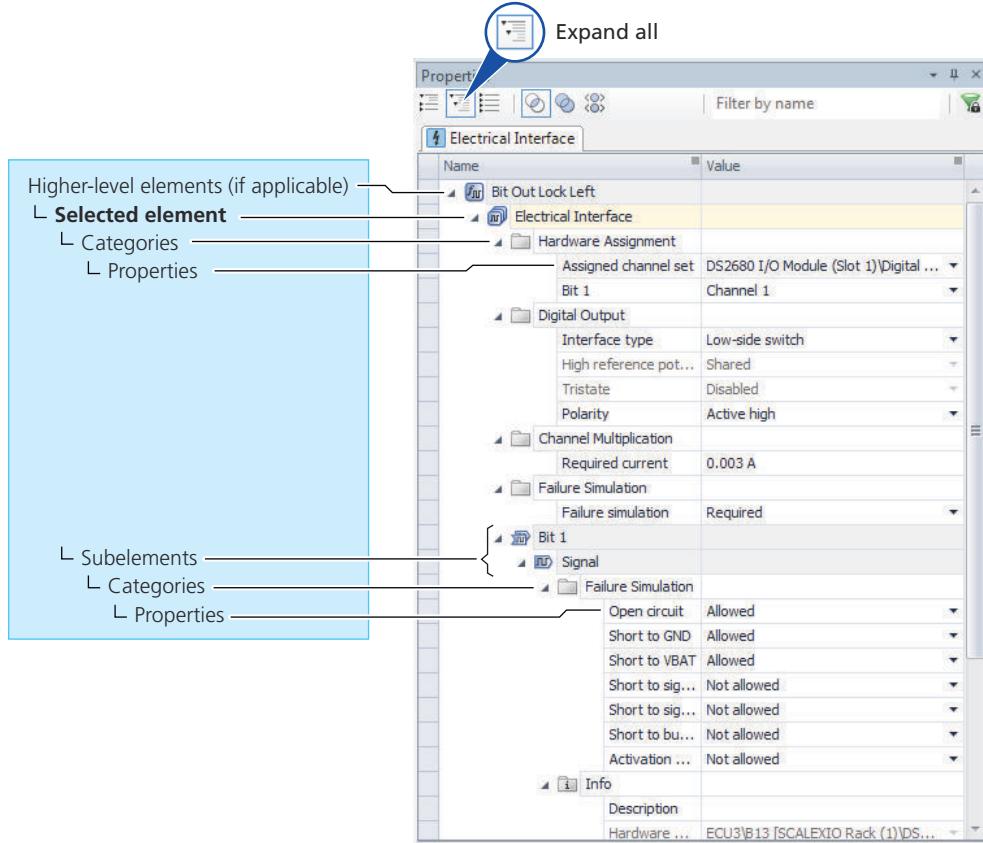


If you switch the expand/collapse mode, your setting is active until you close ConfigurationDesk. Next time you start ConfigurationDesk, the default mode is active.

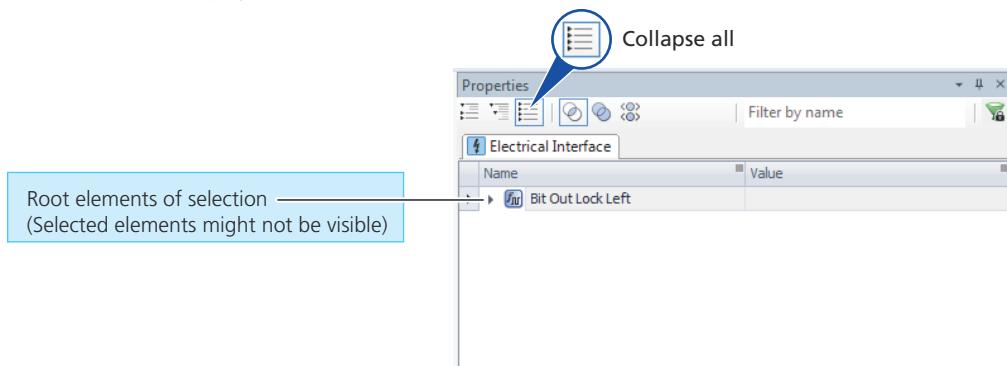
Click the active mode again to undo manual expand/collapse actions.

### Available expand/collapse modes

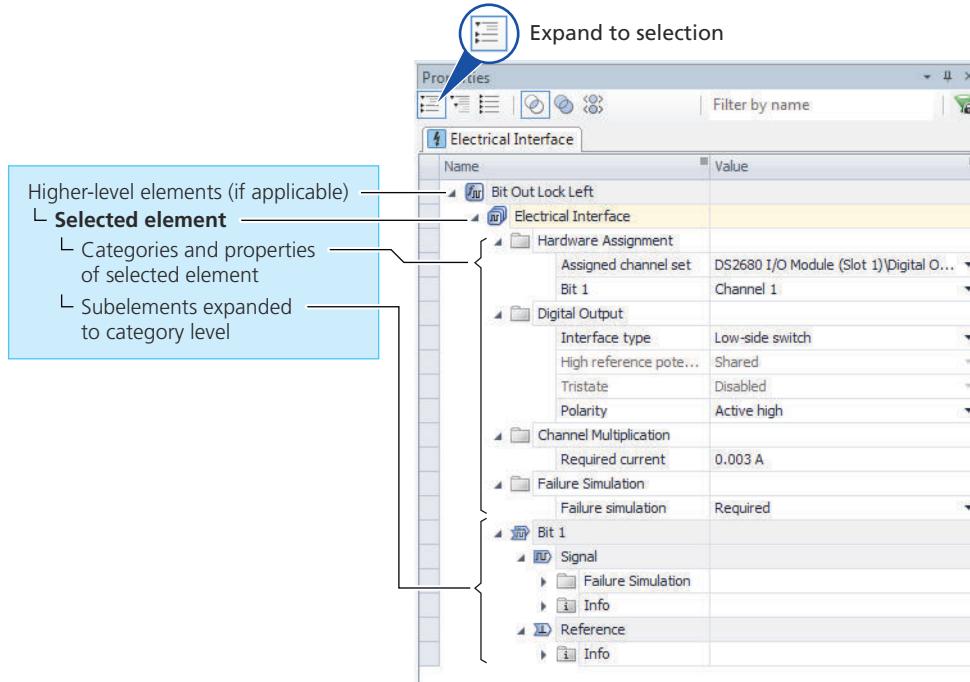
**Expand all** If this mode is active, the hierarchy of elements, categories, and properties of selected elements and their higher-level elements or subelements is completely expanded:



**Collapse all** If this mode is active, the hierarchy of elements, categories, and properties of selected elements and their higher-level elements or subelements is completely collapsed. Only the root level elements of your selection are displayed:



**Expand to selection (default)** If this mode is active, the complete hierarchy of elements, categories, and properties is expanded only for selected elements. Higher-level elements or subelements are expanded to the category level:



## Related topics

### Basics

Configuring Properties in the Properties Browser.....	134
Introduction to the Properties Browser.....	131

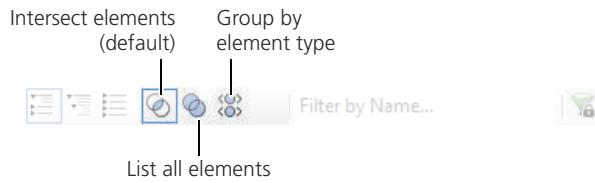
### Examples

Use Scenarios for Modes and Filters in the Properties Browser.....	145
--	-----

## Changing the Intersection Modes in the Properties Browser

### Selecting an intersection mode

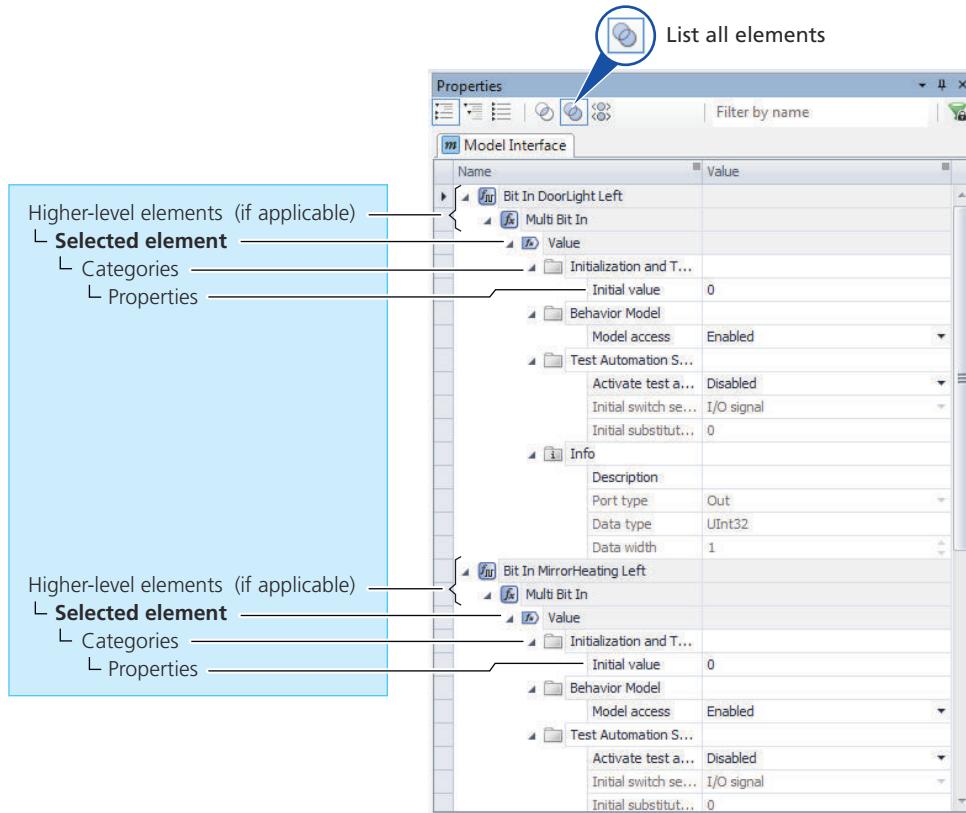
You can select one of three different intersection modes to influence how selected elements and their properties are grouped in the Properties Browser. The currently active mode is indicated by a blue frame:



If you switch the intersection mode, your setting is active until you close ConfigurationDesk. Next time you start ConfigurationDesk, the default mode is active.

#### Available intersection modes

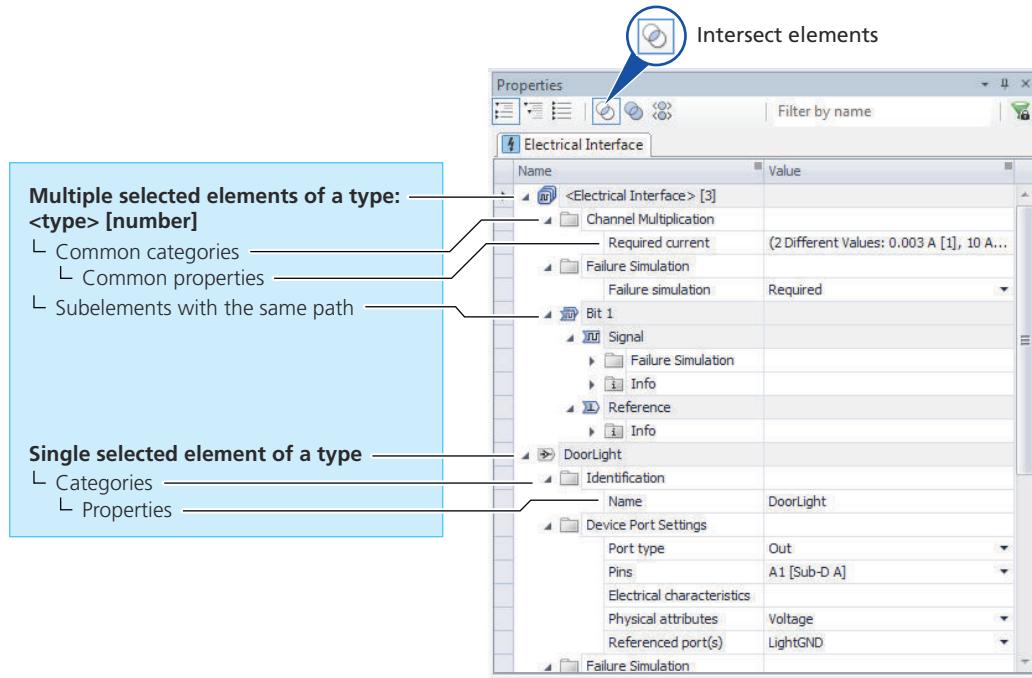
**List all elements** If this mode is active, all elements and properties of your selection are displayed (with complete element hierarchy for blocks in the signal chain):



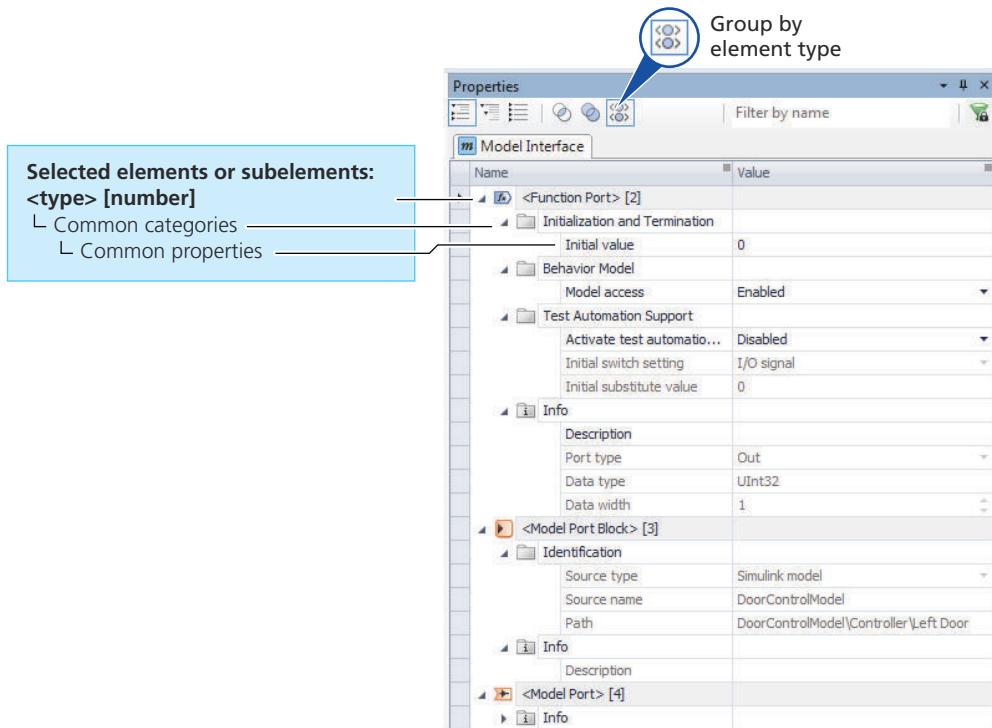
**Intersect elements (default)** If this mode is active:

- Multiple selected elements of the same type are grouped and only their common properties are displayed.
- If the grouped elements of the same type have subelements with the same path, the subelements and their common properties are also displayed.

- Single selected elements of a type are displayed as in the List all elements mode.



**Group by element type** If this mode is active, all selected elements are grouped by their type and their common properties are displayed. Subelements are also displayed and grouped by type, regardless of the hierarchy path:



**Related topics****Basics**

Configuring Properties in the Properties Browser.....	134
Introduction to the Properties Browser.....	131

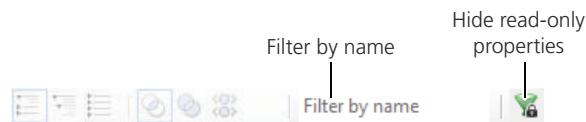
**Examples**

Use Scenarios for Modes and Filters in the Properties Browser.....	145
--	-----

## Using Filters in the Properties Browser

### Filtering the displayed properties

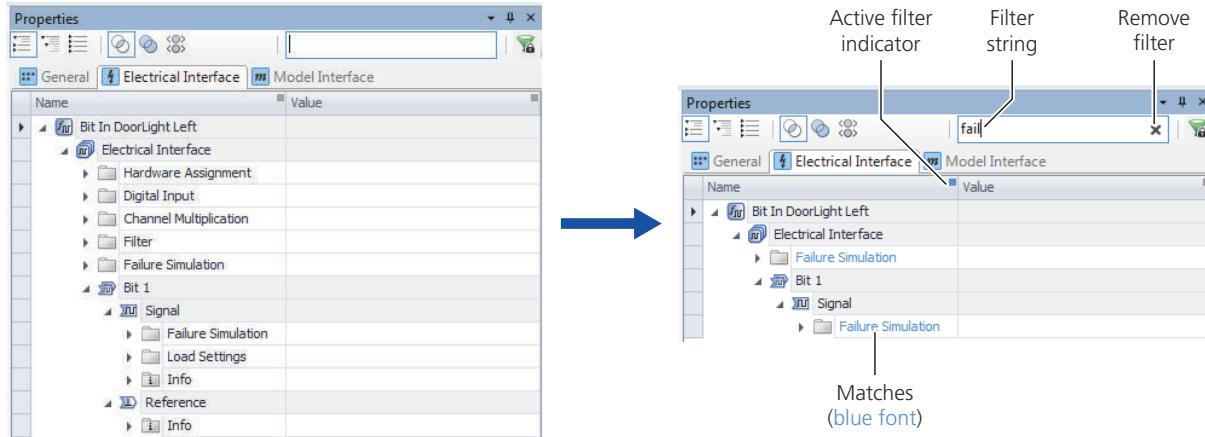
You can reduce the amount of contents displayed for your selection by using filters.



If you use a filter, it is active until you disable it or close ConfigurationDesk.

### Available filters

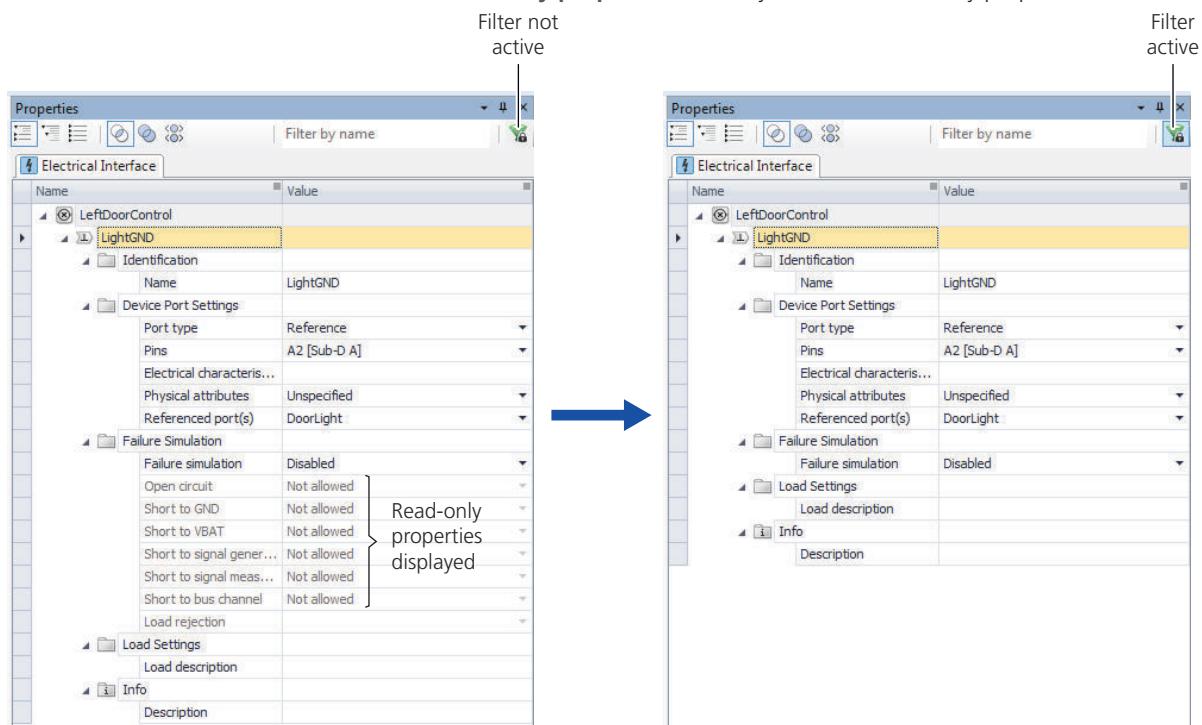
**Filter by name** Lets you filter for elements, categories, and properties whose name contains a specific character or string.



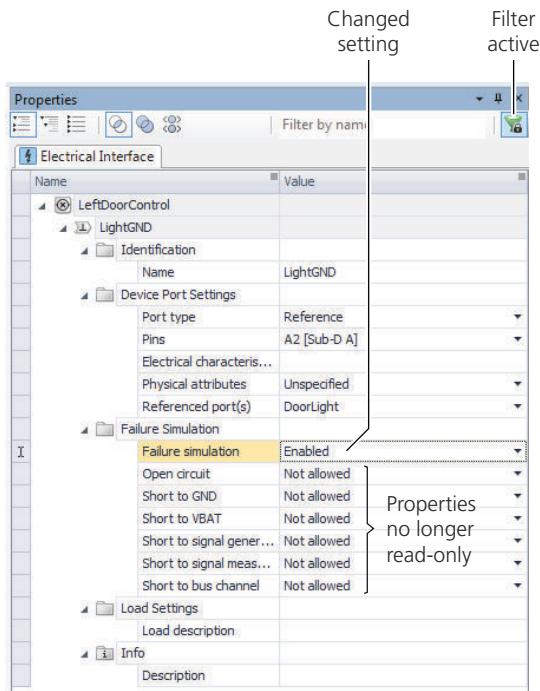
**Note**

- The filter is case-insensitive.
- Higher-level hierarchy elements of matches are also displayed.
- All categories, properties and subelements of matching elements are displayed.
- When you switch the property group page, the same filter setting applies.

**Hide read-only properties** Lets you hide all read-only properties.



If the filter is active and a read-only property becomes configurable when you change the setting of a different property, it is displayed.



## Related topics

### Basics

Configuring Properties in the Properties Browser.....	134
Introduction to the Properties Browser.....	131

### Examples

Use Scenarios for Modes and Filters in the Properties Browser.....	145
--	-----

## Use Scenarios for Modes and Filters in the Properties Browser

### Introduction

The following examples illustrate how to access elements in ConfigurationDesk and use the different modes and filters of the Properties Browser in specific configuration scenarios.

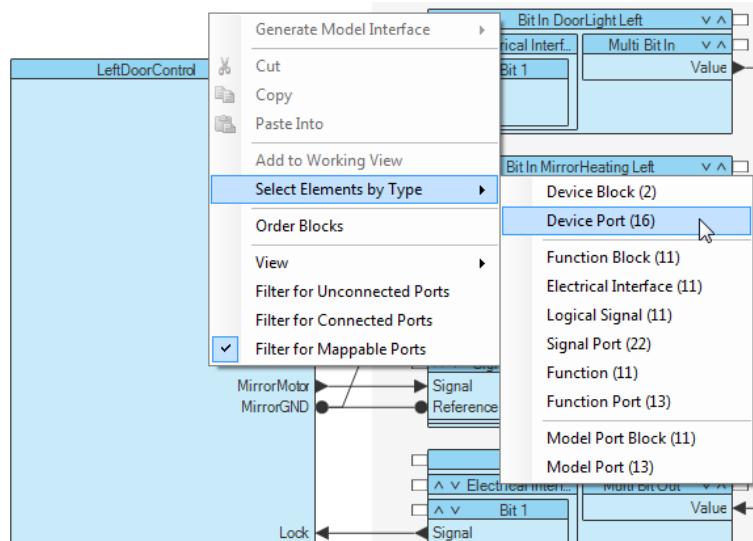
The default mode and filter settings of the Properties Browser are assumed as the starting point:

- The **Expand to selection** expand/collapse mode is active.
- The **Intersect elements** intersection mode is active.
- No filters are active.

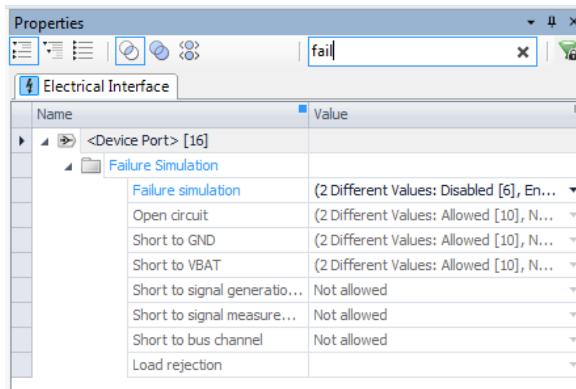
### Configuring failure simulation settings for all device ports

In a scenario involving failure simulation with different failure classes, you can compare and set the allowed failure classes for all device ports used in a ConfigurationDesk application as follows:

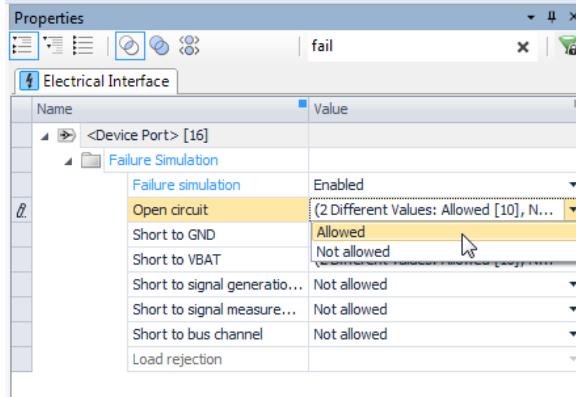
1. In the Global working view, select all device ports by using the Select Elements by Type command:



2. In the Properties Browser, use the Filter by name filter to focus on the failure simulation properties:



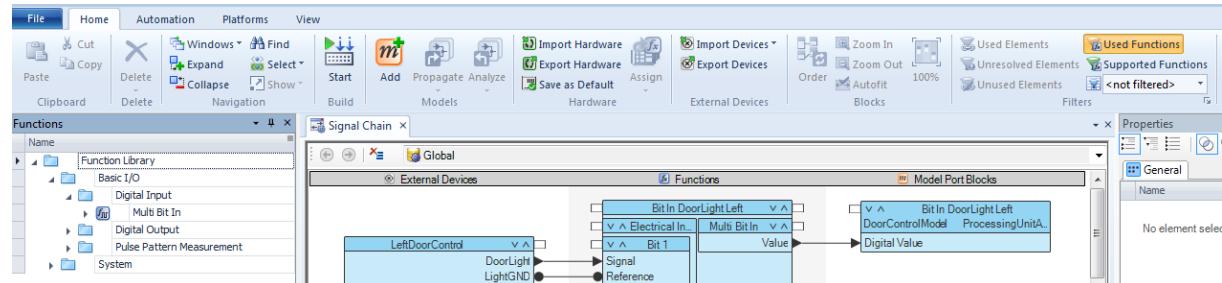
3. Apply the desired failure simulation settings:



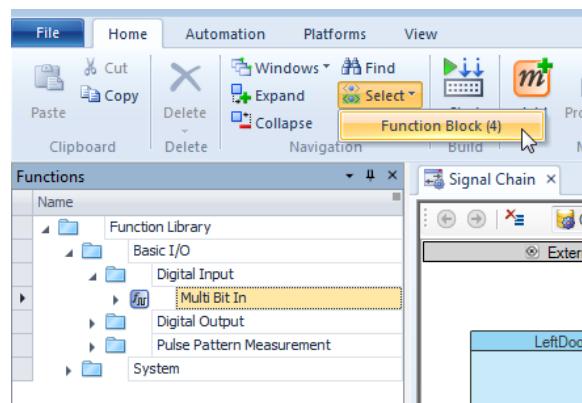
### Configuring initialization settings for all function blocks of a function block type

In a ConfigurationDesk application containing function blocks of various function block types, you can compare and set the initialization settings for function blocks of specific function block types. For example, you can access the initialization settings of all function blocks of the Multi Bit In function block type from the Digital Input function class as follows:

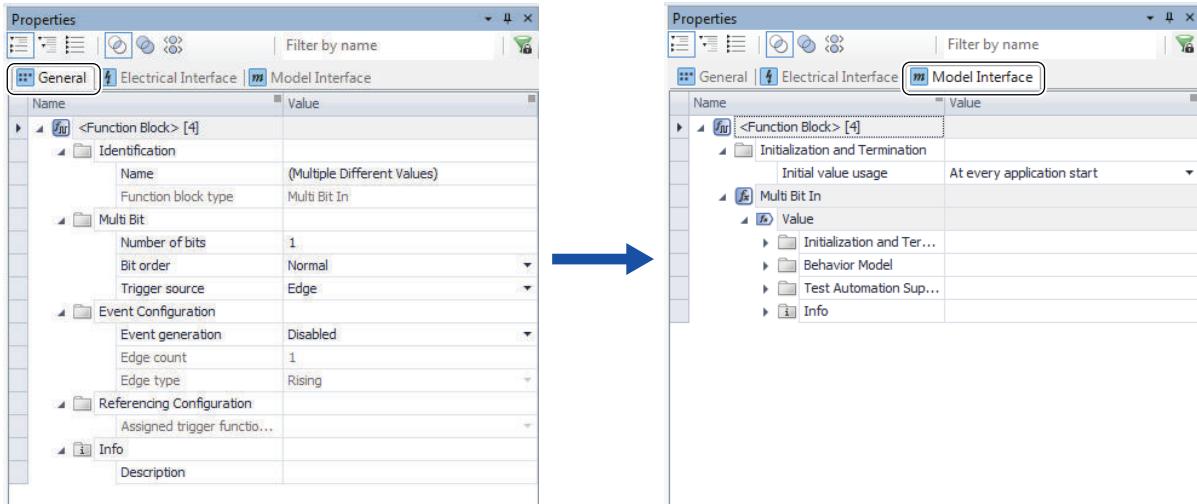
1. In the Function Browser, activate the Used Functions filter on the Home ribbon and expand the Basic I/O – Digital Input function class:



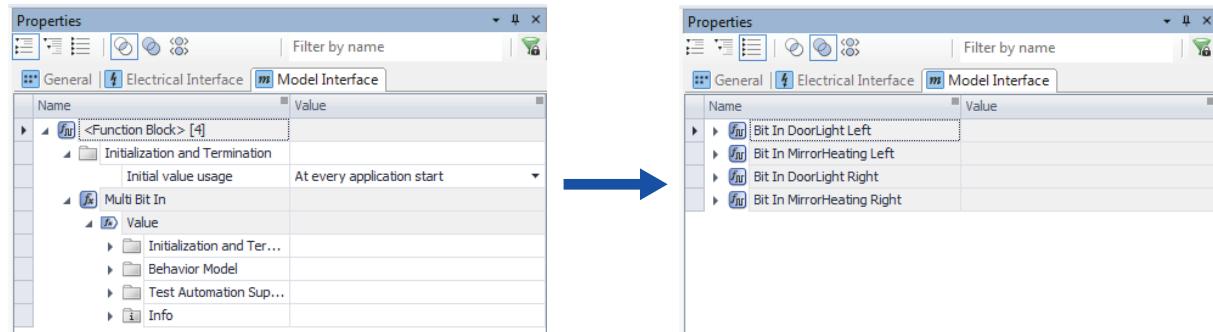
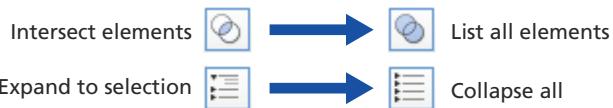
2. Click the Multi Bit In function block type class and use the Select submenu on the Home ribbon to select all instantiated function blocks of the type:



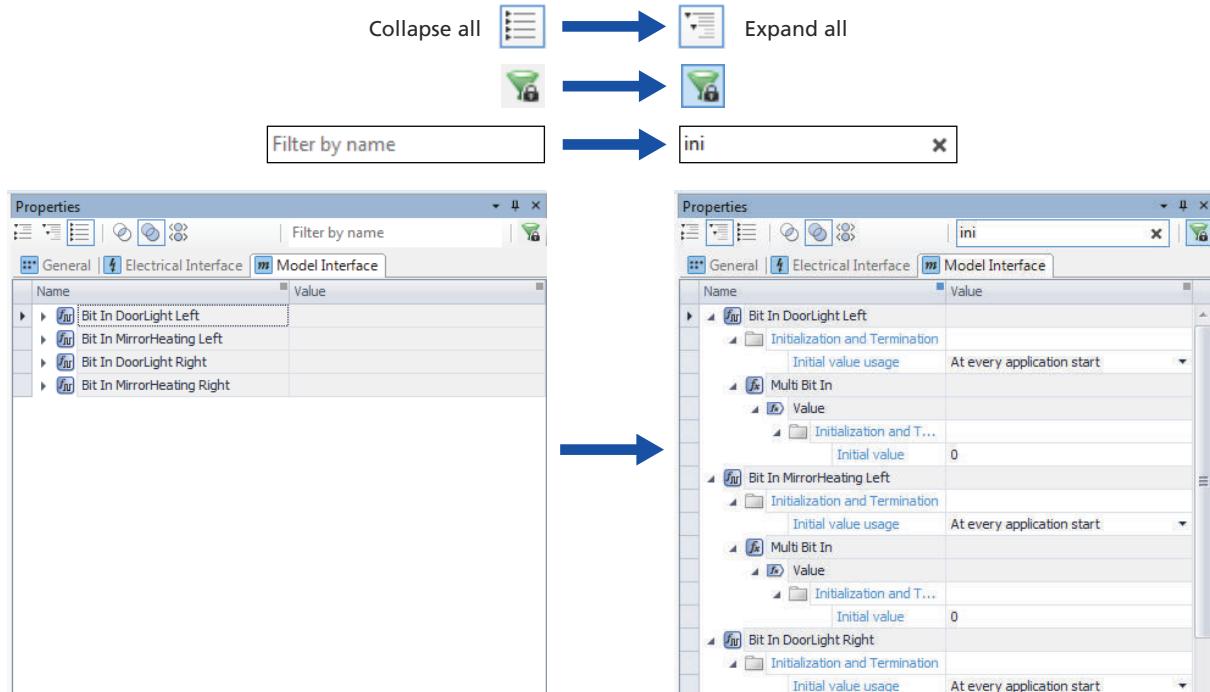
3. In the Properties Browser, switch to the Model Interface page.



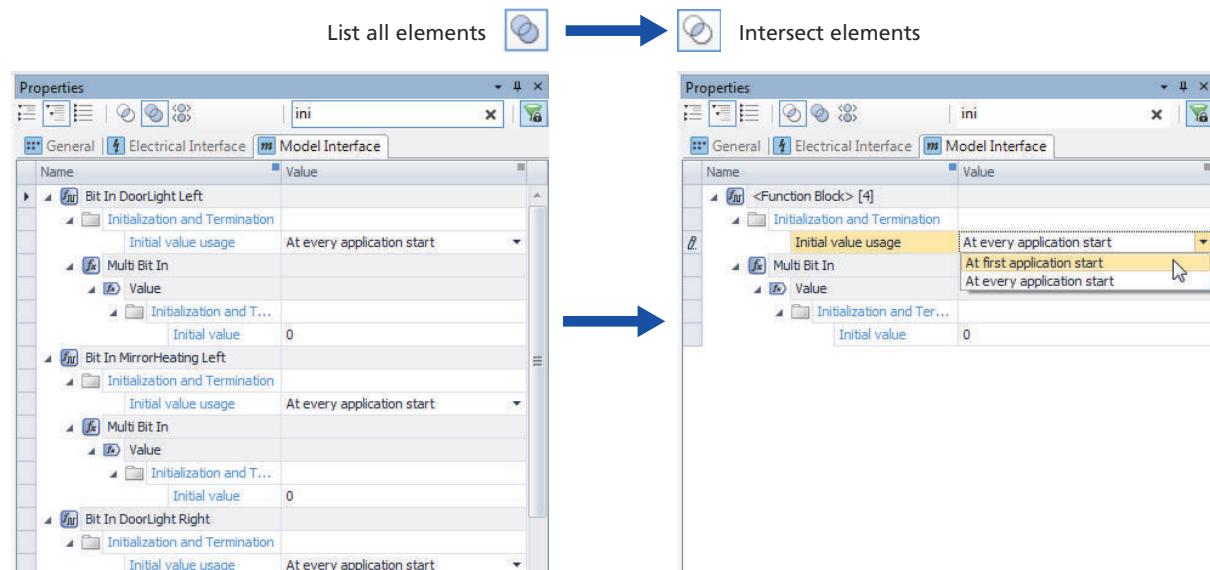
4. To get an overview of which function blocks you selected, switch to the List all elements intersection mode and the Collapse all expand/collapse mode:



5. To compare the initialization settings, switch to the Expand all expand/collapse mode. To reduce the number of displayed properties, activate the Hide read-only properties filter and enter the string 'ini' in the Filter by name field:



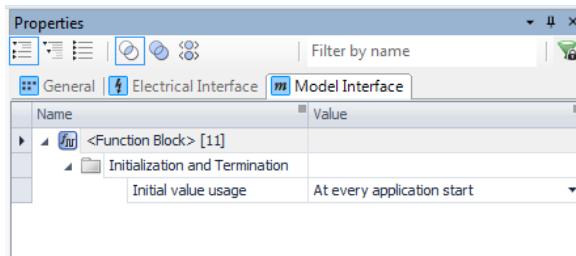
6. To apply the same initialization settings to the selected function blocks and their function ports, switch back to the Intersect elements intersection mode:



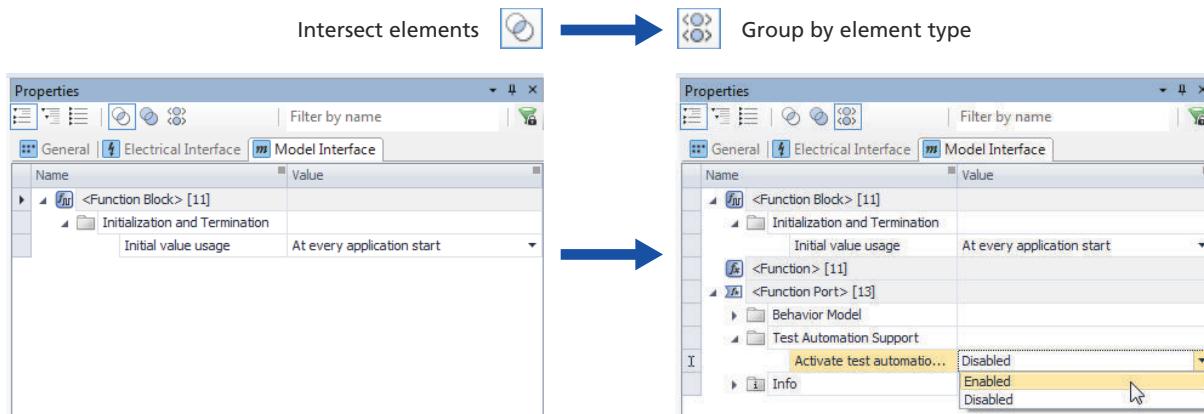
### Configuring test automation settings for multiple function ports

You can set the test automation settings for multiple function ports as follows:

1. Select all the function blocks in your ConfigurationDesk application and switch to the Model Interface page in the Properties Browser. Note that in the following illustration from the Results application in the Tutorial demo project, only one common property and no common subelements are available in the Intersect elements intersection mode because of different hierarchy paths:



2. Switch to the Group by element type intersection mode. The common subelements appear, including the function ports with the Test Automation Support property category where you can enable or disable the test automation support:



### Related topics

#### Basics

Changing the Intersection Modes in the Properties Browser.....	140
Configuring Properties in the Properties Browser.....	134
Expanding and Collapsing Categories in the Properties Browser.....	138
Introduction to the Properties Browser.....	131
Using Filters in the Properties Browser.....	143

#### HowTos

How to Select Elements of the Same Type.....	123
--	-----

# Using Tables to Access and Configure Elements

## Objective

ConfigurationDesk offers various tables that provide access to a specific subset of elements and properties of the active ConfigurationDesk application.

## Where to go from here

## Information in this section

Introduction to Tables.....	151
Customizing Table Rows and Columns.....	154

## Introduction to Tables

### Using tables to manage elements and their properties

Some elements of a ConfigurationDesk application are managed in tables. Tables let you access and modify specific elements or element hierarchies and some element properties can be directly configured in the tables.

The following illustration shows device connector elements in the External Device Connectors table:

Table name	Property column
Name	Port
LeftDoorControl	MirrorHeating
Sub-D A	Lock
A3	LightGND
A6	MirrorMotor
A2	DoorLight
A4	Unlock
A1	MirrorGND
A7	GND
A5	
A8	
RightDoorControl	DoorLight
Sub-D B	LightGND
B1	MirrorHeating
B2	MirrorMotor
B3	MirrorGND
B4	Unlock
B5	Lock
B7	GND
B6	
B8	

Elements shown in table  
(here: device connectors)

**Accessing tables**

You can access tables like other types of panes in ConfigurationDesk:

- Some tables are part of specific view sets by default. The Task Configuration table, for example, is the central pane of the Tasks view set. For more information on view sets and an overview of the panes available in each view set (default settings), refer to [Overview of the User Interface of ConfigurationDesk](#) on page 50.
- Tables can be opened and customized like other panes. Refer to [Customizing View Sets](#) on page 57.
- There are some specific customizing features for table rows and columns. Refer to [Customizing Table Rows and Columns](#) on page 154.
- Some tables can be opened from the context menu of specific elements to show these elements. Refer to [How to Show Elements in a Different Pane](#) on page 125.

**Overview of Tables**

<b>Table</b>	<b>Purpose</b>	<b>View Set</b>
Build Configuration	To create build configuration sets and to configure build settings, for example, build options, or the build and download behavior.	Build
Bus Access Requests	To access bus access requests of a ConfigurationDesk application and assign them to bus accesses.	Buses
Bus Configuration Function Ports	To access and configure function ports and event ports of bus configurations.	Buses
Bus Configurations	To access and configure bus configurations of a ConfigurationDesk application.	Buses
Bus Manipulation Features	To access and configure PDU-related and signal-related bus configuration features of a ConfigurationDesk application for manipulation purposes.	Buses
Bus Inspection Features	To access and configure PDU-related and signal-related bus configuration features of a ConfigurationDesk application for inspection purposes.	Buses
Bus Simulation Features	To access and configure PDU-related and signal-related bus configuration features of a ConfigurationDesk application for simulation purposes.	Buses
Executable Application	To model executable applications (i.e., real-time applications) and the tasks used in them.	Tasks and Multiple Models <sup>1)</sup>
External Device Configuration	To access and configure the most important properties of device topology elements via a table.	Signal Chain <sup>1)</sup>
External Device Connectors	To specify the representation of the physical connectors of your external device including the device pin assignment.	Signal Chain <sup>1)</sup>
Model Communication Package	To create and configure model communication packages which are used for model communication in multimodel applications.	Multiple Models
Pins and External Wiring	To access the external wiring information.	Signal Chain <sup>1)</sup>

Table	Purpose	View Set
Processing Resource Assignment	To configure and inspect the processing resources in an executable application. This table is useful especially for multi-processing-unit applications.	Multiple Models
Task Configuration	To configure the tasks of an executable application.	Tasks

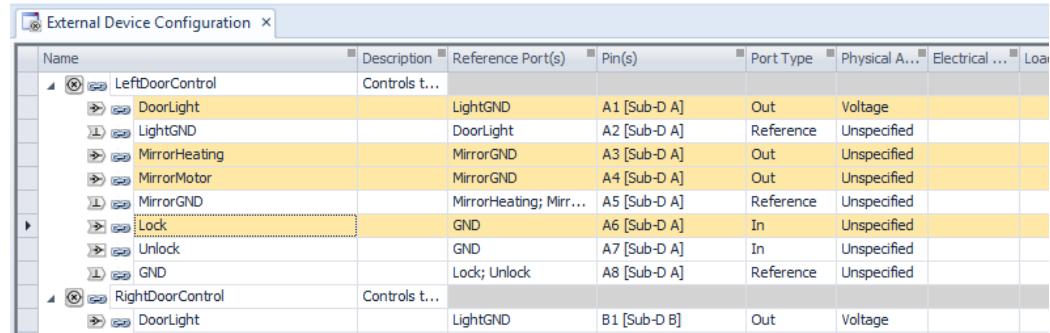
<sup>1)</sup> via ribbon: Home – Navigation – Windows.

**Configuring properties in tables** In general, tables offer the same edit fields for property values as the Properties Browser (refer to [Configuring Properties in the Properties Browser](#) on page 134).

However, there is a configuration feature only supported in tables:

**Using the Fill commands to configure multiple elements** You can set properties to the same value at once for multiple selected elements:

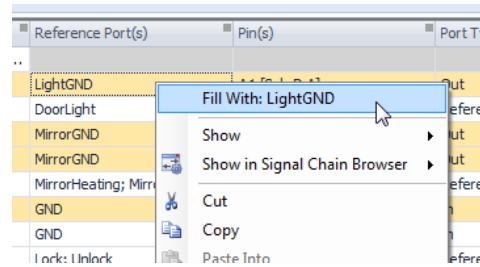
1. In a table, select the elements you want to configure.



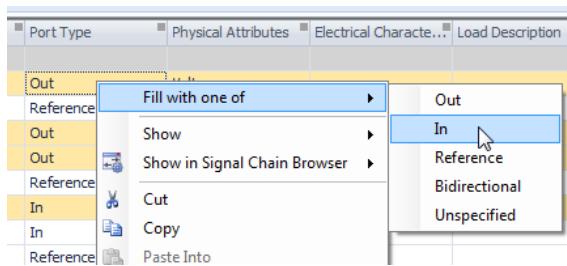
The screenshot shows a table titled "External Device Configuration" with columns: Name, Description, Reference Port(s), Pin(s), Port Type, Physical A..., Electrical ..., and Load. There are two main sections: "LeftDoorControl" and "RightDoorControl". Under "LeftDoorControl", rows include "DoorLight", "LightGND", "MirrorHeating", "MirrorMotor", "MirrorGND", and "Lock". Under "RightDoorControl", rows include "DoorLight". The "Lock" row is highlighted with a yellow selection bar. The "Pin(s)" column for "Lock" contains "GND". The "Port Type" column for "Lock" is "In". The "Electrical ..." column for "Lock" is "Unspecified". The "Load" column for "Lock" is empty.

2. Depending on the property you want to configure, there are two different variants of the Fill command:

- If the property supports various values, such as strings or numerical values, the Fill With: command lets you transfer a property value from one element to the other selected elements. Right-click the table cell containing the property value you want to transfer to all the selected elements and use the Fill With command from the context menu.



- If the property has a limited number of supported values, such as a list of values, the Fill with one of command lets you select a value from a list of available values. Right-click the property of one of the selected elements and select a value.



The properties are set to the same value for all the selected elements.

Reference Port(s)	Pin(s)	Port Type
LightGND	A1 [Sub-D A]	In
DoorLight; MirrorHeating; Mi...	A2 [Sub-D A]	Reference
LightGND	A3 [Sub-D A]	In
LightGND	A4 [Sub-D A]	In
	A5 [Sub-D A]	Reference
LightGND	A6 [Sub-D A]	In

## Related topics

### Basics

Customizing Table Rows and Columns.....	154
Customizing View Sets.....	57
Overview of the User Interface of ConfigurationDesk.....	50

### HowTos

How to Show Elements in a Different Pane.....	125
---	-----

## Customizing Table Rows and Columns

### Customizing tables

In general, tables are a type of pane that can be customized like other panes of view sets. Refer to [Customizing View Sets](#) on page 57.

Additionally, ConfigurationDesk offers some specific customization features for table rows and columns. Most table column customizations are automatically saved so that the order of columns remains the same after opening a different project, for example.

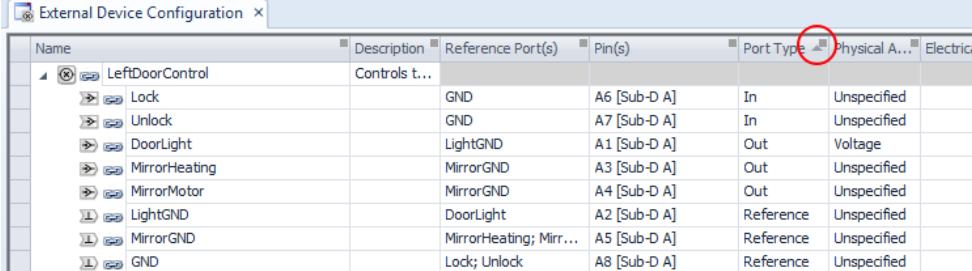
#### Note

Table columns are not reset to their default settings if you reset a view set. You have to undo customizations manually, if necessary.

## Sorting table rows

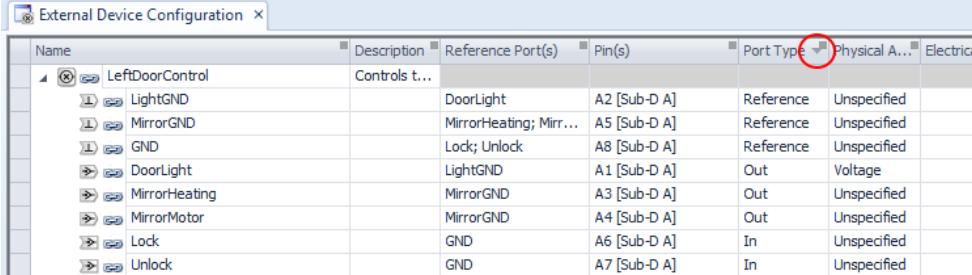
You can sort the rows in a table in ascending or descending order according to the entries in specific columns.

Click a column header to sort the table rows according to the column entries in ascending order. This is indicated by an up arrow:



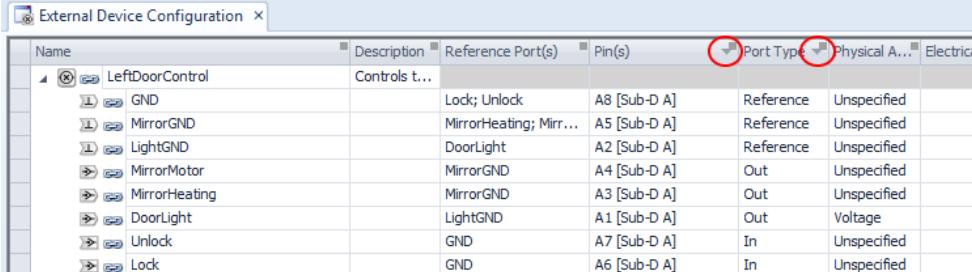
Name	Description	Reference Port(s)	Pin(s)	Port Type	Physical A...	Electrical
LeftDoorControl	Controls t...					
Lock		GND	A6 [Sub-D A]	In	Unspecified	
Unlock		GND	A7 [Sub-D A]	In	Unspecified	
DoorLight		LightGND	A1 [Sub-D A]	Out	Voltage	
MirrorHeating		MirrorGND	A3 [Sub-D A]	Out	Unspecified	
MirrorMotor		MirrorGND	A4 [Sub-D A]	Out	Unspecified	
LightGND		DoorLight	A2 [Sub-D A]	Reference	Unspecified	
MirrorGND		MirrorHeating; Mirr...	A5 [Sub-D A]	Reference	Unspecified	
GND		Lock; Unlock	A8 [Sub-D A]	Reference	Unspecified	

Click the same column header to sort the rows in descending order. This is indicated by a down arrow:



Name	Description	Reference Port(s)	Pin(s)	Port Type	Physical A...	Electrical
LeftDoorControl	Controls t...					
LightGND		DoorLight	A2 [Sub-D A]	Reference	Unspecified	
MirrorGND		MirrorHeating; Mirr...	A5 [Sub-D A]	Reference	Unspecified	
GND		Lock; Unlock	A8 [Sub-D A]	Reference	Unspecified	
DoorLight		LightGND	A1 [Sub-D A]	Out	Voltage	
MirrorHeating		MirrorGND	A3 [Sub-D A]	Out	Unspecified	
MirrorMotor		MirrorGND	A4 [Sub-D A]	Out	Unspecified	
Lock		GND	A6 [Sub-D A]	In	Unspecified	
Unlock		GND	A7 [Sub-D A]	In	Unspecified	

Press **Shift** and click a different column header to also sort by the entries of that column. This applies to rows whose entries are identical in the column by which you first sorted:



Name	Description	Reference Port(s)	Pin(s)	Port Type	Physical A...	Electrical
LeftDoorControl	Controls t...					
GND		Lock; Unlock	A8 [Sub-D A]	Reference	Unspecified	
MirrorGND		MirrorHeating; Mirr...	A5 [Sub-D A]	Reference	Unspecified	
LightGND		DoorLight	A2 [Sub-D A]	Reference	Unspecified	
MirrorMotor		MirrorGND	A4 [Sub-D A]	Out	Unspecified	
MirrorHeating		MirrorGND	A3 [Sub-D A]	Out	Unspecified	
DoorLight		LightGND	A1 [Sub-D A]	Out	Voltage	
Unlock		GND	A7 [Sub-D A]	In	Unspecified	
Lock		GND	A6 [Sub-D A]	In	Unspecified	

Use the column header context menu command Clear Sorting to remove the sorting of a specific column or Apply Original Order to remove any sorting:



**Note**

The sorting applies to elements within their hierarchy levels. An element hierarchy is not modified by this:

Sorted devices	
Name	Description
RightDoorControl	5
LightGND	
MirrorMotor	
Lock	
Unlock	
GND	
DoorLight	2
MirrorHeating	4
MirrorGND	8
LeftDoorControl	6
DoorLight	
MirrorHeating	
MirrorGND	
Unlock	
GND	
MirrorMotor	1
Lock	3
LightGND	7

Column sorted in ascending order

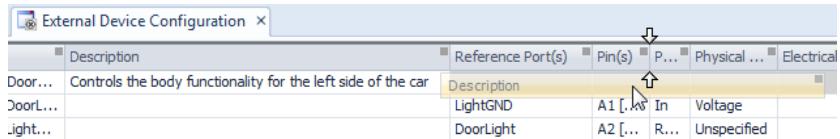
Sorted ports within device

Sorted ports within device

**Adjusting table columns**

There are various ways to adjust the columns of a table:

- You can click a column header and drag the column to a different position.



Description	Reference Port(s)	Pin(s)	P...	Physical ...	Electrical
Door... Controls the body functionality for the left side of the car	LightGND	A1 [...	In	Voltage	
DoorL...	DoorLight	A2 [...	R...	Unspecified	
ight...					

- You can adjust the width of a column using the double-headed arrow into which the pointer turns when you move it between two column headers.



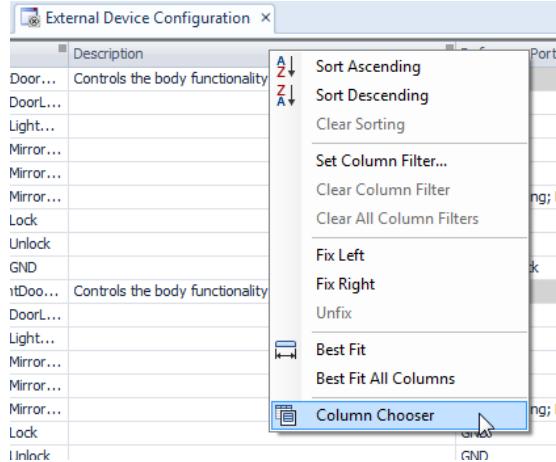
You can also use the Best Fit/Best Fit All Columns commands from the context menu of column headers to optimize the width of one column or all columns according to their content.

- You can fix columns to the left or right side of a table using the Fix Left/Fix Right command from the context menu of a column header.

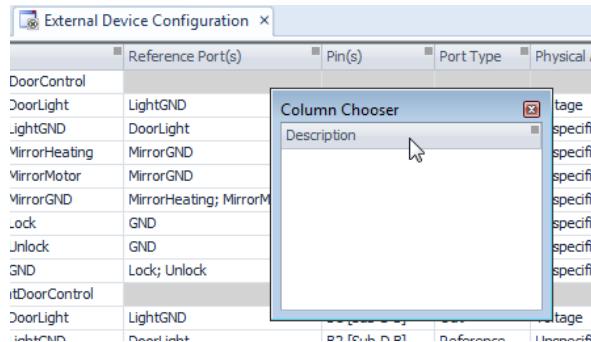
## Removing table columns

If a table contains columns you do not require, you can remove them for a better overview with the **Column Chooser**:

1. Open the Column Chooser via the context menu of a column header.



2. Drag a column header to the Column Chooser to remove it from the table.



You can add removed columns to the table again later by dragging the column header to the desired position from the Column Chooser.

### Tip

You can also drag a column anywhere until an X symbol appears to remove it without opening the Column Chooser.

## Filtering table columns

You can filter one or more columns by using a filter string. This functionality is also available in different types of panes such as browsers. For instructions, refer to [How to Filter Columns by Text and Regular Expressions](#) on page 167.

## Related topics

### Basics

Customizing View Sets.....	57
Introduction to Tables.....	151

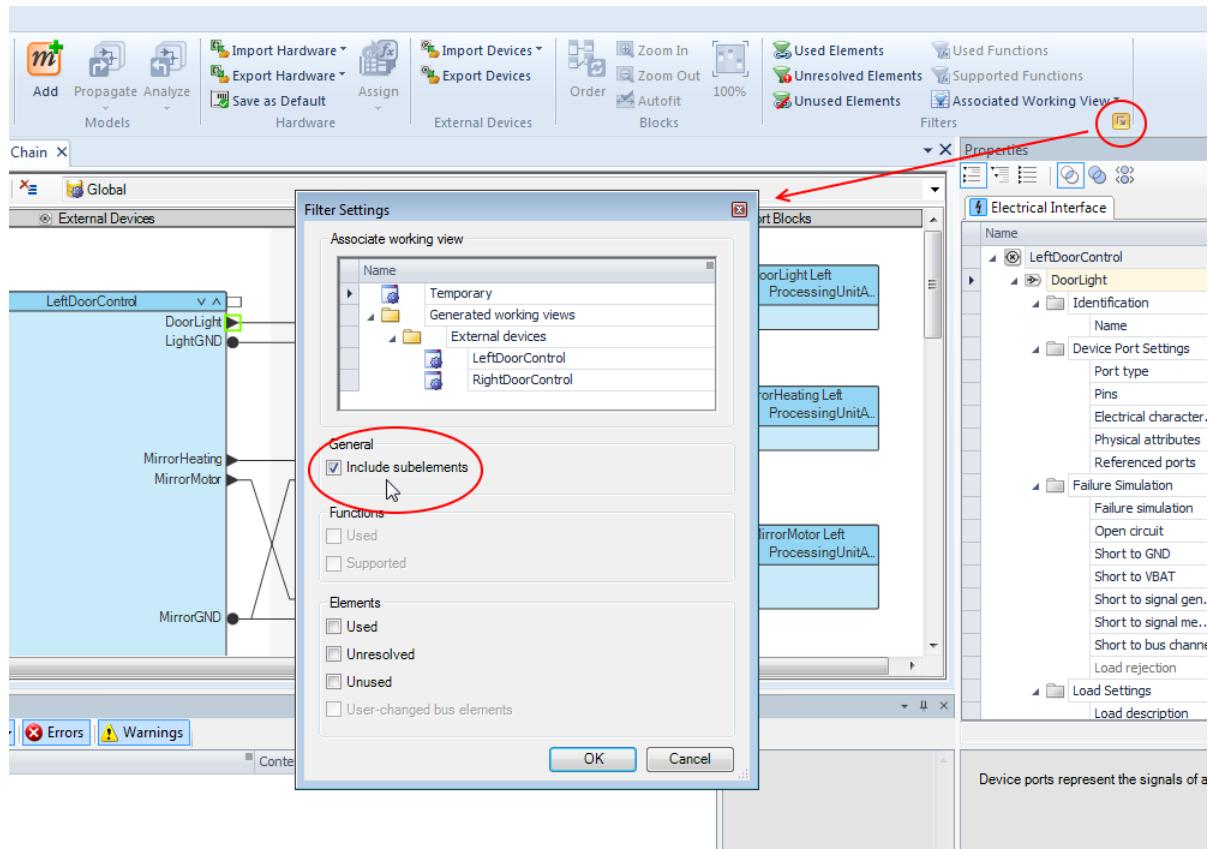
# Using Display Filters

<b>Objective</b>	ConfigurationDesk provides features that let you filter the elements displayed in a pane.										
<b>Where to go from here</b>	<b>Information in this section</b>										
	<table><tr><td>Basics on Display Filters.....</td><td>158</td></tr><tr><td>Available Display Filters.....</td><td>160</td></tr><tr><td>How to Filter for Specific Signal Chain Elements.....</td><td>162</td></tr><tr><td>How to Filter for Signal Chain Elements from Associated Working Views.....</td><td>165</td></tr><tr><td>How to Filter Columns by Text and Regular Expressions.....</td><td>167</td></tr></table>	Basics on Display Filters.....	158	Available Display Filters.....	160	How to Filter for Specific Signal Chain Elements.....	162	How to Filter for Signal Chain Elements from Associated Working Views.....	165	How to Filter Columns by Text and Regular Expressions.....	167
Basics on Display Filters.....	158										
Available Display Filters.....	160										
How to Filter for Specific Signal Chain Elements.....	162										
How to Filter for Signal Chain Elements from Associated Working Views.....	165										
How to Filter Columns by Text and Regular Expressions.....	167										

## Basics on Display Filters

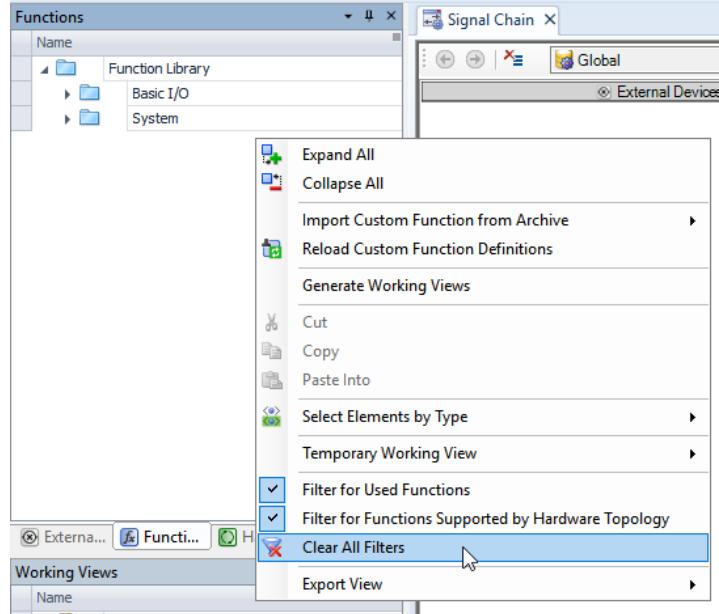
<b>Purpose of display filters</b>	ConfigurationDesk provides filters to display only elements that match the active filter criteria. This makes working with a complex ConfigurationDesk application that contains many elements easier.
<b>Available filters</b>	For a list of available filters, refer to <a href="#">Available Display Filters</a> on page 160.
<b>Common characteristics of display filters</b>	All display filters have the following common characteristics: <ul style="list-style-type: none"><li>▪ A filter setting affects each pane separately.</li><li>▪ You can use one filter or multiple filters in combination. Only the elements matching all filter criteria are displayed.</li><li>▪ In a hierarchy, the higher-level elements of elements matching a filter are also displayed.</li></ul>

- In a hierarchy, ConfigurationDesk lets you display all the subelements of elements matching the filter criteria, even if the subelements themselves do not match the filter criteria. To do so, select the **Include Subelements** checkbox in the Filter Settings dialog.



## Clearing all filters

To make sure that no filter is used and all elements are displayed in the currently active pane, you can use the **Clear All Filters** command from the pane's context menu:



## Related topics

### Basics

[Available Display Filters](#)..... 160

### HowTos

<a href="#">How to Filter Columns by Text and Regular Expressions</a> .....	167
<a href="#">How to Filter for Signal Chain Elements from Associated Working Views</a> .....	165
<a href="#">How to Filter for Specific Signal Chain Elements</a> .....	162

## Available Display Filters

### Filtering signal chain elements

**Filtering for used or unused signal chain elements** The Used Elements and Unused Elements filters let you filter for elements that are used or not used in the signal chain of the active ConfigurationDesk application.

For instructions, refer to [How to Filter for Specific Signal Chain Elements](#) on page 162.

**Tip**

In tables focusing on elements of the external wiring information, such as the Device Connectors table, 'used' refers to elements that are part of the calculated external cable harness.

**Filtering for used function blocks** When you activate the Used Functions filter, only the function block types with instantiated function blocks will be displayed in the function library hierarchy.

For instructions, refer to [How to Filter for Specific Signal Chain Elements](#) on page 162.

**Filtering for unresolved elements** The Unresolved Elements filter lets you filter for elements that are unresolved in the active ConfigurationDesk application. Depending on the type of element, the term 'unresolved' has different meanings:

- For device or hardware topology elements, 'unresolved' refers to elements that are used in the signal chain of the active ConfigurationDesk application but are missing from the respective topology.
- For the model topology, 'unresolved' refers to model port blocks that are used in the signal chain of the active ConfigurationDesk application but are not contained in the behavior model that is connected to the application.
- For elements of the external wiring information, 'unresolved' refers to elements that are referenced by the calculated external cable harness but are missing from the device topology or simulator hardware topology.

For instructions, refer to [How to Filter for Specific Signal Chain Elements](#) on page 162.

**Filtering for function block types supported by the hardware topology** The Supported Functions filter causes only function block types that support the current hardware topology to be displayed in the function library hierarchy. This filter is active by default.

For instructions, refer to [How to Filter for Specific Signal Chain Elements](#) on page 162.

**Filtering for elements from an associated working view** You can use the Associated Working View list to display only elements from a specific working view.

For instructions, refer to [How to Filter for Signal Chain Elements from Associated Working Views](#) on page 165.

**Filtering for imports and outports** You can filter the model topology or a bus configuration for imports and outports using the Filter for Imports and Filter for Outports filters. The filters are available in the corresponding context menus.

**Filtering columns by text and regular expression**

You can set text filter criteria for the elements displayed in columns of the selected pane.

For instructions, refer to [How to Filter Columns by Text and Regular Expressions](#) on page 167.

**Filtering working views**

In working views, you can filter for ports of a specific type.

For instructions, refer to [Filtering Ports](#) on page 197.

**Related topics****Basics**

Basics on Display Filters.....	158
Filtering Ports.....	197

**HowTos**

How to Filter Columns by Text and Regular Expressions.....	167
How to Filter for Signal Chain Elements from Associated Working Views.....	165
How to Filter for Specific Signal Chain Elements.....	162

## How to Filter for Specific Signal Chain Elements

**Objective**

To filter the contents of a pane for specific signal chain elements.

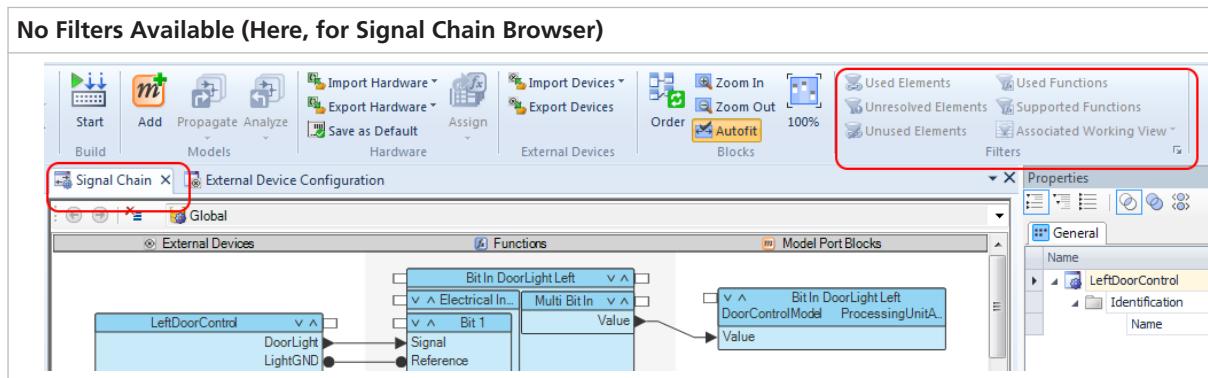
**Method****To filter for specific signal chain elements**

- 1 Click the pane to which you want to apply the filter.

A pane that contains filterable signal chain elements must have the focus. Otherwise, no filters are available. Only filters that work for the current pane are available.

**Filters Available (Here, for External Device Configuration Table)**

The screenshot shows the ConfigurationDesk application window. At the top, there's a toolbar with various icons like Start, Add, Propagate, Analyze, Import Hardware, Export Hardware, Save as Default, and others. Below the toolbar is a menu bar with File, Edit, View, Tools, Help. The main area is a table titled 'External Device Configuration'. The table has columns: Name, Description, Reference Port(s), Pin(s), Port Type, Physical A..., Electrical ..., Load Desc... . There are several rows of data, including LeftDoorControl, DoorLight, LightGND, MirrorHeating, MirrorMotor, MirrorGND, and Lock. To the right of the table is a 'Properties' panel. At the very top right of the application window, there's a 'Filters' toolbar with four buttons: 'Used Elements' (highlighted with a red box), 'Unresolved Elements', 'Unused Elements', and 'Associated Working View'.



- 2 In the Filters ribbon group on the Home ribbon, select the filters that you want to apply to the current pane. For details on the available filters, refer to [Available Display Filters](#) on page 160.

### Result

The selected filters are applied and only the signal chain elements matching the filter criteria are displayed in the pane.

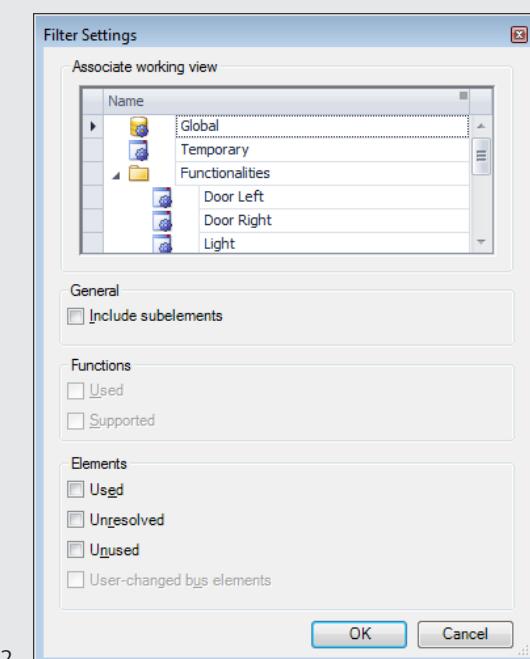
Click a filter again to deactivate it.

### Tip

- Active filters are highlighted on the ribbon:



- You can also open the Filter Settings dialog via the dialog launcher in the Filters ribbon group to activate or deactivate filters for a pane.



### Related topics

#### Basics

Available Display Filters.....	160
Basics on Display Filters.....	158

## How to Filter for Signal Chain Elements from Associated Working Views

### Objective

To display only signal chain elements from a specific working view in a pane.

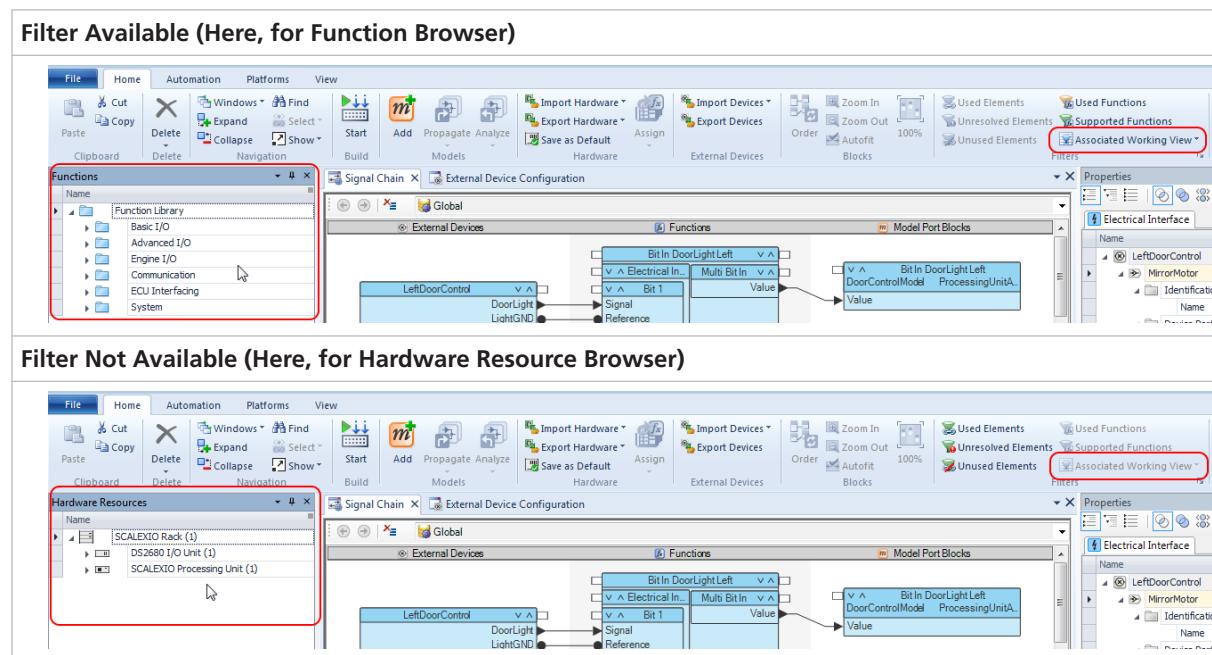
### Precondition

Your ConfigurationDesk application must contain a working view with the signal chain elements you want to filter for.

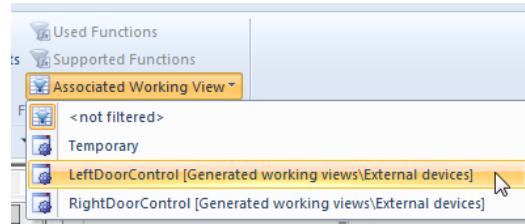
### Method

#### To filter for signal chain elements from an associated working view

- 1 Click the pane that you want to apply the filter to.  
A pane that contains filterable signal chain elements must have the focus. Otherwise, the filter is not available.



- 2 From the list of working views in the Filters ribbon group on the Home ribbon, select the working view with the signal chain elements for which you want to filter the current pane.



Only the signal chain elements from the selected working view are displayed in the pane.

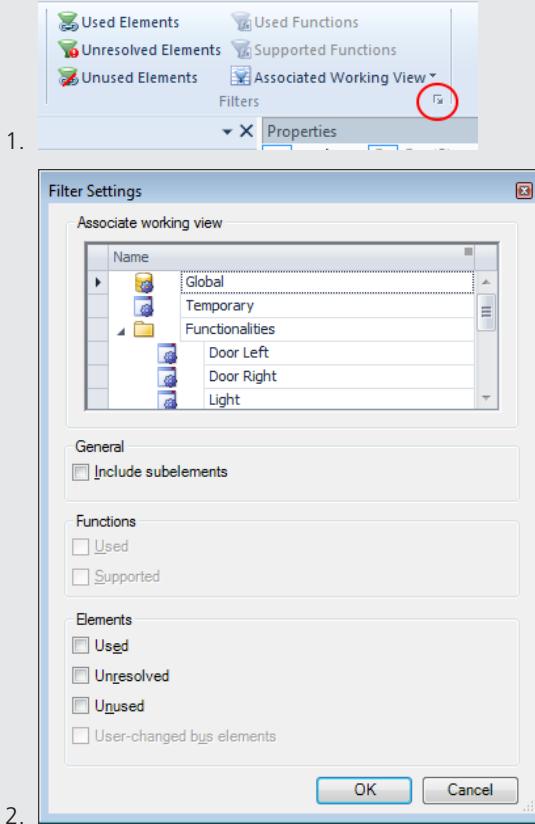
## Result

You filtered the current pane for signal chain elements from a specific working view.

To remove the filter, select <not filtered> from the list of working views.

### Tip

- If you select an empty associated working view, no signal chain elements will be displayed in the pane.
- You can also open the Filter Settings dialog via the dialog launcher in the Filters ribbon group to select an associated working view.



## Related topics

### Basics

Available Display Filters.....	160
Basics on Display Filters.....	158
Using Working Views.....	170

## How to Filter Columns by Text and Regular Expressions

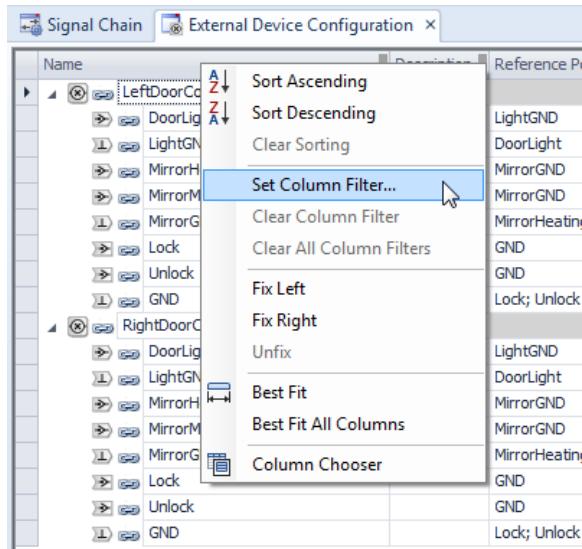
### Objective

To display only the elements in a column that match a filter string.

### Method

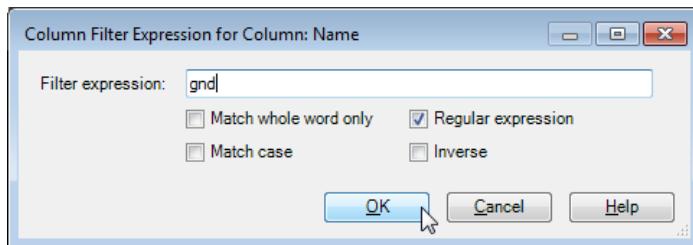
#### To filter columns by text and regular expressions

- 1 Select Set Column Filter from the context menu of a column header.



The Column Filter Expression dialog opens.

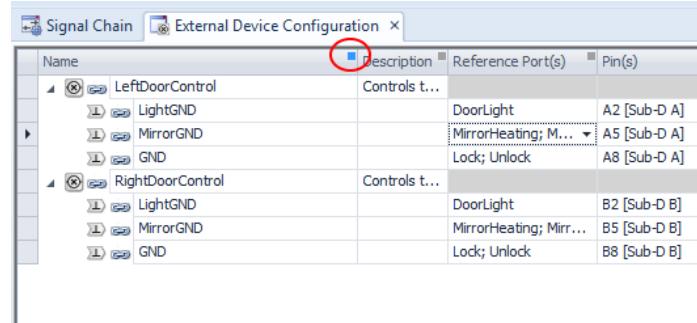
- 2 In the Filter expression edit field of the Column Filter Expression dialog, enter a regular expression (for example, ho.se matches horse and house).



- You can limit the filter to matches where no letter or numeral precedes or follows the filter string by activating the Match whole word only checkbox.
- You can make the filter case-sensitive by activating the Match case checkbox.
- To filter for strings containing regular expression meta-characters, you can disable the support of regular expressions by clearing the Regular expression checkbox.
- You can filter for elements *not* matching the text by activating the Inverse checkbox.

**3** Click OK.

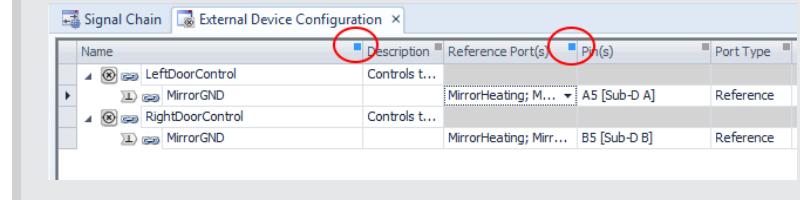
ConfigurationDesk displays all the elements that match your filter settings (and their higher-level elements). The filtered column is marked with a blue square.



Name	Description	Reference Port(s)	Pin(s)
LeftDoorControl	Controls t...	DoorLight	A2 [Sub-D A]
LightGND		MirrorHeating; M...	A5 [Sub-D A]
MirrorGND			
GND		Lock; Unlock	A8 [Sub-D A]
RightDoorControl	Controls t...	DoorLight	B2 [Sub-D B]
LightGND		MirrorHeating; Mir...	B5 [Sub-D B]
MirrorGND			
GND		Lock; Unlock	B8 [Sub-D B]

**Tip**

In a table, you can set filters for multiple columns.



Name	Description	Reference Port(s)	Pin(s)	Port Type
LeftDoorControl	Controls t...	MirrorHeating; M...	A5 [Sub-D A]	Reference
LightGND				
RightDoorControl	Controls t...	MirrorHeating; Mir...	B5 [Sub-D B]	Reference
MirrorGND				

---

## Result

You displayed only elements in a column that match a filter string.

You can reset specific or all column filters via the column header context menu commands Clear Column Filter or Clear All Column Filters.

---

## Related topics

### Basics

Available Display Filters.....	160
Basics on Display Filters.....	158
Using Tables to Access and Configure Elements.....	151

# Handling the Signal Chain in Working Views

---

<b>Objective</b>	Signal chain elements used in a ConfigurationDesk application are handled in working views.
------------------	---

---

Where to go from here	Information in this section
	<a href="#">Using Working Views.....</a> 170
	<a href="#">Customizing the Display of Working Views.....</a> 186

# Using Working Views

## Objective

You can use predefined working views or create user-defined working views and add signal chain elements to them.

## Where to go from here

## Information in this section

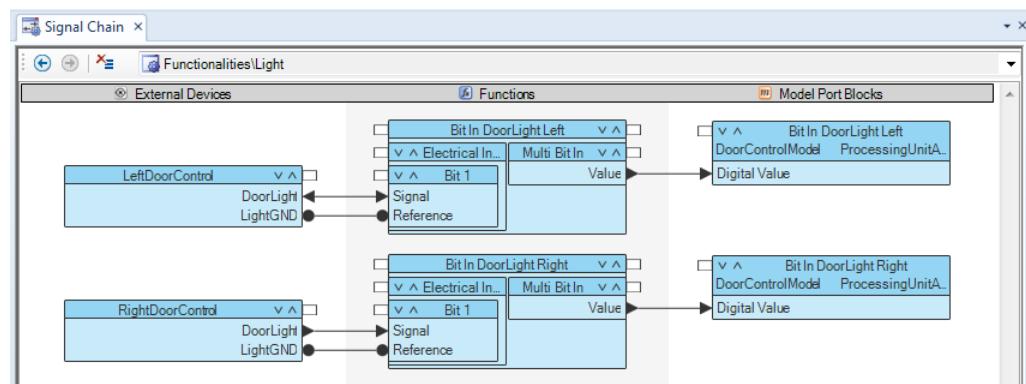
Basics on Working Views.....	170
How to Create Working Views and Working View Groups.....	174
How to Open Working Views in the Signal Chain Browser.....	175
How to Open Working Views in the Model Communication Browser.....	178
How to Generate Working Views from Signal Chain Elements.....	180
How to Export and Import Elements from a Working View.....	182

## Basics on Working Views

### Purpose of working views

A working view is a selection of elements used in the signal chain of the active ConfigurationDesk application. If you open working views, the main elements of the signal chain are represented by blocks (device blocks, function blocks, model port blocks). The blocks have ports that let you create mapping lines between them.

The following illustration shows the Light working view from the CfgStartingWithECUTutorial demo project in the Signal Chain Browser:



## Predefined working views

The predefined empty working views **Global** and **Temporary** are automatically added to a new ConfigurationDesk application. They cannot be renamed or removed.

**Global working view** The **Global** working view is the master working view that always shows the complete signal chain.

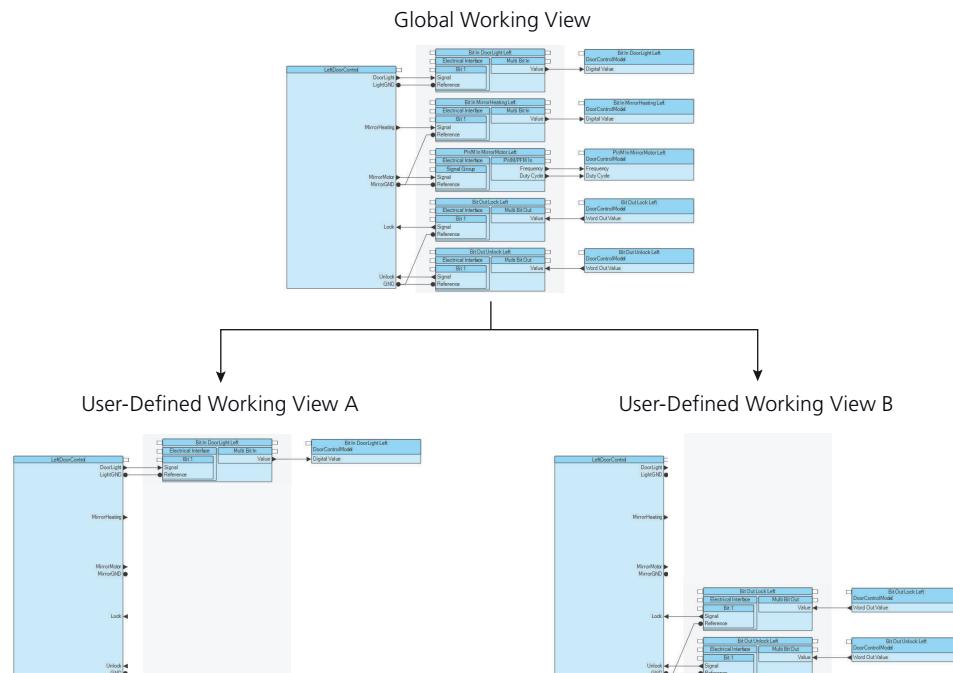
Every change you apply to the signal chain in any working view also affects the **Global** working view.

Changes that you apply to the signal chain in panes or view sets not showing working views might also affect the **Global** working view. For example, adding I/O functionality via the **Model-Function Mapping Browser** in the **Model-Function** view set adds function blocks and model port blocks to the **Global** working view.

**Temporary working view** You can use the **Temporary** working view for drafting a signal chain segment, like a notepad. You can add topology or library elements and signal chain elements connected to them to the **Temporary** working view even if it is not open. The context menu of several panes provides the required **Temporary Working – Add Connected Elements** command.

## User-defined working views

You can create additional user-defined working views, each of which shows a specific segment of the **Global** working view, i.e., a specific segment of the signal chain. This simplifies work on complex signal chains. The principle of user-defined working views is illustrated below.



Every change you apply to the signal chain in a user-defined working view also affects the **Global** working view. However, if you remove an element from a user-defined working view or the **Temporary** working view using the **Remove**

from this Working View or the Remove from Working View command, it still exists in the Global working view.

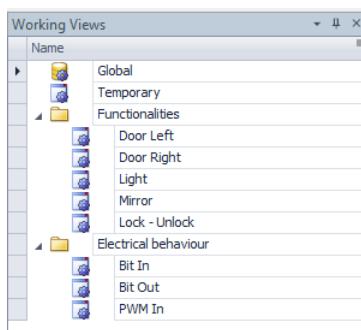
**Tip**

The display of large working views with many signal chain elements requires a large amount of system resources. Take the following steps if your ConfigurationDesk application contains a large number of signal chain elements and you are experiencing performance issues:

- Close the Global working view and avoid using it.
- Use only user-defined working views instead. Split up your signal chain so that user-defined working views do not contain too many elements.

## Managing working views

All available working views of the active ConfigurationDesk application are listed in the Working View Manager. You can use this 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. You can create working view groups to which you can move user-defined working views for a better overview.



For instructions, refer to [How to Create Working Views and Working View Groups](#) on page 174.

## Adding elements to working views

There are multiple ways to add signal chain elements to a working view.

**Adding topology elements or function blocks** You can add topology elements from the External Device Browser or Model Browser, or function blocks from the Function Browser to a working view. If the elements were not already used in the signal chain, they are also added to the signal chain and to the Global working view.

For more information on adding the different types of elements, refer to:

- [Adding Device Topology Elements to the Signal Chain](#) on page 287
- [Adding Function Blocks to the Signal Chain](#) on page 307
- [Adding Model Port Blocks to the Signal Chain](#) on page 432

**Adding elements from a different working view** ConfigurationDesk provides multiple methods to add elements to a working view from a different working view:

- Select elements in working view A and select **Copy** in the context menu. Then open working view B and select **Add to Working View** in the context menu.
- Select elements in working view A and drag the selection to working view B in the **Working View Manager**.
- If an element in your current working view has elements connected to it in the signal chain, you can use the **Add Connected Elements** context menu command to add them all to the current working view. This is a quick way of adding missing elements.

These actions do not change the signal chain and do not affect the **Global** working view. If elements already exist in a target working view, they are not added.

**Generating working views from signal chain elements** You can automatically generate working views and working view groups according to the structure of used signal chain elements in topology browsers or the function library. For instructions, refer to [How to Generate Working Views from Signal Chain Elements](#) on page 180.

#### Working view browsers

You can open working views in different browsers for different purposes.

**Signal Chain Browser** The Signal Chain Browser shows all signal chain elements of working views and the mapping lines between them in three columns from device blocks to function blocks to model port blocks. Refer to [How to Open Working Views in the Signal Chain Browser](#) on page 175.

ConfigurationDesk offers the Signal Chain view set for handling working views in the Signal Chain Browser.

**Model Communication Browser** The Model Communication Browser shows only the model port blocks of working views and the mapping lines between them in two columns from Data Outport blocks to Data Inport blocks. Refer to [How to Open Working Views in the Model Communication Browser](#) on page 178.

ConfigurationDesk offers the Multiple Models view set for handling working views in the Model Communication Browser.

#### Customizing the display of working views

You can customize the display of working views, for example, by zooming or setting highlighting options. Refer to [Customizing the Display of Working Views](#) on page 186.

#### Related topics

##### HowTos

<a href="#">How to Create Working Views and Working View Groups.....</a>	174
<a href="#">How to Export and Import Elements from a Working View.....</a>	182
<a href="#">How to Generate Working Views from Signal Chain Elements.....</a>	180

How to Open Working Views in the Model Communication Browser.....	178
How to Open Working Views in the Signal Chain Browser.....	175

## How to Create Working Views and Working View Groups

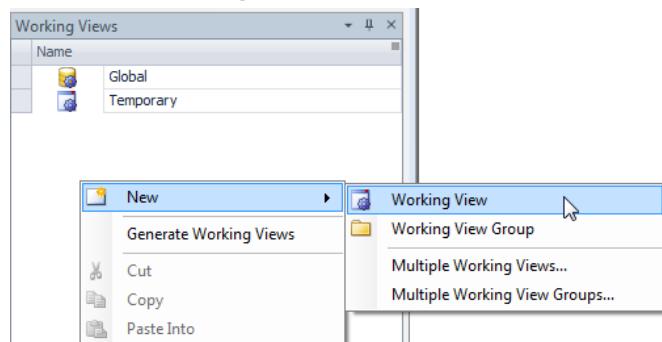
### Objective

To create working views and working view groups to which you can add signal chain elements.

### Method

#### To create working views and working view groups

- 1 Right-click an empty area of the Working View Manager to open the context menu.
- 2 Select New – Working View.



A new working view is created in the Working View Manager. The working view is empty and has a default name. You can now enter a new name for the working view.

- 3 Right-click an empty area of the Working View Manager again and select New – Working View Group from the context menu.
- A new working view group is created in the Working View Manager. You can rename it.
- 4 Right-click the working view group you created and select New – Multiple Working Views.
- A Create Multiple Working Views dialog opens.
- 5 Provide a Name pattern and a Number of instances and click OK to create the working views.

- 6** Repeat the methods described in steps 1 to 5 to create your own working view structure.

**Tip**

You can also:

- Drag working views to working view groups.
- Create working view groups in working view groups.
- Move several working views at the same time after selecting them while pressing the **Ctrl** key.
- Create multiple working view groups at once.
- Use copy, cut & paste functionality to change the working view structure.

**Result**

You created working views and working view groups. Now you can add signal chain elements to the working views.

**Related topics****Basics**

Basics on Working Views.....	170
------------------------------	-----

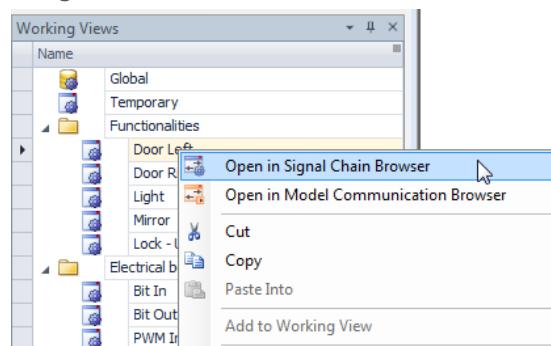
## How to Open Working Views in the Signal Chain Browser

**Objective**

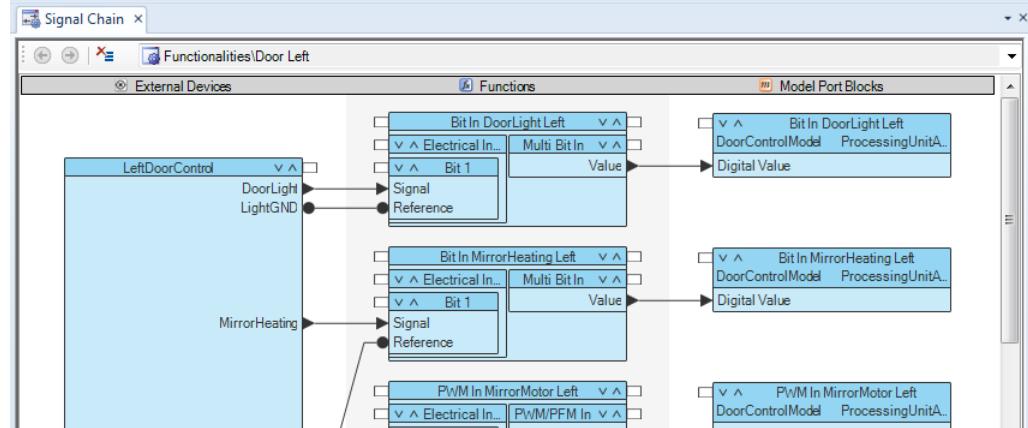
To open working views, such as the Global working view or user-defined working views.

**Method****To open working views in the Signal Chain Browser**

- 1** In the Working View Manager, right-click a working view and select Open in Signal Chain Browser.

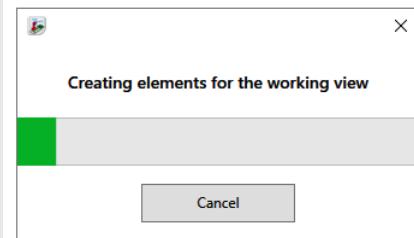


The working view opens in the Signal Chain Browser in the working area.



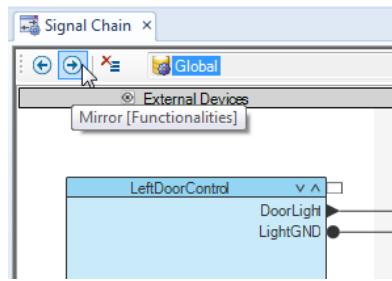
#### Tip

- You can also double-click a working view in the Working View Manager to open it in the Signal Chain Browser.
- By default, the Signal Chain Browser is available in the Signal Chain view set. If you open a working view from a different view set, ConfigurationDesk automatically switches to the Signal Chain view set, unless you opened the Signal Chain Browser in the current view set.
- If you open a large working view with many signal chain elements, a progress bar is displayed while the working view is prepared.

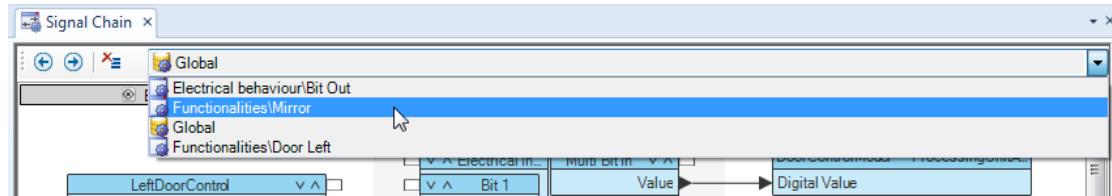


Click Cancel to stop the opening process. The working view is not opened.

- 2 Repeat step 1 to open more working views in the Signal Chain Browser.
- 3 In the Signal Chain Browser, use the left and right arrows to browse the open working views. A tooltip shows the next working view that clicking an arrow will browse to.

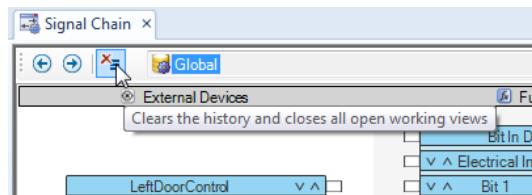


You can also select one of the open working views from a list.



Opening many working views that contain a large number of signal chain elements might decrease performance.

Click to close all working views open in the Signal Chain Browser.



## Result

You opened selected working views in the Signal Chain Browser.

### Tip

- You can also open working views from signal chain elements that are part of the working views. For instructions, refer to [How to Show Elements in a Different Pane](#) on page 125.
- You can hide the External Devices column if you do not require it. Refer to [Customizing the Display of the Working View Columns](#) on page 198.

## Related topics

### Basics

[Basics on Working Views.....](#) 170

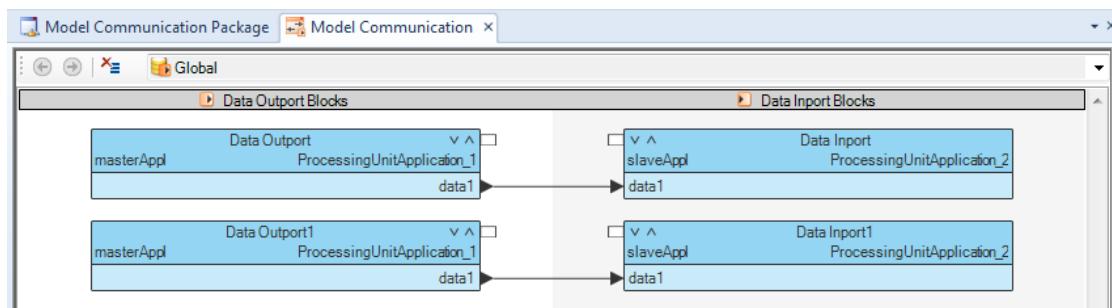
## How to Open Working Views in the Model Communication Browser

### Objective

To show the Data Outport and Data Import blocks from selected working views and the mapping lines between them.

### Purpose of the Model Communication Browser

The Model Communication Browser shows the mappings between models by sorting the Data Outport and Data Import blocks in two different columns. This is useful in multimodel ConfigurationDesk applications, where model communication is used. You can draw, change, or remove mapping lines like in the Signal Chain Browser.



### Tip

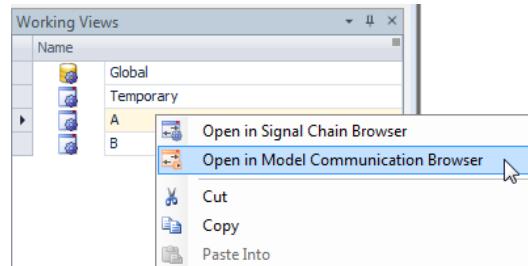
If the protocol of a model communication package is set to Blocking, this is visualized as follows:



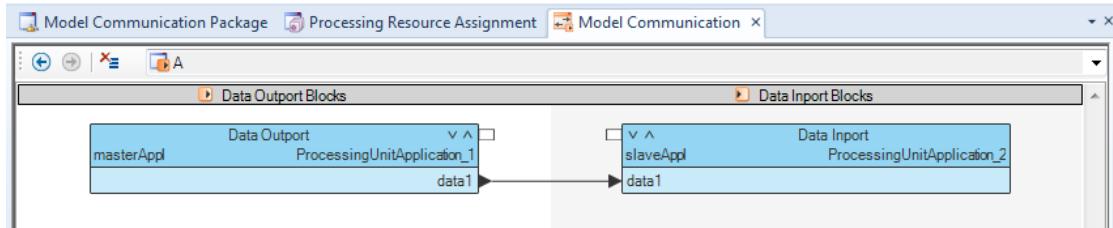
### Method

#### To open working views in the Model Communication Browser

- In the Working View Manager, right-click a working view and select Open in Model Communication Browser.

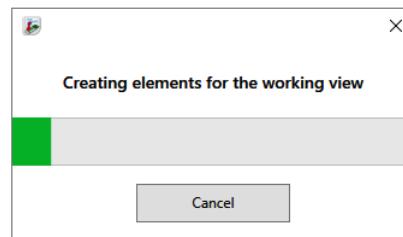


The working view opens in the Model Communication Browser in the working area.



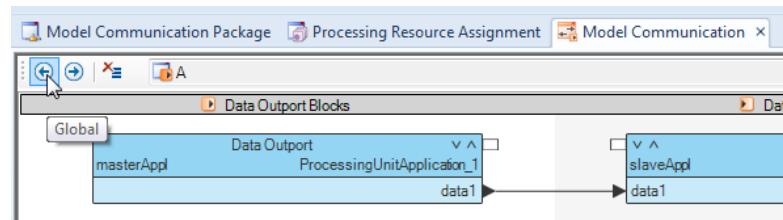
### Tip

- By default, the Model Communication Browser is available in the Multiple Models view set. If you open a working view in the Model Communication Browser from a different view set, ConfigurationDesk automatically switches to the Multiple Models view set, unless you have opened the Model Communication Browser in the current view set.
- If the Model Communication Browser is open in the active view set and the Signal Chain Browser is not, you can also double-click a working view in the Working View Manager to open it in the Model Communication Browser.
- If you open a large working view with many signal chain elements, a progress bar is displayed while the working view is prepared.

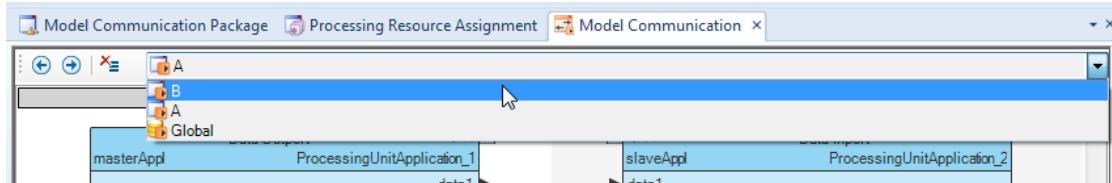


Click Cancel to stop the opening process. The working view is not opened.

- 2 Repeat step 1 to open more working views in the Model Communication Browser.
- 3 In the Model Communication Browser, use the left and right arrows to browse the open working views. A tooltip shows the next working view that clicking an arrow will browse to.

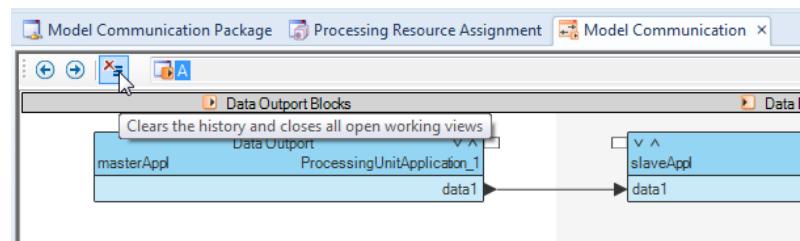


You can also select one of the open working views from a list.



Opening many working views that contain a large number of model port blocks might decrease performance.

Click to close all working views open in the Model Communication Browser.



---

## Result

You opened selected working views in the Model Communication Browser.

### Tip

You can also open working views from signal chain elements that are part of the working views. For instructions, refer to [How to Show Elements in a Different Pane](#) on page 125.

---

## Related topics

### Basics

Basics on Working Views.....	170
Setting Up Model Communication.....	448

## How to Generate Working Views from Signal Chain Elements

---

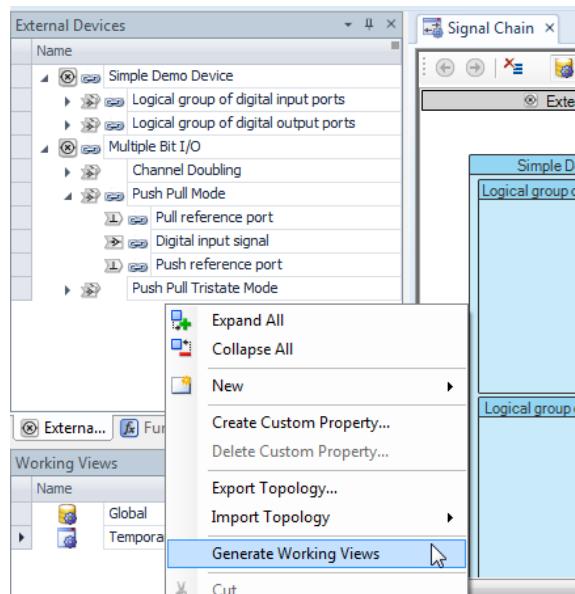
### Objective

To automatically create working views and working view groups according to the structure of used signal chain elements in topology browsers or the function library. The used signal chain elements are added to the according working views.

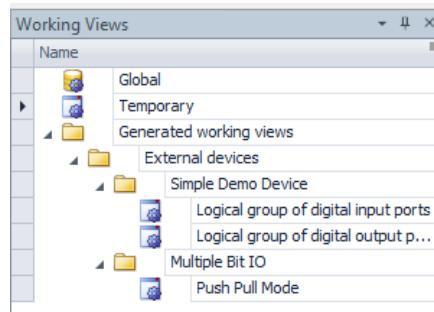
**Method**

**To generate working views and working view groups from signal chain elements**

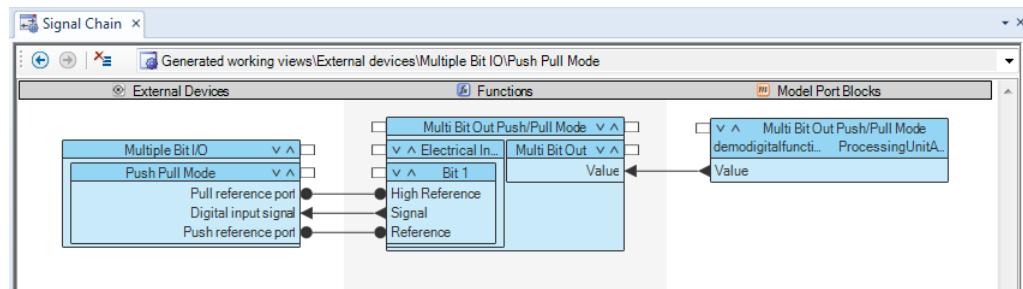
- 1 Right-click an empty area of a topology browser or the Function Browser to open the context menu.
- 2 Select Generate Working Views.



In the Working View Manager, working view groups and working views are created according to the structure of used signal chain elements in the browser:



The used signal chain elements are added to the according working views:



**Tip**

If you execute the command from the Working View Manager, new working views and working view groups are created according to the structure of used signal chain elements from all topology browsers and the Function Browser.

---

**Result**

You generated working views from signal chain elements.

---

**Related topics**

**Basics**

[Basics on Working Views.....](#) 170

## How to Export and Import Elements from a Working View

---

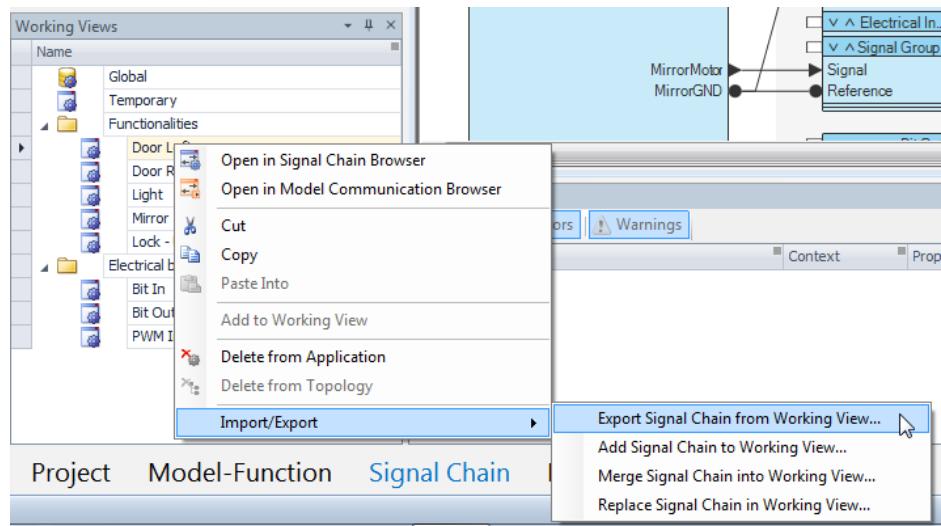
**Objective**

To export the signal chain elements from a working view and import them to a working view in a different ConfigurationDesk application.

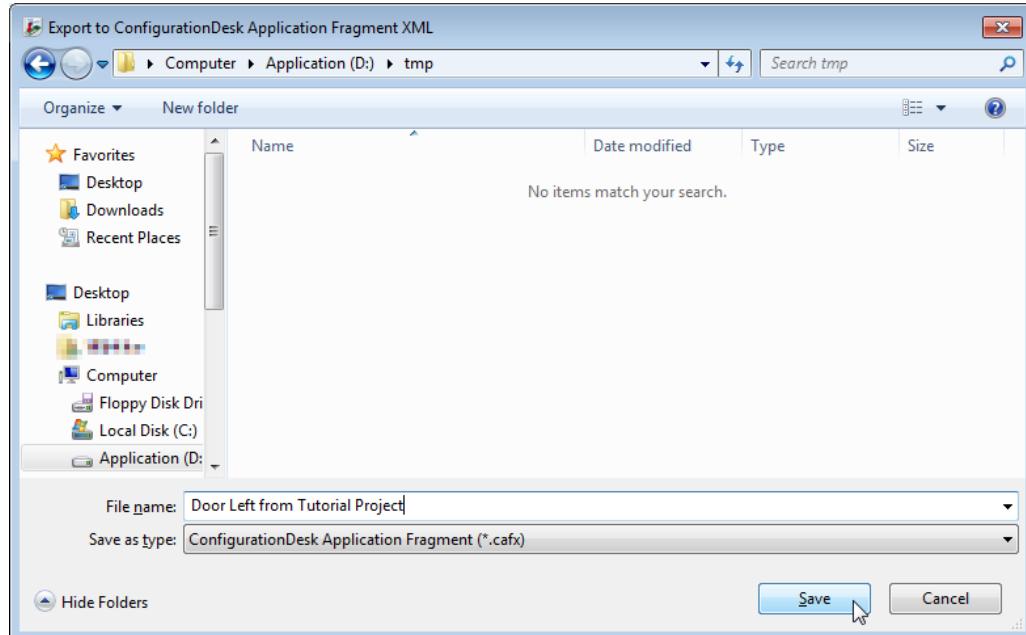
When you export the signal chain elements from a working view, they are saved to a ConfigurationDesk application fragment (CAFX) file.

**Method****To export and import elements from a working view**

- 1 In the Working View Manager, right-click a working view and select Import/Export – Export Signal Chain from Working View from the context menu.



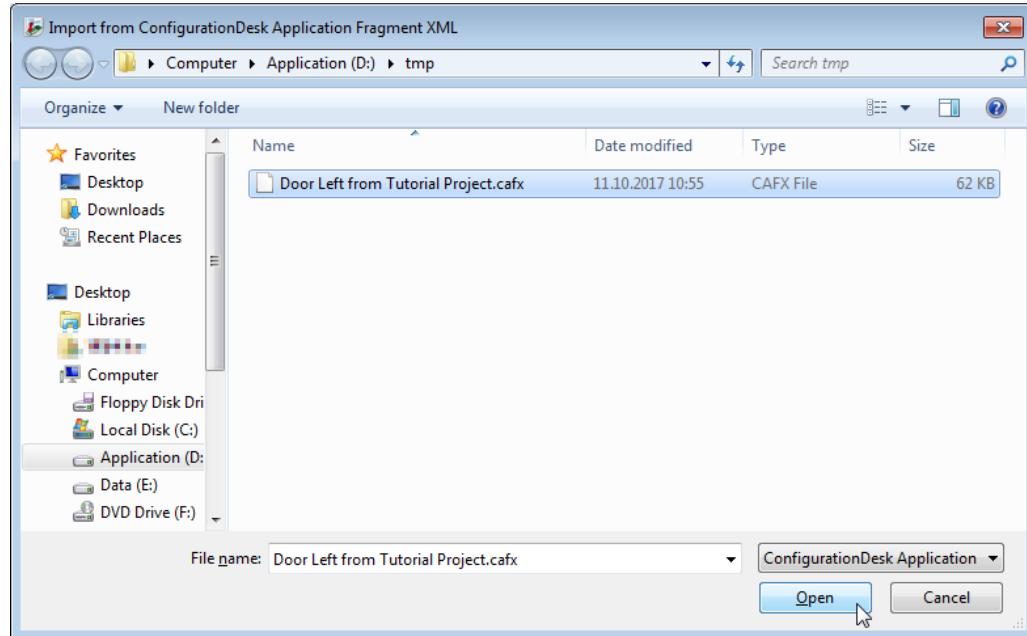
- 2 Select a file name and location for the CAFX file and save it.



- 3 Open the target ConfigurationDesk application.
- 4 In the Working View Manager, right-click a working view and select one of the following commands from the Import/Export submenu:
  - Add Signal Chain to Working View: The imported signal chain elements are added as new elements to the existing elements in the selected working view.

- Merge Signal Chain into Working View: The imported signal chain elements that do not exist in the selected working view yet are added to it. Existing elements with the same name as imported elements are replaced.
- Replace Signal Chain in Working View: Existing elements in the selected working view are removed and the imported elements are added to it.

5 Select the CAFX file that you previously exported.



The elements from the CAFX file are added to the working view according to the command you selected in step 4.

**Note**

- ConfigurationDesk tries to preserve the property settings of the exported elements during import if possible. This might lead to conflicts in the target application, for example, duplicate name conflicts or hardware resource assignment conflicts.
- If exported signal chain elements reference elements that are not part of the exported working view, ConfigurationDesk adds the referenced elements to the signal chain. For example, if you export a working view containing function blocks that reference an Engine Simulation Setup function block which is *not* part of the working view, the referenced function block is added to the signal chain and the reference is restored.  
Note the following specifics:
  - The property settings of the referenced element are its default settings, because only the reference is part of the export and the referenced element has to be created.
  - The referenced element is *not* added to the target working view. It is added to the signal chain and thus to the Global working view.
  - If you use the Add Signal Chain to Working View command, model port blocks are added as new, unresolved blocks with a new identity (ID). Refer to [Characteristics of Model Port Blocks](#) on page 426.

**Result**

You exported and imported elements from and to a working view.

**Related topics****Basics**

Basics on Working Views.....	170
------------------------------	-----

# Customizing the Display of Working Views

<b>Objective</b>	You can customize the display of signal chain elements in working views.
------------------	--

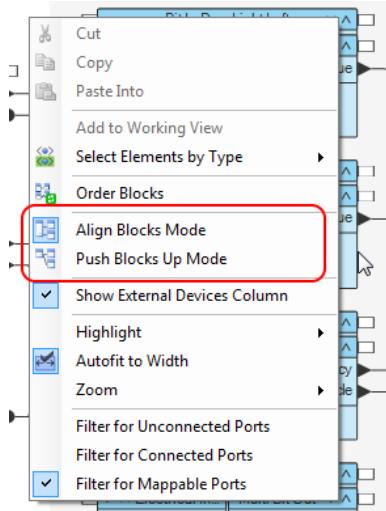
Where to go from here	Information in this section
	<a href="#">Vertical Alignment of Blocks.....</a> 186
	<a href="#">Ordering Blocks.....</a> 187
	<a href="#">Collapsing and Expanding Blocks.....</a> 189
	<a href="#">Changing the Background Color of Blocks.....</a> 193
	<a href="#">Increasing and Decreasing the Size of Displayed Signal Chain Elements.....</a> 194
	<a href="#">Highlighting Signal Chain Conflicts.....</a> 194
	<a href="#">Filtering Ports.....</a> 197
	<a href="#">Customizing the Display of the Working View Columns.....</a> 198

## Vertical Alignment of Blocks

<b>Purpose of alignment settings</b>	ConfigurationDesk offers different alignment settings to optimize the vertical alignment of blocks in a working view according to your requirements.
--------------------------------------	--

**Accessing alignment settings**

You can access the alignment settings via the context menu of a working view.



The alignment settings are set separately for each working view.

**Available alignment modes**

**Align blocks** In the Align Blocks Mode, mapped ports are aligned side by side to avoid diagonal mapping lines whenever possible.

**Tip**

This is the default alignment for working views that have been opened.

**Push blocks up** In the Push Blocks Up Mode, the blocks move towards the top, regardless of port positions and diagonal mapping lines.

**Related topics****Basics**

Basics on Working Views..... 170

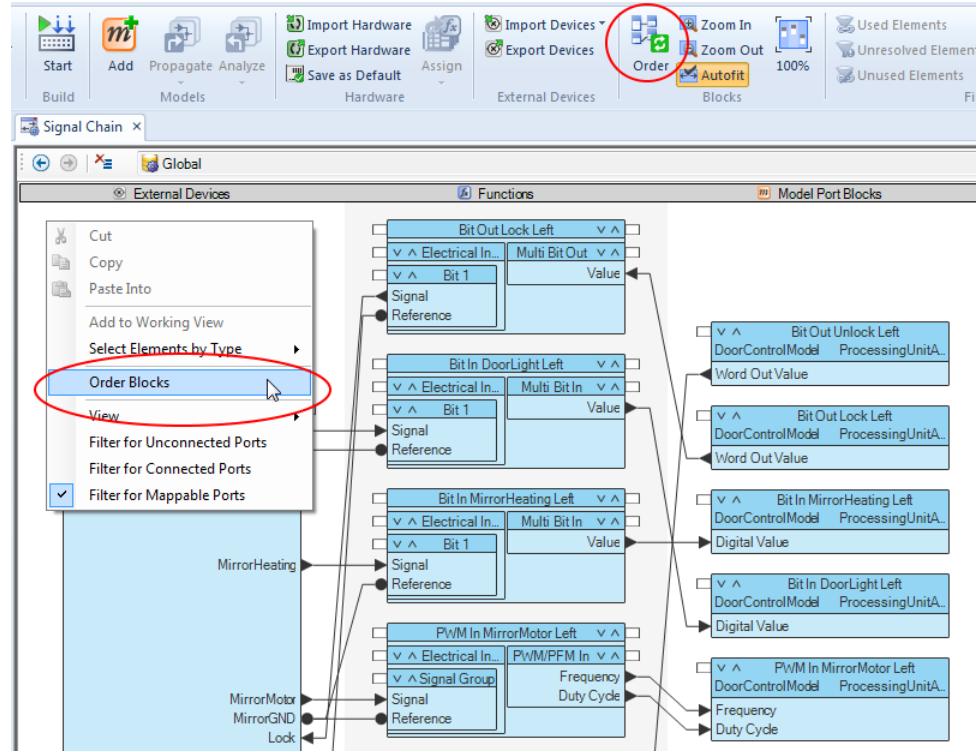
## Ordering Blocks

**Purpose of ordering blocks**

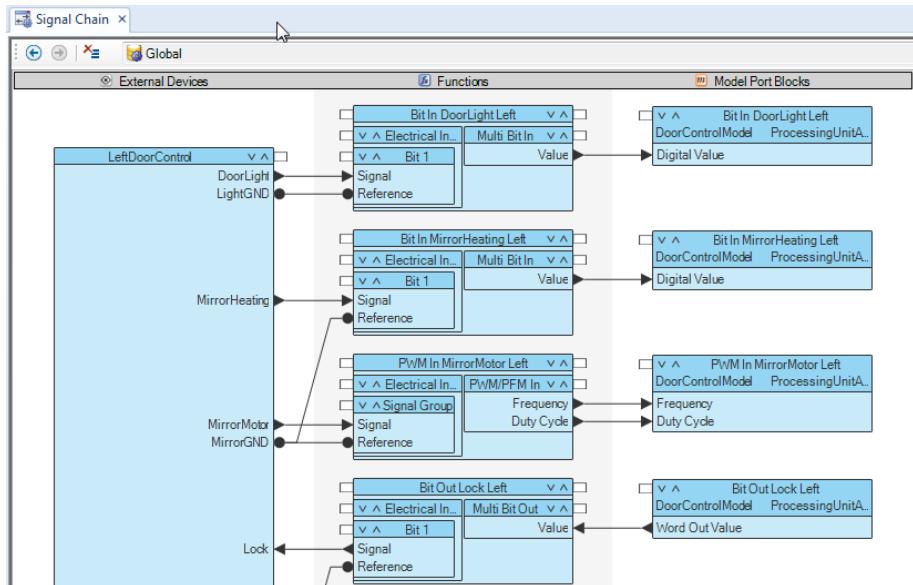
A signal chain consisting of many blocks might be confusing due to intersecting mapping lines. ConfigurationDesk lets you rearrange the blocks of a signal chain so that less mapping lines intersect.

### Instructions for ordering blocks

Use the Order Blocks command from the context menu of a working view or on the Home ribbon to rearrange the blocks.



In the Signal Chain Browser, the function blocks and the model port blocks are rearranged so that as few mapping lines as possible intersect. The sequence of the device blocks remains unchanged.



**Note**

- Ports are not rearranged.
- Reference ports are not taken into account.

**Related topics****Basics**

<a href="#">Basics on Working Views.....</a>	170
--	-----

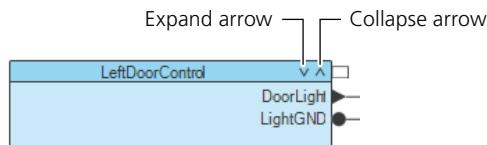
## Collapsing and Expanding Blocks

**Objective**

For a better handling of ports and mappings, you can collapse blocks in a working view.

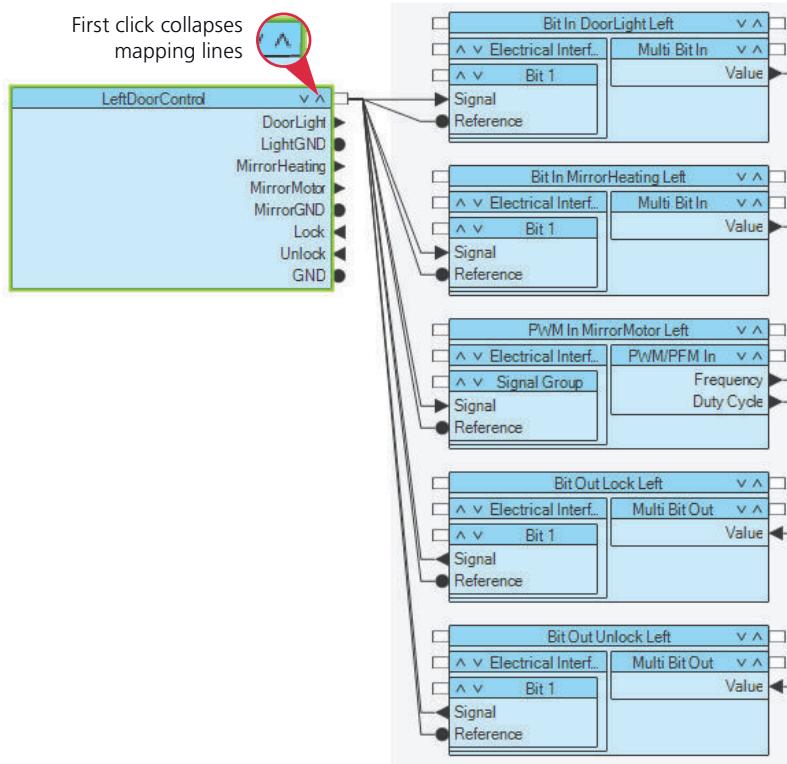
**Collapse and expand arrows**

Every block contains collapse and expand arrows that let you collapse or expand the block. The following illustration shows the collapse and expand arrows in a device block.

**Collapsing and expanding stages**

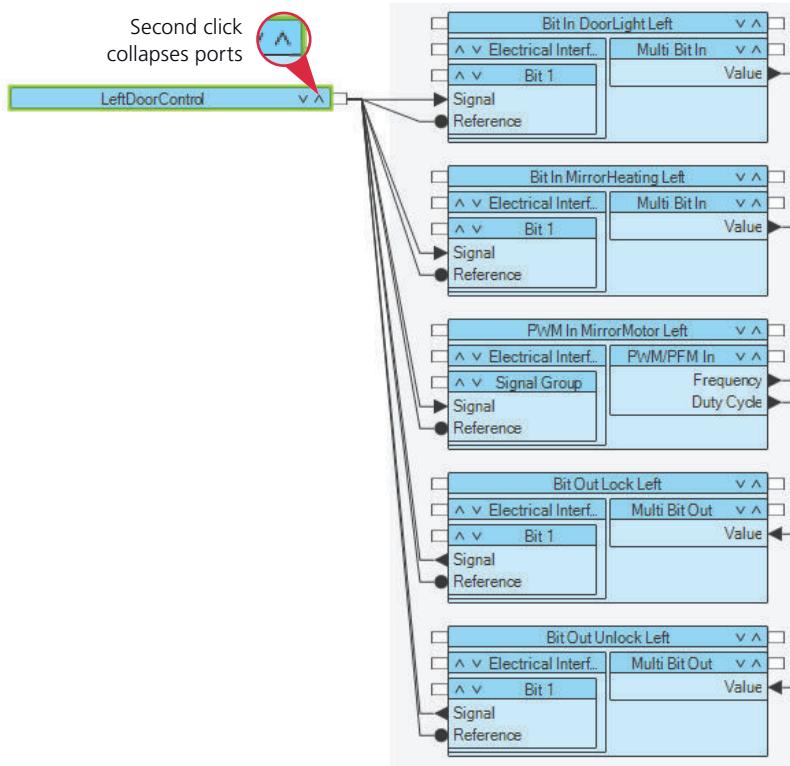
There are two stages for collapsing/expanding blocks:

1. Collapsing mapping lines only: If you click the collapse arrow once, all mapping lines connected to the block are drawn to the block's parent port. This helps you, for example, to get an overview of unmapped ports in a complex mapping structure.



Click the expand arrow to return the block and the mapping lines to their previous state.

2. Collapsing ports and mapping lines: If you click the collapse arrow a second time, the block's ports disappear and only the block header is displayed. This lets you, for example, minimize the display of blocks that you are currently not configuring.



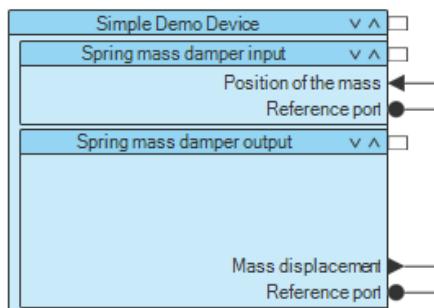
Click the expand arrow once to return to the first stage or twice to return the block and the mapping lines to their previous state.

#### Tip

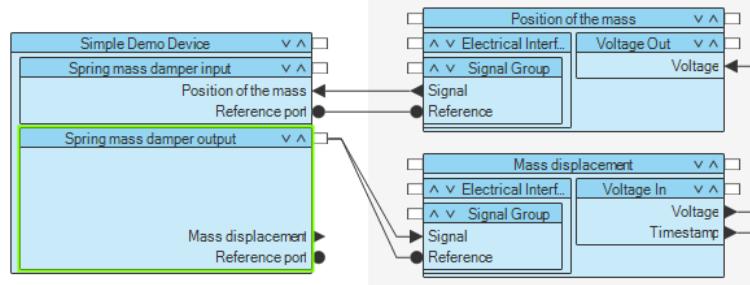
- The blocks are only collapsed in the current working view.
- When you save a ConfigurationDesk application, the expand/collapse states of hierarchies and blocks are preserved.

#### Collapsing and expanding subblocks

Some elements are represented by subblocks within a block, for example, port groups in device blocks or electrical interfaces in function blocks. Each subblock has collapse/expand arrows so you can collapse/expand it separately.



If you click the collapse arrow of a subblock once, all mapping lines below the selected subblock are drawn to the subblock's parent port. For example, if you click the collapse arrow of a device port group once, all mapping lines connected to the ports in the port group are drawn to the parent port of the port group.



#### Tip

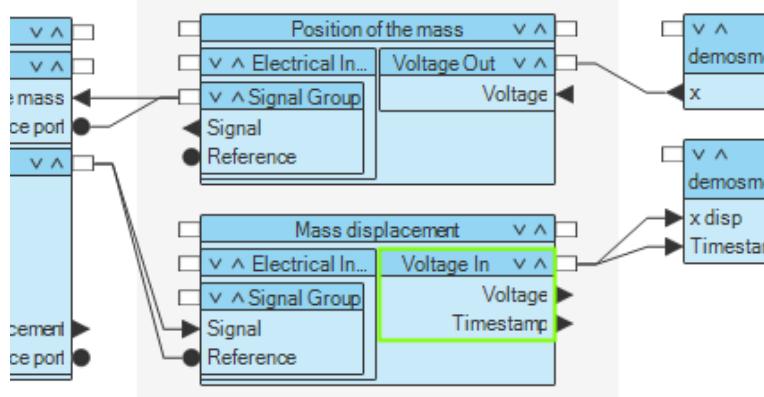
Collapsed subblocks remain collapsed after expanding a completely collapsed block.

#### Collapsing and expanding all blocks in a working view

You can collapse and expand all blocks in the current working view via the Collapse/Expand commands on the Home – Navigation ribbon. The collapsing and expanding stages described above are applied to all blocks and mapping lines of the current working view. The stages of individual blocks or subblocks are not preserved.

#### Points to note for collapsing/expanding function blocks

Function blocks have subblocks with ports and mappings on both sides. Hence you can separately collapse and expand mapping lines on both sides.



To collapse all ports and mapping lines of a function block, you must collapse it at the top level.



## Related topics

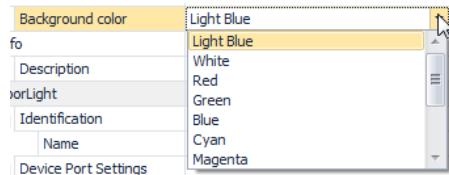
### Basics

[Basics on Working Views.....](#) 170

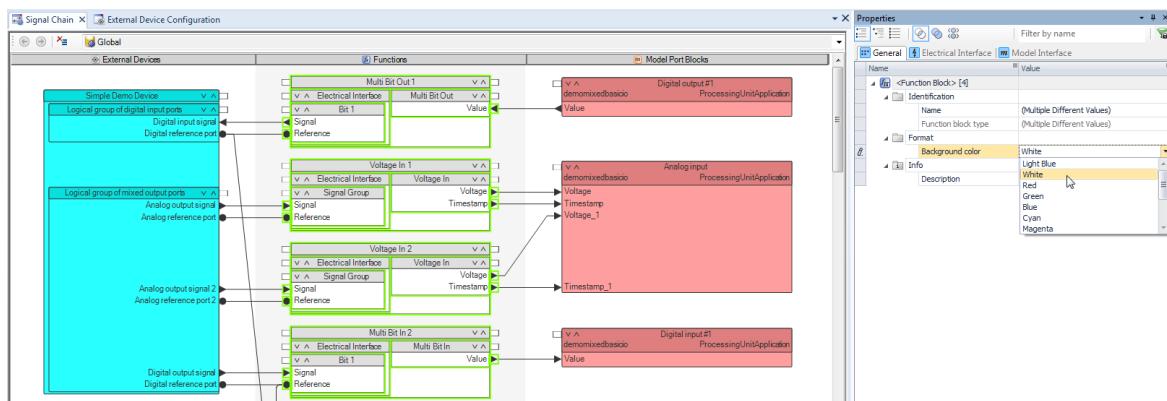
## Changing the Background Color of Blocks

### Configuring the background color property

All device blocks, function blocks, and model port blocks have the **Background color** property that lets you change the background color for selected blocks.



Using different background colors gives you a better overview of the signal chain.



**Related topics**

**Basics**

Basics on Working Views..... 170

## Increasing and Decreasing the Size of Displayed Signal Chain Elements

---

**Purpose of increasing or decreasing the size**

To take a closer look at a specific segment of the signal chain or to get a better overview of large parts of it, there are several settings to increase or decrease the size of displayed signal chain elements in a working view.

**Different zoom settings**

In the Blocks ribbon group on the Home ribbon and in the Zoom context menu of a working view, there are several settings that influence the display size of the signal chain elements. The settings must be set separately for each working view.

**Tip**

You can also use the shortcuts + or - on the numeric keypad or **Ctrl** + mouse wheel to zoom in or out or increase/decrease height.

---

**Related topics**

**Basics**

Basics on Working Views..... 170

## Highlighting Signal Chain Conflicts

---

**Basics on conflicts**

ConfigurationDesk allows for 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 to get proper build results.

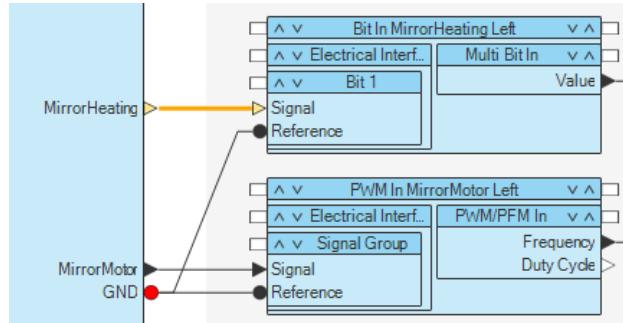
Conflicts can be grouped by severity, as described in the table below.

Severity	Description
Error	Severe conflicts that cause an operation to be aborted, for example, the build process.
Warning	Conflicts that may lead to faulty build results and are recommended to be resolved, but which do not abort operations.

### Highlighting conflicting elements in the signal chain

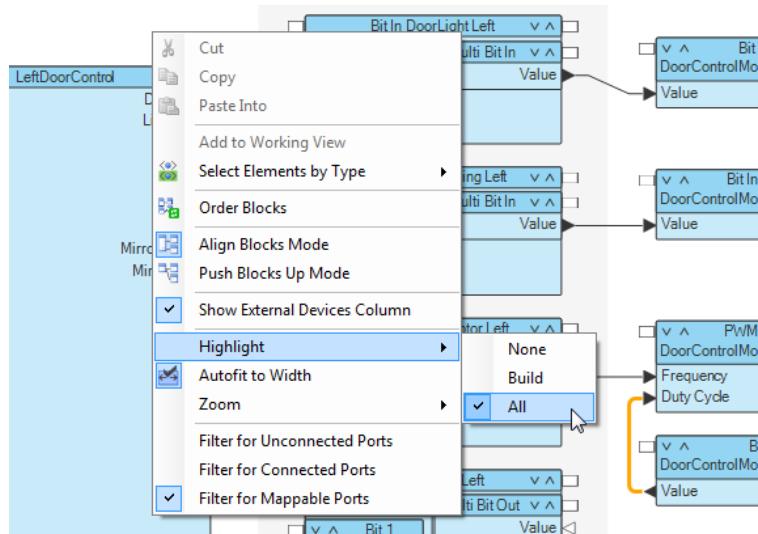
If signal chain elements are involved in a conflict, you can have ConfigurationDesk highlight the elements in a working view. This makes them easier to find. Warning conflicts are highlighted orange, error conflicts red.

In the following illustration, a device port is highlighted in red, indicating an error conflict, and a device port mapping is highlighted orange, indicating a warning conflict.



### Highlighting options

You can choose between different highlighting options via the **Highlight** submenu in the context menu of a working view. If you select the **Build** option, only conflicts that have an effect on the build process are highlighted.

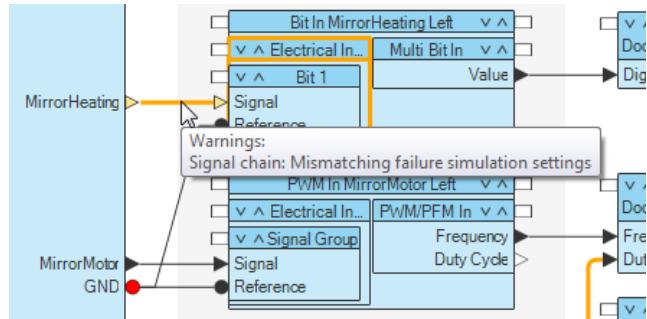


**Tip**

If a working view scrolls very slowly in a ConfigurationDesk application with a large number of signal chain elements, set the highlighting option to None.

**Tooltip messages**

If you point the mouse to a signal chain element that is highlighted in a working view, a tooltip containing the conflict type is displayed.



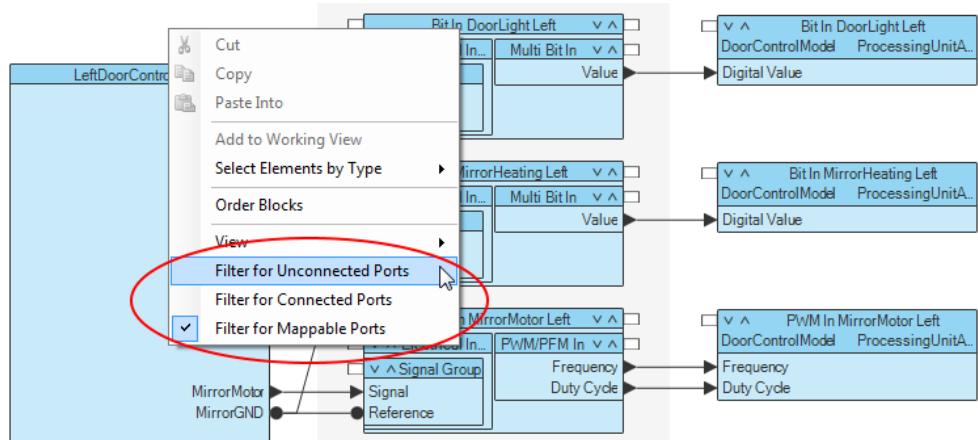
**Related topics**

**Basics**

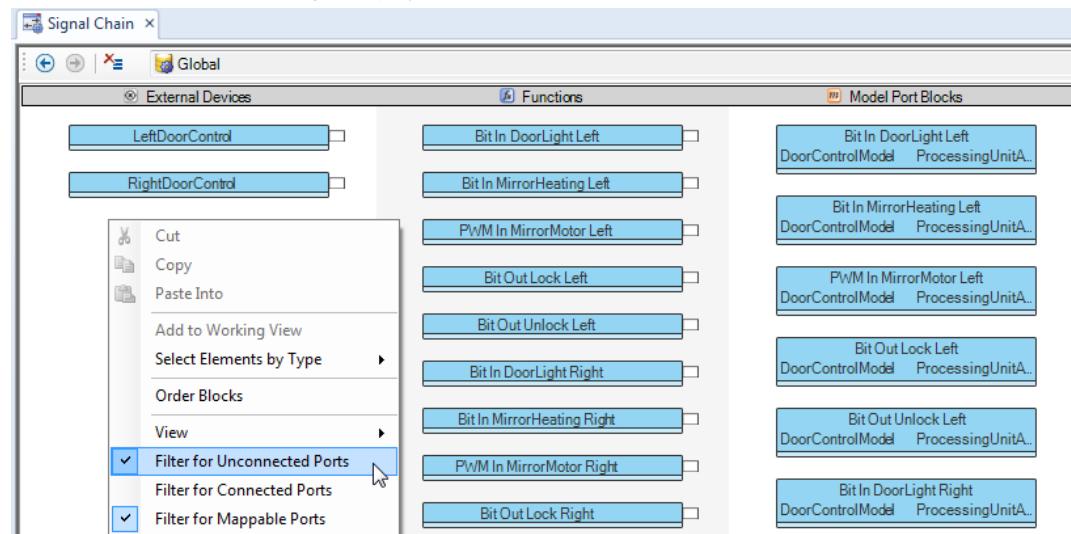
Basics on Device Blocks.....	257
Basics on Function Blocks.....	297
Characteristics of Function Ports.....	303
Characteristics of Model Port Blocks.....	426
Characteristics of Model Ports.....	429
Characteristics of Signal Ports.....	301
Resolving Conflicts.....	677

## Filtering Ports

**Filtering for ports of a specific type** You can filter for specific types of ports via the context menu of working views:



Only the ports matching the filter criteria are displayed in the working view. Structures that contain only ports that are filtered out, for example, an electrical interface of a function block only containing mapped/unmapped ports, are also no longer displayed.



An active filter is indicated by a checkmark to the left of the context menu command. Click a command again to deactivate a filter.

The filter settings apply to all working views in the Signal Chain Browser. They can be configured separately for working views in the Model Communication Browser. The filter settings for both browsers are saved for a project when you close it.

**Available port filters**

The following port filters are available in working views:

Filter	Purpose
Filter for Unconnected Ports	To display only unmapped ports.
Filter for Connected Ports	To display only mapped ports.
Filter for Mappable Ports	To display only ports that can be mapped to other ports. For example, function ports for which the Model access property is set to Disabled are hidden. This filter is active by default.

**Tip**

You can activate Filter for Connected Ports and Filter for Unconnected Ports at the same time. In this case, all blocks will be displayed without any structure elements.

**Related topics****Basics**

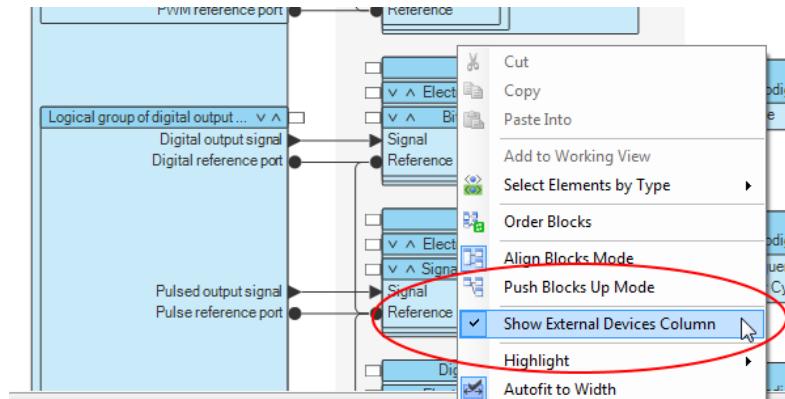
[Available Display Filters.....](#) 160

## Customizing the Display of the Working View Columns

**Defining the visibility of the External Devices column**

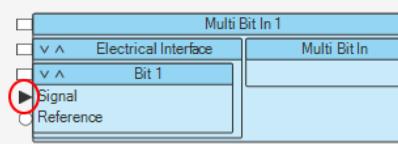
You can define the visibility of the External Devices column for a working view in the Signal Chain Browser.

For this purpose, use the Show External Devices Column command:

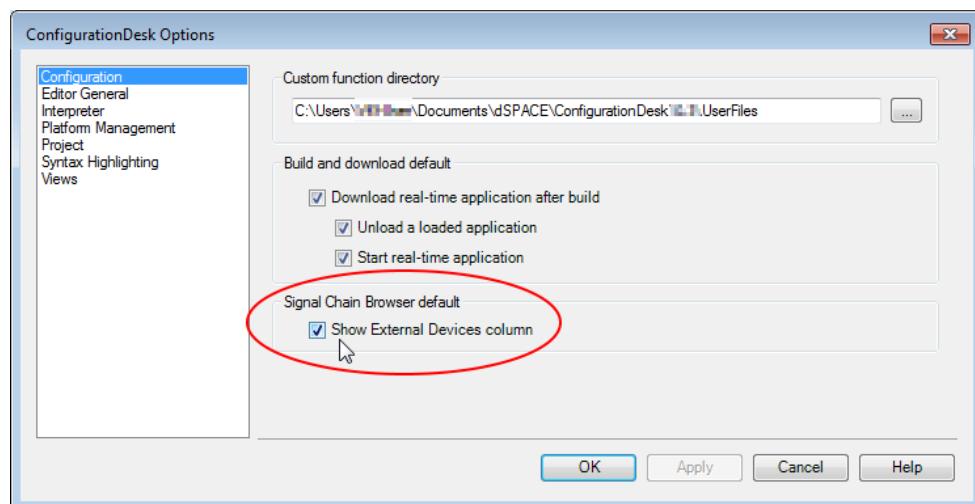


The command applies to each working view separately. The visibility of the External Devices column is indicated by the checkmark to the left.

If the External Devices column is hidden and the working view contains device port mappings, the mapped signal ports are black:



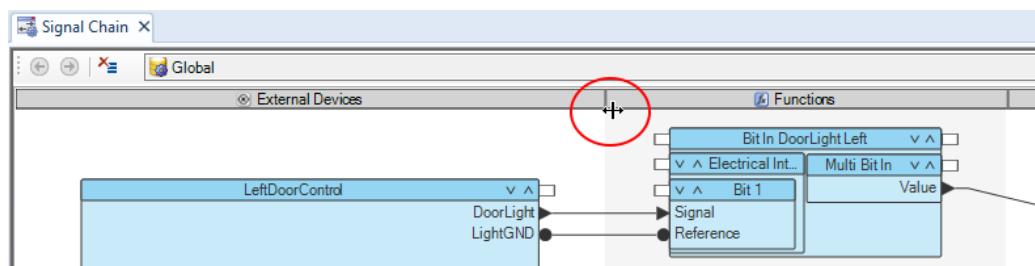
**Changing the default setting** You can specify the default setting for the Show External Devices column command on the Configuration Page of the ConfigurationDesk Options dialog:



If the checkbox is cleared, by default the External Devices column is hidden in working views that you open in the Signal Chain Browser.

#### Adjusting the column width

You can adjust the width of working view columns by clicking the area between two column headers and moving the column border to the left or right while holding the mouse button.



The column width settings are automatically saved for each working view.

**Tip**

To evenly distribute the columns according to the current width of the working view, disable (if necessary) and enable the Autofit to Width mode.

# Managing Real-Time Hardware

---

<b>Objective</b>	Managing real-time hardware includes connecting hardware to ConfigurationDesk and handling hardware topologies. Hardware topologies represent the real-time hardware in ConfigurationDesk and provide information on the hardware resources of a specific hardware system.
------------------	--

---

Where to go from here	Information in this section						
	<table><tr><td>Handling Registered Hardware.....</td><td>202</td></tr><tr><td>Platform Access by Several Users Simultaneously.....</td><td>229</td></tr><tr><td>Working with Hardware Topologies.....</td><td>236</td></tr></table>	Handling Registered Hardware.....	202	Platform Access by Several Users Simultaneously.....	229	Working with Hardware Topologies.....	236
Handling Registered Hardware.....	202						
Platform Access by Several Users Simultaneously.....	229						
Working with Hardware Topologies.....	236						

# Handling Registered Hardware

## Where to go from here

## Information in this section

Registering Hardware in ConfigurationDesk.....	202
Updating the Firmware of SCALEXIO or MicroAutoBox III Hardware.....	214
Configuring Hardware Properties.....	221
Configuring the I/O Ethernet Communication.....	225

## Registering Hardware in ConfigurationDesk

### Objective

It is necessary to register platforms in ConfigurationDesk, for example, if you want to download a real-time application.

## Where to go from here

## Information in this section

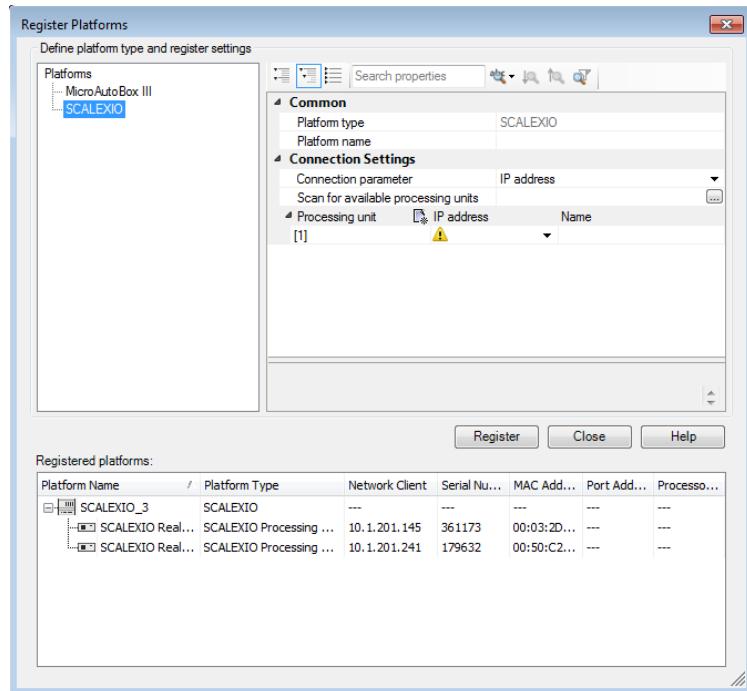
Basics on Registering Real-Time Hardware.....	202
Basics on Connecting a ConfigurationDesk Application to a Hardware System.....	207
How to Register dSPACE Real-Time Hardware.....	208
How to Register Hardware for Multi-PU Applications and Add it to a ConfigurationDesk Application.....	211

## Basics on Registering Real-Time Hardware

### Registering real-time hardware

In ConfigurationDesk, the registered hardware systems are treated as platforms which are displayed and which can be accessed via the Platform Manager.

To register platforms, you must enter their IP addresses, or alias names or board names, or MAC addresses in the Register Platforms dialog (see below).



#### Note

- You can register only one platform in the dialog at a time, and not several platforms by editing several IP addresses (or other values).
- Your host PC must be connected to the same network as the hardware system you want to register in ConfigurationDesk. Using the MAC address, alias name, or board name to find and register the hardware is supported only if the host PC and hardware are part of the same subnetwork. If your hardware system is installed in a different subnetwork connected to your host PC's network via a router or gateway, you must use the IP address for registering. Otherwise ConfigurationDesk is unable to find the hardware system.

After you register a new platform by clicking Register, ConfigurationDesk starts to search the local network for the entered settings (IP addresses, board names, alias names or MAC addresses). If they are found in the same network as your host PC, the corresponding platforms are automatically read out and displayed in the dialog and in the Platform Manager. You have successfully registered your hardware in ConfigurationDesk.

For details on IP addresses and alias names, refer to:

- [SCALEXIO: Setting up the Connection to the Host PC \(SCALEXIO Hardware Installation and Configuration](#)
- [MicroAutoBox III: Setting Up the Connection to the Host PC \(MicroAutoBox III Hardware Installation and Configuration](#)

**Tip**

Platforms can be displayed in the Platform Manager without a project or application being open.

In addition, you can manage registered platforms via the **Manage Recent Platform Configuration** dialog. For example, you can activate or deactivate platforms. This influences the number of platforms displayed in the Platform Manager: Only activated platforms are displayed. New registered platforms are activated by default.

**Using a custom platform name** When you register hardware, you can edit a unique custom name for your platform, which then is used after registration. It is displayed in ConfigurationDesk's Platform Manager as well as in the Platform Managers of other dSPACE software products which support the registered platform and which are installed on your host PC. Thus, you can use this name to identify the platform in different dSPACE products.

Like other registration data, the custom name is stored in the recent platform configuration, but not on the hardware itself. This allows you to reuse the registered platforms (with the user-defined platform names) on other PCs by importing the registration data.

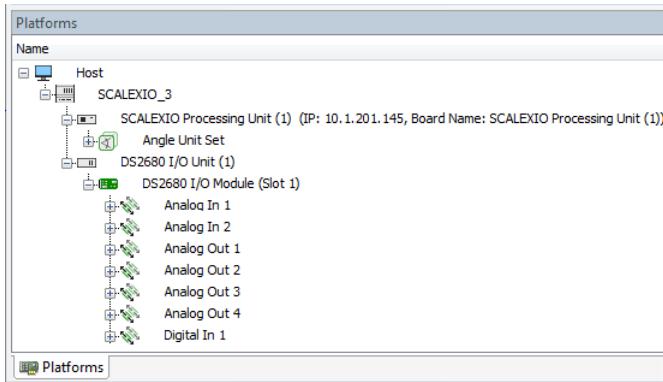
**Note**

If you want to edit or change the custom name after registering the hardware, you must first remove the platform via the **Manage Recent Platform Configuration** dialog and then re-register the hardware.

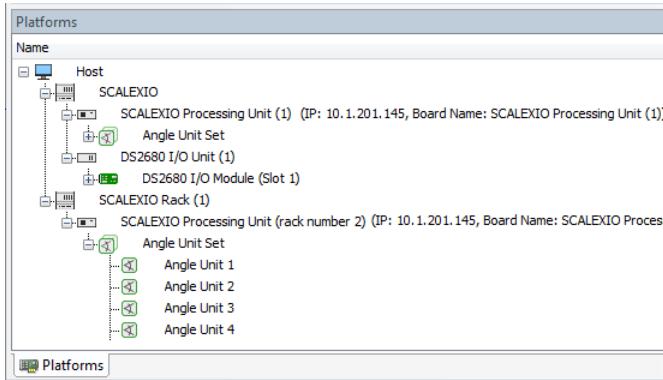
---

**Displaying registered platforms**

Registered platforms are displayed in the Platform Manager if they are physically connected and activated. The Platform Manager displays the hardware topology of each platform in a hierarchical structure with the host PC at the top, followed by the registered hardware system. The hardware system node serves as a container for boards and channels. Each board node serves as a container for channels if present. The board node also indicates the slot position. If a real-time application is loaded to the registered hardware, an application node appears under the processor board node, with the current state (displayed by an icon) of the real-time application.



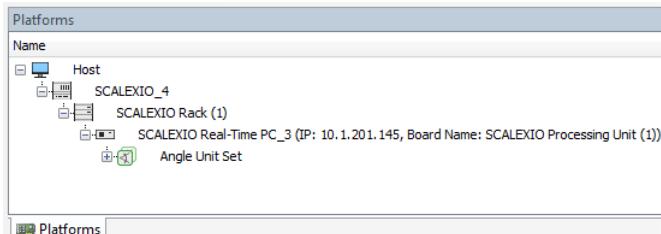
If several platforms are registered, activated and connected to ConfigurationDesk the Platform Manager displays all their hardware topologies in a hierarchical structure. They are displayed in the order in which the corresponding hardware systems are detected in the network.



If you right-click a hardware resource in the Platform Manager, a resource-specific context menu opens. You can use this to download or stop an application, update firmware, etc. The hardware properties of a selected hardware resource are displayed in the Properties Browser.

**Views of the platforms** The Platform Manager provides two views of the platforms. You can select a view on the Platforms ribbon.

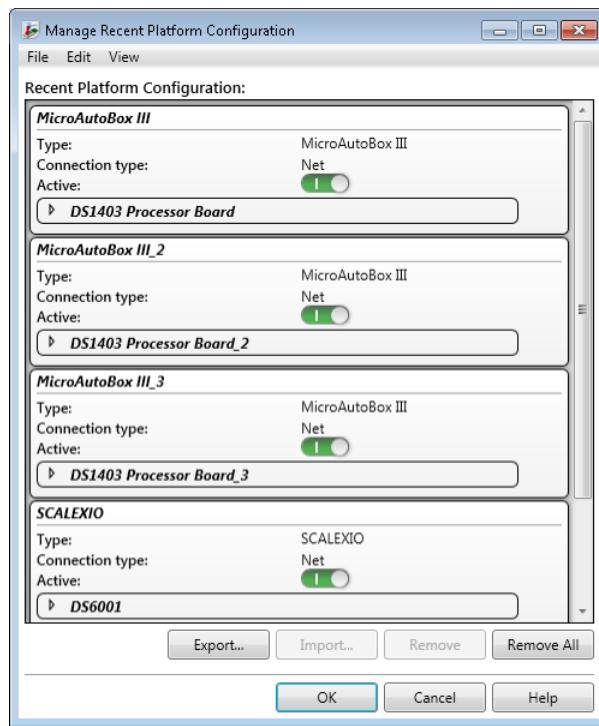
- Assembly view: A mechanical-based view of the hardware systems is displayed, with the SCALEXIO racks or the MicroAutoBox III at the top.



- Network view: The network-based view (logical network) of the hardware systems is displayed, with the processing hardware components (e.g., SCALEXIO Processing Unit, DS1403 Processor Board) at the top.

### Activating, deactivating and removing platforms

Only activated platforms (default setting) are displayed in the Platform Manager. Platforms can be removed from the Platform Manager by clicking their switches in the Manage Recent Platform Configurations dialog. These deactivated platforms disappear from the Platform Manager.



With the Remove button, you can remove the selected platforms from the recent platform configuration, that is, from the registration data stored on your host PC.

#### Note

Individual hardware resources such as boards or channels cannot be removed from the Platform Manager separately. You can remove only entire hardware systems.

### Recovering platforms

Platform information is stored in two files:

- `RecentHardware.xml`
- `RecentHardware_Backup.xml`

Both files are located in:

```
%CommonProgramFiles%\dSPACE\PlatformManagement
```

The `RecentHardware.xml` file is updated each time registered hardware is changed in ConfigurationDesk. Thus, it does not contain platform information

after all the platforms were removed via Clear System on the Platforms ribbon or via Remove All in the Manage Recent Platform Configuration dialog.

The `RecentHardware_Backup.xml` file is updated only when a platform is registered via the Register Platforms dialog. Thus, it still contains registered platforms after all the platforms were removed. You can use the file to restore registered platforms by importing it via the Manage Recent Platform Configuration dialog.

#### Reusing registered platforms

When you register a platform, it is activated by default and displayed in the Platform Manager. All registered platforms (activated and deactivated) are automatically saved and available the next time you open the Manage Recent Platform Configuration dialog. In addition, you can save your settings in a separate XML file using the Export command in the Manage Recent Platform Configuration dialog. This allows you to reuse the registered platforms in other PCs by importing the XML file.

#### Related topics

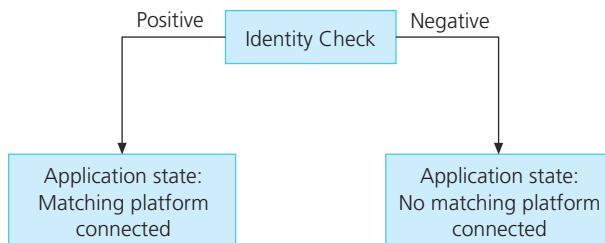
##### HowTos

How to Access Hardware Properties.....	222
How to Change a System Name.....	223
How to Register dSPACE Real-Time Hardware.....	208

## Basics on Connecting a ConfigurationDesk Application to a Hardware System

#### Checking application and hardware system data

If one or more hardware systems are connected to ConfigurationDesk and you activate an application which contains a hardware topology, ConfigurationDesk automatically checks whether the hardware topology of any hardware system displayed in the Platform Manager is identical to the hardware topology of your active application. This check on application and hardware system data can be visualized as follows:



The identity check can have the following results:

- If the active application contains a hardware topology which is identical to the hardware topology of a hardware system displayed in the Platform Manager, the state of the application immediately changes from No matching platform connected to Matching platform connected.

- If the active application contains a hardware topology which is not identical to the hardware topology of any hardware system displayed in the Platform Manager, the state of the application does not change. The application state remains **No matching platform connected**. To change this, you must replace the active hardware topology. For details, refer to [Basics on Hardware Topologies](#) on page 236.
- If the active application only contains an empty hardware topology, the state of the application remains **No matching platform connected**.

Application states are visualized in the status bar.

#### Related topics

##### Basics

<a href="#">Basics on Hardware Topologies</a> .....	236
<a href="#">Registering Hardware in ConfigurationDesk</a> .....	202

##### References

<a href="#">Status Bar (ConfigurationDesk User Interface Reference)</a>
---

## How to Register dSPACE Real-Time Hardware

#### Objective

To access the hardware resources of a hardware system, you must register the hardware system to make it known to ConfigurationDesk. You can register as many hardware systems as you require. Registered hardware systems are treated as platforms which are displayed and which can be accessed via the Platform Manager.

#### Tip

Platform registration data can also be imported to ConfigurationDesk from a separate XML file which contains platform connection data.

#### Registering from other dSPACE software

To register a SCALEXIO or MicroAutoBox III system, you can also use other dSPACE software which supports the related hardware system, for example, ControlDesk or AutomationDesk.

When registering the SCALEXIO or MicroAutoBox III system or managing the registration data, use only one of these software products and close the other products on your host PC. The hardware system is also registered for the other products on your host PC and available the next time you start one of them.

---

**Precondition**

The following preconditions must be fulfilled:

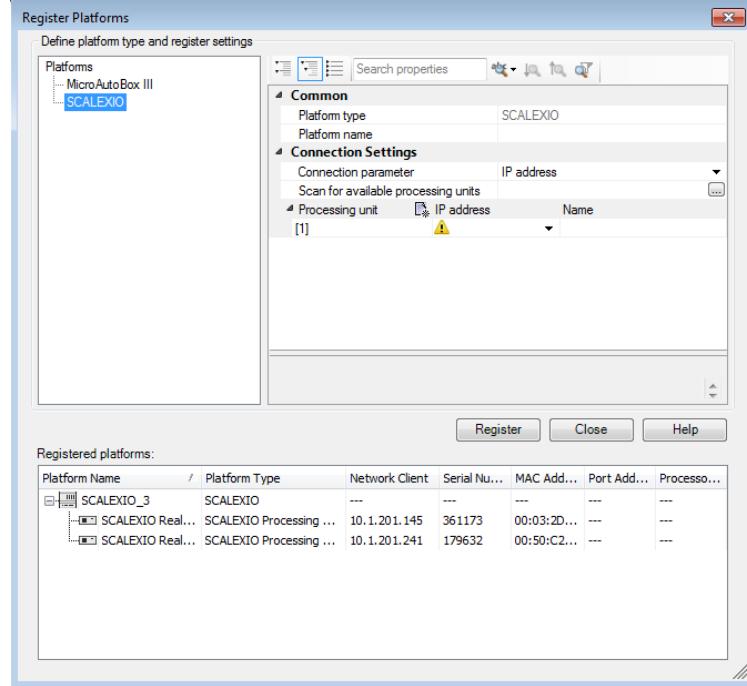
- The hardware system must be connected to the host PC via Ethernet.
- Your host PC must be connected to the same network as the hardware system you want to register in ConfigurationDesk.
- You must know the IP address, or the alias name, or the board name, or the MAC address of the hardware system you want to register in ConfigurationDesk. For details on IP addresses and alias names, refer to:
  - SCALEXIO: [Setting up the Connection to the Host PC \(SCALEXIO Hardware Installation and Configuration\)](#)
  - MicroAutoBox III: [Setting Up the Connection to the Host PC \(MicroAutoBox III Hardware Installation and Configuration\)](#)
- If your hardware system is installed in a different subnetwork connected to your host PC's network via a router or gateway, you must use the IP address for registering. Otherwise ConfigurationDesk is unable to find the hardware system.
- A user-defined platform name must fulfill the following conditions:
  - Allowed characters: A ... Z, a ... z, 0 ... 9, \_
  - Number of characters: max. 64
  - The name must be unique. This means no other registered platform (activated and deactivated) must use this name.

---

**Method****To register dSPACE real-time hardware**

- 1 On the Platforms ribbon, click Platform Management – Register Platforms.

The Register Platforms dialog opens.



- 2** Specify the connection settings for the dSPACE real-time hardware you want to register: Specify either its IP address, or its alias name, or its board name, or its MAC address.

You can also scan the local network for connected platform hardware. To do so, select the Scan local network entry from the IP address drop-down list. The scope is limited to the current subnetwork.

#### Note

- You can register only one platform in the dialog at a time.
- If the dSPACE real-time hardware is installed in a different subnetwork connected to your host PC's network via a router or gateway, you must enter the IP address for registering.

- 3** In the Platform name edit field, specify a unique user-defined platform name if required.
- 4** Click Register to complete the registration of the dSPACE real-time hardware.  
The registered hardware is displayed with its registration settings in the Registered platforms list.
- 5** Repeat steps 2 ... 4 for all the dSPACE real-time hardware you want to register.
- 6** Click Close to close the Register Platforms dialog.

---

<b>Result</b>	<p>You have registered dSPACE real-time hardware independently of ConfigurationDesk projects and applications. ConfigurationDesk created a platform for each item of registered hardware. The registration data is stored in the recent platform configuration.</p> <p>The Platform Manager displays the hardware topology of all the registered platforms. The hardware topologies are displayed in the order in which the registered hardware systems are detected in the network. Each hardware topology is displayed in a hierarchical structure.</p> <p>If a ConfigurationDesk application is loaded, making ConfigurationDesk display hardware can have different effects on its state. For details, refer to <a href="#">Basics on Connecting a ConfigurationDesk Application to a Hardware System</a> on page 207.</p>
<b>Next steps</b>	<p>You can manage the previously registered hardware. For example, ConfigurationDesk lets you remove it from the display in the Platform Manager, or remove it completely from the recent platform configuration, or export the recent platform configuration. Refer to <a href="#">Manage Platforms / Manage Recent Platform Configuration (ConfigurationDesk User Interface Reference)</a>.</p>
<b>Related topics</b>	<p>Basics</p> <p><a href="#">Basics on Hardware Topologies</a>..... 236</p> <p>HowTos</p> <p><a href="#">How to Access Hardware Properties</a>..... 222</p> <p>References</p> <p><a href="#">Register Platforms (ConfigurationDesk User Interface Reference)</a></p>

## How to Register Hardware for Multi-PU Applications and Add it to a ConfigurationDesk Application

---

<b>Objective</b>	To be able to build and download a multi-processing-unit application (multi-PU application), you must add a suitable hardware topology to the Hardware Resources Browser first.
------------------	---

**Note**

For complete processing resource assignment, there must be at least as many processing units in the hardware topology as there are processing unit applications.

**Method****To register hardware for multi-PU applications and add it to a ConfigurationDesk application**

- 1 On the Platforms ribbon, click Platform Management – Register Platforms  
The Register Platforms dialog opens.
- 2 In the Register Platforms dialog, enter a name in the Platform name edit field.

**Note**

The platform name must be unique.

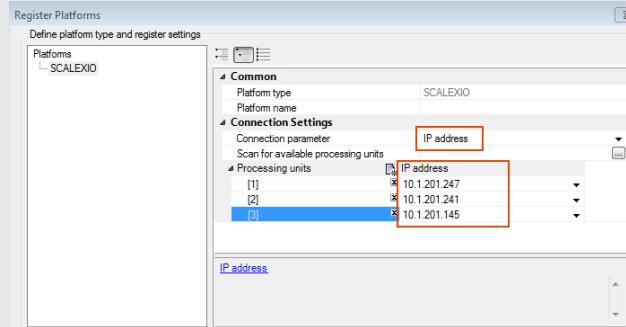
- 3 From the Connection parameter drop-down menu, select the connection parameter.
- 4 Click Scan for available processing units.  
The Scan Local Network for Processing Units dialog opens.
- 5 In the Scan Local Network for Processing Units dialog, select the processing units from the Available processing units list and click . The selected processing units are displayed in the Selected processing units list.
- 6 Click Apply.

**Note**

- The registered processing units must be connected via IOCNET. Otherwise, the download of the multi-processing-unit application is aborted with an error message.
- The firmware versions of the processing units must match. Otherwise, a warning is displayed during registration.

**Tip**

If you know connection parameters (for example, the IPs, MAC addresses, etc.) of the processing units you want to register, you can select the relevant connection parameter from the Connection parameter list, and enter suitable values in the connection parameter's edit field.



For more information, refer to [How to Register dSPACE Real-Time Hardware](#) on page 208.

- 7 From the context menu of the Hardware Resources Browser, select Replace Topology.  
The Replace Hardware Topology dialog opens.
- 8 In the Replace Hardware Topology dialog, select Scan registered hardware and select the platform name you entered in step 2.
- 9 Click OK.

**Result**

You have added the selected hardware topology to your ConfigurationDesk application.

**Note**

- As an alternative, you can create hardware topologies from scratch. For details, refer to [How to Create Hardware Topologies from Scratch](#) on page 241.
- By default, ConfigurationDesk searches for recently registered platforms on startup. If you have registered large multi-processing-unit systems, this may cause undesired delays when ConfigurationDesk is started. To avoid this, you have to clear the Seek connected platforms on startup checkbox on the Platform Management page of the ConfigurationDesk Properties dialog. Refer to [Platform Management Page \(ConfigurationDesk User Interface Reference\)](#).

**Next step**

After you have added a hardware topology to the Hardware Resources Browser, you can assign the processing units to the processing unit applications. Refer to [How to Assign Processing Units to Processing Unit Applications](#) on page 528.

---

<b>Related topics</b>	<b>Basics</b>
	<a href="#">Basics on Multi-Processing-Unit Applications.....</a> 525
	<b>References</b>
	<a href="#">Register Platforms (ConfigurationDesk User Interface Reference)</a>

## Updating the Firmware of SCALEXIO or MicroAutoBox III Hardware

---

<b>Where to go from here</b>	<b>Information in this section</b>
	<a href="#">Basics on Updating SCALEXIO or MicroAutoBox III Firmware.....</a> 214
	<a href="#">How to Update or Repair SCALEXIO or MicroAutoBox III Firmware via ConfigurationDesk.....</a> 219

## Basics on Updating SCALEXIO or MicroAutoBox III Firmware

---

<b>Objective</b>	Your SCALEXIO or MicroAutoBox III hardware always contains the latest firmware available at the time of delivery. dSPACE updates the firmware in a continuous development process to support new features or to fix bugs.  Working with the SCALEXIO or MicroAutoBox III system might be restricted if you work with out-of-date firmware.
------------------	--

**Firmware Update Wizard**

ConfigurationDesk provides access to the Firmware Update Wizard to update and repair the firmware of the SCALEXIO or MicroAutoBox III hardware.

**Note**

The Update Firmware Wizard supports SCALEXIO systems as of dSPACE Release 2015-B. If you want to update the firmware version on a SCALEXIO system to an earlier version, you have to use ConfigurationDesk from an earlier dSPACE Release.

**Distribution of firmware**

Firmware updates are distributed via firmware archives in the regular dSPACE Releases. You can find the latest firmware archives on the dSPACE website at <http://www.dspace.com/go/firmware>. dSPACE recommends to check the status of your firmware periodically and to update the firmware to the latest version.

**Contents of the SCALEXIO firmware archive**

**(SCALEXIOFwArchive.arc)** The firmware archive for SCALEXIO systems with a Linux operating system contains the following firmware components:

- DSx86\_32 UserFirmware
- DSx86\_32 FactoryFirmware
- DS2502 UserFpga
- DS2551 UserFpga
- DS2601 UserFirmware and DS2601 UserFpga
- DS2621 UserFirmware and DS2621 UserFpga
- DS2642 UserFirmware and DS2642 UserFpga
- DS2655 UserFirmware and DS2655 UserFpga
- DS2655M1 UserFpga
- DS2655M2 UserFpga
- DS2656 UserFirmware and DS2656 UserFpga
- DS2671 UserFirmware and DS2671 UserFpga
- DS2672 UserFirmware and DS2672 UserFpga
- DS2680 UserFirmware, DS2680 UserFpga, and DS2680 IoFpga1 ... 3
- DS2690 UserFirmware and DS2690 UserFpga
- DS2907 UserFirmware and DS2907 UserFpga
- DS6001 UserFirmware, DS6001 UserIplFirmware, and DS6001 UserFpga
- DS6051 UserFpga
- DS6071 UserFirmware
- DS6072 UserFirmware
- DS6073 UserFirmware
- DS6101 UserFirmware, DS6101 UserIplFirmware, and DS6101 UserFpga
- DS6121 UserFpga
- DS6201 UserFpga
- DS6202 UserFpga

- DS6221 UserFpga
- DS6241 UserFpga
- DS6301 UserFpga
- DS6311 UserFpga
- DS6321 UserFpga
- DS6333-CS UserFpga
- DS6333-PE UserFpga
- DS6335-CS UserFpga
- DS6341 UserFpga
- DS6342 UserFpga
- DS6351 UserFpga
- DS6601 UserFpga
- DS6602 UserFpga
- DS6651 UserFpga
- Dsx86\_32 HypervisorFirmware

**Tip**

The term *UserFirmware* is used for the standard firmware. Do not mix it up with firmware provided by the user.

**Note**

The archive format for SCALEXIO changed with Firmware Archives 2.1 contained in dSPACE Release 2016-A. To open an archive in the new format, you must use Firmware Manager 2.1 or later.

**Contents of the MicroAutoBox III firmware archive**

**(DS1403FwArchive.arc)** The firmware archive for MicroAutoBox III contains the following firmware components:

- ADC Type4 UserFpga
- AIO Type1 UserFpga
- CAN Type1 UserFirmware
- DIO Type3 UserFpga
- DIO Type4 UserFpga
- DS1403 CnFirmware
- DS1403 CnIpl
- DS1403 UserCpld
- DS1403 UserFirmware
- DS1403 UserFpga
- DS1403 UserIpl
- DS1514 FPGA Base Board IoCpld
- DS1521 UserFpga
- DS4342 CAN FD Interface Module UserFpga

- FPGA Type1 IoFpga (DS1552)
- FPGA Type1 IoFpga (DS1554)

#### Notes on firmware versions

##### Note

- With dSPACE Release 2020-B, dSPACE changes the SCALEXIO firmware from a QNX-based to a Linux-based distribution. If you update to this firmware version, the following items built for dSPACE Release 2020-A and earlier are no longer compatible with the SCALEXIO system and must be (re-)built from source code based on dSPACE Release 2020-B:
  - Real-time applications
  - Binary libraries contained in model containers (i.e., SIC, BSC, FMU, and CTLGZ files)
  - Binary libraries referenced by Simulink models
  - Binary libraries referenced by ConfigurationDesk applications via custom code settings or custom I/O functions
- All the components of a SCALEXIO or MicroAutoBox III system (for example, processing hardware and I/O boards) must contain the same firmware version to function properly. If a component of a system is changed later, for example, to a newer hardware version, you have to update the firmware to a consistent status. The dSPACE software does not support hardware with different firmware versions in one system.
- The firmware is updated to the firmware version which is available within the current RCP and HIL software installation that usually also installs the ConfigurationDesk version. However, if your SCALEXIO or MicroAutoBox III hardware contains a newer firmware version than your current RCP and HIL software installation, a firmware update actually leads to a firmware downgrade.

Firmware version deviations for a SCALEXIO or MicroAutoBox III system are indicated in the Platform Manager. The affected hardware components are marked by the  symbol. You can find associated warning messages in the Message Viewer.

##### Tip

To get information on the firmware versions of the SCALEXIO or MicroAutoBox III hardware components, you can check their properties in the Properties Browser in ConfigurationDesk.

#### Notes on the firmware update process

- Before updating the firmware, think through the effects of updating. For example, note the firmware's compatibility (see below) and think through the effects of updating a system in a network when several users work with one system.
- Before you start a firmware update, you have to decide whether to update the firmware to a later version or to repair the currently installed firmware version.

- Before updating the firmware, ensure that all the components of a SCALEXIO or MicroAutoBox III system (for example, real-time PC and I/O boards) are connected and switched on and are displayed in ConfigurationDesk's Platform Manager. Only the firmware of hardware components that are displayed can be updated.
- Interruptions in the update process (for example, if the connection between the host PC and the SCALEXIO or MicroAutoBox III hardware is interrupted) disable the hardware. If the firmware update is interrupted, you have to restart the update process.
- If you have registered a multi-processing-unit system (multi-PU system), you can update only one processing unit at a time.
- To complete the firmware update, you must restart the SCALEXIO or MicroAutoBox III hardware before you can continue working with the system.

---

#### **Notes on firmware compatibility for building real-time applications**

A dSPACE installation provides the files that are necessary to build the real-time application and the firmware files for the hardware. These files must be compatible with each other. If the real-time application is built with the same dSPACE installation that the firmware version of the hardware comes from, the real-time application is fully compatible with the firmware. You can download the real-time application without restriction.

If the real-time application was built with an earlier dSPACE installation and the firmware version of the hardware comes from a later dSPACE installation, it is possible to download the real-time application to the hardware. Later firmware versions are downward-compatible, so they are always compatible with earlier real-time applications.

If the real-time application was built with a later dSPACE installation and the firmware version of the hardware comes from an earlier dSPACE installation, it is not possible to download the real-time application. dSPACE tools prevent the download as compatibility is not guaranteed. To download the real-time application, you must first update the firmware with ConfigurationDesk (refer to [How to Update or Repair SCALEXIO or MicroAutoBox III Firmware via ConfigurationDesk](#) on page 219) or ControlDesk (refer to [Update Firmware \(ControlDesk Platform Management\)](#)).

---

#### **Limitations for updating SCALEXIO firmware**

Note the following restrictions when you want to update the firmware of your SCALEXIO system:

- If your SCALEXIO system contains a DS2655M2 Digital I/O Module, a firmware update from firmware version 3.2 or earlier to a firmware version 3.3 or later, might lead to an error and the update process is then stopped.
- To finish the firmware update you have to do the following steps:
- Restart the SCALEXIO system.
  - Call the [Refresh Interface Connections \(ConfigurationDesk User Interface Reference\)](#) command in the Platform Manager.
  - Repeat the firmware update process.
  - If your SCALEXIO system contains one or more new hardware components that are not already supported by the currently active firmware on the

SCALEXIO Processing Unit or DS6001 Processor Board, a firmware update might lead to an error and the update process is then aborted.

To finish the firmware update, perform the following steps:

- Restart the SCALEXIO system.
- Call the **Refresh Interface Connections** (ConfigurationDesk User Interface Reference ) command in the Platform Manager.
- Repeat the firmware update process.

---

## Related topics

### HowTos

How to Update or Repair SCALEXIO or MicroAutoBox III Firmware via ConfigurationDesk.....	219
--	-----

### References

Update Firmware (ControlDesk Platform Management 
--

## How to Update or Repair SCALEXIO or MicroAutoBox III Firmware via ConfigurationDesk

---

### Basics

For important notes on the update process, limitations and compatibility information, refer to **Basics on Updating SCALEXIO or MicroAutoBox III Firmware** on page 214.

---

### Result of firmware update

#### Note

The firmware is updated to the firmware version, which is available within the current RCP and HIL software installation that usually also installs your corresponding ConfigurationDesk version. However, if your SCALEXIO or MicroAutoBox III hardware contains a newer firmware version than your current RCP and HIL software installation, a firmware update actually leads to a downgrade of the firmware.

---

### Precondition

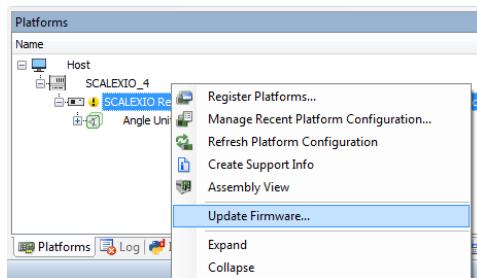
Hardware must be registered in ConfigurationDesk and must be displayed in the Platform Manager. Only the firmware of hardware components that are displayed can be updated.

**Note**

The archive format for SCALEXIO changed with Firmware Archives 2.1 contained in dSPACE Release 2016-A. To open an archive in the new format, you must use Firmware Manager 2.1 or later.

**Method****To update or repair SCALEXIO or MicroAutoBox III firmware via ConfigurationDesk**

- 1 In the Platform Manager, right-click the processing hardware.
- 2 From the context menu, select Update Firmware.



The Update Firmware Wizard with the Select Mode dialog opens.

- 3 Select the operation mode:
  - **Firmware update mode:** The firmware handling process is configured for updating all firmware components of the real-time hardware to a later version.
  - **Firmware repair mode:** The firmware handling process is configured for repairing the selected firmware components of the real-time hardware by reloading the same firmware versions.
- 4 Click Next to continue with the Select Firmware Archive dialog.  
The latest firmware archive for the selected platform is automatically set. Optionally, browse for another firmware archive. This might be useful, for example, if you want to update to a firmware version other than the latest.
- 5 Click Next to continue with the Select Firmware Components dialog.
- 6 Depending on the desired update mode start the update process as follows:
  - **Update firmware:**
    1. In the Select Firmware Components dialog, click Update to start the firmware update process.

In the Update column, the firmware components to be updated are marked and red. The components are not marked for update if the version of the currently installed firmware is identical to or later than the firmware available in the specified firmware archive.
  - **Repair firmware:**
    1. In the Select Firmware Components dialog, select the firmware components to be repaired in the Update column.

You can select only firmware components, whose current and available versions are identical. If the versions differ, the components are not displayed at all.

2. In the Select Firmware Components dialog, click Repair to start the firmware repair process.

This command is enabled only if at least one firmware component is selected for repairing.

If there are updatable firmware components, the update process starts. You can see the progress in the Status column. The initial '--' entry is replaced by a percentage. If the progress information cannot be detected continuously, only the states 50% and 100% are displayed. If the process successfully finished, an OK is shown, otherwise an error message is displayed.

If the firmware update will require more than 40 minutes, an estimate of the time is displayed. Then you can decide whether to start the process.

Interrupting a running firmware update process is not possible.

#### **NOTICE**

##### **Switching off the hardware during the firmware update.**

Risk of corrupted firmware.

Switch off the hardware only after the update process ended successfully or an error message is displayed.

- 7 Power down the SCALEXIO or MicroAutoBox III system and restart it.

---

#### **Result**

The firmware update for the SCALEXIO or MicroAutoBox III hardware is complete.

## Configuring Hardware Properties

---

#### **Objective**

Each hardware resource (for example, boards or their channels) provides properties via the Properties Browser.

---

#### **Where to go from here**

#### **Information in this section**

[How to Access Hardware Properties.....](#) 222

[How to Change a System Name.....](#) 223

**Information in other sections**

[How to Provide Data of Internal Loads to the Hardware.....](#) 397

## How to Access Hardware Properties

<b>Objective</b>	ConfigurationDesk allows you to access details of a displayed hardware system, such as board or channel properties, in the Properties Browser to view and check them.
------------------	---

<b>Accessible properties</b>	The properties which are available for displayed hardware systems are related to, for example, identification information, installation location of boards, or electrical characteristics of channels. For a complete description of the accessible properties, refer to <a href="#">ConfigurationDesk User Interface Reference</a> .
------------------------------	---

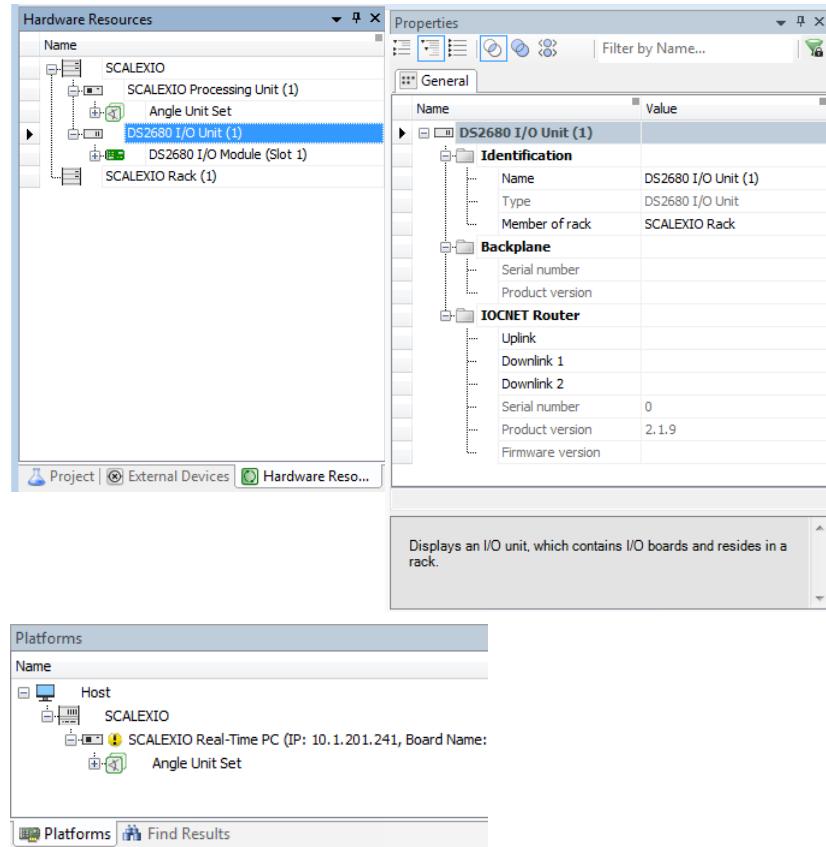
**Tip**

Not all properties of channels are read-only. Some of them can also be configured, for example, Load description and Load rejection enforcement. For instructions, refer to [How to Provide Data of Internal Loads to the Hardware](#) on page 397.

<b>Precondition</b>	Hardware must be registered in ConfigurationDesk. For instructions, refer to <a href="#">How to Register dSPACE Real-Time Hardware</a> on page 208.
---------------------	---

<b>Method</b>	<b>To access hardware properties</b> <ol style="list-style-type: none"><li>1 In the Platform Manager or Hardware Resource Browser, select the hardware resource whose properties you want to access.</li></ol>
---------------	---

- 2 The properties of the hardware resource are displayed in the Properties Browser.

**Result**

You have accessed properties of the selected hardware resource.

**Related topics****References**

[ConfigurationDesk Hardware Resource Properties](#)

## How to Change a System Name

**Objective**

When the hardware is delivered, it has default system names. These are displayed in ConfigurationDesk, where you can change them as required.

**Display of system names**

System names are displayed in the Platform Manager.

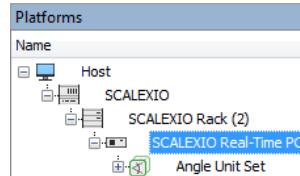
- 
- Allowed characters**
- All characters are allowed, including ? < > : / \ " . \*
  - The number of characters is limited and depends on the characters used (maximum of 32 characters).

- 
- Precondition** Hardware without a loaded application must be registered in ConfigurationDesk.

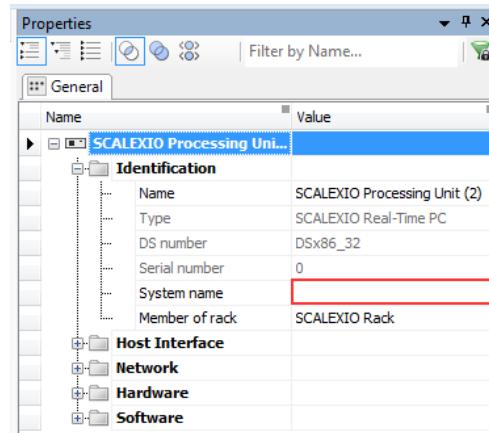
---

**Method** **To change a system name**

- 1 In the Platform Manager, select the system.



The Properties Browser displays the properties of the selected system.



- 2 Click the system name edit field.  
 3 Enter a new system name and press **Enter** or click anywhere outside the edit field.

- 
- Result** The system name is changed and stored on the hardware. The new name is displayed in the Platform Manager. Note that changing the system name does not affect existing alias names.

**Related topics****Basics**

[Basics on Hardware Topologies](#)..... 236

**References**

[Platform Properties \(ConfigurationDesk Hardware Resource Properties\)](#)

## Configuring the I/O Ethernet Communication

**Where to go from here****Information in this section**

[How to Configure Internal Ethernet Switches](#)..... 225

Configuring the Ethernet connections between the Ethernet controllers of the dSPACE hardware and the ports.

[How to Configure I/O Ethernet Ports](#)..... 227

Configuring the physical layer (PHY) of the I/O Ethernet ports, such as the data rate.

## How to Configure Internal Ethernet Switches

**Objective**

Configuring the Ethernet connections between the Ethernet controllers of the dSPACE hardware and the ports.

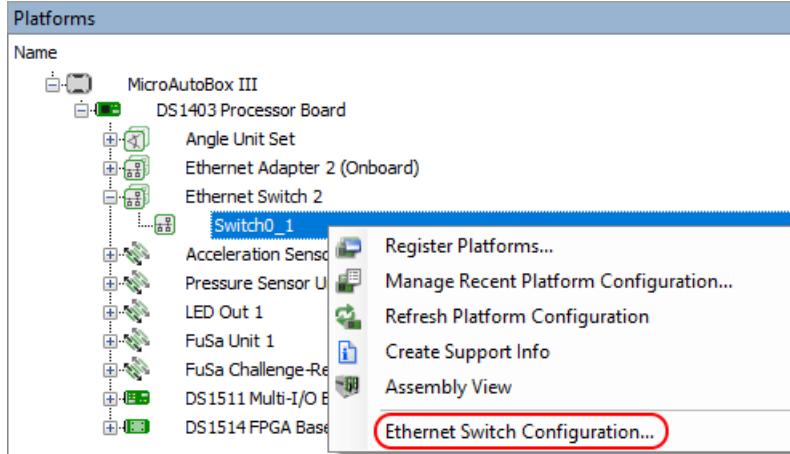
**Possible methods**

Use one of the following methods:

- Select a predefined Ethernet switch configuration via the Platform Manager as described below. This configuration takes effect if no real-time application is running or the running real-time application does not provide a configuration.
- Configure the internal Ethernet switch via a real-time application. This configuration is used as long as the real-time application is running. Refer to [Ethernet Switch \(ConfigurationDesk I/O Function Implementation Guide\)](#) .

**Method****To configure the internal Ethernet switch via the Platform Manager**

- 1 In the Platform Manager, right-click the Ethernet switch to be configured. The following illustration shows the MicroAutoBox III as an example.



- 2 Click Ethernet Switch Configuration.

ConfigurationDesk opens the Ethernet Switch Configuration dialog which lets you specify and manage the Ethernet switch configuration.

- 3 Select a predefined configuration mode.

The information on the provided configurations is provided as tooltips. Move the mouse pointer over a configuration mode to open the tooltip with the respective information.

- 4 Click OK to save the selected Ethernet switch configuration.

**Result**

You selected a predefined Ethernet switch configuration for the dSPACE hardware that will be used each time the dSPACE hardware is switched on.

However, if a real-time application is running with its own Ethernet switch configuration, the configuration of the real-time application is used while the application is running.

**Related topics****Basics**

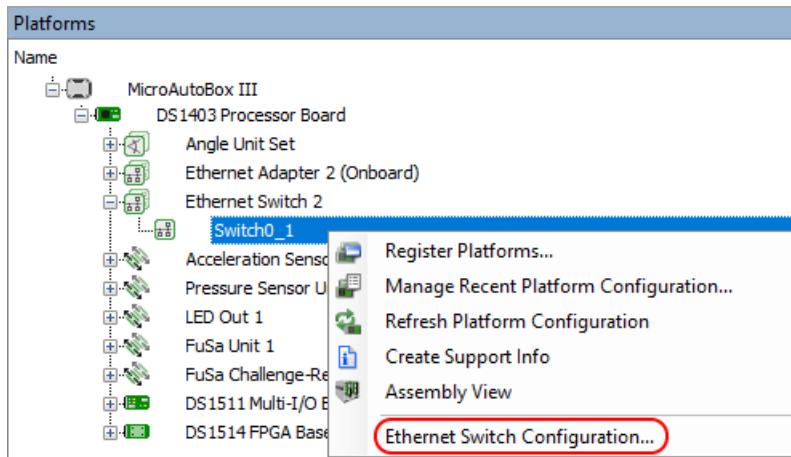
- [Basics on the Internal Ethernet Switch \(MicroAutoBox III Hardware Installation and Configuration\)](#)
- [Ethernet Board Configuration Page \(SCALEXIO Hardware Installation and Configuration\)](#)
- [Ethernet Board Configuration Page \(SCALEXIO Hardware Installation and Configuration\)](#)

## How to Configure I/O Ethernet Ports

<b>Objective</b>	Configuring the physical layer (PHY) of the I/O Ethernet ports, such as the data rate.
------------------	--

<b>Possible methods</b>	Use one of the following methods:
	<ul style="list-style-type: none"> <li>▪ Configure the I/O Ethernet ports that are connected to internal Ethernet switches via the Platform Manager as described below. This configuration takes effect if no real-time application is running or the running real-time application does not provide a configuration.</li> <li>▪ Configure the I/O Ethernet ports via a real-time application. This configuration is used as long as the real-time application is running. Refer to <a href="#">Ethernet Switch (ConfigurationDesk I/O Function Implementation Guide)</a>.</li> </ul>

<b>Method</b>	<b>To configure the I/O Ethernet ports</b>
	<p>1 In the Platform Manager, right-click the Ethernet switch to be configured. The following illustration shows the MicroAutoBox III as an example.</p>



- 2 Click Ethernet Switch Configuration.
- ConfigurationDesk opens the Ethernet Switch Configuration dialog. The lower part of the dialog contains the PHY configurations of the single Ethernet ports that are connected to the Ethernet switch.
- 3 Select the Ethernet port.
  - 4 Select the PHY configuration for the Ethernet port:
    - Enable/disable the PHY of the selected port via the Power property.
    - Enable the autonegotiation mode to automatically detect the data rate and duplex mode, or select a data rate and duplex mode.
    - Set the PHY of each automotive Ethernet port to master or slave.

To establish a link between two automotive Ethernet ports, one PHY must be the master, the other the slave.

- 5 Click OK to save the selected port configuration.
- 

**Result**

You configured the physical layer of the I/O Ethernet ports. The configuration will be used each time the dSPACE hardware is switched on.

However, if a real-time application is running with its own physical layer configuration, the configuration of the real-time application is used while the application is running.

---

**Related topics**

**Basics**

[Basics on the Internal Ethernet Switch \(MicroAutoBox III Hardware Installation and Configuration\)](#)  
[Ethernet Board Configuration Page \(SCALEXIO Hardware Installation and Configuration\)](#)  
[Ethernet Board Configuration Page \(SCALEXIO Hardware Installation and Configuration\)](#)

# Platform Access by Several Users Simultaneously

<b>Objective</b>	Several dSPACE products (also ConfigurationDesk) support synchronized platform management. The platform management instances in these products are synchronized.  The SCALEXIO or MicroAutoBox III system can be accessed by several ConfigurationDesk, ControlDesk, and AutomationDesk users simultaneously. There are some points to note when working with real-time hardware which is also accessed by other users at the same time.
------------------	--

Where to go from here	Information in this section
	<p><a href="#">Synchronized Platform Management with Several dSPACE Products</a>..... 229</p> <p><a href="#">Accessing the SCALEXIO or MicroAutoBox III System with Multiple dSPACE Products Simultaneously</a>..... 232</p> <p>You can access a SCALEXIO or MicroAutoBox III system with multiple dSPACE products at the same time, if they run on the same PC, and you can access a SCALEXIO or MicroAutoBox III system from a second PC by using ControlDesk.</p>

## Synchronized Platform Management with Several dSPACE Products

<b>Introduction</b>	Several dSPACE products support synchronized platform management. The platform management instances in these products are synchronized.
<b>Synchronization of platform management instances</b>	<p>Several dSPACE products have a Platform Manager that displays all the registered platforms with their components and running applications that can be accessed via the products. There are functions to register platforms, to manage the platform configuration, and to handle the real-time applications loaded to the platforms.</p> <p>If you work simultaneously with several of these dSPACE products, each of them has its own platform management instance running. The instances contain consistent information about the connected platforms. This means that when you perform a platform management activity in one instance, the contents of all the other currently running platform management instances are synchronized accordingly.</p>

**Tip**

A platform management instance provides information on platforms and the applications loaded to them only for those platforms that are supported by the respective dSPACE product.

**Performing platform management activities**

The following table shows the platform management activities that are synchronized between the platform management instances. You can see which platform management activities are possible even if another platform management activity in another dSPACE product is currently running. Depending on the activity you perform, simultaneous access to the hardware and real-time applications from several dSPACE products can be restricted, because exclusive access to a single platform or to all registered platforms might be necessary.

**Note**

Not every platform management activity is available in every dSPACE product.

Activity You Want to Perform in Platform Management Instance A	Activity That is Currently Running in Platform Management Instance B											
	Register Platforms	Refresh Interface Connections	Refresh Platform Configuration	Clear System	Manage Recent Platform Configuration	Load/Reload Real-Time Application	Stop RTP(s)	Unload Real-Time Application	Update Firmware	Clear Flash	Explore Logged Data	Online Calibration is Started
Register Platforms	-	-	-	-	-	✓	✓	✓	✓	✓	✓	✓
Refresh Interface Connections	-	-	-	-	-	-	-	-	-	-	-	
Refresh Platform Configuration	-	-	-	-	✓	✓	✓	✓	✓	✓	✓	
Clear System	-	-	-	-	-	-	-	-	-	-	-	
Manage Recent Platform Configuration	-	-	-	-	-	-	-	-	-	-	-	
Load Real-Time Application	✓	-	✓	-	-	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	
Stop RTP(s)	✓	-	✓	-	-	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	
Unload Real-Time Application	✓	-	✓	-	-	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	
Update Firmware	✓	-	✓	-	-	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	
Clear Flash	✓	-	✓	-	-	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	✓ <sup>1)</sup>	

Activity You Want to Perform in Platform Management Instance A	Activity That is Currently Running in Platform Management Instance B
	<b>Register Platforms</b> <b>Refresh Interface Connections</b> <b>Refresh Platform Configuration</b> <b>Clear System</b> <b>Manage Recent Platform Configuration</b> <b>Load/Reload Real-Time Application</b> <b>Stop RTP(s)</b> <b>Unload Real-Time Application</b> <b>Update Firmware</b> <b>Clear Flash</b> <b>Explore Logged Data</b> <b>Online Calibration is Started</b>
Explore Logged Data	✓ – ✓ – – ✓ <sup>1)</sup>
Go Online / Start Online Calibration	✓ – ✓ – – ✓ <sup>1)</sup>

<sup>1)</sup> Only possible for different platforms since the activity you want to perform requires exclusive platform access.

## dSPACE products supporting platform management synchronization

Platform management synchronization is performed when you access dSPACE platforms *simultaneously* using any combination of the following software products:

- AutomationDesk as of Version 3.6p2  
Note that automated access via Platform Management library and XIL API library might also require exclusive access to a platform.
- ConfigurationDesk as of Version 4.4
- ControlDesk as of Version 5.0
- Firmware Manager as of Version 1.0
- ModelDesk as of Version 3.1
- RTT Manager as of Version 2.1

## Firewall settings

To enable platform management synchronization, the firewalls of the PCs must be configured to allow communication for simultaneous platform access.

**Windows firewalls** During the installation of dSPACE software, Windows firewalls are automatically configured to allow communication between the platform management instances for synchronization. You do not have to configure Windows firewalls manually. The first time you start a product involved in platform management synchronization, the firewall asks you to allow access. Confirm the product as trusted software.

**Other firewalls** If the host PC has a firewall different from the Windows firewall, configure that firewall manually to allow communication between the platform management instances of the products.

Related topics

Basics

Accessing the SCALEXIO or MicroAutoBox III System with Multiple dSPACE Products Simultaneously.....	232
---	-----

## Accessing the SCALEXIO or MicroAutoBox III System with Multiple dSPACE Products Simultaneously

**Accessing the SCALEXIO or MicroAutoBox III system from one PC**

You can access a SCALEXIO or MicroAutoBox III system with the following dSPACE products at the same time if they run on the same PC:

- ConfigurationDesk
- ControlDesk
- AutomationDesk
- ModelDesk
- dSPACE XIL API MAPort implementation

**Note**

The products to access the SCALEXIO or MicroAutoBox III system must be installed from the same dSPACE Release.

**Accessing the SCALEXIO or MicroAutoBox III system from a second PC**

Another person can also access the SCALEXIO or MicroAutoBox III system by using ControlDesk running on a second PC.

**Note**

Consider the following restrictions when accessing the system from a second PC:

- Do not execute platform management functions from a second PC.  
These are functions such as:
  - Downloading a real-time application to the SCALEXIO or MicroAutoBox III system
  - Starting/stopping the real-time application on the SCALEXIO or MicroAutoBox III system
- Access to the SCALEXIO or MicroAutoBox III system from a second PC is restricted to measurement and recording tasks:
  - Do not use ControlDesk's Signal Editor from a second PC to stimulate variables of the real-time application running on the SCALEXIO or MicroAutoBox III system
  - Do not use Real-Time Testing from a second PC to perform tests synchronously to the real-time application running on the SCALEXIO or MicroAutoBox III system
  - Do not use ControlDesk's Failure Simulation Module from a second PC to control the failure simulation hardware of the SCALEXIO system
- The products to access the SCALEXIO or MicroAutoBox III system (incl. ControlDesk running on a second PC) must be installed from the same dSPACE Release.

**⚠ CAUTION**

When you use AutomationDesk to carry out automated tests on a SCALEXIO or MicroAutoBox III system, do not use ControlDesk to change the values of model parameters that influence the currently running test, because this will falsify the results.

The person carrying out automated tests will not even be able to detect whether the SCALEXIO or MicroAutoBox III system is currently being accessed by ControlDesk.

**Access levels**

Access levels are provided by the system for accessing the:

- SCALEXIO or MicroAutoBox III system
- Real-time application (executable on the SCALEXIO or MicroAutoBox III system)

Depending on the action you perform, other users' access to the hardware and real-time application can be restricted. There are two access levels:

- Exclusive access  
Other users have no access.
- Shared access  
Other users have shared but not an exclusive access. Actions which need shared access can be carried out by several users at the same time.

#### Access required for specific actions

The following table shows actions in ConfigurationDesk, ControlDesk, AutomationDesk and ModelDesk for which each user needs exclusive or shared access to the hardware and/or to the real-time application.

Action	Required Access Level	
	To SCALEXIO or MicroAutoBox III System	To Real-Time Application
Registering hardware	Exclusive	None
Changing properties of the hardware <sup>1)</sup>	Change the system name or names of hardware components (for example, real-time PC or I/O unit)	Exclusive
	Change internal load description of a channel <sup>[3), 4)</sup>	Exclusive
	Change load rejection settings of a channel <sup>[3), 4)</sup>	Exclusive
Handling real-time applications	Load application	Shared
	Unload application	Shared
	Start application	Shared
	Stop application	Shared
ControlDesk's device state changes <sup>5)</sup>	Start online calibration	Shared
	Start measuring/recording	Shared
Updating the firmware	Exclusive	None <sup>2)</sup>

<sup>1)</sup> If you change a property, the hardware is accessed only when the new value is applied by ConfigurationDesk. The more properties are changed, the longer the update (and therefore the access) takes.

<sup>2)</sup> Not possible when a real-time application is loaded

<sup>3)</sup> Only possible in ConfigurationDesk

<sup>4)</sup> Not applicable for MicroAutoBox III

<sup>5)</sup> Only possible in ControlDesk. Each access level applies for the entire duration of the corresponding device state.

#### Tip

The required access levels are independent of the status of the ConfigurationDesk application (Connected to hardware or Not connected to hardware).

**Examples of handling real-time applications**

- While you *load* a real-time application, other users cannot unload it. However, they can start and stop it.
  - While you *start* a real-time application, other users can access it, for example, to stop it.
  - While one user starts ControlDesk's online calibration, other users cannot load or unload the real-time application. However, they can start and stop it.
- 

**Error messages**

Whenever access is not possible, ConfigurationDesk, ControlDesk, AutomationDesk or ModelDesk displays an error message.

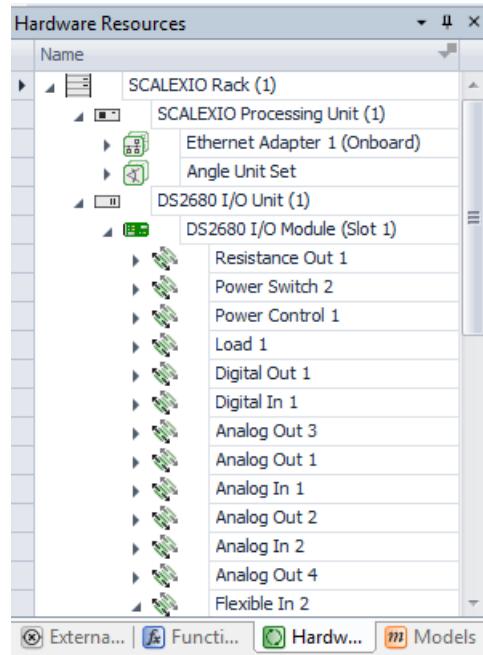
# Working with Hardware Topologies

<b>Objective</b>	Hardware topologies are required for connecting ConfigurationDesk applications to registered hardware systems.														
<b>Where to go from here</b>	<b>Information in this section</b>														
	<table> <tr> <td>Basics on Hardware Topologies.....</td> <td>236</td> </tr> <tr> <td>How to Create Hardware Topologies from Scratch.....</td> <td>241</td> </tr> <tr> <td>How to Extend Existing Hardware Topologies .....</td> <td>243</td> </tr> <tr> <td>How to Export a Hardware Topology Part.....</td> <td>244</td> </tr> <tr> <td>How to Merge Hardware Topologies.....</td> <td>245</td> </tr> <tr> <td>How to Establish a Network Connection in the Hardware Topology (SCALEXIO).....</td> <td>246</td> </tr> <tr> <td>How to Assign Boards to Specific Slots in an I/O Slot Unit.....</td> <td>248</td> </tr> </table>	Basics on Hardware Topologies.....	236	How to Create Hardware Topologies from Scratch.....	241	How to Extend Existing Hardware Topologies .....	243	How to Export a Hardware Topology Part.....	244	How to Merge Hardware Topologies.....	245	How to Establish a Network Connection in the Hardware Topology (SCALEXIO).....	246	How to Assign Boards to Specific Slots in an I/O Slot Unit.....	248
Basics on Hardware Topologies.....	236														
How to Create Hardware Topologies from Scratch.....	241														
How to Extend Existing Hardware Topologies .....	243														
How to Export a Hardware Topology Part.....	244														
How to Merge Hardware Topologies.....	245														
How to Establish a Network Connection in the Hardware Topology (SCALEXIO).....	246														
How to Assign Boards to Specific Slots in an I/O Slot Unit.....	248														

## Basics on Hardware Topologies

<b>Hardware topology of the ConfigurationDesk application</b>	The hardware topology of the active ConfigurationDesk application contains information on the components of a specific hardware system and on channel properties, such as board and channel types and slot numbers. It allows you to work with hardware in your application without a hardware system being displayed in the Platform Manager, if necessary.  Hardware topologies are particularly important for connecting ConfigurationDesk applications to hardware systems and for assigning hardware resources to function blocks.
<b>Displaying hardware topologies</b>	Hardware topologies of ConfigurationDesk applications are displayed in the Hardware Resource Browser.  The Hardware Resource Browser provides two views of the displayed hardware topologies. You can switch between the different views via the Switch to Assembly View or Switch to Network View context menu commands: <ul style="list-style-type: none"><li>▪ Assembly view A mechanical-based view of the hardware topology is displayed. The browser displays the hardware topology of your active application in a hierarchical structure, for example, with the SCALEXIO rack at the top. The SCALEXIO rack</li></ul>

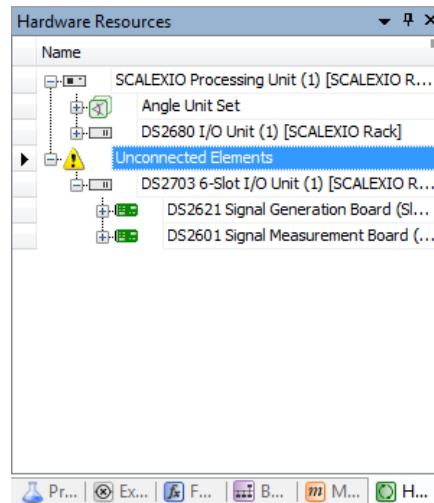
node serves as a container, for example, for units/boxes. An I/O module (or board) node serves as a container for channel sets (channel types and their I/O channels) as shown below.



- Network view

A network-based view (logical network) of the hardware topology is displayed, with the SCALEXIO processing units and processor boards at the top.

If the hardware topology contains units/boxes that are not connected to a processing unit, these units/boxes are identified and displayed below the Unconnected Elements node as shown below.



You can establish the required connections via the Uplink property of the affected units or via the Downlink property of the respective parent units/boxes.

If you right-click a hardware resource in the Hardware Resource Browser, a context menu with resource-specific commands opens.

## Creating and extending hardware topologies

You can create new hardware topologies from scratch or extend existing ones. You can then perform tasks such as implementing and building a real-time application for hardware which is not yet physically available. However, if you want to download and execute the real-time application, you need real hardware.

When you create and extend hardware topologies, ConfigurationDesk provides a set of suitable hardware components to add to your hardware topology. Each board you add to your hardware topology is automatically assigned a slot number, which you can change if it has been added to an I/O slot unit. ConfigurationDesk lets you only create a hardware topology (with components inside) which complies with the rules for building real hardware systems.

For instructions, refer to:

- [How to Create Hardware Topologies from Scratch](#) on page 241
- [How to Extend Existing Hardware Topologies](#) on page 243

For details on the possible arrangements of the components in the SCALEXIO system and a description of these components, refer to [Standard I/O Boards \(SCALEXIO Hardware Installation and Configuration\)](#).

For MicroAutoBox III components, refer to [Hardware Components of the MicroAutoBox III and their Features \(MicroAutoBox III Hardware Installation and Configuration\)](#).

## Exporting and importing hardware topologies

You can export and import hardware topologies by using hardware topology files. You can also scan registered hardware platforms to automatically create a matching hardware topology.

**Exporting a hardware topology or topology part** You can export the data of the complete hardware topology or of a selected system or rack and its subelements to a specific XML file format (HTFX) for reuse in other ConfigurationDesk applications.

For instructions, refer to:

- [How to Export Data of a Specific Application Component](#) on page 119
- [How to Export a Hardware Topology Part](#) on page 244

**Importing or merging a hardware topology file** You can import a hardware topology file to either create or replace a hardware topology or to merge hardware topologies.

For instructions, refer to:

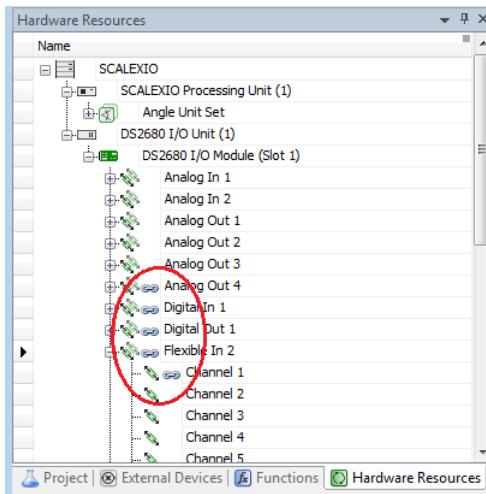
- [How to Import a Hardware Topology](#) on page 115
- [How to Merge Hardware Topologies](#) on page 245

**Scanning registered hardware** When you import a hardware topology, you can scan a registered hardware platform to automatically create a matching hardware topology. As a result, the state of the active ConfigurationDesk

application immediately switches to Matching platform connected. Refer to [Registering Hardware in ConfigurationDesk](#) on page 202.

## Assigning hardware resources

Hardware resources can be assigned to function blocks in ConfigurationDesk. This is important since each function needs one or more channels on the real-time hardware. Channels which are used in the application, i.e. they are assigned to function blocks, are indicated in the Hardware Resource Browser by the following symbol next to the channel name: .



For details on hardware resource assignment, refer to [Assigning Hardware Resources to Function Blocks](#) on page 399.

## Removing hardware resources

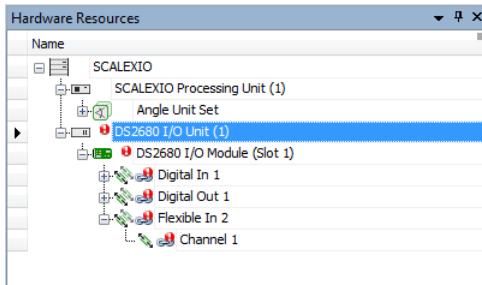
You can remove hardware resources from the hardware topology of your active application by right-clicking a board in the Hardware Resource Browser and selecting Delete from Topology from the context menu.

## Missing hardware resources

If a hardware resource assignment refers to a board that is not part of the hardware topology, it is marked as missing (= unresolved) in the hardware topology. This can result from the following actions:

- You have replaced the active hardware topology by importing a new one or by scanning hardware. The new hardware topology does not contain the board.
- You have deleted the board from the hardware topology of your application.

In both cases, ConfigurationDesk considers the board and all its channels affected by hardware resource assignment to be missing hardware resources. Missing hardware resources are displayed with a warning symbol in the Hardware Resource Browser like this:



If you add the missing hardware resources to the hardware topology again, the warning symbol disappears and the original hardware resource assignment is restored.

**Effects on the build process** Missing hardware resources do not abort the build process. Functions which are affected by missing hardware resources are simulated during the build process, i. e., the initial value specified at the corresponding function port is used. For details, refer to [Details on the Build Process](#) on page 716.

**Making signals of missing hardware resources available** You can move the hardware resource assignment of missing hardware resources via drag & drop to other hardware resources. Refer to [How to Reuse the Hardware Resource Assignment of Missing Hardware](#) on page 415.

#### Changing the slot assignment in I/O slot units

The slot assignment for SCALEXIO boards that you have added to an I/O slot unit can be changed. This can be useful to prepare your hardware topology for a specific slot or pin configuration in your rack. For instructions, refer to [How to Assign Boards to Specific Slots in an I/O Slot Unit](#) on page 248.

#### Configuring internal load settings

If you have added a board with signal measurement channels (such as the DS2601) to your hardware topology, you can configure the Load description and Load rejection enforcement settings for its channels via Properties Browser. These settings are checked during your configuration work and the build process. However, they are not transferred to a physical hardware system. They are overwritten when the hardware topology of your application is replaced. Refer to [Details on Handling Internal Loads](#) on page 392.

#### Related topics

##### Basics

Assigning Hardware Resources to Function Blocks.....	399
Handling Registered Hardware.....	202
Managing Components of ConfigurationDesk Applications.....	105

## How to Create Hardware Topologies from Scratch

### Objective

ConfigurationDesk enables you to create hardware topologies from scratch according to your needs. You can generate these hardware topologies without importing an existing HTFX file or scanning hardware.

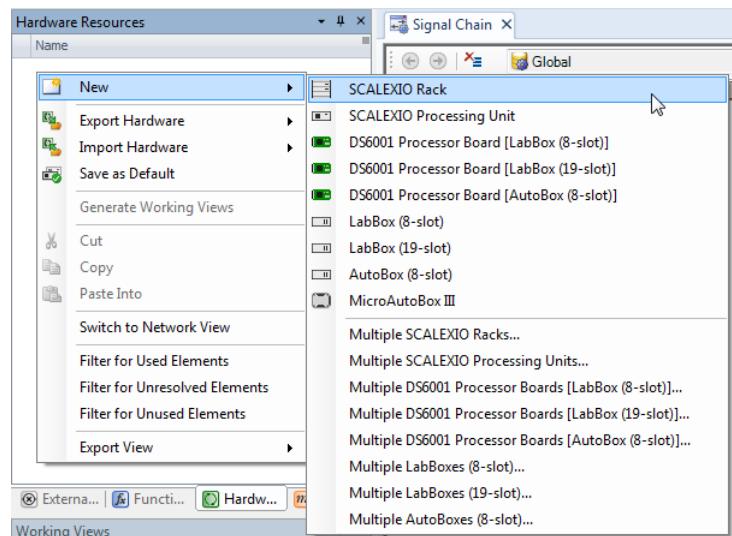
### Preconditions

An application is activated.

### Method

#### To create a hardware topology from scratch

- 1 Right-click in the Hardware Resource Browser and select New from the context menu.
- 2 From the submenu, select, for example, SCALEXIO Rack.



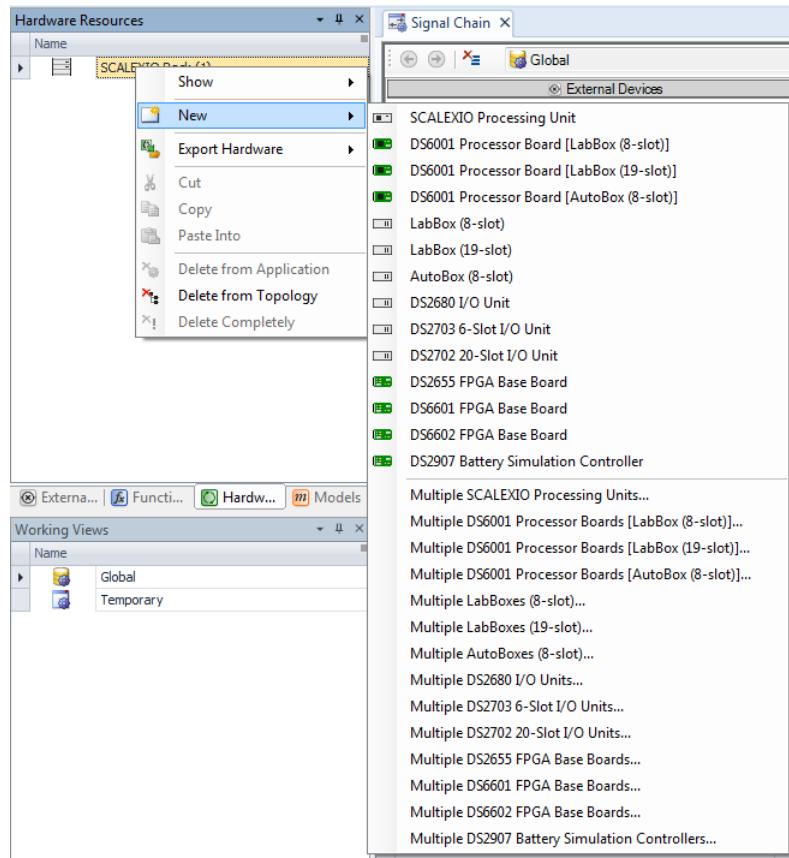
The SCALEXIO rack node is added to the topology and displayed in the Hardware Resource Browser.

#### Tip

You can change the name of the added element after adding it.

- 3 Right-click the SCALEXIO Rack node and select New from the context menu.

- 4 From the submenu, select the hardware resource you want to add to the hardware topology.



The selected hardware resource is added to your hardware topology.

- 5 Repeat steps 3 on page 241 and 4 on page 242 to add more hardware resources to your hardware topology.

## Result

You have created a hardware topology from scratch consisting of several hardware resources. The Hardware Resource Browser displays it in a hierarchical structure.

## Related topics

### HowTos

How to Activate a ConfigurationDesk Application.....	101
How to Add a ConfigurationDesk Application to a Project.....	100

# How to Extend Existing Hardware Topologies

## Objective

You can adapt existing hardware topologies to your requirements, for example, by adding new hardware resources to them.

## Preconditions

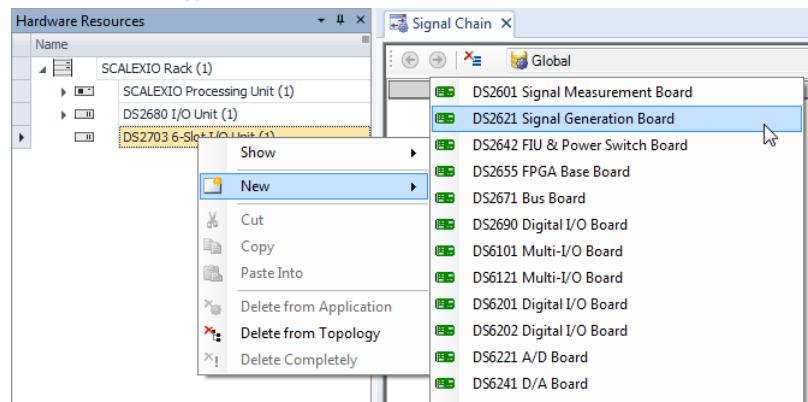
An application is activated and contains a hardware topology with components.

## Method

### To extend an existing hardware topology

- 1 In the Hardware Resource Browser, right-click a unit/box node and select New from the context menu.

A submenu opens, displaying hardware resources which suit the active hardware topology.



- 2 From the submenu, select the hardware resource you want to add to the hardware topology.
- 3 Repeat steps 1 and 2 to add more hardware resources to the hardware topology.

## Result

The selected hardware resources are added to the active hardware topology. They are inserted according to their slot numbers and displayed in the Hardware Resource Browser accordingly.

## Related topics

### HowTos

How to Activate a ConfigurationDesk Application.....	101
How to Import a Hardware Topology.....	115

## How to Export a Hardware Topology Part

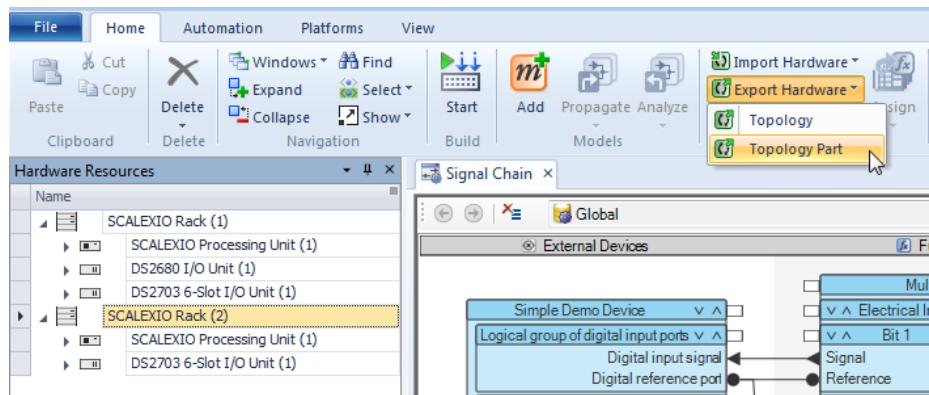
### Objective

ConfigurationDesk lets you export a selected system or rack and all its subelements from the hardware topology to an HTFX file for reuse in other ConfigurationDesk applications.

### Method

#### To export a hardware topology part

- 1 In the Hardware Resource Browser, select the system or rack you want to export.
- 2 On the Home – Hardware ribbon, select Export Hardware – Topology Part.



ConfigurationDesk opens the Export Hardware Topology Part dialog to export the data to an HTFX file. HTFX is the ConfigurationDesk-specific XML file format for hardware topologies.

- 3 Specify a file name and location and click Save.

All hardware topology elements from the currently selected rack or system in the Hardware Resource Browser are exported to the HTFX file.

### Result

You exported a hardware topology part.

### Next steps

You can import the HTFX file to a different ConfigurationDesk application to create or replace a hardware topology or to merge hardware topologies. For instructions, refer to:

- [How to Import a Hardware Topology](#) on page 115
- [How to Merge Hardware Topologies](#) on page 245

### Related topics

#### Basics

[Basics on Hardware Topologies.....](#) 236

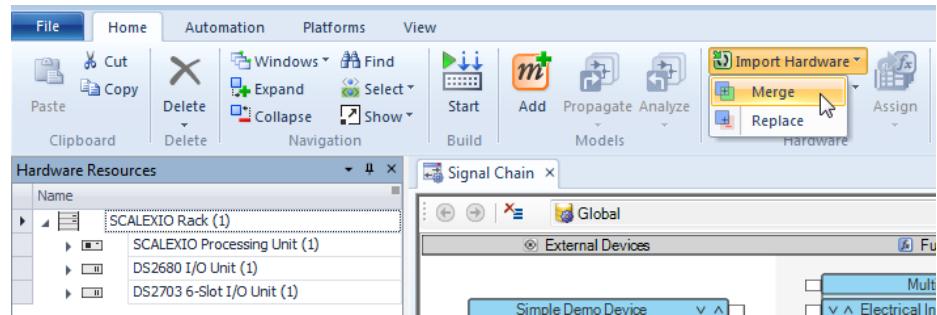
## How to Merge Hardware Topologies

**Objective** To combine the elements and settings of hardware topologies, you can merge them.

**Precondition** A hardware topology stored in an HTFX file must be available.

**Method** **To merge hardware topologies**

- 1 On the Home – Hardware ribbon, select Import Hardware – Merge. The Hardware ribbon group is available only if the Hardware Resource Browser is open in the active view set.



ConfigurationDesk opens an Import Hardware Topology dialog for you to select an HTFX file.

- 2 Select an HTFX file and click Open. Added hardware topology elements are displayed in the Hardware Resource Browser. The settings of imported elements that already exist in the ConfigurationDesk application are updated.

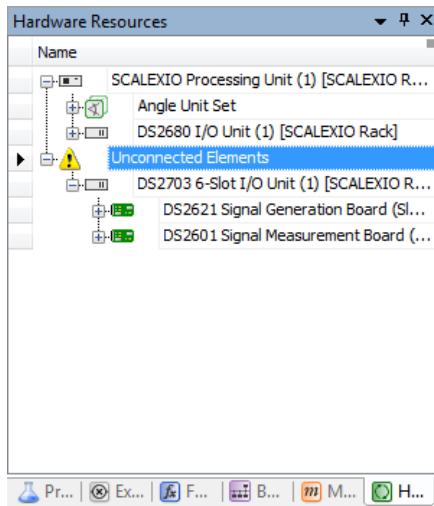
**Result** You merged hardware topologies.

**Related topics** Basics

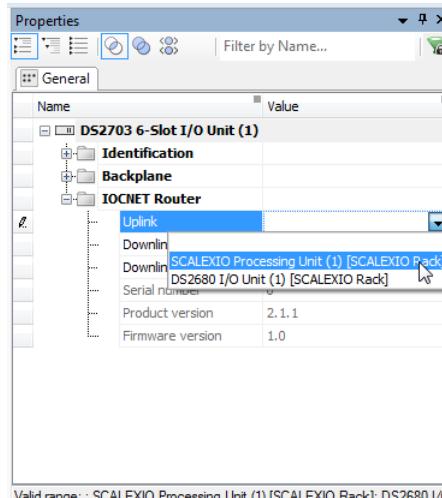
Basics on Hardware Topologies..... 236

## How to Establish a Network Connection in the Hardware Topology (SCALEXIO)

<b>Objective</b>	If you create a hardware topology from scratch, you might have to change or define the network connections between the units/boxes and the processing unit(s) in your SCALEXIO system.
<b>Effects when using registered hardware</b>	<p>If you scan registered hardware and use this hardware in your hardware topology, the network connections are adopted from the assembly of the connected hardware</p> <p>If you change network connections in ConfigurationDesk, you must change these connections also in the assembly of the hardware that is connected to ConfigurationDesk to avoid a mismatch.</p>
<b>Uplink and downlink configuration</b>	<p>The respective hardware components provide Uplink and Downlink properties to establish a network connection. You can access these properties via the Properties Browser.</p> <p>If the hardware topology contains units/boxes that are not connected to a processing unit, these units/boxes are identified and displayed below the Unconnected Elements node in the network view. You can establish the required connections via the Uplink property of the affected units/boxes or via the Downlink property of the respective parent units/boxes.</p>
<b>Precondition</b>	An application is activated and contains a hardware topology with at least one processing unit and one unit/box (DS2680, slot I/O unit, or LabBox).
<b>Method</b>	<p><b>To establish a network connection in the hardware topology</b></p> <p>1 Right-click the Hardware Resource Browser and select Switch to Network View from the context menu.</p> <p>A network-based view (logical network) of the hardware topology is displayed, with the SCALEXIO processing units at the top.</p> <p>If the hardware topology contains units/boxes that are not connected to a processing unit, these units/boxes are identified and displayed below the Unconnected Elements node as shown in the following illustration:</p>



- 2 From the hardware topology select an unit/box that is not connected to a processing unit.
- 3 In the Properties Browser, select a processing unit from the list of the Uplink property.



#### Tip

- You can also select the processing unit in the hardware topology, and select the desired I/O unit from the Downlink property in the Properties Browser.
- You can also drag units/boxes from the Unconnected Elements node to the desired processing unit in the Hardware Resource Browser to establish connections.

- 4 Repeat step 2 on page 247 and 3 on page 247 to establish all the required network connections in your hardware topology.

**Result**

You established at least one network connection between a unit/box and a processing unit in your hardware topology.

## How to Assign Boards to Specific Slots in an I/O Slot Unit

**Objective**

In the Hardware Resource Browser, you can assign boards in an I/O slot unit to specific slot(s). This can be useful to prepare your hardware topology for a specific slot or pin configuration in your rack.

Each board has to be assigned to one, two or three slot(s) depending on its type.

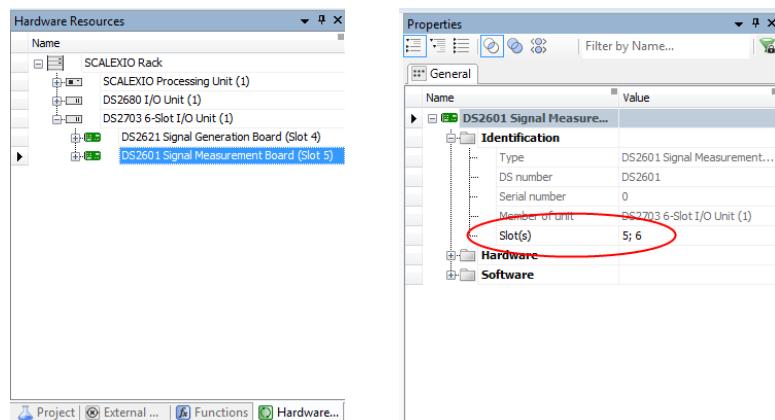
**Restrictions****Note**

When installing SCALEXIO boards, for example a DS2621 Signal Generation Board, in I/O slot units, you have to comply with the slot dependencies and recommendations described in:

- [Slot Dependencies for SCALEXIO HighFlex I/O Boards or MultiCompact I/O Boards \(SCALEXIO Hardware Installation and Configuration\)](#)
- [Slot Dependencies for SCALEXIO HighFlex I/O Boards or MultiCompact I/O Boards \(SCALEXIO Hardware Installation and Configuration\)](#)

**Initial slot assignment**

ConfigurationDesk automatically assigns newly added boards to slots according to the dependencies and recommendations mentioned above. The slot assignment is displayed in the Properties Browser and indicated in parentheses in the Hardware Resource Browser.



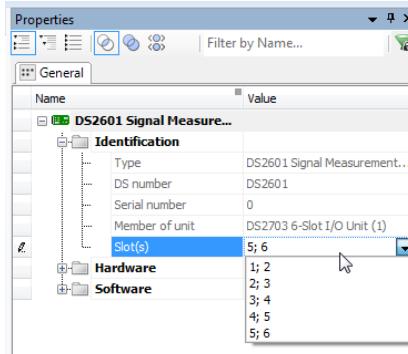
Note that even if the board is assigned to two or three slots, only the first slot is shown in parentheses in the Hardware Resource Browser.

**Precondition**

Your ConfigurationDesk application must contain a hardware topology with I/O slot units.

**Method****To assign a board to slots in an I/O slot unit**

- 1 In the Hardware Resource Browser, select a board that has been added to an I/O slot unit.
- 2 In the Properties Browser, select the Slot(s) property.
- 3 Select the slot(s) you want to use for the board from the list.

**Note**

You cannot assign two boards of the same type to the same slot(s). Boards of different types can be assigned to the same slot(s). This allows you to temporarily assign a board to an occupied slot in order to switch slot positions. Conflicts caused by assigning occupied slots are displayed in the Conflicts Viewer.

**Result**

You have assigned a board to specific slots in an I/O slot unit. If you change the slot configuration for hardware that is currently connected to a registered platform, the application state will change to No matching platform connected.

**Related topics****Basics**

<a href="#">Basics on Hardware Topologies.....</a>	236
<a href="#">DS2702 20-Slot I/O Unit (ConfigurationDesk Hardware Resource Properties)</a>	

**References**

<a href="#">DS2703 6-Slot I/O Unit (ConfigurationDesk Hardware Resource Properties)</a>	
---	--



# Specifying the External Device Interface

---

## Objective

External devices are all the devices which are connected to the dSPACE hardware, for example ECUs or external loads. In ConfigurationDesk, you can draw mapping lines to specify and visualize which I/O functionality is used for which signal of the external device. To do so, you need a representation of the external device interface in ConfigurationDesk.

The external device topology is the basis for using external devices in the signal chain of a ConfigurationDesk application.

---

## Where to go from here

## Information in this section

Basics on External Devices.....	252
Creating and Extending Device Topologies.....	260
Configuring External Devices.....	272
Adding Device Topology Elements to the Signal Chain.....	287

## Basics on External Devices

<b>Objective</b>	The external device topology and device blocks both represent external device interfaces.
	Device blocks are used in the signal chains of ConfigurationDesk applications and are based on device topologies. Each device block is created from a device topology and inherits its structure and configuration.

Where to go from here	Information in this section
	<a href="#">Basics on Device Topologies.....</a> 252
	<a href="#">Basics on Device Pin Assignment.....</a> 255
	<a href="#">Basics on Device Blocks.....</a> 257

## Basics on Device Topologies

<b>Objective</b>	You can create basic characteristics of external device interfaces (ECUs, external loads, etc.) in a device topology.
<b>Device topologies</b>	A device topology is the basis for device blocks in the signal chain. You can access a device topology via the External Device Browser.  To define basic interface characteristics of the external devices, you must configure the device topology elements. You can distinguish between the requirements for the device (e.g. ECU) you want to test and requirements for external loads by specifying the device type. For details on device configuration and device type specification, refer to <a href="#">Basics on Configuring External Devices</a> on page 272.
<b>Using the Signal Chain view set</b>	ConfigurationDesk's Signal Chain view set is best suited for performing tasks regarding the external device. The Signal Chain view set offers the External Device Browser to handle the device topology and the Signal Chain Browser to handle device blocks. You can also use the Windows command on the Home – Navigation ribbon to access additional tables for handling the external device and its pins and connectors.

## Creating, replacing, and merging device topologies

In ConfigurationDesk, you can create a device topology from scratch or import a device topology file. When you import a device topology, you can replace existing topologies or merge device topologies to combine their external device information.

**Creating a device topology from scratch** You can create and extend device topologies step by step via the External Device Browser. For instructions, refer to [How to Create and Extend Device Topologies via External Device Browser](#) on page 260.

**Importing a device topology** You can import an existing device topology from a device topology file (DTFX or XLSX). For instructions, refer to [How to Import a Device Topology](#) on page 110.

**Merging device topologies** You can merge the device topology of the currently active ConfigurationDesk application with another device topology from a device topology file (DTFX or XLSX). This is useful, for example, to create variants of ECUs (for example, with different configuration settings) in one device topology. Each of the variants can then be used in a specific application. For instructions, refer to [How to Merge Device Topologies](#) on page 265.

## Importing and exporting device topologies

You can export and import device topologies to/from DTFX files to exchange them between different ConfigurationDesk applications or projects.

You can also export and import device topologies to/from Microsoft Excel™ sheets (XLSX files), which you can edit independently of ConfigurationDesk.

**Exporting a device topology to XLSX** To edit or create a device topology outside of ConfigurationDesk, you can export the device topology to an Microsoft Excel™ sheet. For instructions, refer to [How to Export a Device Topology to a Microsoft Excel Sheet](#) on page 267.

**Importing a device topology from XLSX** A device topology that you have edited or created in an exported Microsoft Excel™ sheet can then be imported to a ConfigurationDesk application. For details, refer to [Importing a Device Topology from a Microsoft Excel Sheet](#) on page 268.

### Note

An XLSX file for import must comply with certain editing rules. For these rules and general information on the structure of the XLSX file, refer to [Rules for Editing External Device Topologies](#) on page 268.

## Structure of device topologies (port group address)

Device ports represent the signals of an external device in ConfigurationDesk. You can organize device ports in port groups for clarity. For example, you can group them by functionality. You can also specify a port group as a subelement of another port group.

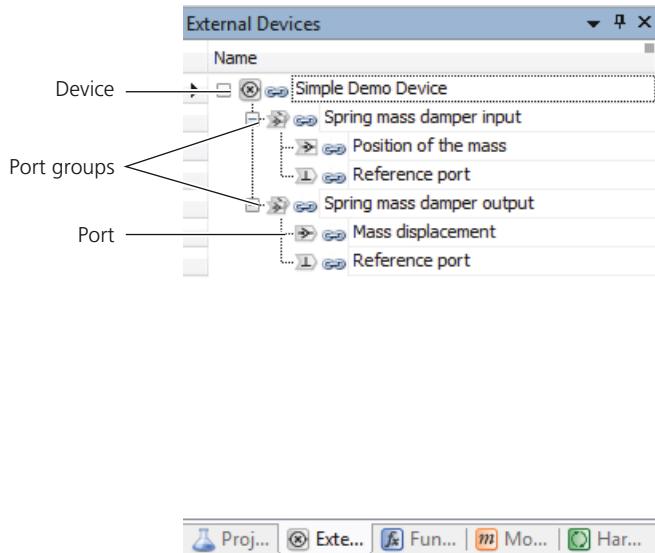
The names in each device topology element must be unique.

Each element can therefore be addressed by the following scheme (absolute address): <device name> \<port group name (layer 1)>\ ... \<port group name (layer n)> \<port name>.

For example: ECU\CommunicationPorts\CAN\_High stands for the CAN\_High port in the CommunicationPorts port group in the ECU device.

The address of an element is called a port group address. The device name is usually clear, so ConfigurationDesk displays and lets you specify port group addresses without the device name. These addresses are called relative port group addresses. Absolute addresses are used if the device name is not clear.

The illustration shows the hierarchical structure of a device topology with devices, port groups and ports in the External Device Browser.



#### Changing the structure of device topologies

In the External Device Browser, you can change the structure of device topologies via drag & drop. For instructions, refer to [How to Move Device Topology Elements](#) on page 265.

#### Using device topology elements

From the device topology, you drag the device topology elements you want to use in the application and drag them to the signal chain. When you drag an element to the signal chain, ConfigurationDesk creates a device block from it. For details on device blocks, refer to [Basics on Device Blocks](#) on page 257.

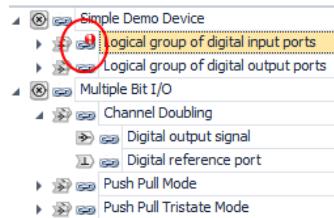
Device topology elements that are used in the signal chain are displayed with a symbol .

#### Note

Device topology elements that are not used in the signal chain (i.e., are not part of a device block) are ignored, for example, when the external cable harness is calculated.

## Deleting device topology elements

You can delete elements from the device topology via the Delete from Topology command in the External Device Browser. If they were used in the signal chain, they are still displayed and marked as unresolved.



## Accessing device topology properties

ConfigurationDesk makes it easy to access, view and configure device topology properties.

For details, refer to [Basics on Configuring External Devices](#) on page 272.

## Related topics

### Basics

Importing a Device Topology from a Microsoft Excel Sheet.....	268
Rules for Editing External Device Topologies.....	268

### HowTos

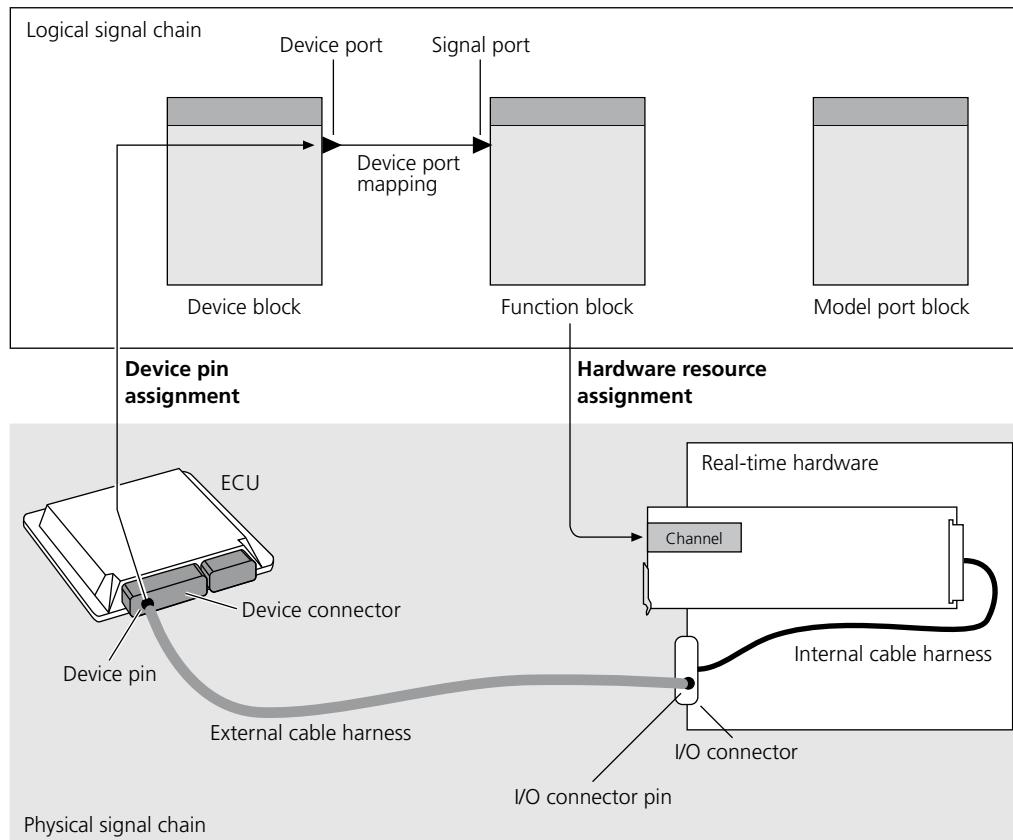
How to Copy and Paste Device Topology Elements.....	263
How to Create and Extend Device Topologies via External Device Browser.....	260
How to Export a Device Topology to a Microsoft Excel Sheet.....	267
How to Merge Device Topologies.....	265
How to Move Device Topology Elements.....	265

## Basics on Device Pin Assignment

### Device pin assignment

Device pin assignment means assigning device ports to device pins. ConfigurationDesk can use the device pin assignment together with the hardware resource assignment and the device port mapping to calculate external wiring information.

The role of the device pin assignment within the logical and physical signal chain is illustrated below:

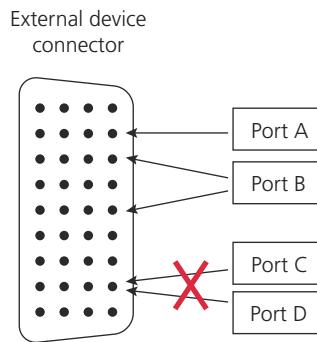


**Note**

You must provide the device pin assignment manually and check it yourself. It cannot be created and checked by ConfigurationDesk.

Device signals can be electrically shared by several pins, so one device port can be assigned to several device pins. However, more than one device port cannot be assigned to one external device pin (ECU or external load).

The illustration below shows the relation of device ports to pins in ConfigurationDesk.



#### Pin address

Device pins are organized and can be addressed like device ports. To identify each device pin clearly, you have to define a valid name which is unique within a device.

Each element can be addressed by the pin address with the following scheme (absolute address): <device name> \<connector name (layer 1)>\ ... \<connector name (layer n)> \<pin name>.

For example: ECU\Chamber A\CAN\_High stands for the CAN\_High pin in the Chamber A connector in the ECU device.

The device name is usually clear, so ConfigurationDesk usually displays and lets you specify pin addresses without the device name. These addresses are called relative pin addresses. Absolute addresses are used if the device name is not clear.

#### Related topics

##### Basics

Assigning Hardware Resources to Function Blocks.....	399
Calculating an External Cable Harness.....	683
Device Port Mapping.....	312

##### HowTos

How to Assign Device Ports to External Device Pins.....	278
How to Group Device Pins.....	283

## Basics on Device Blocks

#### Objective

Device blocks are used in the signal chain of an application to connect external devices with the I/O functionality of the dSPACE hardware. In ConfigurationDesk,

you can map the device ports to signal ports to connect external devices with the I/O functionality.

The basis for device blocks is the device topology.

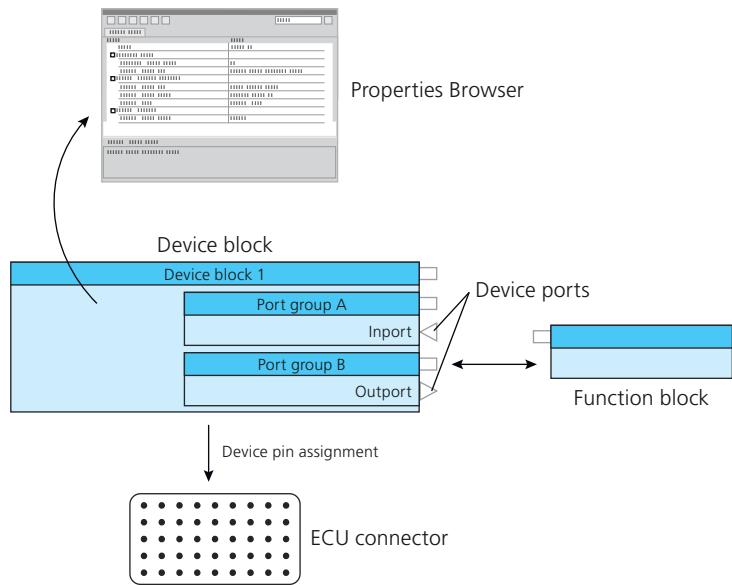
#### Purpose of device blocks

A device block provides:

- The interface to function blocks in the signal chain  
Device blocks contain device ports for mapping them to the signal ports of function blocks.
- The interface to external device pins (device pin assignment).
- Access to properties to configure the device, port groups, and ports.

#### Note

Device topology elements that are not used in the signal chain (i.e., are not part of a device block) are ignored, for example, when the external cable harness is calculated.



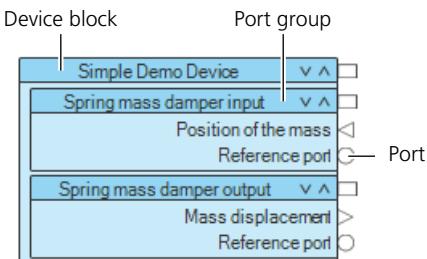
#### Creating device blocks

To create device blocks, you must add device topology elements (devices, port groups, and ports) from the device topology to the signal chain. For details, refer to [Basics on Device Block Handling](#) on page 287.

#### Structure and characteristics of device blocks

Device blocks are created from device topology elements, so they inherit their structure and characteristics. When you add a device topology element to the signal chain, ConfigurationDesk automatically groups all the elements of one device and adds essential elements.

The illustration below shows a device block in a working view. Device blocks contain port groups and ports. For details on the device topology structure, refer to [Basics on Device Topologies](#) on page 252.



#### Possible states of device block elements

Every device block, port group and port has a state, which is determined by the system. The state is displayed in the External Device Browser and the working views.

The following table shows the possible states of the elements and how they are displayed.

State	Display		Description
	Topology	Working View <sup>1)</sup>	
OK	The element is displayed without a warning symbol.	The frame/symbol of the element is black.	The element can be used without any restrictions.
Unresolved	The element is displayed with a warning symbol.	The frame/symbol of the element is highlighted orange. This indicates a warning conflict.	The element itself, or a part of it, is used in the signal chain, but is removed from the device topology.

<sup>1)</sup> To highlight conflicts, you have to set the appropriate highlighting option. For instructions, refer to [Highlighting Signal Chain Conflicts](#) on page 194.

# Creating and Extending Device Topologies

<b>Objective</b>	Device topologies are the representation of external device interfaces 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 them.
------------------	---

Where to go from here	Information in this section
	<p>How to Create and Extend Device Topologies via External Device Browser..... 260</p> <p>How to Copy and Paste Device Topology Elements..... 263</p> <p>How to Move Device Topology Elements..... 265</p> <p>How to Merge Device Topologies..... 265</p> <p>How to Export a Device Topology to a Microsoft Excel Sheet..... 267</p> <p>Importing a Device Topology from a Microsoft Excel Sheet..... 268</p> <p>Rules for Editing External Device Topologies..... 268</p>

## How to Create and Extend Device Topologies via External Device Browser

<b>Objective</b>	You can create and extend device topologies step by step via the External Device Browser.
<b>Basics</b>	For basic information on device topologies, refer to <a href="#">Basics on Device Topologies</a> on page 252.
<b>Preconditions</b>	<ul style="list-style-type: none"><li>▪ A ConfigurationDesk project with an active application is open.</li><li>▪ Note the basic information on configuring and naming external devices in <a href="#">Basics on Configuring External Devices</a> on page 272.</li></ul>

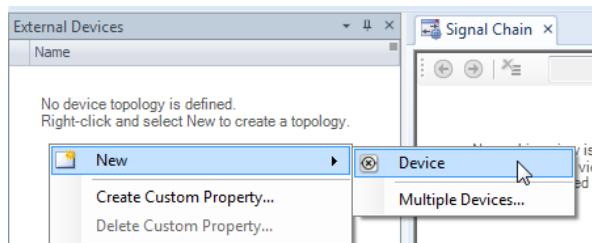
**Possible methods**

In the External Device Browser, you can create and extend a device topology in the following ways:

- You can create and extend it with single elements. For instructions, refer to [Method 1](#) on page 261.
- You can create and extend it with several elements simultaneously. For instructions, refer to [Method 2](#) on page 262.

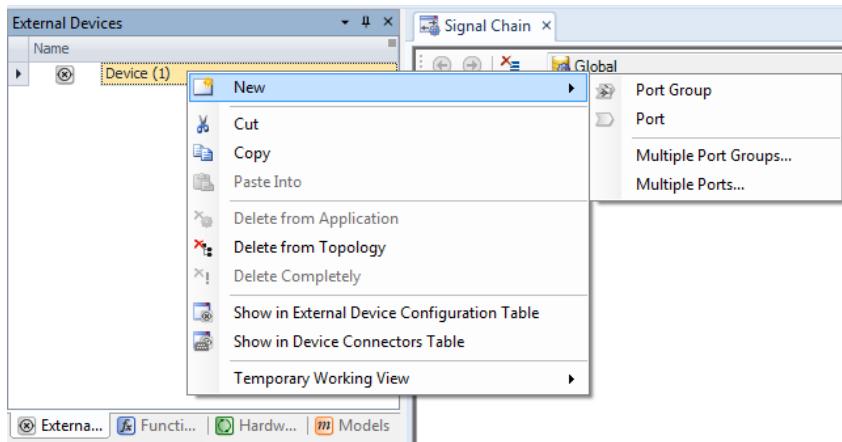
**Method 1****To create and extend a device topology with single elements**

- 1 In the External Device Browser, right-click in a free area.
- 2 From the context menu, select New – Device.



- 3 In the External Device Browser, right-click a device and select New – (Port, Port Group).

You can nest port groups in port groups if you want.



## Method 2

### To create and extend a device topology with several elements simultaneously

- 1 In the External Device Browser, right-click in a free area.
- 2 From the context menu, select New - Multiple Devices.

A Create Multiple Devices dialog opens.



- 3 In the Create Multiple Devices dialog, enter the Name pattern for the new devices and the number of devices you want to create. Device names must be unique. You can use the backslash \ to add an incremented number to the name of each new device. This avoids duplicate names.
- 4 In the External Device Browser, right-click a device and select New - (Multiple Ports, Multiple Port Groups). You can nest port groups in port groups, if you want.  
A Create Multiple Port Groups or Create Multiple Ports dialog opens.
- 5 In the Create Multiple Port Groups or Create Multiple Ports dialog, enter the Name pattern for the new port groups/ports and the number of port groups/ports you want to create.  
Click OK.

## Result

You created and extended a device topology with devices, port groups, and ports.

## Next step

Now you can configure the device topology elements. For instructions, refer to [Configuring External Devices](#) on page 272.

**Related topics****Basics**

Basics on Configuring External Devices.....	272
Basics on Device Topologies.....	252

## How to Copy and Paste Device Topology Elements

**Objective**

An easy way of creating new external device topology elements is to copy & paste existing ones.

**Preconditions**

- A ConfigurationDesk application containing a device topology with elements is active.
- The device topology elements must not have the unresolved state.  
For details on the state, refer to [Basics on Device Blocks](#) on page 257.

**Possible methods**

You can copy & paste device topology elements in the following ways:

- You can copy & paste elements once.  
For instructions, refer to [Method 1](#) on page 263.
- You can paste multiple copies of device topology elements simultaneously.  
For instructions, refer to [Method 2](#) on page 264.

**Method 1****To copy & paste device topology elements once**

- 1 In the External Device Browser, select the device topology element you want to copy.
- 2 On the Home ribbon, click Clipboard – Copy. You can also use the shortcut **Ctrl + C**.
- 3 Right-click the position where you want to paste the elements, and select Paste Before or Paste Into from the context menu.

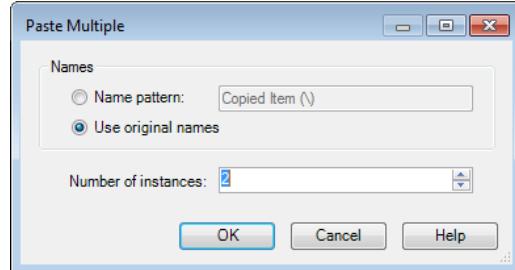
**Note**

Paste Before adds the copied elements before the selected element.  
Paste Into adds the elements to the selected element as subelements.  
The commands Paste Before and/or Paste Into are not displayed if selecting them results in invalid port group addresses. For example, if a port group is copied to the Clipboard and you open the context menu of a device port, the command Paste Into is not available.

**Method 2****To paste multiple copies of device topology elements simultaneously**

- 1 In the External Device Browser, select the device topology elements you want to copy.
- 2 On the Home ribbon, click Clipboard – Copy. You can also use the shortcut **Ctrl + C**.
- 3 Right-click the position where you want to paste the elements, and select Paste Multiple Before or Paste Multiple Into.

A Paste Multiple dialog opens.

**Note**

Paste Multiple Before adds the copied elements before the currently selected element. Paste Multiple Into adds the elements to the selected element as subelements.

The commands Paste Multiple Before and/or Paste Multiple Into are not available if selecting them results in invalid port group addresses. For example, if a port group is copied to the Clipboard and you open the context menu of a device port, the command Paste Multiple Into is not available.

- 4 In the dialog, enter the number of elements you want to paste. You can select Use original names to use the original name of the copied element(s) for all pasted instances. Note that this will cause duplicate names. The Name pattern lets you define the new name for each element. The backslash \ adds an incremented number to the name of each pasted element.
- 5 Click OK.

**Result**

You copied & pasted device topology elements.

The configuration of the element was also copied, except for the Pin(s) and Referenced port(s) properties. The Pin(s) property is only copied if you copy a complete device, because otherwise, more than one port would be assigned to one pin.

For details on properties of device topology elements, refer to [External Device Properties \(ConfigurationDesk User Interface Reference\)](#).

**Related topics****Basics**

<a href="#">Basics on Device Topologies.....</a>	252
--	-----

## How to Move Device Topology Elements

**Objective**

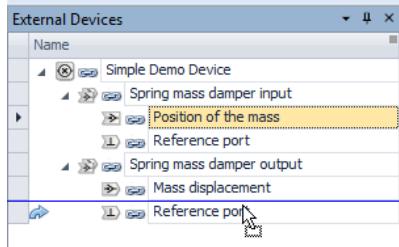
To change the structure of device topologies, you can move the device topology elements.

**Preconditions**

A ConfigurationDesk application containing device topology elements is active.

**Method****To move device topology elements**

- 1 In the External Device Browser, drag the device topology element to the desired position.

**Result**

You moved a device topology element to the desired position. The structure of the device topology has changed.

**Related topics****Basics**

<a href="#">Basics on Device Topologies.....</a>	252
--	-----

## How to Merge Device Topologies

**Objective**

To combine the elements and settings of device topologies, you can merge them.

**Precondition**

A device topology stored in a DTFX or XLSX file must be available.

**Note**

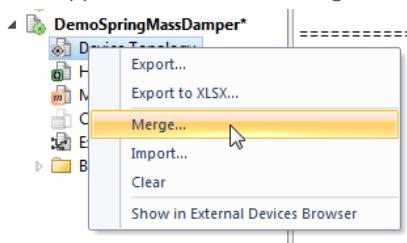
An external device topology stored in an XLSX file must comply with editing rules. Rows that do not comply with the editing rules are not merged. ConfigurationDesk does not check whether all the rows of a table are imported.

For more information, refer to [Rules for Editing External Device Topologies](#) on page 268.

**Method**

**To merge device topologies**

- 1 In the Project Manager, right-click the device topology of the currently active application and select Merge from the context menu.



A standard Open dialog opens.

- 2 Choose the file format you want to import:
  - You can add a DTFX file containing a device topology to the application.
  - You can add a Microsoft Excel™ file (XLSX) containing a device topology to the application.
- 3 Select the file containing the device topology information to be added and click Open.

The new device topology elements are displayed in the External Device Browser below the previously existing elements. Imported elements that already existed in the application are updated.

**Result**

You merged the device topology of the application with other device topology files.

**Related topics****Basics**

[Basics on Device Topologies.....](#) 252

**HowTos**

[How to Export a Device Topology to a Microsoft Excel Sheet.....](#) 267

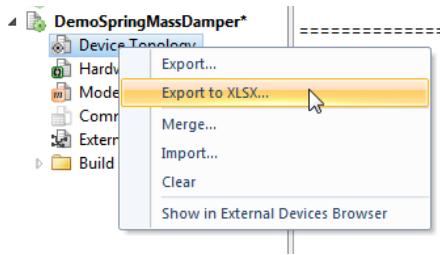
## How to Export a Device Topology to a Microsoft Excel Sheet

**Objective**

Exporting a device topology to a Microsoft Excel Sheet enables you to edit it with Microsoft Excel™.

**Method****To export a device topology to a Microsoft Excel Sheet**

- 1 In the Project Manager, right-click the Device Topology and select Export to XSLX from the context menu.



An Export Device Topology dialog with preset file format opens.

- 2 In the Export Device Topology dialog, choose the folder you want to export the device topology to.
- 3 Enter the file name, with or without the file name extension.
- 4 Click Save.

**Result**

You exported a device topology to a Microsoft Excel™ sheet (XSLX file).

For details on the structure and content of the exported file, refer to [Rules for Editing External Device Topologies](#) on page 268.

**Related topics****Basics**

[Basics on Device Topologies.....](#) 252

[Rules for Editing External Device Topologies.....](#) 268

## Importing a Device Topology from a Microsoft Excel Sheet

**Objective** To add device topologies which you edited in an Microsoft Excel™ sheet (XLSX file) outside of ConfigurationDesk to the application, you can import them.

**Methods** There are two ways to import a device topology stored in a Microsoft Excel™ file (XLSX file) to an application.

- You can import the device topology, replacingexisting topologies with the one from the Microsoft Excel file. For instructions, refer to [How to Import a Device Topology](#) on page 110.
- You can merge a device topology (stored in the Microsoft Excel file) with the device topology of the application. For instructions on merging device topology files, refer to [How to Merge Device Topologies](#) on page 265.

### Note

You must have an external device topology (stored in the Microsoft Excel file) which complies with editing rules. Rows which do not comply with the editing rules are not imported. ConfigurationDesk does not check whether all the rows of a table are imported.

For details, refer to [Rules for Editing External Device Topologies](#) on page 268.

### Related topics

#### Basics

[Rules for Editing External Device Topologies](#)..... 268

#### HowTos

[How to Export a Device Topology to a Microsoft Excel Sheet](#)..... 267

[How to Import a Device Topology](#)..... 110

[How to Merge Device Topologies](#)..... 265

## Rules for Editing External Device Topologies

**Objective** You have to ensure compliance with editing rules yourself if you edit an external device topology outside of ConfigurationDesk.

**Benefit**

You can edit a device topology outside of ConfigurationDesk in Microsoft Excel™. This has the following benefits:

- You can use sources of other tools and documents you are working with.
- You can edit a device topology with a lot of signals.
- You can define Excel™ macros to edit your configuration easily.

**Tip**

If you want to create a new device topology in Microsoft Excel™, it is helpful to export an empty device topology first. Then you have the required table layout already implemented. For instructions, refer to [How to Export a Device Topology to a Microsoft Excel Sheet](#) on page 267.

**Example**

The following illustration shows an example of an exported device topology:

1	A	D	F	U
2	Device Block	Device Port Group	Device Port	
3	Name	Name	Name	
3	Simple Demo Device	Logical group of digital input port: Digital input signal		
4	Simple Demo Device	Logical group of digital input port: Digital reference port		
5	Simple Demo Device	Logical group of mixed output port: Analog output signal		
6	Simple Demo Device	Logical group of mixed output port: Analog reference port		
7	Simple Demo Device	Logical group of mixed output port: Analog output signal 2		
8	Simple Demo Device	Logical group of mixed output port: Analog reference port 2		
9	Simple Demo Device	Logical group of mixed output port: Digital output signal		
10	Simple Demo Device	Logical group of mixed output port: Digital reference port		
11	Combined IO Device	3 phase DMM input	DMM phase 1	

You can expand further columns containing the properties for device blocks, port groups, or device ports.

D	F	G	H	I	
Group	Device Port	Description	Port Type	Physical Attributes	Electri
of digital input ports: Digital input signal	Consumes a digital PWM signal	In	Voltage		
of digital input ports: Digital reference port	Reference port for pulse generati	Reference	Voltage		
of mixed output port: Analog output signal	Generates a digital signal	Out	Voltage		
of mixed output port: Analog reference port		Reference	Voltage		
of mixed output port: Analog output signal 2	Generates a pulsed signal	Out	Voltage		
of mixed output port: Analog reference port 2		Reference	Voltage		
of mixed output port: Digital output signal	Generates a PWM signal	Out	Voltage		
of mixed output port: Digital reference port		Reference	Voltage		
input	DMM phase 1	In	Voltage		

**Layout of device topology tables in Excel™**

- Each device topology property has its own column. The name of the property is the column header in the second row.
- The properties are grouped by device, port group, and port in the first row.
- Each device port has its own row. Rows for empty devices or device port groups can also be included.

**Note**

You can add custom properties to the device topology after it has been imported. For instructions, refer to [How to Add Custom Device Properties](#) on page 281.

The custom properties are available for editing in Excel™ if you export the device topology again afterwards.

**Rules for editing**

The following table shows the editing rules for the Microsoft Excel™ file. False or empty property settings are possible and will be replaced by default values or left empty after import. At least the – Name columns are required for a valid file.

For details on device topology properties, refer to [External Device Properties](#) ([ConfigurationDesk User Interface Reference](#) ).

Column Title (Device Topology Property)	Editing Rules
Device Block – Name	The semicolon ; and backslash \ are not allowed.
Device Block – Description	None
Device Block – Device Type	Allowed settings <ul style="list-style-type: none"> <li>▪ ECU</li> <li>▪ Load</li> </ul> If left empty, ConfigurationDesk sets the property to ECU.
Device Port Group – Name	The semicolon ; and the backslash \ are not allowed. You can also enter hierarchical port groups via relative port group address <sup>1)</sup> .
Device Port Group – Description	None
Device Port – Name	The semicolon ; and backslash \ are not allowed.
Device Port – Description	None
Device Port – Port Type	Allowed settings <ul style="list-style-type: none"> <li>▪ In</li> <li>▪ Out</li> <li>▪ Bidirectional</li> <li>▪ Reference</li> <li>▪ Unspecified</li> </ul> If left empty, ConfigurationDesk sets the property to Unspecified.
Device Port – Physical Attributes	Allowed settings <ul style="list-style-type: none"> <li>▪ Current</li> <li>▪ Voltage</li> <li>▪ Resistance</li> <li>▪ Unspecified</li> </ul> If left empty, ConfigurationDesk sets the property to Unspecified.
Device Port – Electrical Characteristics	None
Device Port – Load Description	None

Column Title <b>(Device Topology Property)</b>	Editing Rules
Device Port – Failure Simulation	Allowed settings <ul style="list-style-type: none"><li>▪ Disabled</li><li>▪ Enabled</li></ul> If left empty, ConfigurationDesk sets the property to Disabled.
Failure classes	Open Circuit
	Short to GND
	Short to VBAT
	Short to Signal Generation Channel
	Short to Signal Measurement Channel
	Short to Bus Channel
Device Port – Referenced port(s)	You can only assign reference ports which belong to the same device. If the port is assigned to several reference ports, the separator is the semicolon ;. Assign the ports that belong to the same signal by entering the relative port group address <sup>1)</sup> .
Device Port – Pin(s)	The semicolon ; is not allowed for pin names. If the port is assigned to several device pins, the separator for external device pins is the semicolon ;. Enter the pin names by their relative pin address. <sup>2)</sup>
Device Port – Load Rejection <sup>3)</sup>	Allowed settings <ul style="list-style-type: none"><li>▪ Enforced</li><li>▪ Not enforced</li></ul> Only available for devices with the Load device type. If the device type is ECU, entries will be ignored.

<sup>1)</sup> A relative port group address is a port group address without the device name. For a port group address scheme, refer to [Basics on Device Topologies](#) on page 252

<sup>2)</sup> A relative pin address is a pin address without the device name. For a pin address scheme, refer to [Basics on Device Pin Assignment](#) on page 255

<sup>3)</sup> Only available if the device topology contains a device of the type Load

## Related topics

### Basics

[Importing a Device Topology from a Microsoft Excel Sheet](#)..... 268

### HowTos

[How to Export a Device Topology to a Microsoft Excel Sheet](#)..... 267

### References

[External Device Properties \(ConfigurationDesk User Interface Reference\)](#)

# Configuring External Devices

<b>Objective</b>	To specify the electrical and general characteristics of the external device interface, you have to configure the device topology elements.
------------------	---

<b>Where to go from here</b>	<b>Information in this section</b>														
	<table> <tr> <td>Basics on Configuring External Devices.....</td> <td>272</td> </tr> <tr> <td>How to Rename Device Topology Elements.....</td> <td>276</td> </tr> <tr> <td>How to Assign Reference Ports.....</td> <td>276</td> </tr> <tr> <td>How to Assign Device Ports to External Device Pins.....</td> <td>278</td> </tr> <tr> <td>How to Add Custom Device Properties.....</td> <td>281</td> </tr> <tr> <td>How to Delete Custom Properties.....</td> <td>282</td> </tr> <tr> <td>How to Group Device Pins.....</td> <td>283</td> </tr> </table>	Basics on Configuring External Devices.....	272	How to Rename Device Topology Elements.....	276	How to Assign Reference Ports.....	276	How to Assign Device Ports to External Device Pins.....	278	How to Add Custom Device Properties.....	281	How to Delete Custom Properties.....	282	How to Group Device Pins.....	283
Basics on Configuring External Devices.....	272														
How to Rename Device Topology Elements.....	276														
How to Assign Reference Ports.....	276														
How to Assign Device Ports to External Device Pins.....	278														
How to Add Custom Device Properties.....	281														
How to Delete Custom Properties.....	282														
How to Group Device Pins.....	283														

## Basics on Configuring External Devices

<b>Objective</b>	You can configure the electrical and general characteristics of your device in ConfigurationDesk. This enables ConfigurationDesk to support your configuration work after you have added device topology elements to the signal chain.
<b>Access to external device properties</b>	<p>ConfigurationDesk makes it easy to access, view, and configure external device properties.</p> <p><b>Properties Browser</b> The Properties Browser shows you all the properties of the device topology elements you selected. For details, refer to <a href="#">Configuring Elements with the Properties Browser</a> on page 131.</p> <p><b>External Device Configuration table</b> The External Device Configuration table shows you the most important properties of the device topology elements. You can easily sort the table entries by property or access the values of a specific property for several device topology elements at once. For details, refer to <a href="#">Using Tables to Access and Configure Elements</a> on page 151.</p>

## Effects of changing property values

Device block properties are closely linked to the device topology elements they are based on. When you change a property of a device topology element, ConfigurationDesk always changes the property of the corresponding device block in the signal chain. The same happens in the other direction if you change the property of a device block.

This has the following benefits:

- You can change the device block configuration by replacing the device topology.
- You can ensure consistency between device topology elements and device blocks in the signal chain.

## Specifying names

To identify each device topology element clearly, you have to define a valid name which is unique within device topology elements (e.g., a port group). For instructions, refer to [How to Rename Device Topology Elements](#) on page 276.

## Specifying device types (ECU or Load)

To distinguish between requirements for the external device you want to test and requirements for external loads, you have to configure the Type device property.

**ECU** This type is recommended for ECU devices you want to test.

**Load** This type is recommended for external loads. If a device has the Load device type, you can enforce Load rejection for its ports to protect your external loads during failure simulation. For details, refer to [Handling Loads](#) on page 385.

### Note

The ports of a device with the ECU device type do not offer the Load rejection property.

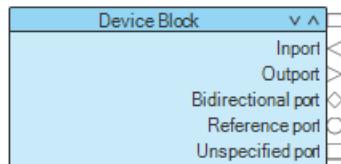
## Assigning device ports to device pins

Device pin assignment means assigning device ports to device pins. ConfigurationDesk can use the device pin assignment together with the hardware resource assignment and the device port mapping to calculate external wiring information. For details, refer to [Basics on Device Pin Assignment](#) on page 255.

## Specifying device port types

To specify the signal direction of device ports, you have to specify the device port type. The device port type is the main aid to mapping work.

The following illustration shows the different device port types and their symbolic representation:



**Device import** Represents a signal which is input from the real-time hardware to your external device.

**Device outport** Represents a signal which is output from your external device to the real-time hardware.

**Device bidirectional port** This port is independent of data direction or a current flow. Bidirectional ports are used, for example, to specify bus communication.

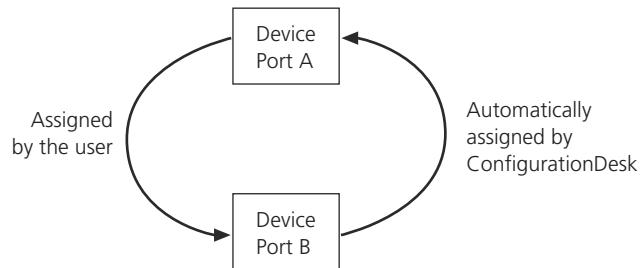
**Device reference port** Represents the reference potential of any other port. For example, when differential signals are used, this is the inverted signal. When single-ended signals are used, it is the ground signal (GND).

**Unspecified device port** The port characteristics are not specified, so ConfigurationDesk cannot help you to perform device port mapping.

#### Assigning device reference ports

By defining a reference between two device ports, you specify that the signal mapped to these ports requires the two potentials (available at the ports) for signal transmission.

ConfigurationDesk automatically references the ports in both directions as shown below.



For instructions, refer to [How to Assign Reference Ports](#) on page 276.

#### Specifying allowed failure classes

You can simulate failures in the wiring of your external device by using a failure simulation system. To control the failure simulation with your experiment software, you have to specify the allowed failure classes.

You have to specify the allowed failure classes for each device port separately. These failure classes can then be simulated via the experiment software. A failure class which could destroy the external device must not be allowed for the related signal. For details, refer to [Specifying Failure Simulation Settings](#) on page 374.

---

**Adding custom device port properties**

You can add custom device port properties to document specific properties of device ports.

For instructions, refer to [How to Add Custom Device Properties](#) on page 281.

---

**Transferring device block settings to function blocks**

You can transfer failure simulation and load settings from the device ports to mapped signal ports of function blocks.

Transferring settings has the advantage that you do not need to configure the settings of the function block manually.

For instructions, refer to [How to Transfer Port Settings from Device Ports to Signal Ports](#) on page 317.

---

**Managing device topology data**

You can manage different configuration settings by replacing device topologies. ConfigurationDesk automatically overwrites the configuration of the device topology elements used in the signal chain.

When you replace the device topology by importing a new one from a DTFX or Microsoft Excel™ file, the following conflicts can occur:

- The state of device block elements can become unresolved. This means ConfigurationDesk cannot find certain device topology elements with the same address in the new device topology. For example, the name of one port group in the new device topology has changed. For details on device block states, refer to [Basics on Device Blocks](#) on page 257.
- The device pin assignment is different to the original device topology and the pins are referenced by the external cable harness. ConfigurationDesk checks it and informs you if there is a conflict. For details on the device pin assignment, refer to [Basics on Device Pin Assignment](#) on page 255.
- If the device port mapping uses device ports that have a different configuration in the new device topology, the state of the mapping line might change to warning. For details on mapping states, refer to [Mapping Conflicts](#) on page 315.

---

**Related topics****Basics**

Configuring Elements with the Properties Browser.....	131
Using Tables to Access and Configure Elements.....	151

**References**

<a href="#">External Device Properties (ConfigurationDesk User Interface Reference)</a>
---

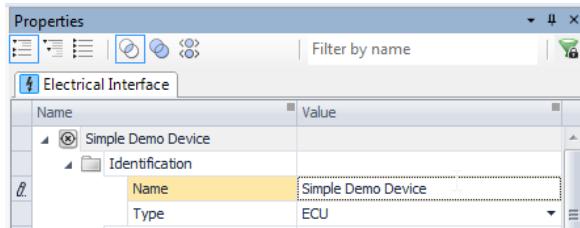
## How to Rename Device Topology Elements

---

<b>Objective</b>	ConfigurationDesk lets you change the name of device topology elements.
------------------	---

---

<b>Method</b>	<b>To rename device topology elements</b>
	<ol style="list-style-type: none"><li>1 Select the device topology element that you want to rename.</li><li>2 In the Properties Browser, click the Name edit field as shown in the following example.</li></ol>



### Tip

In most panes, you can also double-click a device topology element or select it and press **F2** to make the name editable.

- 3 Enter a new name and press **Enter** or click anywhere outside the edit field to confirm.

---

<b>Result</b>	You renamed device topology elements. If the port group address is not unique, for example, if two ports in the same port group have the same name, a conflict is generated and displayed in the Conflicts Viewer.
---------------	--

---

<b>Related topics</b>	Basics
	<a href="#">Basics on Configuring External Devices.....</a> 272

## How to Assign Reference Ports

---

<b>Objective</b>	You can assign the device ports that belong to the same signal via the Referenced port(s) property.
------------------	---

---

<b>Restriction</b>	You can only assign reference ports that belong to the same device.
--------------------	---

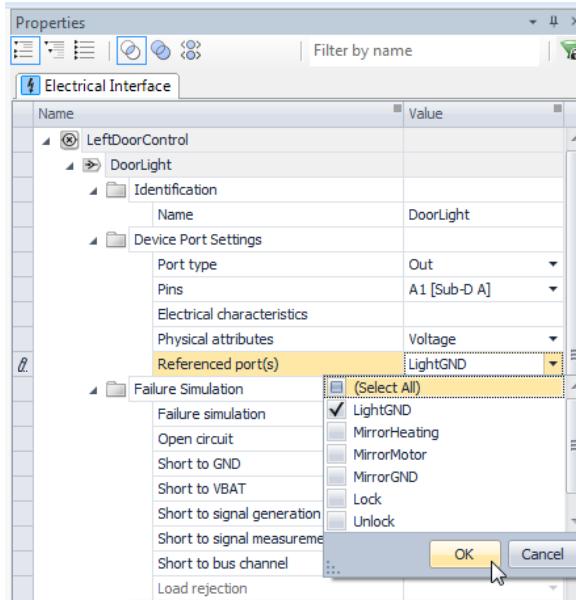
---

<b>Precondition</b>	An application containing device ports is active.
---------------------	---

---

<b>Method</b>	<b>To assign reference ports</b>
---------------	----------------------------------

- 1 In the External Device Browser, select the port that you want to configure.
- 2 In the Properties Browser, click the Referenced port(s) edit field.
- 3 Select the ports that you want to assign as reference ports.

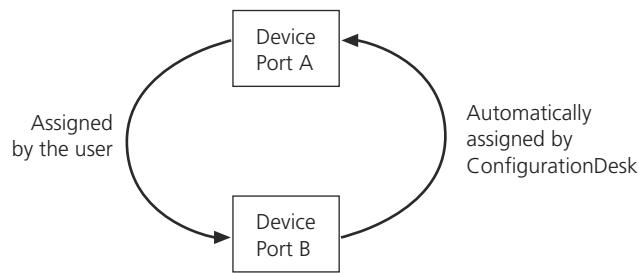


- 4 Click OK.

---

<b>Result</b>	You assigned reference ports to the selected port.
---------------	--

ConfigurationDesk automatically references the ports in both directions as shown below.




---

## Related topics

### Basics

Basics on Configuring External Devices.....	272
---	-----

## How to Assign Device Ports to External Device Pins

### Objective

You must assign the device ports to device pins of your external device connector via the device pin assignment. The device pin assignment is required to calculate the external wiring information.

#### Note

You must provide the device pin assignment manually and check it yourself. It cannot be created and checked by ConfigurationDesk.

#### NOTICE

**If the device pin assignment is incorrect, the wiring information that is calculated (for the external harness) will not match the electrical requirements of the external devices (for example your ECU).**

Risk of damage to the connected devices.

Ensure that the device pin assignment matches the pin configuration of the device.

### Allowed characters

The Pins device port property is string-based. Its values must comply with the following rules:

- All characters, including \* ? | < > : / " , are allowed, with one exception: The semicolon ; is not allowed.
- Number of characters: no limit.

### Precondition

A ConfigurationDesk application containing device ports is active.

### Possible methods

You can assign device ports to external device pins in two different ways:

- To assign device ports via Properties Browser, refer to [Method 1](#) on page 278.
- To assign device ports in the External Device Connectors table, refer to [Method 2](#) on page 280.

### Method 1

#### To assign device ports to external device pins via Properties Browser

- 1 Select the device port that you want to configure.
- 2 In the Properties Browser, click the Pins edit field.

- 3 Enter the names of the external device pins. If you enter more than one pin, use semicolons as separators.

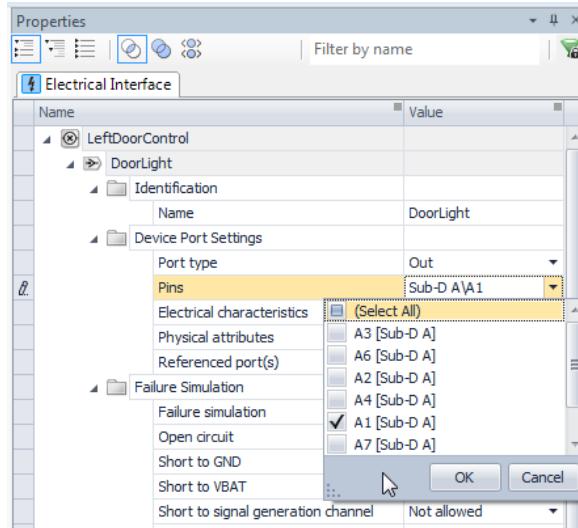
**Tip**

- You can group device pins by entering the relative device pin address.
- You can use the common shortcut keys **Ctrl + C**, **Ctrl + X**, **Ctrl + V**, and **Del** to enter the assigned pins.

- 4 Press **Enter** or click anywhere outside the edit field to confirm.

ConfigurationDesk checks whether the new name is valid. If this is not the case, ConfigurationDesk rejects the new name and a message appears.

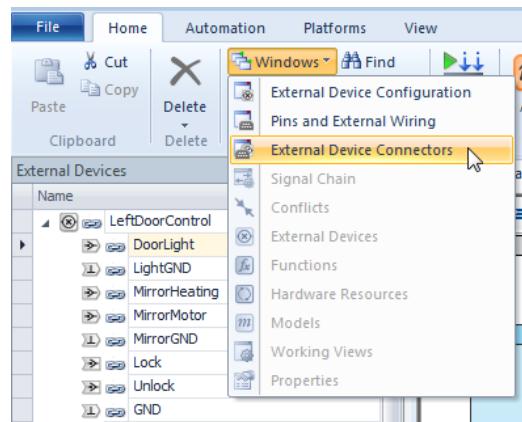
All pin values that you entered can be selected from a list next time you configure the property for a device port.



This is useful if you want to assign several pins to the same device port. Note that you cannot assign the same pin to several device ports. On assignment, the previously connected ports are disconnected from the reassigned pin.

**Method 2****To assign device ports to external device pins in the External Device Connectors table**

- 1 Switch to the Signal Chain view set if necessary.
- 2 On the Home ribbon, select Navigation – Windows – External Device Connectors.



The External Device Connectors table opens.

- 3 In the External Device Connectors table, create the required device connectors and pins. For instructions, refer to [How to Group Device Pins](#) on page 283.
- 4 In the Port column, select the port you want to assign for each device pin.

**Tip**

You can also view and edit the device pin assignment in the Pins and External Wiring table. However, you cannot create new pins or connectors there.

**Result**

You assigned device ports to external device pins.

**Related topics****Basics**

Basics on Device Pin Assignment.....	255
Calculating an External Cable Harness.....	683

**HowTos**

How to Group Device Pins.....	283
-------------------------------	-----

## How to Add Custom Device Properties

### Objective

You can add custom external device properties to document specific characteristics of device topology elements or to define sort keys for a table.

### Allowed characters

- All characters are allowed, including ? | < > : / \ " . \*
- Number of characters: no limit.

### Preconditions

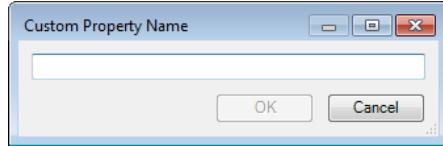
A ConfigurationDesk application which contains a device topology with elements is active.

### Method

#### To add custom device properties

- 1 In the External Device Browser, right-click an empty area to open a context menu.
- 2 In the context menu, click Create Custom Property.

A Custom Property Name dialog opens.



- 3 Enter a name for the property.

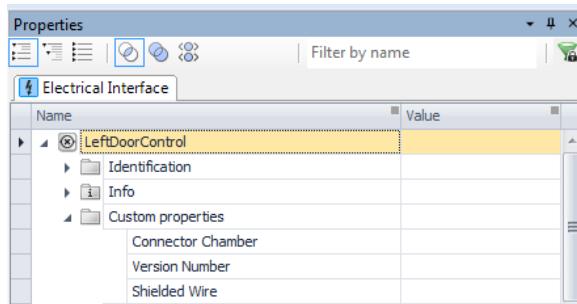
- 4 Click OK.

If the entered name is not unique, an error message appears.

### Result

You added custom properties for the elements of the device topology. Now you can enter string-based descriptions or sort keys in the edit fields of the added properties for device topology elements.

The illustration below shows an example. The Connector Chamber, Version Number and Shielded Wire custom properties have been added and are listed in the Properties Browser for the LeftDoorControl device.



**Related topics**

**Basics**

Basics on Configuring External Devices..... 272

**HowTos**

How to Delete Custom Properties..... 282

## How to Delete Custom Properties

---

**Objective**

You can delete a custom property if you no longer need it.

---

**Preconditions**

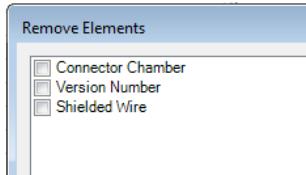
- A ConfigurationDesk application which contains a device topology with elements is active.
  - At least one custom property has been added to the device topology. For instructions, refer to [How to Add Custom Device Properties](#) on page 281.
- 

**Method**

**To delete a custom property**

- 1 In the External Device Browser, right-click an empty area to open a context menu.
- 2 In the context menu, click **Delete Custom Property**.

A Remove Elements dialog opens.



- 3 Select the custom properties you want to delete.
  - 4 Click **OK**.
- 

**Result**

You deleted custom properties.

**Related topics****Basics**

Basics on Configuring External Devices..... 272

**HowTos**

How to Add Custom Device Properties..... 281

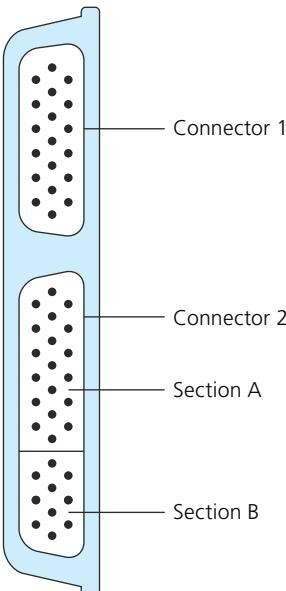
## How to Group Device Pins

**Objective**

If your ECU has several connectors or even connectors subdivided into sections, you can use device connector and device pin elements in the External Device Connectors table to group pins within connectors representing the structure of the real ECU connector. This gives you a better overview of device pins, for example, in the wiring information.

**Better identification of device pins**

The following illustrations show an example of an ECU with several connectors, the corresponding device connector structure in ConfigurationDesk's External Device Connectors table, and resulting wiring information after the external cable harness has been calculated and exported to an Microsoft Excel™ sheet.

ECU Example		Device Connector Structure in ConfigurationDesk							
<p>External device</p>  <p>Connector 1</p> <p>Connector 2</p> <p>Section A</p> <p>Section B</p>		<p>Signal Chain   External Device Connectors</p> <p>Name</p> <ul style="list-style-type: none"> <li>ECU             <ul style="list-style-type: none"> <li>Connector 1                     <ul style="list-style-type: none"> <li>Pin (1)</li> <li>Pin (2)</li> <li>Pin (3)</li> <li>Pin (4)</li> <li>Pin (5)</li> </ul> </li> <li>Connector 2                     <ul style="list-style-type: none"> <li>Section A                             <ul style="list-style-type: none"> <li>Pin (1)</li> <li>Pin (2)</li> <li>Pin (3)</li> <li>Pin (4)</li> </ul> </li> <li>Section B                             <ul style="list-style-type: none"> <li>Pin (1)</li> <li>Pin (2)</li> <li>Pin (3)</li> </ul> </li> </ul> </li> </ul> </li> </ul>							
Exported Wiring Information									
A	B	C	F	G	H	I	J		
External Cable	Pin Group	Device Block	Name	Device Connector	Device Pin	Rack	Unit Name	ProductName	Con
3 1	2	1 ECU	Connector 1	Pin (1)		SCALEXIO Rack (1)	DS2680 I/O Unit (1)	DS2680 I/O Unit	ECU
4 1	2	1 ECU	Connector 1	Pin (2)					
5 2	1	1 ECU	Connector 1	Pin (3)					
6 2	1	1 ECU			SCALEXIO Rack (1)	DS2680 I/O Unit (1)	DS2680 I/O Unit	ECU	
7 2	2				SCALEXIO Rack (1)	DS2680 I/O Unit (1)	DS2680 I/O Unit	ECU	
8 3	1	1 ECU	Connector 1	Pin (4)					
9 3	1	1 ECU	Connector 1	Pin (5)					
10 3	2				SCALEXIO Rack (1)	DS2680 I/O Unit (1)	DS2680 I/O Unit	ECU	
11 4	1	1 ECU	Connector 2\Section A	Pin (1)					
12 4	1	1 ECU	Connector 2\Section A	Pin (2)					
13 4	2				SCALEXIO Rack (1)	DS2680 I/O Unit (1)	DS2680 I/O Unit	ECU	
14 5	1	1 ECU	Connector 2\Section A	Pin (3)					
15 5	1	1 ECU	Connector 2\Section A	Pin (4)					
16 5	2				SCALEXIO Rack (1)	DS2680 I/O Unit (1)	DS2680 I/O Unit	ECU	
17 6	1	1 ECU	Connector 2\Section B	Pin (1)					
18 6	1	1 ECU	Connector 2\Section B	Pin (2)					
19 6	1	1 ECU	Connector 2\Section B	Pin (3)		SCALEXIO Rack (1)	DS2680 I/O Unit (1)	DS2680 I/O Unit	ECU
20 6	2				SCALEXIO Rack (1)	DS2680 I/O Unit (1)	DS2680 I/O Unit	ECU	
21 7	1				SCALEXIO Rack (1)	DS2680 I/O Unit (1)	DS2680 I/O Unit	ECU	

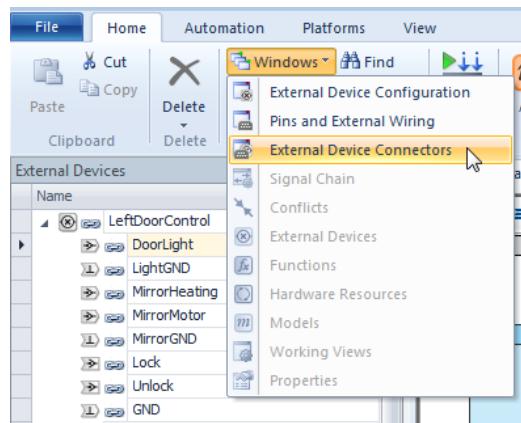
**Precondition**

A ConfigurationDesk application containing device ports is active.

**Method****To group device pins****Note**

The names of device connectors and device pins have to be unique in the devices and device connectors.

- 1 Switch to the Signal Chain view set if necessary.
- 2 On the Home ribbon, select Navigation – Windows – External Device Connectors.



The External Device Connectors table opens.

- 3 In the External Device Connectors table, right-click the device you want to specify the device connector for and select New – Connector. You can select New – Multiple Connectors to add several connectors at once.
- 4 If further subgrouping is required, you can right-click a device connector and add one or more connectors to it via New – Connector or New – Multiple Connectors.
- 5 Repeat steps 3 and 4 until the grouping represents the connector of your real external device (e.g., your ECU).
- 6 If you have already assigned device pins to device ports, the pins are available in the External Device Connectors table and can be moved to connectors via drag & drop. Otherwise, you can use the New – Pin and New – Multiple Pins commands in the context menu of device connectors to add device pins. Note that in this case you have to assign the new device pins to device ports afterwards, for example, in the Port column of the Device Connectors table view.

**Result**

You specified a device connector in ConfigurationDesk according to the structure of your real external device connector.

**Related topics**

**Basics**

Basics on Device Pin Assignment.....	255
Calculating an External Cable Harness.....	683

**HowTos**

How to Assign Device Ports to External Device Pins.....	278
---	-----

# Adding Device Topology Elements to the Signal Chain

<b>Objective</b>	To use device topology elements in the signal chain of your ConfigurationDesk application, you have to add device blocks to the signal chain.
------------------	---

Where to go from here	Information in this section
	<a href="#">Basics on Device Block Handling</a> .....287
	<a href="#">How to Add Device Topology Elements to the Signal Chain</a> .....288

## Basics on Device Block Handling

<b>Adding device blocks to the signal chain</b>	Before you can map your device ports to signal ports, you have to add device topology elements to the signal chain. You might add only device topology elements which you want to map to function blocks. This means you do not need to add the whole device to the signal chain.
---	---

Device blocks are created from device topology elements, so they inherit their structure and characteristics.

When you add a device topology element to the signal chain, the following happens:

- ConfigurationDesk automatically groups all the elements of one device in one device block.

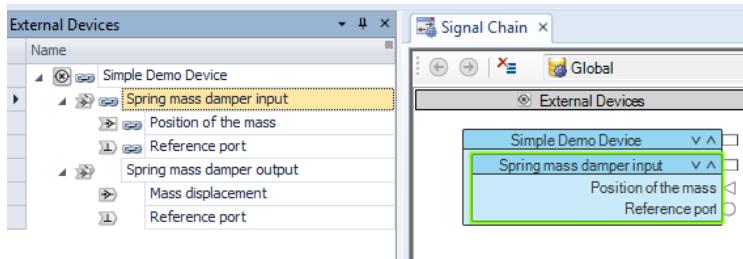
### Tip

Each device topology element can be used only once in the signal chain.

- ConfigurationDesk automatically adds the following device topology elements if they are not already used in the signal chain:
  - Subelements of the added element.
  - Higher-level elements of the added element.

This is required for addressing the elements clearly.

The illustration below shows you an example. The Spring mass damper input port group was added to the signal chain via drag & drop. ConfigurationDesk also adds the subelements Position of the mass and Reference port and the higher-level element Simple Demo Device.



When you add a device topology element to the signal chain, ConfigurationDesk automatically groups all the elements of one device in one device block. In the External Device Browser and in tables, the symbol appears next to the added device topology elements to show you that the elements are used in the signal chain.

ConfigurationDesk works only with one device block of a device. You can add the same device block to several working views, but remember that each working view is only another view of the same device block. For details, refer to [Using Working Views](#) on page 170.

### Removing device block elements from the signal chain

Removing device block elements from the signal chain does not mean removing device topology elements from the active ConfigurationDesk application.

You can remove device blocks, port groups and ports from the signal chain via the Delete from Application command, which is available, for example, on the Home – Delete ribbon. In working views, you can also use the **Del** key.

### Related topics

#### Basics

[Basics on Device Blocks.....](#) 257

#### HowTos

[How to Add Device Topology Elements to the Signal Chain.....](#) 288

## How to Add Device Topology Elements to the Signal Chain

### Objective

To use your external device in the signal chain of a ConfigurationDesk application, you have to add device topology elements (devices, port groups or ports) to the signal chain.

**Restrictions**

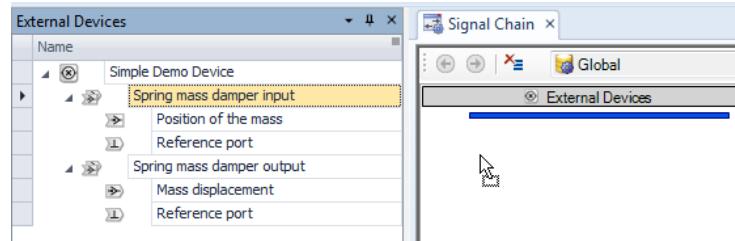
You can add each device topology element to the same working view only once. However, you can add each device block element to several different working views. For details, refer to [Using Working Views](#) on page 170.

**Preconditions**

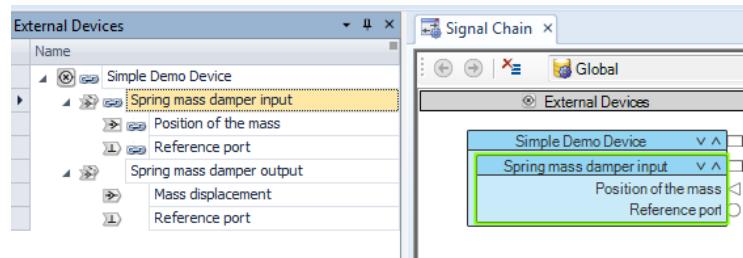
- There are device topology elements in the device topology.
- A working view is open.

**Method****To add a device topology element to the signal chain**

- 1 In the External Device Browser, select the device topology element (device, port group, port) you want to add to the signal chain.
- 2 Drag the device topology element to the working view.



The device topology element and, if necessary, its subelements and higher-level elements are added to the signal chain.

**Result**

You added a device topology element to the signal chain and created a device block.

**Next step**

Now you can map device ports to signal ports.

- [Device Port Mapping](#) on page 312

**Related topics****Basics**

<a href="#">Basics on Device Block Handling</a> .....	287
<a href="#">Basics on Device Blocks</a> .....	257



# Implementing I/O Functionality

---

**Objective**

Implementing and configuring I/O functionality is one of the main work steps in implementing a real-time application. In ConfigurationDesk, I/O functionality is added to the signal chain via function blocks.

---

**Where to go from here****Information in this section**

Basics on Implementing I/O Functionality.....	292
Adding Function Blocks to the Signal Chain.....	307
Device Port Mapping.....	312
Configuring Function Blocks.....	318
Implementing Function Triggers.....	328

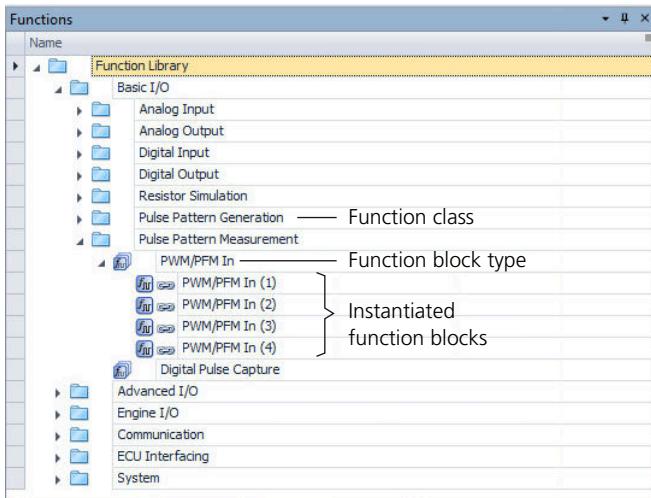
# Basics on Implementing I/O Functionality

<b>Objective</b>	To implement I/O functionality, you have to instantiate a function block from the function library. Every function block has ports, which provide the interfaces to the neighboring blocks in the signal chain.
------------------	---

Where to go from here	Information in this section
	<a href="#">Basics on Instantiating Function Blocks.....</a> 292
	<a href="#">Basics on Function Blocks.....</a> 297
	<a href="#">Characteristics of Signal Ports.....</a> 301
	<a href="#">Characteristics of Function Ports.....</a> 303
	<a href="#">Characteristics of Event Ports.....</a> 305

## Basics on Instantiating Function Blocks

<b>Function block types and function library</b>	<p>The basis for implementing I/O functionality are function block types. Every function block type has unique features which are different from other function block types.</p> <p>To use a function block type in your application, you have to create an instance of it. This instance is called a <i>function block</i>. Function block types can be instantiated multiple times in a ConfigurationDesk application. The types and their instantiated function blocks are displayed in the function library of the Function Browser.</p> <p>The illustration below shows the structure of the function library. Function block types are grouped in function classes. Instantiated blocks are added below the corresponding function block type.</p>
--	--



All displayed function blocks are used in your active ConfigurationDesk application.

#### Note

- The function library allows access to the I/O functionality provided by the software installation on your host PC. Thus, the availability of function block types depends only on the installed plug-ins and not on the configuration of your ConfigurationDesk application.
- A filter for function block types supported by the current hardware topology is active by default. Thus, the displayed function block types depend on the elements of the hardware topology in the Hardware Resource Browser. You can deactivate the filter in the context menu of the Function Browser or on the Home – Filters ribbon. If there are no hardware topology elements, all function block types are displayed.

#### Methods for instantiating function blocks

To instantiate function blocks, you need a function block license. This license is purchased in different sizes that allow you to use a certain number of instantiated function blocks in your active ConfigurationDesk application. There are licenses for the usage of 100, 200, 300, 1000 or an unlimited number of function blocks. For details on function block licenses, refer to [Details on Function Block Licenses](#) on page 42.

There are several methods to instantiate function blocks:

- Via Function Browser
- Via device ports
- Via copy and paste of existing function blocks
- Via drag & drop of a single channel from the hardware topology
- Via drag & drop to the Model-Function Mapping Browser

**Instantiating via Function Browser** You can choose a functionality from the function library and add it to the signal chain via drag and drop. You have to

map the ports of the function block to other blocks in the signal chain afterwards. For instructions, refer to [How to Add Function Blocks to the Signal Chain via Function Browser](#) on page 307.

**Instantiating via device ports** If you already added device ports to the signal chain you can extend them with a suitable function block. ConfigurationDesk suggests one or more functions block types which match the configuration settings of the corresponding device port(s).

After you have selected a functionality, ConfigurationDesk automatically:

- Instantiates new function blocks.
- Maps the device port(s) to the first suitable signal port of the corresponding function block(s).
- Transfers several configuration settings (for example, the allowed failure classes) from a device port to the mapped signal port.

#### Note

This method is only useful if you configured the device port(s) beforehand. The suggested functionality depends only on the settings of the device port. For example: If you have configured a device port as an input, ConfigurationDesk suggest all functions, which can output signals from the function block to an external device.

For instructions, refer to [How to Add Function Blocks to the Signal Chain via Device Ports](#) on page 308.

**Instantiating via copy and paste of existing function blocks** It is useful to copy and paste an already configured function block, if you need several function blocks of the same type and with a similar configuration. You can do this via the Function Browser or via working views.

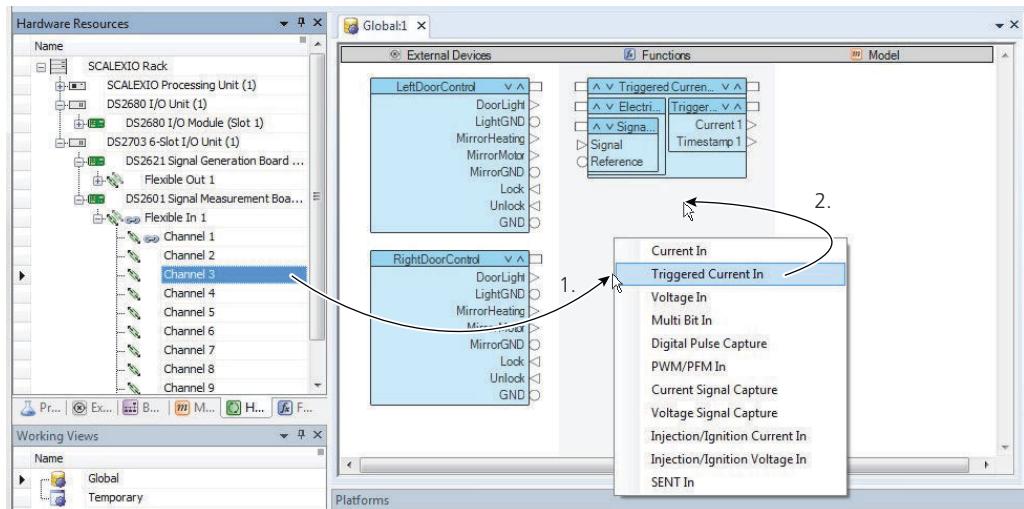
During the paste action, ConfigurationDesk:

- Instantiates the function blocks.
  - Adds the suffix **Copy** to the function block name.
  - Copies the configuration of the function block, except for assigned channels.
- Each function block must be assigned to a specific hardware resource. From this it follows that the assignment cannot be shared with other function blocks.

For instructions, refer to [How to Add Function Blocks via Copy and Paste](#) on page 309.

**Instantiating via drag and drop to a working view** You can instantiate a function block by dragging a single channel from the available hardware topology to a working view and drop it at the desired position.

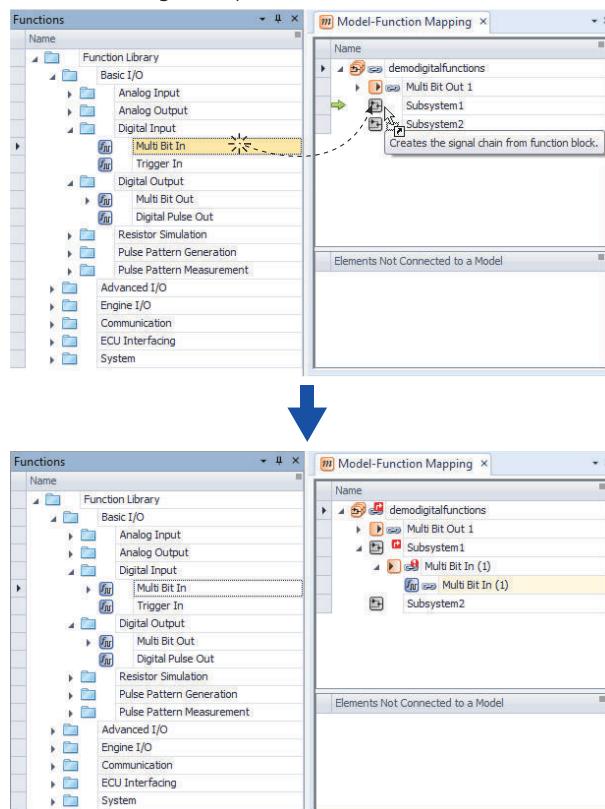
A context menu opens and displays a list of the function block types that support the hardware characteristics of the channel. You can select a function block type from the list to add a function block to the signal chain. See illustration below.



If the function block has more than one channel request, you have to complete the hardware resource assignment after instantiating the function block. For details, refer to [Methods for Assigning Hardware Resources](#) on page 404.

### Instantiating via drag and drop to the Model-Function Mapping Browser

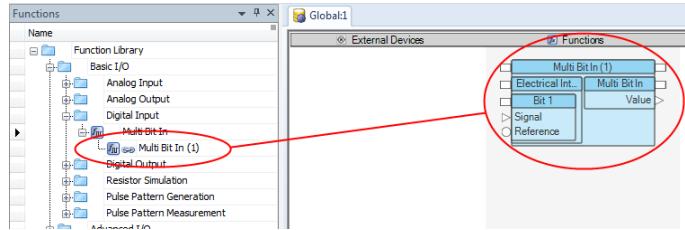
**Browser** The Model-Function Mapping Browser lets you instantiate a function block and create the signal chain for the work with Simulink behavior models via drag & drop.



For more information, refer to [Creating Signal Chains via the Model-Function Mapping Browser](#) on page 555.

## Results of instantiating

After you have instantiated a function block, it is displayed on the active working view and in the Function Browser (below the corresponding function block type) as shown below.



Every function block which is instantiated from the Function Browser or via device ports is given a unique identification name consisting of the function block type and a unique index. You can change this default name as required. For instructions, refer to [How to Rename Function Blocks](#) on page 324.

Each instantiated function block has a state which is determined by the system. For details, refer to [Basics on Function Blocks](#) on page 297.

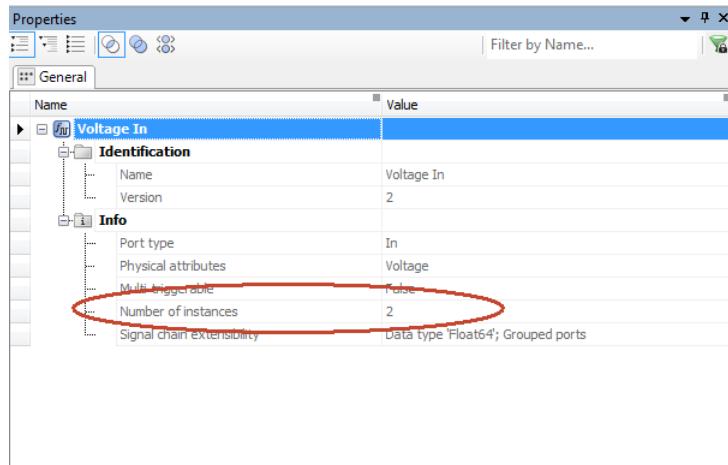
**Displaying in working views** If you have instantiated a function block via copying, it is always displayed in the Global working view and in the Function Browser, and if you pasted it to a specific working view, it is also displayed in that.

### Tip

You can add an instantiated function block to several working views. This does not mean that the function block is used several times in your application. It is used only once, but can be displayed on several working views. For information on using several working views, refer to [Using Working Views](#) on page 170.

**Counting instantiated function blocks** ConfigurationDesk also counts the number of instantiated function blocks for each function block type. So you can find out how many instances of a specific function block type, for example, **Voltage In**, are used in your active ConfigurationDesk application.

The number is displayed in the Properties Browser as shown in the following screenshot.



To access the Number of instances property, select the desired function block type in the Function Browser.

---

#### Showing only instantiated function blocks

By default, all the elements of the function library are displayed in the Function Browser. For a clearer view, you can filter for the instantiated functions blocks from the function library via a context menu command. Then only the instantiated function blocks (with their subelements) and their higher-level elements (for example, the function block types) in the function library hierarchy are displayed.

---

#### Deleting function blocks

You can delete a function block from the signal chain of your ConfigurationDesk application via a context menu command available in the Function Browser and the working area.

## Basics on Function Blocks

---

#### Objective

Function blocks are the central components in the signal chain. Every function block has ports, which provide the interfaces to the neighboring blocks in the signal chain.

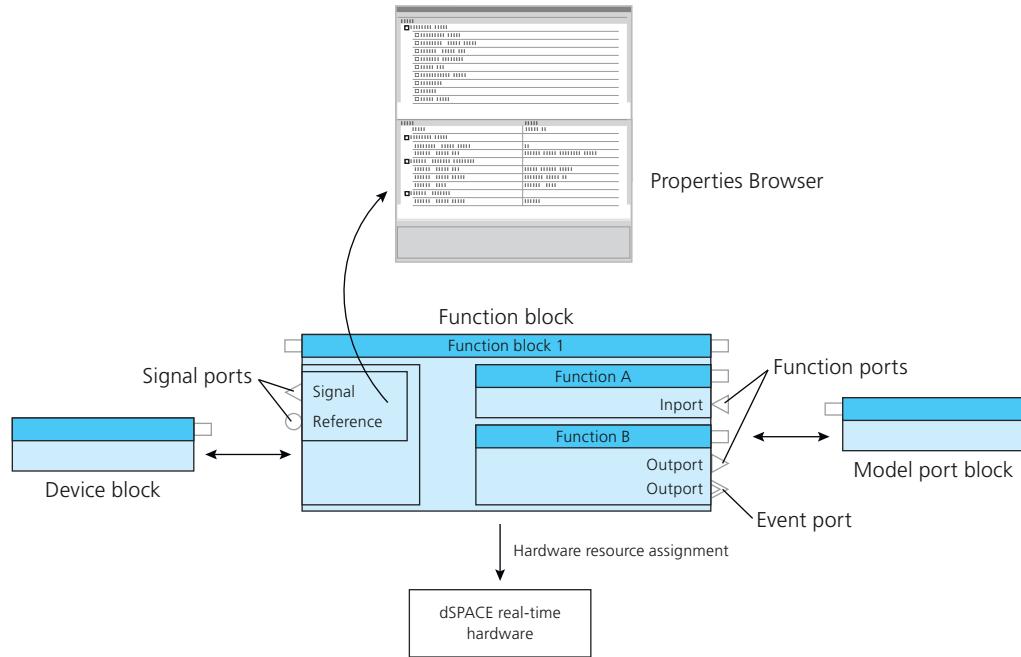
---

#### Purpose of function blocks

A function block provides:

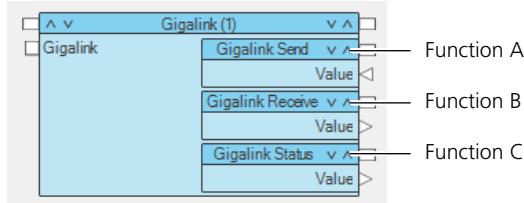
- Specific functionality which is inherited from the corresponding function block type
- Access to the properties for configuring the block properties
- The interface to the real-time hardware (= hardware resource assignment)

- The interfaces to the neighboring blocks in the signal chain:
- Signal ports for mapping the function block to device ports or to other signal ports (device port mapping)
- Function ports and event ports for mapping the function block to model ports (model port mapping)



### Container for functions

A function block serves as a container for functions. Each function block contains at least one function. If two or more functions are combined in one block, their features are closely related. They usually have common properties and they usually need the same types of hardware resources. The following illustration shows an example of a function block which contains three functions.



#### Tip

The complete display of functions in the function block, as shown in the example, depends on the layout of the signal chain in a working view. Functions are only displayed if the block structure is not collapsed. For details, refer to [Collapsing and Expanding Blocks](#) on page 189.

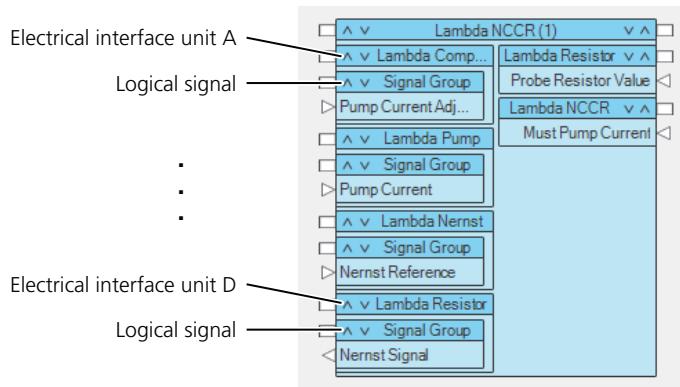
### Container for electrical interface units and their logical signals

A function block also serves as a container for electrical interface units and their logical signals.

An electrical interface unit provides the interface of the function block 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 different channel set to be assigned to. It also provides properties to configure the characteristics of the hardware.

A logical signal combines all the signal ports which belong together to provide the functionality of the signal.

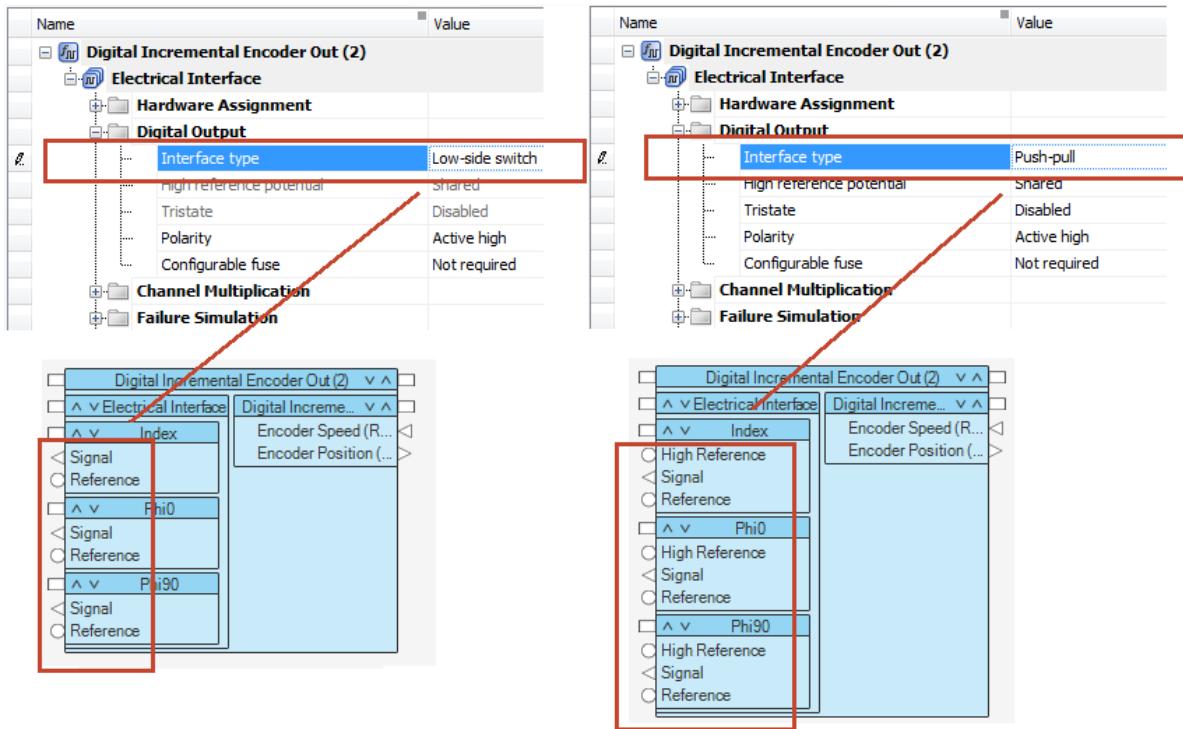
Each function block can contain several electrical interface units, each with several logical signals. The following illustration shows an example of a function block which contains four electrical interface units, each with one logical signal.



### Dependencies between property settings and ports

The setting of function block properties can affect the number and availability of ports at the function block.

The following example shows that changing the **Interface type** property from **Low-side switch** to **Push-pull** affects the appearance of additional signal ports.



You have to note the following behavior, if ports are added or deleted caused by configuration changes: In the example above, three new signal ports (reference ports) are needed after reconfiguration. ConfigurationDesk therefore adds them to the function block. The electrical interface of the function block is changed completely.

All new ports are added with their default configurations. You might have to reconfigure the port properties. However, if ports are already mapped, they are not deleted. A mapped port continues to exist, next to the new ports, until you remove its mapping line(s). In this way ConfigurationDesk supports you in remapping the ports.

#### Possible states of function blocks

ConfigurationDesk determines the status of each function block after every configuration change. It is displayed graphically and in a tooltip (in case of conflicts). The following table shows the possible states, how they are displayed and their effects on code generation during the build process.

State	Graphical Display		Description	Effect on Code Generation
	Working View 1)	Function Browser		
OK	Block frame is black.	No display.	The function block can be used in the application without any restriction.	Code is generated.

State	Graphical Display		Description	Effect on Code Generation
	Working View 1)	Function Browser		
Conflict	Block frame or elements of the block are highlighted (orange). This indicates a warning conflict.	No display.	The required hardware resource(s) is/are not yet assigned to the function block. <a href="#">Assignment Conflicts and Their Effects on Code Generation</a> on page 409.	The effect depends on the assignment as follows: <ul style="list-style-type: none"> <li>▪ Channel set not assigned: The function block is not considered during code generation.</li> <li>▪ Channel not assigned: The I/O access is simulated. In this case, the initial values configured at function outports, are input into the behavior model.</li> </ul>
			The settings for the load rejection behavior in the signal chain differ. For details, refer to <a href="#">Basics on Load Rejection</a> on page 389.	No effect on code generation.
			The block configuration has a failure. For example: Some external configuration elements are not added, such as the wavetable files for configuration of wavetable functionality.	The effect depends on the specific conflict source. For example, default code is generated if wavetable files are missing for wavetable function blocks. In the Conflicts Viewer you can see the effects on code generation (for example, generate no code, generate default code, warning). Warnings are created during the build process and displayed in the Build Log Viewer.
Unresolved <sup>2)</sup>	Block frame is highlighted (orange). This indicates a conflict of the warning category.	No display	The required user files for the corresponding function block type are not available. In this case you cannot: <ul style="list-style-type: none"> <li>▪ Instantiate further function blocks</li> <li>▪ Change the configuration settings</li> </ul>	The function block is not considered during code generation.

<sup>1)</sup> To highlight conflicts, you have to set the appropriate highlighting option. For instructions, refer to [Highlighting Signal Chain Conflicts](#) on page 194.

<sup>2)</sup> Only possible for custom function blocks.

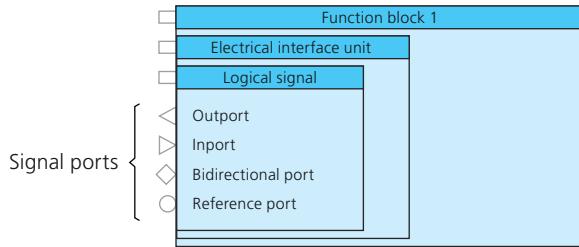
## Characteristics of Signal Ports

### Objective

Signal ports provide the interface to external devices (for example ECUs) via device blocks. They represent the electrical connection points of a function block.

## Port types

The illustration below shows the different types and their graphical display. The presence of specific port types depends on the function block's functionality.



**Signal output** Represents an electrical connection point of a function block which provides signal generation (= output) functionality.

**Signal import** Represents an electrical connection point of a function block which provides signal measurement (= input) functionality.

**Bidirectional signal port** This port is independent of a data direction or current flow. Bidirectional ports are used to implement bus communication, etc.

**Signal reference port** A specific port which represents the connection point for the reference potential of imports, outports and bidirectional ports. For example: With differential signals, this is a reference signal, with single-ended signals, it is the ground signal (GND).

## Port characteristics

The following table lists some general characteristics of signal ports.

Characteristics	Description
Port name	<ul style="list-style-type: none"> <li>▪ Defined by the system, unique per function block</li> <li>▪ Not user-configurable</li> </ul>
Port type	<ul style="list-style-type: none"> <li>▪ Defined by the system</li> <li>▪ Not user-configurable</li> <li>▪ Defines the signal direction of a port and is the basis for device port mapping</li> <li>▪ Possible port types: In, Out, Bidirectional, Reference</li> </ul>
Role	<ul style="list-style-type: none"> <li>▪ Defined by the system</li> <li>▪ Not user-configurable</li> <li>▪ Displays the role of the signal port. Most settings are configurable for signal ports with the Signal role</li> <li>▪ Examples of roles: Signal, Low reference, High reference, Load signal, Load reference</li> </ul>

## Possible port states

The following table shows the possible states, how they are displayed and their effects on code generation during the build process. The state of the port is

determined after every configuration change. It is displayed graphically and in a tooltip.

The port states do not affect the mapping of the signal ports. You can add or delete mapping lines regardless of port state.

State	Display of Port Symbol <sup>1)</sup>	Description	Effect on Code Generation
OK	Black	There are no conflicts.	Code is generated.
Obsolete	Orange. This indicates a warning conflict.	The port is mapped, but due to configuration changes to the function block, the port is no longer used or required. If you remove the mapping line from the obsolete port, this port is deleted. Background: The mapping is not deleted immediately, because this simplifies the remapping of the corresponding device block.	The port is not considered during code generation.

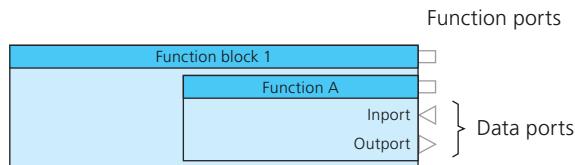
<sup>1)</sup> To highlight obsolete ports, you have to set the highlighting option. For instructions, refer to [Highlighting Signal Chain Conflicts](#) on page 194.

The device port mapping is not considered during the build process, so the different mapping states do not affect code generation. However, device port mapping affects the calculation of the external cable harness.

## Characteristics of Function Ports

**Objective** Function ports provide the data interface to the behavior model via model port blocks.

**Port types** The illustration below shows the different types of function ports. Every function has at least one function port. The presence of specific port types depends on the function block's functionality.



**Function import** At this data port the values from the behavior model are input to be processed by the function.

**Function outport** This data port outputs the value of a function, to be used in the behavior model.

The values available at the data ports can be observed via experiment software, only if you have enabled test automation support in ConfigurationDesk. Then the corresponding variables are written to the variable description file during the build process.

**Function port characteristics**

The following table lists some general characteristics of function ports.

Characteristics	Description
Port name	<ul style="list-style-type: none"> <li>▪ Defined by the system</li> <li>▪ Not user-configurable</li> <li>▪ If test automation support is enabled in ConfigurationDesk, the port name is also used in the variable description file (*.TRC). The name serves as a structure element which groups the variables for the function port values below it.</li> </ul>
Range/value	<ul style="list-style-type: none"> <li>▪ Calculated by the system (depends on the assigned real-time hardware, etc.)</li> <li>▪ System ranges/values are not user-configurable</li> <li>▪ For certain function ports you can reduce the min/max values provided by the system to user-configured values. The min/max values are displayed in the function block properties.</li> </ul>
Unit	<ul style="list-style-type: none"> <li>▪ Defined by the system</li> <li>▪ Not user-configurable</li> <li>▪ The unit is displayed in the function block properties.</li> </ul>
Data width	Vector and scalar data are supported.

**Possible function port states**

The following table shows the possible states, how they are displayed and their effects on code generation during the build process. The status of each port is determined after every configuration change. It is displayed graphically and in a tooltip.

The port states do not affect the mapping of the function ports. You can add or delete mapping lines regardless of port status.

State	Display of Port Symbol <sup>1)</sup>	Description	Effect on Code Generation
OK	Black	There are no conflicts.	Code is generated.
Obsolete	Orange. This indicates a warning conflict.	The port is mapped, but due to configuration changes to the function block, the port is no longer required. If you remove the mapping line from the obsolete port, this port is deleted. Background: The mapping is not deleted immediately, because this simplifies the remapping of the corresponding model port block.	The port is not considered during code generation.
Not mappable	<ul style="list-style-type: none"> <li>▪ Function import: </li> <li>▪ Function output: </li> </ul>	The function port cannot be mapped to model port blocks, because the port's model access is disabled. The port	The port is not considered during code generation.

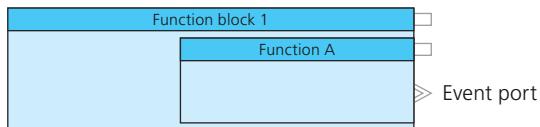
State	Display of Port Symbol <sup>1)</sup>	Description	Effect on Code Generation
		symbols are displayed in the signal chain only if the Filter for Mappable Ports filter is disabled. You can enable model access and therefore reset this state via the Model access property that is available for each function port.	

<sup>1)</sup> To highlight obsolete ports, you have to set the appropriate highlighting option. For instructions, refer to [Highlighting Signal Chain Conflicts](#) on page 194.

## Characteristics of Event Ports

<b>Objective</b>	Event ports provide the interface to trigger asynchronous task of the behavior model via model port blocks.
------------------	---

<b>Port type</b>	The illustration below shows the type of event ports. The presence of event ports depends on the function block's functionality.
------------------	--



Event ports output I/O events to trigger an asynchronous task in the behavior model. The event is triggered by the assigned real-time hardware. Each event port can trigger only one event in the behavior model.

<b>Event port characteristics</b>	The following table lists some general characteristics of event ports.
-----------------------------------	--

Characteristics	Description
Port name	<ul style="list-style-type: none"> <li>▪ Defined by the system</li> <li>▪ Not user-configurable</li> </ul>
Event type	I/O event

<b>Possible event port states</b>	The following table shows the possible states, how they are displayed and their effects on code generation during the build process. The status of each port is determined after every configuration change. It is displayed graphically and in a tooltip.
-----------------------------------	--

The port states do not affect the mapping of the event ports. You can add or delete mapping lines regardless of port status.

<b>State</b>	<b>Display of Port Symbol <sup>1)</sup></b>	<b>Description</b>	<b>Effect on Code Generation</b>
OK	Black	There are no conflicts.	Code is generated.
Obsolete	Orange. This indicates a warning conflict.	The port is mapped, but due to configuration changes to the function block, the port is no longer required. If you remove the mapping line from the obsolete port, this port is deleted. Background: The mapping is not deleted immediately, because this simplifies the remapping of the corresponding model port block.	The port is not considered during code generation.

<sup>1)</sup> To highlight obsolete ports, you have to set the appropriate highlighting option. For instructions, refer to [Highlighting Signal Chain Conflicts](#) on page 194.

# Adding Function Blocks to the Signal Chain

<b>Objective</b>	To use I/O functionality in your application, you have to add a function block to the signal chain.
------------------	---

Where to go from here	Information in this section
	<p>How to Add Function Blocks to the Signal Chain via Function Browser..... 307</p> <p>How to Add Function Blocks to the Signal Chain via Device Ports..... 308</p> <p>How to Add Function Blocks via Copy and Paste..... 309</p>

## Information in other sections

Adding I/O functionality and creating signal chains in one step:	
Creating Signal Chains via the Model-Function Mapping Browser.....	555

## How to Add Function Blocks to the Signal Chain via Function Browser

<b>Objective</b>	To use I/O functionality in your ConfigurationDesk application, you have to add a function block to the signal chain.
------------------	---

<b>Adding via Function Browser</b>	ConfigurationDesk features a Function Browser for you to choose a functionality from. For basic information on the function browser and its contents, refer to <a href="#">Basics on Instantiating Function Blocks</a> on page 292.
------------------------------------	---

<b>Method</b>	<b>To add a function block to the signal chain via Function Browser</b>
	<ol style="list-style-type: none"> <li>1 In the Function Browser, select the <i>function block type</i>, from which you want to instantiate a new function block.</li> <li>2 Drag the function block type to: <ul style="list-style-type: none"> <li>▪ A working view open in the Signal Chain Browser and drop it at the desired position.</li> <li>▪ The Working View Manager and drop it at the desired working view.</li> </ul> </li> </ol>

---

<b>Result</b>	ConfigurationDesk adds the new instantiated function block to the signal chain with its default configuration.
<b>Next Steps</b>	<p>Now you can configure the name and the property settings of the function block and map the ports to other components in the signal chain:</p> <ul style="list-style-type: none"><li>▪ <a href="#">How to Rename Function Blocks</a> on page 324</li><li>▪ <a href="#">Configuring Elements with the Properties Browser</a> on page 131</li><li>▪ <a href="#">Using Tables to Access and Configure Elements</a> on page 151</li><li>▪ <a href="#">Device Port Mapping</a> on page 312</li><li>▪ <a href="#">Model Port Mapping</a> on page 438</li></ul>
<b>Related topics</b>	<p>Basics</p> <div style="background-color: #f0f0f0; padding: 5px; border-radius: 5px;"><a href="#">Creating Signal Chains via the Model-Function Mapping Browser</a>..... 555</div>

## How to Add Function Blocks to the Signal Chain via Device Ports

---

<b>Objective</b>	To use I/O functionality in your ConfigurationDesk application, you have to add a function block to your signal chain.
<b>Adding function blocks via device ports</b>	If you already added device blocks to the signal chain, you can extend them with a suitable function block. ConfigurationDesk suggests one or more function block types which match the configuration settings of the corresponding device port(s). For basic information on adding function blocks, refer to <a href="#">Basics on Instantiating Function Blocks</a> on page 292.
<b>Precondition</b>	A device block is added to the signal chain.

**Note**

This method is only useful if you configured the device port(s) beforehand. The suggested functionality depends only on the settings of the device port. For example: If you have configured a device port as an import, ConfigurationDesk suggest all functions which can output signals from the function block to an external device.

<b>Method</b>	<b>To add a function block to the signal chain via device ports</b> <ol style="list-style-type: none"> <li>1 In a working view, right-click the device port(s) you want to add a function block to.</li> <li>2 From the context menu, choose Extend Signal Chain and select a functionality.</li> </ol>
<b>Result</b>	ConfigurationDesk adds the instantiated function block(s) to the signal chain and displays the new function block(s) below the already existing blocks in the Functions section of the working view.  After you have selected a functionality, ConfigurationDesk automatically: <ul style="list-style-type: none"> <li>▪ Instantiates new function blocks.</li> <li>▪ Maps the device port(s) to the first suitable signal port of the corresponding function block(s).</li> <li>▪ Transfers several configuration settings (for example, the allowed failure classes) from a device port to the mapped signal port.</li> </ul>
<b>Next Steps</b>	Now you can change the name and configure the property settings of the function block and complete the mapping of the ports: <ul style="list-style-type: none"> <li>▪ <a href="#">How to Rename Function Blocks</a> on page 324</li> <li>▪ <a href="#">Configuring Elements with the Properties Browser</a> on page 131</li> <li>▪ <a href="#">Using Tables to Access and Configure Elements</a> on page 151</li> <li>▪ <a href="#">Device Port Mapping</a> on page 312</li> </ul>
<b>Related topics</b>	HowTos <div style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;"> <a href="#">How to Assign Reference Ports</a>..... 276 </div>

## How to Add Function Blocks via Copy and Paste

<b>Objective</b>	It is useful to copy and paste an already configured function block, if you want to add several function blocks of the same type and with a similar configuration. You can do this via the Global working view or another specific working view.
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>▪ A ConfigurationDesk application is active.</li> <li>▪ At least one function block is already instantiated.</li> </ul>

## Possible methods

You can copy and paste function blocks in the following ways:

- You can paste one copy of a function block.  
For instructions, refer to Method 1.
- You can paste multiple copies of a function block.  
For instructions, refer to Method 2.

## Method 1

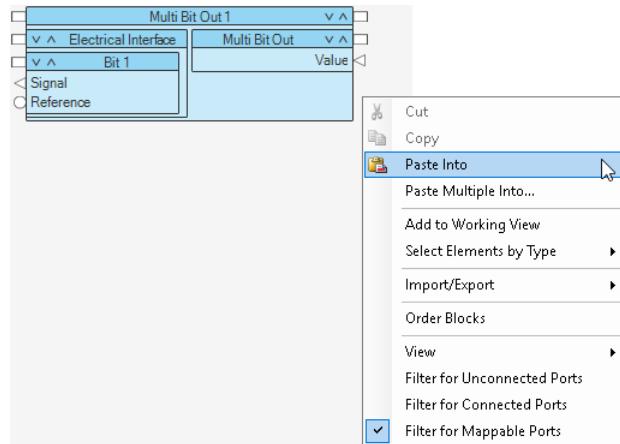
### To add one function block via copy and paste

- 1 In a working view, right-click the function block you want to copy and select Copy.

#### Tip

You can also copy the function block from the Function Browser.

- 2 Select the working view to which you want to paste the copied function block.
- 3 Right-click the working view and select Paste Into.



## Method 2

### To add multiple function blocks via copy and paste

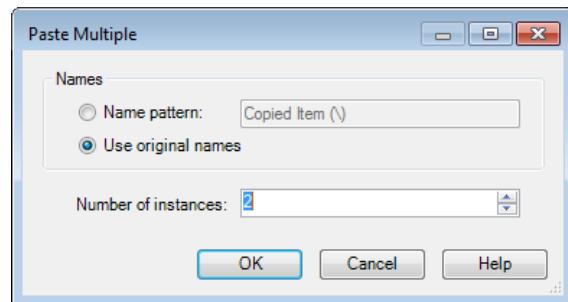
- 1 In a working view, right-click the function block you want to copy and select Copy.

#### Tip

You can also copy the function block from the Function Browser.

- 2 Select the working view to which you want to paste the copied function block.
- 3 Right-click on the working view and select Paste Multiple Into.

A Paste Multiple dialog opens.



- 4 In the dialog, enter the number of function blocks you want to paste to the selected working view.
- 5 Click OK.

---

## Result

You added new function blocks to the signal chain via copy and paste. The new blocks are always displayed in the Global working view and in the Function Browser, and if you pasted them to a specific working view, they are also displayed there.

ConfigurationDesk added the suffix Copy to the function block name. The configuration of the function block was copied, except for the channel assignment and the mapping lines. Each function block must be assigned to a specific hardware resource. The assignment cannot be shared with other function blocks.

# Device Port Mapping

## Where to go from here

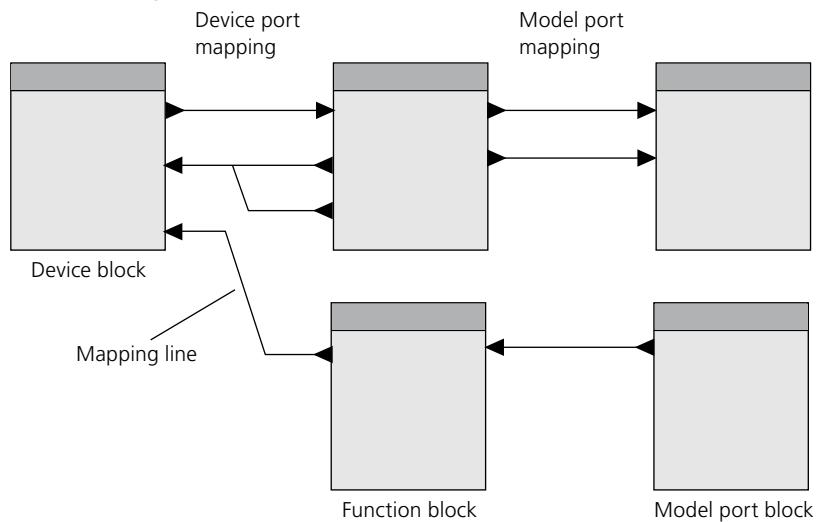
## Information in this section

Methods for Device Port Mapping.....	312
Mapping Conflicts.....	315
Basics on Transferring Port Settings.....	315
How to Transfer Port Settings from Device Ports to Signal Ports.....	317

## Methods for Device Port Mapping

### Definition

Mapping ports means drawing a mapping line between two ports to connect them in the signal chain.



Device port mapping is the mapping of the following ports:

- Device ports to signal ports
- Device ports to other device ports
- Signal ports to other signal ports

In addition to drawing mapping lines, you can also transfer specific configuration settings from device ports to signal ports.

### Mapping methods

ConfigurationDesk provides different methods for device port mapping.

**Automatic mapping by ConfigurationDesk** If you select Extend signal chain from a device port's context menu, ConfigurationDesk automatically creates function blocks. ConfigurationDesk suggests one or more functions block types which match the configuration settings of the corresponding device port. The device port is mapped to the first suitable signal port of the new function block. For instructions, refer to [How to Add Function Blocks to the Signal Chain via Device Ports](#) on page 308.

**Manual mapping** You can draw the mapping lines and map the ports manually.

#### Tip

As an alternative, you can map ports in the signal chain by clicking one port, pressing **Ctrl+Alt** and afterwards clicking another port.

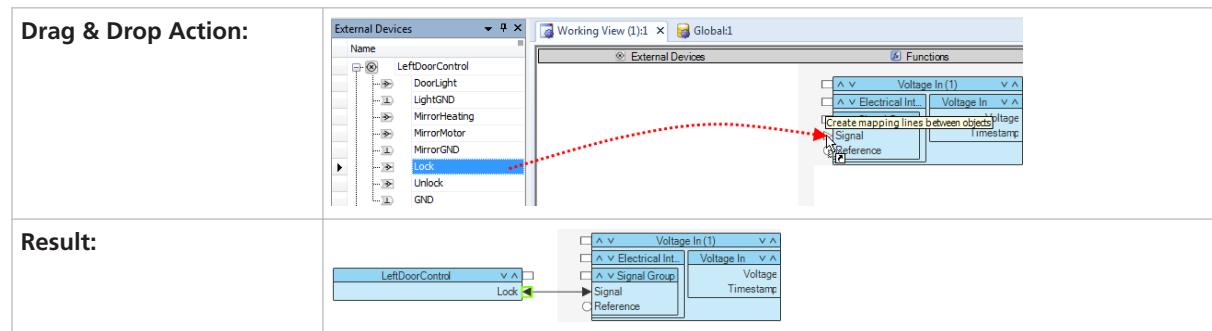
#### Device port mapping via drag & drop

As an alternative to manual mapping, ConfigurationDesk lets you perform device port mapping via drag & drop, i.e., instead of dragging a mapping line from one port to another, you can directly drag a device port from the External Device Browser or a table to signal ports in a working view.

#### Note

- You cannot multiselect items to drag them to target items.
- The target item of the drag & drop operation must be an element of a working view.
- Mapping a device port that is not part of the signal chain via drag & drop automatically adds it to the signal chain.

**Example for device port mapping via drag & drop** Dragging a device port from the External Device Browser to a target signal port of a function block in a working view:



#### Influencing the wiring calculation via mapping

Remember that the connections you map in the signal chain are logical and not real wiring connections. This means, for example, that the mapping of several

signal reference ports (at the function block) to a common signal reference port does not determine the position of the bridge in the external cable harness.

#### Effects of incorrect mapping

##### **NOTICE**

If the device port configuration is incorrect, and/or the device port mapping is incorrect, the wiring information that is calculated (for the external harness) will not match to the electrical requirements of the external devices (for example your ECU).

Risk of damage to the connected devices and/or the dSPACE real-time hardware.

- Configure the device ports and particularly the port type so that they correspond as closely as possible to the characteristics of your device. Use the unspecified port type only as an exception.
- Check the mapping lines. Ensure, that the mapping will not result in damage to the connected device because the electrical characteristics of the mapped ports do not match.

#### Effects on code generation

The device port mapping is not considered during the build process, so it does not affect code generation.

#### Mapping rules

Device port mapping can be performed without any rules. You can map:

- Device ports to signals ports  
This is the basic mapping for implementing a signal chain.
- Device ports (of a device block) to other device ports. This is useful, for example:
  - If you want to connect an external load directly to another external device.
  - If you want to map several reference ports to a common reference.
- Signal ports (of a function block) to other signal ports. This is useful, for example:
  - If you want to use an output signal of a function block as an input for another function block.  
For example, you can combine a Voltage Out function block with a Current Sink function block as a current source.
  - If you want to map several reference ports to a common reference.

If you map bidirectional signal ports or bidirectional device ports to ports with other port types (inport, outport, reference port), a conflict will be generated. These mappings are allowed, but not recommended. Using them is practical only in special cases.

## Mapping Conflicts

<b>Objective</b>	ConfigurationDesk displays the current mapping conflicts in the Conflicts Viewer.  In addition, conflicts are displayed graphically (orange mapping line) and in a tooltip if highlighting of signal chain elements is enabled.
------------------	---

**Mapping conflicts** The following conflicts can be generated by device port mapping:

Conflict	Description	Remedy
The mapping is not recommended.	This mapping is not recommended with regard to electrical engineering. For example, mapping between a bidirectional device port and a signal reference port is not recommended.	Remap the ports or change the port type of the device port, for example from bidirectional to import.
Missing wiring connection.	External wiring information does not match this mapping.	Remap the ports or calculate the wiring information again to resolve the mismatch.
Unconnected device mappings are connected by external wiring.	A mapping line of a port is represented in the wiring information, but there are additional ports with other device mapping lines connected by the external wiring. This means that there might be two unconnected device mapping lines which will be electrically connected by the external wiring.  For example: In the cable harness, several reference signals are connected to a common reference via bridges, but the device port mapping does not contain all the mapping lines for connecting the reference signals.	Complete the mapping or calculate the wiring information again to prevent short circuits in the cable harness.
Invalid pin assignment.	At least one channel is connected according to the calculated wiring information, but there are channels which are not connected according to them.	Check the hardware resource assignment and reassign the hardware, or calculate the wiring information again.

## Basics on Transferring Port Settings

<b>Objective</b>	A subset of device port settings are transferable to the signal port. ConfigurationDesk observes rules for transferring the settings if several device ports are mapped to one signal port.
------------------	---

---

<b>Transferable device port settings</b>	<p>The range of port settings which can be transferred depends on the device type:</p> <ul style="list-style-type: none"><li>▪ ECU device type:<ul style="list-style-type: none"><li>▪ Allowed failure classes</li><li>▪ Load description</li></ul></li><li>▪ Load device type:<ul style="list-style-type: none"><li>▪ Allowed failure classes</li><li>▪ Load description</li><li>▪ Load rejection</li></ul></li></ul> <p>You can transfer all of the above mentioned settings at once via Transfer Settings - All Settings.</p>
<b>Restrictions of receiving transfer settings for signal ports</b>	<p>The following <a href="#">function ports</a> have restrictions on receiving settings:</p> <ul style="list-style-type: none"><li>▪ Signal reference ports cannot receive settings from device ports.</li><li>▪ Load description is available only for signal imports, so it is transferred only to signal imports.</li><li>▪ Allowed failure classes can only be transferred to ports in <a href="#">function blocks</a> for which Failure simulation is set to Required.</li><li>▪ Only signal ports which are mapped directly to the transferring device ports can receive the following property settings:<ul style="list-style-type: none"><li>▪ Allowed failure classes</li><li>▪ Load description</li></ul></li><li>▪ Only signal ports for external loads from a function blocks with load signals which are mapped to the transferring device ports can receive Load rejection settings.</li></ul>
<b>Rules for transferring settings of several elements</b>	<p>If you have mapped several device ports to one signal port, ConfigurationDesk observes rules to transfer the settings.</p> <p>Identical settings of the device ports are transferred to the signal port of the function block.</p> <p>If the settings are not identical, the following rules apply:</p> <ul style="list-style-type: none"><li>▪ If you are transferring the settings from several elements you have selected while pressing the <b>Ctrl</b> key, the setting of the last element you have selected is transferred.</li><li>▪ If you are transferring the settings from device blocks or port groups, the settings of the last port in the block or group is transferred.</li></ul> <p>Sometimes the rules above apply in combination. For example, if you have selected a device port and then a port group while pressing the <b>Ctrl</b> key, the setting of the last port in the port group is transferred.</p>

**Related topics****HowTos**

How to Transfer Port Settings from Device Ports to Signal Ports.....	317
--	-----

## How to Transfer Port Settings from Device Ports to Signal Ports

**Objective**

Transferring port settings simplifies configuration work.

**Basics**

For basic information, refer to [Basics on Transferring Port Settings](#) on page 315.

**Preconditions**

- Device ports whose settings are to be transferred to signal ports have to be mapped to signal ports.
- A working view containing the device ports is open.

**Method****To transfer port settings from device ports to signal ports**

- 1 In the working view, select the device block elements (device block, port group or port), you want to transfer the settings from.
- 2 Right-click the selection to open the context menu.
- 3 In the context menu, select Transfer Settings – (proposed settings).

**Result**

You transferred port settings from device ports to signal ports.

**Related topics****Basics**

Basics on Transferring Port Settings.....	315
Configuring External Devices.....	272
Handling Loads.....	385
Specifying Failure Simulation in ConfigurationDesk.....	373

# Configuring Function Blocks

<b>Objective</b>	You can configure the properties of the function blocks according to your needs. ConfigurationDesk's Properties Browser gives you access to the block and port properties.
------------------	--

Where to go from here	Information in this section
	<a href="#">Basics on Function Block Configuration</a> ..... 318
	<a href="#">Categories of Configurable Parameters</a> ..... 320
	<a href="#">Basics on Hardware Resources and Channel Types</a> ..... 321
	<a href="#">How to Rename Function Blocks</a> ..... 324
	<a href="#">How to View Circuit Diagrams of Hardware Resources</a> ..... 325

## Basics on Function Block Configuration

<b>Objective</b>	Several properties of a function block are user-configurable. Some general aspects of configuring and handling configuration data are described here.
<b>Dependencies to model and hardware</b>	<p>ConfigurationDesk lets you configure I/O functionality independently of the behavior model and the real-time hardware:</p> <ul style="list-style-type: none"> <li>▪ You can configure properties without changing the behavior model. However, in some cases reconfiguration changes the existence of function ports. In this case you have to check the effects on the corresponding model ports and their representation in the behavior model.</li> <li>▪ You can configure properties with and without connected real-time hardware, and even without a hardware topology.</li> </ul>
<b>Access to function block properties</b>	<p>As a precondition for accessing the function block properties, at least one function block must be instantiated and therefore available in the signal chain.</p> <p>Every function block and its elements (for example, the ports) have properties. Some of them are user-configurable, others are only for information purposes.</p> <p>The Properties Browser shows you all the properties of the function block element(s) you have selected in a working view or in a table. If a property is configurable, you can change the setting there.</p>

If you have selected several elements, the Properties Browser displays all the properties available for them. However, only the values which are identical for all selected elements are shown.

**Tip**

You can change the value of a property separately for one element or for several selected elements at once.

For details on the Properties Browser, refer to [Configuring Elements with the Properties Browser](#) on page 131.

---

**Flexible configuration settings**

The concept of ConfigurationDesk allows very flexible configuration settings, with the effect that one setting might not match the setting of another function block property. These configuration conflicts are allowed at first. The conflicts are displayed in the Conflicts Viewer. However, before you build a real-time application, you have to resolve the conflicts to get proper build results.

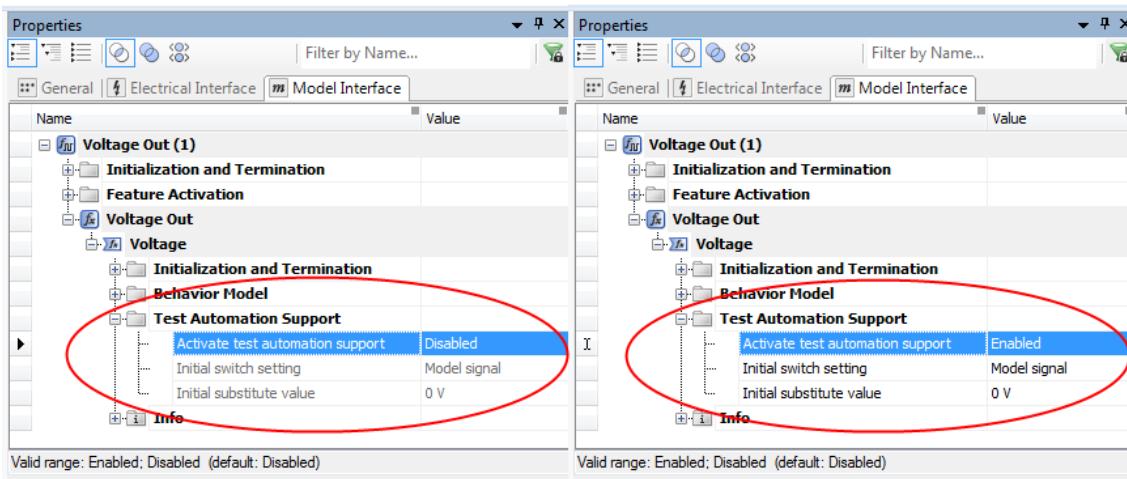
The property which might cause the most conflicts is the **Assigned channel set** property (available for most function block types) for the following reason:

Every function block type supports one or more suitable channel sets (each based on a specific channel type). This means that you can use a function block in combination with hardware resources which have different electrical characteristics (for example, more or less functionality). After you select a channel set, the function block supports only settings and value ranges which are provided by the hardware resources of the selected channel set. All unsupported settings (= mismatches) are then displayed in the Conflicts Viewer.

---

**Dependencies between different properties**

For several properties it might not be possible to change the values or settings, because their features are disabled by another property. In this case the property and the current values are displayed in gray, and cannot be changed. For example, the test automation support is configurable only if the **Activate test automation support** property is set to **Enabled** as shown below.



### Managing configuration data

The configuration data (property settings etc.) of a function block and its usage in the signal chain is saved automatically together with the ConfigurationDesk application to which it belongs.

Exporting an application allows you to save the contents of one application including the function block configuration in a dSPACE archive file (DSA file) and transfer the application to another project.

You can import and export an application via context menu commands in the Project Manager. For details on handling applications in ConfigurationDesk, refer to [Managing ConfigurationDesk Applications](#) on page 100.

## Categories of Configurable Parameters

### Objective

The configurable parameters of a function block can be categorized according to their characteristics. You cannot change these characteristics.

### Inline parameter

The inline parameter type is the standard parameter category for function blocks in ConfigurationDesk. You cannot access them via other software tools such as the experiment software.

Inline parameters are not specially marked either in the properties or in the user documentation.

#### Note

Parameters which affect the behavior of several function blocks, for example all function blocks instantiated from a specific function block type, do not exist.

---

**Tunable parameter**

Unlike inline parameters, you can change the value of tunable parameters via your experiment software. For access of these parameters from outside ConfigurationDesk, they are available in the variable description file which is generated during the build process.

There are two types of tunable parameters:

- Stop-Run-Tunable parameter

You can change this parameter in the experiment tool, when the real-time application is running or stopped. The changes do not take effect until the application status changes from stopped to running.

- Run-Time-Tunable parameter

You can change this parameter in the experiment tool, when the real-time application is running or stopped. The changes take effect immediately.

Tunable parameters are not specially labeled or marked in the properties. However, they are indicated in the user documentation. Refer to the online help via **F1** key, for example.

## Basics on Hardware Resources and Channel Types

---

**Hardware resource**

A hardware resource is 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.

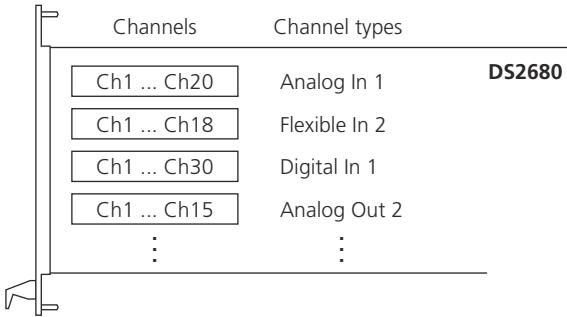
This means that not every function block can be executed on every hardware resource. There could be limitations on a function block's features and/or the physical ranges.

---

**Channel types**

To identify all the hardware resources (channels) in the hardware system that provide exactly the same characteristics, ConfigurationDesk provides channel types, for example, the **Flexible In 1** channel type.

An I/O board in a hardware system can have channels of several channel types. Channels of one channel type can be available on different I/O boards. The illustration below shows (as an example) the DS2680 I/O Unit with a selection of available channel types and the number of channels related to these channel types.

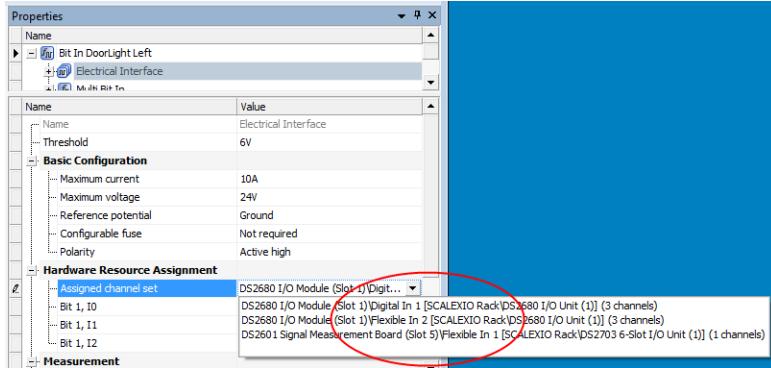


### Definition of channel set

A channel set is 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.

### Function block support of channel types

Every function block type supports one or more suitable channel types. If there are several channel types, you can select the one you need by assigning a channel set which supports the channel type as shown below:



Support of several channel types lets you use a function block in combination with hardware resources which have different electrical characteristics (for example, more or less functionality). After you assign a channel set which is based on a specific channel type, the function block supports only the settings and value ranges which are provided by that channel type. All unsupported settings (= mismatches) are then displayed in the Conflicts Viewer.

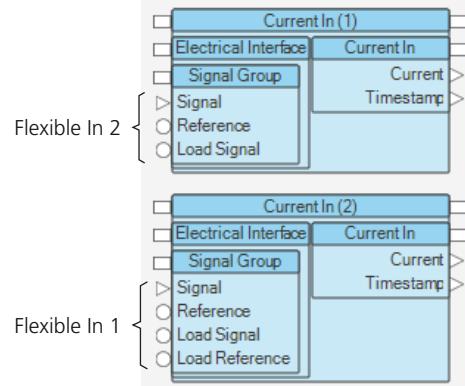
**Assigning a channel set** You can assign a channel set if the hardware topology of the ConfigurationDesk application contains hardware which provides at least one channel set which is suitable for the function block. You can change this assignment at any time during configuration. However, to avoid conflicts and to reduce conflict resolutions, you should assign a channel set as early as possible.

**Using a channel set based on a different channel type** Changing the channel type (via channel set assignment) affects the function block as follows:

- The functionality of the function block changes. For example, channel type A features noise generation, channel type B does not. The extent of the

differences varies and depends on the specific function block type and the selected channel type.

- The signal ports providing the electrical interface of the function block are firmly tied to the characteristics of the I/O circuits of the channel type. The number and type of signal ports that are provided therefore depend on the selected channel set (based on a specific channel type) as shown below. Note that sometimes the availability of signal ports depends on the settings of several other properties such as the Reference potential property.



The circuit diagrams are available in the software and in the documentation. Refer to [Circuit Diagrams of Channel Types \(ConfigurationDesk I/O Function Implementation Guide\)](#).

#### Aspects on choosing a channel type

Note the following points when choosing a channel type (via channel set assignment):

- You should use dSPACE real-time hardware resources that are as suitable and economical as possible for the function block functionality you require. This means:
  - Use only tailored channels, i.e., ones that are not oversized for the required functionality.
  - Use channel sets and channels of hardware that is already present and available.
- Specify the maximum values of the expected electrical signal (for signal imports) or the required maximum values (for signal exports) via the Required voltage and Required current properties according to the required values of the external devices, not to the values the hardware supports. For basics, refer to [Specifying Current and Voltage Values for Channel Multiplication \(ConfigurationDesk I/O Function Implementation Guide\)](#).
- Your electrical devices and/or the wiring of these devices may require specific I/O characteristics, for example, differential signals or a specific reference potential. One channel type might be more suitable than others for a specific use case.

The circuit diagrams of the channel types will help you choose a suitable type. These circuit diagrams are available in the software and in the user documentation. Refer to [Circuit Diagrams of Channel Types \(ConfigurationDesk I/O Function Implementation Guide\)](#).

To help you choose a suitable channel type, the documentation contains tables showing the differences between the channel types in detail. Refer to the *Hardware Dependencies (<Function Block Name>)* chapters (in [ConfigurationDesk I/O Function Implementation Guide](#)) which are available for each function block type.

---

**Related topics**

**Basics**

Assigning Hardware Resources to Function Blocks.....	399
--	-----

## How to Rename Function Blocks

---

**Objective**

Each function block has a unique default name defined by the system. You can change this name to comply with your project definitions, etc.

---

**Display and use of function block names**

Function block names are used for block identification. They are displayed:

- In the graphical user interface (Function Browser, working views, tables, Properties Browser, Conflicts Viewer)
- In error, warning and information messages
- In exported ConfigurationDesk files, for example in the variable description file (\*.TRC)

---

**Allowed characters**

- All characters except for the following ones are allowed:
  - The single characters " and \
  - The character strings \*/ and /\*
  - Number of characters: 1 ... 127

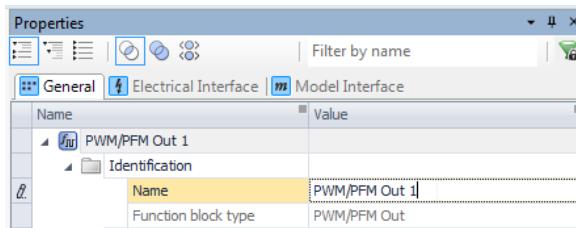
---

**Precondition**

A function block is instantiated.

**Method****To rename function blocks**

- 1 Select the function block that you want to rename.
- 2 In the Properties Browser, click the Name edit field as shown in the following example.

**Tip**

In most panes, you can also double-click the function block or select it and press **F2** to make the name editable.

- 3 Enter a new name and press **Enter** or click anywhere outside the edit field to confirm.

**Result**

You renamed a function block. If the new name is already used for a function block in the application, a conflict is generated and displayed in the Conflicts Viewer.

## How to View Circuit Diagrams of Hardware Resources

**Objective**

Circuit diagrams provide details of the electrical characteristics of the hardware resources of the real-time hardware. You can access them via instantiated function blocks.

**Purpose of circuit diagrams**

You should view circuit diagrams if you want:

- To choose a suitable channel set which is based on a specific channel type. Your electrical devices and/or the wiring of these devices may require specific I/O characteristics. One channel type might be more suitable than others for a specific use case.
- To get information on the possible hardware configurations. Some hardware resources are configurable via jumper settings, for example, you have to specify internal and/or external loads.
- To get details on the mapping of signal ports to device ports and on the wiring to external devices, and to view the differences between mapping and wiring.

**Available diagram types**

ConfigurationDesk provides the following diagram types:

- At least one use-case-specific circuit diagram is available for each channel type. For example, for the Flexible In 1 channel type, there is a circuit diagram showing the Current and voltage measurement of low-side controlled load use case.
- General diagrams for channel multiplication
- Diagrams for complex function block types where channels of different channel types are combined to get the required functionality, for example, for the Lambda DCR function block type.

**Tip**

The circuit diagrams shown in ConfigurationDesk are also available in the user documentation. For diagrams on the channel types, refer to [Circuit Diagrams of Channel Types \(ConfigurationDesk I/O Function Implementation Guide\)](#).

**Contents**

Circuit diagrams can provide the following hardware details:

- The connection to external devices, such as ECUs and loads. This includes the mapping between signal ports and device ports as well as the wiring.
- Configuration settings for internal and/or external loads with explanations
- Switch setting with explanations of their effects
- Locations and trigger levels of fuses

**Precondition**

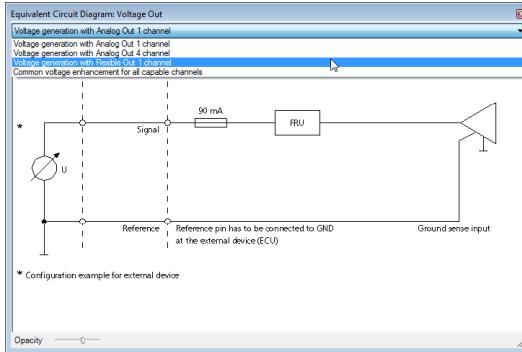
A least one function block is instantiated.

**Method****To view circuit diagrams of hardware resources**

- 1 In a working view, right-click the function block and select Show - Open Equivalent Circuit Diagram from the context menu as shown below.



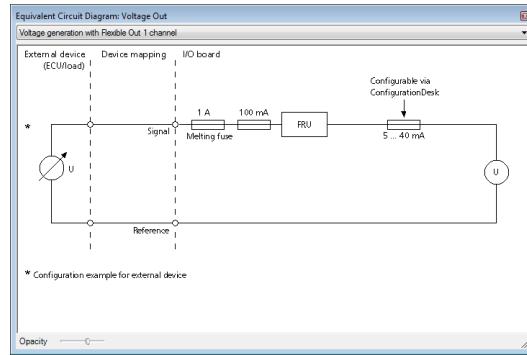
A dialog with the first available circuit diagram opens.



- From the list at the top of the dialog, select the desired circuit diagram.

## Result

ConfigurationDesk shows the circuit diagram that you selected.



## Related topics

### Basics

[Basics on Hardware Resources and Channel Types.....](#) 321

# Implementing Function Triggers

## Objective

The functions of ConfigurationDesk are triggered by tasks defined in the behavior model. You should be familiar with the basics on function triggers, and you should know how to avoid invalid data transfer to or from the behavior model.

## Where to go from here

### Information in this section

Basics on Function Triggers.....	328
How to Avoid Multiple Function Triggers.....	332

### Information in other sections

When you implement function triggers, you should be familiar with the basics of task modeling.	
Basics on Tasks, Events, and Runnable Functions.....	480

## Basics on Function Triggers

### Introduction

If you are familiar with function triggers, you know the timing of the I/O access and you can avoid trigger problems caused by multiple trigger sources.

### I/O access bound to function modules

I/O access is bound to the execution of so-called *function modules*. Function modules are always part of a task. The following parts of a behavior model define a function module:

**Data port blocks (from the Model Interface Blockset) with the same sample time** A function module consists of all the blocks in the model that have the same sample time, if at least one data port block has the following characteristics:

- The block's sample time matches the sample time.
- The block is not contained in an atomic subsystem.

If these conditions are fulfilled, all the data port blocks with the same characteristics are assigned to the function module.

**Data port blocks (from the Model Interface Blockset) in atomic subsystems with the same sample time** A function module consists of all the blocks in an atomic subsystem that have the same sample time, if at least one data port block has the following characteristics:

- The block's sample time matches the sample time.
- The block is contained in the atomic subsystem. The relevant atomic subsystem is the one directly above the data port block. For example, in the hierarchy `Model/Atomic1/Virtual1/Atomic2/Virtual2/MyBlock` the `MyBlock` data port block defines a function module for the `Atomic2` subsystem, not for `Atomic1`.

If these conditions are fulfilled, all the data port blocks with the same characteristics are assigned to this function module.

**Simulink Import/Outport blocks with the same sample time** A function module can consist of all the blocks in the model that have the same sample time, if at least one Simulink Import/Outport has the following characteristics:

- The block's sample time matches the sample time.
- The block resides on the root level of the model.

If these conditions are fulfilled, all the Simulink Imports/Outports with the same characteristics are assigned to the function module.

#### Note

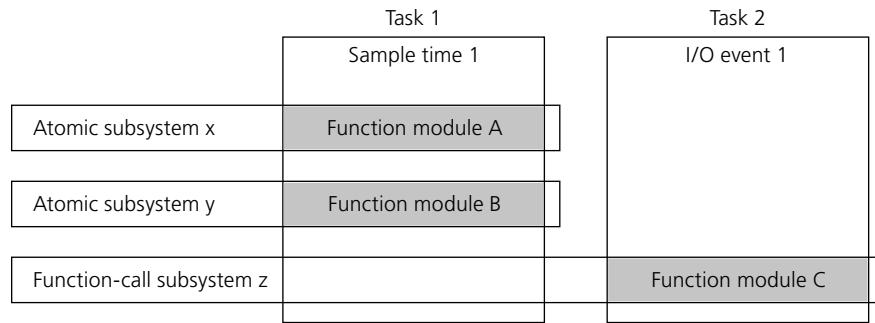
Data port blocks not contained in atomic subsystems and Simulink Imports/Outports on the root level of the behavior model define separate function modules, even if the blocks' sample times are identical.

---

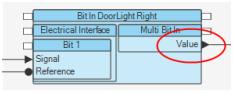
#### Example of function modules

Suppose your behavior model contains three atomic subsystems, and each subsystem contains at least one data port block. Two of these atomic subsystems are calculated with the same Sample time 1. The third atomic subsystem is a function-call subsystem connected to a Hardware-Triggered Runnable Function block. In ConfigurationDesk, the associated runnable function is assigned to a task that is triggered by an I/O event.

Code generation would result in Function module A and Function module B, both of which are part of Task 1. Function module C is part of asynchronous Task 2.

**Timing of the I/O access**

The table below describes the relationship between the task triggers, the function triggers, and the port status, that is, the temporal characteristics of I/O access. It is assumed that the task consists of one function module.

Execution Step During Real-Time Simulation		Resulting Port Status
1.	Task is triggered. The functions are triggered to update the values of the signal imports, i.e., the input signals are sampled.	New values are available at signal imports in ConfigurationDesk. 
2.	Execution of the task begins. Execution of the function module begins. The functions that are mapped to all the data imports assigned to the function module are triggered to update the values of the function outports.	New values are available at function outports in ConfigurationDesk. 
3.	Data read <sup>1)</sup>  Execution of model code <sup>1)</sup>	New values are available at data imports in the behavior model.  New values are available at data outports in the behavior model. 
	Data write <sup>1)</sup>	New values are available at function imports in ConfigurationDesk. 
4.	Execution of the function module finishes. The functions that are mapped to all the data outports assigned to the function module are triggered to	New values are available at signal outports in ConfigurationDesk. 

Execution Step During Real-Time Simulation	Resulting Port Status
update the values of the signal outports.	

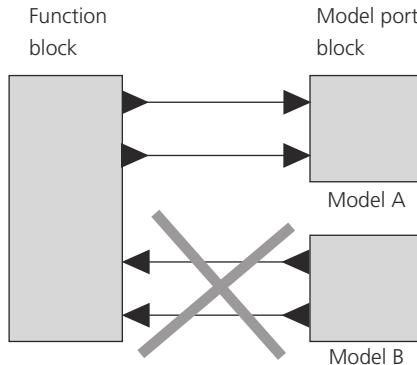
<sup>1)</sup> For details on the execution order of blocks, refer to the Simulink Coder documentation from MathWorks.

### Note

I/O access is performed only during real-time simulation. During Simulink simulation, Data Imports output the Default value and Data Outports do not output any signal at all.

#### Multi-model restriction

In general, model port blocks that reside in different behavior models must not be mapped to the same function block.



However, the following function blocks are an exception to this rule if the mapped model port blocks belong to the same application process:

- Bus Configuration block
- CAN block
- LIN block
- Gigalink block
- Power On Signal In block
- Non-Volatile Memory Access block

For an overview of the mapping rules, refer to [Rules for Model Port Mapping](#) on page 443.

#### Malfunction due to multiple trigger sources

When a function block for which multiple trigger sources are not allowed is triggered by multiple function modules, a conflict is displayed in the [Conflicts Viewer](#).

You have to resolve this conflict before you start the build process to get proper build results. You should avoid multiple function triggers. Refer to [How to Avoid Multiple Function Triggers](#) on page 332.

**Note**

The assignment of a data port to its associated function module can be determined only during the model code generation and not during the model analysis. If you move data port blocks in the behavior model from one subsystem to another, this action might trigger the "Function block: Multiple triggering" conflict even if the reason for this conflict was eliminated and the model was re-analyzed in ConfigurationDesk. Generating the model code (either by starting the build process or by generating it explicitly) updates this information and resets the conflict.

**Related topics****Basics**

Implementing I/O Functionality.....	291
Modeling Concept (Model Interface Package for Simulink - Modeling Guide 	
Specifying the Model Interface.....	417

## How to Avoid Multiple Function Triggers

**Objective**

To avoid invalid data transfer to or from the behavior model.

**Invalid data transfer**

If a function has multiple trigger sources, default code is generated during the build process instead of task-based read/write access. In consequence, only initial values are transferred to or from the behavior model.

**Method****To avoid multiple function triggers**

- 1 Make sure that a function is triggered by only one function module. You can use one of the following methods:
  - ConfigurationDesk:  
Change the mapping of functions and model port blocks in ConfigurationDesk.
  - Behavior model:  
Move the model port blocks into appropriate subsystems in the behavior model.

---

<b>Result</b>	The function has only one valid trigger source.
---------------	---

---

<b>Related topics</b>	Basics
-----------------------	--------

Basics on Function Triggers.....	328
----------------------------------	-----



# Adding Bus and Gigalink Communication to the Signal Chain

## Objective

To connect the external device and the real-time hardware via a bus interface, you must implement an appropriate communication. In addition, you can connect a SCALEXIO system to another SCALEXIO or PHS-bus-based system via Gigalink.

## Where to go from here

### Information in this section

Building the Signal Chain for CAN Bus Communication.....	336
Building the Signal Chain for LIN Bus Communication.....	344
Building the Signal Chain for FlexRay Communication.....	351
Building the Signal Chain for Communication Using an Ethernet Interface.....	362
Building the Signal Chain for Gigalink Communication.....	363

### Information in other sections

#### ConfigurationDesk UART Implementation

Shows you how to implement a protocol for UART serial communication in ConfigurationDesk.

# Building the Signal Chain for CAN Bus Communication

## Where to go from here

## Information in this section

Basics on Implementing CAN Communication.....	336
Recommended Workflow for Implementing CAN Communication.....	339

## Basics on Implementing CAN Communication

### Introduction

CAN communication can be modeled via the RTI CAN MultiMessage Blockset in a Simulink behavior model. If you do this, you can add the behavior model to the ConfigurationDesk application. In the application, you must use CAN function blocks to build the signal chain and to access real-time hardware.

Alternatively, you can use the Bus Manager in ConfigurationDesk to configure and implement CAN communication in ConfigurationDesk. For basic information and instructions on using the Bus Manager, refer to [ConfigurationDesk Bus Manager Implementation Guide](#).

### Workflows and demo

Implementing CAN communication in ConfigurationDesk is a required step for connecting an ECU to a CAN bus that is simulated by a SCALEXIO system.

- For an overview of all the required steps, refer to [CAN Bus Connection \(SCALEXIO – Hardware and Software Overview\)](#).
- For an overview of a recommended workflow for implementing CAN communication in ConfigurationDesk, refer to [Recommended Workflow for Implementing CAN Communication](#) on page 339.
- For a demo application with an example implementation of CAN communication using the RTI CAN MultiMessage Blockset, refer to [Using the CANMMAppl Application \(ConfigurationDesk Demo Projects\)](#).

### Mandatory components

When you configure CAN communication via the RTI CAN MultiMessage Blockset, the following components are mandatory for implementing the CAN communication in ConfigurationDesk.

Component	Description
Configuration Port blocks	Configuration Port blocks are specific model port blocks. They are mandatory for providing CAN bus settings that are specified via the RTI CAN MultiMessage blockset to ConfigurationDesk. Refer to <a href="#">Configuration Port blocks</a> on page 337.

Component	Description
CAN function blocks	CAN function blocks are mandatory for specifying the access to real-time hardware and configuring hardware-related settings for CAN communication. Refer to <a href="#">CAN function blocks</a> on page 338.
Suitable hardware resources	Hardware resources that are suitable for CAN function blocks are mandatory for accessing real-time hardware. The hardware topology must contain suitable hardware resources with one unused channel for each CAN controller. For an overview of suitable hardware resources and the hardware dependencies, refer to <a href="#">SCALEXIO Hardware Dependencies (CAN)</a> ( <a href="#">ConfigurationDesk I/O Function Implementation Guide</a> ).

## Configuration Port blocks

Configuration Port blocks are specific model port blocks. They can be created only when you analyze a Simulink model in ConfigurationDesk.

### Note

You cannot create Configuration Port blocks by extending the signal chain in ConfigurationDesk.

However, Configuration Port blocks can be created during model analysis only if the following preconditions are fulfilled:

- In the Simulink model, `dsrt.tlc` is specified as the system target file. Refer to [Overview of the RTI CAN MultiMessage Blockset \(RTI CAN MultiMessage Blockset Reference\)](#).
- S-functions are created for all RTICANMM blocks of the Simulink model, e.g., via the [Create S-Function for All CAN Blocks](#) command in the Options menu of the RTICANMM GeneralSetup block.

When the preconditions are fulfilled, ConfigurationDesk creates one Configuration Port block for each identified RTICANMM ControllerSetup block during model analysis. The created Configuration Port blocks are named after the related RTICANMM ControllerSetup blocks.

Each Configuration Port block provides one configuration port that is named after the related CAN controller. Configuration Port blocks and configuration ports are indicated by the following symbols, e.g., in the Model Browser:

Symbol	Element
	Configuration Port block
	Configuration port

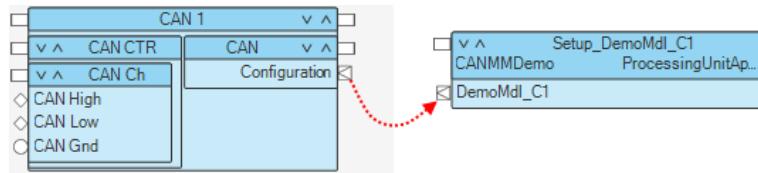
The configuration port must be mapped to a Configuration port of a CAN function block to implement the related CAN bus settings in the signal chain.

#### Note

If you do not map a configuration port of a Configuration Port block to a Configuration port of a CAN function block, the related CAN communication is not implemented in the real-time application when you start the build process.

### CAN function blocks

**Mapping CAN function blocks to Configuration Port blocks** You can map CAN function blocks to Configuration Port block only manually, i.e., you must manually draw a mapping line between the Configuration port of a CAN function block and the configuration port of a Configuration Port block.



When you do this, settings specified for the related RTICANMM ControllerSetup block in the Simulink model are available for the CAN function block. As a result, some properties of the CAN function block are read-only and/or property values are derived from the RTICANMM ControllerSetup block.

#### Tip

Select the CAN function block to view the CAN communication settings in the Properties Browser.

You can map each CAN function block to only one Configuration Port block that was created for an RTICANMM ControllerSetup block. Therefore, you must use one CAN function block for each Configuration Port block that is relevant for CAN communication.

**Configuring CAN function blocks** To access real-time hardware, you must assign a hardware resource to the CAN function block. Additionally, and depending on the assigned hardware, you can specify additional hardware-related settings via the CAN function block, for example:

- Using low-power mode and partial networking
- Using bus statistics
- Enabling failure simulation

- Enabling feedthrough mode

**NOTICE**

Depending on the real-time hardware, the hardware might be destroyed if the CAN bus carries high voltages while feedthrough mode is enabled. Refer to [Configuring the Basic Functionality \(CAN\) \(ConfigurationDesk I/O Function Implementation Guide](#) .

Depending on the settings, signal ports and/or function ports are added to the CAN function block. You can, for example:

- Map signal ports to ports of external devices.
  - Map function ports to model ports to exchange data with the Simulink model.
- For more information on the CAN function block and its configurable settings, refer to [CAN \(ConfigurationDesk I/O Function Implementation Guide](#) .

---

**Related topics****Basics**

<a href="#">CAN (ConfigurationDesk I/O Function Implementation Guide</a> 	201
<a href="#">Managing Real-Time Hardware</a> .....	201

## Recommended Workflow for Implementing CAN Communication

---

**Introduction**

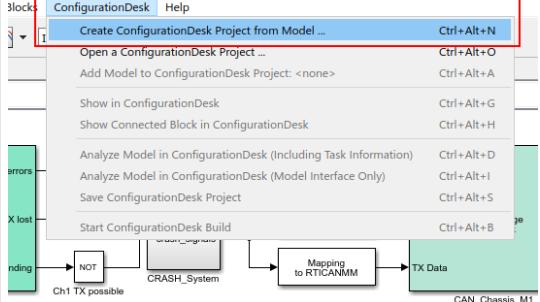
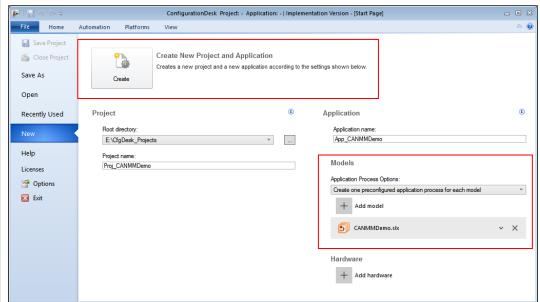
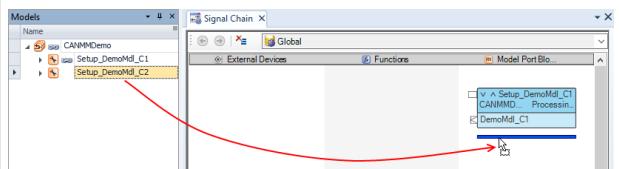
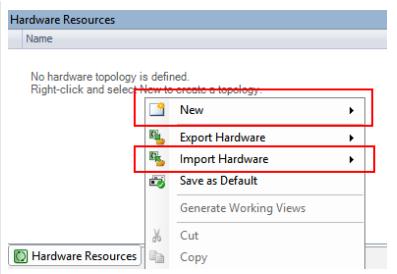
When you model CAN communication in Simulink using the RTI CAN MultiMessage Blockset, it is recommended that you adhere to the following workflow for implementing the CAN communication in ConfigurationDesk. This can simplify your work and prevent conflicts.

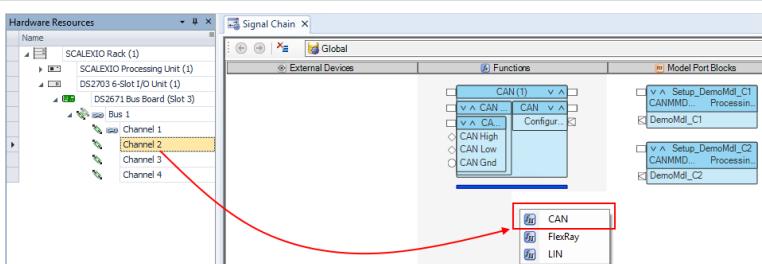
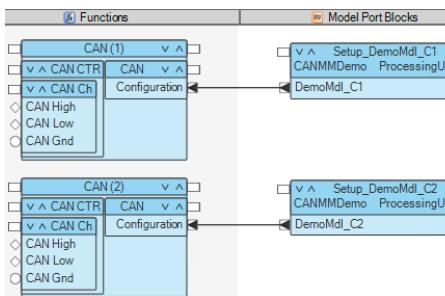
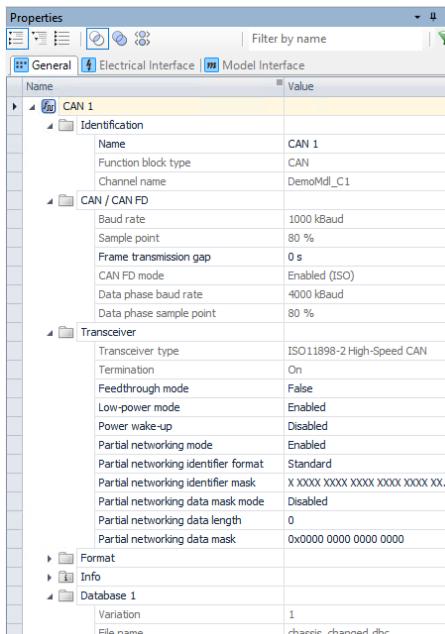
For basic information and required preconditions, refer to [Basics on Implementing CAN Communication](#) on page 336.

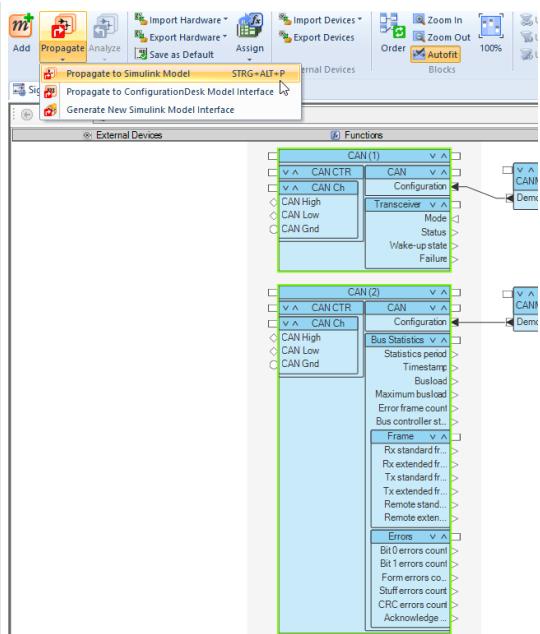
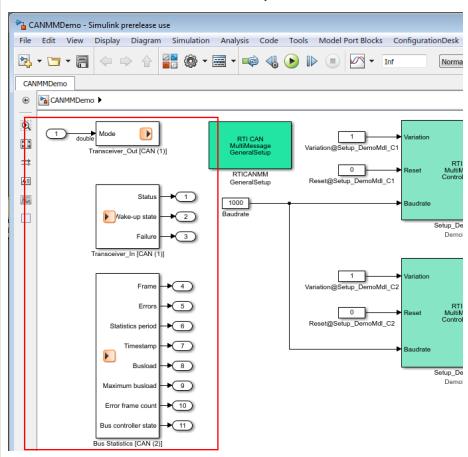
---

**Workflow**

The following table shows detailed workflow steps and indicates the corresponding chapters, which give you detailed information (basic information, instructions etc.).

Step	Work	Refer to ...
1	<p>1 Add the Simulink model to a new ConfigurationDesk project.</p> <p>1 Create a ConfigurationDesk project from the Simulink Editor.</p> 	Remote Access to ConfigurationDesk (Model Interface Package for Simulink - Modeling Guide <a href="#">(link)</a> )
2	<p>2 Add the Simulink model to the ConfigurationDesk application.</p> 	Managing ConfigurationDesk Projects and Applications on page 85
2	<p>2 Drag the Configuration Port blocks from the Model Browser to a working view.</p> 	How to Add Model Port Blocks to the Signal Chain via the Model Browser on page 436
3	<p>3 Create a new or import a hardware topology.</p> 	Basics on Hardware Topologies on page 236
4	<p>4 Add the required number of CAN function blocks to the working view and assign hardware resources to the function blocks. You must use one CAN function block for each Configuration Port block.</p> <p>You can add function blocks by dragging suitable channel types to the working view. When you do this, you automatically assign hardware resources to the function blocks.</p>	<ul style="list-style-type: none"> <li>▪ Basics on Instantiating Function Blocks on page 292</li> <li>▪ Assigning Hardware Resources to Function Blocks on page 399</li> </ul>

Step	Work	Refer to ...
		<ul style="list-style-type: none"> <li>■ <a href="#">SCALEXIO Hardware Dependencies (CAN) (ConfigurationDesk I/O Function Implementation Guide)</a></li> </ul>
5	<p>Map the Configuration port of each CAN function block to a configuration port of a Configuration Port block manually.</p> 	<a href="#">Methods for Model Port Mapping</a> on page 438
6	<p>Configure the CAN function blocks according to your requirements.</p> 	<ul style="list-style-type: none"> <li>■ <a href="#">Implementing I/O Functionality</a> on page 291</li> <li>■ <a href="#">Configuring the Function Block (CAN) (ConfigurationDesk I/O Function Implementation Guide)</a></li> </ul>
7	<p>If function ports are added to the CAN function blocks in the previous step, map the function ports to model ports.</p>	<a href="#">Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model</a> on page 558

Step	Work	Refer to ...
	<p>If they are available, drag model port blocks from the Model Browser to the working view and map the ports manually. Otherwise, select all CAN function blocks and propagate the changes to the Simulink model.</p> <p><b>Note</b></p> <p>You cannot undo the changes in the Simulink model. To prevent a propagate operation from accidentally changing parts of a Simulink model, you can protect specific subsystems by setting them to read-only in Simulink.</p> 	
8	<p>If you propagated changes to Simulink in the previous step, connect the new blocks in the Simulink model and synchronize the model interfaces.</p> <p>1 Connect the new model port blocks in the Simulink model.</p> 	<p>Introduction to the Model Interface Package for Simulink (Model Interface Package for Simulink - Modeling Guide </p>

Step	Work	Refer to ...
2	Synchronize the Simulink model and the ConfigurationDesk model interface.	<a href="#">Synchronizing the Simulink Model Interface and the ConfigurationDesk Model Interface (Model Interface Package for Simulink - Modeling Guide)</a>
9	Resolve conflicts via the Conflicts Viewer.	<a href="#">Resolving Conflicts on page 677</a>

# Building the Signal Chain for LIN Bus Communication

## Where to go from here

## Information in this section

Basics on Implementing LIN Communication.....	344
Recommended Workflow for Implementing LIN Communication.....	347

## Basics on Implementing LIN Communication

### Introduction

LIN communication can be modeled via the RTI LIN MultiMessage Blockset in a Simulink behavior model. If you do this, you can add the behavior model to the ConfigurationDesk application. In the application, you must use LIN function blocks to build the signal chain and to access real-time hardware.

Alternatively, you can use the Bus Manager in ConfigurationDesk to configure and implement LIN communication in ConfigurationDesk. For basic information and instructions on using the Bus Manager, refer to [ConfigurationDesk Bus Manager Implementation Guide](#).

### Workflows and demo

Implementing LIN communication in ConfigurationDesk is a required step for connecting an ECU to a LIN bus that is simulated by a SCALEXIO system.

- For an overview of all the required steps, refer to [LIN Bus Connection \(SCALEXIO – Hardware and Software Overview\)](#).
- For an overview of a recommended workflow for implementing LIN communication in ConfigurationDesk, refer to [Recommended Workflow for Implementing LIN Communication](#) on page 347.
- For a demo application with an example implementation of LIN communication using the RTI LIN MultiMessage Blockset, refer to [Using the LINMMAppl Application \(ConfigurationDesk Demo Projects\)](#).

### Mandatory components

When you configure LIN communication via the RTI LIN MultiMessage Blockset, the following components are mandatory for implementing the LIN communication in ConfigurationDesk.

Component	Description
Configuration Port blocks	Configuration Port blocks are specific model port blocks. They are mandatory for providing LIN bus settings that are specified via the RTI LIN MultiMessage blockset to ConfigurationDesk. Refer to <a href="#">Configuration Port blocks</a> on page 345.

Component	Description
LIN function blocks	LIN function blocks are mandatory for specifying the access to real-time hardware and configuring hardware-related settings for LIN communication. Refer to <a href="#">LIN function blocks</a> on page 346.
Suitable hardware resources	Hardware resources that are suitable for LIN function blocks are mandatory for accessing real-time hardware. The hardware topology must contain suitable hardware resources with one unused channel for each LIN controller. For an overview of suitable hardware resources and the hardware dependencies, refer to <a href="#">SCALEXIO Hardware Dependencies (LIN)</a> ( <a href="#">ConfigurationDesk I/O Function Implementation Guide</a> ).

## Configuration Port blocks

Configuration Port blocks are specific model port blocks. They can be created only when you analyze a Simulink model in ConfigurationDesk.

### Note

You cannot create Configuration Port blocks by extending the signal chain in ConfigurationDesk.

However, Configuration Port blocks can be created during model analysis only if the following preconditions are fulfilled:

- In the Simulink model, `dsrt.tlc` is specified as the system target file. Refer to [Overview of the RTI LIN MultiMessage Blockset \(RTI LIN MultiMessage Blockset Reference\)](#).
- S-functions are created for all RTILINMM blocks of the Simulink model, e.g., via the [Create S-Function for All LIN Blocks](#) command in the Options menu of the RTILINMM GeneralSetup block.

When the preconditions are fulfilled, ConfigurationDesk creates one Configuration Port block for each identified RTILINMM ControllerSetup block during model analysis. The created Configuration Port blocks are named after the related RTILINMM ControllerSetup blocks.

Each Configuration Port block provides one configuration port that is named after the related LIN controller. Configuration Port blocks and configuration ports are indicated by the following symbols, e.g., in the Model Browser:

Symbol	Element
	Configuration Port block
	Configuration port

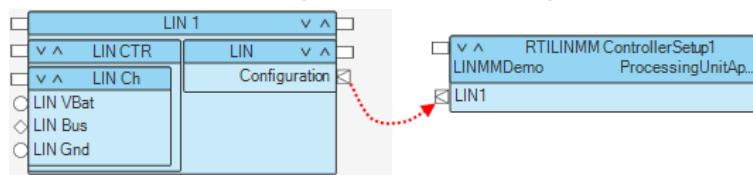
The configuration port must be mapped to a Configuration port of a LIN function block to implement the related LIN bus settings in the signal chain.

#### Note

If you do not map a configuration port of a Configuration Port block to a Configuration port of a LIN function block, the related LIN communication is not implemented in the real-time application when you start the build process.

#### LIN function blocks

**Mapping LIN function blocks to Configuration Port blocks** You can map LIN function blocks to Configuration Port block only manually, i.e., you must manually draw a mapping line between the Configuration port of a LIN function block and the configuration port of a Configuration Port block.



When you do this, settings specified for the related RTILINMM ControllerSetup block in the Simulink model are available for the LIN function block. As a result, some properties of the LIN function block are read-only and/or property values are derived from the RTILINMM ControllerSetup block.

#### Tip

Select the LIN function block to view the LIN communication settings in the Properties Browser.

You can map each LIN function block to only one Configuration Port block that was created for an RTILINMM ControllerSetup block. Therefore, you must use one LIN function block for each Configuration Port block that is relevant for LIN communication.

**Configuring LIN function blocks** To access real-time hardware, you must assign a hardware resource to the LIN function block. Additionally, and depending on the assigned hardware, you can specify additional hardware-related settings via the LIN function block, such as enabling power wake-up or failure simulation.

For more information on the LIN function block and its configurable settings, refer to [LIN \(ConfigurationDesk I/O Function Implementation Guide\)](#).

#### LIN tasks

For each LIN frame that is specified as a TX frame in the Simulink model, one LIN task is available in ConfigurationDesk after model analysis. You can access and configure the tasks in the Task Configuration table, for example.

**Tip**

The preconfigured settings of the LIN tasks are sufficient in most use scenarios. Therefore, you usually do not have to make any manual changes.

For basic information on tasks, refer to [Basics on Tasks, Events, and Runnable Functions](#) on page 480. For information on configuring tasks, refer to [Configuring Tasks in ConfigurationDesk](#) on page 513.

**Specifying the priority of LIN tasks** You can prioritize LIN tasks over other tasks of the real-time application via the priority of the tasks. LIN tasks have a default priority of 9. For each LIN task, you can specify the priority in the range 1 ... 127. A low value means a high priority.

**Note**

The response time of a dSPACE system increases if higher-priority tasks delay the sending of the response after a LIN header is received. Depending on the baud rate and turnaround time specified for higher-priority tasks, the response space (which separates the header and the response of a message) might be increased so much that the length of a complete frame exceeds the time allowed to transmit a message frame ( $T_{FRAME\_MAX}$ ).

For more information on  $T_{FRAME\_MAX}$ , refer to [LIN specification 1.2](#).

**Related topics****Basics**

<a href="#">LIN (ConfigurationDesk I/O Function Implementation Guide</a>		201
<a href="#">Managing Real-Time Hardware</a>		

## Recommended Workflow for Implementing LIN Communication

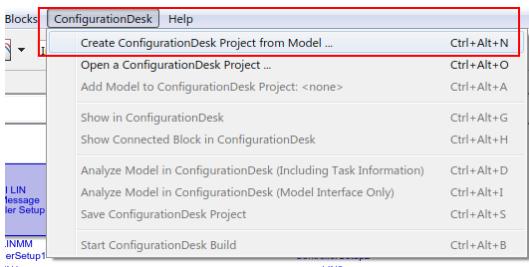
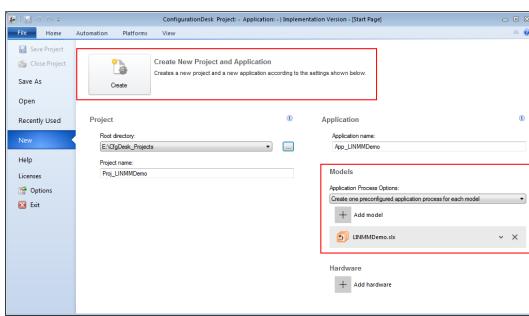
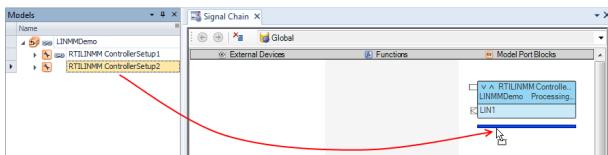
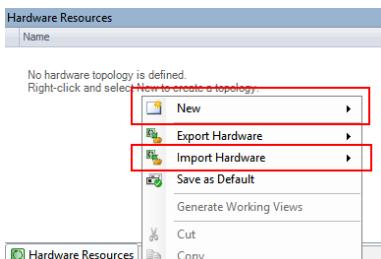
**Introduction**

When you model LIN communication in Simulink using the RTI LIN MultiMessage Blockset, it is recommended that you adhere to the following workflow for implementing the LIN communication in ConfigurationDesk. This can simplify your work and prevent conflicts.

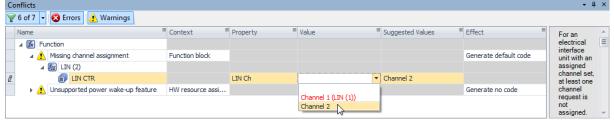
For basic information and required preconditions, refer to [Basics on Implementing LIN Communication](#) on page 344.

**Workflow**

The following table shows detailed workflow steps and indicates the corresponding chapters, which give you detailed information (basic information, instructions etc.).

Step	Work	Refer to ...
1	<p>Add the Simulink model to a new ConfigurationDesk project.</p> <p>1 Create a ConfigurationDesk project from the Simulink Editor.</p> 	Remote Access to ConfigurationDesk (Model Interface Package for Simulink - Modeling Guide <a href="#">(link)</a> )
2	<p>Add the Simulink model to the ConfigurationDesk application.</p> 	Managing ConfigurationDesk Projects and Applications on page 85
2	<p>Drag the Configuration Port blocks from the Model Browser to a working view.</p> 	How to Add Model Port Blocks to the Signal Chain via the Model Browser on page 436
3	<p>Create a new or import a hardware topology.</p> 	Basics on Hardware Topologies on page 236
4	<p>Add the required number of LIN function blocks to the working view and assign hardware resources to the function blocks. You must use one LIN function block for each Configuration Port block.</p> <p>You can add function blocks by dragging suitable channel types to the working view. When you do this, you automatically assign hardware resources to the function blocks.</p>	<ul style="list-style-type: none"> <li>▪ Basics on Instantiating Function Blocks on page 292</li> <li>▪ Assigning Hardware Resources to Function Blocks on page 399</li> <li>▪ SCALEXIO Hardware Dependencies (LIN) (ConfigurationDesk I/O</li> </ul>

Step	Work	Refer to ...
	<p>The screenshot shows the Hardware Resources window on the left and the Signal Chain window on the right. In the Hardware Resources window, a DS2671 Bus Board (Slot 3) is selected, showing its internal structure with Bus 1, Channel 1, Channel 2, Channel 3, and Channel 4. A red arrow points from the 'Channel 2' node in the hardware resources to the 'Configuration' port of the 'LIN1' function block in the Signal Chain window. The Signal Chain window displays various function blocks like LIN (1), LIN (2), RTILINMM ControllerS..., and LINMDemo Processing blocks.</p>	<a href="#">Function Implementation Guide</a> ( )
5	<p>Map the Configuration port of each LIN function block to a configuration port of a Configuration Port block manually.</p> <p>The screenshot shows the Signal Chain window with two LIN function blocks, LIN1 and LIN2. Each has its 'Configuration' port connected to a corresponding 'Configuration' port of a 'RTILINMM ControllerS...' model port block. This visualizes the manual mapping process described in the step.</p>	<a href="#">Methods for Model Port Mapping on page 438</a>
6	<p>Configure the LIN function blocks according to your requirements.</p> <p>The screenshot shows the Properties window for the 'LIN (1)' function block. Under the 'Identification' tab, the name is set to 'LIN (1)'. Under the 'LIN' tab, the 'Baud rate' is set to '19200 Baud'. Under the 'Transceiver' tab, the 'Transceiver type' is set to 'ISO9141 LIN'. Other settings like 'Termination' and 'Power wake-up' are also visible.</p>	<ul style="list-style-type: none"> <li><a href="#">Implementing I/O Functionality on page 291</a></li> <li><a href="#">Configuring the Function Block (LIN) (ConfigurationDesk I/O Function Implementation Guide)</a> ( )</li> </ul>
7	<p>Configure the LIN tasks according to your requirements.</p> <p>The screenshot shows the Task Configuration window. It lists tasks such as 'LINMDemo', 'RTILINMM TX [Latency max. 50us]', and 'RTILINMM TX [Latency max. 50us]'. The 'RTILINMM TX [Latency max. 50us]' task is highlighted with a red box, showing its configuration details: Priority 9, Real-Time... 39, Periodic Task 1, and other parameters.</p>	<a href="#">Configuring Tasks in ConfigurationDesk on page 513</a>

Step	Work	Refer to ...
8	Resolve conflicts via the Conflicts Viewer. 	<a href="#">Resolving Conflicts</a> on page 677

# Building the Signal Chain for FlexRay Communication

## Where to go from here

## Information in this section

Basics on Implementing FlexRay Communication.....	351
Recommended Workflow for Implementing FlexRay Communication.....	356

## Basics on Implementing FlexRay Communication

### Introduction

FlexRay communication can be modeled via the dSPACE FlexRay Configuration Package in a Simulink behavior model. If you do this, you can add the behavior model to the ConfigurationDesk application. In the application, you must use FlexRay function blocks to build the signal chain and to access real-time hardware.

### Workflows

Implementing FlexRay communication in ConfigurationDesk is a required step for connecting an ECU to a FlexRay bus that is simulated by a SCALEXIO or MicroAutoBox III system.

- For an overview of all the required steps, refer to [Overview of the Workflow \(Model Interface Package for Simulink - Modeling Guide](#) .
- For an overview of a recommended workflow for implementing FlexRay communication in ConfigurationDesk, refer to [Recommended Workflow for Implementing FlexRay Communication](#) on page 356.

### Mandatory components

When you configure FlexRay communication via the dSPACE FlexRay Configuration Package, the following components are mandatory for implementing the FlexRay communication in ConfigurationDesk.

Component	Description
Configuration Port blocks	Configuration Port blocks are specific model port blocks. They are mandatory for providing FlexRay bus settings that are specified via the FlexRay Configuration Package to ConfigurationDesk. Refer to <a href="#">Configuration Port blocks</a> on page 352.
FlexRay function blocks	FlexRay function blocks are mandatory for specifying the access to real-time hardware and configuring hardware-related settings for FlexRay communication. Additionally, FlexRay function blocks provide various features for separately controlling the FlexRay communication for each FlexRay controller and FlexRay channel. Refer to <a href="#">FlexRay function blocks</a> on page 353.

Component	Description
Suitable hardware resources	<p>Hardware resources that are suitable for FlexRay function blocks and that match the hardware configuration in the FlexRay Configuration Tool are mandatory for accessing real-time hardware. The hardware topology must contain suitable hardware resources with one unused channel for each FlexRay controller. If you work with a dual-channel FlexRay configuration, at least two unused hardware channels are required.</p> <ul style="list-style-type: none"> <li>▪ For an overview of suitable hardware resources and the hardware dependencies, refer to <a href="#">Hardware Dependencies (FlexRay) (ConfigurationDesk I/O Function Implementation Guide</a> .</li> <li>▪ For more information on configuring the required hardware in the FlexRay Configuration Tool, refer to <a href="#">How to Configure Hardware (FlexRay Configuration Tool Guide</a> .</li> <li>▪ For more information on dual-channel FlexRay configuration, refer to <a href="#">Dual Channel Configurations (FlexRay Configuration Tool Guide</a> .</li> </ul>

**Configuration Port blocks**

Configuration Port blocks are specific model port blocks. They can be created only when you analyze a Simulink model in ConfigurationDesk.

**Note**

You cannot create Configuration Port blocks by extending the signal chain in ConfigurationDesk.

During model analysis, ConfigurationDesk creates one Configuration Port block for each identified FLEXRAYCONFIG UPDATE block. The created Configuration Port blocks are named after the related FLEXRAYCONFIG UPDATE blocks.

Each Configuration Port block provides one configuration port that is named **FlexRay Model Port (ConfigID <specified configuration ID>)**. The specified configuration ID is derived from the associated FLEXRAYCONFIG UPDATE block in the Simulink model. Configuration Port blocks and configuration ports are indicated by the following symbols, e.g., in the Model Browser:

Symbol	Element
	Configuration Port block
	Configuration port

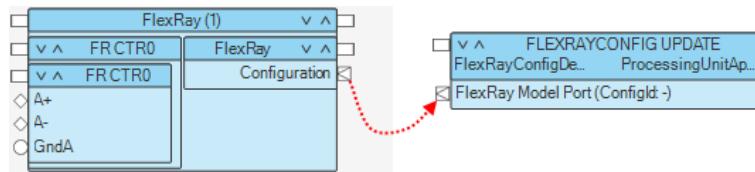
The configuration port must be mapped to a Configuration port of a FlexRay function block to implement the related FlexRay bus settings in the signal chain.

#### Note

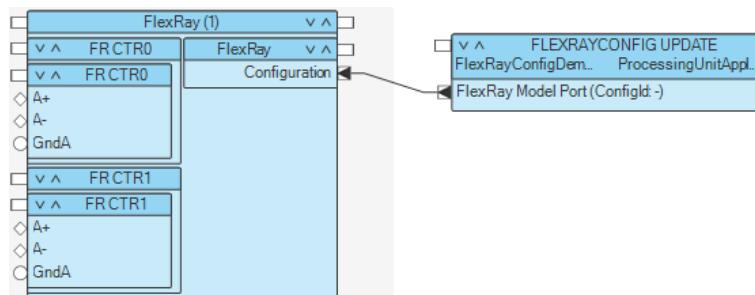
- If you do not map a configuration port of a Configuration Port block to a Configuration port of a FlexRay function block, the related FlexRay communication is not implemented in the real-time application when you start the build process.
- After analyzing the Simulink model, the model topology might contain elements that are reserved, e.g., for synchronizing parts of your Simulink model with the FlexRay bus. When you map Configuration Port blocks to FlexRay function blocks, these elements are deleted from the model topology, even if they are used in the signal chain. To prevent unintended behavior, it is recommended to map the available Configuration Port blocks to FlexRay function blocks before you use other elements of the model topology in the signal chain.

#### FlexRay function blocks

**Mapping FlexRay function blocks to Configuration Port blocks** You can map FlexRay function blocks to Configuration Port block only manually, i.e., you must manually draw a mapping line between the Configuration port of a FlexRay function block and the configuration port of a Configuration Port block.



When you do this, settings specified for the related FLEXRAYCONFIG UPDATE block in the Simulink model are available for the FlexRay function block. As a result, some properties of the FlexRay function block are read-only and/or property values are derived from the FLEXRAYCONFIG UPDATE block. Depending on the derived settings, signal ports are added to the FlexRay function block, as shown in the following example.



#### Tip

Select the FlexRay function block to view the FlexRay communication settings in the Properties Browser.

You can map each FlexRay function block to only one Configuration Port block. Therefore, you must use one FlexRay function block for each Configuration Port block that is relevant for FlexRay communication.

**Points to note for assigning hardware resources** To access real-time hardware, you must assign suitable hardware resources to the FlexRay function block. When you do this, note the following points:

- If you work with FlexRay channel A and FlexRay channel B of one controller, both channels must be assigned to adjacent hardware channels on the same bus board or module.
- If you use multiple FlexRay controllers for one FlexRay function block, you can assign a hardware resource located on different bus boards or modules to each FlexRay controller.
- If you use one or more DS2671 Bus Boards for FlexRay communication, it is recommended not to connect more than two FlexRay channels to the same FlexRay bus. For more information on using the DS2671 Bus Board for FlexRay communication, refer to [Buses and Serial Interfaces of the DS2671 Bus Board \(SCALEXIO Hardware Installation and Configuration\)](#).

**Configuring FlexRay function blocks** FlexRay function blocks provide various settings that let you configure the function blocks, for example:

- Hardware-related settings, such as termination, feedthrough mode, or failure simulation.

#### NOTICE

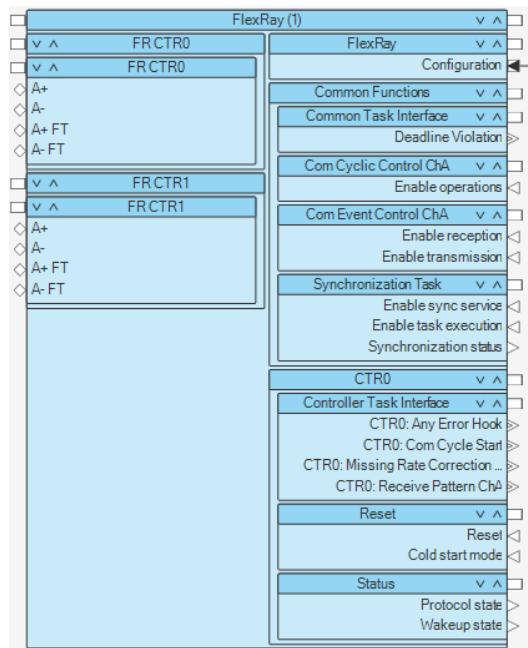
Depending on the real-time hardware, the hardware might be destroyed if the FlexRay bus carries high voltages while feedthrough mode is enabled. Refer to [Configuring the Basic Functionality \(FlexRay\) \(ConfigurationDesk I/O Function Implementation Guide\)](#).

- Settings for configuring the interaction between the real-time application and the FlexRay communication, such as the initial behavior of the FlexRay communication when the real-time application starts or the reaction of the real-time application if a deadline violation (DLV) occurs.
- Settings for configuring FlexRay features, such as controlling the cyclic FlexRay communication, using synchronization tasks, or using events of FlexRay controllers.

#### Tip

Most of the settings depend on the assigned real-time hardware, the used FlexRay channels (A, B), and/or the number of FlexRay controllers that are used by the FlexRay function block. It is therefore useful to assign hardware resources and to map the function block to a Configuration Port block before you configure settings of the function block.

Depending on the configured settings, signal ports, function ports, and/or event ports are added to the FlexRay function block, as shown in the following example.



You can, for example:

- Map signal ports to ports of external devices.
- Map function ports to model ports to exchange data with the Simulink model.
- Map event ports to Runnable Function blocks to use the events for triggering function execution in the Simulink model.

For more information on the FlexRay function block and its configurable settings, refer to [FlexRay \(ConfigurationDesk I/O Function Implementation Guide](#) (book).

## FlexRay tasks

Depending on the FlexRay communication that is configured via the FlexRay Configuration Package, various FlexRay application tasks, communication tasks, and synchronization tasks can be available in the ConfigurationDesk application after model analysis. For information on the FlexRay tasks that can be available, refer to [Basics of Task Creation \(FlexRay Configuration Tool Guide](#) (book).

Additionally, if I/O events of FlexRay function blocks are assigned to tasks, these tasks are relevant for the FlexRay communication as well.

In ConfigurationDesk, you can access and configure the tasks in the Task Configuration table, for example. For basic information on tasks in ConfigurationDesk, refer to [Basics on Tasks, Events, and Runnable Functions](#) on page 480. For information on configuring tasks, refer to [Configuring Tasks in ConfigurationDesk](#) on page 513.

### Tip

The preconfigured settings of the FlexRay tasks are sufficient in most use scenarios. Therefore, you usually do not have to make any manual changes.

**Specifying the priority of FlexRay tasks** You can prioritize FlexRay tasks over other tasks of the real-time application via the priority of the tasks. For each FlexRay task, you can specify the priority in the range 1 ... 127. A low value means a high priority.

If you change the priority of an asynchronous task (i.e., a task to which an I/O event of the FlexRay function block is assigned), you must adjust the priority restrictions of the runnable function that is assigned to the task. You must do this in the ConfigurationDesk model interface and in the Simulink model interface, for example, by propagating the changes to Simulink.

---

#### Related topics

#### Basics

FlexRay (ConfigurationDesk I/O Function Implementation Guide 	201
Managing Real-Time Hardware.....	
Modeling a FlexRay Bus Interface (Model Interface Package for Simulink - Modeling Guide 	

## Recommended Workflow for Implementing FlexRay Communication

---

#### Introduction

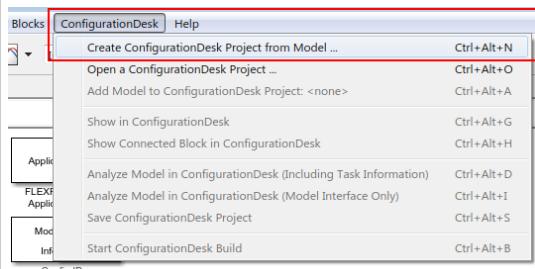
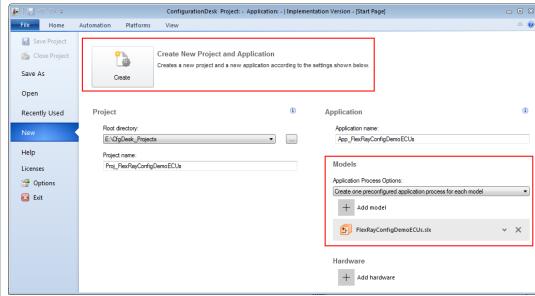
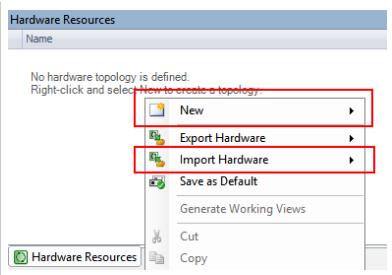
When you model FlexRay communication in Simulink using the dSPACE FlexRay Configuration Package, it is recommended that you adhere to the following workflow for implementing the FlexRay communication in ConfigurationDesk. This can simplify your work and prevent conflicts.

For basic information and required preconditions, refer to [Basics on Implementing FlexRay Communication](#) on page 351.

---

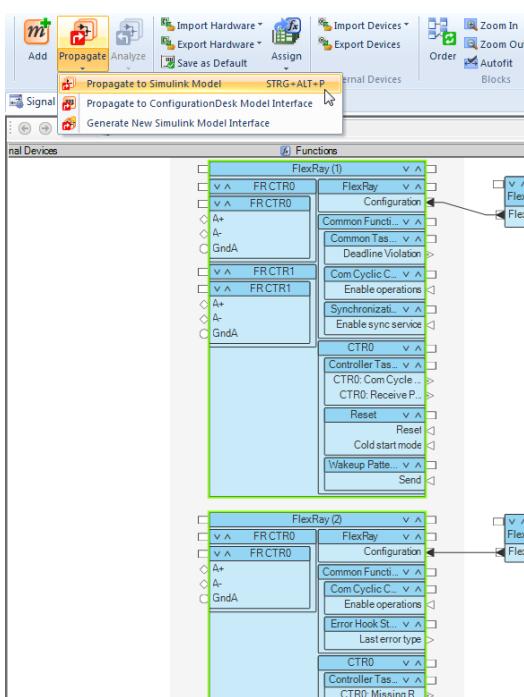
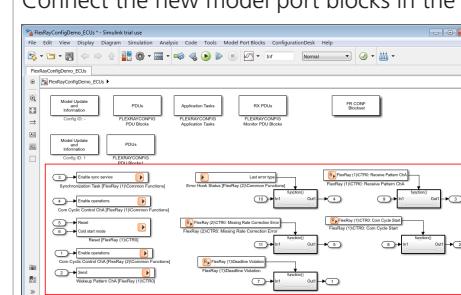
#### Workflow

The following table shows detailed workflow steps and indicates the corresponding chapters, which give you detailed information (basic information, instructions etc.).

Step	Work	Refer to ...
1	<p>1 Add the Simulink model to a new ConfigurationDesk project.</p> <p>1 Create a ConfigurationDesk project from the Simulink Editor.</p> 	<a href="#">Remote Access to ConfigurationDesk (Model Interface Package for Simulink - Modeling Guide)</a>
2	<p>2 Add the Simulink model to the ConfigurationDesk application.</p> 	<a href="#">Managing ConfigurationDesk Projects and Applications on page 85</a>
2	2 Drag the Configuration Port blocks from the Model Browser to a working view.	<a href="#">How to Add Model Port Blocks to the Signal Chain via the Model Browser on page 436</a>
3	<p>3 Create a new or import a hardware topology.</p> 	<a href="#">Basics on Hardware Topologies on page 236</a>
4	<p>4 Add the required number of FlexRay function blocks to the working view and assign hardware resources to the function blocks. You must use one FlexRay function block for each Configuration Port block.</p> <p>You can add function blocks by dragging suitable channel types to the working view. When you do this, you automatically assign hardware resources to the function blocks.</p>	<ul style="list-style-type: none"> <li>▪ <a href="#">Basics on Instantiating Function Blocks on page 292</a></li> <li>▪ <a href="#">Assigning Hardware Resources to Function Blocks on page 399</a></li> <li>▪ <a href="#">Hardware Dependencies (FlexRay) (ConfigurationDesk)</a></li> </ul>

Step	Work	Refer to ...
		I/O Function Implementation Guide (book icon)
5	Map the Configuration port of each FlexRay function block to a configuration port of a Configuration Port block manually.	Methods for Model Port Mapping on page 438
6	If the mapped Configuration Port blocks add further FlexRay controllers and/or channels to the FlexRay function blocks, assign hardware resources to the controllers/channels.	Points to note for assigning hardware resources on page 354
7	Configure the FlexRay function blocks according to your requirements.	<ul style="list-style-type: none"> <li>▪ <a href="#">Implementing I/O Functionality</a> on page 291</li> <li>▪ <a href="#">Configuring the Function Block (FlexRay)</a> (ConfigurationDesk I/O Function Implementation Guide book icon)</li> </ul>

Step	Work	Refer to ...
8	<p>If function ports and/or event ports are added to the FlexRay function blocks in the previous step, map the function ports to model ports and the event ports to Runnable Function blocks, respectively.</p>	<a href="#">Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model on page 558</a>

Step	Work	Refer to ...
	<p>If they are available, drag the required blocks from the Model Browser to the working view and map the ports manually. Otherwise, select all FlexRay function blocks and propagate the changes to the Simulink model.</p> <p><b>Note</b></p> <p>You cannot undo the changes in the Simulink model. To prevent a propagate operation from accidentally changing parts of a Simulink model, you can protect specific subsystems by setting them to read-only in Simulink.</p> 	
9	<p>If you propagated changes to Simulink in the previous step, connect the new blocks in the Simulink model and synchronize the model interfaces.</p> <p>1 Connect the new model port blocks in the Simulink model.</p> 	<p>Introduction to the Model Interface Package for Simulink (Model Interface Package for Simulink - Modeling Guide </p>

Step	Work	Refer to ...
2	Synchronize the Simulink model and the ConfigurationDesk model interface.	<a href="#">Synchronizing the Simulink Model Interface and the ConfigurationDesk Model Interface (Model Interface Package for Simulink - Modeling Guide)</a>
10	Configure the FlexRay tasks according to your requirements.	<a href="#">Configuring Tasks in ConfigurationDesk on page 513</a>
11	If you changed the priority of asynchronous tasks in the previous step, propagate the changes to Simulink.	<a href="#">Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model on page 558</a>
12	Resolve conflicts via the Conflicts Viewer.	<a href="#">Resolving Conflicts on page 677</a>

# Building the Signal Chain for Communication Using an Ethernet Interface

## Where to go from here

## Information in this section

[Implementing an Ethernet Interface.....](#) 362

## Information in other sections

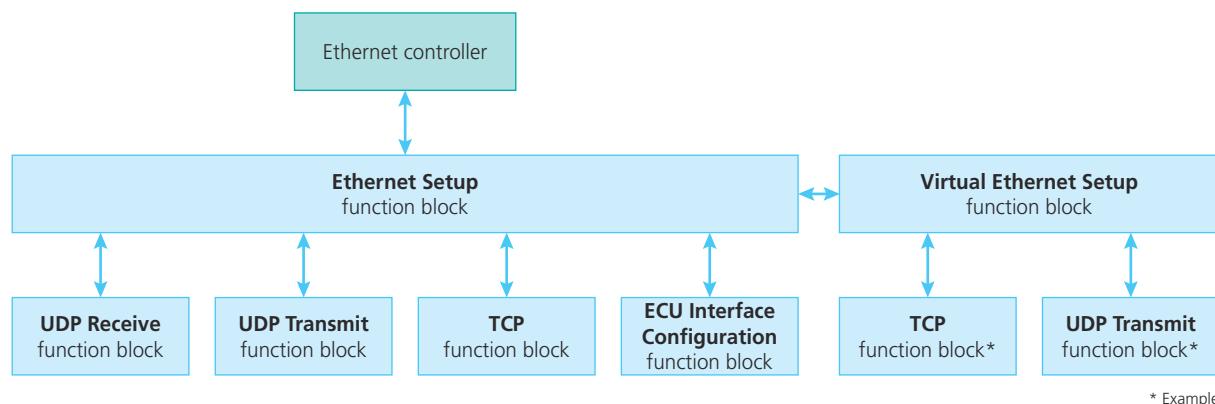
[Configuring the I/O Ethernet Communication.....](#) 225

## Implementing an Ethernet Interface

### Introduction

The Ethernet Setup function block provides access to an Ethernet controller. Several other function blocks can use the access at the same time to transport data values. Furthermore, you can add a virtual Ethernet controller to your network to provide additional IP configurations.

The following illustration gives you an overview of the Ethernet implementation in ConfigurationDesk.



For more information, refer to [Ethernet \(ConfigurationDesk I/O Function Implementation Guide](#)

### Related topics

### Basics

[Ethernet \(ConfigurationDesk I/O Function Implementation Guide](#)

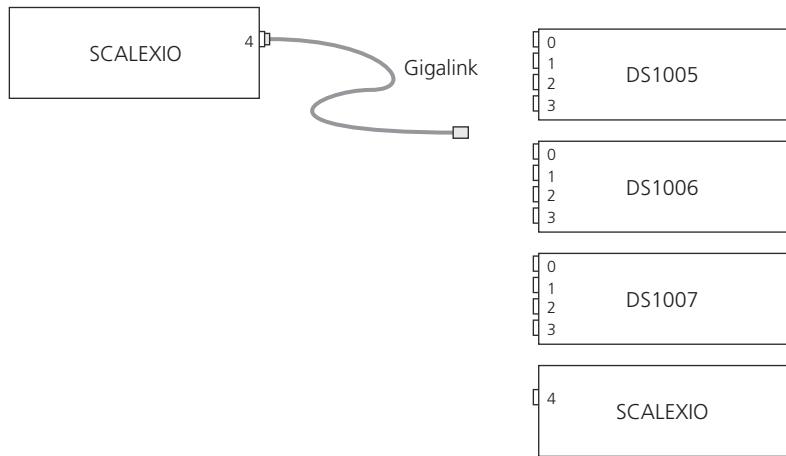
# Building the Signal Chain for Gigalink Communication

**Objective** If you want to send signals to or receive signals from another real-time application running on a SCALEXIO system or PHS-bus system based on DS1005, DS1006 or DS1007, you can use Gigalink communication.

Where to go from here	Information in this section
	<a href="#">Basics on Gigalink Communication.....</a> 363 <a href="#">How to Build the Signal Chain for Gigalink Communication.....</a> 366

## Basics on Gigalink Communication

**Use scenario** A real-time application running on a SCALEXIO system can send signals to and receive signals from another real-time application via Gigalink connection. Gigalink is a fiber-optic connection for bidirectional serial data transmission at high speed (1.25 Gbit/s). You can connect a SCALEXIO system either to a PHS-bus-based system (DS1005, DS1006 or DS1007 modular system) or to another SCALEXIO system as shown in the illustration below.

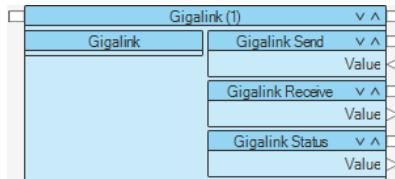


PHS-bus-based systems provide four Gigalink ports (0 ... 3). SCALEXIO Processing Units provide up to eight Gigalink ports (1 ... 8), but only one of them (default: port number four) is available on the system's backplane. DS6001 Processor Boards provide two Gigalink ports available on the front side. To

enable signal transmission, you have to know this port number and configure the port in ConfigurationDesk.

## Implementing Gigalink communication

You can configure a Gigalink port of a SCALEXIO system via ConfigurationDesk by using the Gigalink function block type. For implementation details, refer to [How to Build the Signal Chain for Gigalink Communication](#) on page 366.



## Main characteristics

These are the characteristics of the Gigalink data transmission.

**Gigalink channels** Each Gigalink provides eight independent data channels [0 ... 7]. The transmitter and the corresponding receiver must refer to the same Gigalink channel.

**Maximum signal width** A maximum of 1024 signals can be transmitted on each Gigalink channel as a single vector which is sent in its entirety. Signals can be scalar or vectorial and must be of **Double** data type. If a signal is not of **Double** data type, it is automatically cast to that type.

**Connection status** You can monitor the connection status of the Gigalink via the Gigalink Status function of the Gigalink function block. These are the possible status values:

- 1: Data transmission is possible.
- 0: Data transmission is not possible.

## Unsynchronized communication

By default, data transmission through a Gigalink channel is not synchronized. This means:

- The receiver does not check whether new data is available.
- The data arriving at the receiver overwrites the existing data.  
Note that only a complete new data set overrides the existing data (swinging-buffer method).
- The receiver reads the same data again if no new data has been sent meanwhile.

## Synchronized communication

### Note

Synchronized communication via Gigalink is supported between SCALEXIO systems and between a SCALEXIO system and a PHS-bus-based system. The PHS-bus-based system can only act as an event receiver, while the SCALEXIO systems can be both event receiver and event sender.

ConfigurationDesk supports event-based synchronization of applications running on two or more SCALEXIO systems via Gigalink connection as follows:

- You can add timer events to tasks and send the timer events via the Gigalink connection to another Gigalink member.  
For each timer event, you must enable the **Send** event via Gigalink property, and specify the Gigalink port and channel number (for example, in the **Executable Application** table).  
Note that only timer events can be configured to be sent via Gigalink connection. Up to eight events can be used for each Gigalink port.
- The Gigalink function block can receive events from another Gigalink member. The received events can be used to trigger a task which is executed synchronously on the connected systems.
- In addition, you can specify synchronized reading of data by using the blocking or non-blocking mode of the Gigalink function block:
  - **Blocking:** If no new data is available the receiver waits for the sender to deliver new data. Data consistency and synchronized data transfer are especially important in control applications where a delay might cause stability problems.

#### Note

Note that the execution of the real-time application waits until the signal provides new values. Then the execution continues. Waiting for new signal values can cause task overrun and loop deadlock situations.

- **Non-blocking:** If no new data is available the receiver reads the data vector of the previous sampling step. As a result there are no waiting times but the signal may be delayed by one sampling.

#### Note

To achieve synchronized communication, you have to ensure that the timer task configured to be sent via Gigalink has no offset. If the offset is used, this can lead to the two applications having different simulation times.

**Use scenario for using Gigalink events** ControlDesk supports synchronized data acquisition across ECUs, and RCP and HIL systems. If you want to use simulation time groups to collect platforms/devices with synchronized simulation times, you have to specify Gigalink events in ConfigurationDesk. Simulation time groups can improve synchronization accuracy. For details, refer to [Using Simulation Time Groups \(ControlDesk Measurement and Recording\)](#).

#### Limitation for multicore and multi-processing-unit applications

In a multicore or a multi-PU application with Gigalink communication all the channels of a specific Gigalink port must be mapped to data ports from model implementations that are assigned to the same application process.

If you map the channels of one Gigalink port to data ports of model implementations that are assigned to different application processes, this error is

detected when the real-time application is executed on the hardware. The real-time application is then terminated and an error message is generated.

Note that ConfigurationDesk does not generate conflicts during signal chain implementation, Gigalink function block configuration, or the build process.

## Related topics

### Basics

[Connecting a PHS-Bus-Based System \(SCALEXIO Hardware Installation and Configuration\)](#)

## How to Build the Signal Chain for Gigalink Communication

### Objective

To send signals to and receive signals from other real-time applications running on SCALEXIO or DS1005/DS1006/DS1007 platforms, you must build the signal chain for Gigalink communication in ConfigurationDesk using the Gigalink function block.

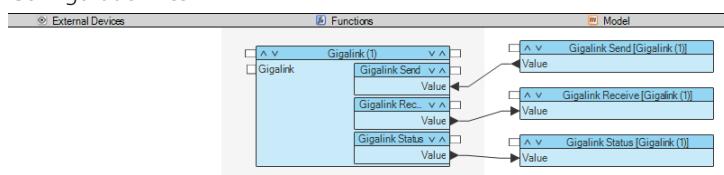
### Preconditions

A working view is open.

### Method

#### To build the signal chain for Gigalink communication

- 1 From the Function Browser, drag a Gigalink function block to the working view.  
ConfigurationDesk adds the instantiated Gigalink function block to the signal chain.
- 2 In the working view, right-click the Gigalink function block and select Extend Signal Chain – Create Suitable Model Port Block from the context menu.  
ConfigurationDesk adds model port blocks to the signal chain. The following illustration shows an example of a signal chain for Gigalink data transfer in ConfigurationDesk:



- 3 In the working view, select the logical signal representation/electrical interface unit of the Gigalink function block.

- 4** The properties of the electrical interface unit are displayed in the Properties Browser:

- SCALEXIO Processing Unit: Set the Gigalink number to 4.

Internally, a SCALEXIO Processing Unit provides up to eight IOCNET connectors, but only one of them (default: port number four) is available at the back of the SCALEXIO rack.

DS6001 Processor Board: Set the Gigalink number to 1 or 2.

DS6001 Processor Boards have two IOCNET ports that can be used for Gigalink connections. The port numbers are numbered 19 and 20 but they must refer to 1 and 2.

- Select the Channel number.

A Gigalink provides eight data channels [0 ... 7]. The transmitter and the corresponding receiver must refer to the same channel number. Each channel can be assigned only once, i.e., channels that are already assigned to a port no longer appear in the list.

In multimodel applications all the channels of a specific Gigalink port must be mapped to only one behavior model to avoid errors. For details, refer to [Basics on Gigalink Communication](#) on page 363.

- 5** Select the Gigalink Send function:

- Specify the Send data width. This is the number of signals to be transferred through the data channel. The data width specified for the function inport in ConfigurationDesk must equal the data width specified for the Data Outport block in the behavior model.

- 6** Select the Gigalink Receive function:

- Specify the Receive data width. This is the number of signals to be transferred through the data channel. The data width specified for the function outport in ConfigurationDesk must equal the data width specified for the Data Import block in the behavior model.

#### Note

A mismatch between data widths is indicated by an orange mapping line if highlighting of signal chain elements is enabled.

- 7** Specify synchronized communication between Gigalink members:

- Specify the update mode for reading new data via the Protocol property. The protocol affects the data update as follows:
  - **Blocking:** If no new data is available the receiver waits for the sender to deliver new data. Data consistency and synchronized data transfer are especially important in control applications where a delay might cause stability problems.

- Non-blocking: If no new data is available the receiver reads the data vector of the previous sampling step. As a result there are no waiting times but the signal may be delayed by one sampling.
  - Add timer events to be sent to another Gigalink member: Add a timer event to a task (for example, in the Executable Application table). For the timer event, enable the Send event via Gigalink property, and specify the Gigalink port and the channel number.
  - Specify Event reception. If you enable event reception, the function block receives events from a Gigalink member. They can be used to trigger a task which is executed synchronously on the connected systems.  
For basics on synchronized communication, refer to [Basics on Gigalink Communication](#) on page 363.
- 8 Repeat steps 1 ... 7 if you want to use further Gigalink data channels. Each combination of the Gigalink number and channel number can be used only once.

---

**Result** You have built the logical signal chain to use your real-time application for Gigalink data transmission.

---

**Next steps**

- To make the model port blocks available in the behavior model, you must first generate an interface model. For basic instructions, refer to [How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model](#) on page 543. Then you must copy the generated model port blocks from the interface model and paste them in your behavior model using the Paste and Keep IDs command from the Model Port Blocks menu.
- If you connect two SCALEXIO systems, you have to change the IOCNET Link Board Configuration in the SCALEXIO System Configuration home page from IOCNET/GIGALINK (auto-detect) to GIGALINK (fixed) for both SCALEXIO systems. For details, refer to [How to Connect Another SCALEXIO System via Gigalink \(SCALEXIO Hardware Installation and Configuration\)](#).

---

**Related topics**

Basics

[Gigalink \(ConfigurationDesk I/O Function Implementation Guide\)](#)

References

[Gigalink \(ConfigurationDesk Function Block Properties\)](#)

# Adding FPGA Applications to the Signal Chain

## Objective

The custom-coded FPGA application that you want to use in ConfigurationDesk has to be generated with the RTI FPGA Programming Blockset. Then, you add the FPGA application to ConfigurationDesk as FPGA custom function blocks.

## Where to go from here

### Information in this section

[Steps to Implement an FPGA Application](#)..... 370

### Information in other sections

Designing and building an FPGA application:

#### [RTI FPGA Programming Blockset Guide](#)

Provides basic information and detailed instructions on working with the RTI FPGA Programming Blockset.

#### [RTI FPGA Programming Blockset Handcode Interface Guide](#)

Provides basic information and detailed instructions on handcoding HDL and Verilog code for use with dSPACE hardware.

Implementing and handling an FPGA custom function block:

#### [Implementing FPGA Custom Function Blocks \(ConfigurationDesk I/O Function Implementation Guide\)](#)

Supported hardware:

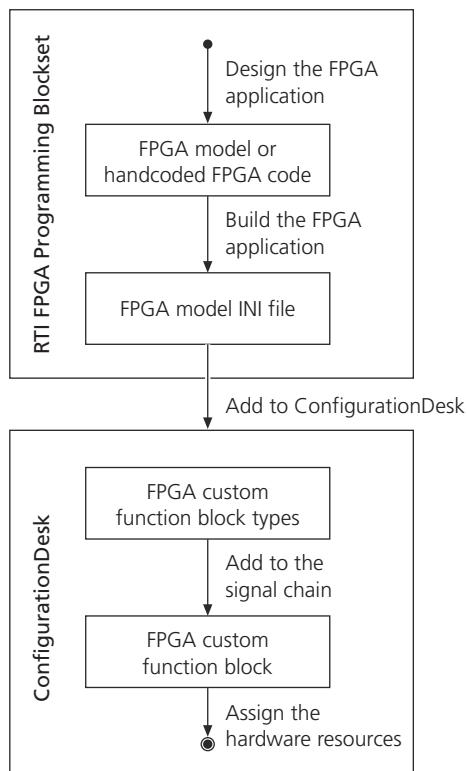
#### [Hardware for FPGA Applications \(SCALEXIO Hardware Installation and Configuration\)](#)

To execute FPGA applications, a SCALEXIO system must have an FPGA base board with I/O modules.

DS1514 FPGA Base Board Data Sheet (MicroAutoBox III)  
Hardware Installation and Configuration [\(PDF\)](#)

## Steps to Implement an FPGA Application

### Implementation overview



### Designing FPGA applications

There are two methods to design FPGA applications:

- Modeling FPGA applications in Simulink:

The RTI FPGA Programming Blockset provides blocks to model the interface to the external devices and the interface to the behavior model. The I/O functionality itself must be modeled with Xilinx® blocks. Refer to [Modeling the FPGA Application \(RTI FPGA Programming Blockset Guide\)](#).

- Handcoding FPGA applications:

The RTI FPGA Programming Blockset provides templates to handcode the interface to the external devices and the interface to the behavior model. The I/O functionality itself must be designed with VHDL code. Refer to [Detailed Instructions on the Handcode Workflow \(RTI FPGA Programming Blockset Handcode Interface Guide\)](#).

FPGA applications for the following use cases are supported in ConfigurationDesk:

- FPGA applications supporting single-core real-time applications.
- FPGA applications supporting multicore real-time applications.
- FPGA applications supporting inter-FPGA communication.

For more information on how the specific FPGA application types are represented by function blocks, refer to [Types of FPGA Applications \(ConfigurationDesk I/O Function Implementation Guide\)](#).

#### **Building FPGA applications**

You start the build process via the RTI FPGA Programming Blockset. The build results consist of the generated FPGA bitstream to program the FPGA and files that are needed to add the FPGA application to ConfigurationDesk. The required build result files are archived in the FPGA model INI file.

**Adding port-specific functions to the function ports** The FPGA model INI file also provides templates for user functions that can be used in ConfigurationDesk to trigger the execution of port-specific functions, for example, the conversion of values or other computations. For more information, refer to [How to Build FPGA Applications \(MicroAutoBox III, SCALEXIO\) \(RTI FPGA Programming Blockset Guide\)](#).

#### **Adding FPGA applications to ConfigurationDesk**

You can export the FPGA applications to ConfigurationDesk or you import the FPGA model INI file including the FPGA application. Added FPGA applications are available as FPGA custom function block types in the Function Browser.

Depending on the selected file directory to add the FPGA application, the FPGA custom function block types are available in the Function Browser of one ConfigurationDesk project or in all ConfigurationDesk projects.

For more information on the file directory to add FPGA applications and instructions on adding, refer to [Handling FPGA Custom Function Blocks in ConfigurationDesk \(ConfigurationDesk I/O Function Implementation Guide\)](#).

#### **Adding FPGA custom function blocks to the signal chain**

You can add FPGA custom function blocks to the signal chain, for example, via drag & drop from the Function Browser to a working view.

#### **Assigning the hardware resources**

With the RTI FPGA Programming Blockset, you design an FPGA application for a certain arrangement of I/O modules that are inserted to the I/O module slots of the FPGA board. Therefore, you can assign the hardware resources only if the connected I/O modules matches by module type and I/O module slot. Refer to [Hardware Dependencies of FPGA Custom Function Blocks \(ConfigurationDesk I/O Function Implementation Guide\)](#).

#### **Loading the bitstream to the FPGA board**

The build process of ConfigurationDesk adds the FPGA bitstream from the FPGA model INI file to the real-time application. When you download the real-time

application to the dSPACE hardware, the FPGA bitstream is automatically loaded to the FPGA base board that is assigned by the FPGA custom function block.

# Specifying Failure Simulation in ConfigurationDesk

## Objective

By using a failure simulation system, you can simulate failures in the cable harness of your ECU. To control the failure simulation with your experiment software, you have to specify the allowed failure classes in ConfigurationDesk.

## Where to go from here

### Information in this section

Introduction to Failure Simulation.....	373
Specifying Failure Simulation Settings.....	374
Load Rejection and Signal Dropping.....	378
Providing the Specified Data to Experiment Software.....	381

## Introduction to Failure Simulation

### Definitions

Failure simulation means inserting electrical failures in the wiring of your ECU to analyze the behavior of the system when wiring problems occur. For example, you can simulate situations in which an ECU pin is short-circuited to ground or the battery voltage, or an ECU pin is not connected (cable break, open circuit).

### Required components

To use failure simulation with the SCALEXIO system, it has to be equipped, for example, with a DS2680 I/O Unit or a DS2642 FIU & Power Switch Board. These boards provide a central failure insertion unit (FIU). The central FIU switches the signal from the ECU pin to simulate the failure.

For controlling failure simulation, your experiment software has to be extended with a Failure Simulation module, which lets you set and control the failures in a graphical environment.

## Workflow for using failure simulation

Using the failure simulation feature is performed in several steps. One of them is specifying the failure simulation support in ConfigurationDesk.

For the entire workflow and an overview on using failure simulation with the SCALEXIO system, refer to [Basics of Electrical Errors Simulation in a SCALEXIO System \(SCALEXIO – Hardware and Software Overview\)](#).

# Specifying Failure Simulation Settings

---

## Objective

To control failure simulation via experiment software, you must apply failure simulation settings in ConfigurationDesk before providing the data to the experiment software. You must specify the failure simulation settings for each signal separately. In the case of channel multiplication, the behavior of the combined channels is equivalent to the behavior of a single channel.

### Note

The settings for failure simulation must be derived from the electrical characteristics of your external device (e.g., your ECU). Make sure that the settings cannot lead to damage to your ECU.

---

## Required settings

You must apply the following settings before providing failure simulation data to your experiment software:

**Allowed failure classes** Each kind of failure which can be simulated is defined as a failure class. The failure classes that are allowed to be simulated must be configured at device ports and signal ports in ConfigurationDesk. These failure classes can then be simulated via the experiment software.

The following failure classes are provided by the SCALEXIO system and are supported by the software tools involved:

- Open circuit
- Short to GND
- Short to VBAT
- Short to signal measurement channel
- Short to signal generation channel
- Short to bus channel

**Load rejection or bus signal dropping (optional)** You can use load rejection or bus signal dropping to protect hardware such as sensitive loads or bus transceivers against damage during failure simulation. Enforcing load rejection or bus signal dropping for a signal in ConfigurationDesk means that you cannot use failure simulation without load rejection or bus signal dropping for that signal in the experiment software.

For details and restrictions on load rejection and signal dropping, see [Load Rejection and Signal Dropping](#) on page 378.

**Multiple failure support (optional)** Using the Failure Simulation module of your experiment software, you can simulate failures for multiple signals simultaneously. For example, you can simulate a situation where a short to GND for one signal and an open circuit for a different signal happen at the same time. Multiple failures are not restricted to the same unit/box: You can simulate multiple failures simultaneously for signals assigned to channels on different units/boxes or even from different SCALEXIO racks.

As a precondition, the Activation by FRU relay property must be set to Allowed in ConfigurationDesk at all signal ports that are part of a multiple failure scenario. This way, the SCALEXIO system can use the electromechanical relays of the failure routing unit (FRU) to activate the failure directly by switching the relays under load instead of just switching the signal to the failplane. For details on failure simulation with a SCALEXIO system, refer to [Hardware for Electrical Error Simulation on SCALEXIO Systems \(SCALEXIO – Hardware and Software Overview\)](#).

#### NOTICE

##### Risk of increased wear and permanent damage

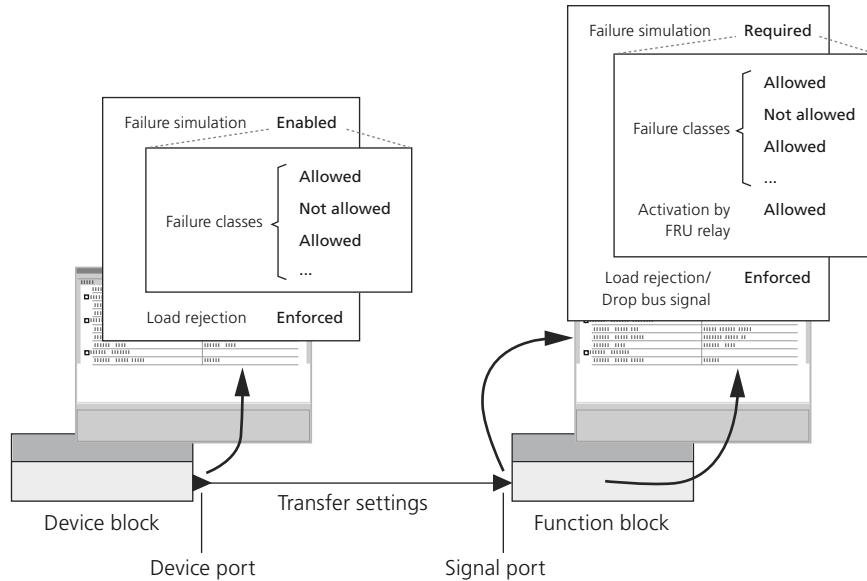
When you use the “Activation by FRU relay” feature, there is a risk of increased wear and permanent damage to the relays of the dSPACE failure simulation hardware. For details, refer to [Safety Precautions for Simulating Electrical Errors with a SCALEXIO System \(SCALEXIO – Hardware and Software Overview\)](#).

---

#### Specifying the failure simulation settings via Properties Browser

Before you can specify the failure simulation settings, you must enable failure simulation support for the device port(s) and set the failure simulation to Required at the electrical interface of the function block.

You can specify the failure simulation settings in the signal chain at device ports and at signal ports (of function blocks) as shown below:



- You have to allow failure classes at each port of a device block and at each signal port of a function block.
- You can enforce load rejection at each port of a Load-type device block and at each signal port of a signal measurement function block. At signal ports of bus function blocks, you can enforce bus signal dropping .
- You can support the simulation of multiple simultaneous failures by setting the Activation by FRU relay property to Allowed at each signal port of a function block.

#### Transfer settings

To simplify configuration work, you can transfer the settings for the allowed failure classes from a device port to the mapped signal port of the function block. If a signal port is mapped multiple times by several device ports, a common subset of settings is found and the result is transferred to the signal port.

For instructions, refer to [How to Transfer Port Settings from Device Ports to Signal Ports](#) on page 317.

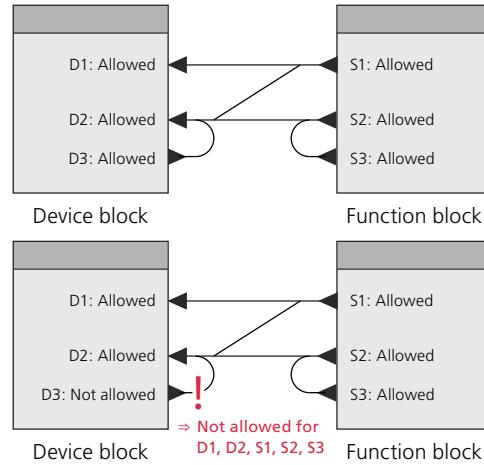
#### Mapping rules

Mappings between device ports and signal ports have different effects on specific failure simulation settings.

**Allowed failure classes** For each signal port, the failure class setting is determined by comparing the settings for all mapped device and signal ports during the build process.

If one mapped device or signal port setting is set to Not Allowed, it overrides all device or signal port settings currently set to Allowed. This means that:

- If a signal port is mapped to a single device port, a failure class has to be set to Allowed at both ports to be available in the experiment software.
- In order to allow a failure class for a signal port with multiple mappings, it has to be set to Allowed at the signal port itself and at each device and signal port which is mapped to it. This applies to direct and indirect mappings as shown in the illustrations below.



#### Tip

A conflict is shown in the **Conflicts Viewer** if a failure class is allowed at a signal port but not allowed at a mapped device port.

**Load rejection** For each signal port, the load rejection setting is determined by comparing the settings for all mapped device ports and the signal port itself during the build process.

If one mapped device port or the signal port itself is set to Enforced, load rejection is enforced for the signal in the experiment software.

The load rejection settings of mapped signal ports are ignored. Thus, different load rejection settings can be provided to the experiment software for mapped signal ports.

For details on enforcing load rejection for signal measurement, see [Basics on Load Rejection](#) on page 389.

**Bus signal dropping** Mappings are not considered during the build process.

**Multiple failure support** Mappings are not considered during the build process.

If you use several signal ports for one signal, you must set Activation by FRU relay to Allowed for each signal port that you want to use in a multiple failure scenario.

<b>Specifying the failure simulation settings for device ports via Microsoft Excel™ file</b>	The device port settings can be specified in an XLSX file (Microsoft Excel™), together with the settings for other device port properties. This file can then be imported to a ConfigurationDesk application. For details, refer to <a href="#">How to Export a Device Topology to a Microsoft Excel Sheet</a> on page 267.
<b>Allowing short-circuits between two signals</b>	To simulate short circuits between two signals, you have to define a failure with the signals in the Failure Simulation module of your experiment software. However, you can define a short circuit failure only if the settings of the allowed failure classes (in ConfigurationDesk) support the short circuit.  As a precondition, each signal has to allow the other's failure class. The following example explains this: Signal A is assigned to a signal generation channel. Signal B is assigned to a signal measurement channel. To support short circuit between these two channels, you have to allow "Short to signal measurement channel" for signal A and "Short to signal generation channel" for signal B.  Note that there are restrictions regarding load rejection support for this use case. For details, refer to <a href="#">Load Rejection and Signal Dropping</a> on page 378.
<b>Limitation regarding multiple failures and current enhancement</b>	Channel multiplication for current enhancement (channels connected in parallel) does not support multiple failures. This means that in multiple failure scenarios, you cannot simultaneously switch more than one failure for a signal where channels are multiplied for current enhancement.  For details on using channel multiplication for current enhancement, see <a href="#">Specifying Current and Voltage Values for Channel Multiplication</a> (ConfigurationDesk I/O Function Implementation Guide  ).

## Load Rejection and Signal Dropping

<b>Objective</b>	The SCALEXIO hardware allows you to cut off a signal from loads or other connected devices during failure simulation to protect them against damage.
<b>Different methods to cut off a signal</b>	<ul style="list-style-type: none"> <li>▪ Signal measurement channels can be cut off via load rejection. You can enforce load rejection using the Load rejection property in ConfigurationDesk. For more details on load rejection for external and internal loads connected to signal measurement channels, see <a href="#">Basics on Load Rejection</a> on page 389.</li> <li>▪ Signal generation channels can be cut off via signal generation dropping. You can drop the signal generation to protect it against damage during failure simulation and prevent internal electronic fuses from blowing, which would cause the failure to be instantly deactivated. Dropping the signal generation will also prevent unintended effects in the case of shorts to signal measurement channels.</li> </ul>

Signal generation dropping is exclusively configured in the Failure Simulation module of your experiment software and there is no related ConfigurationDesk property.

- Bus transceiver channels can be cut off via bus signal dropping. This is to protect bus transceivers against damage during failure simulation. You can enforce bus signal dropping using the **Drop bus signal** property in ConfigurationDesk.

### Technical restrictions

Load rejection or signal dropping is not provided for every failure class, due to technical restrictions. The supported failure classes for each kind of I/O channel are described below.

To avoid the restrictions and to configure load rejection or signal generation dropping for all failure classes, you can use the electromechanical relays of the failure routing unit (FRU) to activate failures. As a precondition, the **Activation by FRU relay** property must be set to **Allowed** in ConfigurationDesk at all involved signal ports.

The FRU relays are only used if load rejection is not possible otherwise.

#### NOTICE

##### Risk of increased wear and permanent damage

When you use the “Activation by FRU relay” feature, there is a risk of increased wear and permanent damage to the relays of the dSPACE failure simulation hardware. For details, refer to [Safety Precautions for Simulating Electrical Errors with a SCALEXIO System \(SCALEXIO – Hardware and Software Overview\)](#).

### Support of load rejection for signal measurement channels

The following table shows the failure classes and their load rejection support.

Signal Assigned to ...	Failure Class	Load Rejection Provided
Signal measurement channel	Open circuit	✓
	Short to GND	✓
	Short to VBAT	✓
	Short to signal measurement channel <sup>1)</sup>	–
	Short to signal generation channel <sup>2)</sup>	✓, but the signal generator cannot be

Signal Assigned to ...	Failure Class	Load Rejection Provided
		dropped at the same time
	Short to bus transceiver channel	-, but the bus signal transceiver can be dropped

- <sup>1)</sup> If you configure a short between two signal measurement channels and load rejection is set for at least one of the channels in ConfigurationDesk, ControlDesk will not support this configuration.
- <sup>2)</sup> If you configure a short between a signal measurement channel and a signal generation channel, you can only configure load rejection or signal generation dropping, but not both at the same time.

**Support of signal generation dropping**

The following table shows the failure classes and their signal generation dropping support.

Signal Assigned to ...	Failure Class	Signal Generation Dropping Provided
Signal generation channel	Open circuit	✓
	Short to GND	✓
	Short to VBAT	✓
	Short to signal measurement channel <sup>1)</sup>	✓, but the load on the signal measurement channel cannot be rejected at the same time
	Short to signal generation channel	-
	Short to bus transceiver channel	-, but the bus signal transceiver can be dropped

- <sup>1)</sup> If you configure a short between a signal measurement channel and a signal generation channel, you can only configure load rejection or signal generation dropping, but not both at the same time.

**Support of bus signal dropping**

The following table shows the failure classes and their support of bus signal dropping.

Bus signal dropping is provided for every failure class, but there are some technical restrictions concerning shorts to signal measurement and signal generation channels.

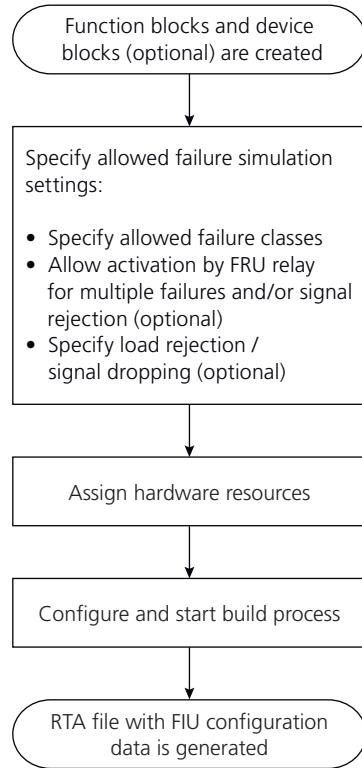
Signal Assigned to ...	Failure Class	Bus Signal Dropping Provided
Bus transceiver channel	Open circuit	✓
	Short to GND	✓
	Short to VBAT	✓
	Short to signal measurement channel <sup>1)</sup>	✓, but the load on the signal measurement channel cannot be rejected
	Short to signal generation channel	✓, but the signal generation cannot be dropped
	Short to bus transceiver channel	✓

<sup>1)</sup> If you configure a short to a signal measurement channel and load rejection is enforced for the signal measurement channel in ConfigurationDesk, ControlDesk will not support this configuration.

## Providing the Specified Data to Experiment Software

**Objective** During the build process, ConfigurationDesk provides the data required for failure simulation to your experiment software.

**Providing data** The following illustration shows the steps which are necessary to specify and provide the complete failure simulation data required in your experiment software.



The FIU configuration data contains information about the signal and device ports, their allowed failure classes, activation by FRU relay (allowed, not allowed), the load rejection behavior (enforced, not enforced), and the assigned hardware resources.

Depending on the kind of device port mapping, ConfigurationDesk uses two different ways to identify a signal in the FIU configuration data:

- *Via port group address*

If a signal port of a function block is mapped to exactly one device port, the device port's port group address is used:

`<device name>\<port group name (layer 1)>\ ... \<port group name (layer n)>\<port name>`

- *Via signal port name*

This method is used in all other cases, i.e.:

- If multiple signal ports are mapped to one device port
- If one signal port is mapped to multiple device ports
- If a signal port is not mapped to any device port at all
- If duplicate names within the device topology would lead to the same identifier

To avoid ambiguous entries in these cases, the names of the signal port and the function block it resides in are used:

`<function block name>\<signal port name>`

The FIU configuration data is added to the real-time application (RTA) file which is generated during the build process.

The RTA file is the executable object file. After the build process the RTA file can be downloaded to the real-time hardware.

**Tip**

- The FIU configuration data is specified and built independently of the behavior model.
- No device topology is required in your ConfigurationDesk application.

---

**Effects on changing**

If you change settings after building a real-time application, you have to start the build process again and download the real-time application again to make the changes available in your experiment software.

---

**Related topics****Basics**

Assigning Hardware Resources to Function Blocks.....	399
Building Real-Time Applications.....	697



# Handling Loads

## Objective

To measure actuator signals, you have to use electrical loads such as resistors or real actuators such as electric motors. These loads can be represented in your ConfigurationDesk application.

## Where to go from here

## Information in this section

Basics on Using Internal or External Loads.....	385
Basics on Load Rejection.....	389
Details on Handling Internal Loads.....	392
Details on Handling External Loads.....	395
How to Provide Data of Internal Loads to the Hardware.....	397

## Basics on Using Internal or External Loads

## Objective

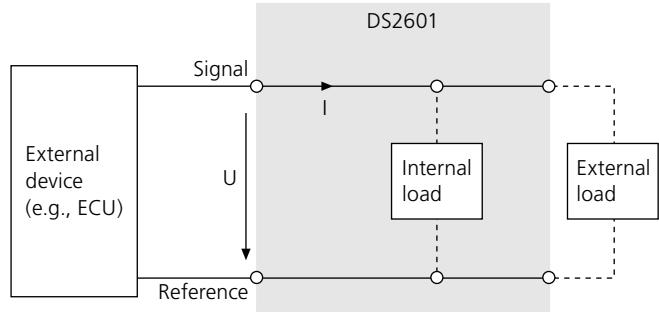
ConfigurationDesk distinguishes between internal loads and external loads.

## Definition of internal and external loads

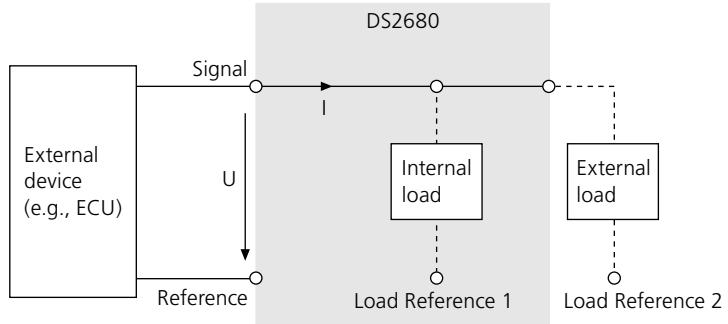
Internal loads are loads which are mounted directly on the dSPACE real-time hardware. External loads are connected to the dSPACE real-time hardware via I/O connector.

Loads can only be used in combination with boards which contain signal measurement channels, such as the DS2601, the DS2680 or the DS2690.

The following illustrations show simplified I/O circuits and the electrical connection of internal and external loads for the real-time hardware mentioned above.

**DS2601 Signal Measurement Board**

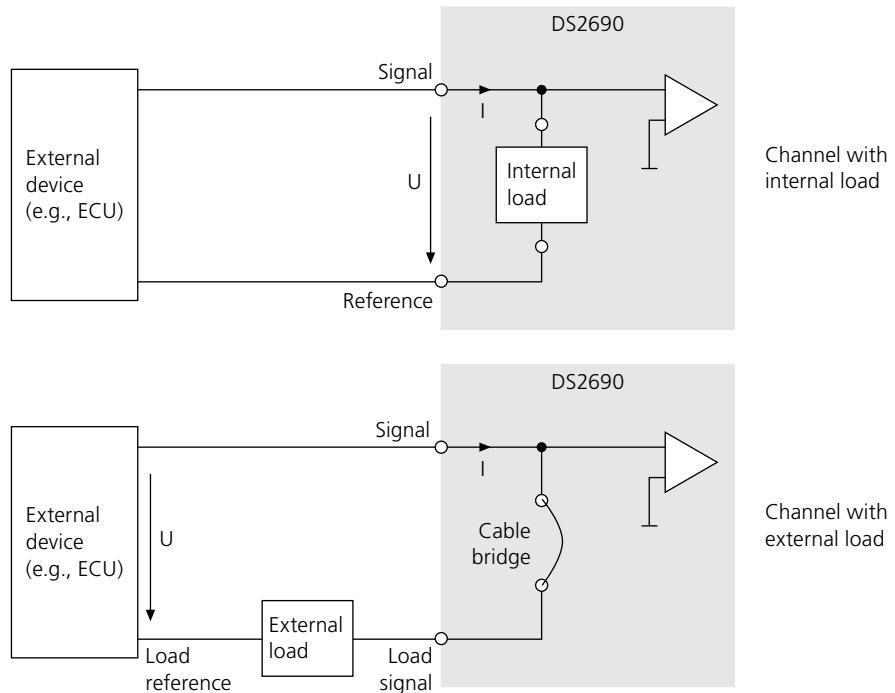
As shown in the illustration above, internal and external loads can be used in parallel.

**DS2680 I/O Module**

The simplified I/O circuit above shows that, in principle, internal and external loads can be used in parallel.

Note that the actual I/O circuit depends on the channel type used on the DS2680. For detailed circuit diagrams refer to [Circuit Diagrams of Channel Types \(ConfigurationDesk I/O Function Implementation Guide\)](#).

### DS2690 Digital I/O Board



As shown in the illustration above, you can use either an internal load or an external load here.

### Deciding which to use

The following points will help you decide whether you want to use an internal or an external load:

- Internal loads are limited to a lower max. input power than external loads. On the DS2601 Signal Measurement Board, for example, a max. internal load of 2 W is allowed for every channel.
- Internal loads need no internal or external wiring. They are inserted directly into load sockets, which are mounted on the real-time hardware.
- Valid only for DS2601 and DS2680: As shown in the I/O circuit above, it is possible to use internal loads and external loads together for the same channel.

However, this is useful only in specific cases. For example, you can use an inductance as the external load and at the same time a freewheeling diode as an internal load to protect the FIU (Failure Insertion Unit) of your real-time hardware.

- If you specify a max. current at a signal port (in the function block), which is higher than that provided by a single channel, multiple channels are required. You should always use an external load in this case.

The example of the DS2601 Signal Measurement Board explains why: The max. permitted load power consumption on the DS2601 is 2 W. If the current specified for a signal exceeds 10 A, channels are combined. However, there is no useful load with  $0.02 \Omega$  (and more) which provides less than 2 W at a current flow of 10 A.

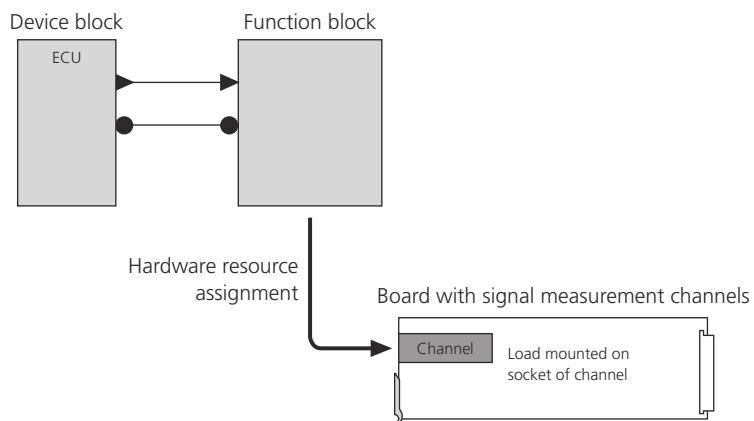
## Differences on implementing

Two general preconditions must be fulfilled for both internal and external loads:

- You have to add a function block to your signal chain which supports the use of loads to simulate actuators. Suitable function blocks have a signal import, for example, blocks which are instantiated from the Multi Bit In function block type.
- Your hardware topology must include a board which contains signal measurement channels, for example, the DS2601.

However, there are differences in handling.

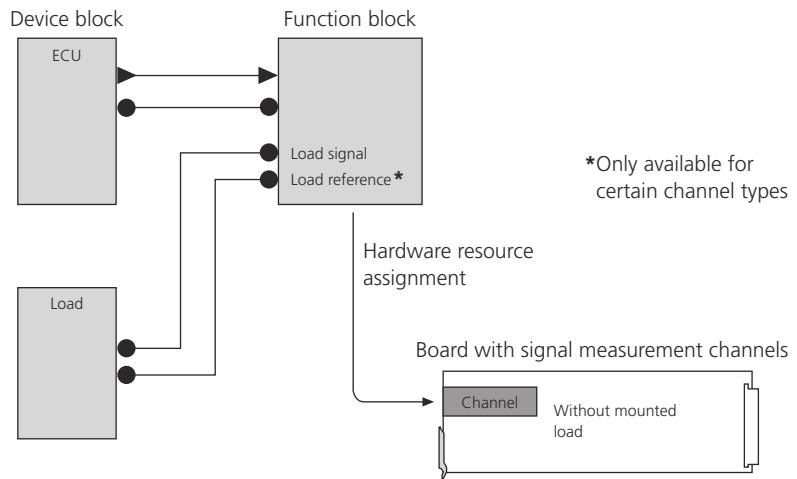
**Internal loads** Internal loads are mounted directly on a socket of a signal measurement channel. No further mapping in the signal chain and no external wiring are necessary. The main configuration task is to assign the signal import of a function block to the channel which holds the required load. This is shown in the illustration below.



To document the internal loads mounted on the hardware, you can enter string-based descriptions. These can be stored on the hardware independently of any ConfigurationDesk application.

For further details on using internal loads, refer to [Details on Handling Internal Loads](#) on page 392.

**External loads** External loads are connected to the I/O connector of the real-time hardware. You have to create a representation of the load in the signal chain via a specific device block (Load type). You have to map the ports of this device block to specific ports of the function block, as shown in the illustration below.



\*Only available for certain channel types

ConfigurationDesk takes external loads into account in the external cable harness. The wiring information is calculated and provided in the same way as for other external devices such as ECUs.

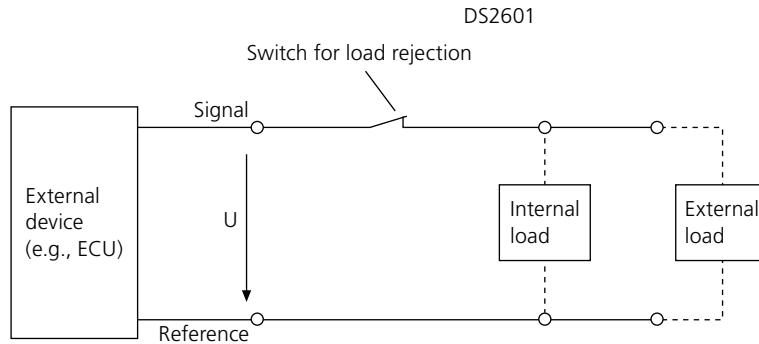
For further details on using external loads, refer to [Details on Handling External Loads](#) on page 395.

## Basics on Load Rejection

<b>Objective</b>	You can specify the load rejection behavior for both internal and external loads.
------------------	---

<b>Usage of load rejection</b>	To protect sensitive loads against damage, they can be automatically disconnected by the system when a failure is simulated, for example a short circuit. This disconnection can be enforced by the load rejection configuration in ConfigurationDesk.
--------------------------------	--

Disconnection means, that the load is cut off from the signal electrically. The following simplified I/O circuit for a DS2601 signal measurement board shows that one switch cuts off both loads at the same time. The circuit looks different for other boards such as the DS2680 or DS2690, but the position of the switch for load rejection is the same.

**Note**

Load rejection is not provided for every failure class, due to technical restrictions. For details, refer to [Load Rejection and Signal Dropping](#) on page 378.

**NOTICE****Risk of damage to a connected load**

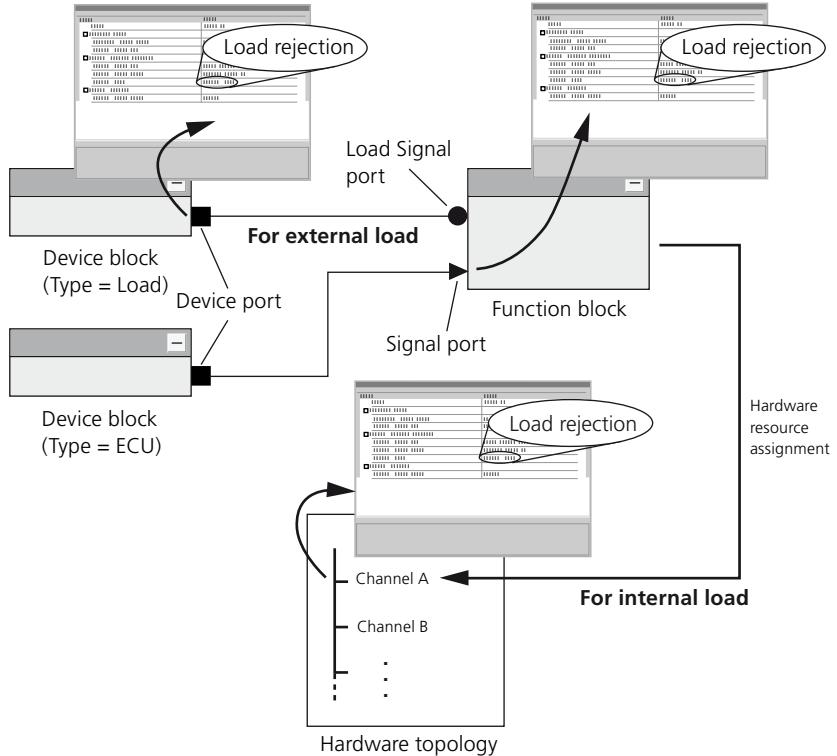
When FRU relays are switched for electrical error simulation, load rejection can be delayed up to 30 ms due to the switching times of the relays involved. A connected load can be damaged during these 30 ms.

**Enforcing load rejection**

If you enforce load rejection in ConfigurationDesk, you cannot use failure simulation without rejecting loads via your experiment software. If load rejection is not enforced in ConfigurationDesk, the load still can be rejected via experiment software when a failure is simulated.

In ConfigurationDesk, you can enforce load rejection at different positions in the signal path, as shown below. If load rejection is enforced at one of these positions, other settings are ignored. The load is then rejected during failure simulation.

You have to specify the load rejection behavior for each signal separately.



You can specify the load rejection behavior:

- At each *device port* which is configured as Load type. Specify load rejection here if you use an external load. The default setting is Enforced.
- At each *signal import* of a function block. The default setting is Enforced.

#### Note

If multiple channels are used, the setting at the function block affects all the channels on the real-time hardware to which the signal is assigned.

- At each *signal measurement channel* available in the hardware topology. You can specify the load rejection here if you use an internal load. The default setting is Not enforced. You can change the setting, for example, via Platform Manager. For details, refer to [How to Provide Data of Internal Loads to the Hardware](#) on page 397.

#### Note

If you enforce load rejection directly on the hardware, the load is rejected even if you are using an external load.

#### Transfer settings

To simplify configuration work, you can transfer the settings for the rejection of an external load from a device port of a load device to the relevant signal port of

the mapped function block. For instructions, refer to [How to Transfer Port Settings from Device Ports to Signal Ports](#) on page 317.

#### Warning messages

A warning is displayed in ConfigurationDesk and a message is generated during the build process in the following cases:

- Load rejection is not enforced by a signal port (of a function block), but by the assigned hardware.
- Load rejection is not enforced by a signal port (of a function block), but by the external load's device port mapped to the load signal port (of a function block).

#### Related topics

##### Basics

Configuring External Devices.....	272
Configuring Function Blocks.....	318

##### HowTos

How to Provide Data of Internal Loads to the Hardware.....	397
--	-----

## Details on Handling Internal Loads

#### Objective

Internal loads are loads which are mounted directly on the dSPACE real-time hardware. No additional internal or external wiring is necessary.

#### Preparatory steps

After you have mounted the load on the channel of the real-time hardware, you can provide the data of this load to the corresponding channel. The data is stored on the hardware independently of any ConfigurationDesk application. The data comprises:

- Load description

You can document the mounted load with a string-based description. For example, you could enter the resistance value "100 Ω" or "4.7 Ω / 5 W". Another possibility is to use inventory numbers. The entries do not affect the channel's behavior. Note that ConfigurationDesk does not check if your entries comply with the real hardware equipment.

This property is used for load compare checks and the automatic hardware resource assignment.

- Load rejection

If your load must be protected against damage during a failure simulation, you should enforce load rejection (default setting = not enforced). For basics, refer to [Basics on Load Rejection](#) on page 389.

The data can be displayed and changed via the channel properties in the Properties Browser. You can access them via:

- **Platform Manager**

Here you can change the settings independently of any ConfigurationDesk application.

For instructions, refer to [How to Provide Data of Internal Loads to the Hardware](#) on page 397.

- **Hardware Resource Browser**

Here, the settings are only relevant for the active ConfigurationDesk application.

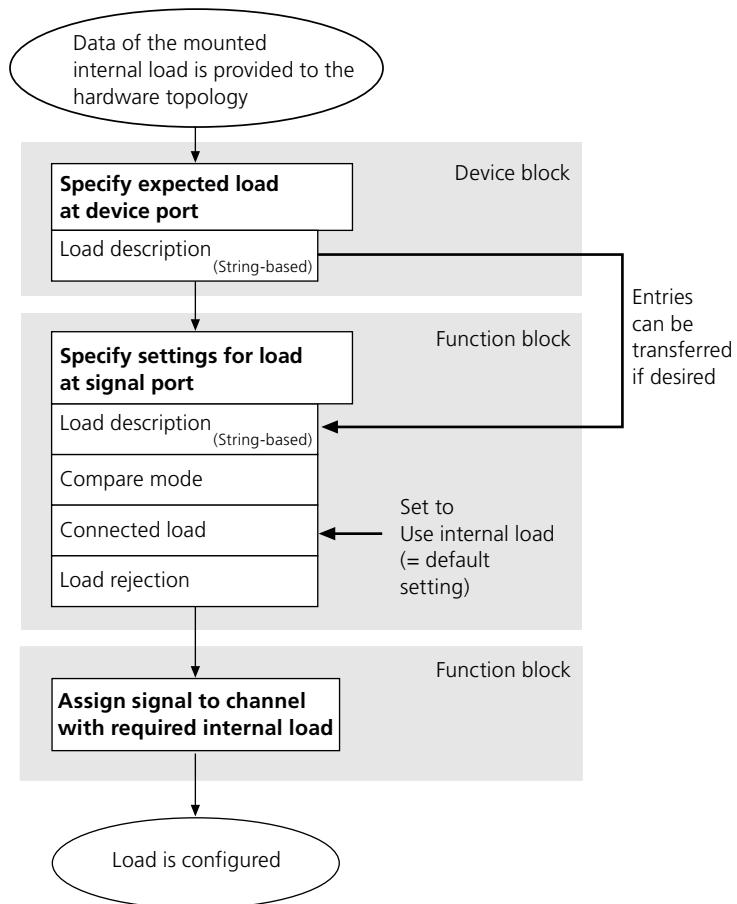
The settings are checked during your configuration work and the build process. However, they cannot be transferred to a registered hardware system. They are overwritten when the hardware topology of your application is replaced.

The "Matching platform connected" application state changes to "No matching platform connected" when the load settings are changed in the Platform Manager or in the Hardware Resources Browser.

---

### Typical workflow

The following illustration shows a useful workflow for specifying an internal load.



<b>Load descriptions</b>	<p>To document your load requests and avoid incorrect hardware resource assignment, ConfigurationDesk lets you enter string-based descriptions of loads at different positions in the signal chain.</p> <p>As shown in the workflow above, the blocks provide the following properties to specify the load descriptions:</p> <ul style="list-style-type: none"><li>▪ <b>Load description for each device port of a device whose type is set to ECU.</b> You can enter a description for an expected (required) load that has to be connected to the signal. This entry is only for documentation purposes. It is not included in the load compare checks. When you want to use an internal load, you can transfer these settings to the Load description property of the signal port. If a signal port is mapped to several device ports, the expected load is transferred only if the settings for all the device ports match exactly. If they do not match, the description is deleted at the signal port and no load description is specified there.</li><li>▪ <b>Load description at each signal import (function block)</b> You can enter a load description to be used for the load compare check during configuration and during download. For example, during configuration, ConfigurationDesk checks this load description against the load description which is stored on the hardware of the assigned channel. For more details on the checks, see below.</li></ul>
<b>Load compare check</b>	<p>For using internal loads, ConfigurationDesk provides a load compare check to ensure that the load mounted on the hardware matches the required load.</p> <p>When a channel is assigned to a function block, the function block's load description is checked against the load description defined for the hardware. This check can be performed either by ConfigurationDesk or by the user as follows:</p> <ul style="list-style-type: none"><li>▪ <b>Compare mode is set to Load description is compared by system</b> ConfigurationDesk considers entries to be identical only if they contain the same characters (= string-based comparison). A failed check causes a conflict which is displayed in the Conflicts Viewer. The build process generates a warning message.</li><li>▪ <b>Compare mode is set to Load description is compared by user</b> You have to verify that the descriptions are compatible yourself. ConfigurationDesk does not perform a check, so different load descriptions do not cause conflicts and generate warnings.</li></ul>
<b>Load compare check during download</b>	<p>If you download a real-time application to your hardware, ConfigurationDesk checks if the channel's internal load description which was used during the build process matches the internal load description stored on the hardware. If there is a deviation, then the application is possibly not compatible with the hardware system. ConfigurationDesk generates a warning message and you can abort the download.</p>

You cannot disable this check.

**NOTICE**
**Ignoring the warning message can cause material damage.**

Risk of damage to the connected external device.

- Cancel the download of the real-time application if you are not sure about the consequences.
  - Disable the affected channels of the internal loads if you do not need them.
  - Ensure that rejecting internal loads does not damage the connected external device.
- If the two internal load descriptions conflict, load rejection of the affected internal load will always be enforced.

**Related topics**
**Basics**

Basics on Connecting Internal Loads (DS2601) (SCALEXIO Hardware Installation and Configuration 	
Configuring External Devices.....	272
Configuring Function Blocks.....	318

## Details on Handling External Loads

**Objective**

External loads are connected to the I/O connector of your real-time hardware.

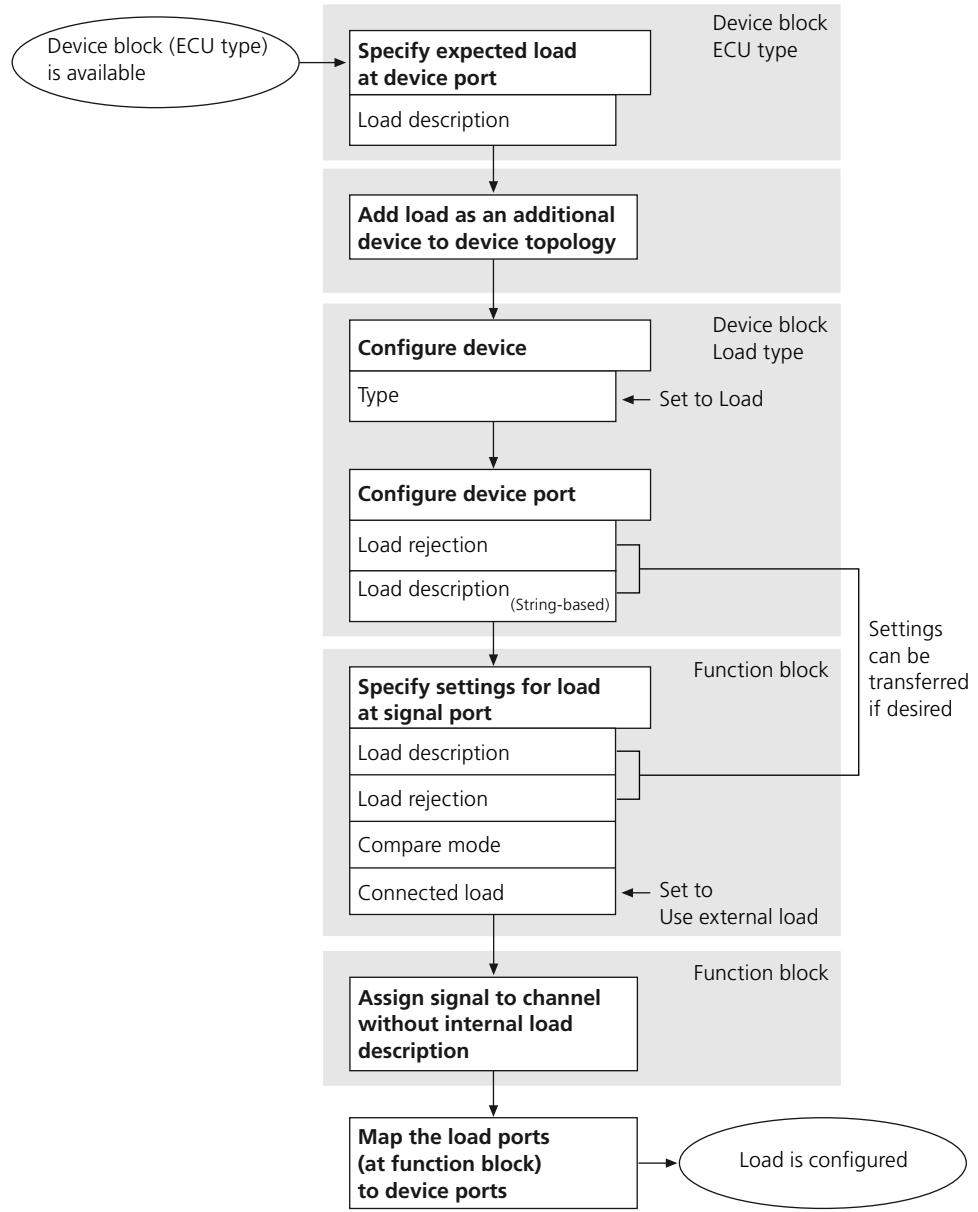
**Main working steps**

Using an external load in ConfigurationDesk consists mainly of the following tasks:

- You have to add a device block to the device topology and set its type to Load.
  - At the function block:
    - You have to specify that an external load must be connected to the hardware for each signal import separately.
    - You have to assign a channel set to the function block.
- Then for the signal two additional reference ports appear at the function block, named Load Signal and Load Reference (the latter is only available for certain channel types).
- You have to map the ports of the device block to the load ports at the function blocks.

**Typical workflow**

The following illustration shows a useful workflow for providing an external load.

**Load descriptions**

To document your load requests, ConfigurationDesk lets you enter string-based descriptions of loads at different positions in the signal chain.

As shown in the workflow above, the blocks provide properties to specify the load descriptions:

- Load description for each device port of a device whose type is set to ECU. You can enter a description for an expected (required) load that has to be connected to the signal. This is only for documentation purposes.

- Load description at a device port, whose type is set to Load.  
You can enter a description for the characteristics of the load. This is only for documentation purposes.
- Load description at each signal import (function block).  
This property should be used only for internal loads. Reason: It is used for load compare checks during configuration and during download. These checks are not intended for external loads. As a result, generated warning messages are useless.  
However, if you need to enter a description, you can disable the checks by setting the Compare mode property to Load description is compared by the user.

**Compare checks**

No checks are provided for external loads during configuration and during download.

**Related topics****Basics**

Configuring External Devices.....	272
Configuring Function Blocks.....	318

## How to Provide Data of Internal Loads to the Hardware

**Objective**

Channels on signal measurement boards (such as the DS2601) provide properties to specify the load which is mounted on a load socket on the channel (= internal load).

**Data**

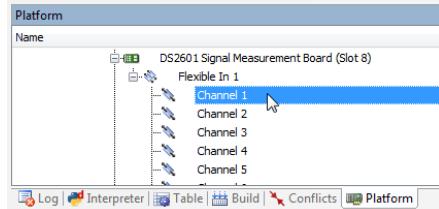
You can provide a load description and the load rejection behavior to every channel of the hardware. The entries are stored on the hardware. They can be displayed by ConfigurationDesk's Platform Manager, independently of any ConfigurationDesk application. For details on the data, refer to [Details on Handling Internal Loads](#) on page 392.

**Precondition**

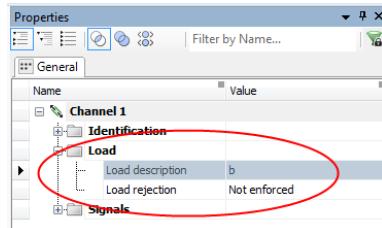
- Hardware must be registered in ConfigurationDesk.
- The registered hardware must contain at least one board which contains signal measurement channels, such as the DS2601 or DS2680.

**Method****To provide data of internal loads to the hardware**

- 1 In the Platform Manager, select the signal measurement board and the channel whose load data you want to specify.



The Properties Browser contains the Load description and Load rejection properties for the selected channel.



- 2 Specify the properties as required.

**Result**

Your entries and settings are stored on the hardware immediately and independently of any ConfigurationDesk application.

# Assigning Hardware Resources to Function Blocks

## Objective

Most function blocks require at least one hardware resource of the real-time hardware (channels on an I/O board or I/O unit) in order to be executed and perform the I/O functionality.

With ConfigurationDesk and the architecture of the dSPACE real-time hardware, the execution of a function block is not tied to any specific hardware. Function blocks can be assigned to any hardware resource which is suitable for the functionality.

## Where to go from here

### Information in this section

Basics on Hardware Resource Assignment.....	399
Methods for Assigning Hardware Resources.....	404
Assignment Conflicts and Their Effects on Code Generation.....	409
How To Assign Hardware Resources Manually via Properties Browser.....	410
How to Assign Hardware Resources Manually via Drag & Drop.....	412
How to Assign Hardware Resources Automatically.....	414
How to Reuse the Hardware Resource Assignment of Missing Hardware.....	415

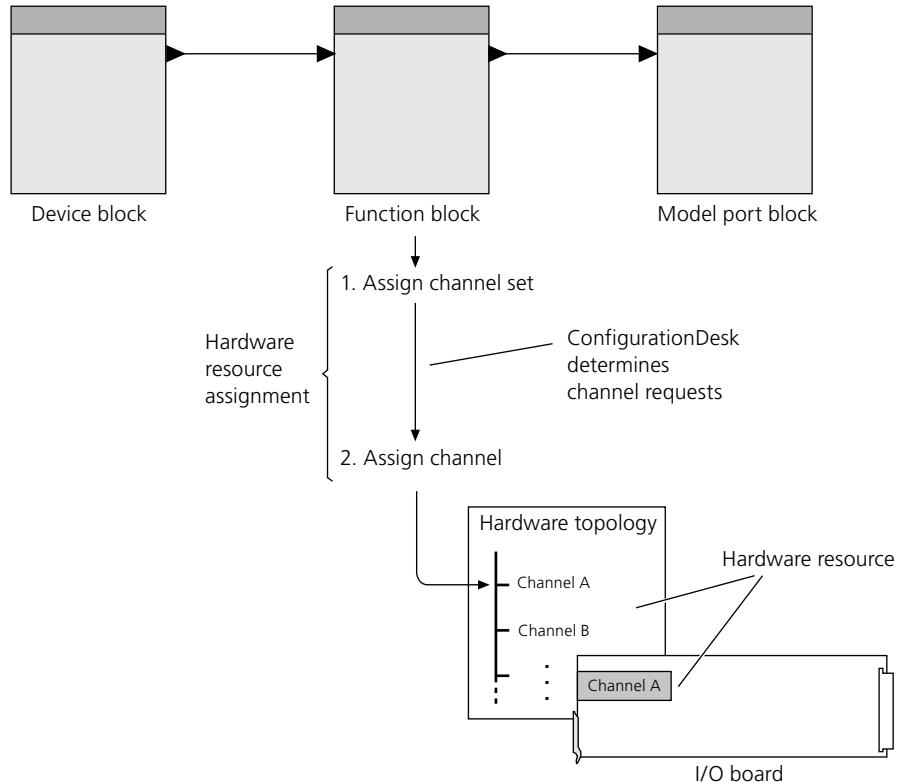
## Basics on Hardware Resource Assignment

### Assigning hardware resources

Most function blocks require at least one hardware resource (channel) to perform the I/O functionality. 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. The required channels are called *channel requests*. 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.

Mapping channel requests to a specific channel of the hardware topology is called hardware resource assignment. The illustration below shows hardware resource assignment in the context of the signal chain:



**Definition of channel set** A channel set is 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.

You can assign a channel set if the hardware topology of the ConfigurationDesk application contains hardware which provides at least one channel set which is suitable for the function block. You can change this assignment at any time during configuration. However, to avoid conflicts and to reduce conflict resolutions, you should assign a channel set as early as possible.

There are different methods for assigning a hardware resource (channel) to a channel request, for example, automatic assignment. For details, refer to [Methods for Assigning Hardware Resources](#) on page 404.

Apart from the Conflicts Viewer, assignment conflicts are displayed graphically at the function block with a colored frame (orange or red frame color) and in a tooltip if highlighting of signal chain elements is enabled. For a description of all possible states, refer to [Assignment Conflicts and Their Effects on Code Generation](#) on page 409.

#### Details on a channel request

The logical signal plays a major role in channel requests. The logical signal summarizes a group of electrical potentials connected to a function block (see illustration below).

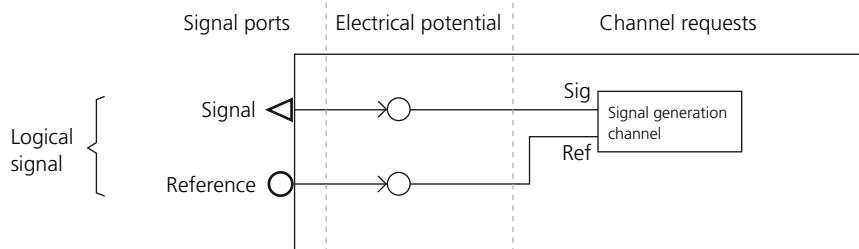
Many function blocks provide only one logical signal, for example, PWM/PFM Out. Others have several logical signals, for example, Digital Incremental Encoder Out (Phi0, Phi90 and Index), or they have a variable number of logical signals depending on their configuration, for example, Multi Bit Out (Bit 1, Bit 2 ... depending on the Number of bits property).

Usually a logical signal consists of two potentials: "Signal" and "Reference". However, there are logical signals with more potentials, for example, digital output signals operated in push/pull mode (High Reference, Signal, Reference), input signals with external loads (Signal, Reference, Load Signal, Load Reference) and bus signals (CAN High, CAN Low, CAN GND).

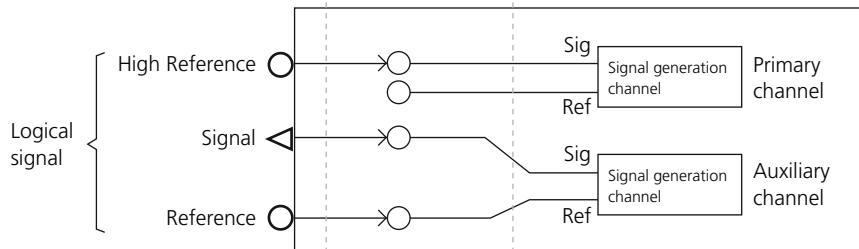
Each logical signal causes at least one channel request. One logical signal can also have several channel requests. Channel requests are available after you have assigned a channel set to the logical signal.

The illustration below shows channel requests according to different configurations with using the PWM/PFM Out function block type.

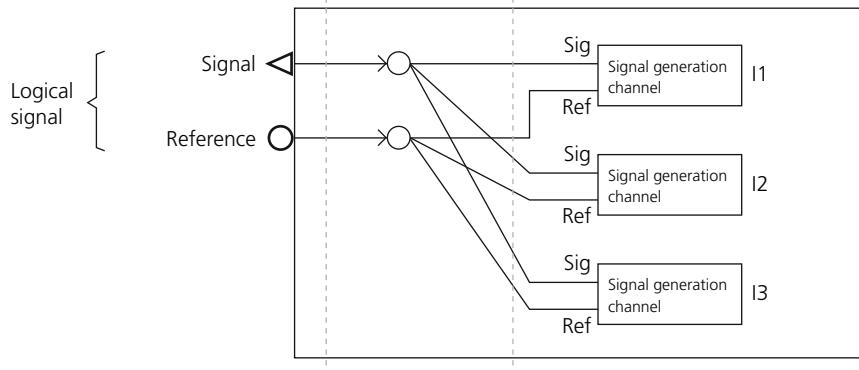
#### PWM/PFM Out in low-side switch mode



#### PWM/PFM Out in push/pull mode



#### PWM/PFM Out in low-side switch mode (with channel multiplication caused by current enhancement)



The illustration below shows two examples of hardware resource assignment, which can occur when channel multiplication is performed: Channel requests with the syntax *Logical Signal, Ux* identify channel multiplication for voltage enhancement (= series connection), and channel requests with the syntax *Logical Signal, Ix* identify channel multiplication for current enhancement (= parallel connection).

Hardware Resource Assignment	
Assigned channel set	DS2621 Signal Ge...
... Voltage Out, U0	
... Voltage Out, U1	
... Voltage Out, U2	

Hardware Resource Assignment	
Assigned channel set	DS2680 I/O Modul...
... Current Sink, I0	
... Current Sink, I1	
... Current Sink, I2	
... Current Sink, I3	

### Effects when using an existing cable harness

You can use the representation of an existing cable harness in your ConfigurationDesk application, for example, to avoid building a new cable harness.

If you assign hardware resources which do not match the pins of the external cable harness, a wiring conflict is shown at the device port mapping. You can solve the conflicts by reassigning the hardware resources or remapping ports in the signal chain.

#### Note

Recalculating the wiring information can also solve the mapping conflicts. However, in this case the external cable harness changes. If you want to use an existing cable harness, you must not recalculate the wiring information.

### Effects when changing the channel set after configuration was completed

If you change the channel set after you configured the function block or the signal chain (for example, the device port mapping), the following worksteps are necessary:

- Changing the channel set may result in changed features and changed value ranges for certain properties. ConfigurationDesk checks the settings of all properties for conflicts. Mismatching configuration settings are displayed in the Conflicts Viewer. The Conflicts Viewer helps you to resolve the conflicts.
- If the channel type also changes when you change the channel set, ConfigurationDesk may add new unmapped signal ports to the function block. You have to modify the mapping according to the circuit diagrams of the selected channel type. After this, the obsolete ports disappear.
- You have to recalculate the external cable harness.

#### Tip

Remember that if you recalculate an external cable harness in ConfigurationDesk, you have to build a new real existing one.

## Methods for Assigning Hardware Resources

<b>Objective</b>	ConfigurationDesk provides different methods to assign a hardware resource to a function block.
------------------	---

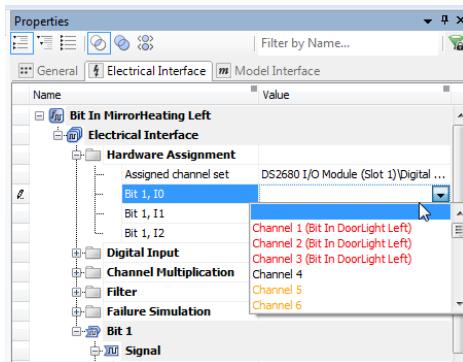
<b>Possible methods</b>	<p>You can choose between the following methods:</p> <ul style="list-style-type: none"> <li>▪ Manual assignment via <b>Properties Browser</b>.</li> <li>▪ Manual assignment of resources from the hardware topology via drag &amp; drop.</li> <li>▪ Automatic assignment via context menu commands (available in different components, for example, working views).</li> <li>▪ Reusing the hardware resource assignment of missing hardware.</li> </ul>
-------------------------	---

There are some differences, which are described below.

<b>General rules</b>	<p>All the methods use the following rules for assignment:</p> <ul style="list-style-type: none"> <li>▪ Existing assignments are protected. ConfigurationDesk does not delete them without explicit user action even when they are invalid.</li> <li>▪ The wiring information on an external cable harness loaded to an application is considered in hardware resource assignment.</li> </ul>
----------------------	---

<b>Manual assignment</b>	<p>You can assign a suitable channel set to each function block manually via the <b>Properties Browser</b>. ConfigurationDesk automatically determines suitable channel sets for the requests of the function block and displays a list of suitable channel sets for you to select from. After assigning the channel set, you have to assign the channels to the channel requests manually. The offered channel sets displayed in the <b>Properties Browser</b> are colored:</p> <ul style="list-style-type: none"> <li>▪ Black: You can assign this channel set without conflicts.</li> <li>▪ Orange: When you assign this channel set, at least one conflict arises and is shown as a tooltip and in the <b>Conflicts Viewer</b>.</li> </ul>
--------------------------	--

After you have assigned a channel set, channel requests are added to the **Properties Browser**, where you have to assign a single channel from the channel set to complete the hardware resource assignment.



ConfigurationDesk displays the available channels in different colors:

- Black: You can assign this channel without conflicts.
- Orange: The channel is already assigned to the same function block, or the channel has a conflict, for example, the assignment does not match the existing cable harness.

If the channel is assigned to the same function block and in addition to another function block, the name of the other function block is displayed in parentheses.

When you assign such a channel, at least one conflict appears. The conflict is shown as a tooltip and in the Conflicts Viewer.

- Red: The channel is already assigned to another function block. The name of the function block which uses the channel is displayed in parentheses.

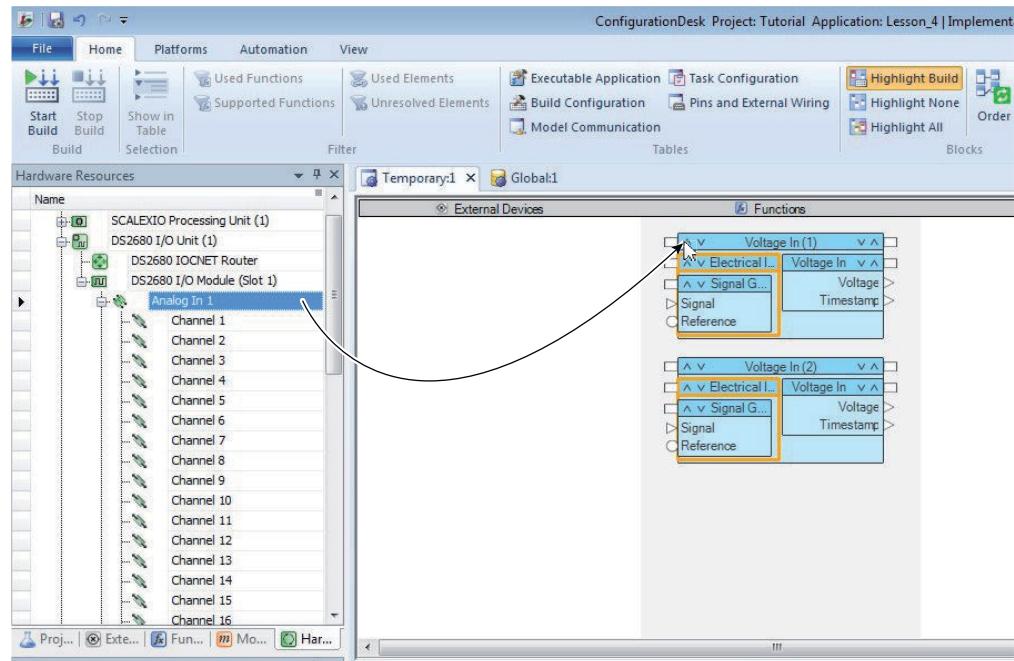
When you assign such a channel, at least one conflict appears. The conflict is shown as a tooltip and in the Conflicts Viewer.

### Assignment via drag & drop

You can assign hardware resources (channel sets and/or channels) by dragging them from the available hardware topology to a specific function block.

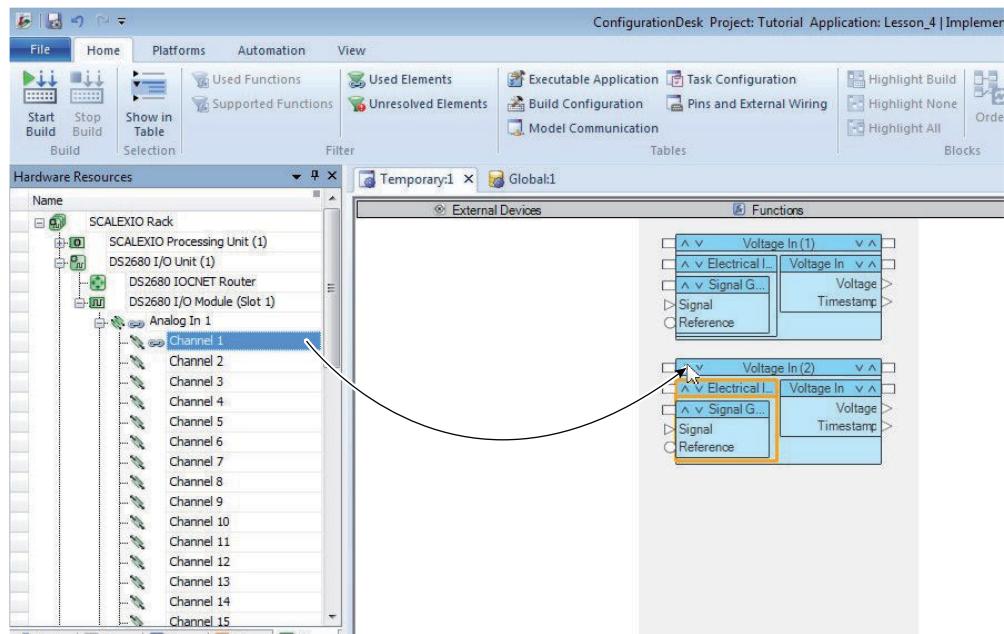
The following rules apply for dragging and dropping of channel sets:

- Channel sets can be dropped (= assigned) if they match all the properties required, or if they do not match all the properties required but the conflict can be resolved by reconfiguration or by remapping at another position in the signal chain.
- If you drop a channel set to the function block, the required channels are also assigned beginning with the unassigned channel with the lowest number. Existing channel assignments are not deleted or changed.



The following rules apply for dragging and dropping of channels:

- Channels can be dropped (=assigned) if they (and therefore the related channel set) match all the properties required, or if they do not match all the properties required but the conflict can be resolved by reconfiguration or by remapping at another position in the signal chain.
- You can drag a channel to the function block, if the related channel set is already assigned to the function block, or if there is no channel set assigned to the function block. In the latter case, the associated channel set is assigned simultaneously.
- You cannot drag a channel to a function block that is assigned to a channel set which the channel does not belong to.
- Already assigned channels cannot be assigned multiple times.

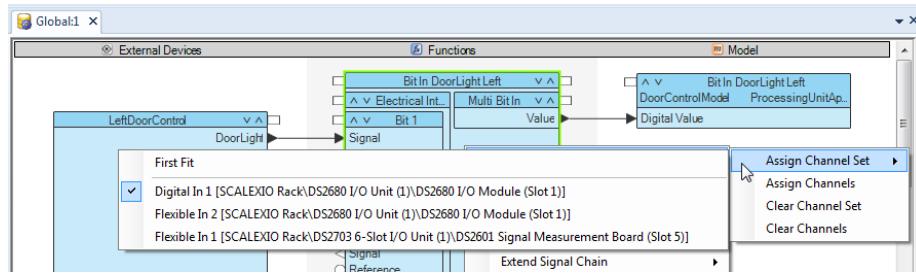


### Automatic assignment

ConfigurationDesk automatically determines suitable channel sets for the requests of a function block and displays a list of suitable channel sets as context menu commands for you to select from. The order of the list depends on the order of the hardware topology.

The list contains only sets which match specific basic requirements according to the following rules:

- Channel sets are provided only if they already have the required number of unassigned channels. Obsolete assignments are treated as assigned.
- Channel sets are provided only if they match all the properties required. However, channels sets are also provided, if they do not match all the properties provided, but the conflict can be resolved by reconfiguration or by remapping at another position in the signal chain.
- If an external cable harness is loaded to the application, channel sets are only provided if they match the wiring information.



As shown above, ConfigurationDesk displays the location for each channel set in the topology (rack name, unit name, slot number).

The context menu does not contain any channel set entries in the following cases:

- No suitable hardware resource exists.
- You have selected several function blocks.
- The function block contains more than one electrical interface unit, which require different hardware resources.

If you use the **Assign Channel Set – First fit** command, ConfigurationDesk assigns the first suitable channel set to the selected function block(s). Note, that this assignment is function-block-oriented, i.e.:

- A function block is never assigned partially: Either all channel requests of a function block are assigned (without conflict) or no channel request is assigned.
- As a result, if you select several function blocks to assign them automatically, some blocks may be assigned while others are not.

You can access the automatic assignment via a context menu command in different ConfigurationDesk components (selected function blocks in working views and the Function Browser).

#### Note

If you reuse an external cable harness, the automatic hardware resource assignment analyzes the relations between the device port mapping and the external cable harness before it assigns the hardware. In exceptional cases, this analysis might not cover all complex relations. This could lead to a hardware resource assignment that results in a conflict.

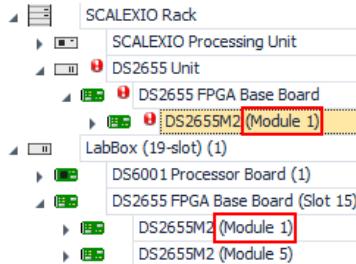
#### Reusing the assignment of missing hardware

You can move the hardware resource assignment of missing hardware resources via drag & drop to other hardware resources.

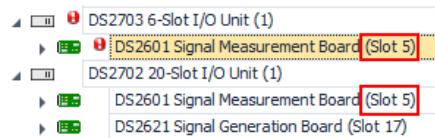
**Rules for moving hardware resource assignments** The following rules apply for moving hardware resource assignments:

- Rules concerning boards:
  - The missing board and the available board must have the same DS number.

- If I/O modules are installed to the missing board, the missing I/O modules must be installed to the same I/O module slots as the available I/O modules.



- Rules concerning I/O unit/box: The missing boards must be installed to the same slots as the boards of the available I/O unit/box.



The type of the I/O unit/box can be different, e.g., you can move the hardware resource assignment of a missing slot I/O unit to a LabBox.

- You can move the hardware resource assignment of processing units. It is possible to move the assignment between different types of processing units (for example SCALEXIO Processing Unit and DS6001 Processor Board), if all required resources are present, such as angle units.
- You cannot move the hardware resource assignment of single channel sets or single channels.

**Moving to assigned hardware resources** If you move the hardware resource assignment of a missing hardware to a hardware resource that is already assigned to a function block, the hardware resource will be assigned multiple times. The multiple assignments cause conflicts that are displayed in the Conflicts Viewer.

#### Deleting a hardware resource assignment

Assignments have to be deleted explicitly. ConfigurationDesk provides different methods:

- **Via Properties Browser**  
Select the empty item from the drop-down list to delete the assigned channel set (or channel).
- **Via the Clear Channel Set and Clear Channels context menu commands**  
These commands are available for selected function block(s) in working views. If you use one of these commands, the assigned channel set (and/or the assigned channels) used at the selected function block is/are deleted.

#### Note

ConfigurationDesk never deletes a hardware resource assignment automatically due to configuration changes. This is only done when you delete a function block from your application.

# Assignment Conflicts and Their Effects on Code Generation

<b>Objective</b>	ConfigurationDesk displays the current conflicts of hardware resource assignment for each channel request in the Conflicts Viewer. The conflicts may influence code generation during the build process of the real-time application.  In addition, if highlighting of signal chain elements is enabled, conflicts are displayed graphically at the function block with a colored frame (orange or red frame color) and in a tooltip. Error messages are generated during code generation and displayed in the Build Log Viewer.
<b>Possible conflicts</b>	The following table describes some possible hardware resource assignment conflicts.
Conflict	Description
Missing channel set (or channel) assignment	No or not all required hardware resources are assigned to the function block.
Unresolved channel request	The hardware resource has been assigned, but is no longer required by the function block (= obsolete).  Example: A function block first requires five channels, but needs only two channels after reconfiguration. Three channel requests are obsolete but the channels are still assigned to the function block.  In the hardware topology (shown in the Hardware Resource Browser) the "obsolete" resource (channel) is still displayed as used. The Properties Browser displays obsolete hardware resources in the related drop-down list with the suffix "obsolete".  To use channels of obsolete channel requests for other channel requests, you have to delete the assignment beforehand. Within a function block, you can use "obsolete" channels without explicitly deleting the assignment. In this case, obsolete channel requests and their assignments are removed automatically after reassigning the channel.
Unresolved channel or channel set assigned	The hardware resource was assigned while it was available in the hardware topology. However, the hardware topology of the application was changed and the current hardware topology does not contain the hardware resource (= missing resource). To solve this conflict, you must replace the hardware topology or assign another available hardware resource. Note that assigning another hardware resource may introduce wiring conflicts, so the external cable harness must be recalculated.
Multiple assignments to channel requests	A channel is assigned to more than one channel request in your ConfigurationDesk application.
Invalid channel assignment	Channel dependency conflict: A condition for the channel's location in relation to another channel is violated (for example, channel x must be located next to channel y).
Unsupported property settings	The assigned hardware resource exists, but does not match one or more of the properties required. The conflict can be resolved by reconfiguration or by remapping at another place in the signal chain.  For example, it fails the load compare check. This means that the load description stored on the channel does not match the load description of the logical signal.

<b>Effects on code generation</b>	Code generation is influenced by hardware resource assignment conflicts. There are the following main effects: <ul style="list-style-type: none"><li>▪ Missing channel set assignment, Multiple assignments to channel requests: No code is generated. The function block is ignored during code generation.</li><li>▪ Missing channel assignment, Unresolved channel request, Unresolved channel assignment, Unresolved channel set assignment, Invalid channel assignment: The I/O access is simulated. That means that the initial values configured at function outports are input to the behavior model.</li></ul>
-----------------------------------	---

**Note**

All I/O accesses of a function block are simulated, if at least one channel request is not assigned.

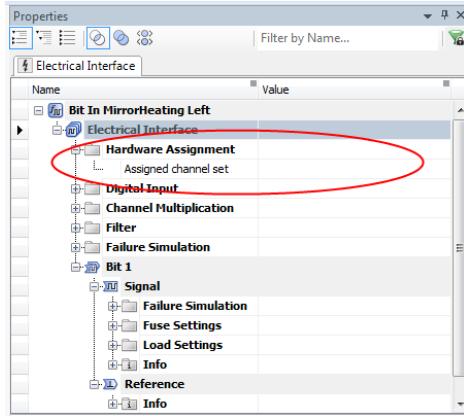
<b>Effects on executing a real-time application</b>	When a real-time application is downloaded, the hardware channels are initialized first. They are set to a secure state during initialization, which means that the channels on the real-time hardware are cut off from the external device electrically, and no signals are available at the I/O connectors of the dSPACE hardware. All channels assigned to a channel request whose state is Unresolved or channels which are not used (= not assigned) remain "secure" during the execution of the real-time application.
---	--

## How To Assign Hardware Resources Manually via Properties Browser

---

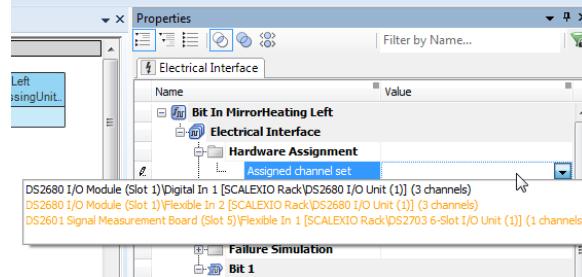
<b>Objective</b>	To assign hardware resources manually you can use the Properties Browser, which gives you access to the hardware resource assignment for every function block separately.
<b>Preconditions</b>	<ul style="list-style-type: none"><li>▪ At least one function block is instantiated.</li><li>▪ A hardware topology containing hardware resources must be available in your application.</li></ul>
<b>Method</b>	<b>To assign hardware resources manually via Properties Browser</b> <b>1</b> In a working view, select the function block. – or – In a table, select the function block.

The Electrical Interface page of the Properties Browser displays the properties for hardware resource assignment as shown below:



**2** Open the drop-down list.

The drop-down list provides all the suitable and available channel sets for the corresponding electrical interface unit of the function block as shown below:

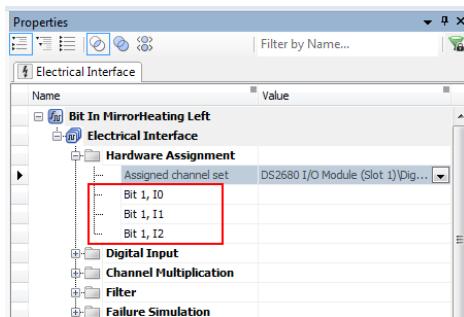


**3** Select a channel set from the drop-down list according to your request. The entries are displayed with colored markup:

- Black: You can assign this channel set without conflicts.
- Orange: When you assign this channel set, at least one conflict appears. After assigning, the corresponding conflict is shown as a tooltip and in the Conflicts Viewer.

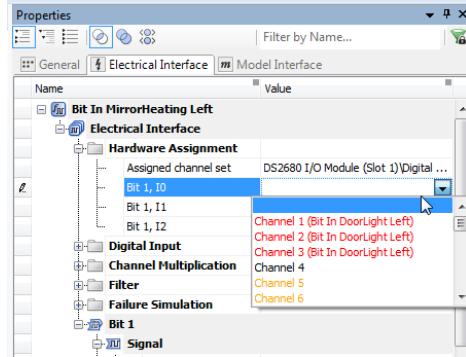
If you select the empty item, the current channel set assignment is deleted.

After selecting a channel set, one or more channel requests appear in the Properties Browser as shown below:



**4** Open the drop-down list.

The drop-down list provides all the suitable and available channels for the corresponding channel request as shown below:



Select a channel for each channel request from the related drop-down list.  
The entries are displayed with colored markup:

- Black: You can assign this channel without conflicts.
- Orange: The channel is already assigned from the same function block, or the channel has a conflict, for example, the assignment does not match the existing cable harness.

If the channel is assigned to the same function block and in addition to another function block, the name of the other function block is displayed in parentheses.

When you assign such a channel, at least one conflict appears. The conflict is shown as a tooltip and in the Conflicts Viewer.

- Red: The channel is already assigned to another function block. The name of the function block which uses the channel is displayed in parentheses.

When you assign such a channel, at least one conflict appears. The conflict is shown as a tooltip and in the Conflicts Viewer.

If you select the empty item, the current channel assignment is deleted.

#### Result

You have assigned hardware resources to a channel request. The settings are saved automatically with the application.

## How to Assign Hardware Resources Manually via Drag & Drop

#### Objective

Via drag & drop, you can assign hardware resources from the hardware topology of your ConfigurationDesk application manually to function blocks.

#### Preconditions

- At least one function block is instantiated.
- A hardware topology containing hardware resources must be available in your application.

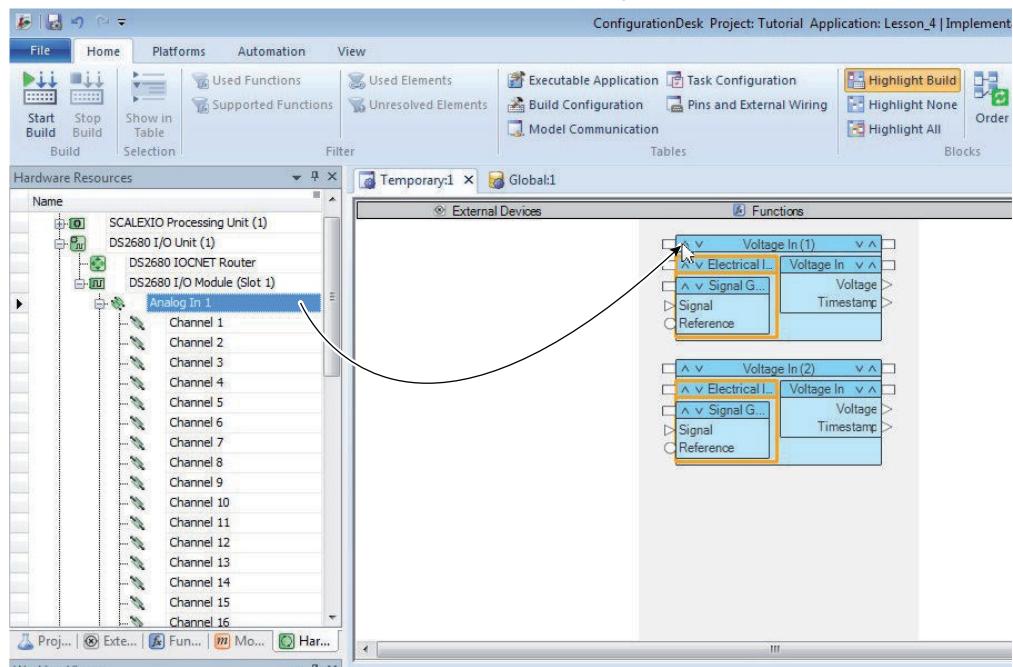
**Method****To assign hardware resources manually via drag & drop**

- 1 Open a working view that contains the function blocks to which you want to assign hardware resources.
- 2 In the hardware topology of your ConfigurationDesk application, navigate to a channel set that is suitable for the desired function block (or its electrical interface unit) and provides the requests of the function block.

**Note**

You only can drop (= assign) hardware resources that are suitable for the function block and which comply with the rules specified in [Methods for Assigning Hardware Resources](#) on page 404.

- 3 Drag the channel set to the title bar (or its electrical interface unit) of the function block as shown in the following screenshot.

**Result**

You have assigned a channel set to a function block via drag & drop. The settings are saved automatically with the application.

You can drag a single channel to a function block (or its electrical interface unit(s)) in the same way. Here, the rules for dragging and dropping also apply. Refer to [Methods for Assigning Hardware Resources](#) on page 404.

## How to Assign Hardware Resources Automatically

### Objective

To assign hardware resources to function blocks, you can use the automatic assignment provided by ConfigurationDesk.

### Basics

The automatic assignment assigns suitable hardware resources which match all the properties required. However, resources are assigned, which do not match all the properties provided, but the conflict can be resolved by reconfiguration or by remapping at another position in the signal chain.

If an external cable harness is loaded to the application, hardware resources are only assigned if they match the wiring information.

For more basic information, refer to [Methods for Assigning Hardware Resources](#) on page 404.

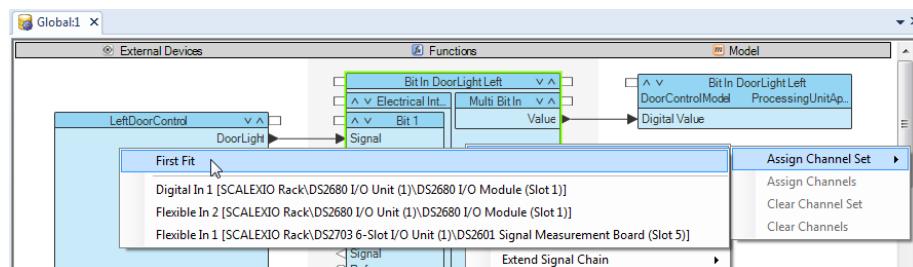
### Preconditions

- At least one function block is instantiated.
- A hardware topology containing hardware resources must be available in your application.

### Method

#### To assign hardware resources automatically

- 1 In a working view, select the function block(s) to which you want to assign a hardware resource.
- 2 Right-click the selected block(s), select **Hardware Assignment**, **Assign Channel Set** and then choose **First Fit**.



### Result

The first suitable channel set is assigned to the function block. Channel requests are assigned also. If there is no suitable hardware resource available for a function block, no hardware resource is assigned.

The settings are saved automatically with the application.

# How to Reuse the Hardware Resource Assignment of Missing Hardware

## Objective

You can move the hardware resource assignment of missing hardware resources to hardware resources of the current hardware topology.

## Rules for moving hardware resource assignments

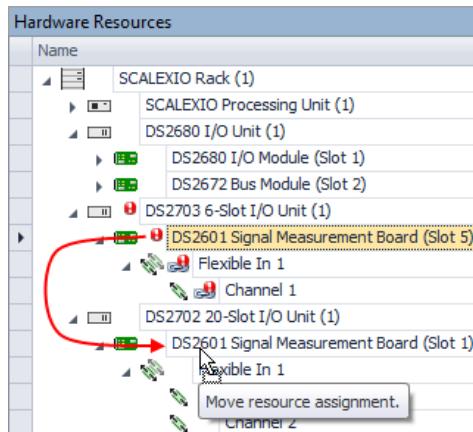
You have to observe the rules for moving hardware resource assignments. Refer to [Methods for Assigning Hardware Resources](#) on page 404.

## Method

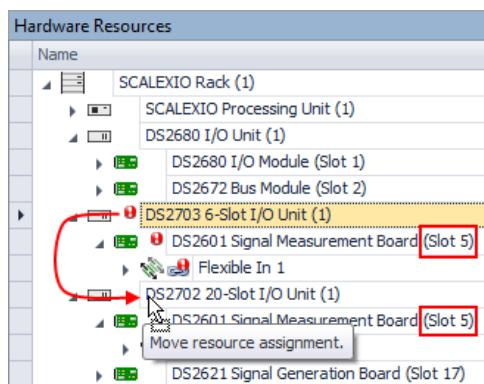
### To reuse the hardware resource assignment of missing hardware

- 1 Open the Signal Chain view set.
- 2 In the Hardware Resource Browser, drag & drop the missing hardware resource to the available hardware resources.

The following example shows how to drag & drop a missing board to an available board:



The following example shows how to drag & drop a missing I/O unit to an available I/O unit:



The missing boards must be installed to the same slots as the boards of the available I/O unit/box.

**Result**

You moved the hardware resource assignment of missing hardware resources to hardware resources of the current hardware topology.

---

**Related topics**

Basics

Methods for Assigning Hardware Resources..... 404

# Specifying the Model Interface

---

## Introduction

In ConfigurationDesk, the model interface is implemented via model port blocks and model ports. To specify it, you must know the basic characteristics of model port blocks and how to handle them.

---

## Where to go from here

### Information in this section

Basics on the Model Interface.....	418
Basics on Model Port Blocks in ConfigurationDesk.....	426
Adding Model Port Blocks to the Signal Chain.....	432
Model Port Mapping.....	438
Setting Up Model Communication.....	448
Configuring Model Port Blocks.....	466

## Basics on the Model Interface

<b>Introduction</b>	When you specify the model interface, you need to understand how it is handled and how data is interchanged.
---------------------	--

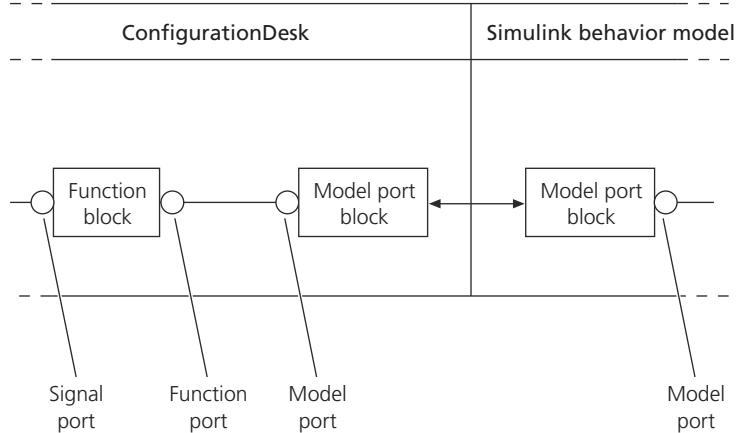
<b>Where to go from here</b>	<b>Information in this section</b>								
	<table><tr><td>Handling the Model Interface.....</td><td>418</td></tr><tr><td>Data Interchange Between ConfigurationDesk and Behavior Model.....</td><td>420</td></tr><tr><td>Specifying Options for Creating Model Port Blocks.....</td><td>421</td></tr><tr><td>Specifics on Handling the ConfigurationDesk Model Interface of Multimodel Applications.....</td><td>424</td></tr></table>	Handling the Model Interface.....	418	Data Interchange Between ConfigurationDesk and Behavior Model.....	420	Specifying Options for Creating Model Port Blocks.....	421	Specifics on Handling the ConfigurationDesk Model Interface of Multimodel Applications.....	424
Handling the Model Interface.....	418								
Data Interchange Between ConfigurationDesk and Behavior Model.....	420								
Specifying Options for Creating Model Port Blocks.....	421								
Specifics on Handling the ConfigurationDesk Model Interface of Multimodel Applications.....	424								

## Handling the Model Interface

<b>Introduction</b>	When working with ConfigurationDesk, you must distinguish between the following parts of the real-time application: <ul style="list-style-type: none"><li>▪ I/O functionality<ul style="list-style-type: none"><li>Part of the real-time model which is implemented in ConfigurationDesk</li></ul></li><li>▪ Behavior model<ul style="list-style-type: none"><li>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.</li></ul></li></ul>
	You can add different types of behavior models to a ConfigurationDesk application: <ul style="list-style-type: none"><li>▪ Simulink behavior models<ul style="list-style-type: none"><li>For more information, refer to <a href="#">Working with Simulink Behavior Models</a> on page 531.</li></ul></li><li>▪ Different types of container files, e.g., Simulink implementation containers, or FMUs<ul style="list-style-type: none"><li>For more information, refer to <a href="#">Working with Container Files as Behavior Models</a> on page 585.</li></ul></li></ul>

### Components of the ConfigurationDesk model interface

To connect the I/O functionality in ConfigurationDesk with the behavior model, you need a [model interface](#). The [ConfigurationDesk model interface](#) is implemented via [model ports](#) that are grouped in [model port blocks](#). Connecting a signal to a model port in ConfigurationDesk makes the signal available in the [behavior model](#) and vice versa. The following illustration shows an example for ConfigurationDesk connected to a Simulink behavior model:



### Adding the behavior model to ConfigurationDesk

To implement a real-time application, ConfigurationDesk requires information on the interface of the behavior model. You must therefore add the behavior model to your ConfigurationDesk application. For instructions, refer to [How to Import a Model Topology](#) on page 112.

### Using model port blocks in the signal chain

Model port blocks are displayed in the [Model Browser](#) and in the [Model-Function Mapping Browser](#). You can use them in the [signal chain](#) by dragging them from the Model Browser to a working view. If you are working with Simulink behavior models, you can use the [Model-Function Mapping Browser](#), for example, to create new signal chains or to model asynchronous tasks. The model port blocks automatically become a part of your application and are used for code generation.

### Multimodel applications

You can link several behavior models to the I/O functionality in ConfigurationDesk. In this multimodel application, the [ConfigurationDesk model interface](#) also provides the model communication via model ports between the different behavior models. For details on this use case, refer to:

- [Specifics on Handling the ConfigurationDesk Model Interface of Multimodel Applications](#) on page 424
- [Setting Up Model Communication](#) on page 448

**Related topics****Basics**

[Analyzing Simulink Behavior Models.....](#) 561

**HowTos**

[How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model.....](#) 543

**References**

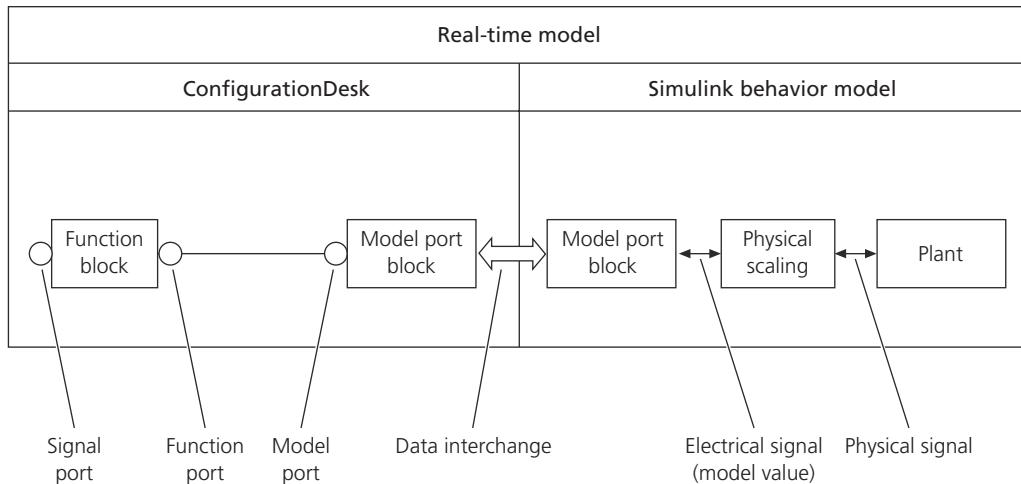
[Analyze Simulink Model \(Model Interface Only\) \(ConfigurationDesk User Interface Reference\)](#)

[Generate Simulink Model Interface – All Unresolved Blocks – New Model \(ConfigurationDesk User Interface Reference\)](#)

## Data Interchange Between ConfigurationDesk and Behavior Model

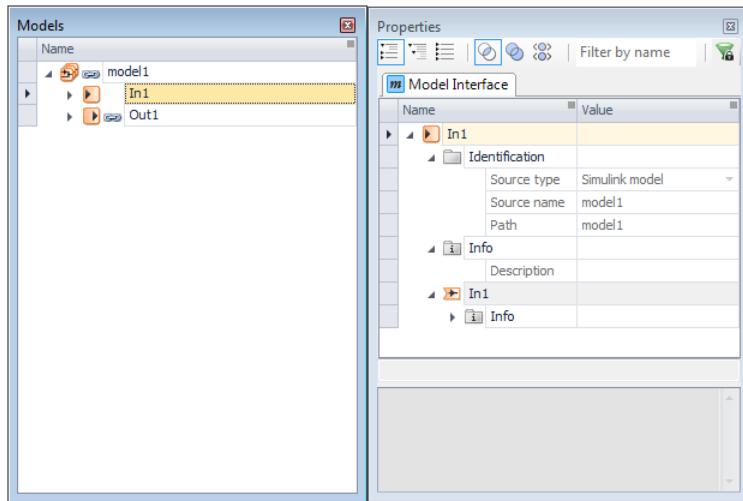
**Interchanging data**

Some tasks in ConfigurationDesk, for example an analysis of a Simulink behavior model, trigger a data interchange between ConfigurationDesk and the behavior model. The following illustration visualizes the data interchange between ConfigurationDesk and a Simulink behavior model:



The data interchange affects the properties of all the model port blocks and model ports which are available in ConfigurationDesk.

In ConfigurationDesk, you can view and check the properties of a model port block via the Properties Browser.



## Related topics

### Basics

[Analyzing Simulink Behavior Models.....](#) 561

### HowTos

<a href="#">How to Import a Model Topology.....</a>	112
<a href="#">How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model.....</a>	543

### References

[Analyze Simulink Model \(Including Task Information\) \(ConfigurationDesk User Interface Reference\)](#)  
[Analyze Simulink Model \(Model Interface Only\) \(ConfigurationDesk User Interface Reference\)](#)

## Specifying Options for Creating Model Port Blocks

### Introduction

ConfigurationDesk lets you specify settings that are used for creating model port blocks in ConfigurationDesk, for example, via a propagate operation.

There are also some points to note concerning the saturation of function port and model port values when you create suitable model port blocks.

---

<b>Possible settings</b>	You can specify the following settings: <ul style="list-style-type: none"><li>▪ <b>Model port data type:</b> Inherited or Float64</li><li>▪ <b>Model port block structure:</b> Function-based, Block-based or Port-based</li></ul> To specify the settings, you must select a function block type in the <b>Function Browser</b> .
<b>Model port data type</b>	Lets you specify the data type of model ports using the following commands: <b>Generate New Simulink Model Interface</b> , <b>Propagate to Simulink Model</b> and <b>Propagate to ConfigurationDesk Model Interface</b> .  Possible values are: <ul style="list-style-type: none"><li>▪ <b>Float64:</b> All data ports obtain this data type.</li><li>▪ <b>Inherited:</b> All data ports inherit the data type of the function ports they are derived from.</li></ul> <div style="background-color: #f0f0f0; padding: 10px; border-radius: 5px;"><p><b>Tip</b></p><ul style="list-style-type: none"><li>▪ Converting and casting different data types between mapped model ports and function ports can increase the turnaround time of the real-time application. Only use different data types if absolutely necessary.</li><li>▪ For the <b>Bus Configuration</b> function block type, you can specify the model port data type for the function block type as well as for each function block instance.</li></ul></div>
<b>Model port block structure</b>	Lets you specify the structure of newly created or updated model port blocks when using the following commands: <b>Generate New Simulink Model Interface</b> , <b>Propagate to Simulink Model</b> , <b>Propagate to ConfigurationDesk Model Interface</b> .  Possible values are: <ul style="list-style-type: none"><li>▪ <b>Function-based:</b> Each function of a function block corresponds to one model port block. All model ports in the model port block have the same data direction.</li><li>▪ <b>Block-based:</b> The structure of the function block defines the number and the structure of the corresponding model port blocks. One model port block exists for each data direction. The model ports in a model port block are structured according to the structure of the corresponding function. In the Simulink behavior model, each model port block has <i>exactly one</i> model port that is structured according to the structure of the related model port block in ConfigurationDesk.</li><li>▪ <b>Port-based:</b> Each function port corresponds to a separate model port block (with one model port).</li></ul>

**Tip**

For the Bus Configuration function block type, you can specify the model port block structure for the function block type as well as for each function block instance.

### Saturation rules for Extend Signal Chain

ConfigurationDesk lets you specify user saturation values for function ports (refer to [Specifying User Saturation \(ConfigurationDesk I/O Function Implementation Guide\)](#)). If you use System min/max values, a data type transformation between the model port and the function port is performed automatically if necessary. If you specify your own User min/max values for function ports and then use the Extend Signal Chain - Create Suitable Model Port Block command, the following applies to the value ranges of function imports and function exports:

**Function imports** The following applies to the value range saturation of function imports:

- The value range of the model outport > the User min/max values range of the function import:  
Values outside the value range of the function import are saturated.
- The value range of the model outport < the User min/max values range of the function import:  
Values of the model outport are transmitted without saturation.
- The value range of the model outport overlaps the User min/max values range of the function import:  
Values outside the overlapping value ranges are saturated.
- The value ranges of the model outport and the User min/max values range of the function import are disjoint:  
The initial value of the function import is used.

**Function outports** You cannot specify your own saturation values for function outports. The following applies to the value range saturation of function outports:

- The value range of the model import > the value range of the function outport:  
Values of the function outport are transmitted without saturation.
- The value range of the model import < the value range of the function outport:  
Values outside the value range of the model import are saturated.
- The value range of the model import overlaps the value range of the function outport:  
Values outside the overlapping value ranges are saturated.
- The value ranges of the model import and the value range of the function outport are disjoint:  
The initial value of the model import is used.

**Entries for the value range of function imports and outports**

If you extend the signal chain with suitable model port blocks, ConfigurationDesk uses the specified currently valid value range as follows:

- The specified value range is written to the Unit property of the generated model port.
- The valid value range is written to the value range of the function port in the TRC file generated during the build process.

**Related topics**

**References**

[Model Port Block Properties \(ConfigurationDesk User Interface Reference\)](#)

[Model Port Properties \(ConfigurationDesk User Interface Reference\)](#)

## Specifics on Handling the ConfigurationDesk Model Interface of Multimodel Applications

---

**Introduction**

If you implement a [multimodel application](#), there are some aspects of the [model interface](#) that you need to know in addition to the standard behavior, described in [Handling the Model Interface](#) on page 418.

**Adding behavior models to ConfigurationDesk**

ConfigurationDesk supports two different use scenarios for adding behavior models to an application to implement a multimodel application:

- You can add several individual behavior models to the active application in ConfigurationDesk step-by-step.

For an overview of this use scenario, refer to [Workflow for Creating a Multicore Real-Time Application Using Multiple Behavior Models](#) on page 548.

- (Only applicable to Simulink behavior models) You can separate several behavior models from an overall model and add them all to the active application at once. The separated models remain part of the overall model, which you can still use for offline simulation in the modeling tool.

Information on the separated models and on the interconnections between them are provided via the model communication description file ([MCD file](#)). This file is generated in the modeling tool and must be added to the active application in ConfigurationDesk.

For an overview of this use scenario, refer to [Workflow for Creating a Multi-PU Application Using One Overall Behavior Model](#) on page 573.

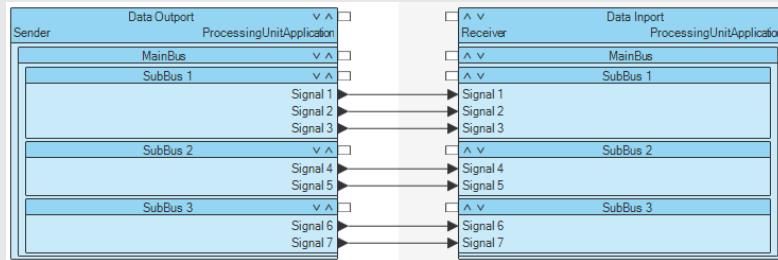
**Model communication**

The different behavior models linked to your ConfigurationDesk application can exchange signal data. In ConfigurationDesk, a [data output](#) of one model

(sending model) must be connected via a mapping line to a [data import](#) of another model (receiving model).

### Tip

Suppose two model port blocks have the same structure, and their ports are connected one-to-one as shown in the illustration below.



In this case, ConfigurationDesk generates optimized code during the build process which leads to an improved performance of the real-time application.

### Note

Mapping model ports in an unfavorable way might reduce the run-time performance. For optimum run-time performance, it is recommended to observe specific mapping rules. Refer to [Model Communication Recommendations for Model Port Blocks with Structured Data Ports](#) on page 464

Communication between models is based on communication packages whose protocol settings specify how signals are updated in the models. You can configure this model communication in the [Model Communication Package](#) table view. For further basic information and detailed instructions, refer to [Setting Up Model Communication](#) on page 448.

# Basics on Model Port Blocks in ConfigurationDesk

**Introduction** Model port blocks and model ports in ConfigurationDesk have characteristics that you need to know when implementing real-time applications.

Where to go from here	Information in this section
	Characteristics of Model Port Blocks..... <a href="#">426</a>
	Characteristics of Model Ports..... <a href="#">429</a>

## Characteristics of Model Port Blocks

<b>Definition of model port blocks</b>	Model port blocks are a part of the <a href="#">ConfigurationDesk model interface</a> . They are necessary for setting up a part of the signal chain, for example, because they provide the data flow between the functions in ConfigurationDesk and the behavior model. This data flow is implemented via model ports. Each model port block can contain one or more model ports of the same data direction. These are the types of model port blocks: <ul style="list-style-type: none"> <li>▪ Model port blocks with data imports These supply data from ConfigurationDesk's function outports to the behavior model via one or more <a href="#">data imports</a>.</li> <li>▪ Model port blocks with data outports These supply data from behavior model signals to ConfigurationDesk's function inports via one or more <a href="#">data outports</a>.</li> <li>▪ Runnable Function blocks These provide runnable functions that are executed in tasks. A Runnable Function block provides one runnable function port. To model an asynchronous task, the runnable function port can be mapped to an event port provided by a function block.</li> </ul>
--	--

- Configuration port blocks

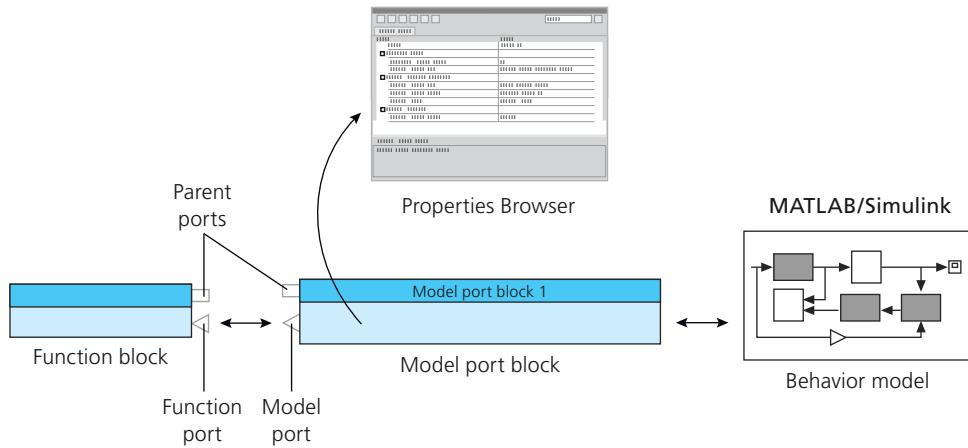
These are specific model port blocks that are created in ConfigurationDesk during model analysis for each of the following blocks found in the Simulink behavior model:

- RTICANMM ControllerSetup block
- RTILINMM ControllerSetup block
- FLEXRAYCONFIG UPDATE block

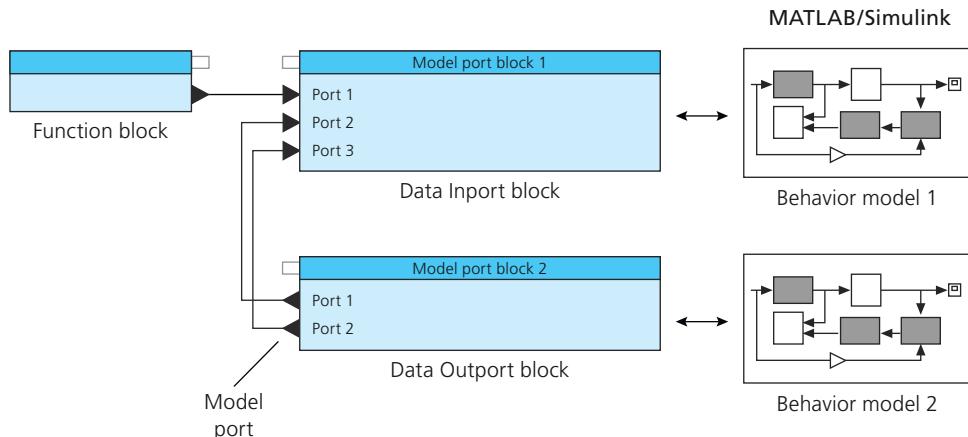
### Purpose of model port blocks

Model port blocks in ConfigurationDesk provide:

- Access to properties for viewing block parameters
- The interface to the behavior model
- The interface to the function ports for mapping function ports to model ports (model port mapping).



- Mapping between two model port blocks that belong to two different behavior models to provide model communication in a multimodel application.



Note that the model ports of a certain model port block can be mapped to function blocks as well as to the model port providing the interface to another behavior model, as shown in the example above.

#### Possible states of model port blocks

Each model port block that is part of an application has a particular state. The following table shows the possible states of model port blocks and how they are displayed in ConfigurationDesk:

State	Display of Model Port Block <sup>1)</sup>		Description	Effect on Code Generation
	Model Browser	Working View		
OK	Block name is displayed without a warning symbol.	Block frame is black.	The model port block is resolved (for details see below). It can be used in the application without restrictions.	Code is generated for the model port block.
Warning	Block name is displayed with a warning symbol.	Block frame is orange. This indicates a warning conflict.	The model port block is unresolved (for details see below).	The model port block is not used during code generation. Warnings are displayed during the build process.

<sup>1)</sup> To highlight warnings, you have to set the appropriate highlighting option. For instructions, refer to [Highlighting Signal Chain Conflicts](#) on page 194.

#### Resolved and unresolved model port blocks

Model port blocks can be:

**Resolved** In this case they are contained in the model topology which is linked to your ConfigurationDesk application. They have been added to ConfigurationDesk via a Simulink behavior model or a container file.

##### Note

If a model port block is resolved, its header contains the name of the model that provides it (below the name of the model port block).

**Unresolved** In this case they are not contained in the model topology which is linked to your ConfigurationDesk application. Unresolved model port blocks can be the result of the following activities:

- They were added to the signal chain via function blocks.
- They were added to the signal chain as inverse model port blocks via the Create Inverse Block command.
- They were removed from the Simulink behavior model which is linked to your active application and afterwards the interface of the behavior model was reanalyzed by ConfigurationDesk.
- The model port block is used in a multimodel application and resides with the same identity in two or more Simulink behavior models which are linked to your active ConfigurationDesk application. Additionally, two or more of these behavior models are assigned to application processes.

In addition this unresolved model port block is displayed in the model topology with the following icon:

Model port blocks of Simulink behavior models which are unresolved are resolved by generating and analyzing model interfaces, or by propagating changes in the ConfigurationDesk to the Simulink model. For more information, refer to the following topics:

- [Basics of Connecting ConfigurationDesk and Simulink Behavior Models](#) on page 534
- [Analyzing Simulink Behavior Models](#) on page 561
- [Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model](#) on page 558

Model port blocks which are unresolved can be easily removed from the application if they are no longer needed. You can delete them by right-clicking the blocks in the Model Browser or in a working view and selecting **Delete from Application** from the context menu.

## Related topics

### Basics

<a href="#">Characteristics of Model Ports</a> .....	429
--	-----

### HowTos

<a href="#">How to Rename Model Port Blocks</a> .....	468
---	-----

### References

<a href="#">Model Port Block Properties (ConfigurationDesk User Interface Reference)</a>	429
--	-----

## Characteristics of Model Ports

### Types of model ports

Each model port block has at least one model port. 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 import** At this data port the values of a function are input for use in the behavior model.

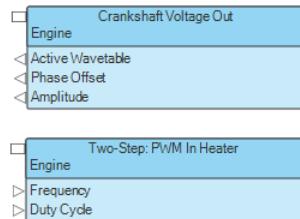
In a multimodel application, data imports can also be used to input values from a data outport in another behavior model (= [model communication](#)).

**Data output** This data port outputs values of the behavior model for processing by a function.

In a multimodel application, data outports can also be used to output values for a data import in another behavior model (= model communication).

**Configuration port** A specific outport that reads configuration settings defined in the behavior model and makes them available to the mapped function port.

Data ports of the same data direction are combined in one model port block. The following illustration shows three data outports combined in a model port block and two data imports also combined in a model port block:



**Runnable function port** This model port is provided by a Runnable Function block. It can be mapped to the event port of a function block in order to model an asynchronous task.

#### Model port properties

The values of the model port properties are affected by the data interchange between ConfigurationDesk and the behavior model. For details, refer to [Data Interchange Between ConfigurationDesk and Behavior Model](#) on page 420. You can enter a name and a description for ports of model port blocks that were created in ConfigurationDesk, and that are still unresolved.

##### Note

Properties of model port blocks that are part of a behavior model cannot be configured in ConfigurationDesk. To change these properties, use the corresponding model port block in the behavior model.

#### Possible states of model ports

Each port in ConfigurationDesk has a particular state. The following table shows the possible states, how they are displayed and their effects on code generation during the build process.

State	Display of Port Symbol <sup>1)</sup>	Description	Effect on Code Generation
OK	The port symbol is black.	There are no conflicts.	Code is generated.
Obsolete	The port symbol is orange. This indicates a warning conflict.	The port is not contained in the behavior model to which the active application is linked, but the corresponding model port block is resolved. An obsolete port is mapped to at least one function port. If you delete the mapping line from the	The port is not considered during code generation. A warning message is displayed during the build process.

State	Display of Port Symbol <sup>1)</sup>	Description	Effect on Code Generation
		application, the obsolete port is also deleted. Background: The mapping is not deleted immediately, because this simplifies the remapping of the corresponding function block.	

<sup>1)</sup> To highlight obsolete ports, you have to set the appropriate highlighting option. For instructions, refer to [Highlighting Signal Chain Conflicts](#) on page 194.

#### Note

The port states do not affect the mapping of the model ports. You can add or delete mapping lines regardless of the port state.

#### Display of protocol setting for data imports

If the protocol of the according communication package is set to Blocking, a rectangle is added to the port symbol of data imports:



For details, refer to [Setting Up Model Communication](#) on page 448.

#### Related topics

##### Basics

[Characteristics of Model Port Blocks](#)..... 426

##### HowTos

[How to Access Model Port Block and Model Port Data](#)..... 466

##### References

[Model Port Properties \(ConfigurationDesk User Interface Reference\)](#)

# Adding Model Port Blocks to the Signal Chain

## Introduction

To create real-time applications, model port blocks must be made available in the signal chain. You can do this by adding model port blocks to the signal chain and mapping them to function blocks.

## Where to go from here

### Information in this section

Basics on Adding Model Port Blocks to the Signal Chain.....	432
How to Add Model Port Blocks to the Signal Chain via Function Blocks.....	435
How to Add Model Port Blocks to the Signal Chain via the Model Browser.....	436

## Basics on Adding Model Port Blocks to the Signal Chain

### Introduction

ConfigurationDesk provides two methods for adding [model port blocks](#) to the [signal chain](#): via function blocks and via the [Model Browser](#).

### Adding model port blocks via function blocks

You can let ConfigurationDesk add model port blocks to the signal chain by using the following commands:

- Generate New Simulink Model Interface
- Propagate to Simulink Model
- Propagate to ConfigurationDesk Model Interface
- Extend Signal Chain - Create Suitable Model Port Blocks

#### Note

This method is not possible for Configuration Port blocks. They can only be added to the signal chain via the [Model Browser](#). For details, refer to [How to Add Model Port Blocks to the Signal Chain via the Model Browser](#) on page 436.

Suitable model port blocks are created according to the function block's port types. Their data types depend on the setting of the function block type's [Model port data type](#) option. Refer to [Specifying Options for Creating Model Port Blocks](#) on page 421. Other model port properties such as port width and unit are taken from the corresponding function port.

For all created model ports, ConfigurationDesk automatically maps them to the function ports from which they were created. This means that you do not have to map ports yourself, so you avoid mapping errors. ConfigurationDesk performs mapping according to specific mapping rules. For details on mapping rules, refer to [Rules for Model Port Mapping](#) on page 443.

#### Tip

ConfigurationDesk lets you specify settings that are used for creating model port blocks in ConfigurationDesk, for example, via a propagate operation. Refer to [Specifying Options for Creating Model Port Blocks](#) on page 421.

#### Adding model port blocks via the Model Browser

You can add model port blocks to the signal chain by dragging them from the Model Browser into a working view.

#### Note

You can add the same model port block to several working views. Note that it is not used in your application several times but only once.

If your signal chain already contains function ports, you can connect the model ports to them by drawing mapping lines in a working view. ConfigurationDesk allows only mapping lines which agree with specific mapping rules. For details, refer to [Rules for Model Port Mapping](#) on page 443.

#### Adding model port blocks for Simulink models via the Model-Function Mapping Browser

In the Model-Function Mapping Browser, you can add model port blocks to the signal chain for the work with Simulink behavior models via drag & drop. Subsequently, you can propagate the newly created model port blocks directly to the Simulink behavior model. Refer to [Working with Simulink Behavior Models](#) on page 531.

#### Naming of model port blocks and model ports

The names of model port blocks and model ports are displayed on the graphical user interface (Model Browser, working views, tables, and Properties Browser) and in information and error messages.

ConfigurationDesk names model port blocks according to the following rules:

- A model port block which is or was contained in the behavior model that is linked to your application is named after the corresponding element in the behavior model. E.g., for Simulink behavior models, the model port blocks in ConfigurationDesk are named after the corresponding model port block in the Simulink behavior model.
- Model port blocks which are created in ConfigurationDesk by adding them to the signal chain via function blocks are named as follows:
  - If only one model port block is created for a function block, the function block name is used.

- The name of the model port block is derived from the name of the function and the path of the function within the function block structure. If the function block contains just one function, the model port block is named according to the function block name.
- An `_In/_Out` suffix is added to the original function (or function block) name if two model port blocks are created for a single function block.
- Generated structured model ports are named according to the corresponding function port groups.

If you want to change the names of model port blocks, refer to [How to Rename Model Port Blocks](#) on page 468.

#### Note

ConfigurationDesk does not support Unicode characters in model port block names. If model port blocks are created via the Extend Signal Chain command, the model port block names are derived from the associated function block names. In this case, make sure that no Unicode characters are used in function block names.

#### Usage of model port blocks

In ConfigurationDesk, model port blocks are not necessarily all used in the active application. This is visualized as follows:

Usage	Display in Model Browser	Description
Used model port blocks		They are used in the signal chain of your active application.
Unused model port blocks		They are not used in the signal chain of your active application.

#### Deleting model port blocks

Model port blocks which are used in an application can be removed from it. You can right-click the block, for example, in a working view or in a table, and select **Delete from Application** in the context menu. The model port block then disappears from the model section in the working view and no longer has the  symbol in the Model Browser.

#### Related topics

#### HowTos

[How to Add Model Port Blocks to the Signal Chain via Function Blocks.....](#) 435

How to Add Model Port Blocks to the Signal Chain via the Model Browser..... 436

#### References

Delete from Application (ConfigurationDesk User Interface Reference )  
Extend Signal Chain – Create Suitable Model Port Block (ConfigurationDesk User Interface Reference )

## How to Add Model Port Blocks to the Signal Chain via Function Blocks

<b>Objective</b>	You can easily extend your signal chain by adding <a href="#">model port blocks</a> via <a href="#">function blocks</a> .
------------------	---

<b>Precondition</b>	The following preconditions must be fulfilled:
---------------------	--

- A working view must be open.
- Function blocks must be inserted in the signal chain.

<b>Method</b>	<b>To add a model port block to the signal chain via function blocks</b>
---------------	--

- 1 In the working view, right-click the function block (or a contained function group or function) you want to add a model port block to.

#### Tip

You can highlight several functions, function groups, or function blocks for multiselection.

- 2 Depending on your use scenario, select one of the following commands from the context menu:
  - Propagate to ConfigurationDesk Model Interface
  - Extend Signal Chain - Create Suitable Model Port Blocks
  - Propagate to Simulink Model
  - Generate New Simulink Model Interface

#### Tip

You can influence the structure and the data type of the model port block(s) to be created by specifying the Model port block structure and the Model port data type properties of the function block type. For details, refer to [Configuration Page \(ConfigurationDesk User Interface Reference !\[\]\(fa79ca182f6cc641dbbcde59a7998310\_img.jpg\)](#)).

**Result**

Depending on your selection, ConfigurationDesk adds one or more model port blocks with a suitable data direction to the signal chain. It also automatically maps the model ports to function ports..

If you selected the **Propagate to Simulink Model** command, suitable model port blocks are created in the Simulink behavior model. If there are multiple Simulink behavior models in your ConfigurationDesk application, the model port blocks are created on the root level of the first Simulink behavior model in the model topology.

If you selected **Generate New Simulink Model Interface**, suitable model port blocks are generated in a new Simulink model. You can use the new Simulink model for modeling a new behavior model, or you can copy the model port blocks from the new Simulink model to the desired Simulink behavior model using the standard **Copy** command together with the **Paste and Keep IDs** command.

By default, the new model port blocks are named according to fixed rules. If the blocks are still unresolved, you can rename them. For more information, refer to [Basics on Adding Model Port Blocks to the Signal Chain](#) on page 432.

The working view displays the new model port blocks in the last position in the Model Port Blocks column.

---

**Related topics**

Basics

[Basics on Adding Model Port Blocks to the Signal Chain](#)..... 432

HowTos

[How to Add Model Port Blocks to the Signal Chain via the Model Browser](#)..... 436

References

[Extend Signal Chain – Create Suitable Model Port Block \(ConfigurationDesk User Interface Reference\)](#)

## How to Add Model Port Blocks to the Signal Chain via the Model Browser

---

**Objective**

If you want to add [model port blocks](#) to your [signal chain](#) manually, you must use the [Model Browser](#).

**Preconditions**

The following preconditions must be fulfilled:

- A working view must be open.
- A model topology must be loaded.

**Method****To add a model port block to the signal chain via the Model Browser**

- 1 In the Model Browser, select the model port block you want to add to the signal chain.

**Tip**

You can press and hold down the **Ctrl** key for selective multiselection.

- 2 Drag the model port block to the working view and drop it.

**Result**

ConfigurationDesk adds the selected model port block to the signal chain. The working view displays it in the model section at the position where you dropped it. The Model Browser displays it as used, indicated by the following symbol next to the model port block name: .

**Next step**

You can now map the added model port(s) to function port(s). For details, refer to [Rules for Model Port Mapping](#) on page 443.

**Related topics****Basics**

[Basics on Adding Model Port Blocks to the Signal Chain](#)..... 432

**HowTos**

[How to Add Model Port Blocks to the Signal Chain via Function Blocks](#)..... 435

**References**

[Extend Signal Chain – Create Suitable Model Port Block \(ConfigurationDesk User Interface Reference\)](#) 

# Model Port Mapping

## Where to go from here

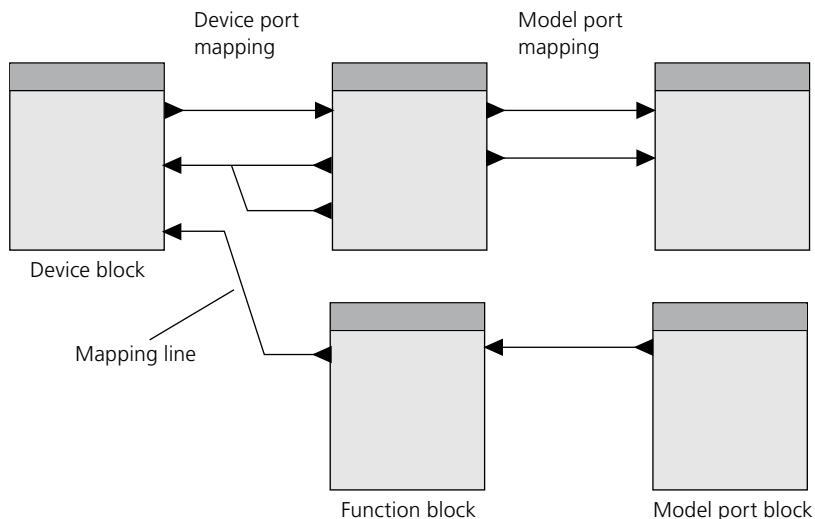
## Information in this section

Methods for Model Port Mapping.....	438
Rules for Model Port Mapping.....	443
Mapping States and Their Effects.....	445
Mapping Algorithm for Model Port Blocks With Structured Data Ports.....	446

## Methods for Model Port Mapping

### Definition

Mapping ports means, drawing a mapping line between two ports to connect them in the signal chain. Model port mapping is the mapping of ports as shown below.



Model port blocks are mapped to the function blocks according to specific mapping rules. Refer to [Rules for Model Port Mapping](#) on page 443.

### Mapping methods

ConfigurationDesk provides different methods for model port mapping.

**Implicit automatic mapping by ConfigurationDesk** If you select one of the following commands from a function block's context menu, ConfigurationDesk automatically creates [model port blocks](#) which are mapped to the [function block](#):

- Generate New Simulink Model Interface
- Propagate to Simulink Model
- Propagate to ConfigurationDesk Model Interface
- Extend Signal Chain - Create Suitable Model Port Block

**Creating mapping lines by port name** You can easily map function ports and model ports if the port names match.

- *I/O <-> Model*: In the Signal Chain Browser, right-click a free area or a selection containing unmapped function ports and model ports with matching names and select Create Mapping Lines by Name – IO <-> Model from the context menu.
- *Model <-> Model*: In the Signal Chain Browser or the Model Communication Browser, right-click a free area or a selection containing unmapped model ports from different models with matching names and select Create Mapping Lines by Name – Model <-> Model from the context menu.

ConfigurationDesk maps ports with matching names (function ports to model ports or model ports to model ports) according to the mapping rules.

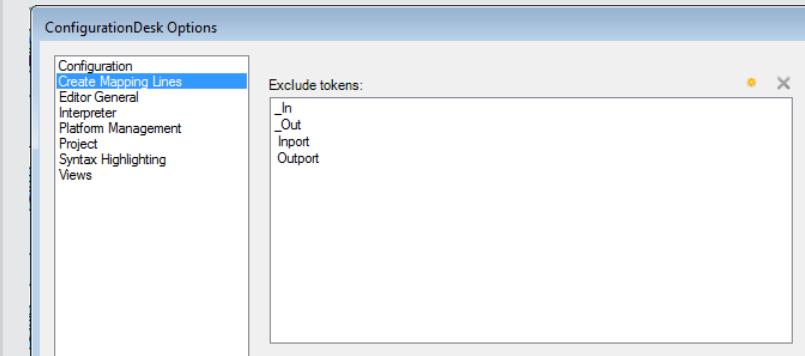
If there is more than one possible match, ConfigurationDesk tries to determine the best one by going through the higher-level elements (e.g., function port groups or structured data ports). In this case, ConfigurationDesk starts at the ports and compares their names up to the related function block or model port block. The largest number of identically named hierarchy elements is considered to be the best match.

**Note**

- If you execute the command for a complete working view after right-clicking a free area, the function block and model port block names have to match for a mapping to be created.
- ConfigurationDesk creates as many mappings as possible according to the best match principle.
- Existing mappings are not changed.
- Configuration ports are ignored.
- Conflicts caused by created mappings are ignored.

**Tip**

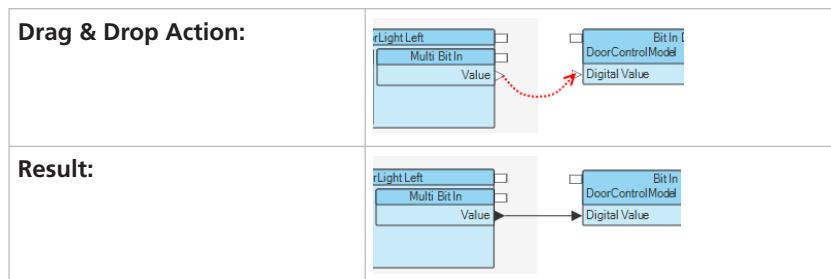
On the Create Mapping Lines page of the ConfigurationDesk Options dialog, you can define exclude tokens, i.e., strings that are ignored when ConfigurationDesk tries to determine a best match. A number of strings are already excluded by default:



Use the and buttons to add or remove tokens.

**Manual mapping (two ports)** You can draw a [mapping line](#) and map two ports manually. ConfigurationDesk enforces the mapping rules and allows only mapping lines which agree with them.

Click one port and move the cursor to the other port while holding the mouse button down:

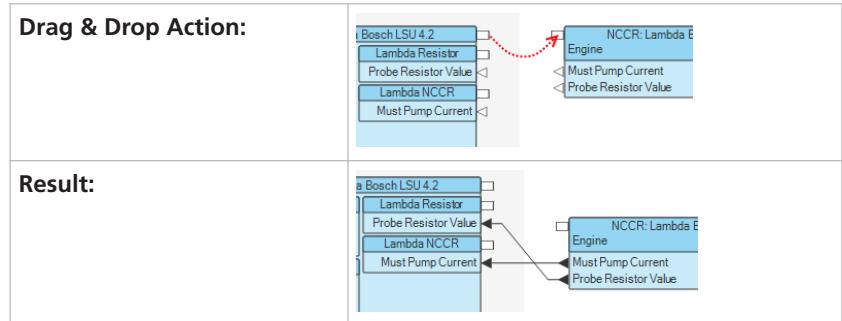


**Tip**

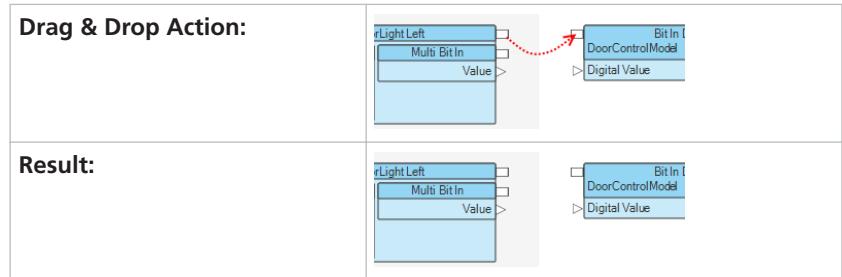
As an alternative, you can map ports in the signal chain by clicking one port, pressing **Ctrl+Alt** and afterwards clicking another port.

**Manual mapping (multiple ports)** Every function block, function group, function, function port group, model port block, and model port group has a parent port to make manual mapping of multiple ports easier. You can connect a parent port to another parent port in order to create mapping lines between function and model ports, and for model communication. All child ports with the same name are mapped. ConfigurationDesk enforces the mapping rules and allows only mapping lines which agree with them. For more information, refer to [Mapping Algorithm for Model Port Blocks With Structured Data Ports](#) on page 446.

Click one parent port and move the cursor to the other parent port while holding the mouse button down:



The port names must be identical. Otherwise no mapping lines are created:



### Model port mapping via drag & drop

As an alternative to manual mapping, ConfigurationDesk lets you perform model port mapping via drag & drop, i.e., instead of dragging a mapping line from one parent port to another, you can directly drag one item from a working view and drop it to another. You can also drag items from topology browsers or tables to target items in a working view. As for mapping of multiple ports, the port names must be identical. The mapping lines are then created automatically according to the rules for manual mapping of multiple ports (see above).

**Note**

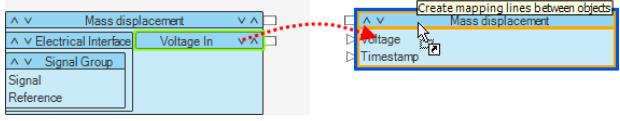
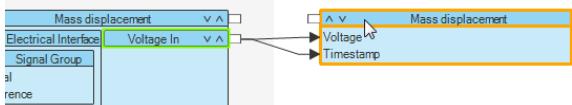
- You cannot multiselect items to drag them to target items.
- The target item of the drag & drop operation must be an element of a working view.

**Examples for model port mapping via drag & drop**

Dragging a function block to a target model port block in a working view:

<b>Drag &amp; Drop Action:</b>	
<b>Result:</b>	

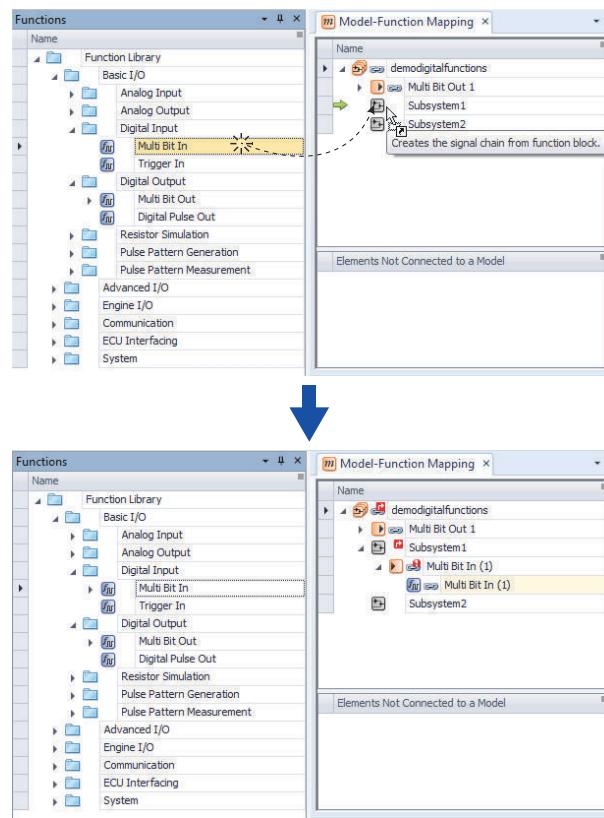
Dragging a collapsed function to a target model port block in a working view:

<b>Drag &amp; Drop Action:</b>	
<b>Result:</b>	

It is also possible to drag a function block type from the Function Browser, for example, to a model port block in a working view. ConfigurationDesk then creates an instance of the function block type and maps its ports to the ports of the model port blocks.

**Mapping via the Model-Function Mapping Browser**

Via the Model-Function Mapping Browser, you can instantiate new function blocks, create suitable model port blocks, and map them to the function blocks in one step via drag & drop. To do so, you must drag a function block type from the Functions Browser to a subsystem or the root level of the desired Simulink behavior model. The following illustration shows an example:



For details, refer to [Working with Simulink Behavior Models](#) on page 531.

#### Effects on calculation of wiring information

The model port mapping is not considered during the calculation of wiring information for the external cable harness.

#### Related topics

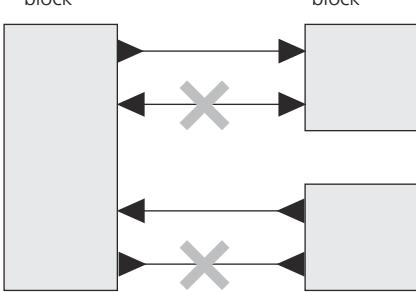
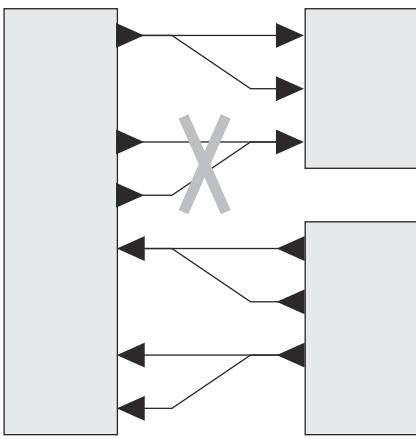
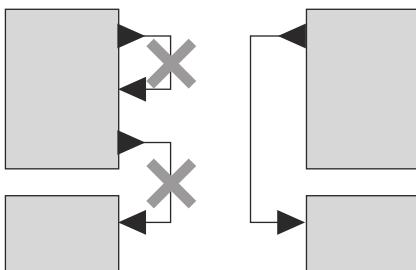
##### Basics

Collapsing and Expanding Blocks.....	189
Rules for Model Port Mapping.....	443

## Rules for Model Port Mapping

#### Mapping rules

Model port mapping can be performed only if specific rules are observed. ConfigurationDesk allows lines to be mapped only if they agree with these rules. The rules are visualized below.

Mapping Rules	Example Illustrations
<p>Function ports and model ports can be mapped only if they have the same data direction in the signal chain, i.e.:</p> <ul style="list-style-type: none"> <li>▪ Function outports can be mapped to model inports.</li> <li>▪ Function imports can be mapped to model outports.</li> <li>▪ Event ports can be mapped to runnable function ports.</li> </ul>	
<p>Multiple mapping:</p> <ul style="list-style-type: none"> <li>▪ Data imports (of model port blocks) can be mapped only once, i.e., data imports can only have one input source. Data outports can be mapped multiple times.</li> <li>▪ If a function port is mapped to several model ports, this can result in multiple I/O triggering of the function. For the effect of this, refer to <a href="#">Basics on Function Triggers</a> on page 328.</li> </ul>	
<p>Mapping to equal block types:</p> <ul style="list-style-type: none"> <li>▪ A function port cannot be mapped to another function port.</li> <li>▪ A model port can be mapped to another model port (model communication).</li> </ul>	

Mapping Rules	Example Illustrations
<p>Model port blocks that reside in different behavior models must not be mapped to the same function block if the related behavior models are assigned to different application processes. If you do so, a <b>Multiple assignments to different application processes conflict</b> appears in the <b>Conflicts Viewer</b>.</p>	

**Related topics****Basics**

Characteristics of Function Ports.....	303
Characteristics of Model Ports.....	429

## Mapping States and Their Effects

**Mapping states**

The mapping between [function ports](#) and [model ports](#) can have the following states:

Status	Display of Mapping Line <sup>1)</sup>	Description	Effect on Code Generation
OK	Black	The mapping has no conflicts.	Code is generated without warnings.
Warning	Orange	The data ranges of the mapped ports are unequal.	<p>Code is generated. Warning messages are displayed during the build process. ConfigurationDesk takes only the cut set of vector elements depending on the data direction:</p> <ul style="list-style-type: none"> <li>▪ If the data range of the source is greater than the sink: Dispensable elements are ignored.</li> <li>▪ If the data range of the source is smaller than the sink: Missing elements are filled in with 0.</li> </ul>
Warning	Orange	Model port blocks that belong to different application processes are mapped to the same function block.	The function block is not part of a valid signal chain and is considered part of

Status	Display of Mapping Line <sup>1)</sup>	Description	Effect on Code Generation
			the default application process. As a result, only its initial code is generated.

<sup>1)</sup> To display colored lines, you have to set the appropriate display filter. For instructions, refer to [Highlighting Signal Chain Conflicts](#) on page 194.

## Related topics

### Basics

[Rules for Model Port Mapping](#)..... 443

## Mapping Algorithm for Model Port Blocks With Structured Data Ports

### Introduction

ConfigurationDesk allows you to map the [parent ports](#) of

- Two model port blocks with structured data ports
- A model port block with structured data ports and a function block
- Any substructures contained in model port blocks and function blocks

### Mapping algorithm

If you map two parent ports, ConfigurationDesk applies the following mapping rules:

- Imports are mapped to outports, outports are mapped to imports. Configuration ports as well as already mapped model imports are ignored.
  - All child ports with identical names are mapped.
- If there is more than one possible match, ConfigurationDesk tries to determine the best one by walking through the higher-level elements (function port groups or structured data ports). In this case, ConfigurationDesk starts at the ports and compares their names up to the related function or model port block. The largest number of identically named hierarchy elements is considered to be the *best match*.

#### Note

In the case of ambiguities, the block structure is analyzed, and the ports with the most closely matched paths are mapped. A mapping is created only, if the *best match* principle leads to a definite decision. All ports, for which a definite decision cannot be made, remain unaffected.

- Mapping lines that cause a warning during the build process (for example, due to incompatible data widths) are created. In this case, a conflict is shown in the Conflicts Viewer. You have to solve the conflict before you start the build process.

- Mapping lines that would cause an error during the build process are not created.

If structured model ports or function ports do not match, you have to draw the mapping lines manually.

**Note**

- The mapping algorithm does not update or delete existing mapping lines.
- Mapping lines including obsolete ports are considered as valid.
- Mapping model ports in an unfavorable way might reduce the run-time performance. For optimum run-time performance, it is recommended to observe specific mapping rules. Refer to [Model Communication Recommendations for Model Port Blocks with Structured Data Ports](#) on page 464.

# Setting Up Model Communication

## Introduction

If the model topology in your ConfigurationDesk application consists of more than one model (multimodel application), the models can exchange signal data. ConfigurationDesk lets you set up and configure the model communication between these models.

## Where to go from here

### Information in this section

Basics on Model Communication.....	448
Configuring Data Snapshots.....	451
Basics on Function Modules (Advanced).....	454
Ensuring Data Consistency in Model Communication (Advanced).....	455
How to Create Communication Packages.....	459
How to Assign Data Import Blocks to Other Communication Packages.....	460
How to Change the Protocol of Communication Packages.....	461
Working with Model Port Blocks That Provide Variable-Size Signals.....	462
Model Communication Recommendations for Model Port Blocks with Structured Data Ports.....	464

## Basics on Model Communication

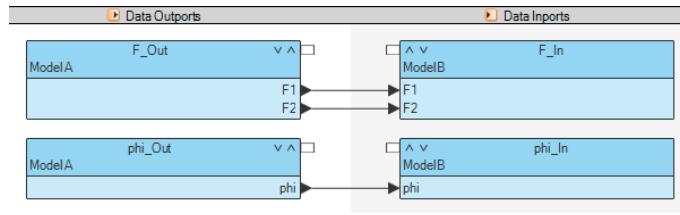
### Setting up model communication

If the [model topology](#) in your application consists of more than one model, the models can exchange signal data via [model port blocks](#).

To set up [model communication](#) in ConfigurationDesk, you must use a mapping line to connect a [data outport](#) (sending model) to a [data import](#) (receiving model).

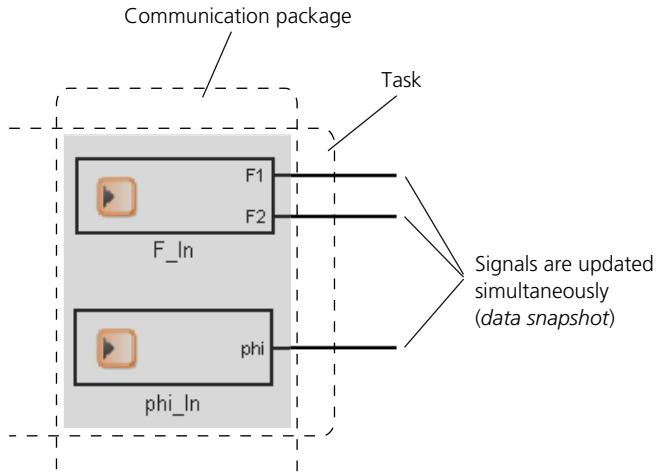
#### Tip

The best way to set up model communication is using the [Model Communication Browser](#), where the Data Import and Data Outport blocks are shown in two different columns. For details, refer to [How to Open Working Views in the Model Communication Browser](#) on page 178.



### Communication packages, tasks, and data snapshots

Communication between models is based on communication packages. A communication package bundles Data Import blocks. Hence, it also bundles the signals that are received by these blocks. If Data Import blocks are executed within the same task and belong to the same communication package, their data imports are read simultaneously. If Data Outport blocks that are connected to the Data Import 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*). ConfigurationDesk lets you create and configure these communication packages. In the illustration below, the Data Import blocks belong to the same task and the same communication package.



Bundling in communication packages not only ensures consistency. A further advantage is that several signals are sent together, not individually. This causes less overhead, and thus increases performance.

**Standard Communication Package** By default, in ConfigurationDesk all the Data Import blocks that are involved in model communication are assigned to the Standard Communication Package. You cannot rename or delete the Standard Communication Package. In the example below, both the F\_In Data Import block in the ModelB model and the phi\_In Data Import block in the ModelB model belong to the Standard Communication Package.

Name	Protocol	Model	Connected Model
Standard Communication Package	Non-blocking		
phi_In		ModelB	ModelA
phi		ModelB	ModelA
F_In		ModelB	ModelA
F2		ModelB	ModelA
F1		ModelB	ModelA

**User communication packages** You can create your own communication packages and assign Data Import blocks to them. You can rename and delete user communication packages. If you delete a user communication package, its blocks are automatically reassigned to the Standard Communication Package.

Name	Protocol	Model	Connected Model
Standard Communication Package	Non-blocking		
F_In		ModelB	ModelA
Model Communication Package (1)	Non-blocking		
phi_In		ModelB	ModelA

For information on the configuration of data snapshots, refer to [Configuring Data Snapshots](#) on page 451.

#### Ensuring data consistency

An optimum of data consistency is ensured within a communication package. If necessary, ConfigurationDesk bundles the signals within a communication package in further internal packages. This ensures an optimum of data consistency for cases when the Data Import blocks or the connected Data Outport blocks are located in different tasks, or in the same task in different *function modules*.

For details, refer to the following topics:

- [Basics on Function Modules \(Advanced\)](#) on page 454
- [Ensuring Data Consistency in Model Communication \(Advanced\)](#) on page 455

#### Avoiding errors

To avoid errors, ConfigurationDesk assists you throughout the development stages:

**Automatic actions during configuration** ConfigurationDesk performs the following actions automatically:

- If you delete a user communication package that contains Data Import blocks, these blocks are automatically moved into the Standard Communication Package.

#### Note

The Standard Communication Package cannot be deleted.

- If you move the last remaining Data Import block out of a communication package, the package still exists, but is ignored during the build process.
- If you cut all mapping lines used for model communication off a Data Import block, the block is automatically removed from its communication package.

**Warnings during the build process** During the build process, ConfigurationDesk checks the validity of the model communication. Invalidities are displayed in the Build Log window, for example:

```
Checking model communication ...
Connected data output of data import 'phi' of block 'phi_In' is
'Unresolved'. No model communication code is generated for this port.
```

**Automatic actions during simulation** During simulation, signal incompatibilities are handled as follows:

- A signal is always cast to the type of the receiving Data Import.
- If the width of a vector signal is greater than the width N of the receiving Data Import, only the first N vector elements of the signal are passed through the port.

#### Note

This strategy leads to a loss of data.

- If the width of a vector signal is less than the width N of the receiving Data Import, the unfilled signal vector elements are set to the port's default values.

#### Related topics

##### Basics

[Configuring Data Snapshots.....](#) 451

##### HowTos

[How to Change the Protocol of Communication Packages.....](#) 461

## Configuring Data Snapshots

#### Introduction

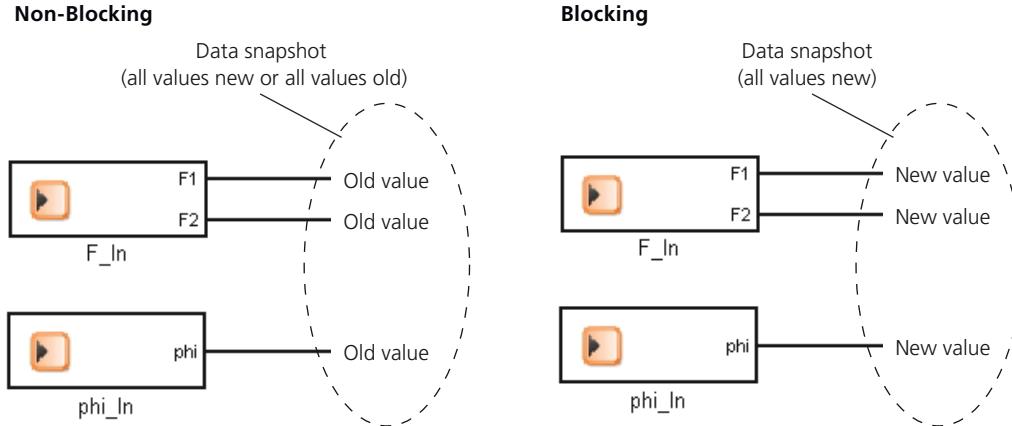
Communication packages<sup>②</sup> guarantee simultaneous signal updates within a running task (*data snapshot*). ConfigurationDesk lets you configure these data snapshots.

#### Data snapshot contents with different protocols

Communication packages affect the timing of the data snapshot via their Protocol, which you can set to either Non-Blocking or Blocking. The default is Non-Blocking.

In the illustration below, it is assumed that:

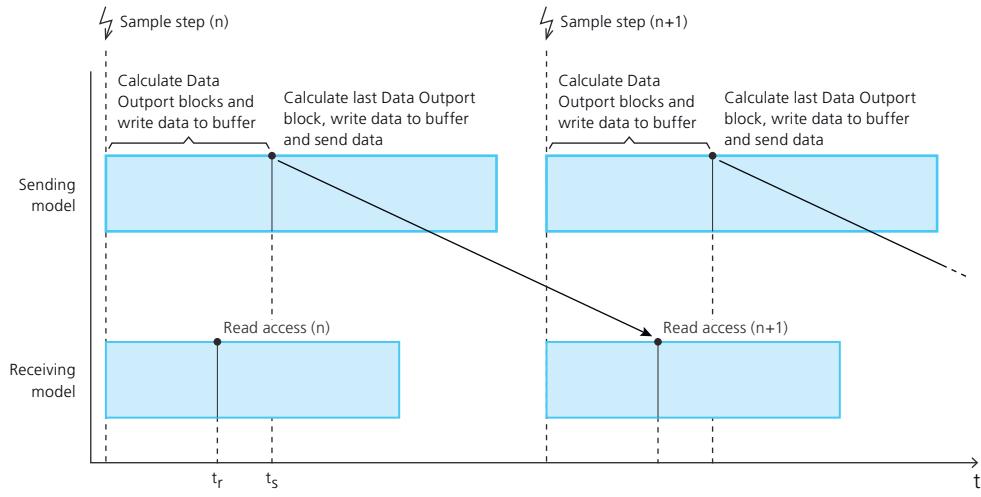
- the Data Import blocks belong to the same task, the same function module and the same communication package and
- the connected Data Outport blocks belong to the same task and the same function module



Independently of the specified protocol, the connected Data Outport blocks send their data together as one data package as soon as the last Data Outport block is executed. The data snapshot is taken as follows:

**Non-Blocking** The data snapshot for all Data Import blocks that are part of the same communication package is taken as soon as the first read access to a signal occurs, i.e., as soon as the first of the Data Import blocks is executed. This means:

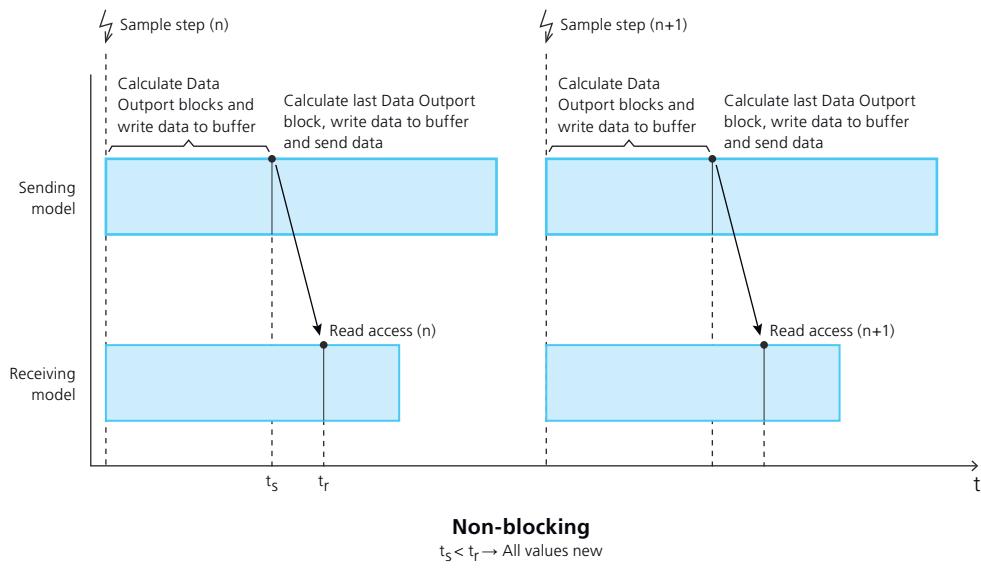
- If the first read access occurs *before* the last Data Outport block has triggered the sending of the data (sending time  $t_s$  is later than receiving time  $t_r$ ), the Data Import blocks read consistent values from the previous sample step (old values). This is shown in the illustration below.



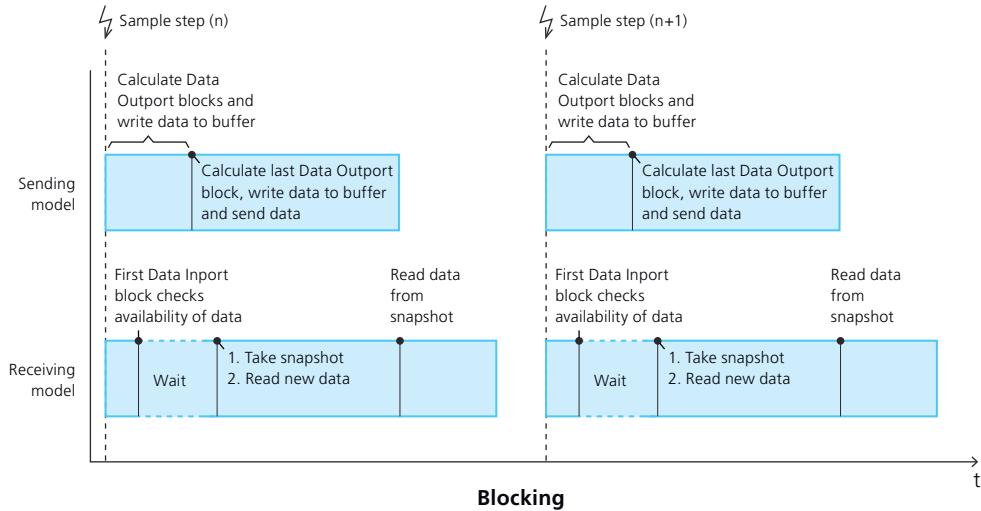
#### Non-blocking

$t_s > t_r \rightarrow$  All values old

- If the first read access occurs *after* the last Data Outport block has triggered the sending of the data (sending time  $t_s$  is earlier than receiving time  $t_r$ ), the Data Import blocks read consistent values from the actual sample step (new values). This is shown in the illustration below.



**Blocking** The data snapshot is taken after all the signals provide new values. When the first of the **Data Import** blocks that are part of the same communication package is executed, it checks if new values for all **Data Imports** are available. The calculation of the receiving model then waits until new data is available. This is shown in the illustration below.



#### Note

Waiting for new signal values can cause task overrun. For details, refer to [Basics on Task Overruns](#) on page 483.

**Related topics****HowTos**

How to Change the Protocol of Communication Packages..... 461

## Basics on Function Modules (Advanced)

**Introduction**

Within the [communication packages](#), ConfigurationDesk automatically combines the Data Import blocks into further communication packages to optimize data consistency. The signals are assigned to these communication packages according to their assignment to *function modules*.

**Note**

The following description applies to Simulink behavior models.

**What is a function module in a Simulink model?**

In a Simulink model, blocks with the following conditions build a function module:

**Data port blocks (from the Model Interface Blockset) with the same sample time** A function module consists of all the blocks in the model that have the same sample time, if at least one data port block has the following characteristics:

- The block's sample time matches the sample time.
- The block is not contained in an atomic subsystem.

If these conditions are fulfilled, all the data port blocks with the same characteristics are assigned to the function module.

**Data port blocks (from the Model Interface Blockset) in atomic subsystems with the same sample time** A function module consists of all the blocks in an atomic subsystem that have the same sample time, if at least one data port block has the following characteristics:

- The block's sample time matches the sample time.
- The block is contained in the atomic subsystem. The relevant atomic subsystem is the one directly above the data port block. For example, in the hierarchy `Model/Atomic1/Virtual1/Atomic2/Virtual2/MyBlock` the `MyBlock` data port block defines a function module for the `Atomic2` subsystem, not for `Atomic1`.

If these conditions are fulfilled, all the data port blocks with the same characteristics are assigned to this function module.

**Simulink Import/Outport blocks with the same sample time** A function module can consist of all the blocks in the model that have the same sample time, if at least one Simulink Import/Outport has the following characteristics:

- The block's sample time matches the sample time.
- The block resides on the root level of the model.

If these conditions are fulfilled, all the Simulink Imports/Outports with the same characteristics are assigned to the function module.

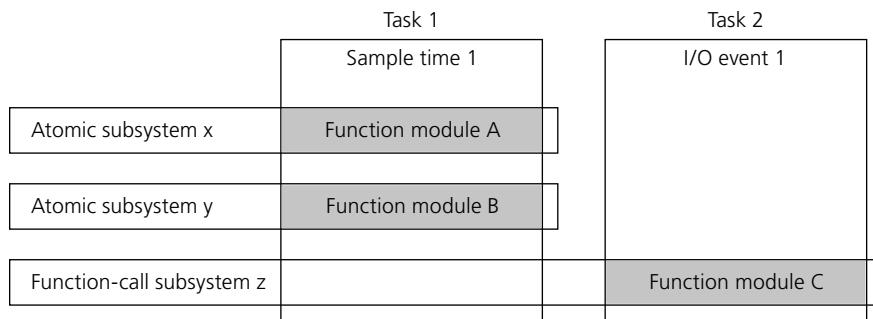
#### Note

Data port blocks not contained in atomic subsystems and Simulink Imports/Outports on the root level of the behavior model define separate function modules, even if the blocks' sample times are identical.

#### Example of function modules

Suppose your behavior model contains three atomic subsystems, and each subsystem contains at least one data port block. Two of these atomic subsystems are calculated with the same Sample time 1. The third atomic subsystem is a function-call subsystem connected to a Hardware-Triggered Runnable Function block. In ConfigurationDesk, the associated runnable function is assigned to a task that is triggered by an I/O event.

Code generation would result in Function module A and Function module B, both of which are part of Task 1. Function module C is part of asynchronous Task 2.



## Ensuring Data Consistency in Model Communication (Advanced)

#### Introduction

Within the [communication packages](#), ConfigurationDesk automatically combines the Data Import blocks into further communication packages to optimize data consistency.

**Note**

The following description applies to Simulink behavior models.

**Function module communication packages and consistency buffers**

ConfigurationDesk automatically assigns the signals to further communication packages according to their assignment to function modules. For details, refer to [Basics on Function Modules \(Advanced\)](#) on page 454. These communication packages are called function module communication packages below. Each function module communication package contains one or more consistency buffers.

**Function module communication package** A communication package in ConfigurationDesk (Standard Communication Package or user communication package) internally comprises 1..n function module communication package(s). The number and the partitioning of the function module communication packages in a communication package is equal to the number of function modules that have at least one Data Import block of the communication package assigned.

**Note**

The data snapshot mechanism (refer to [Basics on Model Communication](#) on page 448) applies for each function module communication package.

**Consistency buffer** A function module communication package comprises 1..n consistency buffer(s). The number of consistency buffers in a function module communication package results from the number of function modules from which Data Outport blocks send data to Data Import blocks within the function module communication package. Within a consistency buffer, the signals are always consistent, which means that they result from the same sample step.

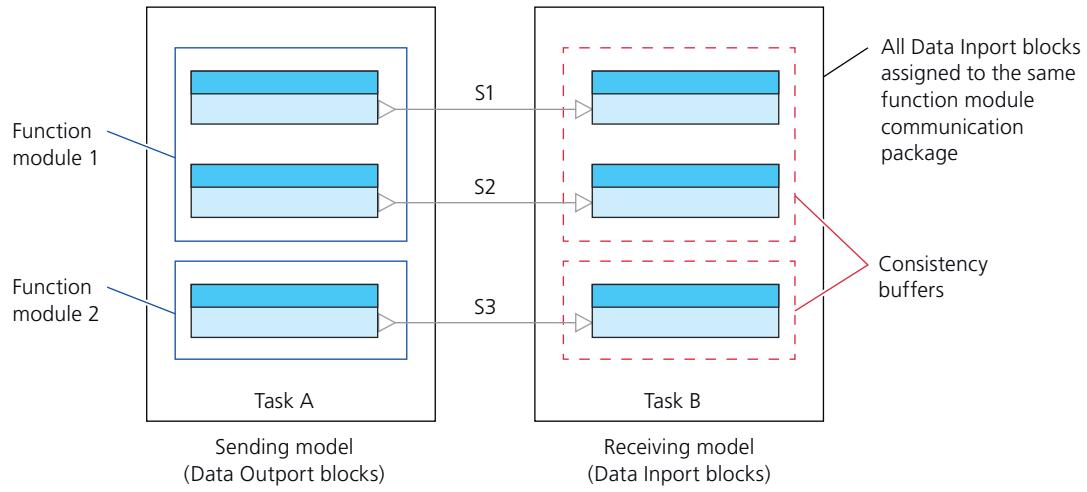
**Note**

The signal data is sent when all data in the consistency buffer is new. Thus, the Data Outport block which is calculated last determines the time at which the data is sent.

**Example with one function module communication package**

In the example below, the following signals form a consistency buffer:

- [S1/S2]
- S3



The protocol you specify for the communication package in ConfigurationDesk influences on the actuality of the data within a consistency buffer, as shown in the table below:

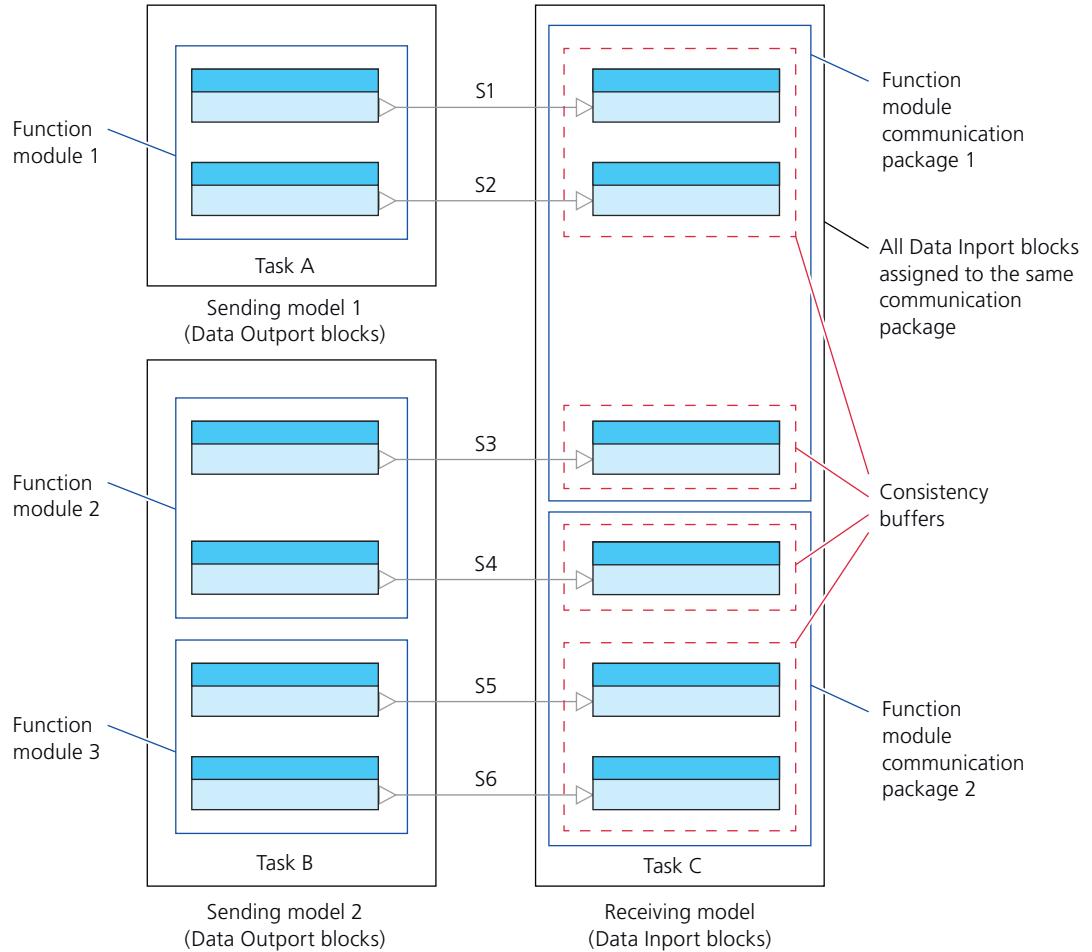
Protocol	Content of Consistency Buffer [S1/S2]	Content of Consistency Buffer S3
Blocking	New data	New data
Non-blocking	Old or new data <sup>1)</sup>	Old or new data <sup>1)</sup>

<sup>1)</sup> Old and new data are not mixed within a consistency buffer.

#### Example with multiple function module communication packages

In the example below, the following signals form a consistency buffer:

- [S1,S2]
- S3
- S4
- [S5,S6]



The following table shows the actuality of the consistency buffer data, depending on the specified protocol:

Protocol	Content of Consistency Buffer [S1,S2]	Content of Consistency Buffer S3	Content of Consistency Buffer S4	Content of Consistency Buffer [S5,S6]
Blocking	New data	New data	New data	New data
Non-blocking	Old or new data <sup>1)</sup>	Old or new data <sup>1)</sup>	Old or new data <sup>1)</sup>	Old or new data <sup>1)</sup>

<sup>1)</sup> Old and new data are not mixed within a consistency buffer.

**Note**

Suppose the Data Import blocks in the receiving model are executed in a task with a higher sample time than the Data Outport blocks in the sending model, and you have specified the Blocking protocol for the communication package. In this case, the signal data within a function module communication package may not result from the same sample step. The data within a consistency buffer, however, always results from the same sample step.

**Related topics****Basics**

Basics on Function Modules (Advanced).....	454
Configuring Data Snapshots.....	451

**References**

Model Communication Package Table (ConfigurationDesk User Interface Reference 
---

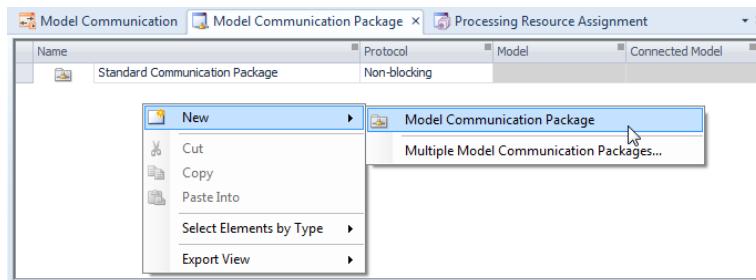
## How to Create Communication Packages

**Objective**

To assign Data Import blocks to a [communication package](#)  other than the Standard Communication Package.

**Method****To create a communication package**

- 1 In the Multiple Models view set, open the Model Communication Package table.
- 2 Right-click the empty area of the table.
- 3 Select New - Model Communication Package.



A new communication package is created.

Name	Protocol	Model	Connected Model
Standard Communication Package	Non-blocking		
Model Communication Package(1)	Non-blocking		

- 4 Rename the user communication package.

**Note**

Communication packages require unique names.

---

**Result**

You have created a new communication package.

---

**Related topics**

**Basics**

Basics on Model Communication..... 448

**References**

Model Communication Package Table (ConfigurationDesk User Interface Reference 

## How to Assign Data Import Blocks to Other Communication Packages

---

**Objective**

To assign the signal(s) of a Data Import block to another [communication package](#) .

---

**Precondition**

- The Model Communication Package table must be open.
- There is at least one user communication package (target package). Refer to [How to Create Communication Packages](#) on page 459.

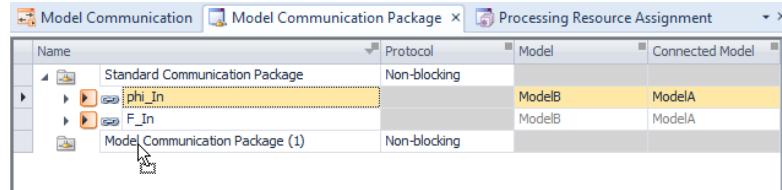
---

**Method**

**To assign a Data Import block to another communication package**

- Select the Data Import block that you want to move (here: phi\_In).
- Drag the block to the new communication package (here: Model Communication Package(1)).

The arrow on the left indicates the drop position.

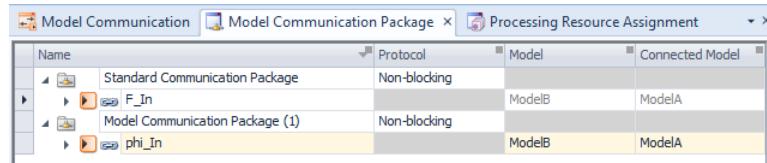


A screenshot of the 'Model Communication' software interface. The window title is 'Model Communication'. Inside, there's a table titled 'Model Communication Package'. The table has columns: 'Name', 'Protocol', 'Model', and 'Connected Model'. There are two rows under 'Standard Communication Package': 'phi\_In' (Protocol: Non-blocking, Model: ModelB, Connected Model: ModelA) and 'F\_In' (Protocol: Non-blocking, Model: ModelB, Connected Model: ModelA). Below these is a row for 'Model Communication Package (1)' (Protocol: Non-blocking). A mouse cursor is shown with an arrow pointing towards the 'phi\_In' row, indicating it is being moved. The 'Connected Model' column for 'phi\_In' is highlighted in yellow.

Name	Protocol	Model	Connected Model
Standard Communication Package	Non-blocking		
phi_In	Non-blocking	ModelB	ModelA
F_In	Non-blocking	ModelB	ModelA
Model Communication Package (1)	Non-blocking		

## Result

You have moved a Data Import block from one communication package to another. Hence, you have assigned the signal(s) of the Data Import block to the second communication package.



A screenshot of the 'Model Communication' software interface, similar to the previous one but showing the result of the move. The 'Model Communication Package' table now shows 'phi\_In' under 'Model Communication Package (1)' (Protocol: Non-blocking, Model: ModelB, Connected Model: ModelA), and 'F\_In' under 'Standard Communication Package' (Protocol: Non-blocking, Model: ModelB, Connected Model: ModelA).

Name	Protocol	Model	Connected Model
Standard Communication Package	Non-blocking		
F_In	Non-blocking	ModelB	ModelA
Model Communication Package (1)	Non-blocking		
phi_In	Non-blocking	ModelB	ModelA

## Related topics

### Basics

[Basics on Model Communication.....](#) 448

### HowTos

[How to Create Communication Packages.....](#) 459

### References

[Model Communication Package Table \(ConfigurationDesk User Interface Reference !\[\]\(00c1e09a6172fa8dcd361da287c7bf5a\_img.jpg\)](#)

## How to Change the Protocol of Communication Packages

### Objective

To change the timing for updating the signals, i.e., for taking the data snapshot.

### Precondition

The Model Communication Package table must be open.

**Method****To change the protocol of a communication package**

- 1 Select the communication package.
- 2 Change the Protocol value of the communication package (here: Model Communication Package(1)):

Name	Protocol	Model	Connected Model
Standard Communication Package	Non-blocking	ModelB	ModelA
F_In	Non-blocking	ModelB	ModelA
Model Communication Package (1)	Non-blocking Non-blocking Blocking	ModelB	ModelA
phi_In		ModelB	ModelA

If the Data Import blocks are executed within the same task and belong to the same communication package, the data snapshot is taken as follows:

- Non-Blocking:

The data snapshot is taken as soon as the first read access to a signal occurs.

**Note**

The signals provide new or old values.

- Blocking:

The data snapshot is taken after all the signals provide new values.

**Note**

Waiting for new signal values can cause task overrun.

**Result**

You have changed the protocol of the communication package.

**Related topics****Basics**

[Basics on Model Communication.....](#) 448

**References**

[Model Communication Package Table \(ConfigurationDesk User Interface Reference](#)

## Working with Model Port Blocks That Provide Variable-Size Signals

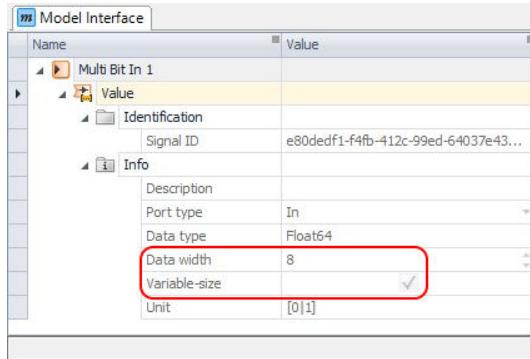
**Introduction**

A Simulink behavior model can provide model ports with variable-size signals. The data width of variable-size signals can vary at run time. Establishing model

communication with variable-size model ports in ConfigurationDesk is subject to specific rules.

### Display of variable-size model ports

In ConfigurationDesk, variable-size model ports are indicated via the Variable-size checkbox. For variable-size model ports, the Data width property displays the maximum number of signal elements. Refer to the following illustration:



In the Model Browser, variable-size model ports are marked with a symbol.

### Valid mappings for variable-size model ports

In a ConfigurationDesk application, you can map variable-size model ports to other variable-size or fixed-size model ports. If you do this, the following preconditions must be fulfilled:

- A variable-size model import can be mapped to a variable-size model output or fixed-size model output.
- The data width of a variable-size model import must be  $\geq$  the data width of the mapped outputs.

#### Note

In the signal chain, you cannot create mapping lines that do not meet the above requirements. Mapping lines between variable-size model port blocks and function blocks are not supported either. ConfigurationDesk displays an error message when you draw such mapping lines. If an existing mapping line is invalid after a model analysis, this mapping line is displayed in red. Refer to the following illustration:



Additionally, a conflict is displayed in the Conflicts Viewer. You must resolve this conflict, before you can start a build process.

**Model communication of structured ports with variable-size signals**

You can map structured model port blocks with variable-size ports to other structured model port blocks with variable-size ports or fixed-size ports to transfer structured signals between behavior models. If you do this, the following applies:

- If the data width of each port matches the data width of the port mapped to it, and no associated conflict is displayed in the **Conflicts Viewer**, the complete structured signal is transmitted during model communication.
- If the mapped ports have different data widths, and no associated conflict is displayed in the **Conflicts Viewer**, the elements of the structured signal are transferred individually.

**Note**

If a mapping line of a structured model port with variable-size signals causes an associated conflict, you must resolve this conflict before you can start a build process.

## Model Communication Recommendations for Model Port Blocks with Structured Data Ports

---

**Introduction**

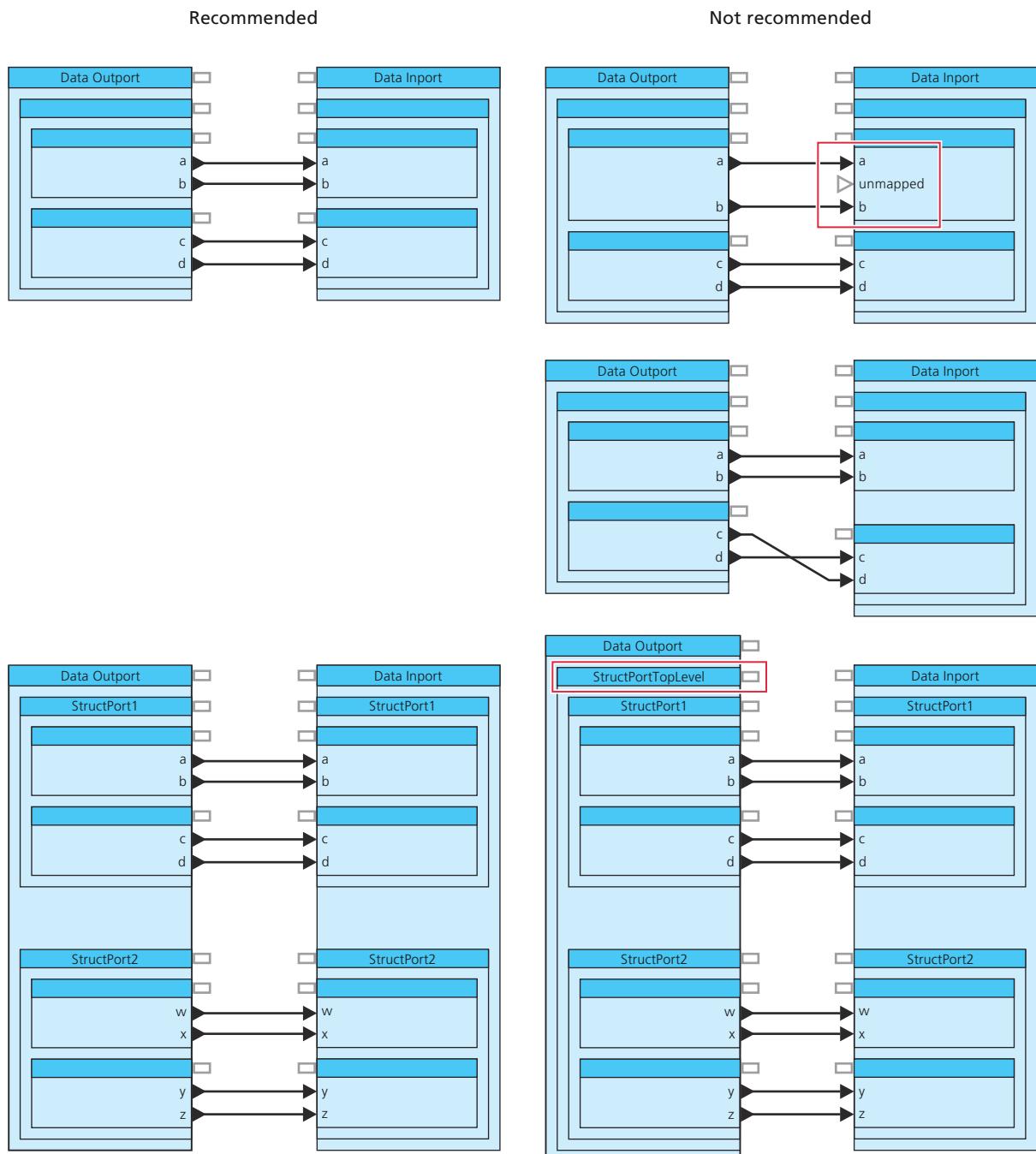
Mapping model port blocks with [structured data ports](#) to model ports of other behavior models might have an impact on the run-time performance. For optimum run-time performance, it is recommended to consider the recommendations below when you set up model communication using model port blocks with structured data ports.

**Recommended mappings for structured data ports**

Note the following mapping recommendations to optimize run-time performance:

- Map all model ports of structured data ports to matching ports, i.e., make sure that the structures of the mapped ports match.
- Make sure that the data types and the data widths of the mapped ports match.
- Avoid cross-connections of model ports.

Refer to the following illustration:



# Configuring Model Port Blocks

## Introduction

ConfigurationDesk makes it easy to access the properties of model port blocks and model ports. You can view and check them, but you cannot configure them. The only exception is that you can change the names of unresolved model port blocks which have been created in ConfigurationDesk via function blocks.

## Where to go from here

### Information in this section

- [How to Access Model Port Block and Model Port Data.....](#) 466
- [How to Rename Model Port Blocks.....](#) 468

## How to Access Model Port Block and Model Port Data

### Objective

To view and check port properties, you must know how to access [model port block](#) and model port data.

### Accessible properties

The properties available for model port blocks and model ports are block and port types, data widths, etc. For a complete description of the accessible properties, refer to [ConfigurationDesk User Interface Reference](#).

### Restriction

Except for the names of unresolved model port blocks which you have created in ConfigurationDesk via function blocks, you cannot change model port block and model port properties in ConfigurationDesk. They can only be changed in the behavior model. For details on model port blocks in Simulink behavior models, refer to [Creating the Interface of Behavior Models \(Model Interface Package for Simulink - Modeling Guide\)](#).

To trace a model port block back from ConfigurationDesk to Simulink, refer to [Switching Between Model Port Blocks in ConfigurationDesk and in Simulink](#) on page 566.

### Precondition

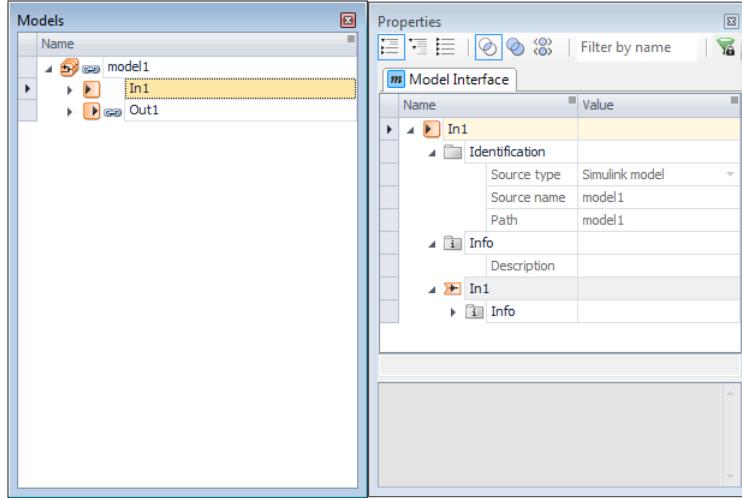
The Properties Browser must be open.

### Method

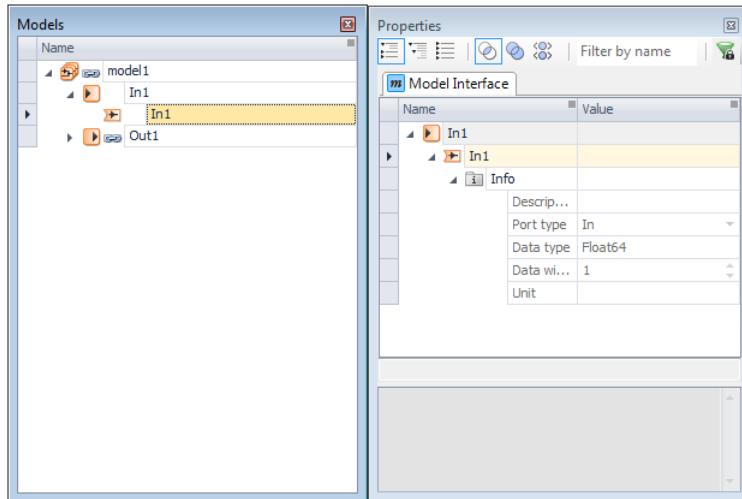
#### To access model port block and model port data

- 1 In the Model Browser, click the model port block whose model port data you want to view.

- 2** The Properties Browser contains the name of the selected model port block.



- 3** In the Model Browser, click the model port whose data you want to view. The port's properties are then displayed in the Properties Browser.

**Result**

You have accessed model port block and model port data.

**Related topics****Basics**

[Characteristics of Model Ports.....](#) 429

**References**

[Model Port Block Properties \(ConfigurationDesk User Interface Reference\)](#)

## How to Rename Model Port Blocks

### Objective

ConfigurationDesk lets you change the names of unresolved [model port blocks](#) which were added to the signal chain via function blocks.

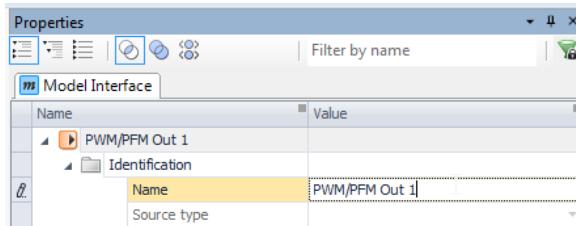
### Precondition

The model port block you want to rename must be created in ConfigurationDesk and must be unresolved.

### Method

#### To rename model port blocks

- 1 Select the model port block that you want to rename.
- 2 In the Properties Browser, click the Name edit field as shown in the following example.



#### Tip

In most panes, you can also double-click the model port block or select it and press **F2** to make the name editable.

- 3 Enter a new name and press **Enter** or click anywhere outside the property value field to confirm.

#### Note

ConfigurationDesk does not support Unicode characters in model port block names. If model port blocks are created via the Extend Signal Chain command, the model port block names are derived from the associated function block names. In this case, make sure that no Unicode characters are used in function block names.

### Result

You renamed a model port block.

#### Tip

You can also rename the model ports of unresolved model port blocks in the same way.

**Related topics**

**Basics**

[Characteristics of Model Port Blocks.....](#) 426

**References**

[Model Port Block Properties \(ConfigurationDesk User Interface Reference\)](#) 



# Modeling Executable Applications and Tasks

## Introduction

To model executable applications in ConfigurationDesk, you must be familiar with executable application terms and their definitions. Modeling executable applications includes modeling and configuring tasks.

ConfigurationDesk lets you use application processes without behavior models, and create real-time applications for multi-processing-unit systems.

## Where to go from here

## Information in this section

Introduction to Modeling Executable Applications and Tasks.....	472
Modeling Asynchronous Tasks.....	502
Configuring Tasks in ConfigurationDesk.....	513
Using Application Processes Without Behavior Models.....	521
Creating Multi-Processing-Unit Applications With ConfigurationDesk.....	525

# Introduction to Modeling Executable Applications and Tasks

## Where to go from here

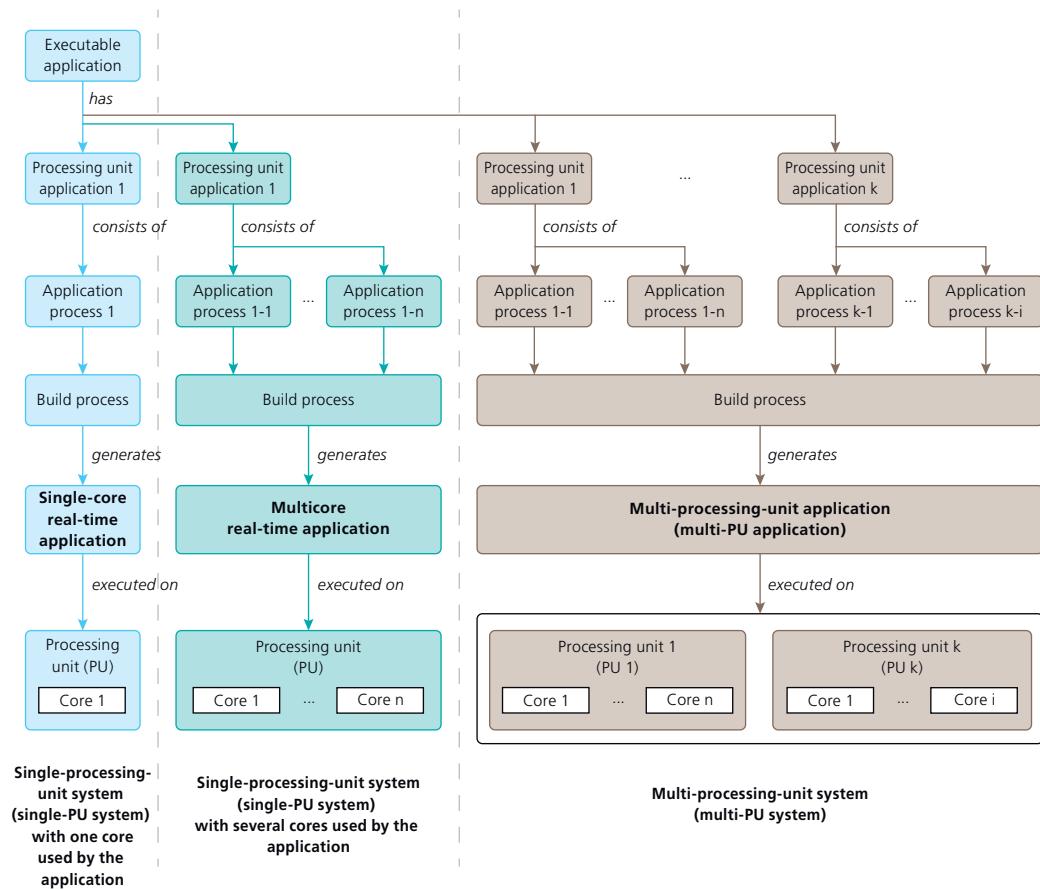
## Information in this section

Terms and Definitions for Building Executable Applications.....	472
Basics on Modeling Executable Applications.....	475
Basics on Tasks, Events, and Runnable Functions.....	480
Basics on Task Overruns.....	483
Basics on Modeling Tasks in ConfigurationDesk.....	486
Different Simulink Tasking Modes and Their Effects in ConfigurationDesk.....	488
How to Model an Executable Application Manually from Scratch.....	489
Using Multiple Model Implementations in the Same Application Process.....	491
Rules for Optimizing the Configuration of Application Processes.....	495
Replacing Model Implementations.....	498
Advanced: Effects of Replacing Components Used in an Application Process.....	500

## Terms and Definitions for Building Executable Applications

### Composition of executable applications

The composition of your executable application depends on whether you use a single- or a multi-processing-unit system. The illustration below shows different components of executable applications for single-processing-unit systems and for multi-processing-unit systems.



## Terms and definitions

The following table shows important terms and their definitions.

Category	Term	Definition
Application terms	Executable application/real-time application	<p>The build result for a ConfigurationDesk application. There are different types of executable applications/real-time applications:</p> <ul style="list-style-type: none"> <li>▪ Single-core real-time applications</li> <li>▪ Multicore real-time applications</li> </ul>

Category	Term	Definition
		<ul style="list-style-type: none"> <li>▪ Multi-processing-unit applications</li> </ul> <p>An executable application/real-time application consists of one or more processing unit applications.</p>
	Single-core real-time application	A specific executable application to be executed on only one core of the real-time hardware.
	Multicore real-time application	A specific executable application to be executed on several cores of one processing unit of the real-time hardware.
	Multi-processing-unit application (multi-PU application)	A specific executable application to be executed on several processing units of the real-time hardware.
	Processing unit application	A component of an executable application/real-time application. A processing unit application contains one or more application processes.
	Application process	A component of a processing unit application. An application process contains one or more tasks.
Hardware terms	Processing unit (PU)	A hardware assembly consisting of a motherboard or a dSPACE processor board, possibly additional interface hardware for connecting I/O boards, and an enclosure, for example, a SCALEXIO Real-Time PC.
	Single-processing-unit system (single-PU system)	A system consisting of exactly one processing unit and the directly connected I/O units and I/O routers.
	Multi-processing-unit system (multi-PU system)	A system consisting of 2...n single-processing-unit systems, which are connected via a real-time capable network connection (e.g., IOCNET), and which are all accessible from a host PC.

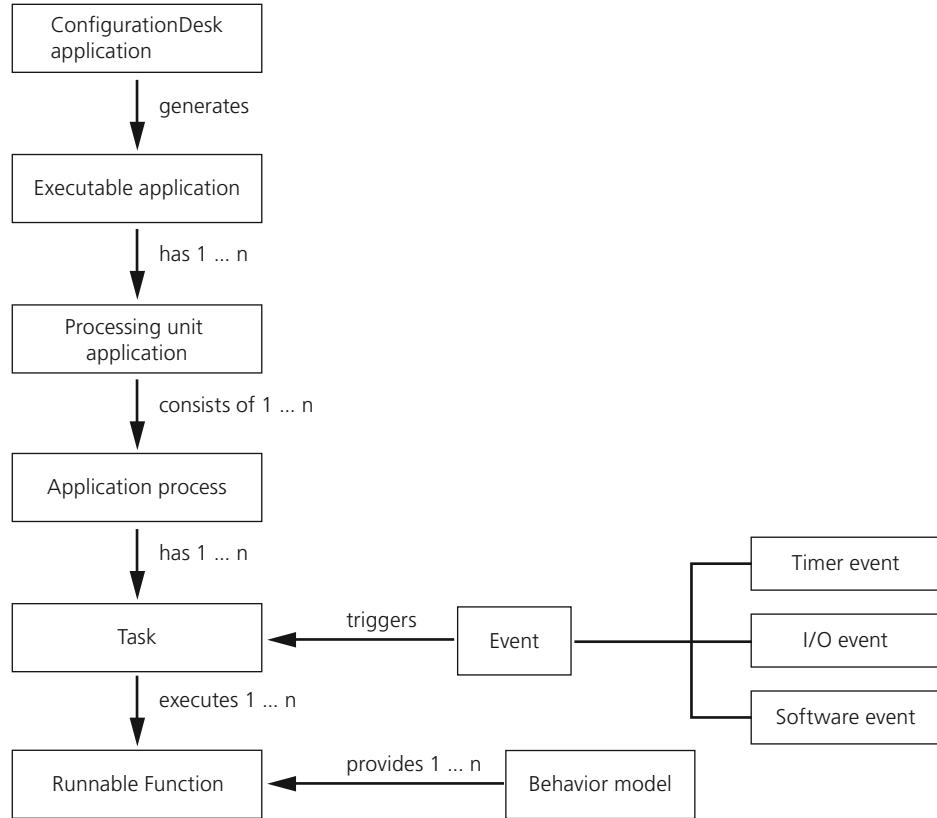
**Related topics****Basics**

Creating Multi-Processing-Unit Applications With ConfigurationDesk.....	525
Workflow for Creating a Multicore Real-Time Application Using Multiple Behavior Models.....	548

## Basics on Modeling Executable Applications

### Composition of an executable application

The following illustration shows the schematic composition of a ConfigurationDesk application.



If you create a new [ConfigurationDesk application](#), it automatically contains an executable application with the same name.

### Processing unit application

An executable application must consist of at least one processing unit application as its basic structuring element.

#### Note

The number of processing unit applications is limited to 32. If you create more processing unit applications than the maximum, the build process is aborted with an error message.

**Application processes**

A processing unit application must have at least one application process as the basic structuring element.

**Note**

Because ConfigurationDesk supports multi-processing-unit applications that can be executed on the single cores of a multi-processing-unit system, an executable application can consist of as many application processes as cores are available. However, if you want to run a real-time application on the cores of a single-processing-unit system, the maximum number of application processes is (number of cores)-1, because one core is reserved for the execution of system services.

For multi-processing-unit systems, the maximum number of application processes is (number of cores)-(number of processing units), because each processing unit needs one core for the execution of system services.

If at least one of the processing unit applications within the executable application consists of more application processes than the allowed number of cores, the download of the resulting multicore or multi-processing-unit application in ConfigurationDesk is aborted with an error message.

**Tasks in application processes**

An application process must have one or more tasks. You can assign an executable application component (i.e., a model implementation) with the predefined tasks it provides (including their runnable functions and events) to an application process. Additionally, ConfigurationDesk lets you create new periodic and asynchronous tasks to execute runnable functions. The execution of tasks is triggered by timer events, I/O events, or software events. Refer to [Basics on Tasks, Events, and Runnable Functions](#) on page 480.

**Note**

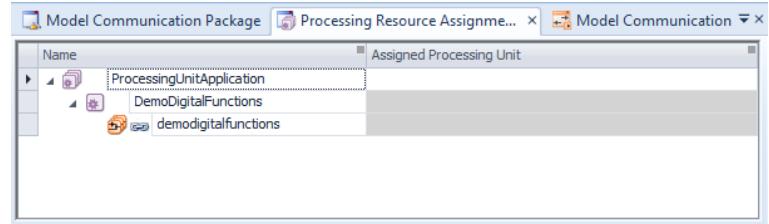
You cannot use software events to trigger tasks created in ConfigurationDesk.

The tasks in ConfigurationDesk execute one or more runnable functions.

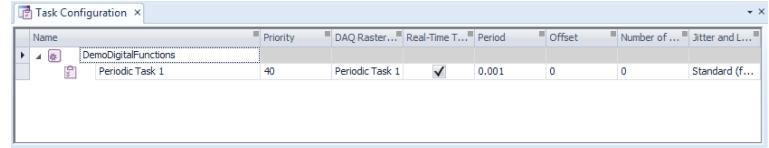
**Components of an executable application in ConfigurationDesk**

ConfigurationDesk provides the following browsers and tables for modeling executable applications:

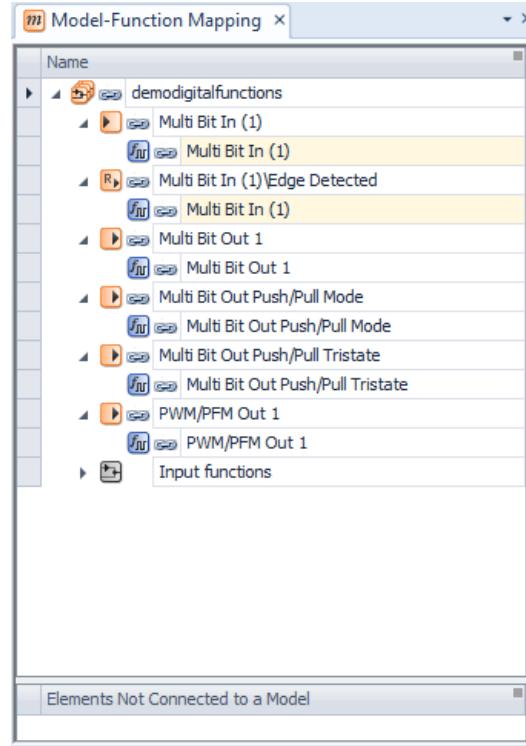
- The Processing Resource Assignment table (Multiple Models view set) for creating processing-unit applications and application processes, and for assigning processing units to application processes.



- The Task Configuration table (Tasks view set) for creating, modeling, and configuring tasks.



- The Model-Function Mapping Browser (Model-Function view set) for creating signal chains and asynchronous tasks for Simulink behavior models.



## Methods of modeling executable applications

ConfigurationDesk provides the following methods of modeling executable applications:

- Using preconfigured application processes created by ConfigurationDesk for model implementations.
- Modeling an executable application manually from scratch.

**Using preconfigured application processes created by ConfigurationDesk** If you add a model implementation to a

ConfigurationDesk application, ConfigurationDesk creates a preconfigured application process for it. This means, that ConfigurationDesk creates a new application process and assigns the executable application component, i.e., the model implementation including its predefined tasks, to it. Additionally, ConfigurationDesk creates new tasks for each runnable function provided by the model that is not assigned to a predefined task. ConfigurationDesk assigns the corresponding runnable function and timer events (for periodic tasks) to the new tasks. To complete the executable application, you only need to assign I/O events to the tasks that have to be triggered asynchronously.

**Modeling manually from scratch** ConfigurationDesk lets you model your executable application manually from scratch. Modeling an executable application comprises creating application processes as well as creating, modeling, and configuring the tasks running in it.

For more information on modeling executable applications from scratch, refer to [How to Model an Executable Application Manually from Scratch](#) on page 489.

#### Note

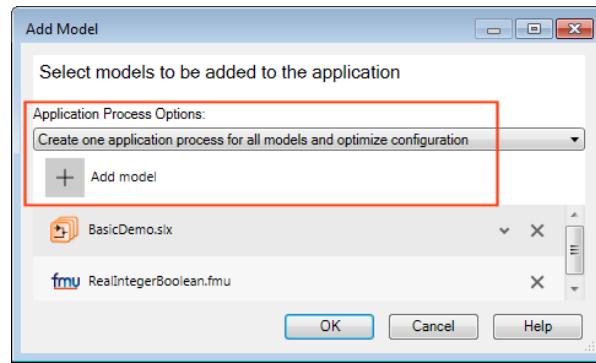
- To generate code, each function block must be assigned to an application process. Otherwise, a conflict is shown. However, a function block cannot explicitly be assigned to application processes. Instead, it is assigned implicitly to an application process via:
  - Its mapping lines to model port blocks of model implementations that are assigned to application processes.
  - The assignment of its I/O events to tasks that are assigned to application processes.
  - The hardware assignment: A function block must have a direct or indirect hardware connection to processing hardware.
- ConfigurationDesk provides methods that simplify your work if you use Simulink models as behavior models. Refer to the following topics:
  - [Working with Simulink Behavior Models](#) on page 531
  - [Modeling Workflow \(Model Interface Package for Simulink - Modeling Guide\)](#)

#### Methods of adding model implementations

ConfigurationDesk provides different methods for adding multiple model implementations to the same application process:

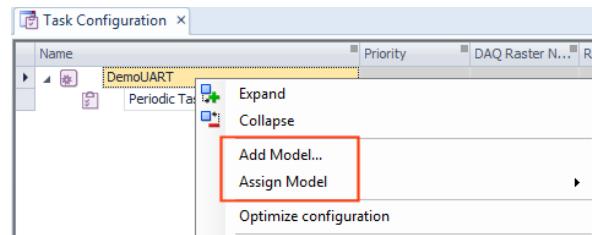
- Via the Add Model dialog:

The Application Process Options drop-down list lets you specify to create one application process for all the selected model implementations and optimize the application process configuration in one step. Refer to the following illustration.



- Via the Task Configuration table:
- The Add Model command in the context menu of an application process lets you add a model implementation to the model topology and assign it to the application process in one step.
- The Assign Model command in the context menu of an application process lets you assign additional model implementations that are already part of the model topology to the application process.

You can then optimize the application process configuration via the Optimize Configuration command from the context menu of the application process. Refer to the following illustration.



#### Resolved and unresolved executable application components

The following rules apply to executable application components of the behavior model type:

**Resolved** The behavior model is a part of the model topology.

**Unresolved** The behavior model is not a part of the model topology.

A behavior model that is assigned to an application process is marked with a symbol.

#### Resolved and unresolved elements

The following rules apply to tasks, events, and runnable functions that are part of an application process:

**Resolved** An executable application component providing the element is assigned to the application process.

**Unresolved** No executable application component providing the element is assigned to the application process. The unresolved state is indicated by a symbol.

**Handling unsupported modeling constellations**

To give you more flexibility in modeling executable applications, you can create unsupported constellations when you model an executable application. For example, you can specify the same name for multiple tasks in the same application process. These constellations are initially allowed and are displayed as [conflicts](#) in ConfigurationDesk's [Conflicts Viewer](#). However, before you start a build process for your executable application, you have to resolve the conflicts to achieve proper build results.

## Basics on Tasks, Events, and Runnable Functions

---

**Definition of terms**

**Tasks** Tasks are pieces of code whose execution is controlled by a real-time operating system (RTOS). Only one task can run at a time. However, multiple tasks can run concurrently, taking turns to use the resources of the processing unit or processor board. Each task is assigned a priority according to its relative importance. The RTOS suspends a low-priority task so that a high-priority task is given a turn (preemptive multitasking). When the high-priority task has been executed, the suspended low-priority task resumes execution. In ConfigurationDesk, a task executes one or more runnable functions.

**Runnable functions** Runnable functions are functions that are called by a task to compute results. A runnable function can be executed in a periodic or asynchronous task.

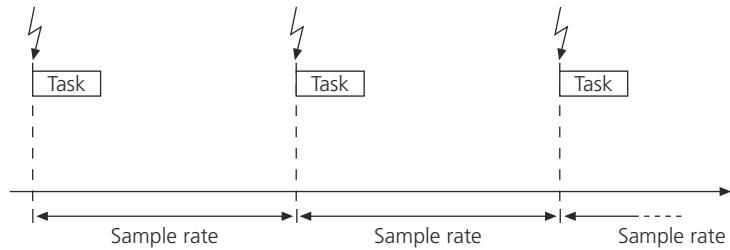
**Events** The execution of tasks is triggered by events. The following event types are available:

- Timer events are periodic events with a sample rate and an optional offset.
- I/O events are asynchronous events triggered by I/O functions.
- Software events are parts of predefined tasks provided by the behavior model. They are available in ConfigurationDesk after model analysis.

**Task types**

You can implement the following types of tasks:

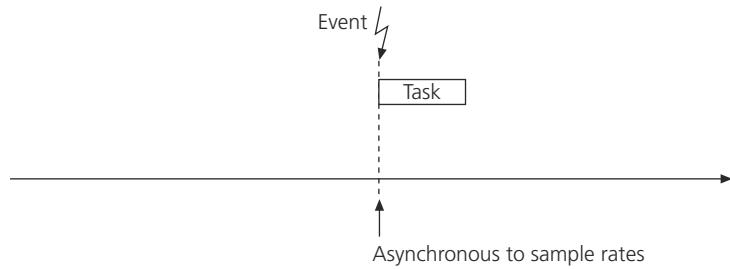
**Periodic tasks** Periodic tasks are triggered by timer events according to a specific sample rate, e.g., 20 ms.



Periodic tasks are generally generated by creating a new task and creating a timer event for the new task.

**Asynchronous tasks** Asynchronous tasks are triggered by asynchronous events, e.g., the rising edge of an input signal. The events occur asynchronously to the simulation time (real time).

Some function block types in ConfigurationDesk can generate I/O events. You can assign these events to tasks to trigger the tasks asynchronously.

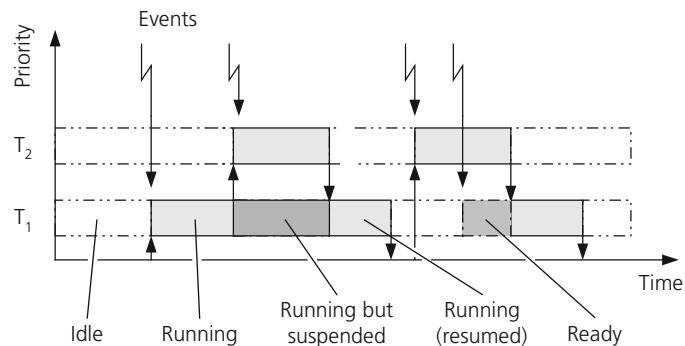


#### Note

FlexRay and LIN communication is based on the interaction between several asynchronous tasks.

### Preemptive multitasking

Suppose a real-time application consists of two tasks,  $T_1$  and  $T_2$ .  $T_1$  is a low-priority task, and  $T_2$  a high-priority task. When  $T_1$  is running, it is preempted by  $T_2$ , and resumes execution after  $T_2$  finishes. When  $T_1$  is triggered again, it must wait until  $T_2$  finishes before starting.



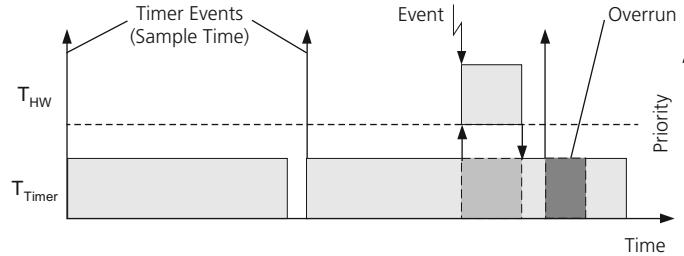
**Task states** The following task states are possible:

State	Meaning
Idle	The task is inactive, waiting to be triggered.
Ready	The task has been triggered but cannot start due to a high-priority task that is currently running. It is waiting to start execution.

State	Meaning
Running	The task has started running. This state is true until the task finishes running, regardless of whether it is preempted by a high-priority task or not.

**Task overrun**

An overrun situation occurs if a task is requested to start but has not finished its previous execution. Consider the situation shown below:



There is no overrun between the first and the second time the periodic task  $T_{\text{Timer}}$  is triggered, but if high-priority task  $T_{\text{HW}}$  needs to be calculated, there is not enough time for low-priority task  $T_{\text{Timer}}$  to finish before it is requested to start again. If such an overrun situation occurs for a task, the real-time application reacts according to the task's overrun settings, refer to [Basics on Task Overruns](#) on page 483.

**Data acquisition rasters**

To measure variables stored in the TRC file synchronously with a task during real-time simulation, for example, in dSPACE's ControlDesk, a DAQ raster must be assigned to the task. You can assign DAQ rasters to a maximum of 31 tasks per application process. For details, refer to [Basics on Measurement Rasters](#) ([ControlDesk Measurement and Recording](#)).

**Real-Time Testing**

Using Real-Time Testing (RTT), you can execute tests synchronously with a task during real-time simulation. You can enable RTT for exactly one task. For details on performing RTT, refer to [Real-Time Testing Guide](#).

**Note**

The Real-time testing property must be enabled for exactly one task in each application process. If you do not use Real-Time Testing, the Real-Time Testing calls in the generated code do not influence the performance of the real-time application.

**Monitoring tasks**

The TRC file provides the following *task information variables* for each task of a real-time application. These can be monitored, for example, in dSPACE ControlDesk. This is useful for analyzing an overrun situation, etc.:

- Task Call Counter:  
Indicates how often the task has been triggered since application start. It is reset to 0 when the application starts.
- Task Turnaround Time:  
Returns the time that passes between the task trigger and the end of task execution. It can include the time required by higher-priority tasks that interrupt the task.
- Overrun Count:  
Indicates how often an overrun occurred since application start. It is reset to 0 when the application starts.

## Basics on Task Overruns

**Introduction**

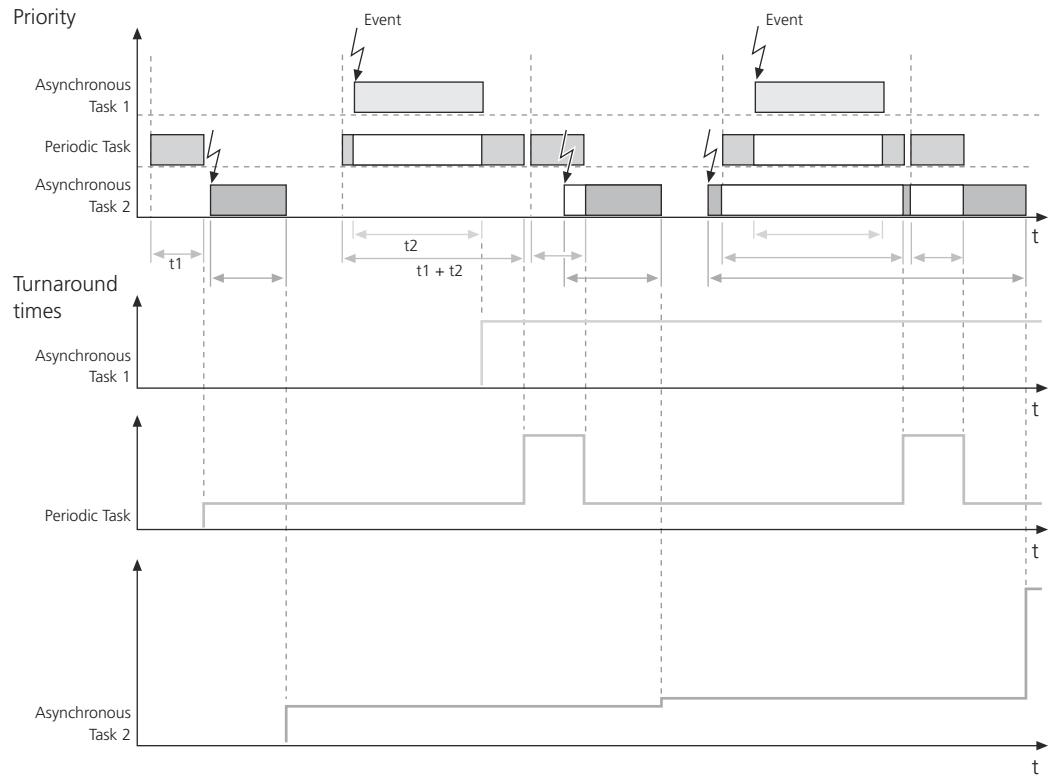
A simulation cannot be performed in real time while task overruns occur. You should be familiar with strategies to avoid task overruns. To keep an application running, though not in real time, task overruns can simply be ignored.

**Timing requirements**

Ensure that each [task](#) in your model has a sample time greater than the sum of the task-switching time and turnaround time.

**Task-switching time** The task-switching time depends on the hardware.

**Turnaround time** The turnaround time for a task is the time that passes between the triggering and the end of execution. It can include the time required by higher-priority tasks that interrupt it. The illustration below shows the turnaround times for a number of tasks.

**Note**

Conditional execution paths in the model code can cause the turnaround time of a task to vary from one execution step to another.

**Tip**

In the real-time application, you can measure the turnaround time of each task via ControlDesk.

### ConfigurationDesk-specific strategies to avoid overruns

ConfigurationDesk provides the following features to avoid task overruns and prevent the running application from being stopped:

**Smoothing the start-up behavior of periodic tasks** During the very first task execution cycles after a real-time application has been started, a task overrun might occur due to inefficient memory access (cold cache). Such task overruns can be avoided by prolonging the sample time of periodic tasks during the start phase of real-time applications. You can specify a prolonging factor and a time period for applying it. For detailed instructions, refer to [How to Configure the Start-Up Behavior of a Periodic Task](#) on page 516.

The task start-up behavior of periodic tasks can also be configured in experiment tools such as dSPACE's ControlDesk.

#### Note

While the sample time is prolonged, the simulation is not performed in real time.

**Splitting I/O access into multiple function modules** I/O access is bound to the execution of so-called function modules (for details on function modules, refer to [Basics on Function Triggers](#) on page 328). If a task performing I/O consists of only one function module, all the I/O access is done at the beginning of the task or function module. This accumulation of I/O accesses might result in a noticeable waiting time.

If you work with Simulink models, the following applies:

If I/O functions are affected by a delay, you can solve the problem by moving their data port blocks to an additional atomic subsystem. This results in a task that consists of multiple function modules which are executed consecutively.

#### Tip

Identifying the data port blocks that cause the delay commonly requires an iterative approach.

You can determine the execution order of function modules that belong to the same task by specifying different block priorities for the atomic subsystems which the function modules are related to. Refer to the **Priority** parameter in the Block Properties dialog of the subsystem block in the behavior model.

### Basic strategies to avoid overruns

These are basic strategies to avoid task overruns. They are not bound to ConfigurationDesk.

**Adapting task priorities to sample times** Inappropriate task prioritization can cause task overruns. For example, if a task has a fast sample rate, minor computation effort, and a low priority, and another task has a slower sample rate and complex calculations to perform but a higher priority, overruns are likely to happen to the fast task. In this case, you can solve the overrun problem by prioritizing the fast task higher than the slower task.

#### Note

Tasks that are related to faster sample times must be assigned higher priorities, i.e., lower priority values.

The table below shows an example with typical task priority values, including FlexRay and LIN communication tasks:

Task	Priority Value
Periodic 1 (20 ms)	40
Periodic 2 (100 ms)	41

Task	Priority Value
Asynchronous 1 (timer event: 5 ms)	12
Asynchronous 2 (function event)	11
FlexRay (synchronization)	0
FlexRay (application/communication)	1
LIN	9

**Swapping computations into slower tasks** A calculation overload in fast tasks can cause task overruns. For example, if non-time-critical but complex computations are performed at the fast base sample rate of the behavior model, overruns are likely to happen. Ensure that these model parts are executed at a slower sample rate.

#### Handling task overruns

The application can handle a task overrun by ignoring it, which prevents the running application from being stopped. In this case the trigger event that caused the task overrun is lost. The possible range of the number of accepted overruns is [0 ... infinite]. For details, refer to [How to Configure Tasks in ConfigurationDesk](#) on page 515.

##### Note

While task overruns are ignored, the simulation is not performed in real time.

## Basics on Modeling Tasks in ConfigurationDesk

#### Introduction

ConfigurationDesk allows you to model the tasks in your executable application very flexibly. You can use predefined tasks provided by the [behavior model](#), or you can create new tasks in ConfigurationDesk.

#### Modeling steps

Modeling tasks comprises the following steps:

1. Creating a task
2. Assigning an event to the task
3. Assigning one or more runnable functions to the task
4. Configuring task properties

For detailed instructions, refer to [How to Model an Executable Application Manually from Scratch](#) on page 489.

#### Handling predefined and user-defined tasks

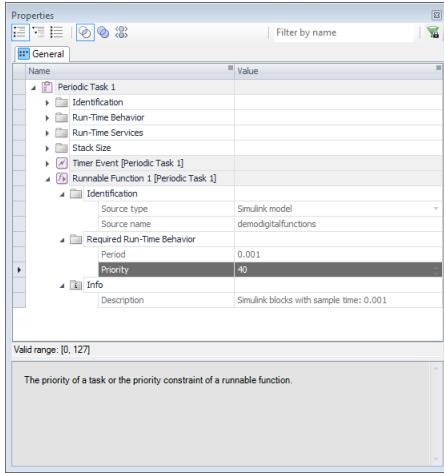
You can assign an executable application component with predefined tasks (including their runnable functions and events) to an application process. If

runnable functions are not assigned to a predefined task, you have to model a new task for them in ConfigurationDesk. To do so, you must first create the task and then create a timer event for it or assign an existing event to it. Finally, you have to assign runnable functions to the task. You can perform task configuration (i.e., specify task properties like priority, DAQ raster name, etc.) for predefined tasks as well as for user-defined tasks created in ConfigurationDesk. For details on task configuration, refer to [Configuring Tasks in ConfigurationDesk](#) on page 513.

### Task priority restriction

When configuring predefined tasks, you must comply with the following rule:

A component might provide runnable functions with a task priority restriction. For example, all runnable functions provided by a Simulink model have a task priority restriction which indicates the relative task priority within the model.



When configuring the tasks the runnable functions are assigned to, you must make sure that the relationship between the task priorities matches the relationship between the predefined priorities, even if the absolute priority values differ from the predefined values. The following table provides an example of runnable functions with their task priority restrictions and allowed task priorities:

Runnable Function Name	Task Priority Restriction of Runnable Function	Allowed Task Priority (Example)
RF1 (assigned to task T1)	10	22
RF2 (assigned to task T2)	40	77
RF3 (assigned to task T3)	41	88

According to the task priority restrictions of the runnable functions, the task priority value of T1 must be lower than the task priority value of T2, and the task priority value of T2 must be lower than the task priority value of T3.

**Note**

- If you configure task priorities that do not comply with the task priority restrictions, a conflict is shown in the **Conflicts Viewer**. You have to resolve this conflict before you start the build process.
- For user-defined tasks and tasks provided by Simulink behavior models and Functional Mock-up Units (FMUs), lower values indicate higher task priorities. For tasks provided by V-ECU implementations, it is the other way round: Higher values indicate higher task priorities.

**Cycle time restriction**

A component might provide runnable functions with a cycle time restriction. A cycle time restriction indicates a 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**.

**Related topics****References**

- [Event Properties \(ConfigurationDesk User Interface Reference\)](#)
- [Runnable Function Properties \(ConfigurationDesk User Interface Reference\)](#)
- [Task Properties \(ConfigurationDesk User Interface Reference\)](#)

## Different Simulink Tasking Modes and Their Effects in ConfigurationDesk

**Introduction**

Simulink lets you select either the single-tasking mode or the multitasking mode for the behavior model. You can switch between the modes via the **Treat each discrete rate as a separate task** checkbox on the **Solver** page of the **Configuration Parameters** dialog. You should know which elements the behavior model provides for modeling executable applications depending on the selected tasking mode.

**Components resulting from different Simulink tasking modes**

Regardless of the tasking mode, the behavior model provides the following elements for modeling an executable application:

- A runnable function for the Simulink base rate task (no predefined task).
- A runnable function without predefined task for each **Hardware-Triggered Runnable Function** block in the behavior model.
- A predefined task including a runnable function and with an assigned software event for each **Software-Triggered Runnable Function** block in the behavior model.

**Single-tasking mode** In single-tasking mode, no further components are provided. The calculation of all periodic parts of the model is performed in the runnable function for the Simulink base rate.

**Multitasking mode** In multitasking mode, the different sample rates of the behavior model which do not equal the base rate are available as predefined tasks in ConfigurationDesk, because Simulink triggers periodical software events to execute these tasks.

To make predefined elements available in ConfigurationDesk, you have to execute the **Analyze Simulink Model (Including Task Information)** command for the behavior model.

---

## Related topics

## References

[Analyze Simulink Model \(Including Task Information\) \(ConfigurationDesk User Interface Reference\)](#)

# How to Model an Executable Application Manually from Scratch

## Objective

ConfigurationDesk lets you model your executable application manually from scratch.

### Note

- The following steps relate to the use of Simulink behavior models in ConfigurationDesk.
- When you model your executable application, you can create unsupported constellations, which are displayed as [conflicts](#) in the [Conflicts Viewer](#). The conflicts that can occur are described below. Before you build your executable application, ensure that all the conflicts are resolved.

---

## Preconditions

- At least one [behavior model](#) providing [runnable functions](#) was added to your [ConfigurationDesk application](#). Refer to [How to Import a Model Topology](#) on page 112).
- A complete [model analysis](#) was performed for the behavior model via the **Analyze Simulink Model (Including Task Information)** command to make predefined elements available in your ConfigurationDesk application. Refer to [Working with Simulink Behavior Models](#) on page 531.

Method	To model an executable application manually from scratch
	<p><b>1</b> In the Task Configuration table in the Tasks view set, right-click an empty area.</p> <p><b>2</b> From the context menu, select New - Application Process. The new application process is displayed in the Task Configuration table.</p> <p><b>3</b> Assign the behavior model to the application process: Right-click the application process and select Assign Model from the context menu.</p> <p><b>4</b> Right-click the application process and select New - Task from the context menu. The new task is displayed in the Task Configuration table.</p>
	<p><b>Note</b></p> <p>If the application process has no task, this is displayed as a conflict. You have to add one or more tasks with unique names to the application process to resolve this conflict. If the names are not unique, this is also displayed as a conflict.</p>
	<p><b>5</b> Assign the runnable functions to the task by using the Assign Runnable Function command from the context menu of the task. Refer to <a href="#">Assign Runnable Function (ConfigurationDesk User Interface Reference)</a>.</p>
	<p><b>Note</b></p> <p>If a runnable function is assigned to multiple tasks, this is displayed as a conflict.</p>
	<p><b>6</b> Sort the runnable functions of the task in the order in which you want to execute them.</p> <p><b>7</b> To model a periodic task, right-click the task and select New - Timer Event from the context menu to create a new timer event.</p>
	<p><b>Note</b></p> <p>You can rename the timer events in ConfigurationDesk. The names must be unique. If they are not, this is displayed as a conflict.</p>
	<p>To model an asynchronous task, enable event generation for a function block and assign the I/O event to the task. Refer to <a href="#">How to Model Asynchronous Tasks for Models with Runnable Function Blocks (Manually)</a> on page 511.</p>
	<p><b>Tip</b></p> <p>You can assign an available event to a task by using the Assign Event command from the context menu of the task. Refer to <a href="#">Assign Event (ConfigurationDesk User Interface Reference)</a>.</p>

- 8** Configure the task properties according to your requirements. Refer to [How to Configure Tasks in ConfigurationDesk](#) on page 515.

For multimodel applications, repeat steps 2 - 8 for all the behavior models available in your ConfigurationDesk application.

The following steps are necessary only if you want to build a multi-processing-unit application.

- 9** In the Multiple Models view set, right-click an empty area of the Processing Resource Assignment table.

- 10** From the context menu, select New - Processing Unit Application.

The new processing unit application is displayed in the Processing Resource Assignment table.

**Tip**

To create multiple-processing-unit applications, you can select New - Multiple Processing Unit Application from the context menu and enter the desired number of processing unit applications in the Processing Unit Application dialog.

- 11** In the Processing Resource Assignment table, assign the application processes to the desired processing unit application via drag & drop.

**Result**

You modeled an executable application from scratch. You can now configure and start the build process.

**Related topics**

**Basics**

[Different Simulink Tasking Modes and Their Effects in ConfigurationDesk](#)..... 488

**References**

[Add Model \(ConfigurationDesk User Interface Reference\)](#)  
[Analyze Simulink Model \(Including Task Information\) \(ConfigurationDesk User Interface Reference\)](#)  
[Assign Event \(ConfigurationDesk User Interface Reference\)](#)  
[Assign Model \(ConfigurationDesk User Interface Reference\)](#)  
[Assign Runnable Function \(ConfigurationDesk User Interface Reference\)](#)  
[New - Multiple Processing Unit Applications \(ConfigurationDesk User Interface Reference\)](#)

## Using Multiple Model Implementations in the Same Application Process

**Introduction**

In multicore real-time applications or multi-processing-unit applications with only one model implementation assigned to each application process, each model

implementation is executed on a separate core. ConfigurationDesk lets you assign more than one model implementation to the same application process. As a result all the model implementations that are assigned to the same application process can be executed on the same core of the processing-unit.

#### Supported model types and versions

Assigning multiple models to the same application process is supported for the following model implementations:

- Simulink behavior models
- Simulink implementation containers (SIC files) created with the Model Interface Package for Simulink 4.1 (dSPACE Release 2019-A) or later.
- Precompiled SIC files created with dSPACE Release 2020-B.
- SIC files created with TargetLink 5.1 (dSPACE Release 2020-B).
- Bus simulation containers (BSC files) that contain CAN or LIN bus communication and that are created with the Bus Manager of dSPACE Release 2020-B.
- Precompiled BSC files created with dSPACE Release 2020-B.
- Functional Mock-up Units (FMUs).
- Precompiled FMUs created with dSPACE Release 2020-B or later.

#### Note

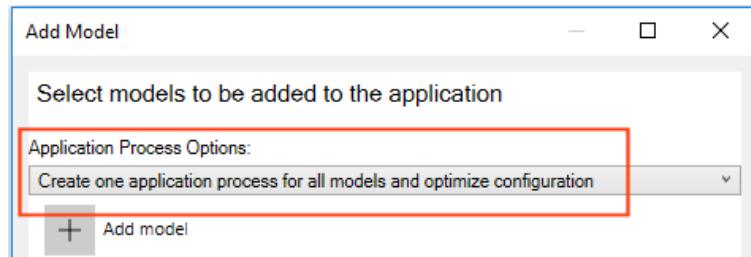
If a multimodel application process contains unsupported model implementation types, a conflict is shown in the Conflicts Viewer. You must resolve this conflict before you can start the build process.

#### Methods of adding multiple model implementations

ConfigurationDesk provides different methods for adding multiple model implementations to the same application process:

- Via the Add Model dialog:

The Application Process Options drop-down list lets you specify to create one application process for all the selected model implementations and optimize the application process configuration in one step. Refer to the following illustration:

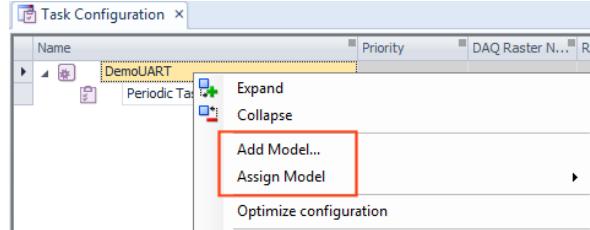


- Via the Task Configuration table:

- The Add Model command in the context menu of an application process lets you add a model implementation to the model topology and assign it to the application process in one step.

- The Assign Model command in the context menu of an application process lets you assign additional model implementations that are already part of the model topology to the application process.

Refer to the following illustration:



You can then optimize the application process configuration via the Optimize Configuration command from the context menu of the application process.

#### Optimizing application process configuration

Suppose you add multiple model implementations to a ConfigurationDesk application and create one application process for them. In this case, ConfigurationDesk automatically optimizes the application process configuration. This includes the following steps:

1. Grouping runnable functions in tasks.
2. Specifying an execution order for the runnable functions within the tasks.
3. Specifying the task priorities.

For more information, refer to [Rules for Optimizing the Configuration of Application Processes](#) on page 495.

#### Optimizing the configuration automatically

Due to changes in the models within an application process or due to changes in the model communication, it is possible that the configuration of the application process is no longer optimal. To optimize the configuration, ConfigurationDesk provides the Optimize configuration automatically property for application processes. If this checkbox is selected, the configuration optimization is performed automatically at the following points in time:

- During each build process before the conflicts are calculated.
- After you perform the following actions:
  - You reload a model container.
  - You analyze a model implementation.
  - You delete a model.
  - You delete a model assignment.
  - You assign a model to an application process.
  - You replace a model.

**Note**

- You can use the **Optimize configuration automatically** checkbox on the Configuration page of the Options dialog to specify the default behavior of the application process property. If you select this checkbox, the **Optimize configuration automatically** checkbox is selected for each newly created application process by default.
- You can also manually perform the configuration optimization using the **Optimize Configuration** context menu command of the relevant application process.

**Execution order of runnable functions**

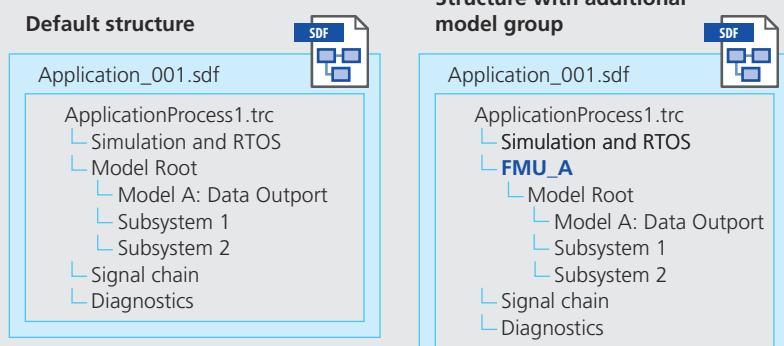
ConfigurationDesk specifies an initial execution order of the runnable functions within a task based on the data flow. If you set up or changed model communication, and/or you added additional runnable functions to a task, you can have ConfigurationDesk specify a suitable execution order for the runnable functions within a task via the **Optimize Configuration** command from the context menu of the task. You can also manually specify an execution order, if required.

**TRC file structure of a multimodel application process**

A TRC file created for a multimodel application process has additional groups. The TRC file contains a group for each model implementation associated with the application process. Each model group is named after the corresponding model implementation. If you work with a BSC file that contains a Simulink implementation container (SIC file), the model group is named after the model implementation described by the SIC file.

**Tip**

If only one model implementation is assigned to an application process, the related TRC file has the default structure without additional model groups. If you want ConfigurationDesk to add an additional model group in the TRC file for application processes with one model implementation, you have to select the **Insert model name TRC file group** property in the global build settings of the Build Configuration table. The following illustration shows the default structure and the structure with an additional model group:



### Using BSC files in multimodel applications

When you use BSC files in multimodel applications, you must observe the following naming restrictions:

- If a BSC file contains a Simulink implementation container (SIC file), the name of the model implementation that the SIC file describes must be unique in the application process to which the BSC file is assigned. This applies to the following:
  - All model implementations assigned to the application process.
  - All model implementations described by SIC files that are included in other BSC files assigned to the application process.
- If you assign two or more BSC files to an application process, the names of the bus configurations that are included in the BSC files must be unique across all BSC files.

#### Note

If the BSC files in a multimodel application process do not comply with the naming restrictions, conflicts are shown in the [Conflicts Viewer](#). You must resolve these conflicts before you can start the build process.

For more information on limitations concerning multimodel application processes, refer to [Limitations Concerning Multiple Model Implementations Assigned to the Same Application Process](#) on page 768.

**Build results for BSC files in multimodel application processes** During the build process, ConfigurationDesk generates an EXPSWCFG file for application processes to which a BSC file is assigned. For application processes to which multiple BSC files are assigned, the generated EXPSWCFG file contains the bus configuration data of all the BSC files.

### Warning in case of variables with the same symbol name

If several model implementations are assigned to the same application process, and their binaries contain variables with the same symbol name in the common section, this can lead to unexpected behavior during the real-time simulation. Therefore, ConfigurationDesk displays a warning during the build process.

## Rules for Optimizing the Configuration of Application Processes

### Rules for optimizing application process configuration

ConfigurationDesk automatically optimizes the configuration of newly created application processes if they are added via the Add Model dialog and the [Create one application process for all models and optimize configuration](#) option is selected. The configuration is also optimized if the [Optimize configuration automatically](#) checkbox is selected for the relevant application process. Additionally, you can trigger the optimization via the [Optimize](#)

Configuration context command of the application process. Optimizing the application process configuration comprises the following steps:

1. Grouping runnable functions in tasks.
2. Specifying the execution order of the runnable functions within a task.
3. Specifying the task priorities.

#### Note

- If there are changes in the behavior model, or you set up or changed model communication or added additional model implementations to the application process, you must optimize the configuration by selecting the **Optimize Configuration** command from the context menu of the application process.
- If you added additional runnable functions to a task only, and you do not want ConfigurationDesk to make changes to the whole application process, you can let ConfigurationDesk order the runnable functions within the task based on the data flow by selecting the **Optimize Configuration** command from the context menu of the task.

#### Rules for grouping runnable functions in tasks

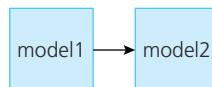
Runnable functions that have the same cycle time restriction (indicated by the **Period** property in the Properties Browser) are combined in the same task. If there is already a task that is triggered with the required sample time, the runnable functions are added to that task. Otherwise, ConfigurationDesk creates a new task with a suitable new timer event, and assigns the runnable functions to that task.

#### Rules for specifying the execution order of runnable functions

For tasks to which more than one runnable function are assigned, ConfigurationDesk specifies the execution order of the runnable functions. The execution order depends on the connection of the models providing the runnable functions. In the following examples, **model1** provides the **rf1** runnable function, and **model2** provides the **rf2** runnable function.

**model1 and model2 are not connected** ConfigurationDesk specifies the execution order of **rf1** and **rf2** according to the alphabetical order of the related model names.

**model1 sends data to model2** Refer to the following illustration:

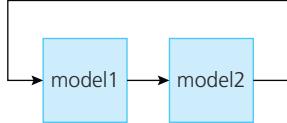


In the preceding illustration, **rf1** must be executed before **rf2**. Thus, ConfigurationDesk specifies the following execution order:

rf1: 1

rf2: 2

**model1 and model2 are connected in a closed loop** Refer to the following illustration.



In the preceding example, the execution order depends on the setting of the Protocol property of the related model communication package.

ConfigurationDesk interrupts the closed loop where the model communication package is specified as non-blocking. If the loop is interrupted before model1, rf1 is executed before rf2, for example.

#### Rules for specifying the task priorities

The specification of the task priorities depends on the sample times with that the related tasks are triggered.

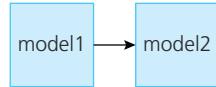
**Tasks are triggered with different sample times** Tasks with a smaller sample time get a higher priority.

**Tasks are triggered with the same sample times** The priority of the tasks depends on the specified communication of the models providing the tasks. In the following examples, model1 provides the task1 task, and model2 provides the task2 task

- model1 and model2 are not connected:

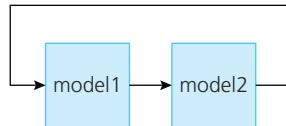
ConfigurationDesk sorts model1 and model2 in an alphabetical order according to their model names in the Model-Function Mapping Browser. The task priorities are specified according to the specified model order.

- model1 sends data to model2:



task1 gets a higher priority than task2.

- model1 and model2 are connected in a closed loop:



ConfigurationDesk breaks up the loop at a connection where the model communication is configured as non-blocking. If the loop is disconnected before model1, task1 gets a higher priority than task2.

**Two tasks contain at least one runnable functions that is provided by the same model** The task with the runnable function that has a smaller priority restriction gets a smaller priority value.

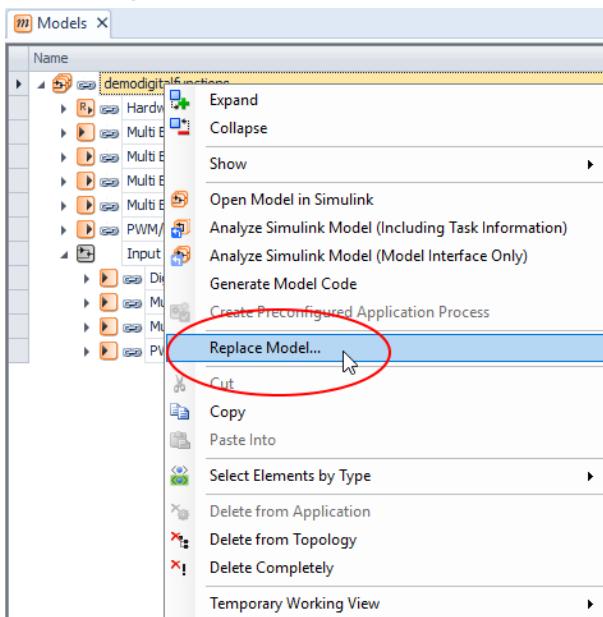
## Replacing Model Implementations

### Introduction

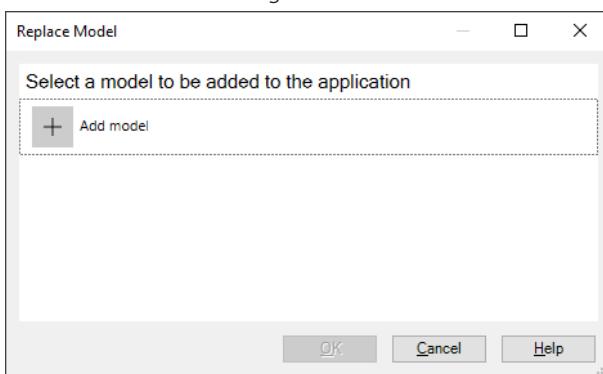
ConfigurationDesk provides methods that let you replace a model implementation with another model implementation.

### Basics of replacing a model implementation

ConfigurationDesk lets you replace a resolved model implementation in your model topology with another model implementation. When you replace a model implementation, it is removed from the model topology and, if applicable, from the application process to which it is assigned. Afterwards, the removed model implementation is replaced by the selected model implementation in both the model topology and the application process. To replace a model implementation, you must select the Replace Model command from the context menu. Refer to the following illustration:



When you select the Replace Model command, the Replace Model dialog opens where you can select only one model implementation to replace the old one. Refer to the following illustration:



After you have selected a model implementation and confirmed the dialog by clicking OK, the old model implementation is replaced with the selected one.

### Results of replacing a model implementation

Replacing a model implementation provides the following results:

- The old model implementation is deleted from the model topology and the new one is added.
- If the old model implementation was assigned to an application process, this assignment is deleted and the new model implementation is assigned to this process.
- For model port blocks/model ports of FMUs and V-ECU implementations that exist in both model implementations in the identical hierarchy level and with the same name, the model port blocks/model ports remain resolved. This also applies if the name of the old model implementation is not identical to the name of the new model implementation.

#### Note

In FMUs and V-ECU implementations, the model name is part of the ID of the model port blocks/model ports. If FMUs or V-ECU implementations are replaced by FMUs or V-ECU implementations with a different name, the model name in the ID of model port blocks/model ports is also replaced. Therefore, the ID also changes for model port blocks/model ports that have the same name in both models.

### Replacing a behavior model via automation API

You can replace a model implementation from the model topology by calling the "ReplaceModel" operation with the *Configure(ICaComponent)* method. The following table shows the parameters and their data types for replacing model implementations:

Parameter	Data Type
Path to the new model	String
Flag to analyze model	Boolean
Initialization command	String
Name of the model to be replaced	String

**Example** The following example shows you how to replace a model implementation:

```
result = ModelTopology.Configure('ReplaceModel',
[r'c:\System_B.slx,False,'','System_A.fmu']) replaces the
System_A.fmu FMU with the System_B.slx Simulink model.
```

For more information, refer to [ICaComponent <>Collection>>](#) ([ConfigurationDesk Automating Tool Handling](#)).

**Related topics****References**

[ICaComponent <>Collection>> \(ConfigurationDesk Automating Tool Handling !\[\]\(4f900328831e6083aaf85add08f24ad8\_img.jpg\)  
\[Replace Model \\(ConfigurationDesk User Interface Reference !\\[\\]\\(207905081adab5ae24f627bde755ab7f\\_img.jpg\\)\]\(#\)](#)

## Advanced: Effects of Replacing Components Used in an Application Process

### Introduction

The scenarios described below show the effects of replacing the components of an [application process](#). If you replace the components used in an application process, the state of elements used in an application process might switch from resolved to unresolved.

### Replacing a behavior model by a model with the same name

Consider the following scenario: You have imported a behavior model to your ConfigurationDesk application and assigned it to an application process. The model provides at least one runnable function. You have created an asynchronous task and assigned an imported runnable function to it. Now you replace the model as follows:

1. You delete the behavior model from the model topology.

Results:

- The behavior model is displayed as unresolved in the application process.
- The runnable function you assigned to the asynchronous task is displayed as unresolved in the Task folder.
- The I/O event assigned to the task is displayed and remains resolved.

2. You add a new behavior model to the model topology. The behavior model and the runnable function provided by it have the same names as the ones you deleted from the model topology in the previous step. Further elements can be named differently.

Results:

- In the application process, the behavior model is displayed as resolved.
- The runnable function you assigned to the asynchronous task is resolved in the task.

### Replacing a behavior model by a model with a different name

Consider the following scenario: You have added a behavior model to the model topology and assigned it to an application process. The model provides at least one runnable function. You have created a task and assigned the runnable function to it. Now you replace the model as follows:

1. You unassign the behavior model from the application process.
2. You assign a different behavior model with a different name to the application process. The runnable function provided by the new behavior model has the same name as the runnable function in the behavior model that was unassigned from the application process.

Results:

- All the predefined tasks of the new behavior model are available in the application process
- In the application process, the runnable function is resolved in the user-defined task.

# Modeling Asynchronous Tasks

<b>Introduction</b>	Asynchronous tasks are tasks that are triggered by I/O events. I/O events are generated by function blocks. You can easily model asynchronous tasks in ConfigurationDesk.
---------------------	---

Where to go from here	Information in this section
	<a href="#">Basics on Modeling Asynchronous Tasks.....</a> 502 <a href="#">How to Enable Event Generation for Function Blocks.....</a> 504 <a href="#">How to Model Asynchronous Tasks for Models with Hardware-Triggered Runnable Function Blocks.....</a> 505 <a href="#">How to Model Asynchronous Tasks and Create Suitable Runnable Function Blocks in One Step.....</a> 508 <a href="#">How to Model Asynchronous Tasks for Models with Runnable Function Blocks (Manually).....</a> 511

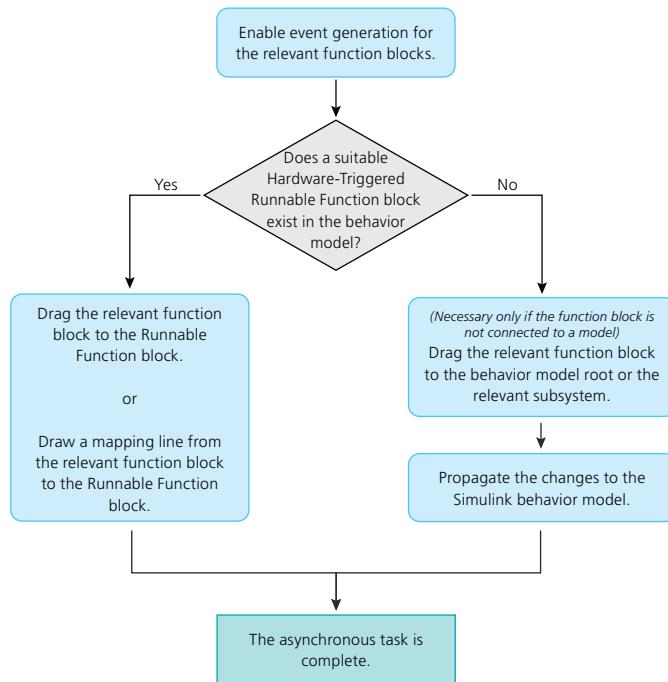
## Basics on Modeling Asynchronous Tasks

<b>Introduction</b>	An asynchronous task is a piece of code that executes one or more runnable functions and that is triggered by one or more I/O events. If you use Simulink behavior models in a ConfigurationDesk application, the runnable functions are provided by Hardware-Triggered Runnable Function blocks from the Model Interface Blockset. I/O events are provided by function blocks for which event generation is enabled.
---------------------	---

<b>Required knowledge</b>	For modeling asynchronous tasks, you must be familiar with the Model-Function Mapping Browser. Refer to <a href="#">User-Friendly Connection of ConfigurationDesk and Simulink Models</a> on page 553.
---------------------------	--

<b>Work steps for modeling asynchronous tasks</b>	The steps you have to perform for modeling asynchronous tasks depend on your approach: <ul style="list-style-type: none"> <li>▪ Approach 1: You create a behavior model, including suitable Hardware-Triggered Runnable Function blocks from the Model Interface Blockset, and add it to a ConfigurationDesk application.</li> <li>▪ Approach 2: You added a behavior model without suitable Hardware-Triggered Runnable Function blocks to a ConfigurationDesk application.</li> </ul>
---	---

The following illustration shows possible workflows for modeling asynchronous tasks, depending on your approach.



#### Note

- It is possible to assign multiple I/O events to an asynchronous task. This is not taken into account in the illustration above but in the workflow steps described below.
- For event types other than I/O events, multiple assignments to a task are not supported and cause a conflict in the Conflicts Viewer.
- If multiple I/O events trigger a task at the same time, a task overrun might occur. Refer to [Basics on Task Overruns](#) on page 483.

#### Workflow details

For instructions on the workflow steps, refer to the following topics, depending on your starting point:

- If the Simulink behavior model provides Hardware-Triggered Runnable Function blocks from the Model Interface Blockset, refer to [How to Model Asynchronous Tasks for Models with Hardware-Triggered Runnable Function Blocks](#) on page 505.

#### Tip

For Simulink behavior models with Hardware-Triggered Runnable Function blocks, it is also possible to model the asynchronous tasks manually. Refer to [How to Model Asynchronous Tasks for Models with Runnable Function Blocks \(Manually\)](#) on page 511.

- If the Simulink behavior model does not provide any Hardware-Triggered Runnable Function blocks from the Model Interface Blockset, refer to [How to Model Asynchronous Tasks and Create Suitable Runnable Function Blocks in One Step](#) on page 508.

**Related topics****Basics**

User-Friendly Connection of ConfigurationDesk and Simulink Models..... 553

## How to Enable Event Generation for Function Blocks

**Objective**

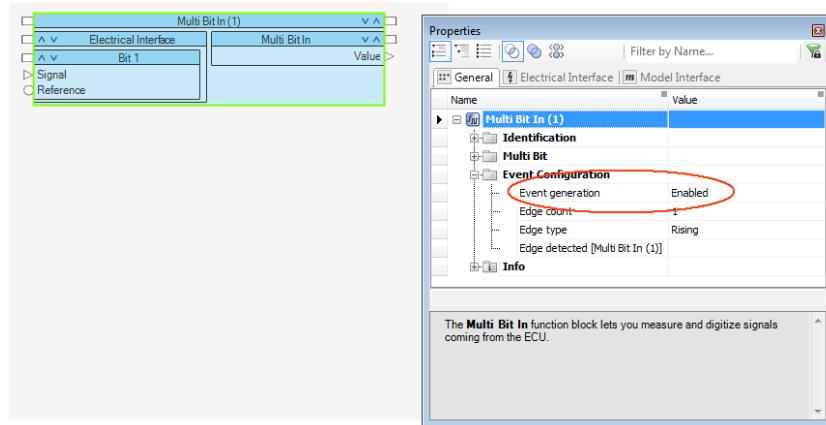
To be able to use [I/O events](#) generated by [function blocks](#), you must enable event generation for the relevant blocks.

**Precondition**

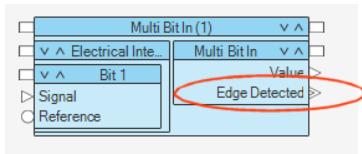
The [signal chain](#) must contain a function block that provides event generation.

**Method****To enable event generation for function blocks**

- Select the function block that provides event generation in the signal chain or in the Model-Function Mapping Browser.
- From the Event generation list in the function block's Properties Browser, select Enabled.

**Result**

You enabled event generation for the function block. The event is displayed in the function block's Properties Browser. The name of the event is derived from the event condition. A new event port is added to the function block in the signal chain.



## How to Model Asynchronous Tasks for Models with Hardware-Triggered Runnable Function Blocks

### Objective

ConfigurationDesk offers a simple way to use Hardware-Triggered Runnable Function blocks provided by Simulink behavior models for modeling asynchronous tasks.

### Preconditions

To perform the steps below, the following preconditions must be fulfilled:

- You must have added a Simulink behavior model containing at least one Hardware-Triggered Runnable Function block to the ConfigurationDesk application. In the Simulink behavior model, the Hardware-Triggered Runnable Function block must be connected to a function-call subsystem.
- You must have performed a model analysis. The Hardware-Triggered Runnable Function block is then represented by the following elements in ConfigurationDesk:
  - A Runnable Function block in the signal chain.
  - A runnable function in the Task Configuration table.
  - The model must be assigned to an application process.
  - You must have added the relevant function block to the signal chain.
  - You must have enabled event generation for the relevant function block. Refer to [How to Enable Event Generation for Function Blocks](#) on page 504.

### Possible methods

You can choose between two methods when modeling asynchronous tasks for models providing Hardware-Triggered Runnable Function blocks:

- Via the Model-Function Mapping Browser. Refer to Method 1.
- Via the signal chain. Refer to Method 2.

**Method 1****To model asynchronous tasks for models with Hardware-Triggered Runnable Function blocks via the Model-Function Mapping Browser**

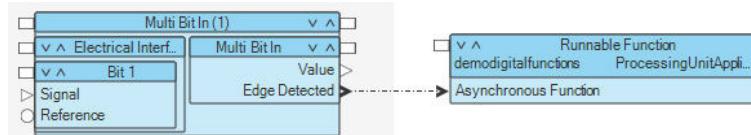
- 1 Open the Model-Function view set.
- 2 In the Model-Function Mapping Browser, drag the relevant function block to the relevant Runnable Function block provided by the Simulink behavior model.

**Note**

If the function block provides multiple I/O events, a list opens for you to select an I/O event. In order to model further asynchronous tasks, you can drag the function block to other unconnected Runnable Function blocks.

**Method 2****To model asynchronous tasks for models with Hardware-Triggered Runnable Function blocks via the signal chain**

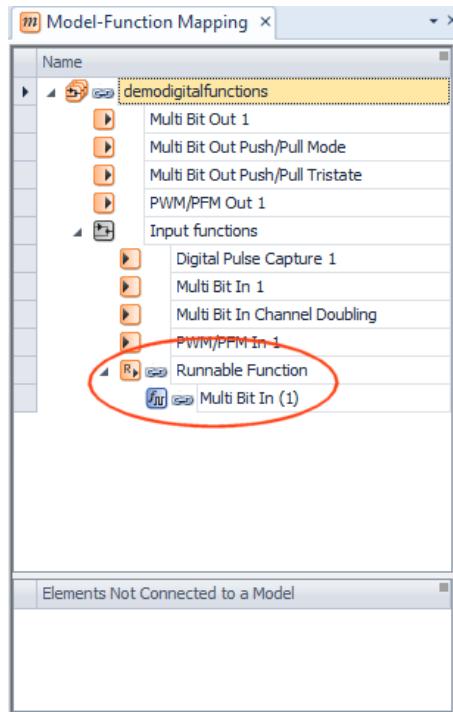
- 1 Open the Signal Chain view set.
- 2 In the Model Browser, drag the relevant Runnable Function block to the signal chain.
- 3 In the signal chain, map the event port of the function block to the Runnable Function port.



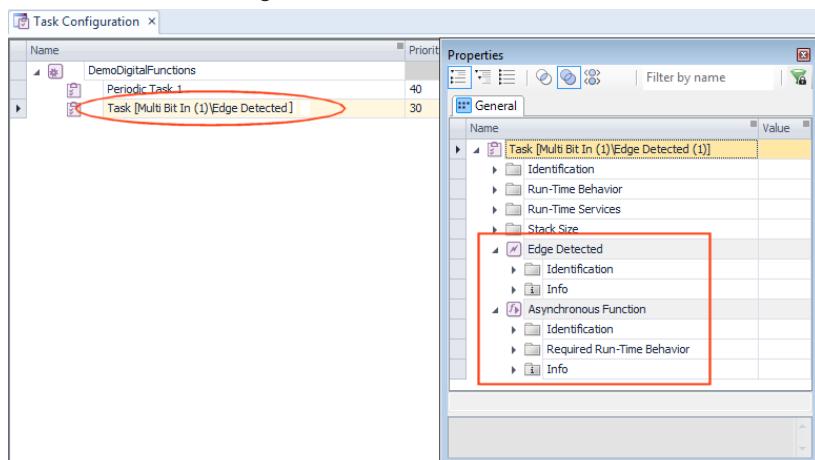
**Result**

You have modeled an asynchronous task for a model with a **Hardware-Triggered Runnable Function** block. This comprises the following results:

- In the Model-Function Mapping Browser, ConfigurationDesk creates a subnode with the selected function block for the relevant Runnable Function block.



- If the runnable function and the I/O event are not assigned to an asynchronous task, ConfigurationDesk creates a new task and assigns the runnable function and the I/O event to it. The task is displayed in the Task Configuration table as shown in the following illustration:



The task has the following configuration:

Property	Configuration
Task name	Task [ <code>&lt;function block name&gt;\&lt;event port name&gt;</code> ]
Task priority	Required priority specified in the Hardware-Triggered Runnable Function block dialog in the Simulink model.
Runnable function name	Runnable function name specified in the Hardware-Triggered Runnable Function block dialog in the Simulink model.
I/O event name	Event port name.

If either the runnable function provided by the Hardware-Triggered Runnable Function block or the selected I/O event is already assigned to an asynchronous task, the task is enhanced by the selected runnable function or the selected I/O event.

## How to Model Asynchronous Tasks and Create Suitable Runnable Function Blocks in One Step

---

**Objective** ConfigurationDesk offers a simple way to model asynchronous tasks and to create suitable Runnable Function blocks in one step.

---

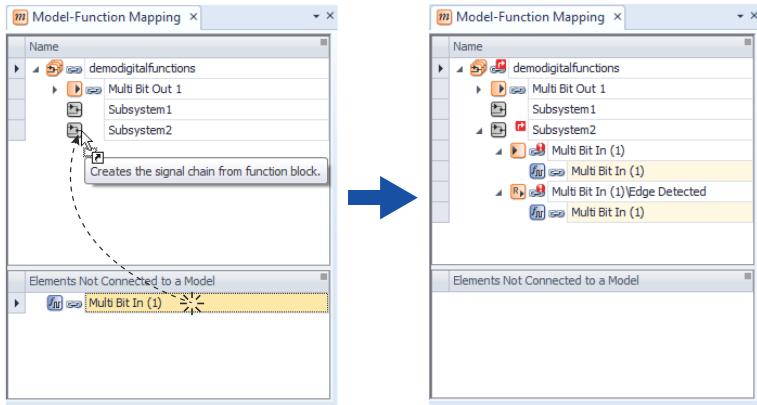
**Preconditions** To perform the steps below, the following preconditions must be fulfilled:

- You must have added the relevant Simulink behavior model to the active ConfigurationDesk application.
- The Simulink behavior model must be assigned to an application process.
- You must have added the relevant function block to the signal chain.
- Event generation must be enabled for the function block. Refer to [How to Enable Event Generation for Function Blocks](#) on page 504.

---

**Method** **To model asynchronous tasks and create suitable Runnable Function blocks in one step**

- 1 Open the Model-Function view set.
- 2 (Necessary only if the relevant function block is contained in the Elements Not Connected to a Model area of the Model-Function Mapping Browser. If the function block is connected to a Simulink behavior model, you can skip this step.) In the Model-Function Mapping Browser, drag the relevant function block from the Elements Not Connected to a Model area to the model root node or the relevant subsystem of the Simulink behavior model.

**Tip**

You can also drag hardware channels or channel sets from the Hardware Resource Browser to the subsystem. This applies to channels or channel sets that support function block types with default event generation, for example, FlexRay blocks. ConfigurationDesk then lets you select a suitable function block type for the channel. If you drag a channel or channel set to a Simulink model or a subsystem in the Model-Function Mapping Browser, ConfigurationDesk performs the following actions in one step:

- Creating suitable function blocks.
- Assigning hardware resources to the function blocks automatically.
- Creating suitable Runnable Function blocks and mapping them to the event ports of the function blocks.
- Creating and modeling new asynchronous tasks.

For more information, refer to [User-Friendly Connection of ConfigurationDesk and Simulink Models](#) on page 553.

### 3 On the Home ribbon, click Models – Propagate.

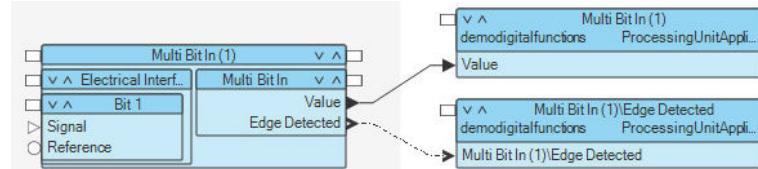
**Result**

You have modeled an asynchronous task and created a suitable Runnable Function block in one step. This comprises the following results:

- ConfigurationDesk creates a new task and assigns the function block's I/O event and a new runnable function to it. The task has the following default configuration:

Parameter	Configuration
Task name	Task [<function block name>\<event port name>]
Task priority	30 (default)
Runnable function name	<function block name>\<event port name>
I/O event name	Event port name

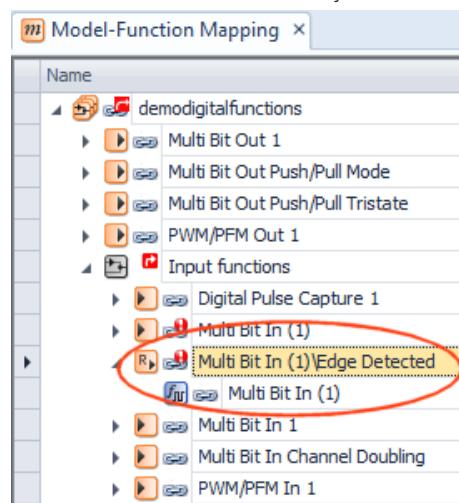
- In the signal chain, ConfigurationDesk creates a new Runnable Function block that is mapped to the function block's event port.



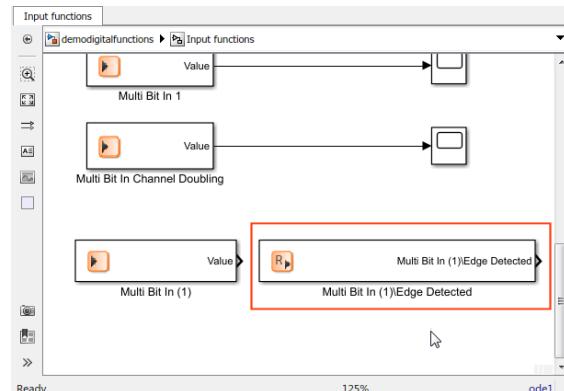
#### Note

If you perform this action, ConfigurationDesk extends the signal chain with model port blocks for all unmapped function ports. If you want ConfigurationDesk to create only Runnable Function blocks, you have to set the function block's Model access property to Disabled.

- In the Model-Function Mapping Browser, ConfigurationDesk creates a new subnode with a Runnable Function block and the function block below the model root or the selected subsystem.



- In the model root or the selected subsystem of the Simulink behavior model, a new Runnable Function block is created.



You can now connect the Hardware-Triggered Runnable Function block to the relevant function-call subsystem.

## How to Model Asynchronous Tasks for Models with Runnable Function Blocks (Manually)

<b>Objective</b>	You can model asynchronous tasks by assigning <a href="#">I/O events</a> to a <a href="#">task</a> manually.
------------------	--

<b>Preconditions</b>	<ul style="list-style-type: none"> <li>▪ You imported a <a href="#">behavior model</a> that provides at least one <a href="#">Runnable Function</a>. Refer to <a href="#">How to Import a Model Topology</a> on page 112.</li> <li>▪ You created a <a href="#">processing unit application</a> and an <a href="#">application process</a>. Refer to <a href="#">How to Model an Executable Application Manually from Scratch</a> on page 489.</li> <li>▪ You assigned the behavior model to the application process.</li> <li>▪ The Executable Application table must be open.</li> </ul>
----------------------	---

<b>Method</b>	<p><b>To model asynchronous tasks for models with Runnable Function blocks (manually)</b></p> <ol style="list-style-type: none"> <li>1 In the Executable Application table, right-click the application process and select New - Task from the context menu.</li> <li>2 Configure the task properties in the Properties Browser according to your requirements. Refer to <a href="#">How to Configure Tasks in ConfigurationDesk</a> on page 515.</li> <li>3 Assign the runnable function to the new task via drag &amp; drop.</li> <li>4 Drag the function block from the Function Browser to the working view. The function block is displayed in the Executable Application table.</li> <li>5 Assign a hardware resource to the function block.</li> <li>6 In the Properties Browser, select Enabled from the Event generation list. The I/O event appears in the Executable Application table.</li> <li>7 Specify the condition of the event. For example, specify the Edge count and Edge type properties according to your requirements.</li> <li>8 Assign the I/O event to the task via drag &amp; drop.</li> </ol>
---------------	--

### Note

- Alternatively, you can assign the I/O event to the task using the Assign event command from the context menu of the task. Refer to [Assign Event \(ConfigurationDesk User Interface Reference\)](#).
- If an event is assigned to multiple tasks, this is displayed as a conflict.

Repeat the above steps for all the runnable functions you want to trigger with asynchronous I/O events.

- 9 Repeat steps 4 - 8 for all the function blocks whose I/O event must trigger the task.
- 

**Result**

You modeled an asynchronous task manually for a behavior model with a Hardware-Triggered Runnable Function block.

# Configuring Tasks in ConfigurationDesk

## Introduction

Configuring tasks in ConfigurationDesk means specifying several task properties. To avoid task overruns due to a cold cache during the first execution of tasks, you can configure the start-up behavior of periodic tasks.

## Where to go from here

## Information in this section

<a href="#">Basics on Configuring Tasks</a> .....	513
<a href="#">How to Configure Tasks in ConfigurationDesk</a> .....	515
<a href="#">How to Configure the Start-Up Behavior of a Periodic Task</a> .....	516
<a href="#">Configuring Delayed Tasks</a> .....	517
<a href="#">Configuring Application Processes for Bus Monitoring</a> .....	519

## Basics on Configuring Tasks

### Introduction

Configuring a [task](#) in ConfigurationDesk means specifying its properties.

### Task properties

The table below shows the available task properties and their descriptions:

Configuration Property	Description
DAQ raster name	To measure variables (e.g., task information variables) stored in the TRC file synchronously with a task, for example, in dSPACE's ControlDesk, the DAQ service must be enabled for that task. The sample time of the task defines the measurement raster or DAQ raster. In a real-time application, each application process can have up to 31 tasks that have their DAQ service enabled. A task's DAQ service is enabled if the DAQ raster name field is not empty. If you have ConfigurationDesk create a predefined application process automatically, the fastest periodic task has its DAQ service enabled. The DAQ service must be enabled for at least one task in each application process. Otherwise, a conflict is shown in the Conflicts Viewer, and the build process is aborted.
Jitter and latency optimization	The Jitter and latency optimization property can have one of the following settings: <ul style="list-style-type: none"> <li>▪ Standard (full functionality) (Default setting for newly created tasks) The task is created as a standard task. This setting should be the first choice for newly created tasks, because it includes no limitations regarding the contents of an application process.</li> </ul>

Configuration Property	Description
	<ul style="list-style-type: none"> <li>▪ Low jitter, low latency The task has low jitter and low latency. You should use Low jitter, low latency to reduce jitter and latency if required. If you use this setting, you must keep in mind the limitations described in <a href="#">Limitations for Low jitter, low latency tasks</a> on page 514.</li> <li>▪ No jitter, low latency The task has no jitter and low latency which lets you achieve smaller sample times for real-time applications. You should configure a task as No jitter, low latency only in cases with high requirements concerning run time, jitter, and latency. If you do, you must keep in mind the limitations described in <a href="#">Limitations for No jitter, low latency tasks</a> on page 515.</li> </ul>
Name	This is the name of the task which is used as an alias, for example, in task-specific messages, or later in dSPACE's experiment tools such as ControlDesk. You can also use the task name to find the task information variables (e.g., Task Call Counter) in the TRC file.
Number of accepted overruns	You can specify the number of overruns that can occur for a specific task before the application stops. 0 means that the application stops at the first overrun. -1 means that an unlimited number of overruns is allowed. The default is 0.
Priority	<p>Each task is assigned a priority according to its relative importance. The RTOS suspends a low-priority task so that a high-priority task is given a turn (preemptive multitasking). When the high-priority task has been executed, the suspended low-priority task resumes execution.</p> <p>For user-defined tasks and tasks provided by Simulink behavior models and Functional Mock-up Units (FMUs), lower values indicate higher task priorities. For tasks provided by V-ECU implementations, it is the other way round: Higher values indicate higher task priorities.</p>
Real-Time Testing	<p>Using Real-Time Testing, you can execute tests synchronously with a task. This is especially useful for FlexRay as the RTT services must be synchronous with a FlexRay application task. If you have ConfigurationDesk create a predefined application process automatically, Real-Time Testing is enabled for the fastest periodic task. You can enable RTT for exactly one task. For details on performing RTT, refer to <a href="#">Real-Time Testing Guide</a>.</p> <p>Real-Time Testing must be enabled for exactly one task in each application process. Otherwise, a conflict is shown in the <a href="#">Conflicts Viewer</a>, and the build process is aborted. If you do not use real-time testing, the real-time testing calls in the generated code do not influence the performance of the real-time application.</p>

**Limitations for Low jitter, low latency tasks** The following limitations apply to application processes that contain tasks with Low jitter, low latency:

- The Low jitter, low latency tasks must have a higher priority than tasks with a different Jitter and Latency Optimization setting.
- A Low jitter, low latency task can only trigger other Low jitter, low latency tasks. In reverse, it can also only be triggered by Low jitter, low latency tasks.
- Real-Time Testing is not supported for Low jitter, low latency tasks.

#### Note

If the preconditions described above are not fulfilled, a conflict is shown in the [Conflicts Viewer](#).

- It is forbidden to use system calls within a Low jitter, low latency task, because they re-introduce jitter.

- It is not possible to use Low jitter, low latency tasks on core 0 of a DS6001 Processor Board. A task configured in this way is automatically executed in Standard (full functionality) mode.
- Behavior models containing blocks from the MotionDesk Blockset cannot be executed in a task configured as Low jitter, low latency.
- Ethernet functionality is not supported for application processes containing a task configured as Low jitter, low latency.
- It cannot be guaranteed that tasks configured as Low jitter, low latency are compatible with dSPACE solutions.

**Limitations for No jitter, low latency tasks** The following limitations apply to application processes that contain a task that is configured as No jitter, low latency:

- The task must be the only task in the application process, and it must be triggered by a timer event. Otherwise, a conflict is shown in the Conflicts Viewer.
- It is not recommended to use system calls within a No jitter, low latency task, because they re-introduce jitter.
- The background task of the application process is not executed.
- It cannot be guaranteed that tasks configured as No jitter, low latency are compatible with dSPACE solutions.
- Behavior models containing blocks from the MotionDesk Blockset cannot be executed in a task configured as No jitter, low latency.
- FPGA signal tracing is not supported for application processes containing a task configured as No jitter, low latency.
- Ethernet functionality is not supported for application processes containing a task configured as No jitter, low latency.

#### Scheduling policy

The scheduling policy is always set to FCFS, except for FlexRay tasks (LCFS).

#### Related topics

##### HowTos

[How to Configure Tasks in ConfigurationDesk.....](#) 515

## How to Configure Tasks in ConfigurationDesk

#### Objective

To configure a task, you have to specify different task properties.

#### Precondition

- The Task Configuration table must be open.
- The ConfigurationDesk application must contain at least one task.

---

<b>Method</b>	<p><b>To configure tasks in ConfigurationDesk</b></p> <ol style="list-style-type: none"> <li>1 In the Task Configuration table, select the task you want to configure.</li> <li>2 In the Task Configuration table, configure the following task properties according to your requirements:           <ul style="list-style-type: none"> <li>▪ Data acquisition (DAQ) raster name</li> <li>▪ Jitter and latency optimization</li> <li>▪ Name</li> <li>▪ Number of accepted overruns</li> <li>▪ Priority</li> <li>▪ Real-Time Testing</li> </ul> </li> </ol> <div style="background-color: #e0e0e0; padding: 5px; border-radius: 5px; margin-top: 10px;"> <p><b>Note</b></p> <p>For a detailed description on the task properties, refer to <a href="#">Basics on Configuring Tasks</a> on page 513.</p> </div>
<b>Result</b>	You configured the task.
<b>Related topics</b>	<p>Basics</p> <div style="background-color: #e0e0e0; padding: 5px; border-radius: 5px; margin-top: 10px;"> <p><a href="#">Basics on Configuring Tasks</a>..... 513</p> </div> <p>References</p> <div style="background-color: #e0e0e0; padding: 5px; border-radius: 5px; margin-top: 10px;"> <p><a href="#">New - Task (ConfigurationDesk User Interface Reference</a> </p> </div>

## How to Configure the Start-Up Behavior of a Periodic Task

---

<b>Objective</b>	To avoid task overruns due to a cold cache during the first execution cycles of periodic tasks.
<b>Multicore and multi-PU applications</b>	Configuring the start-up behavior of periodic tasks affects all the application processes of the application in the same way.
<b>Method</b>	<p><b>To configure the start-up behavior of a periodic task</b></p> <ol style="list-style-type: none"> <li>1 Open the Build Configuration table in the Build view set.</li> <li>2 In the Build Configuration table, click Global Build Settings.</li> </ol>

- 3 In the Properties Browser, specify the Time-Scaled Period and the Time Scale Factor.

The Time-Scaled Period defines the period of time [0 ... T] during which the task sample time is prolonged after application start.

The Time Scale Factor defines the prolonging factor, which must be equal to or greater than 1.0.

## Result

You have configured the startup-behavior of periodic tasks.

## Configuring Delayed Tasks

### Introduction

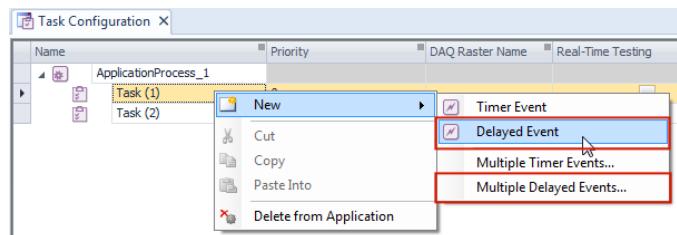
ConfigurationDesk lets you create tasks that are delayed in relation to another task.

### Specifying tasks to be delayed

To execute a task with a delay to another task (source task), ConfigurationDesk lets you create delayed events for the task. A delayed event triggers the task when the specified delay time has expired after the execution of the source task started. ConfigurationDesk provides the following commands to create delayed events:

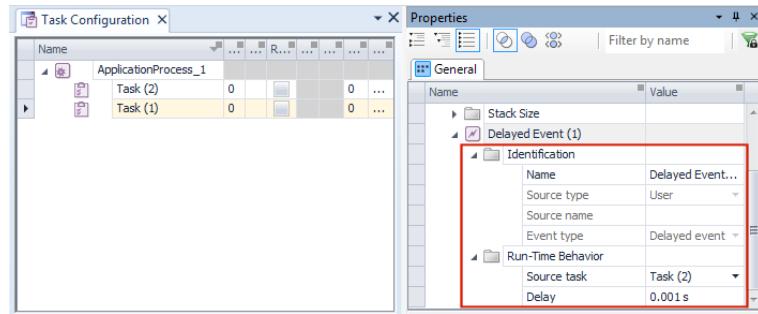
- New - Delayed Event
- New - Multiple Delayed Events

The commands can be selected from the context menu of the task to be delayed in the Task Configuration table. Refer to the following illustration:



## Configuration of delayed events

A delayed event is configured via its Delay, Source task, and Name properties. Refer to the following illustration:



**Delay** The delay specifies the time span from the start of the execution of the source task to the trigger of the delayed event in seconds. You can specify a semicolon-separated list of multiple floating-point values  $\geq 0$ . The default value is 0.001.

If multiple values are specified in form of a semicolon-separated list, the delayed event triggers the delayed task multiple times after the start of the execution of the source task.

### Note

You have to select a delay value that triggers the delayed task before the source task is triggered two more times. For periodically executed source tasks, this means that the delay of the delayed task must be smaller than  $2 * \text{period}$  of the timer event that triggers the source task.

For example, if the period of the timer event that triggers the source task is 0.001 s, the delay of the delayed event must be  $< 0.002$  s.

**Source task** The source task is the task that triggers the delayed event. As soon as the delay time expires after the execution of the source task has started, the source task triggers the delayed event to execute the delayed task.

All tasks in the same application process as the delayed task can be specified as a source task, except for the delayed task itself. Invalid source tasks are highlighted in red in the Source task list in the Properties Browser.

For more information on delayed event properties, refer to [Delayed Event Properties \(ConfigurationDesk User Interface Reference\)](#).

## Related topics

## References

- [Delayed Event Properties \(ConfigurationDesk User Interface Reference\)](#)
- [New – Delayed Event \(ConfigurationDesk User Interface Reference\)](#)
- [New – Multiple Delayed Events \(ConfigurationDesk User Interface Reference\)](#)

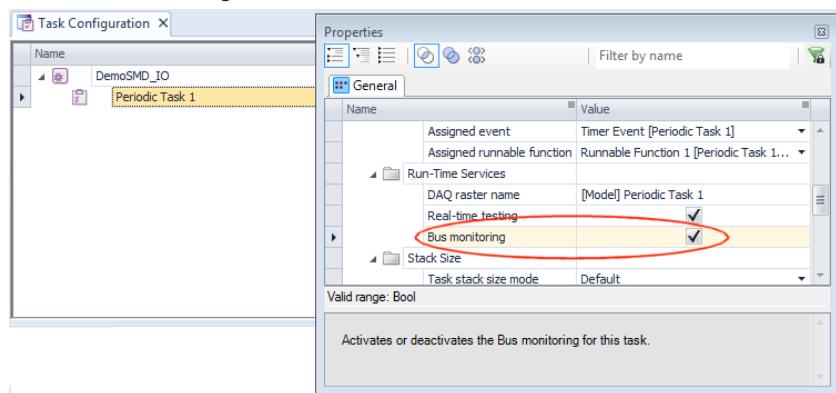
## Configuring Application Processes for Bus Monitoring

### Introduction

Bus monitoring lets you observe the communication transmitted on a bus. In ConfigurationDesk, you can enable bus monitoring for each application process so that you can monitor CAN, LIN, or Ethernet communication via experiment software, such as ControlDesk.

### Enabling bus monitoring for an application processes

ConfigurationDesk provides the Bus monitoring property in the Task Configuration table. The property lets you activate bus monitoring for an application process. You can enable bus monitoring for each application process. Refer to the following illustration:



If enabled, the bus monitoring service is executed in the selected task.

### Requirements for bus monitoring tasks

The following requirements for using bus monitoring must be met:

- Bus monitoring must be activated for exactly one task in each application process.  
If bus monitoring is activated for more than one task or for no task in an application process, a conflict is displayed in the Conflicts Viewer. You must resolve this conflict before you start the build process. Otherwise, the build process is aborted.
- Bus monitoring must only be specified for periodic tasks.  
If you activate bus monitoring for an asynchronous task, a warning is displayed via a conflict in the Conflicts Viewer. You are recommended to resolve this conflict before you start the build process. Otherwise, unexpected results might be displayed in bus monitoring devices in the experiment software.

### Bus monitoring in preconfigured application processes

In preconfigured application processes, bus monitoring is activated for the fastest periodic task by default. You can activate bus monitoring for a different task according to your requirements.

**Related topics**

**References**

[Task Properties \(ConfigurationDesk User Interface Reference\)](#) 

# Using Application Processes Without Behavior Models

## Introduction

ConfigurationDesk provides methods for working with application processes that do not need to have an assigned model implementation. You can use such application processes for bringing simulators into service or for wire testing, for example.

## Where to go from here

### Information in this section

[Introduction to Application Processes Without Behavior Models](#)..... 521

[How to Create Application Processes That Provide Default Tasks](#)..... 523

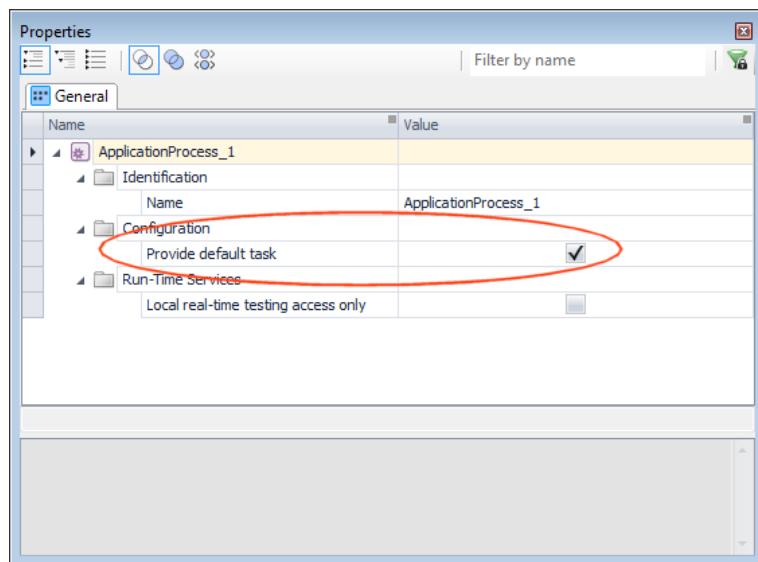
## Introduction to Application Processes Without Behavior Models

### Introduction

ConfigurationDesk lets you create [application processes](#) that provide a default task which is preconfigured for most use cases. If you work with application processes that provide a default task, you do not need to map the I/O functions to the ports of a [behavior model](#).

### Characteristics of the default task

The default task is assigned a resolved [Runnable Function](#) named Communication Step Function and a [timer event](#) with a default configuration (Period = 0.001, Offset = 0). You can change the configuration of the timer event, or assign a new timer event to the default task (indicated by a ). However, the runnable function assigned to the default task cannot be changed (indicated by a ). In the Properties Browser, the Provide default task checkbox is selected.



For instructions on creating application processes that provide default tasks, refer to [How to Create Application Processes That Provide Default Tasks](#) on page 523.

#### Note

You can select the **Provide default task** checkbox for application processes that already exist in your executable application. If the application process has an assigned model implementation, a conflict is shown in the **Conflicts Viewer**. You have to resolve the conflict before you start the build process. Otherwise, the build process will be aborted with an error message.

#### Assignment of function blocks to application processes with default tasks

For the assignment of function blocks (that are not yet assigned to an application process via model port mapping) to application processes with default tasks the following applies:

**Assignment in single-processing-unit applications** In single-processing-unit applications, the function blocks are assigned to the first application process with a suitable task.

**Assignment in multi-processing-unit applications** In multi-processing-unit applications, ConfigurationDesk evaluates the assignment of processing units to processing unit applications. Thus, for each function block results an assignment to a processing unit application based on the function blocks' hardware resource

assignment. Within each processing unit application, the function blocks are assigned to the first application process with a suitable task.

#### Note

- For each processing unit application, it is recommended to create only one application process that provides a default task.
- The assignment of the function blocks to the application processes that provide default tasks is not visible in the Executable Application table in the application processes' Components folder.

#### Related topics

##### HowTos

[How to Create Application Processes That Provide Default Tasks.....](#) 523

##### References

[Application Process Properties \(ConfigurationDesk User Interface Reference\)](#)  
[New - Application Process \(Providing Default Task\) \(ConfigurationDesk User Interface Reference\)](#)  
[New - Multiple Application Processes \(Providing Default Task\) \(ConfigurationDesk User Interface Reference\)](#)

## How to Create Application Processes That Provide Default Tasks

#### Objective

To be able to work with [application processes](#) that do not need to have an assigned [model implementation](#), you must create specific application processes that provide a default task.

#### Method

##### To create application processes that provide default tasks

- 1 In the Task Configuration table, right-click an empty area.
- 2 From the context menu, select New - Application Process (Providing Default Task).

If there is no processing unit application yet, ConfigurationDesk creates a new one. ConfigurationDesk gives the processing unit application a new application process that provides a default task.

#### Result

You have created an application process that provides a default task.

## Next steps

You can now use ConfigurationDesk's test automation support to test the real-time hardware using stimulus signals on certain I/O channels or to test the external devices, for example. For details, refer to [Configuring Test Automation Support \(ConfigurationDesk I/O Function Implementation Guide](#) .

After testing, you can clear the **Provide default task** checkbox in the application processes **Properties Browser** and assign the desired behavior model to the application process. The default task then is converted into a standard user task. The runnable function is replaced with a base rate runnable function when the behavior model is assigned.

---

## Related topics

### Basics

[Introduction to Application Processes Without Behavior Models.....](#) 521

### References

[Application Process Properties \(ConfigurationDesk User Interface Reference](#) 

[New - Application Process \(Providing Default Task\) \(ConfigurationDesk User Interface Reference](#) 

[New - Multiple Application Processes \(Providing Default Task\) \(ConfigurationDesk User Interface Reference](#) 

# Creating Multi-Processing-Unit Applications With ConfigurationDesk

<b>Introduction</b>	ConfigurationDesk lets you build real-time applications for multi-processing-unit systems (multi-PU systems). A multi-PU system is a system of several coupled processing units.								
<b>Required knowledge</b>	Working with multi-processing-unit applications in ConfigurationDesk requires a knowledge of the following topics: <ul style="list-style-type: none"><li>▪ <a href="#">Modeling Executable Applications and Tasks</a> on page 471</li><li>▪ <a href="#">Communication Network of a SCALEXIO System (SCALEXIO Hardware Installation and Configuration)</a></li></ul>								
<b>Where to go from here</b>	<b>Information in this section</b>  <table><tr><td><a href="#">Basics on Multi-Processing-Unit Applications</a>.....</td><td>525</td></tr><tr><td><a href="#">How to Create Processing Unit Applications</a>.....</td><td>527</td></tr><tr><td><a href="#">How to Assign Processing Units to Processing Unit Applications</a>.....</td><td>528</td></tr></table> <b>Information in other sections</b>  <table><tr><td><a href="#">How to Register Hardware for Multi-PU Applications and Add it to a ConfigurationDesk Application</a>.....</td><td>211</td></tr></table>	<a href="#">Basics on Multi-Processing-Unit Applications</a> .....	525	<a href="#">How to Create Processing Unit Applications</a> .....	527	<a href="#">How to Assign Processing Units to Processing Unit Applications</a> .....	528	<a href="#">How to Register Hardware for Multi-PU Applications and Add it to a ConfigurationDesk Application</a> .....	211
<a href="#">Basics on Multi-Processing-Unit Applications</a> .....	525								
<a href="#">How to Create Processing Unit Applications</a> .....	527								
<a href="#">How to Assign Processing Units to Processing Unit Applications</a> .....	528								
<a href="#">How to Register Hardware for Multi-PU Applications and Add it to a ConfigurationDesk Application</a> .....	211								

## Basics on Multi-Processing-Unit Applications

<b>Introduction</b>	A multi-processing-unit system (multi-PU system) is a system of several coupled processing units. Such systems can be useful in cases like the following: <ul style="list-style-type: none"><li>▪ Increasing the available real-time processing power for large real-time applications</li><li>▪ Increasing the number of possible I/O connections</li><li>▪ Testing the interaction between all the ECUs of a target vehicle in combination without integrating them in a real vehicle</li></ul> For an executable application (real-time application) to be executed on the single processing units of a multi-PU system, it must be partitioned into several parts.
---------------------	--

Such partitioned executable applications are called multi-processing-unit applications (multi-PU applications) below.

---

**Multi-PU applications in ConfigurationDesk**

ConfigurationDesk provides a method for you to partition your executable application and assign the parts to the processing units of a multi-PU system that you registered in ConfigurationDesk beforehand. To partition your executable application, you can create several processing unit applications, each composed of one or more application processes.

---

**Assigning processing units to processing unit applications**

To execute processing unit applications on specific processing units, you have to assign the processing units to the processing unit applications. ConfigurationDesk provides the Processing Resource Assignment table of all the processing unit applications, with their application processes and assigned processing units.

After the build and download process, each processing unit application is executed on its assigned processing unit. Each application process contained in the processing unit application is executed on one core of the assigned processing unit.

**Note**

If you create multi-PU applications, it is mandatory for each processing unit application to have a processing unit assigned.

---

**Model communication in multi-PU applications**

Just as in multicore real-time applications, you can easily set up model communication in multi-PU applications by mapping appropriate model ports. Model communication can be performed between any desired application processes via model ports, even if the processing unit applications containing the application processes are assigned to separate processing units that are not connected directly via IOCNET. An indirect connection via one or more processing units is sufficient in this case. Nonetheless, for time-critical model communication, it is recommended to assign the relevant application processes to the same processing unit application or to processing unit applications that are assigned to processing units which are connected directly via IOCNET.

---

**Specific use scenarios**

There are different use scenarios depending on your initial situation. Refer to the following topics depending on your starting point:

- [Workflow for Creating a Multi-PU Application Using Multiple Behavior Models](#) on page 550.
- [Workflow for Creating a Multi-PU Application Using One Overall Behavior Model](#) on page 573.

---

**Limitations of multi-PU applications** For a list of limitations of multi-PU applications, refer to [Limitations Concerning Multi-Processing-Unit Systems](#) on page 767.

---

<b>Related topics</b>	<b>References</b>
-----------------------	-------------------

<a href="#">Create Preconfigured Application Process - In Existing Processing Unit Application</a> (ConfigurationDesk User Interface Reference 
<a href="#">Create Preconfigured Application Process - In New Processing Unit Application</a> (ConfigurationDesk User Interface Reference 
<a href="#">New - Processing Unit Application</a> (ConfigurationDesk User Interface Reference 

## How to Create Processing Unit Applications

---

<b>Objective</b>	To divide an executable application into several parts in order to create a multi-PU application, you must create several <a href="#">processing unit application</a>  in ConfigurationDesk.
------------------	---

<b>Restrictions</b>	The following restrictions apply to processing unit applications:
---------------------	---

- The maximum number of allowed processing unit applications is 32. If you create more processing unit applications than the maximum, a conflict is shown in the [Conflicts Viewer](#).
- Since each processing unit application must be executed on a separate processing unit, the number of processing unit applications must not be higher than the number of processing units in the hardware topology. Otherwise, a conflict is shown in the [Conflicts Viewer](#).

<b>Possible methods</b>	You can create processing unit applications in the executable application in the following ways:
-------------------------	--

- You can create one processing unit application. For instructions, refer to Method 1.
- You can create several processing units at a time. For instructions, refer to Method 2.

<b>Method 1</b>	<b>To create one processing unit application</b>
-----------------	--

- 1 In the Processing Resource Assignment table, right-click the executable application.
- 2 From the context menu, select New — Processing Unit Application.

You can now repeat the above steps to create the required number of processing unit applications.

## Method 2

### To create several processing unit applications at a time

- 1 In the Processing Resource Assignment table, right-click the executable application.
  - 2 From the context menu, select New — Multiple Processing Unit Application.  
The Processing Unit Application dialog opens.
  - 3 In the Processing Unit Application dialog, enter the number of instances, and, if required, a name pattern.
  - 4 Click OK.
- 

## Result

You have created processing unit applications. The processing unit applications created by ConfigurationDesk are numbered consecutively. You can rename them.

### Note

The names of the processing unit applications must be unique. Otherwise, a conflict is shown in the Conflicts Viewer.

## Related topics

### Basics

[Terms and Definitions for Building Executable Applications](#)..... 472

### References

[New - Multiple Processing Unit Applications \(ConfigurationDesk User Interface Reference\)](#)

[New - Processing Unit Application \(ConfigurationDesk User Interface Reference\)](#)

## How to Assign Processing Units to Processing Unit Applications

---

### Objective

You can assign processing units to processing unit applications via the Processing Resource Assignment table.

**Preconditions**

To perform the steps described below, the following preconditions must be fulfilled:

- You must have opened a ConfigurationDesk application.
- You must have created several processing unit applications. Refer to the following topics:
  - [New - Processing Unit Application \(ConfigurationDesk User Interface Reference\)](#)
  - [New - Multiple Processing Unit Applications \(ConfigurationDesk User Interface Reference\)](#)
- You must have added a suitable hardware topology. Refer to [How to Register Hardware for Multi-PU Applications and Add it to a ConfigurationDesk Application](#) on page 211.

**Method****To assign processing units to processing unit applications**

- 1 In the Multiple Models view set, open the Processing Resource Assignment table.
- 2 Assign a processing unit to each processing unit application by selecting a processing unit from the Assigned Processing Unit list.

**Note**

You must not assign the same processing unit to several processing unit applications. Otherwise, a conflict is shown in the **Conflicts Viewer**. Processing units that are already assigned to processing unit applications are marked red in the Assigned Processing Unit list.

**Result**

You have assigned the processing units to the processing unit applications. After the build and download process, each processing unit application is executed on the assigned processing unit.

**Related topics****References**

[Processing Resource Assignment Table \(ConfigurationDesk User Interface Reference\)](#)



# Working with Simulink Behavior Models

---

<b>Introduction</b>	ConfigurationDesk lets you add Simulink models as behavior models to a ConfigurationDesk application.
---------------------	---

---

<b>Where to go from here</b>	<b>Information in this section</b>								
	<table><tr><td>Handling the Model Interfaces of ConfigurationDesk and Simulink Behavior Models.....</td><td>532</td></tr><tr><td>User-Friendly Connection of ConfigurationDesk and Simulink Models.....</td><td>553</td></tr><tr><td>Creating Multimodel Real-Time Applications With Models Separated From One Overall Model.....</td><td>569</td></tr><tr><td>Special Use Cases of Working With Simulink Behavior Models.....</td><td>578</td></tr></table>	Handling the Model Interfaces of ConfigurationDesk and Simulink Behavior Models.....	532	User-Friendly Connection of ConfigurationDesk and Simulink Models.....	553	Creating Multimodel Real-Time Applications With Models Separated From One Overall Model.....	569	Special Use Cases of Working With Simulink Behavior Models.....	578
Handling the Model Interfaces of ConfigurationDesk and Simulink Behavior Models.....	532								
User-Friendly Connection of ConfigurationDesk and Simulink Models.....	553								
Creating Multimodel Real-Time Applications With Models Separated From One Overall Model.....	569								
Special Use Cases of Working With Simulink Behavior Models.....	578								

---

# Handling the Model Interfaces of ConfigurationDesk and Simulink Behavior Models

<b>Introduction</b>	The I/O functionality in ConfigurationDesk and Simulink behavior models are connected via model interfaces.																		
<b>Required knowledge</b>	To work with Simulink behavior models, it is assumed that you are familiar with the basics of specifying the model interface. Refer to <a href="#">Specifying the Model Interface</a> on page 417.																		
<b>Where to go from here</b>	<b>Information in this section</b>																		
	<table border="0"> <tr> <td><a href="#">Handling MATLAB via ConfigurationDesk</a>.....</td> <td>532</td> </tr> <tr> <td><a href="#">Basics of Connecting ConfigurationDesk and Simulink Behavior Models</a>.....</td> <td>534</td> </tr> <tr> <td><a href="#">Effects of Changing Model Interfaces of Simulink Behavior Models</a>.....</td> <td>538</td> </tr> <tr> <td><a href="#">Synchronizing the Model Interfaces of ConfigurationDesk and Simulink</a>.....</td> <td>540</td> </tr> <tr> <td><a href="#">Transferring the ConfigurationDesk Model Interface to a New Simulink Interface Model</a>.....</td> <td>542</td> </tr> <tr> <td><a href="#">How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model</a>.....</td> <td>543</td> </tr> <tr> <td><a href="#">How to Initialize Simulink Behavior Models</a>.....</td> <td>546</td> </tr> <tr> <td><a href="#">Workflow for Creating a Multicore Real-Time Application Using Multiple Behavior Models</a>.....</td> <td>548</td> </tr> <tr> <td><a href="#">Workflow for Creating a Multi-PU Application Using Multiple Behavior Models</a>.....</td> <td>550</td> </tr> </table>	<a href="#">Handling MATLAB via ConfigurationDesk</a> .....	532	<a href="#">Basics of Connecting ConfigurationDesk and Simulink Behavior Models</a> .....	534	<a href="#">Effects of Changing Model Interfaces of Simulink Behavior Models</a> .....	538	<a href="#">Synchronizing the Model Interfaces of ConfigurationDesk and Simulink</a> .....	540	<a href="#">Transferring the ConfigurationDesk Model Interface to a New Simulink Interface Model</a> .....	542	<a href="#">How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model</a> .....	543	<a href="#">How to Initialize Simulink Behavior Models</a> .....	546	<a href="#">Workflow for Creating a Multicore Real-Time Application Using Multiple Behavior Models</a> .....	548	<a href="#">Workflow for Creating a Multi-PU Application Using Multiple Behavior Models</a> .....	550
<a href="#">Handling MATLAB via ConfigurationDesk</a> .....	532																		
<a href="#">Basics of Connecting ConfigurationDesk and Simulink Behavior Models</a> .....	534																		
<a href="#">Effects of Changing Model Interfaces of Simulink Behavior Models</a> .....	538																		
<a href="#">Synchronizing the Model Interfaces of ConfigurationDesk and Simulink</a> .....	540																		
<a href="#">Transferring the ConfigurationDesk Model Interface to a New Simulink Interface Model</a> .....	542																		
<a href="#">How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model</a> .....	543																		
<a href="#">How to Initialize Simulink Behavior Models</a> .....	546																		
<a href="#">Workflow for Creating a Multicore Real-Time Application Using Multiple Behavior Models</a> .....	548																		
<a href="#">Workflow for Creating a Multi-PU Application Using Multiple Behavior Models</a> .....	550																		

## Handling MATLAB via ConfigurationDesk

<b>Details on the MATLAB start-up</b>	If you execute an operation in ConfigurationDesk that requires access to MATLAB, and MATLAB is not running, ConfigurationDesk starts it. The working directory of MATLAB depends on the following points: <ul style="list-style-type: none"> <li>▪ MATLAB is already started</li> <li>▪ The working directory of the running MATLAB session is used.</li> </ul>
---------------------------------------	---

- MATLAB is started by ConfigurationDesk
  - If MATLAB is started due to an operation (for example, Show model port block, or Analyze Model (Including Task Information)) performed on an element in a Simulink behavior model in the ConfigurationDesk application, the MATLAB working directory is set to the model path.  
If the path does not exist because the model reference is not valid, the following rule is applied:
  - If MATLAB is started in any other context, the MATLAB working directory is set to the root directory of the active ConfigurationDesk application.

**Note**

If you have connected several MATLAB installations to your dSPACE installation, you must select one of them as the preferred connection. For details, refer to [Introduction to Connecting a MATLAB Installation \(Managing dSPACE Software Installations\)](#).

**Closing MATLAB**

To ensure efficient model handling, ConfigurationDesk prevents MATLAB from being closed while a MATLAB-related task is running in ConfigurationDesk. If you close ConfigurationDesk, MATLAB is not automatically closed, too.

**Opening the Simulink behavior model with initial values**

In ConfigurationDesk, you can specify an initialization command that is executed before the Simulink behavior model added to the ConfigurationDesk application is opened by ConfigurationDesk.

The command is not executed if the Simulink behavior model is already open. To activate the initialization command, you must close the model and reopen it from ConfigurationDesk, or you can type the command in the MATLAB Command Window and execute it.

Note the following points when using an initialization command:

- The command is executed in the MATLAB workspace.
- The working directory is set to the model's path during execution.
- The command can consist of several MATLAB commands.
- You can change the working directory and enlarge the MATLAB search path.
- The model will not be opened if an error occurs.

**Example of a valid initialization command**

```
cd(fullfile(getenv('DSPACE_ROOT'), 'Work', 'MyProject'));
modelVariant='6CylinderDiesel'; load('gear_data.mat');
init_motor(modelVariant);
```

For details on the initialization command, refer to [Model Implementation Properties \(ConfigurationDesk User Interface Reference\)](#).

**Saving Simulink behavior models via ConfigurationDesk**

When you save an application, the Simulink behavior model linked to it is not automatically saved, too. ConfigurationDesk checks, however, if the Simulink behavior model has been changed. If this is true, a confirmation prompt appears in MATLAB asking you if you want to save the Simulink behavior model, too. If the Simulink behavior model uses model referencing, your application is saved together with the top-level model and all open and modified referenced models. If the Simulink behavior model uses open or changed Simulink libraries, a confirmation prompt appears asking you if you want to save them, too.

**Note**

Confirmation prompts are displayed in MATLAB. Depending on your Windows setting, it can happen that the confirmation prompts do not appear in the foreground, but in the background. You can avoid this behavior by using two monitors, one for ConfigurationDesk, one for MATLAB.

**Related topics****Basics**

Analyzing Simulink Behavior Models.....	561
Switching Between Model Port Blocks in ConfigurationDesk and in Simulink.....	566

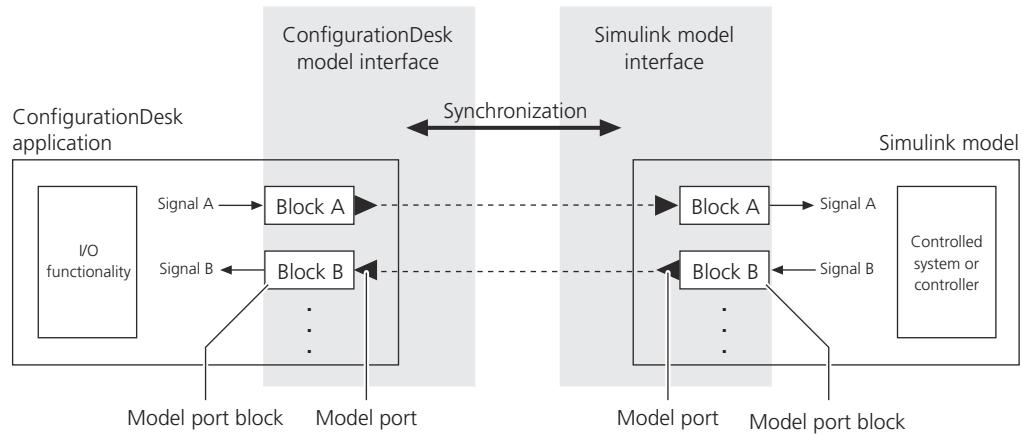
**HowTos**

How to Initialize Simulink Behavior Models.....	546
How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model.....	543

## Basics of Connecting ConfigurationDesk and Simulink Behavior Models

**Model interfaces in ConfigurationDesk and the Simulink behavior model**

The connection between the I/O functionality in ConfigurationDesk and Simulink behavior models is realized via model interfaces. There is a ConfigurationDesk model interface and a Simulink model interface. Both model interfaces are implemented via [model port blocks](#) containing at least one model port. Model port blocks can be created in ConfigurationDesk or in the Simulink behavior model. If you add a Simulink model to your active ConfigurationDesk application, ConfigurationDesk analyzes the model interface and the task information of the Simulink behavior model automatically by default. The ConfigurationDesk application then contains a model interface representing the structure of the Simulink behavior model.



#### Block and port identification in Simulink models

Model port blocks as well as model ports need a unique identity. This ensures that model port blocks and model ports can be clearly identified independently of their names or positions in a model. A model port block in ConfigurationDesk and the corresponding model port block in the Simulink behavior model are a pair if they have the same identity (ID).

For details on ID handling, refer to [Basics on Model Port Block IDs and Signal IDs \(Model Interface Package for Simulink - Modeling Guide\)](#).

#### Modifying and synchronizing the model interfaces

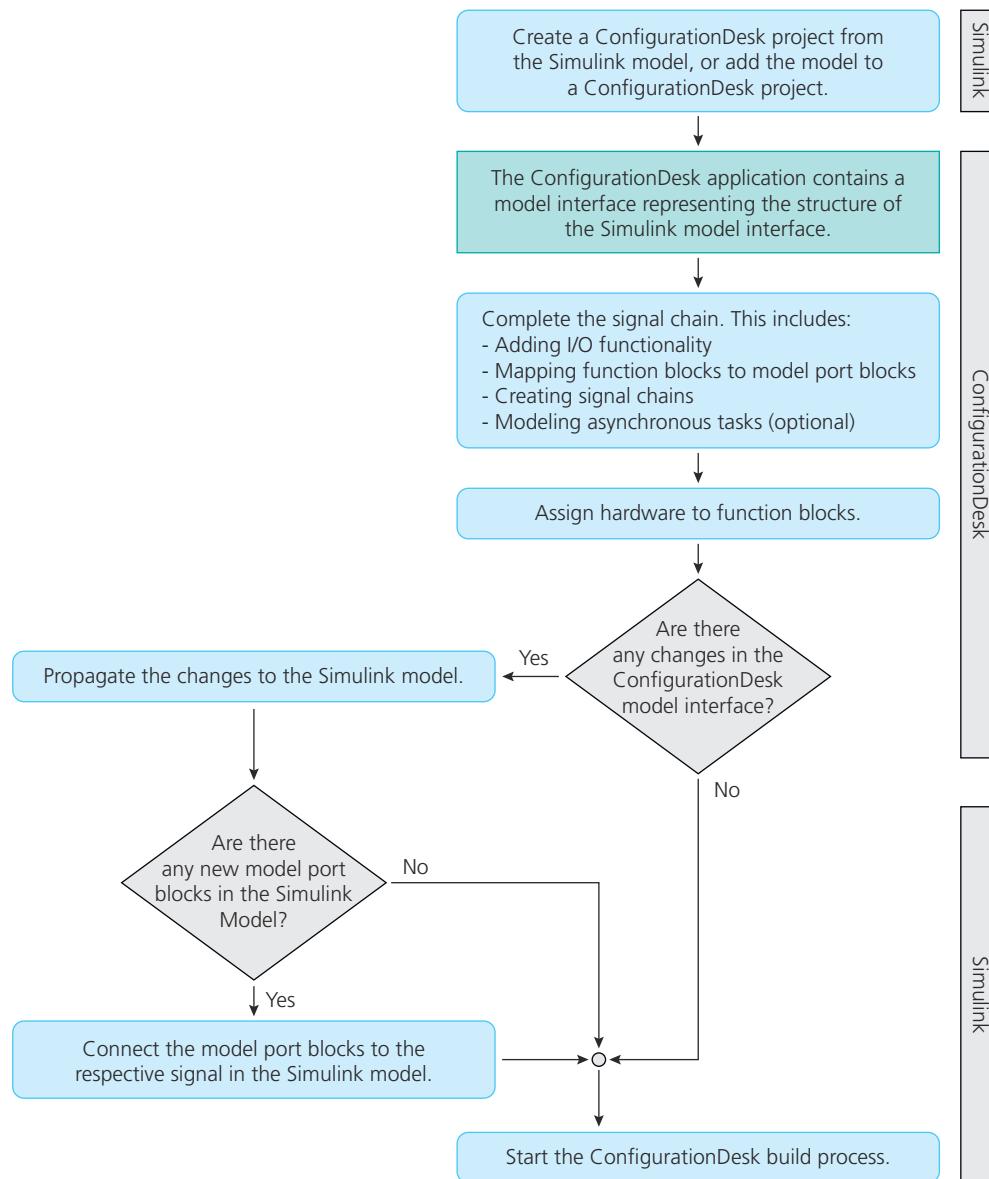
If either the ConfigurationDesk model interface or the Simulink model interface is modified, for example, by adding new model port blocks, you must synchronize the model interfaces to make the model port blocks available in both models.

##### Modifications in the ConfigurationDesk model interface

ConfigurationDesk lets you create signal chains by extending function blocks with suitable model port blocks. The new model port blocks in ConfigurationDesk automatically become a part of your application, but they are not yet used for code generation because they are still unresolved. This means that they are not contained in the Simulink behavior model that is added to your ConfigurationDesk application. To make them resolved, you must propagate the new model port blocks to the Simulink behavior model.

**Modifications in the Simulink model interface** In the Simulink behavior model, you can add new model port blocks to the Simulink model interface using the blocks from the Model Interface Blockset. You can make new model port blocks in the Simulink model interface available in ConfigurationDesk by analyzing the Simulink model interface in ConfigurationDesk.

The following illustration shows a possible workflow of modifying and synchronizing the model interfaces in ConfigurationDesk and in the Simulink behavior model:



For details on the effects on modifying the model interface of Simulink behavior models and synchronizing both model interfaces, refer to the following topics:

- [Effects of Changing Model Interfaces of Simulink Behavior Models on page 538](#)
- [Synchronizing the Model Interfaces of ConfigurationDesk and Simulink on page 540.](#)

#### User-friendly connection between ConfigurationDesk and Simulink

ConfigurationDesk provides the **Model-Function Mapping Browser** for a simplified handling of Simulink behavior models. The **Model-Function Mapping Browser** lets you easily create signal chains via drag & drop. You can then propagate any changes in the ConfigurationDesk model interface directly to

the Simulink behavior model. For details, refer to [User-Friendly Connection of ConfigurationDesk and Simulink Models](#) on page 553.

#### Data type mapping

If you add a Simulink behavior model to a ConfigurationDesk application and perform a model analysis, the data types specified in the Simulink behavior model are automatically mapped to the data types supported by ConfigurationDesk like this:

Simulink Data Type	ConfigurationDesk Data Type
int8	Int8 (-128 ...+127)
uint8	UInt8 (0 ... +255)
int16	Int16 (-32768 ... +32767)
uint16	UInt16 (0 ... 65535)
int32	Int32 (-2147483648 ... +2147483647)
uint32	UInt32 (0 ... +4294967295)
int64	Int64 (-9223372036854775808 ... 9223372036854775807)
uint64	Uint64 (0 ... 18446744073709551615)
Boolean	Bool (0/1)
single	Float32 (-3.402823E+38 ... +3.402823E+38)
double	Float64

#### Transferring model port blocks to a Simulink model via a new interface model

ConfigurationDesk lets you transfer model port blocks to a Simulink model via a new [interface model](#). You can use the interface model as the basis for a new Simulink behavior model, or you can transfer model port blocks from the interface model to an existing Simulink behavior model. Refer to the following topics:

- [Transferring the ConfigurationDesk Model Interface to a New Simulink Interface Model](#) on page 542.
- [How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model](#) on page 543.

#### Initializing Simulink models

ConfigurationDesk allows you to call MATLAB commands to initialize Simulink [behavior models](#). This also enables you to use the same model for different variants. Refer to [How to Initialize Simulink Behavior Models](#) on page 546.

## Effects of Changing Model Interfaces of Simulink Behavior Models

### Modified model port blocks

If you modify model port blocks in the Simulink [behavior model](#) which is added to your active ConfigurationDesk application and run it through [model analysis](#) again, the modifications take immediate effect in ConfigurationDesk. The following table lists actions in the Simulink behavior model and their effect on the display of model port blocks in ConfigurationDesk after a model analysis and on code generation.

User Action in Simulink Behavior Model	Display of Model Port Blocks	Effect on Code Generation
Adding a new model port block to your Simulink behavior model.	It is displayed in the Model-Function Mapping Browser.	It is used for code generation.
Removing a model port block from your Simulink behavior model.	<p>If the model port block is used in the application the model port block becomes unresolved. This is displayed as follows:</p> <ul style="list-style-type: none"> <li>▪ It is displayed with a warning symbol in the Model-Function Mapping Browser.</li> <li>▪ If the highlighting of signal chain elements is enabled, the color of the block frame changes from black to orange in working views.</li> </ul> <p>If the model port block is not used in the application, it is no longer displayed in the Model-Function Mapping Browser.</p>	It is no longer used for code generation.
Copying a new model port block with its identity to your Simulink behavior model which was created in ConfigurationDesk. (You must have generated the new model port block beforehand. For instructions, refer to <a href="#">How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model</a> on page 543).	<p>The model port block becomes unresolved. This is displayed as follows:</p> <ul style="list-style-type: none"> <li>▪ It is no longer displayed with a warning symbol in the Model-Function Mapping Browser.</li> <li>▪ If the highlighting of signal chain elements is enabled, the color of the block frame changes from orange to black in working views.</li> </ul>	It is used for code generation.
Moving a model port block to a different subsystem.	It is displayed at a different position in the Model-Function Mapping Browser.	It is used for code generation.

If you want to copy model port blocks from the temporary interface model to the Simulink behavior model, you must use the standard Copy command together with the Paste and Keep IDs command. For more information on the model port identity, refer to [Basics on Model Port Block IDs and Signal IDs \(Model Interface Package for Simulink - Modeling Guide\)](#).

For further MATLAB actions such as changing model port block properties or model port properties, refer to [Synchronizing the Simulink Model Interface and the ConfigurationDesk Model Interface \(Model Interface Package for Simulink - Modeling Guide\)](#).

#### **Modifications related to separated models based on an overall model**

If your multimodel application is based on an overall model, the information on the separated models and their interconnections is provided via an MCD file loaded to the model topology of your active ConfigurationDesk application.

If model separation changes are made (in the Model Separation Setup tool), the MCD file must be updated to make the changes available in ConfigurationDesk. The following table lists actions in the Simulink behavior model and the corresponding required actions in ConfigurationDesk.

User Action in Overall Simulink Behavior Model	Action Required in Modeling Tool	Actions Required in ConfigurationDesk
Changing connections between top-level subsystems which are available as separated models	Create selected models (and update the MCD file).	1. Reload the MCD file. <sup>1)</sup> 2. Check the updated model communication in the Model Communication Package table. If necessary, change the settings.
Defining a new top-level subsystem to be separated as a model.	Create selected models (and update the MCD file).	1. Reload the MCD file. <sup>1)</sup> 2. Analyze the model interface of the new model which is added to the model topology. 3. Add new model port blocks which are not part of the application to the signal chain. 4. Complete the model port mapping. 5. Check the updated model communication in the Model Communication Package table. If necessary, change the settings.
Changing functions in a part of the overall model which is assigned to a separated model.		
Changing the name of a separated model.	Create selected models (and update the MCD file).	1. Reload the MCD file. <sup>1)</sup>

<sup>1)</sup> If the reload command is not available, the reference to the MCD file is cleared. In this case you have to add the MCD file to the model topology again. For instructions, refer to [How to Import a Model Topology](#) on page 112.

#### **Related topics**

##### **Basics**

Analyzing Simulink Behavior Models.....	561
Transferring the ConfigurationDesk Model Interface to a New Simulink Interface Model.....	542

##### **HowTos**

How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model.....	543
---	-----

## Synchronizing the Model Interfaces of ConfigurationDesk and Simulink

### Synchronizing model interfaces

If you modify either the ConfigurationDesk model interface or the Simulink model interface, both model interfaces must be re-synchronized. You can synchronize the model interfaces in ConfigurationDesk as well as in Simulink.

**Changes in the ConfigurationDesk model interface** If you modified the ConfigurationDesk model interface, you can propagate the changes to the Simulink model via the **Propagate to Simulink Model** command. For details, refer to [Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model](#) on page 558.

**Changes in the I/O functionality in ConfigurationDesk** If you modified the I/O functionality in ConfigurationDesk, and the modifications affect the ConfigurationDesk model interface, you can update the ConfigurationDesk model interface and the Simulink model interface in one step via the **Propagate to Simulink Model** command.

**Changes in the Simulink model interface** To make modifications in the Simulink model interface known to ConfigurationDesk, you can start a model analysis via one of the following context menu commands of the Simulink model in the Model-Function Mapping Browser:

- [Analyze Simulink Model \(Including Task Information\)](#)
- [Analyze Simulink Model \(Model Interface Only\)](#)

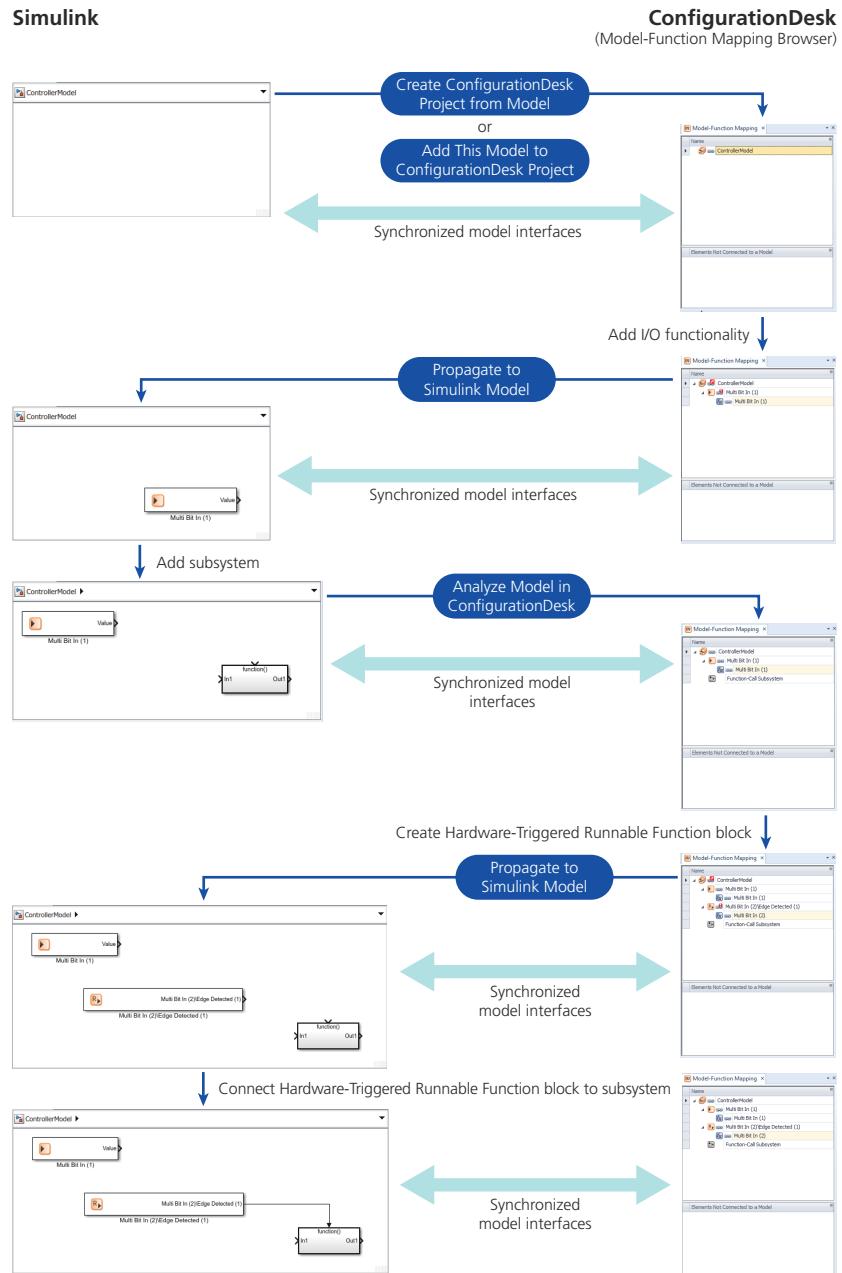
For details, refer to [Analyzing Simulink Behavior Models](#) on page 561.

#### Tip

Commands for analyzing the Simulink model in ConfigurationDesk are also available in Simulink.

### Example of modifying and re-synchronizing model interfaces

The following illustration shows an example of modifying and re-synchronizing the model interfaces in Simulink and in ConfigurationDesk:



**Note**

The following applies if you use the **Propagate to Simulink Model** command:

- You cannot undo the changes in the Simulink model. To prevent a propagate operation from accidentally changing parts of a Simulink model, you can protect specific subsystems by setting them to read-only in Simulink. When analyzed, these subsystems are marked with a lock symbol in ConfigurationDesk.
- Possible formatting of the model port blocks in Simulink, e.g., the color, is lost.
- This command cannot be used for Configuration Port blocks.

## Transferring the ConfigurationDesk Model Interface to a New Simulink Interface Model

### Introduction

ConfigurationDesk provides methods to transfer the ConfigurationDesk model interface to a new Simulink interface model. You can use the new interface model to model a new Simulink behavior model, or you can transfer model port blocks from the new interface model to an existing Simulink behavior model.

### Workflow steps

To transfer the ConfigurationDesk model interface to an existing Simulink behavior model via a new Simulink interface model, the following workflow steps are necessary:

1. Generating interfaces for the Simulink behavior model

The unresolved model port blocks in your ConfigurationDesk application must be generated as model port blocks in a new Simulink model called the interface model. You can do so by using ConfigurationDesk's **Generate New Simulink Model Interface** command.

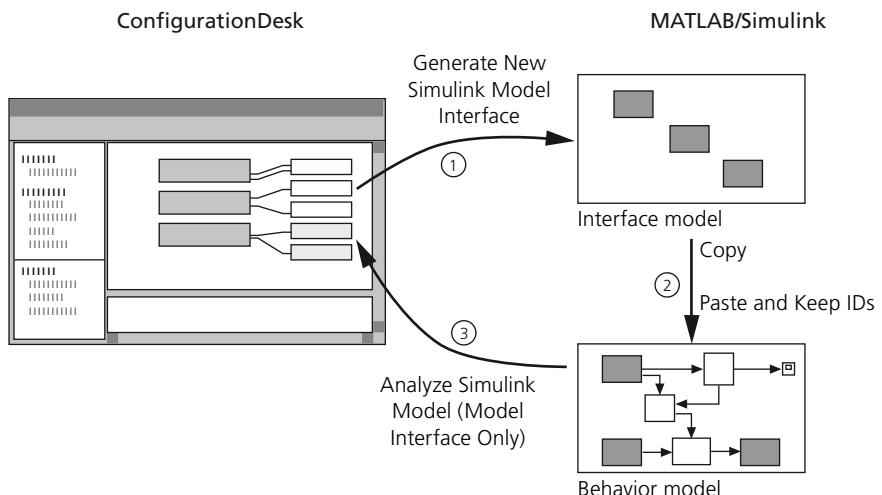
2. Copying model port blocks from the interface model to the Simulink behavior model

You can copy the generated model port blocks blocks from the interface model and paste them to your Simulink behavior model. For this purpose, you must use the standard **Copy** command together with the **Paste and Keep IDs** command from the **Model Port Blocks** menu.

3. Analyzing interfaces of the Simulink behavior model

To make the changes in your Simulink behavior model known to ConfigurationDesk, you must analyze its model interface by executing ConfigurationDesk's **Analyze Simulink Model (Model Interface Only)** command. During model analysis, the model topology of your active ConfigurationDesk application is updated with the model port block data of the Simulink behavior model.

As a final result of these work steps, the new model port blocks in ConfigurationDesk are resolved and used for code generation from now on. These work steps can be visualized as follows:



#### Note

If you generate the Simulink model interface via the **Generate New Simulink Model Interface** command, the values of the model port block and model port properties contained in ConfigurationDesk are transferred to the [interface model](#). When you copy the model port blocks from the interface model to the Simulink behavior model by using the standard **Copy** command together with the **Paste and Keep IDs** command, the model port blocks keep their identities. If you subsequently perform an analysis of the Simulink model, ConfigurationDesk identifies these model port blocks and displays them as resolved. Refer to [Basics of Connecting ConfigurationDesk and Simulink Behavior Models](#) on page 534.

For details on the steps described above, refer to the following topics:

- [How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model](#) on page 543
- [Adding the Generated Model Interface to Your Behavior Model via an Interface Model \(Model Interface Package for Simulink - Modeling Guide\)](#)

## How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model

### Objective

You can have ConfigurationDesk generate [model port blocks](#) of your [signal chain](#) into an [interface model](#). Then you can copy the generated blocks into an existing Simulink behavior model.

**Notes on MATLAB**

If you have connected several MATLAB installations to your dSPACE installation, you must select one of them as the preferred connection. For details, refer to [Introduction to Connecting a MATLAB Installation \(Managing dSPACE Software Installations\)](#).

---

**Precondition**

The following preconditions must be fulfilled:

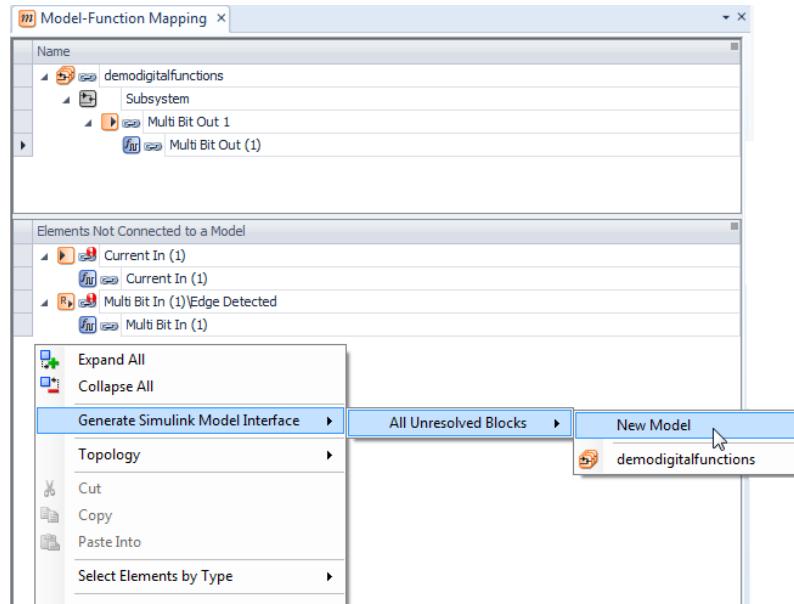
- You must have opened a ConfigurationDesk application with an added Simulink behavior model.
  - The ConfigurationDesk application must contain at least one model port block that is not yet contained in the Simulink behavior model.
- 

**Method****To transfer unresolved model port blocks to a Simulink behavior model via an interface model**

- 1 In ConfigurationDesk, open the Model-Function view set.
- 2 (Optional) If you want to transfer only specific model port blocks to the Simulink behavior model, select the desired model port blocks in the Elements Not Connected to a Model area of the Model-Function Mapping Browser.
- 3 Right-click the Elements Not Connected to a Model area of the Model-Function Mapping Browser.
- 4 This step depends on your choice:
  - If you want to transfer only the selected model port blocks to the Simulink behavior model:  
From the context menu of the Elements Not Connected to a Model area, select **Generate Simulink Model Interface - Selected Blocks - New Model**.
  - If you want to transfer all unresolved model port blocks to the Simulink behavior model:

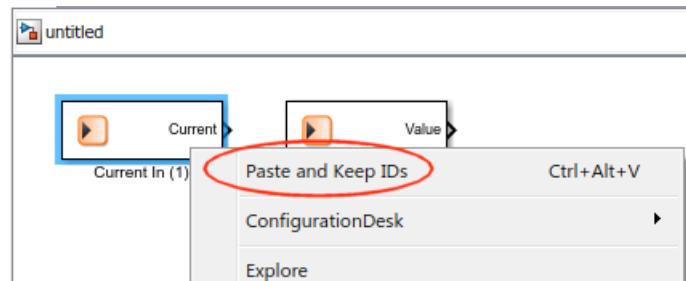
From the context menu, select Generate Simulink Model Interface - All Unresolved Blocks - New Model.

The following illustration shows an example in which all unresolved model port blocks are transferred to the Simulink behavior model.



A new Simulink model is created. The new Simulink model contains a suitable model port block from the Model Interface Blockset for each unresolved model port block in ConfigurationDesk you want to transfer to the Simulink behavior model. These model port blocks must be copied to the Simulink behavior model with a special command.

- 5 From the context menu of the model port blocks, select Copy.
- 6 Open the Simulink behavior model and right-click in a free area.
- 7 From the context menu of the Simulink behavior model, select Paste and Keep IDs.



The model port blocks are copied to the Simulink behavior model where they can be connected to other blocks.

#### Note

If you use the **Copy** command together with the standard **Paste** command to copy the data port blocks from the generated interface model to your Simulink behavior model, the block identities are lost. The model port blocks are then treated as new model port blocks in ConfigurationDesk. They have no mapping information for the signal chain in ConfigurationDesk.

- 8** From the ConfigurationDesk menu in the Simulink behavior model, select one of the following commands:
  - Analyze Model in ConfigurationDesk (Including Task Information)
  - Analyze Model in ConfigurationDesk (Model Interface Only)

#### Note

The commands for analyzing Simulink models are also available in ConfigurationDesk, for example, in the context menu of the Simulink behavior model in the **Model-Function Mapping Browser**.

ConfigurationDesk performs a model analysis. As a result, the model port blocks are now resolved in ConfigurationDesk.

#### Result

You have transferred unresolved model port blocks to a Simulink behavior model via an interface model.

#### Related topics

##### Basics

Analyzing Simulink Behavior Models.....	561
Handling MATLAB via ConfigurationDesk.....	532

##### References

Generate Simulink Model Interface – All Unresolved Blocks – New Model (ConfigurationDesk User Interface Reference  <td></td>	
--	--

## How to Initialize Simulink Behavior Models

#### Objective

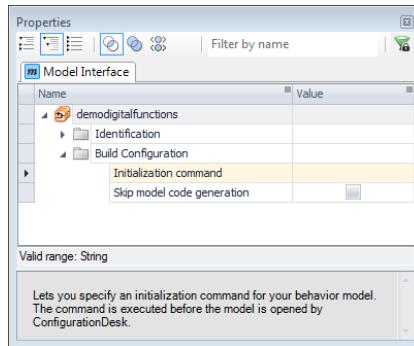
ConfigurationDesk allows you to call MATLAB commands to initialize Simulink behavior models .

**Preconditions**

If you want to use a MATLAB function as an initialization command, it must be located in the Simulink behavior model folder or in a folder on the MATLAB search path.

**Method****To initialize Simulink behavior models**

- 1 In the Model-Function Mapping Browser, click the model item.  
The properties of the model are displayed in the Properties Browser.
- 2 In the Initialization command edit field, enter the model initialization command.



ConfigurationDesk lets you specify locations of scripts, data, etc. via relative paths in the Initialization command edit field. This enables you to transfer the ConfigurationDesk project to another path (refer to [How to Back up and Transfer a Project](#) on page 98). For this purpose, ConfigurationDesk provides the following path macros:

Path Macro Name	Description
%ProjectRoot%	Expands to the root directory of the current project
%ApplicationRoot%	Expands to the root directory of the currently activated ConfigurationDesk application
%ModelRoot%	Expands to the directory the selected Simulink model is stored in

**Result**

You specified an initialization command for your Simulink behavior model. The command is executed when the Simulink behavior model opens.

**Related topics**

**Basics**

[Handling MATLAB via ConfigurationDesk.....](#) 532

**References**

[Model Implementation Properties \(ConfigurationDesk User Interface Reference\)](#)

## Workflow for Creating a Multicore Real-Time Application Using Multiple Behavior Models

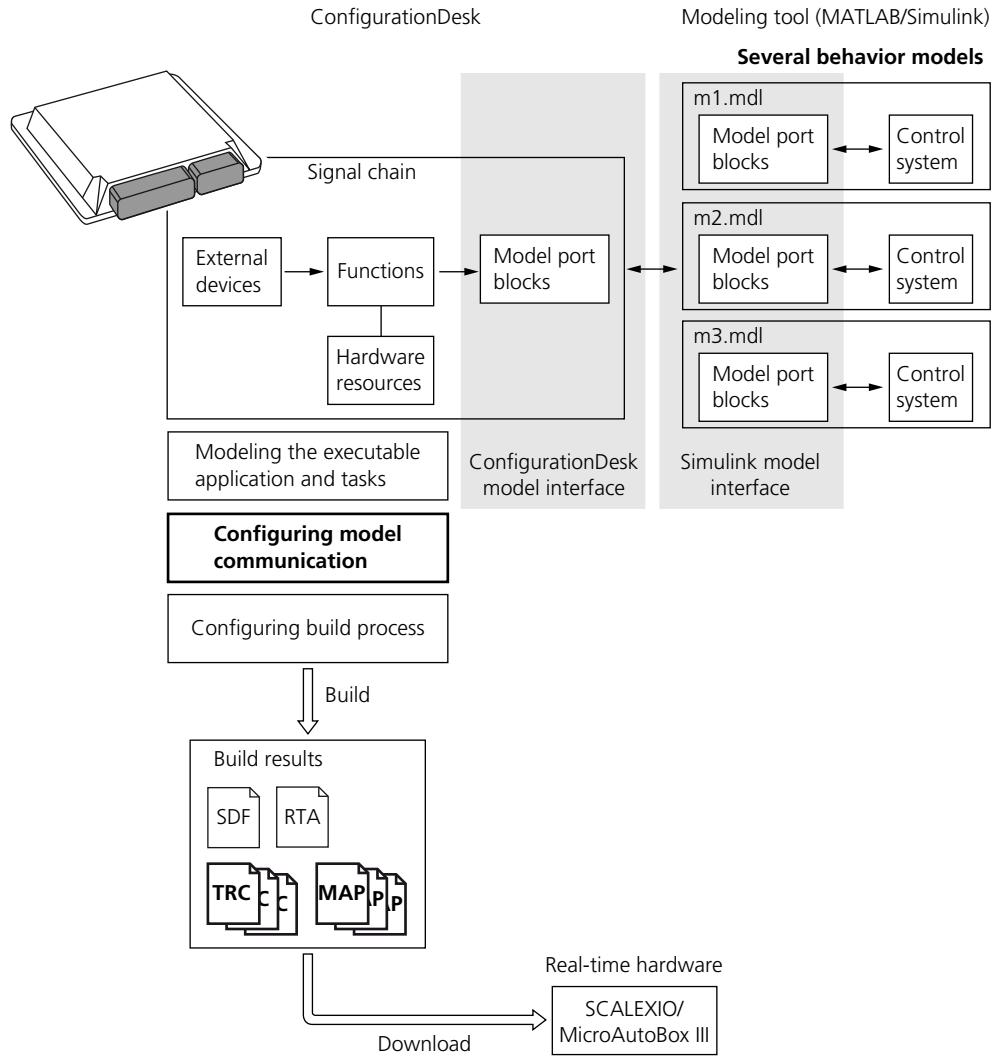
---

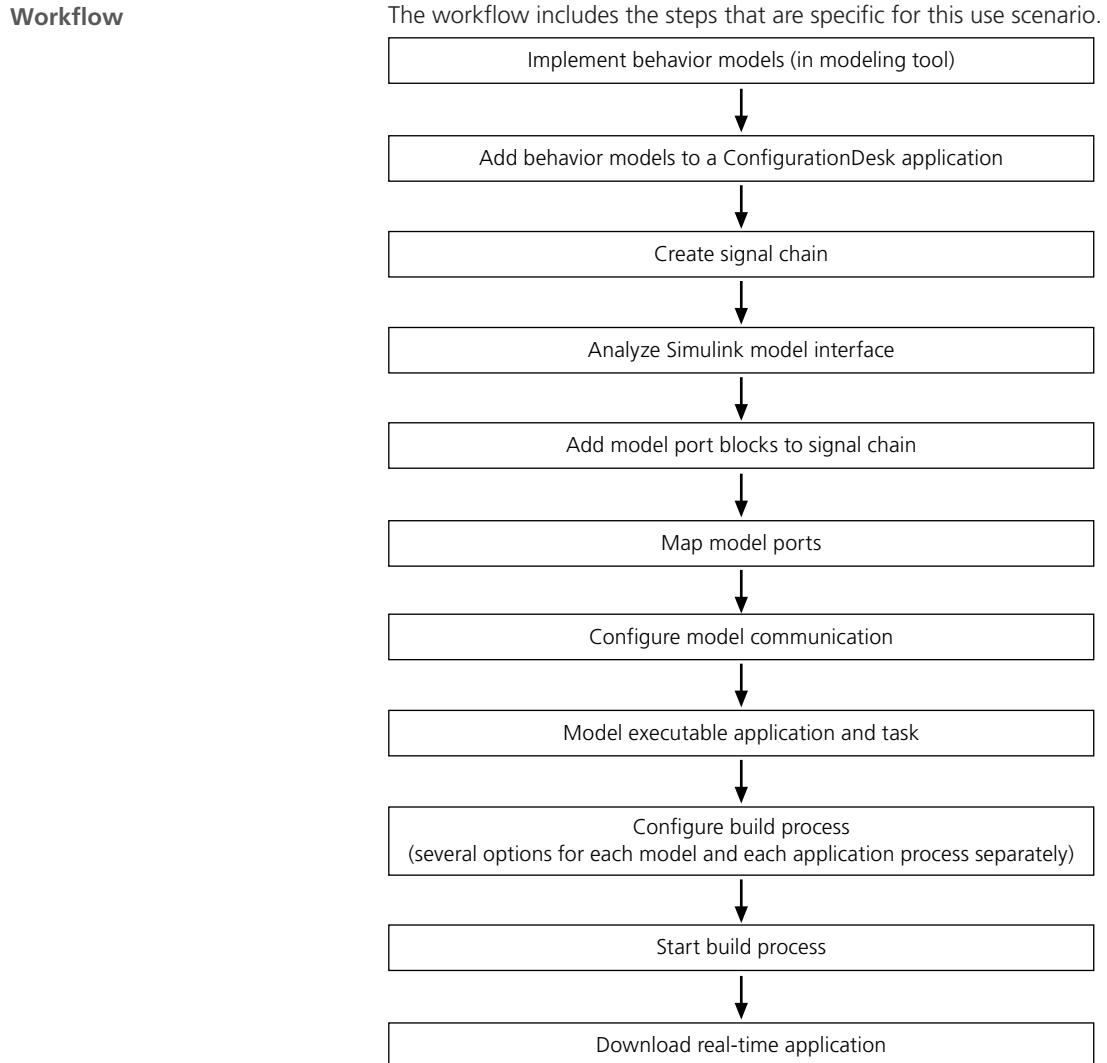
**Use scenario**

You can implement an application, where several single behavior models can be linked to ConfigurationDesk. With this you can build a multicore real-time application which can be downloaded to dSPACE real-time hardware to execute the models in parallel on single cores of the processor.

## Overview

The following illustration gives you an overview of the use scenario. Parts that are specific to this scenario are in bold letters.





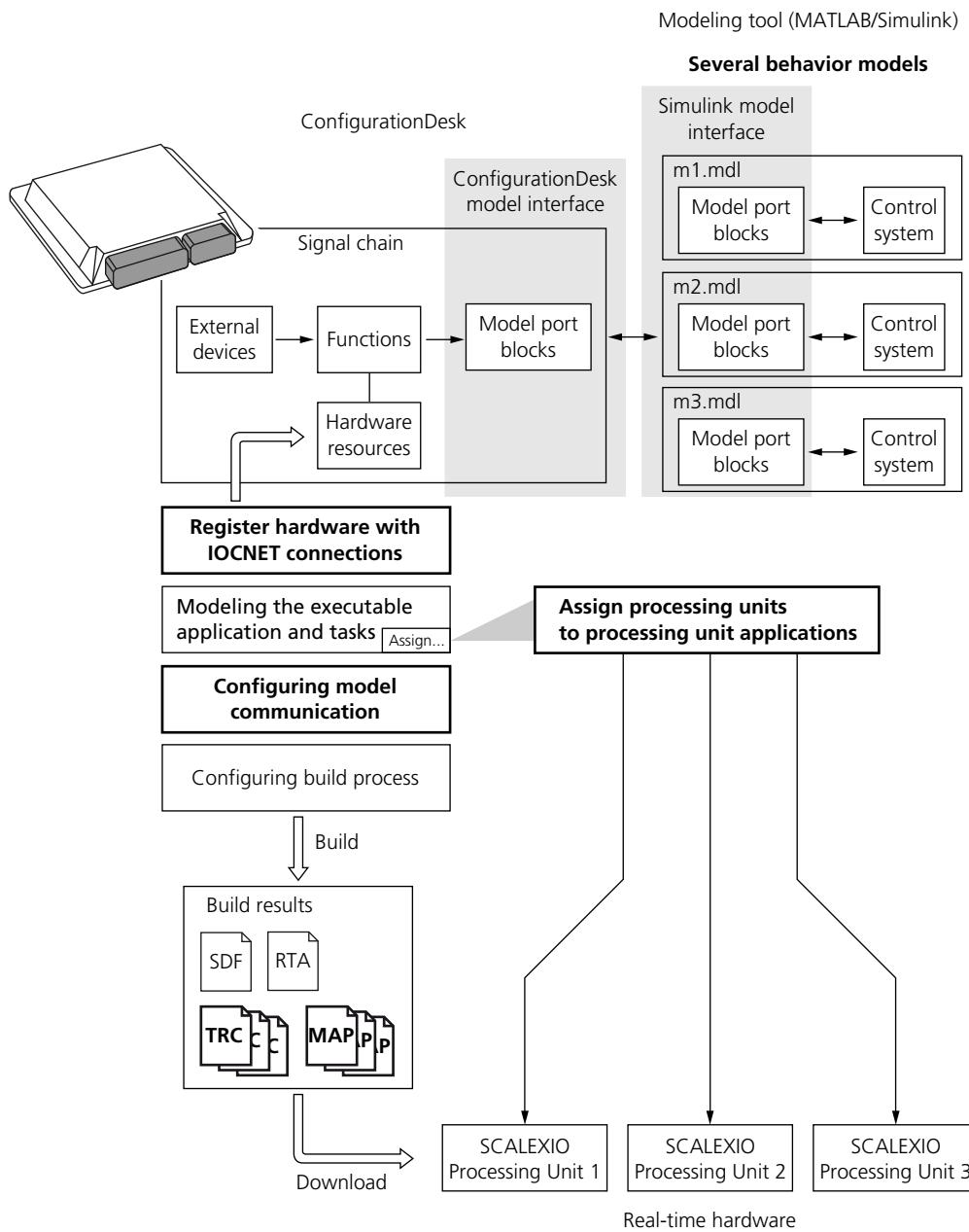
## Workflow for Creating a Multi-PU Application Using Multiple Behavior Models

### Use scenario

You can implement an application, where several single behavior models can be linked to ConfigurationDesk. With this you can build a multi-PU application which can be downloaded to dSPACE real-time hardware to execute the models in parallel on several SCALEXIO Processing Units.

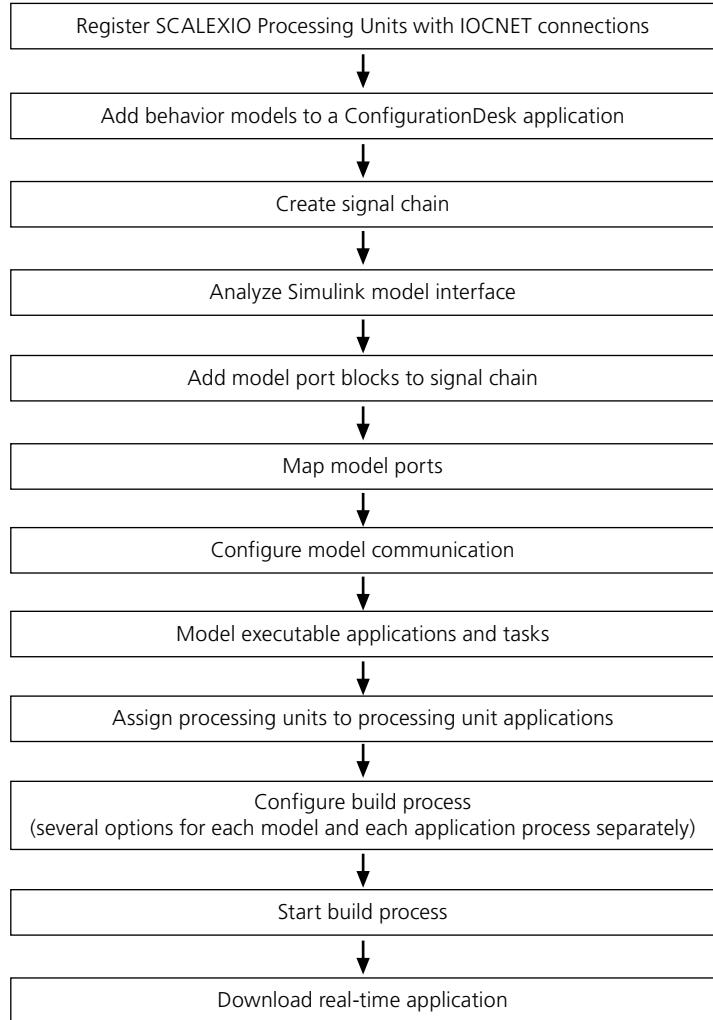
## Overview

The following illustration gives you an overview of the use scenario. Parts that are specific to this scenario are in bold letters.



## Workflow

The workflow includes the steps that are specific for this use scenario.



# User-Friendly Connection of ConfigurationDesk and Simulink Models

## Introduction

ConfigurationDesk provides the Model-Function Mapping Browser to simplify the work with Simulink behavior models.

## Where to go from here

### Information in this section

Working with the Model-Function Mapping Browser.....	553
Creating Signal Chains via the Model-Function Mapping Browser.....	555
Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model.....	558
Analyzing Simulink Behavior Models.....	561
Deleting Model Port Blocks in ConfigurationDesk and in Simulink Simultaneously.....	564
Switching Between Model Port Blocks in ConfigurationDesk and in Simulink.....	566

## Working with the Model-Function Mapping Browser

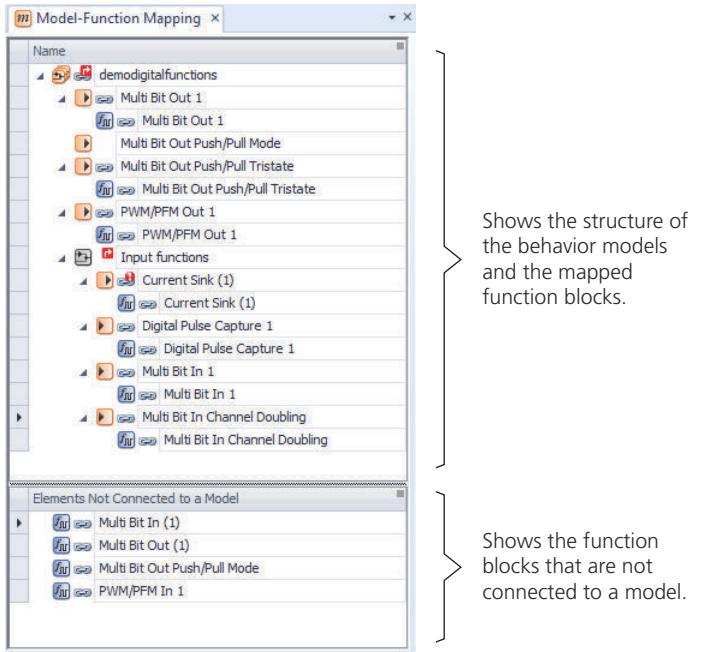
### Purpose of the Model-Function Mapping Browser

The Model-Function Mapping Browser in ConfigurationDesk and the commands for the remote access to ConfigurationDesk in Simulink behavior models allow a convenient connection of the Simulink behavior models to I/O functionality in ConfigurationDesk. For example, you can switch between a model port block in ConfigurationDesk and its representative in the Simulink behavior model.

For details on the commands for the remote access to ConfigurationDesk in Simulink behavior models, refer to [User-Friendly Connection Between Simulink and ConfigurationDesk \(Model Interface Package for Simulink - Modeling Guide\)](#).

### Overview of the Model-Function Mapping Browser

The Model-Function Mapping Browser is located in the Model-Function view set. It shows the structure of behavior models and the I/O functionality at a glance. The following illustration gives an overview of the available areas of the Model-Function Mapping Browser:



### Benefits of the Model-Function Mapping Browser

If you work with Simulink models as behavior models, the Model-Function Mapping Browser has the following benefits:

- It lets you quickly create and update signal chains for the work with Simulink behavior models.
- It provides commands, that let you update the ConfigurationDesk model interface and the Simulink model interface based on the configuration of the I/O functionality.
- It lets you switch between a model port block in ConfigurationDesk and its representative in the Simulink behavior model.
- It lets you delete signal chains in ConfigurationDesk and the related model port blocks in the Simulink behavior model in one step.

### Creating and updating signal chains for Simulink behavior models

The Model-Function Mapping Browser lets you create and update signal chains via drag & drop. Refer to [Creating Signal Chains via the Model-Function Mapping Browser](#) on page 555.

### Propagating changes in the ConfigurationDesk model interface to Simulink models

If you make any changes in the ConfigurationDesk model interface, you have to propagate these changes to the Simulink behavior model. Refer to [Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model](#) on page 558.

---

<b>Analyzing Simulink behavior models</b>	If there are any changes in the Simulink model interface, you have to make these changes known to ConfigurationDesk. Refer to <a href="#">Analyzing Simulink Behavior Models</a> on page 561.
<b>Switching between model port blocks in ConfigurationDesk and Simulink</b>	You can easily switch between model port blocks in ConfigurationDesk and in the Simulink behavior model. Refer to <a href="#">Switching Between Model Port Blocks in ConfigurationDesk and in Simulink</a> on page 566.
<b>Deleting model port blocks in ConfigurationDesk and Simulink simultaneously</b>	ConfigurationDesk provides methods for you to delete model port blocks in ConfigurationDesk and in Simulink simultaneously. Refer to <a href="#">Deleting Model Port Blocks in ConfigurationDesk and in Simulink Simultaneously</a> on page 564.

## Creating Signal Chains via the Model-Function Mapping Browser

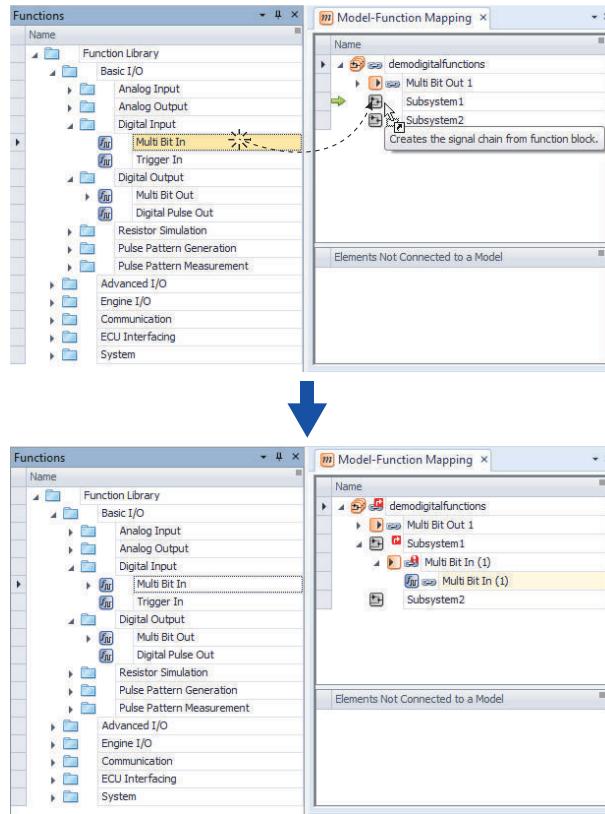
---

<b>Introduction</b>	The Model-Function Mapping Browser lets you create signal chains for the work with Simulink behavior models via drag & drop. The following methods are available: <ul style="list-style-type: none"> <li>▪ Adding I/O functionality and creating signal chains in one step.</li> <li>▪ Modeling asynchronous tasks and creating suitable Runnable Function blocks for the Simulink behavior model.</li> </ul>
<b>Adding I/O functionality and creating signal chains in one step</b>	To add I/O functionality and create new signal chains in your ConfigurationDesk application in one step, you have to drag a function block from the Functions Browser to a subsystem or the model root of a Simulink behavior model. If you do so, these are the results: <ul style="list-style-type: none"> <li>▪ ConfigurationDesk adds an instance of the selected function block to the ConfigurationDesk application.</li> <li>▪ ConfigurationDesk automatically creates suitable model port blocks for the instantiated function block.</li> <li>▪ The newly created model port block is automatically mapped to the function block.</li> </ul>

### Note

You can also map elements from the Hardware Resources Browser to subsystems or the model root of a Simulink behavior model. ConfigurationDesk then creates function blocks, maps them to the model interface, and assigns hardware resources to them in one step.

The following illustration shows an example and its result:



#### Note

- If you have already instantiated function blocks in your ConfigurationDesk application that are not mapped to model port blocks yet, you can drag them from the Elements Not Connected to a Model area of the Model-Function Mapping Browser to a subsystem or the model root of a Simulink behavior model. ConfigurationDesk then creates a suitable model port block and maps it to the function block automatically.
- Mapping function blocks and model port blocks via drag & drop is not supported for function blocks with configuration ports, such as CAN blocks or LIN blocks. These blocks must be mapped manually in the Signal Chain Browser.

#### Modeling asynchronous tasks and creating suitable Runnable Function blocks

Modeling an asynchronous task and creating a suitable Runnable Function block comprises the following steps:

1. Drag the desired function block from the Functions Browser to the Elements Not Connected to a Model area of the Model-Function Mapping Browser.
2. In the Properties Browser, enable event generation for the function block. Refer to [Modeling Asynchronous Tasks](#) on page 502.

3. Configure the function block. For example, specify its name, and add or remove ports according to your requirements.
4. Drag the function block from the Elements Not Connected to a Model area to the desired subsystem or the model root of the Simulink behavior model.

#### Note

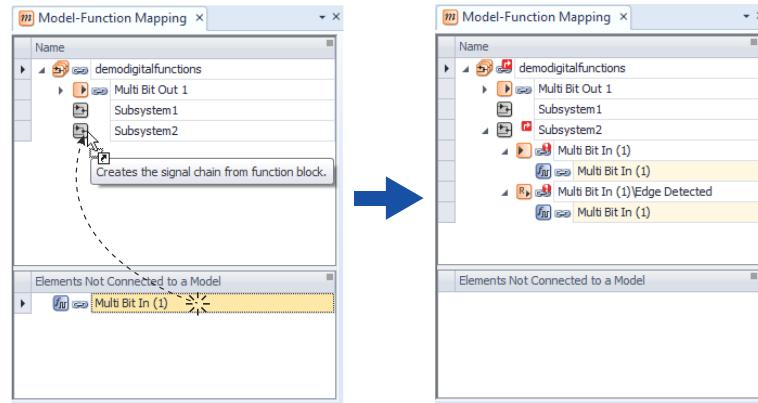
Alternatively, you can perform the following steps:

1. Drag a function block from the Functions Browser to a subsystem or the model root of a Simulink behavior model.
  2. Enable event generation for the function block.
  3. Select the Propagate to Simulink Model command.
- ConfigurationDesk then automatically creates a Runnable Function block and an asynchronous task.

If you perform the steps above, these are the results:

- ConfigurationDesk then creates a suitable Runnable Function block for the function block.
- The Runnable Function block is automatically mapped to the function block.

The following illustration shows an example and its result:



- ConfigurationDesk creates a new asynchronous task that has a suitable runnable function and the relevant I/O event assigned.

The following illustration shows the new asynchronous task in the Task Configuration table and the assigned runnable function and I/O event in the Properties Browser:

Name	Priority	DAQ R...	Real-Ti...	Period	Offset	Number...	Jitter a...
DemoDigitalFunctions							
Periodic Task 1	40	Periodic	<input checked="" type="checkbox"/>	0.001	0	0	Standard
<b>Task [Multi Bit In 1 Edge Detected]</b>	30					0	Standard

Properties

Name	Value
Task [Multi Bit In 1 Edge Detected]	
Edge Detected	
Identification	
Run-Time Behavior	
Run-Time Services	
Stack Size	
Edge Detected	
Identification	
Info	
Multi Bit In 1 Edge Detected	
Identification	

The default priority value of the new asynchronous task is 30. You can specify the priority in the Task Configuration table according to your needs. After a propagate operation, the related Hardware-Triggered Runnable Function block in the Simulink model has the specified priority.

### Next steps

After you created new model port blocks or modified existing model port blocks in ConfigurationDesk, you must propagate the changes to the Simulink behavior model to synchronize the model interfaces in ConfigurationDesk and in the Simulink behavior model. Refer to [Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model](#) on page 558.

## Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model

### Introduction

If you changed the ConfigurationDesk model interface, you must propagate the changes to the Simulink model to keep the model interfaces in ConfigurationDesk and in Simulink synchronized.

### Propagating changes to the Simulink behavior model

If you modified the function blocks in your ConfigurationDesk application, you can propagate the changes to the Simulink behavior model using the [Propagate to Simulink Model](#) command, for example, in the context menu of the Model-Function Mapping Browser. If you execute this command, these are the results:

Depending on the selected element, ConfigurationDesk performs the following actions:

Selected Element	Result
Function block	<p>ConfigurationDesk updates the model port blocks that are mapped to the function block and/or creates new model port blocks:</p> <ul style="list-style-type: none"> <li>▪ If model port blocks that are mapped to the function block have unmapped model ports, these model ports are deleted.</li> <li>▪ If function ports of the function block are not mapped yet, they are mapped to model ports. New model port blocks are created and model port blocks that are already mapped to the selected function block are updated with new model ports, depending on the following data: <ul style="list-style-type: none"> <li>▪ The setting specified for the <b>Model port block structure</b> property of the related function block type.</li> </ul> </li> </ul>

Selected Element	Result
	<ul style="list-style-type: none"> <li>▪ The functions of the function block to which the unmapped function ports belong.</li> <li>▪ The port type of the unmapped function ports.</li> <li>▪ If event ports of the function block are not mapped yet, new Runnable Function blocks are created and mapped to the event ports. The related runnable functions and I/O events are assigned to tasks. If required, new tasks are created for this purpose.</li> </ul> <p>Then, all the model port blocks that are mapped to the function block are propagated to the Simulink model.</p>
Model port block	<ul style="list-style-type: none"> <li>▪ If the model port block is mapped to a function block, the ConfigurationDesk model interface is updated according to the related function block (as described above). Then, all the model port blocks that are mapped to the function block are propagated to the Simulink model interface.</li> <li>▪ If the model port block is not mapped to a function block, only the model port block is propagated to the Simulink model interface.</li> </ul>
Subsystem	The model port blocks that belong to the subsystem are propagated to the Simulink model interface.
Model	All the model port blocks that belong to the model are propagated to the Simulink model interface.

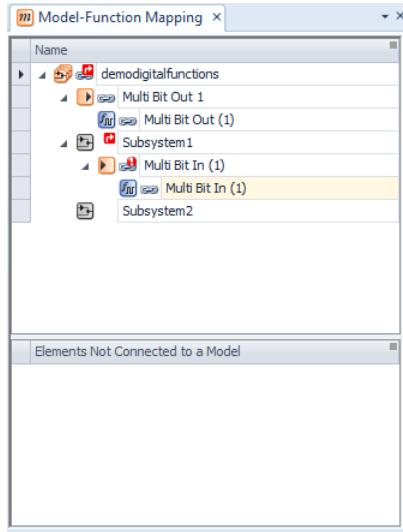
**Note**

The following applies if you use the **Propagate to Simulink Model** command:

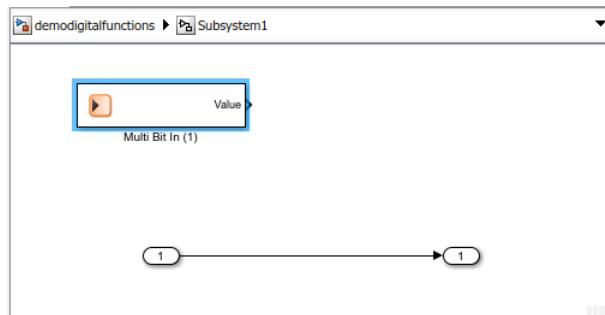
- You cannot undo the changes in the Simulink model. To prevent a propagate operation from accidentally changing parts of a Simulink model, you can protect specific subsystems by setting them to read-only in Simulink. When analyzed, these subsystems are marked with a lock symbol in ConfigurationDesk.
- Possible formatting of the model port blocks in Simulink, e.g., the color, is lost.
- This command cannot be used for Configuration Port blocks.

### Example of propagating changes to a Simulink behavior model

The following illustration shows an example of a ConfigurationDesk application containing unresolved model port blocks:



If you select the Propagate to Simulink Model command from the context menu of the Multi Bit In (1) function block, the Multi Bit In (1) model port block is propagated to Subsystem1 in the demodigitalfunctions behavior model:



### Propagating changes to the ConfigurationDesk model interface

If you modified the function blocks in your ConfigurationDesk application, you can propagate the modifications to the model port blocks in ConfigurationDesk using the Propagate to ConfigurationDesk Model Interface command, for example, in the context menu of the Model-Function Mapping Browser. If you execute this command, these are the results:

ConfigurationDesk updates the model port blocks that are mapped to the selected function block and/or creates new model port blocks:

- If model port blocks that are mapped to the function block have unmapped model ports, these model ports are deleted.

- If function ports of the function block are not mapped yet, they are mapped to model ports. New model port blocks are created and model port blocks that are already mapped to the selected function block are updated with new model ports, depending on the following data:
  - The setting specified for the Model port block structure property of the related function block type.
  - The functions of the function block to which the unmapped function ports belong.
  - The port type of the unmapped function ports.
- If event ports of the function block are not mapped yet, new Runnable Function blocks are created and mapped to the event ports.

---

**Display of unresolved model port blocks**

Model port blocks that are a part of the ConfigurationDesk model interface and do not have a representative model port block in the Simulink model interface, are unresolved. Unresolved model port blocks are marked with a  symbol. If there are any changes in the ConfigurationDesk model interface that must be propagated to the Simulink behavior model, this state is indicated in the Model-Function Mapping Browser by a  symbol.

## Analyzing Simulink Behavior Models

---

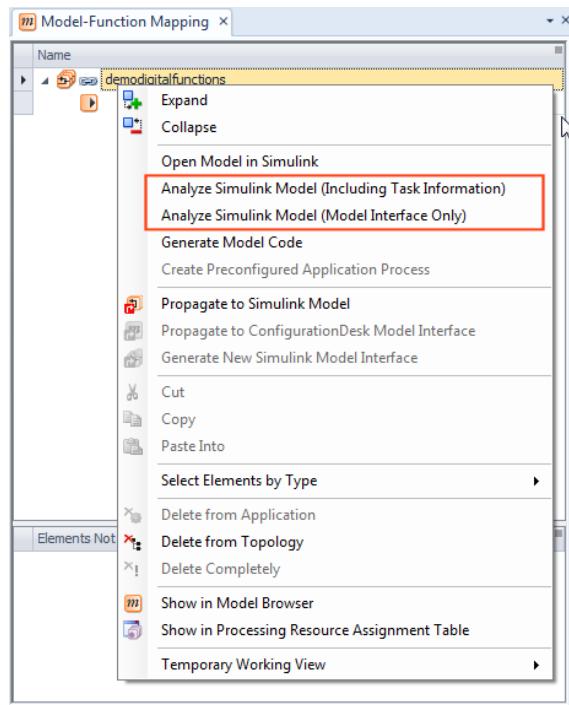
**Introduction**

If you have changed model port blocks in your Simulink [behavior model](#), you can make your changes available in ConfigurationDesk by performing a [model analysis](#).

ConfigurationDesk provides the following commands for analyzing a Simulink behavior model:

- [Analyze Simulink Model \(Model Interface Only\)](#)
- [Analyze Simulink Model \(Including Task Information\)](#)

You can access the commands, for example, in the context menu of the Simulink behavior model:



### Analyzing the model interface of a Simulink behavior model

Executing the Analyze Simulink Model (Model Interface Only) command provides the following results:

- If the Simulink behavior model is not already open, it is opened in MATLAB and its interface and its topology are analyzed. Afterwards the model topology in ConfigurationDesk is automatically updated by the new model port block data of the analyzed Simulink behavior model.
- The values of the model port block and model port properties specified in the Simulink behavior model are transferred to ConfigurationDesk. Existing values of the corresponding properties specified in ConfigurationDesk are overwritten.
- Depending on the modifications in your Simulink behavior model, the display of model port blocks in ConfigurationDesk can change. For details, refer to [Effects of Changing Model Interfaces of Simulink Behavior Models](#) on page 538.

#### Note

If the behavior model contains In Bus Element or Out Bus Element blocks, model initialization is required. Therefore, an analysis of the model interface via the Analyze Simulink Model (Model Interface Only) command is not possible in this case.

**Analyzing a Simulink model including task information**

Executing the **Analyze Simulink Model (Including Task Information)** command provides the following results:

- If the Simulink behavior model is not already open, it is opened in MATLAB, and the model interface and the elements provided by the behavior model are analyzed.
- By analogy with the **Analyze Simulink Model (Model Interface Only)** command, the values of the [model port block](#) and [model port](#) properties specified in the behavior model are transferred to the ConfigurationDesk. Existing values of the corresponding properties specified in ConfigurationDesk are overwritten.
- The task information provided by the Simulink behavior model is analyzed and displayed in the **Task Configuration** table.

**Tip**

To execute the **Analyze Simulink Model (Including Task Information)** command, the Simulink *Update Diagram* functionality must be executed for the Simulink behavior model to update information on tasks, events, and runnable functions. This can be very time-consuming, especially with large models. If you work with the signal chain and need to update only the model topology, it is sufficient and less time-consuming to execute the **Analyze Simulink Model (Model Interface Only)** command.

**Restrictions**

**Read/Write permissions** The Read/Write permissions option in Simulink's Function Block Parameters dialog must be set to a value other than "NoReadWrite" for all subsystems you want to include in the model analysis. Subsystems which are set to "NoReadWrite" are invisible to ConfigurationDesk's model analysis feature.

**Note**

If a model port block in ConfigurationDesk is mapped, but its parent subsystem is later set to "NoReadWrite" in the Simulink behavior model, ConfigurationDesk displays it as unresolved after the next model analysis. If the "NoReadWrite" setting is removed, the model port block is displayed as resolved, and the model mapping is restored after the next model analysis.

**Different results after model analysis and code generation for In Bus Element blocks**

In the following case, the results of the model analysis and the code generation differ:

- The signal of an In Bus Element block is specified by a Simulink.Bus object.
- The In Bus Element block is connected to a Bus Selector block that selects a subset of the bus signals.
- No other signal is used.

In the above case, a model port block with the complete bus hierarchy is displayed in ConfigurationDesk after a model analysis. However, after code generation, a model port block with only those bus signals that are selected by the Bus Selector block is displayed in ConfigurationDesk.

---

**Related topics**

**Basics**

Handling MATLAB via ConfigurationDesk..... 532

**References**

Analyze Simulink Model (Model Interface Only) (ConfigurationDesk User Interface Reference )

---

## Deleting Model Port Blocks in ConfigurationDesk and in Simulink Simultaneously

---

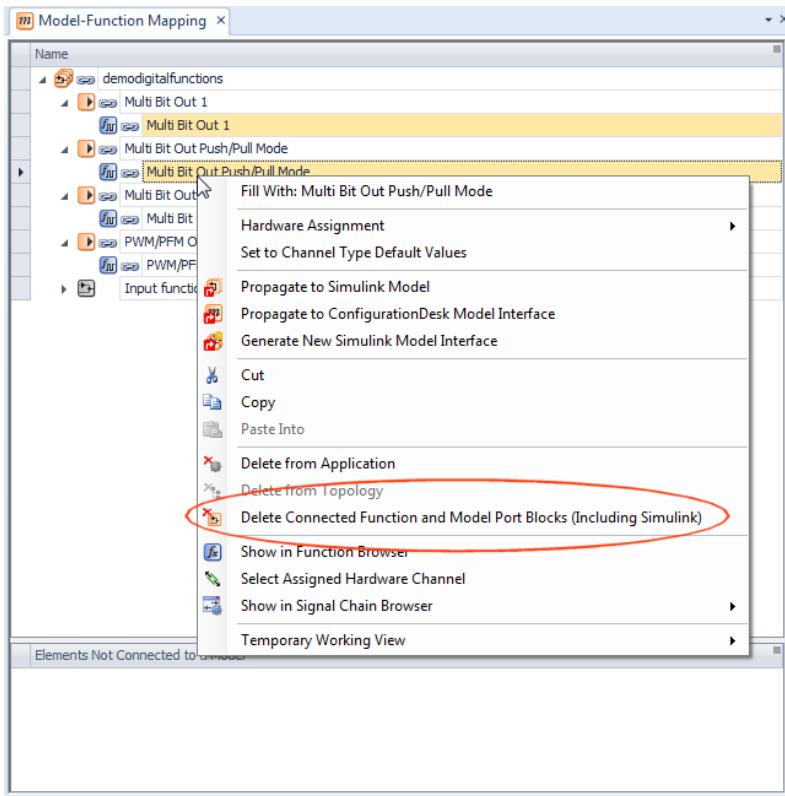
**Introduction**

ConfigurationDesk lets you delete model port blocks and their representatives in the Simulink behavior model simultaneously.

---

**Deleting blocks in ConfigurationDesk and in Simulink simultaneously**

ConfigurationDesk provides the Delete Connected Function and Model Port Blocks (Including Simulink) command. This command is accessible via the context menu of the selected model port blocks in the Model-Function Mapping Browser:



If you execute this command, the following blocks are deleted in one step:

- In ConfigurationDesk: The selected model port blocks.
- In ConfigurationDesk: The function blocks to which the selected model port blocks are mapped.

#### Note

If the function block is mapped to a Runnable Function block in ConfigurationDesk, the related asynchronous task including the runnable function and the I/O event is also deleted.

- In Simulink: The model port blocks that represent the model port blocks you selected in ConfigurationDesk.

Since the model port blocks are deleted in the Simulink model and in ConfigurationDesk, both model interfaces are still synchronized. In this case, it is not necessary to analyze the Simulink model in ConfigurationDesk, or to propagate the changes from ConfigurationDesk to the Simulink model.

**Note**

- A corresponding command is also available in the context menu of a selected model port block in the Simulink behavior model. Refer to [Delete Selection and Connected Blocks in ConfigurationDesk \(Model Interface Package for Simulink Reference\)](#).
- You cannot undo this command in Simulink.

## Switching Between Model Port Blocks in ConfigurationDesk and in Simulink

### Introduction

If you want to know which [model port block](#) in the active ConfigurationDesk application corresponds to which model port block in your Simulink [behavior model](#), ConfigurationDesk helps you to switch between the model port block in ConfigurationDesk and its representative in the Simulink model.

### Possible methods

ConfigurationDesk provides the following methods for switching between a model port block in ConfigurationDesk and its representative in the Simulink behavior model:

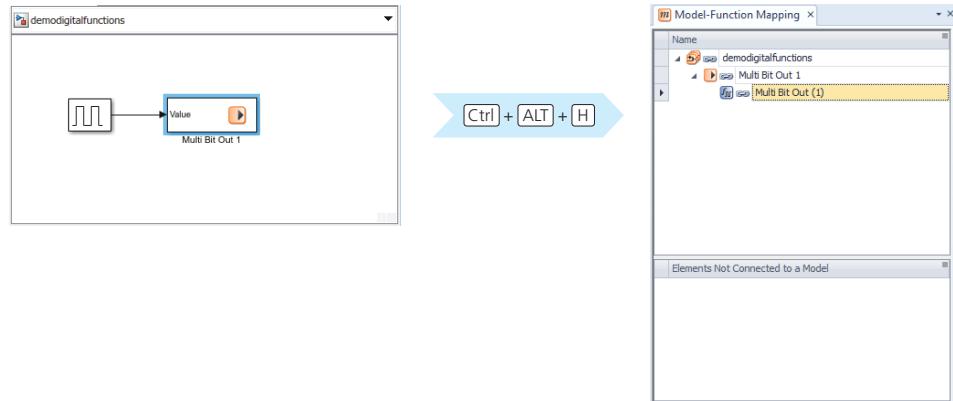
- Switching via keyboard shortcuts.
- Switching via a context menu command.

### Switching via keyboard shortcuts

Both ConfigurationDesk and the Model Interface Package for Simulink provide the same keyboard shortcuts that let you switch between a Simulink model port block and its related blocks in ConfigurationDesk.

Shortcut Key	Description
Ctrl+Alt+G	Lets you switch between a model port block and the related model port block in Simulink or in ConfigurationDesk. The following illustration shows an example:

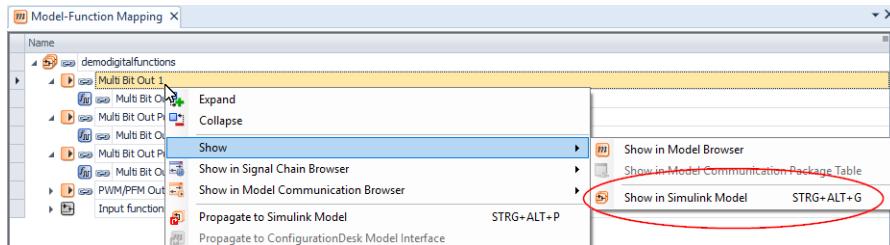
Shortcut Key	Description
Ctrl+Alt+H	Lets you switch between a Simulink model port block and the connected function block in ConfigurationDesk. The following illustration shows an example:



Switching between the blocks lets you directly view and change the configuration of a related Simulink model port block, for example. If necessary, you can analyze the Simulink behavior model to make the changes known in ConfigurationDesk.

#### Switching via a context menu command

The context menu of a selected model port block provides the Show - Show in Simulink Model command:



If you select this command, the relevant model port block is highlighted in the Simulink behavior model. If the relevant model port block is no longer available in the Simulink behavior model or has been moved to a different subsystem, an error message is displayed in MATLAB.

#### Note

If a model port block corresponds to In Bus Element and Out Bus Element blocks, the Show in Simulink Model command highlights all the In Bus Element or Out Bus Element blocks that belong to the related bus signal.

#### Notes on MATLAB

If you have connected several MATLAB installations to your dSPACE installation, you must select one of them as the preferred connection. For details, refer to

Introduction to Connecting a MATLAB Installation (Managing dSPACE Software Installations ).

---

**Problems with Simulink Editor**

Depending on your Windows setting, it can happen that the Simulink Editor does not appear in the foreground, but in the background. Instead, the Windows taskbar displays a flashing message referring to the Simulink Editor. You can avoid this behavior by using two monitors, one for ConfigurationDesk, one for MATLAB.

---

**Related topics**

**Basics**

[Handling MATLAB via ConfigurationDesk.....](#) 532

**References**

[Show in Simulink Model \(ConfigurationDesk User Interface Reference !\[\]\(63966621b5bccb8a3698f84b90070ae8\_img.jpg\)\)](#)

# Creating Multimodel Real-Time Applications With Models Separated From One Overall Model

## Introduction

You can implement multimodel real-time applications with different Simulink behavior models that are separated from one overall model. The separated models are used to build multicore or multi-PU real-time applications.

## Where to go from here

### Information in this section

Basics of Creating Multicore or Multi-PU Applications Using One Overall Model.....	569
Workflow for Creating a Multicore Real-Time Application Using One Overall Behavior Model.....	570
Workflow for Creating a Multi-PU Application Using One Overall Behavior Model.....	573
Working With MCD Files.....	576

## Basics of Creating Multicore or Multi-PU Applications Using One Overall Model

### Use scenario

You can implement an application with different Simulink behavior models that are linked to ConfigurationDesk but that are also part of an overall model. dSPACE provides a block to separate models from an overall model in MATLAB/Simulink. The overall model remains unchanged.

In ConfigurationDesk, the separated models are used to build a multicore real-time application or a multi-processing-unit (multi-PU) real-time application, which then can be downloaded to dSPACE real-time hardware to execute the models in parallel on single cores of the processor or on several SCALEXIO Processing Units. Refer to the following topics:

- [Workflow for Creating a Multicore Real-Time Application Using One Overall Behavior Model](#) on page 570.
- [Workflow for Creating a Multi-PU Application Using One Overall Behavior Model](#) on page 573.

### Using MCD files

During model separation in MATLAB/Simulink, a model communication description file (MCD file) is generated. The MCD file contains a list of the

separated models and a list of the Data Import blocks and Data Outport blocks created in the models including their model communication. For more information, refer to [Working With MCD Files](#) on page 576.

---

**Related topics**

**Basics**

Terms and Definitions for Building Executable Applications.....	472
Workflow for Creating a Multicore Real-Time Application Using Multiple Behavior Models.....	548
Workflow for Creating a Multi-PU Application Using Multiple Behavior Models.....	550

---

## Workflow for Creating a Multicore Real-Time Application Using One Overall Behavior Model

---

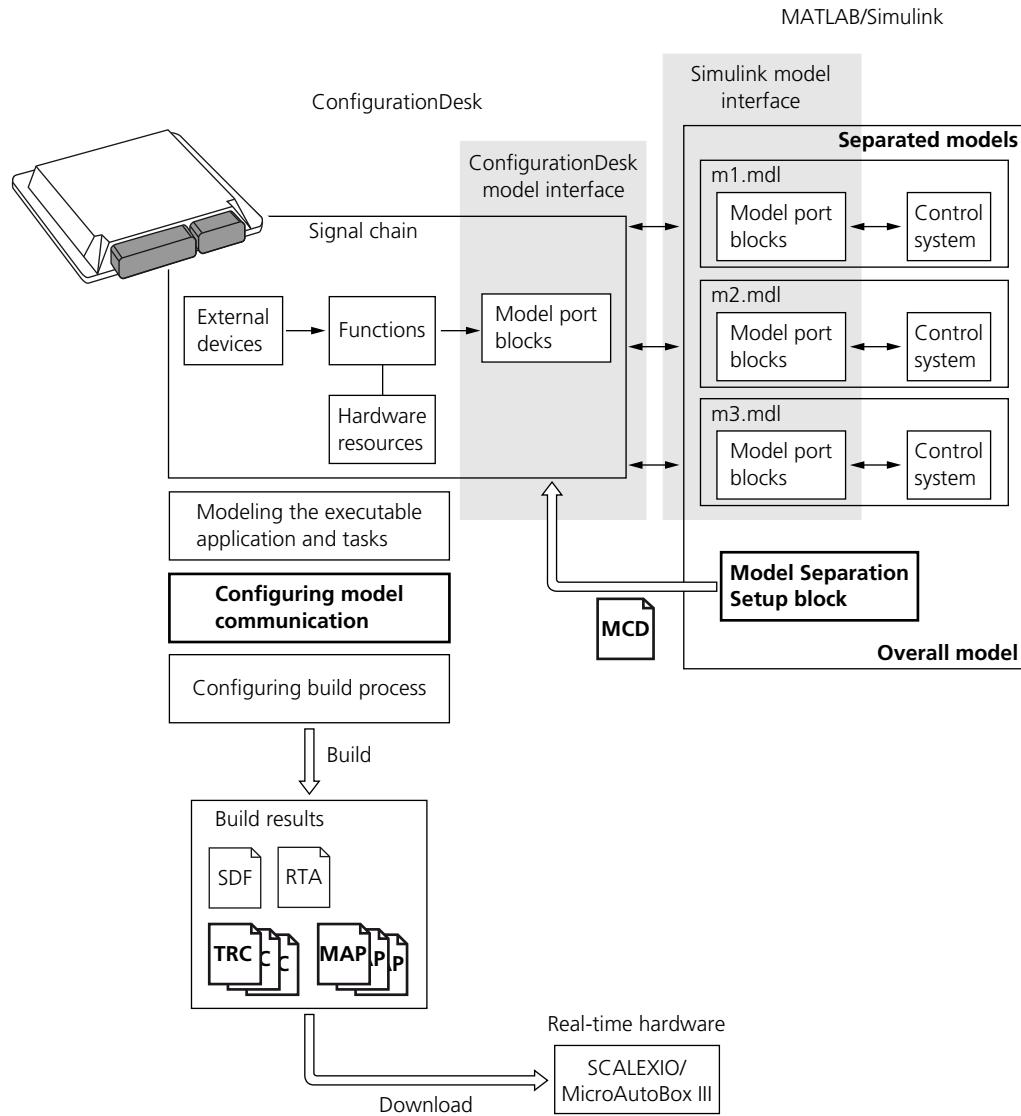
**Use scenario**

You can implement an application with different behavior models that are linked to ConfigurationDesk but that are also part of an overall model. You can use this overall model for offline simulation in the modeling tool.

dSPACE provides a block to separate models from an overall model in MATLAB/Simulink. The overall model remains unchanged. The separated models are used to build a multicore real-time application, which then can be downloaded to dSPACE real-time hardware to execute the models in parallel on single cores of the processor.

**Overview**

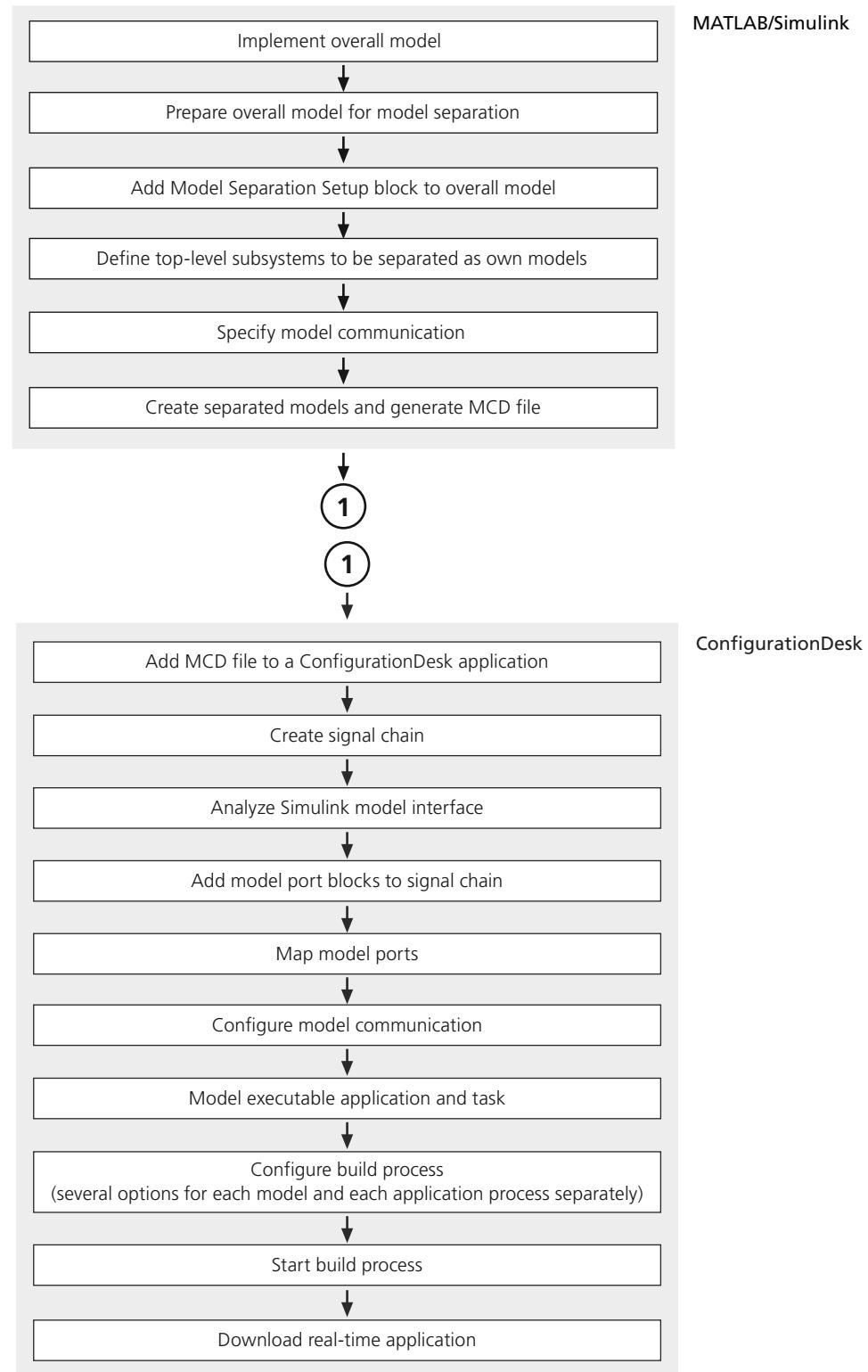
The following illustration gives you an overview of the use scenario. Parts that are specific to this scenario are in bold letters.



To separate models from an overall model in MATLAB/Simulink, you can use the **Model Separation Setup block**. This also generates an MCD file, which contains information on the separated models and their interconnections in the overall model.

**Workflow**

The workflow includes the steps that are specific for this use scenario.



## Workflow for Creating a Multi-PU Application Using One Overall Behavior Model

---

### Use scenario

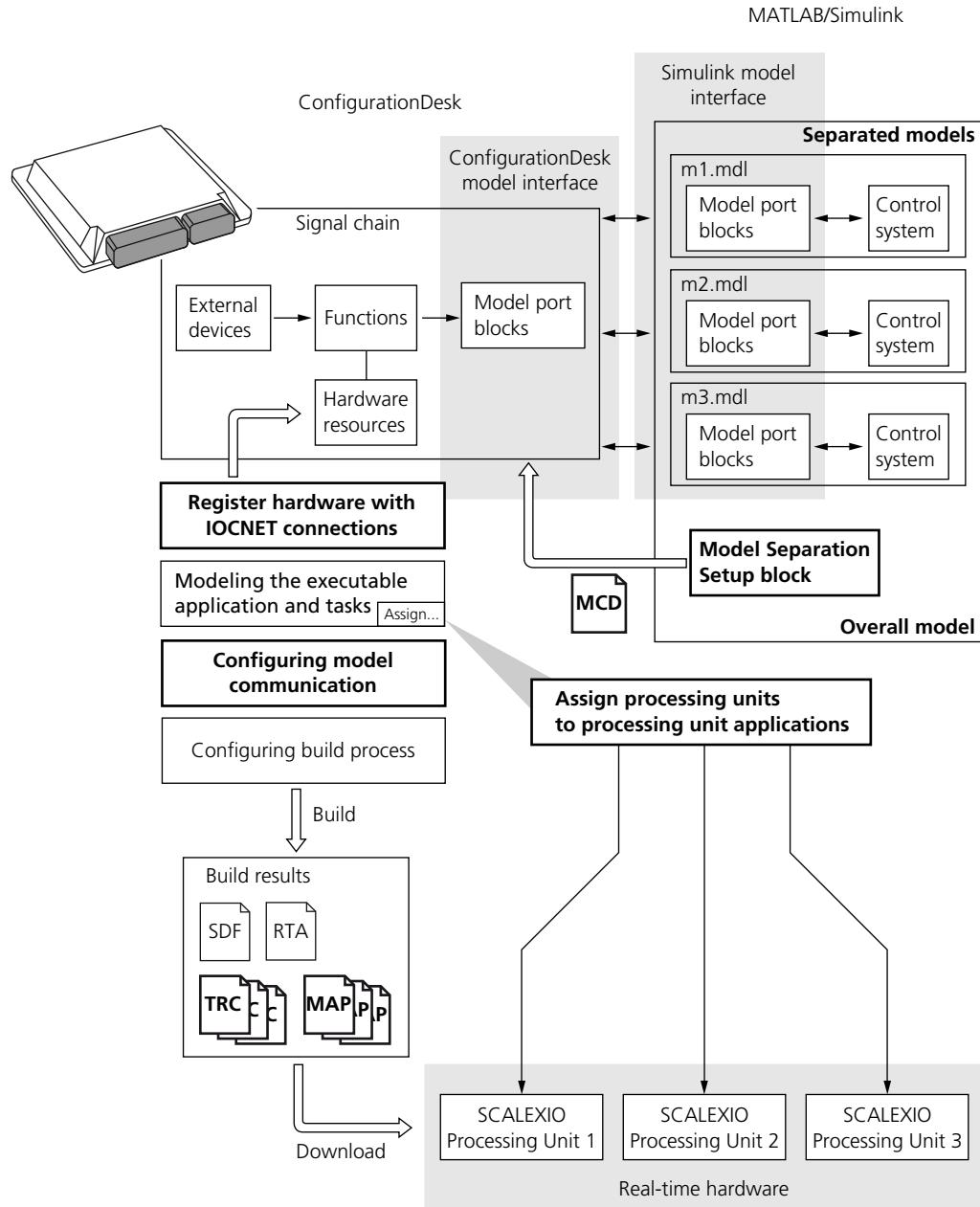
You can implement an application with different behavior models that are linked to ConfigurationDesk but that are also part of an overall model. You can use this overall model for offline simulation in the modeling tool.

dSPACE provides a block to separate models from an overall model in MATLAB/Simulink. The overall model remains unchanged. The separated models are used to build a multi-PU application, which then can be downloaded to dSPACE real-time hardware to execute the models in parallel on several SCALEXIO Processing Units.

---

### Overview

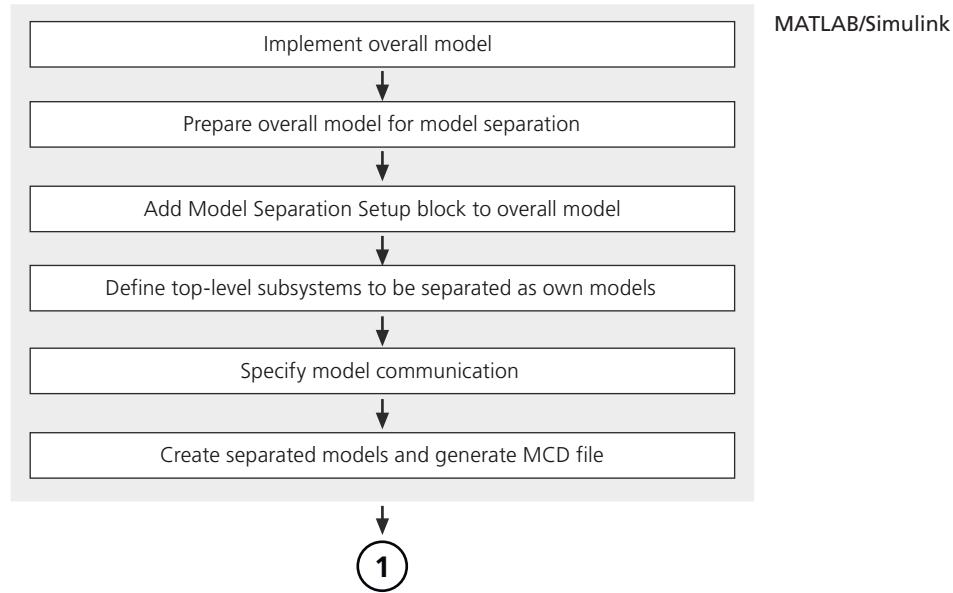
The following illustration gives you an overview of the use scenario. Parts that are specific to this scenario are in bold letters.

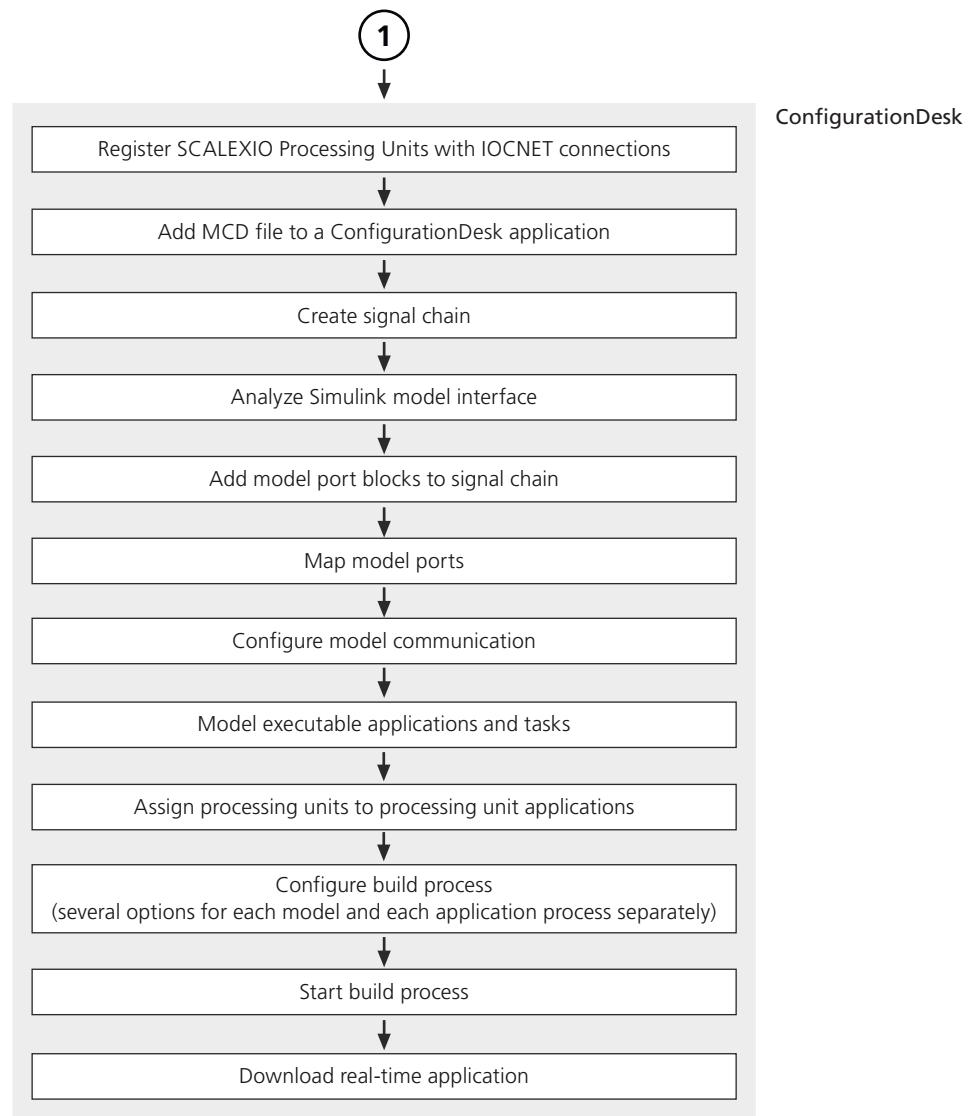


To separate models from an overall model in MATLAB/Simulink, you can use the **Model Separation Setup block**. This also generates an MCD file, which contains information on the separated models and their interconnections in the overall model.

**Workflow**

The workflow includes the steps that are specific for this use scenario.





## Working With MCD Files

### Description of MCD files

An MCD file is a model communication description file that is used to implement a multimodel application. It lets you add several Simulink behavior models that were separated from an overall Simulink model to the [model topology](#). Refer to [Separating Models from an Overall Model to Build Multimodel Applications \(Model Interface Package for Simulink - Modeling Guide\)](#).

## Handling the MCD file

The MCD file is used only if your [Simulink model interface](#) is based on models that are separated from an overall behavior model.

To import the information to ConfigurationDesk, you must add the MCD file to your active ConfigurationDesk application via the Project Manager in the same way as you add an MDL, an SLX, or an MTFX file. For instructions, refer to [How to Import a Model Topology](#) on page 112.

If you add the MCD file to the model topology without performing an analysis of the Simulink model interface, this is the result:

- The Model Browser displays the separated models and all the model ports that are involved in model communication. The model port blocks are displayed as unresolved blocks.
- All the model ports that are involved in model communication are added to the signal chain in the global working view. The interconnections between the behavior models are displayed as mapping lines.
- The Model Communication Package table lists all the Data Import blocks that are involved in model communication.

### Note

If you perform a model interface analysis immediately after adding the MCD files, the model topology is updated with the complete model interface data of your behavior models.

ConfigurationDesk stores the MCD file as a reference in the model topology of the active ConfigurationDesk application. To handle this reference, ConfigurationDesk provides the reload and clear functions. You can access them via Model Topology context menu in the Project Manager.

**Reload MCD file** If you reload the MCD file, the model topology and the Model Communication Package table are updated immediately with the information stored in the MCD file. As a result, interconnections between models can be deleted. Connections which have been specified exclusively in ConfigurationDesk are definitively deleted.

Keep in mind that the MCD file only contains information on separated models and their interconnections. To provide other changes in the behavior model related to the model interface, you have to execute a model analysis again. Changes take effect only when a new MCD file is generated with the Model Separation Setup block in MATLAB/Simulink.

**Clear MCD file** You can delete the reference to the (MCD file) from the model topology. This may be useful in the following cases:

- You want to use the behavior models in your ConfigurationDesk application regardless of any changes in the overall model.
- You decide to specify model communication exclusively in ConfigurationDesk. If you want to reload the MCD file after clearing the reference, you have to re-add the MCD file to the application via the Project Manager. For instructions, refer to [How to Import a Model Topology](#) on page 112.

# Special Use Cases of Working With Simulink Behavior Models

Where to go from here	Information in this section
	<p>Using Model Port Blocks to Handle Simulink Behavior Model Variants..... 578</p> <p>Creating Model Port Blocks With Structured Data Ports for Bus Signals..... 579</p> <p>Simplified Preparation of Model Interfaces for Model Communication..... 582</p> <p>How to Create Inverse Model Port Blocks for Model Communication..... 583</p>

## Using Model Port Blocks to Handle Simulink Behavior Model Variants

**Introduction** ConfigurationDesk lets you use [model port blocks](#) with duplicate IDs in a [multimodel application](#). This enables you to work with different variants of Simulink behavior models.

**Using model port blocks with duplicate identities** You can add Simulink behavior models containing model port blocks with duplicate identities (i.e., Simulink behavior models with identical interfaces) to your multimodel application. You can then drag the corresponding model port blocks in ConfigurationDesk to your working view and map their ports to function ports or other model ports as usual. To model your executable application, you can assign one of the behavior models with identical interfaces to an application process without causing a conflict.

### Note

If you assign two or more Simulink behavior models containing model port blocks with duplicate identities to application processes, the model port blocks are marked with a  icon in the Model Browser and in the Model-Function Mapping Browser. Additionally, a conflict is shown in the Conflicts Viewer (refer to [Model Conflicts \(ConfigurationDesk Conflicts\)](#)). Model port blocks that are affected by this conflict are not used for code generation. During the build process, a warning is displayed.

**Switching between Simulink behavior model variants**

Suppose Model\_A and Model\_B are Simulink behavior models containing model port blocks with duplicate identities, and you want to replace Model\_A with Model\_B in your executable application. To do so, you must delete the assignment of Model\_A to the application process and then assign Model\_B to that application process (refer to [Modeling Executable Applications and Tasks](#) on page 471). Then, all the model port blocks of Model\_A that are used in the signal chain are automatically resolved to the corresponding model port blocks of Model\_B.

**Related topics****Basics**

[Characteristics of Model Port Blocks](#)..... 426

**References**

[Add Model \(ConfigurationDesk User Interface Reference\)](#)  
[Delete Assignment \(Table Items\) \(ConfigurationDesk User Interface Reference\)](#)

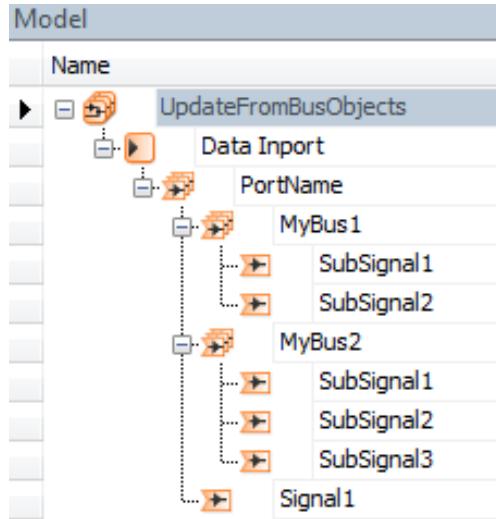
## Creating Model Port Blocks With Structured Data Ports for Bus Signals

**Introduction**

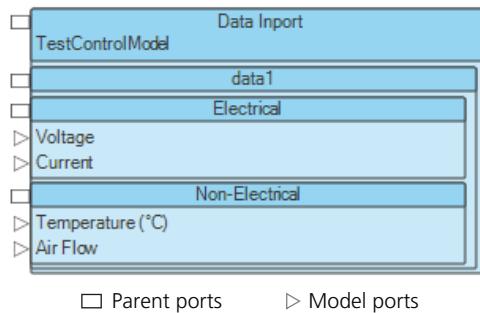
ConfigurationDesk lets you work with Simulink bus signals. You can import model port blocks with structured data ports from a Simulink model to your ConfigurationDesk application, or create model port blocks with structured data ports in ConfigurationDesk to transfer bus signals (for example, from a CAN function block) to a Simulink model.

**Importing data port blocks with structured data ports**

You can import model port blocks with structured data ports from a Simulink model to ConfigurationDesk by performing a model analysis. After a model analysis, a model port block with structured data imports or exports is displayed in the **Model Browser**, marked with a  symbol. The structure of the model ports is shown as a hierarchical tree.



You can drag the model port blocks with structured data ports from the Model Browser to ConfigurationDesk's working view just like other model port blocks. In the working view, the structured ports also display the structure of the related bus signals, and they provide parent ports and model ports.



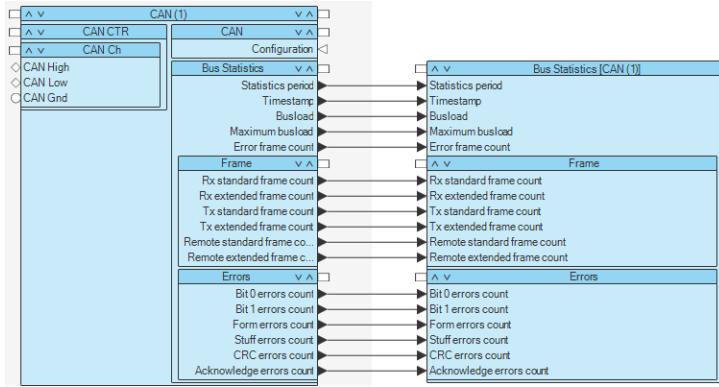
**Mapping structured model ports** You can map sets of structured data ports to suitable sets of function ports or other model ports easily by mapping the related parent ports. If you do, the child ports corresponding to the parent ports are mapped automatically according to specific rules. For details, refer to [Mapping Algorithm for Model Port Blocks With Structured Data Ports](#) on page 446.

#### Note

Mapping model port block with structured data ports to model ports of other behavior models might reduce the run-time performance. For optimum run-time performance, it is recommended to observe specific mapping rules. Refer to [Model Communication Recommendations for Model Port Blocks with Structured Data Ports](#) on page 464.

### Creating model port blocks with structured data ports in ConfigurationDesk

To transfer bus signals (for example, from a CAN function block) to a Simulink model, you need suitable model port blocks. You can use the Extend Signal Chain - Create Suitable Model Port Block command (refer to [Basics on Adding Model Port Blocks to the Signal Chain](#) on page 432) to generate model port blocks with structured data ports. The structured model ports match the structure of the function block. ConfigurationDesk automatically maps the function ports to the generated model ports.



You can now generate a Simulink model interface and add it to your Simulink model. For details, refer to [Adding the Generated Model Interface to Your Behavior Model via an Interface Model \(Model Interface Package for Simulink - Modeling Guide\)](#).

#### Note

For CAN function blocks, you must enable the Bus statistics checkbox in the Properties Browser to generate model port blocks with structured model ports from function blocks.

**Naming conventions for generated model port blocks and structured model ports** Model port blocks and their structured data ports generated from a function block are named according to the following conventions:

- The name of the model port block is derived from the name of the function and the path of the function within the function block structure. If the function block contains just one function, the model port block is named according to the function block name.
- An `_In/_Out` suffix is added to the original function (or function block) name if two model port blocks are created for a single function block.

Below are some examples for the syntax of the names of generated model port blocks:

Simple case (block contains only one function):

```
Model port block name = <Block instance name> or  
Model port block name = <Block instance name>_In/Out
```

Complex case (nested structure):

```
Model port block name = <Function name> [<Block instance  
name>\<Function group 1 name>\<Function group 2 name>] or
```

```
Model port block name = <Function name>_In/Out [<Block
instance name>\<Function group 1 name>\<Function group 2
name>]
```

- Generated structured data ports are named according to the corresponding function port groups.

#### Restrictions for models with In Bus Element/Out Bus Element blocks

If a Simulink behavior model contains In Bus Element or Out Bus Element blocks, the following restrictions apply:

- The analysis of a model that contains root-level In Bus Element and Out Bus Element blocks requires an initialization of the model. Therefore, analyzing the model interface via the [Analyze Simulink Model \(Model Interface Only\)](#) command is not possible.
- In the following case, the results of the model analysis and the code generation differ:
  - The signal of an In Bus Element block is specified by a Simulink.Bus object.
  - The In Bus Element block is connected to a Bus Selector block that selects a subset of the bus signals.
  - No other signal is used.

In the above case, a model port block with the complete bus hierarchy is displayed in ConfigurationDesk after a model analysis. However, after code generation, a model port block with only those bus signals that are selected by the Bus Selector block is displayed in ConfigurationDesk.

#### Related topics

#### References

[Analyze Simulink Model \(Including Task Information\) \(ConfigurationDesk User Interface Reference\)](#)

[Analyze Simulink Model \(Model Interface Only\) \(ConfigurationDesk User Interface Reference\)](#)

## Simplified Preparation of Model Interfaces for Model Communication

#### Introduction

To set up [model communication](#) between [model implementations](#) in ConfigurationDesk, you need suitable [model port blocks](#) in the [model topology](#). To simplify your work, ConfigurationDesk lets you create model port blocks that have the same configuration but the inverse data direction as model port blocks that already exist in the model topology.

**Note**

The Model Interface Package for Simulink also provides this feature, so you can create inverse model port blocks in ConfigurationDesk as well as in your Simulink model.

**Characteristics of inverse model port blocks**

The inverse model port blocks have the following characteristics:

- They have the same names as the original model port blocks.
- They have the same port groups and port names as the original model port blocks.
- They have the inverse data direction of the original model port blocks.
- If the original model port blocks are unmapped, the inverse model port blocks are automatically mapped to the original model port blocks. Otherwise, the inverse model port blocks are unmapped, so you have to map them manually.
- They are unresolved.

You can work with the inverse model port blocks as usual. This means that you can use them to set up model communication, or to generate a [Simulink model interface](#). For instructions, refer to [How to Create Inverse Model Port Blocks for Model Communication](#) on page 583.

**Related topics****HowTos**

[How to Create Inverse Model Port Blocks for Model Communication](#)..... 583

**References**

[Create Inverse Block \(ConfigurationDesk User Interface Reference\)](#)

## How to Create Inverse Model Port Blocks for Model Communication

**Objective**

Creating [inverse model port blocks](#) lets you easily create a [model interface](#) for setting up [model communication](#) in ConfigurationDesk.

**Preconditions**

The following preconditions must be fulfilled:

- You must have opened a ConfigurationDesk application with at least one model implementation in its model topology.
- The model implementation must contain model port blocks that are visible in a working view.

---

<b>Method</b>	<b>To create inverse model port blocks for model communication</b> <ol style="list-style-type: none"><li>1 In the signal chain, select the model port blocks you want to create inverse model port blocks for.</li><li>2 From the context menu, select <b>Create Inverse Block</b>.</li></ol>
<b>Result</b>	You created an inverse model port block for each selected model port block. The inverse model port blocks have the same names, the same port group names and the same port names as the original model port blocks, but they have the inverse data direction.  If the original model port blocks are unmapped, ConfigurationDesk maps them automatically to the preconfigured inverse model port blocks. Otherwise, the preconfigured inverse model port blocks are unmapped, so you have to map them manually.
<b>Next steps</b>	To use the preconfigured inverse model port blocks for model communication, you have to perform the following steps: <ul style="list-style-type: none"><li>▪ Complete model communication, if necessary. Refer to <a href="#">Basics on Model Communication</a> on page 448.</li><li>▪ Make the preconfigured inverse model port blocks available in the desired Simulink behavior model. Refer to <a href="#">How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model</a> on page 543.</li></ul>
<b>Related topics</b>	<p>Basics</p> <p><a href="#">Simplified Preparation of Model Interfaces for Model Communication</a>..... 582</p> <p>References</p> <p><a href="#">Create Inverse Block (ConfigurationDesk User Interface Reference)</a></p>

# Working with Container Files as Behavior Models

---

<b>Introduction</b>	Code container files contain the code of behavior models. ConfigurationDesk lets you work with different types of container files.										
<b>Where to go from here</b>	<b>Information in this section</b>										
	<table><tr><td>Adding Container Files to a ConfigurationDesk Application.....</td><td>586</td></tr><tr><td>Working with Simulink Implementation Containers.....</td><td>588</td></tr><tr><td>Working with Bus Simulation Containers.....</td><td>596</td></tr><tr><td>Working with Functional Mock-up Units.....</td><td>607</td></tr><tr><td>Working with V-ECU Implementations.....</td><td>625</td></tr></table>	Adding Container Files to a ConfigurationDesk Application.....	586	Working with Simulink Implementation Containers.....	588	Working with Bus Simulation Containers.....	596	Working with Functional Mock-up Units.....	607	Working with V-ECU Implementations.....	625
Adding Container Files to a ConfigurationDesk Application.....	586										
Working with Simulink Implementation Containers.....	588										
Working with Bus Simulation Containers.....	596										
Working with Functional Mock-up Units.....	607										
Working with V-ECU Implementations.....	625										

# Adding Container Files to a ConfigurationDesk Application

<b>Introduction</b>	ConfigurationDesk lets you add different container file types to the ConfigurationDesk application.				
<b>Where to go from here</b>	<b>Information in this section</b>				
	<table><tr><td>Introduction to Working With Container Files as Behavior Models .....</td><td>586</td></tr><tr><td>How to Add Model Implementation Containers to a ConfigurationDesk Application.....</td><td>586</td></tr></table>	Introduction to Working With Container Files as Behavior Models .....	586	How to Add Model Implementation Containers to a ConfigurationDesk Application.....	586
Introduction to Working With Container Files as Behavior Models .....	586				
How to Add Model Implementation Containers to a ConfigurationDesk Application.....	586				

## Introduction to Working With Container Files as Behavior Models

<b>Overview of supported container file types</b>	ConfigurationDesk lets you work with different container file types as behavior models. The following container file types are supported: <ul style="list-style-type: none"><li>▪ Simulink implementation container files (SIC files). Refer to <a href="#">Working with Simulink Implementation Containers</a> on page 588.</li><li>▪ Bus simulation container files (BSC files). Refer to <a href="#">Working with Bus Simulation Containers</a> on page 596.</li><li>▪ Functional Mock-up Units (FMU files). Refer to <a href="#">Working with Functional Mock-up Units</a> on page 607.</li><li>▪ V-ECU implementation container files (VECU or CTLGZ files). Refer to <a href="#">Working with V-ECU Implementations</a> on page 625.</li></ul>
---	--

<b>Adding container files to a ConfigurationDesk application</b>	You can add container files to your ConfigurationDesk application via the Add Model command. Refer to <a href="#">How to Add Model Implementation Containers to a ConfigurationDesk Application</a> on page 586.
--	--

## How to Add Model Implementation Containers to a ConfigurationDesk Application

<b>Objective</b>	To be able to use container files as behavior models, you must add them to your ConfigurationDesk application.
------------------	--

---

**Precondition**

You must have an open ConfigurationDesk project with an active ConfigurationDesk application.

---

**Method****To add container files to a ConfigurationDesk application**

- 1 Right-click in the Model Browser and select Add Model from the context menu.  
The Add Model dialog opens.
  - 2 In the Add Model dialog, click Add a model and select Add model topology from file.  
A standard Open dialog opens.
  - 3 Select the container file type and the container file you want to add to the ConfigurationDesk application and click Open.  
The selected file is added to the Selected model(s) list.
  - 4 Repeat steps 2 - 3 for all the files you want to add to the ConfigurationDesk application.
  - 5 Click OK.
- 

**Result**

The selected files are added to the active ConfigurationDesk application. ConfigurationDesk creates a preconfigured application process in the Executable Application table for all the selected files.

# Working with Simulink Implementation Containers

<b>Introduction</b>	ConfigurationDesk lets you add Simulink implementation containers to a ConfigurationDesk application. So you can integrate them into your executable application and execute them on your real-time hardware. Simulink implementation containers are code containers based on a Simulink behavior model.										
<b>Where to go from here</b>	<b>Information in this section</b>										
	<table border="0"> <tr> <td>Basics on Simulink Implementation Containers.....</td> <td>588</td> </tr> <tr> <td>Workflow for Integrating Simulink Implementation Containers in Executable Applications.....</td> <td>590</td> </tr> <tr> <td>Adding Simulink Implementation Containers to a ConfigurationDesk Application.....</td> <td>591</td> </tr> <tr> <td>Creating Precompiled SIC Files.....</td> <td>592</td> </tr> <tr> <td>Updating Simulink Implementation Containers in ConfigurationDesk.....</td> <td>595</td> </tr> </table>	Basics on Simulink Implementation Containers.....	588	Workflow for Integrating Simulink Implementation Containers in Executable Applications.....	590	Adding Simulink Implementation Containers to a ConfigurationDesk Application.....	591	Creating Precompiled SIC Files.....	592	Updating Simulink Implementation Containers in ConfigurationDesk.....	595
Basics on Simulink Implementation Containers.....	588										
Workflow for Integrating Simulink Implementation Containers in Executable Applications.....	590										
Adding Simulink Implementation Containers to a ConfigurationDesk Application.....	591										
Creating Precompiled SIC Files.....	592										
Updating Simulink Implementation Containers in ConfigurationDesk.....	595										

## Basics on Simulink Implementation Containers

<b>Description of a Simulink implementation container</b>	A Simulink implementation container is a container file containing the model code of a Simulink behavior model. You can add a Simulink implementation container file to your ConfigurationDesk application in the same way as adding other model implementations, such as Simulink behavior models or V-ECU implementations. The file name extension is SIC.
---	--

### Note

You do not need a MATLAB installation to use a Simulink implementation container in ConfigurationDesk.

<b>Generating SIC files</b>	A Simulink implementation container file can be created as follows: <ul style="list-style-type: none"><li>▪ From a Simulink behavior model by using the Model Interface Package for Simulink. For details, refer to <a href="#">Basics on Simulink Implementation Containers (Model Interface Package for Simulink - Modeling Guide)</a>.</li></ul>
-----------------------------	---

- From a TargetLink model using TargetLink 4.4 (dSPACE Release 2018-B) and later.

**Note**

The following limitations apply to SIC files created with TargetLink:

- SIC files generated with TargetLink 4.4 are not supported in application processes to which multiple model implementations are assigned.
- You cannot use SIC files that are generated with TargetLink for BSC file generation.

#### Importing and updating Simulink implementation containers

ConfigurationDesk lets you import Simulink implementation containers by adding them to your ConfigurationDesk application just like Simulink behavior models. When you import a Simulink implementation container, ConfigurationDesk generates a local copy of the Simulink implementation container in its file system. If required, you must update this Simulink implementation container in ConfigurationDesk.

#### Using SIC files with A2L file fragment

You can add a SIC file containing an A2L fragment to a ConfigurationDesk application. If you do so, an A2L file is created for the application process to which the SIC file is assigned during the build process. The A2L file contains the following elements:

- Model-specific variables provided by the A2L fragment of the related SIC file
- Variables for I/O functions and test automation
- DAQ raster names and the XCP service port number

You can assign these SIC files to separate application processes and configure an XCP service for them. For more information, refer to [Specifying Options for the Build Process](#) on page 707.

#### Using SIC files providing shared objects

During the build process, shared objects that are provided by the Simulink implementation containers and match the registered platform are archived in the real-time application. After the real-time application has been downloaded, the shared objects are available on the real-time system so that the Simulink implementation container can access them during simulation.

#### Restricted support of SIC files generated for dsrt64.tlc

ConfigurationDesk lets you add SIC files generated for the dsrt64.tlc target system file with the Model Interface Package for Simulink to a ConfigurationDesk application. You can use these SIC files for the generation of BSC files. However, you cannot use the SIC files to build a real-time application. In this case, ConfigurationDesk creates a conflict and aborts the build process. For more information on generating BSC files, refer to [Generating Bus Simulation Containers \(ConfigurationDesk Bus Manager Implementation Guide\)](#).

---

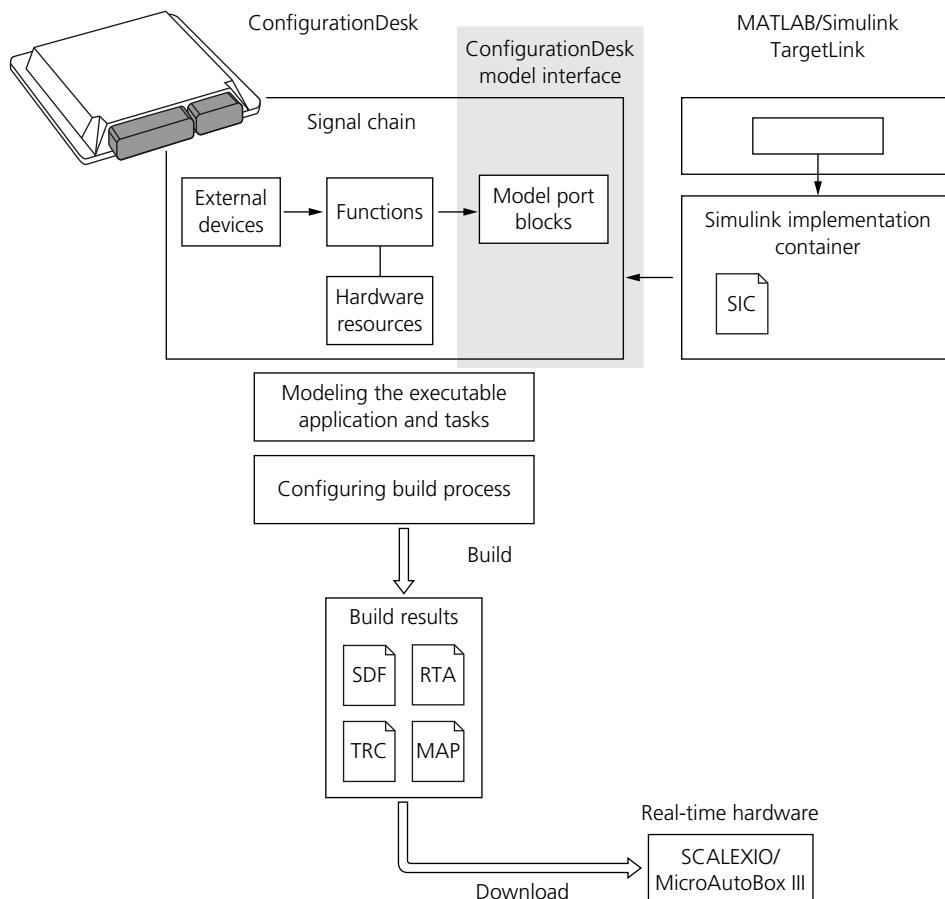
<b>Support of earlier SIC file versions</b>	ConfigurationDesk supports SIC files that are generated with combinations of earlier Model Interface Package for Simulink and MATLAB versions. This makes it possible to use earlier SIC file versions in real-time applications that run on a SCALEXIO Real-Time PC, MicroAutoBox III, or a DS6001 Processor Board. For information on the compatibility of earlier SIC file versions, refer to <a href="#">Compatibility of ConfigurationDesk 6.7</a> on page 36.
---	---

## Workflow for Integrating Simulink Implementation Containers in Executable Applications

---

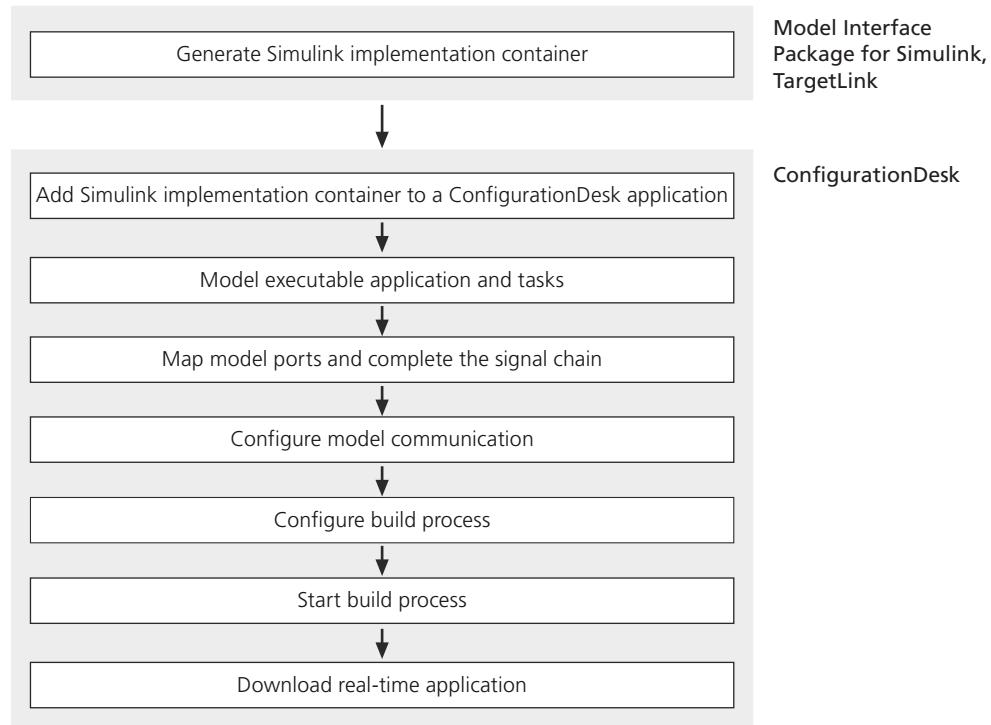
<b>Use scenario</b>	With ConfigurationDesk, you can integrate Simulink implementation containers in your executable application.
---------------------	--

**Overview** The following illustration gives you an overview of the use scenario:



**Workflow**

The workflow includes the steps that are specific to implementing a real-time application containing Simulation implementation containers.



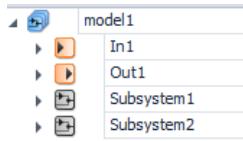
## Adding Simulink Implementation Containers to a ConfigurationDesk Application

**Adding an SIC file**

You can add an SIC file to your active ConfigurationDesk application by using the Add model command from the context menu of the Model Browser. Refer to [How to Add Model Implementation Containers to a ConfigurationDesk Application](#) on page 586. As an alternative, you can add an SIC file to a ConfigurationDesk application via the Project Manager. Refer to [How to Import a Model Topology](#) on page 112.

**Result of adding an SIC file to a ConfigurationDesk application**

ConfigurationDesk generates a local copy of the selected Simulink implementation container in its file system and displays its model topology in the Model Browser, marked with a  symbol.

**Next steps**

After you add the Simulink implementation container to your ConfigurationDesk application, perform the following steps:

- Model the executable application. Refer to [Modeling Executable Applications and Tasks](#) on page 471.
- Complete the signal chain.
- Configure and start the build and download process. Refer to [Building Real-Time Applications](#) on page 697.

## Creating Precompiled SIC Files

**Variants of precompiling SIC files**

ConfigurationDesk provides a method for you to precompile an [SIC file](#). The following variants are possible:

- Precompiled SIC files without readable source files  
ConfigurationDesk lets you convert SIC files with source files into SIC files without readable source files, but with a SCALEXIO-compatible or MicroAutoBox III-compatible library file which may be desirable for IP protection. In addition, the build process becomes faster when you use such a precompiled SIC file. You can add the converted SIC files to your ConfigurationDesk application.
- Precompiled SIC files that contain the original source files  
ConfigurationDesk lets you create precompiled SIC files that still contain the original source files. This can be useful if you want to use them on other platforms, e.g., in VEOS Player.

**Note**

Precompiled SIC files that contain the original source files are not IP-protected.

**Converting SIC files using the automation interface**

You can use the `SetCustomInformation` method of ConfigurationDesk's automation interface to convert an SIC file into a precompiled SIC file for use with SCALEXIO or MicroAutoBox III. For details on the `SetCustomInformation` method, refer to [ICaApplicationMain <>Interface>> \(ConfigurationDesk Automating Tool Handling\)](#).

The `SetCustomInformation` method has the following parameters:

- String
- Array

The following table shows the parameters and their values for converting SIC files into precompiled SIC files.

Parameter	Possible Values
String	<code>PrecompileSIC</code>
Array	<p>The array must consist of 1..5 entries. The following entries are possible:</p> <ol style="list-style-type: none"> <li>1. <b><code>pathToOriginalSic</code></b> A string containing the path of an existing SIC file (*.sic)</li> <li>2. [optional] <b><code>pathToTargetSic</code></b> <ul style="list-style-type: none"> <li>▪ A string containing the path to the file to be created</li> <li>▪ A string containing a path to the folder that the precompiled SIC file must be copied to</li> <li>▪ Only the name of an SIC file (*.sic)</li> <li>▪ <b>None</b> if you do not want to specify a target path</li> </ul> </li> <li>3. [optional] <b><code>additionalCompilerOptions</code></b> <ul style="list-style-type: none"> <li>▪ A string containing compiler options</li> <li>▪ <b>None</b> if you do not want to specify any compiler options</li> </ul> </li> <li>4. [optional] <b><code>keepSources</code></b> A string indicating if the precompiled SIC file must keep the original source files. The following values are possible:           <ul style="list-style-type: none"> <li>▪ True: The created SIC file contains the original source files and a precompiled SCALEXIO-compatible or MicroAutoBox III-compatible library. When you use the SIC file with ConfigurationDesk, the precompiled library is used to build the real-time application. When you use the SIC file, for example, in VEOS Player, the original source files are used to build the VEOS application.</li> <li>▪ False (default): The precompiled SIC file does not contain the original source files. It can only be used with SCALEXIO or MicroAutoBox III.</li> </ul> </li> <li>5. [optional] <b><code>targetPlatformType</code></b> A string specifying the target platform. The following values are possible:           <ul style="list-style-type: none"> <li>▪ <b>SCALEXIO_LNX</b></li> <li>▪ <b>DS1403</b></li> </ul> </li> </ol>

**Note**

- With dSPACE Release 2020-B, the operating system on SCALEXIO platforms has changed from a QNX®-based to a Linux™-based distribution. Precompiled SIC files that were built for dSPACE Release 2020-A and older are no longer compatible with the SCALEXIO system. You must precompile the SIC files from source code based on dSPACE Release 2020-B using the SCALEXIO\_LNX target platform identifier.
- If no platform is specified, SCALEXIO\_LNX is used as the default target platform.
- If you specify only the name of the target SIC file, the target SIC file is created in the path of the original SIC file.
- If you do not specify the name of the target SIC file, the target SIC file has the name of the original SIC file, extended with the \_precompiled suffix.
- If you do not specify a string for the `keepSources` entry, the target SIC file does not contain the original sources.

**Tip**

You can create precompiled SIC files without having to open a ConfigurationDesk project or application.

**Examples**

The following examples show you how to convert SIC files into precompiled SIC files:

- `Application.SetCustomInformation(["PrecompileSIC"], [r"C:\MyExample.sic"])`  
creates C:\MyExample\_precompiled.sic.
- `Application.SetCustomInformation(["PrecompileSIC"], [r"C:\MyExample.sic", r"WithoutSourceCode.sic"])`  
creates C:\WithoutSourceCode.sic.
- `Application.SetCustomInformation(["PrecompileSIC"], [r"C:\MyExample.sic", r"C:\DestinationPath\WithoutSourceCode.sic"])`  
creates C:\DestinationPath\WithoutSourceCode.sic.
- `Application.SetCustomInformation(["PrecompileSIC"], [r"C:\MyExample.sic", r"D:\DestinationPath"])`  
creates D:\DestinationPath\MyExample\_precompiled.sic.
- `Application.SetCustomInformation(["PrecompileSIC"], [r"C:\MyExample.sic", r"D:\DestinationPath", r"-DMACRODEFINED"])`  
creates D:\DestinationPath\MyExample\_precompiled.sic and during the compilation of the contained C Code the MACRODEFINED macro is defined.
- `Application.SetCustomInformation(["PrecompileSIC"], [r"C:\MyExample.sic", None, None, True])`

creates C:\MyExample\_precompiled.sic. The created SIC file contains the original source files and a precompiled SCALEXIO\_LNX-compatible library. When you use the SIC file with ConfigurationDesk, the precompiled library is used to build the SCALEXIO application. When you use the SIC file, for example, in VEOS Player, the original source files are used to build the VEOS application.

---

**Building real-time applications containing precompiled SIC files**

If an SIC file contains both a precompiled library in a supported version and source code, the precompiled library is used for the build process. The source code is ignored.

## Updating Simulink Implementation Containers in ConfigurationDesk

---

**Introduction**

If there are any modifications in the Simulink implementation container, you can make these modifications known to ConfigurationDesk.

**Updating Simulink implementation containers**

When you import a Simulink implementation container, ConfigurationDesk generates a local copy of the Simulink implementation container in its file system. Additionally, ConfigurationDesk saves both the absolute path to the Simulink implementation container's file location and its path relative to the ConfigurationDesk application's root directory. To make modifications in the Simulink implementation container known to ConfigurationDesk, you must update the Simulink implementation container in ConfigurationDesk by using the Reload command from the context menu of the Simulink implementation container, for example, in the Model Browser.

The Reload command tries to find the Simulink implementation container at a file location relative to the ConfigurationDesk application's root directory. If that is not successful, ConfigurationDesk tries to find the Simulink implementation container at the file location it was originally imported from. ConfigurationDesk then updates the data in the file system and in the ConfigurationDesk application with the data from the Simulink implementation container. If the Simulink implementation container cannot be updated successfully, the model topology remains unchanged. In this case, information on the failure causes is displayed in the Message Viewer.

**Resolved and unresolved Simulink implementation containers**

The handling of resolved and unresolved Simulink implementation containers and their elements complies with the handling of Simulink models. Refer to [Basics on Modeling Executable Applications](#) on page 475.

# Working with Bus Simulation Containers

<b>Introduction</b>	A bus simulation container lets you implement bus communication in a real-time application. You can add bus simulation container files (BSC files) to a ConfigurationDesk application to integrate them into your executable application, and execute them on your real-time hardware.
---------------------	--

Where to go from here	Information in this section
	<a href="#">Adding Bus Simulation Containers to a ConfigurationDesk Application</a> ..... 596 <a href="#">Workflow for Integrating Bus Simulation Containers in Executable Applications</a> ..... 599 <a href="#">Updating Bus Simulation Containers in ConfigurationDesk</a> ..... 601 <a href="#">Creating Precompiled BSC Files</a> ..... 602 <a href="#">Build Results for Real-Time Applications Containing Bus Simulation Containers</a> ..... 605

## Adding Bus Simulation Containers to a ConfigurationDesk Application

<b>Description of a bus simulation container</b>	A bus simulation container  lets you implement bus communication in an executable application. A bus simulation container file is configured and generated with the Bus Manager. Refer to <a href="#">Generating Bus Simulation Containers (ConfigurationDesk Bus Manager Implementation Guide)</a>  . The file name extension is BSC, therefore it is also called BSC file  .
--	---

<b>Adding a BSC file</b>	You can add a BSC file to your active ConfigurationDesk application in the same way as you add a Simulink model or a V-ECU implementation, for example. For more information, refer to <a href="#">How to Add Model Implementation Containers to a ConfigurationDesk Application</a> on page 586. As an alternative, you can add a BSC file to your ConfigurationDesk application via the Project Manager. Refer to <a href="#">How to Import a Model Topology</a> on page 112.
--------------------------	---

<b>Results of adding a BSC file to a ConfigurationDesk application</b>	Adding a BSC file to a ConfigurationDesk application provides the following results: <ul style="list-style-type: none"> <li>▪ ConfigurationDesk creates a Configuration port for each bus access request configured in the bus simulation container. You can map the Configuration</li> </ul>
--	---

ports to CAN or LIN function blocks in the working view. If you do this, some parameters of the function block are set to the values of the mapped Configuration port of the bus simulation container.

#### Tip

You can map multiple Configuration ports of a bus simulation container to one CAN or LIN function block. When you do this, you can assign the Configuration ports to the same hardware resource. This is useful, for example, if the Configuration ports were generated for bus access requests that represent the same communication cluster.

#### Note

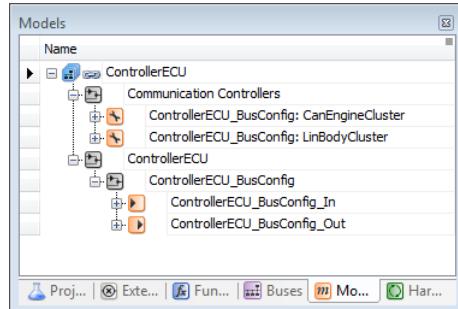
- If you do not map a configuration port of a Configuration Port block to a Configuration port of a CAN function block or a LIN function block, the related CAN or LIN communication is not implemented in the real-time application when you start the build process.
- You must not map the Configuration ports of a BSC file to bus function blocks that are used to specify the [bus access](#) for [bus access requests](#) of bus configurations.

- ConfigurationDesk creates model port blocks for the model ports provided by a BSC file.

#### Note

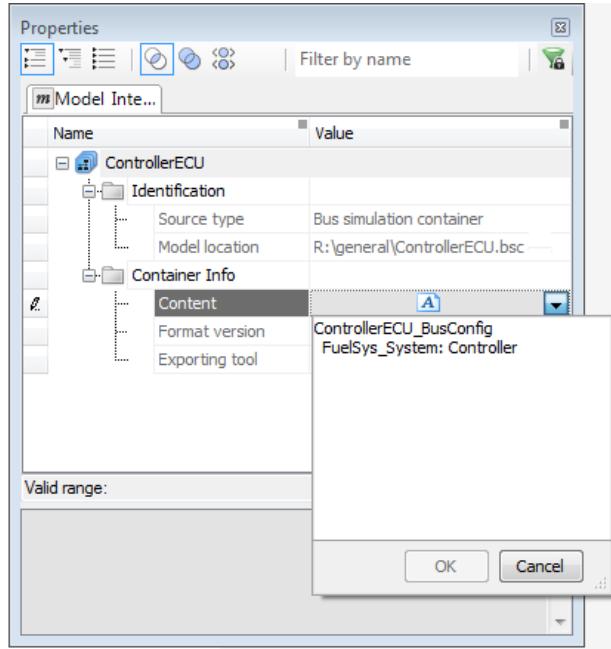
Mapping model ports of BSC files to model ports of other behavior models in an unfavorable way might reduce the run-time performance. For optimum run-time performance, it is recommended to observe specific mapping rules. Refer to [Model Communication Recommendations for Model Port Blocks with Structured Data Ports](#) on page 464.

If you add a BSC file to a ConfigurationDesk application, ConfigurationDesk displays the bus simulation container's model topology in the **Model Browser**, marked with a  symbol. The following illustration shows an example:



The properties of a bus simulation container are displayed in ConfigurationDesk's Properties Browser. The names of the communication matrix, clusters, and restbus participants are displayed in a text information field of the Content

property. Refer to [Model Implementation Properties \(ConfigurationDesk User Interface Reference\)](#).



#### Predefined task when using the PDU RX Interrupt feature of the Bus Manager

When you map a CAN function block to a Configuration port, ConfigurationDesk creates a predefined asynchronous task with an assigned I/O event and a runnable function. The default name is **CAN RX Task (<number>)**. If you use the PDU RX Interrupt feature of the Bus Manager, the task is required to ensure that CAN messages are received by the real-time application with a low latency. For this purpose, the task requires a high priority. Therefore, the default priority of the task is 0, which is the highest priority.

##### Note

To ensure optimum run-time behavior, do not change the default priority.

For more information on the PDU RX Interrupt feature, refer to [Triggering the Execution of Functions in a Behavior Model via RX Interrupts \(ConfigurationDesk Bus Manager Implementation Guide\)](#).

#### Using BSC files providing shared objects

During the build process, shared objects that are provided by the bus simulation containers and match the registered platform are archived in the real-time application. After the real-time application has been downloaded, the shared objects are available on the real-time system so that the bus simulation container can access them during simulation.

**Related topics****Basics**

Build Results for Real-Time Applications Containing Bus Simulation Containers.....	605
Updating Bus Simulation Containers in ConfigurationDesk.....	601

## Workflow for Integrating Bus Simulation Containers in Executable Applications

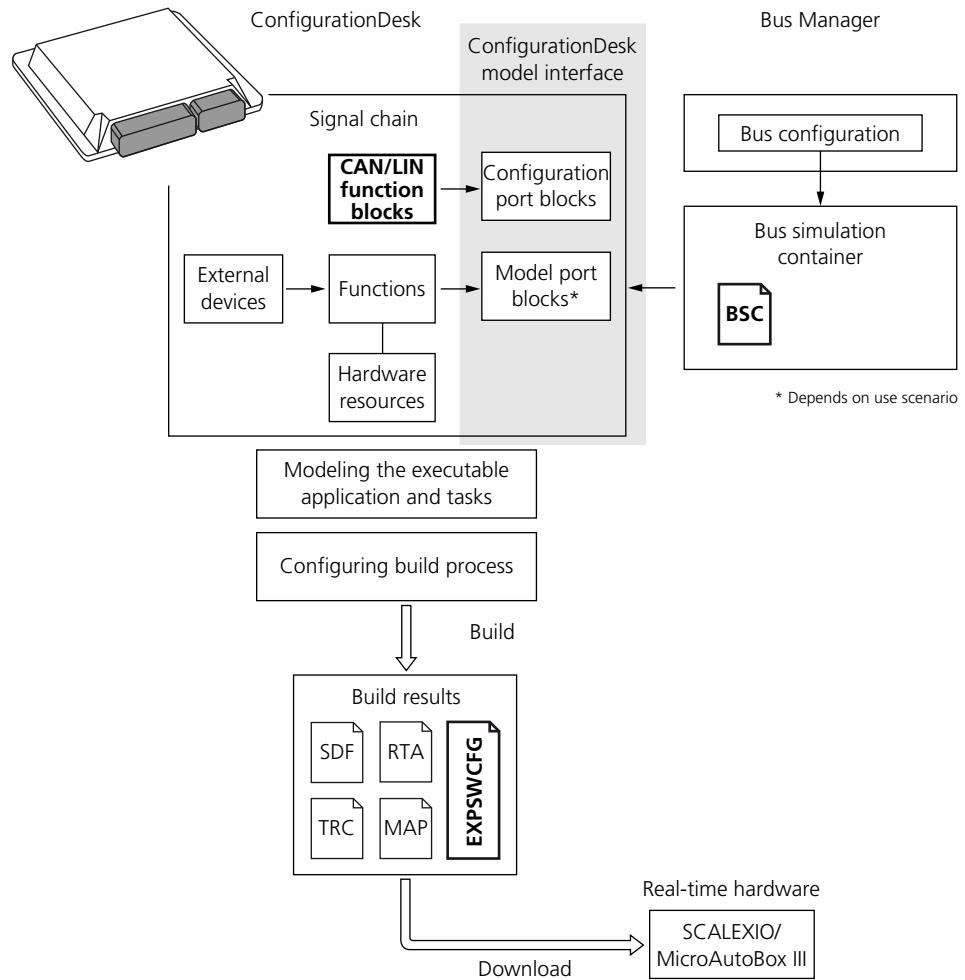
---

**Use scenario**

With ConfigurationDesk, you can integrate bus simulation containers generated by the Bus Manager in your executable application.

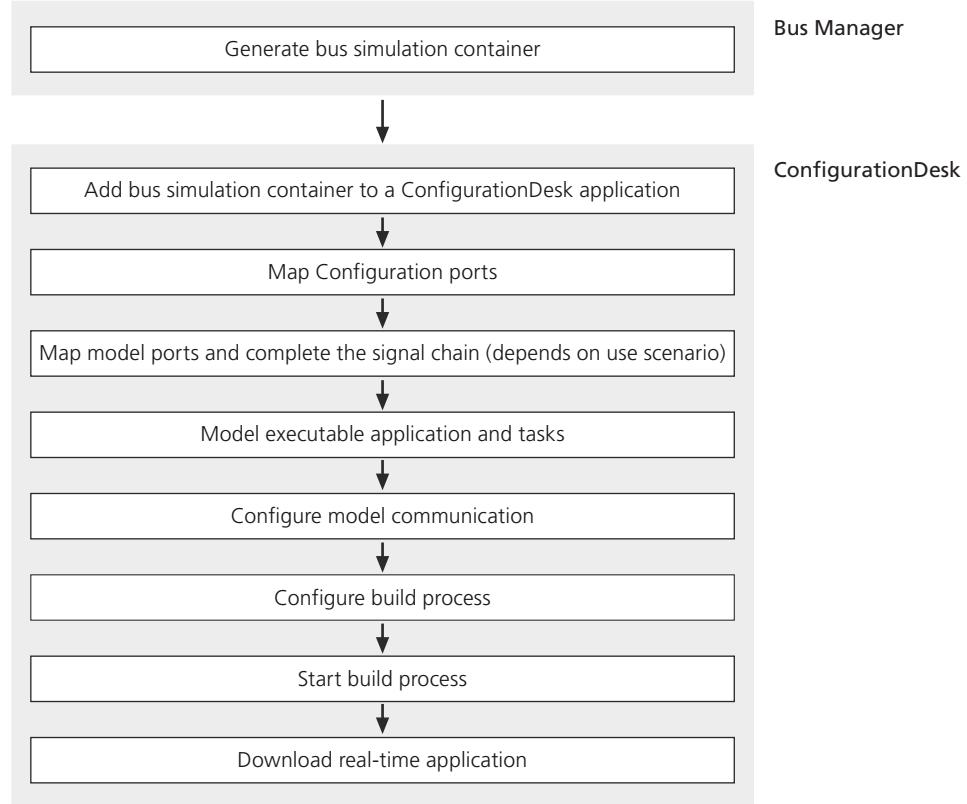
**Overview**

The following illustration gives you an overview of the use scenario. Parts that are specific to bus simulation containers are in bold letters.



**Workflow**

The workflow includes the steps that are specific to implementing a real-time application container a bus simulation container.



## Updating Bus Simulation Containers in ConfigurationDesk

**Working steps for updating bus simulation containers**

If you want to change a setting in the BSC file, the bus configuration of the bus simulation container must be modified using the Bus Manager, and a new BSC file must be generated. To make the modifications in the bus simulation container known to ConfigurationDesk, you must update the bus simulation container in ConfigurationDesk by using the Reload command from the context menu of the bus simulation container, for example, in the Model Browser.

**Update process**

When you add a bus simulation container to a ConfigurationDesk application, ConfigurationDesk generates a local copy of it in its file system. Additionally, ConfigurationDesk saves both the absolute path to the bus simulation container's file location and its path relative to the ConfigurationDesk application's root directory.

The Reload command first tries to find the bus simulation container at a file location relative to the ConfigurationDesk application's root directory. If that is not successful, ConfigurationDesk tries to find the bus simulation container at the file location it was originally added from. ConfigurationDesk then updates the data in the file system and in the ConfigurationDesk application with the data from the bus simulation container.

**Note**

If the bus simulation container cannot be updated successfully, the model topology remains unchanged. In this case, information on the failure causes is displayed in the Message Viewer.

---

**Update effects on function port mapping**

If you update a bus simulation container whose Configuration ports and/or model ports are mapped to function blocks, the mappings are updated according to the updated bus simulation container. In the following cases, unresolved mappings occur:

- If Configuration ports and/or model ports that are mapped to function blocks are not available in the updated bus simulation container, the affected Configuration Port blocks and model ports become unresolved.
- If the following names changed in the updated bus simulation container, the related Configuration Port blocks become unresolved:
  - Bus configuration name
  - Names of the communication matrix nodes in the bus configuration structure
  - Cluster name

If you delete all the mapping lines in all working views of an unresolved Configuration Port block or model port, the Configuration Port block/model port is removed from the model topology.

---

**Related topics**

**Basics**

[Adding Bus Simulation Containers to a ConfigurationDesk Application.....](#) 596

## Creating Precompiled BSC Files

---

**Variants of precompiling BSC files**

ConfigurationDesk lets you precompile [BSC files](#). The following variants are possible:

- Precompiled BSC files without readable source files  
ConfigurationDesk lets you convert BSC files with source files into BSC files without readable source files, but with a SCALEXIO-compatible or

MicroAutoBox III-compatible library file. This is useful for IP protection. In addition, the build process is faster with this type of precompiled BSC file. You can add the converted BSC files to your ConfigurationDesk application.

- Precompiled BSC files that contain the original source files

ConfigurationDesk lets you create precompiled BSC files that still contain the original source files. This can be useful if you want to use them on other platforms, e.g., in the VEOS Player.

#### Note

Precompiled BSC files that contain the original source files are not IP-protected.

### Converting BSC files using the automation interface

You can use the `SetCustomInformation` method of the ConfigurationDesk automation interface to convert a BSC file into a precompiled BSC file for use with SCALEXIO or a MicroAutoBox III. For more information on the `SetCustomInformation` method, refer to [ICaApplicationMain <>Interface>>](#) ([ConfigurationDesk Automating Tool Handling](#) ).

The `SetCustomInformation` method has the following parameters:

- String
- Array

The following table shows the parameters and their values for converting BSC files to precompiled BSC files.

Parameter	Possible Values
String	<code>PrecompileBSC</code>
Array	<p>The array must consist of 1..5 entries. The following entries are possible:</p> <ol style="list-style-type: none"> <li>1. <code>pathToOriginalBsc</code> A string containing the path of an existing BSC file (*.bsc)</li> <li>2. [optional] <code>pathToTargetBsc</code> <ul style="list-style-type: none"> <li>▪ A string containing the path to the file to be created</li> <li>▪ A string containing a path to the folder that the precompiled BSC file must be copied to</li> <li>▪ Only the name of a BSC file (*.bsc)</li> <li>▪ <code>None</code> if you do not want to specify a target path</li> </ul> </li> <li>3. [optional] <code>additionalCompilerOptions</code> <ul style="list-style-type: none"> <li>▪ A string containing compiler options</li> <li>▪ <code>None</code> if you do not want to specify any compiler options</li> </ul> </li> <li>4. [optional] <code>keepSources</code> A string indicating if the precompiled BSC file must keep the original source files. The following values are possible: <ul style="list-style-type: none"> <li>▪ <code>True</code>: The created BSC file contains the original source files and a precompiled SCALEXIO-compatible or MicroAutoBox III-compatible library. When you use the BSC file with ConfigurationDesk, the precompiled library is used</li> </ul> </li> </ol>

Parameter	Possible Values
	<p>to build the real-time application. When you use the BSC file, for example, in VEOS Player, the original source files are used to build the VEOS application.</p> <ul style="list-style-type: none"> <li>▪ False (default): The precompiled BSC file does not contain the original source files. It can only be used with SCALEXIO or a MicroAutoBox III.</li> </ul> <p>5. [optional] <b>targetPlatformType</b> A string specifying the target platform. The following values are possible:</p> <ul style="list-style-type: none"> <li>▪ SCALEXIO_LNX</li> <li>▪ DS1403</li> </ul>

**Note**

- With dSPACE Release 2020-B, the operating system on SCALEXIO platforms has changed from a QNX®-based to a Linux™-based distribution. Precompiled BSC files that were built for dSPACE Release 2020-A and older are no longer compatible with the SCALEXIO system. You must precompile the BSC files from source code based on dSPACE Release 2020-B using the SCALEXIO\_LNX target platform identifier.
- If no platform is specified, SCALEXIO\_LNX is used as the default target platform.
- If you specify only the name of the target BSC file, the target BSC file is created in the path of the original BSC file.
- If you do not specify the name of the target BSC file, the target BSC file has the name of the original BSC file, extended with the \_precompiled suffix.
- If you do not specify a string for the **keepSources** entry, the target BSC file does not contain the original sources.

**Tip**

You can create precompiled BSC files without having to open a ConfigurationDesk project or application.

**Examples**

The following examples show how to convert BSC files to precompiled BSC files:

- `Application.SetCustomInformation(["PrecompileBSC"], [r"C:\MyExample.bsc"])`  
creates C:\MyExample\_precompiled.bsc
- `Application.SetCustomInformation(["PrecompileBSC"], [r"C:\MyExample.bsc", r"WithoutSourceCode.bsc"])`  
creates C:\WithoutSourceCode.bsc

- Application.SetCustomInformation(["PrecompileBSC"], [r"C:\MyExample.bsc", r"C:\DestinationPath\WithoutSourceCode.bsc"])
 creates C:\DestinationPath\WithoutSourceCode.bsc
- Application.SetCustomInformation(["PrecompileBSC"], [r"C:\MyExample.bsc", r"D:\DestinationPath"])
 creates D:\DestinationPath\MyExample\_precompiled.bsc
- Application.SetCustomInformation(["PrecompileBSC"], [r"C:\MyExample.bsc", r"D:\DestinationPath", r"-DMACRODEFINED"])
 creates D:\DestinationPath\MyExample\_precompiled.bsc. During the compilation of the contained C Code, the MACRODEFINED macro is defined.
- Application.SetCustomInformation(["PrecompileBSC"], [r"C:\MyExample.bsc", None, None, True])
 creates C:\MyExample\_precompiled.bsc. The created BSC file contains the original source files and a precompiled SCALEXIO\_LNX-compatible library. When you use the BSC file with ConfigurationDesk, the precompiled library is used to build the SCALEXIO application. When you use the BSC file, for example, in the VEOS Player, the original source files are used to build the VEOS application.

---

**Creating BSC files precompiled for multiple platforms**

If a BSC file that is precompiled for a specific target platform still contains the original source files, you can precompile this BSC file for other target platforms. This means you can create a BSC file that is precompiled for multiple platforms. When precompiling a previously precompiled BSC file for the same target platform, the precompiled files in the BSC file are replaced.

---

**Building real-time applications containing precompiled BSC files**

If a BSC file contains both a precompiled library in a supported version and source code, the precompiled library is used for the build process. The source code is ignored.

## Build Results for Real-Time Applications Containing Bus Simulation Containers

---

**Files generated during the build process**

During the build process, ConfigurationDesk generates the following files for each application process that has a bus simulation container assigned:

- A TRC file
- An EXPSCFG file

**TRC file entries**

The TRC file created during the build process contains the following variable entries:

- Model variables in the **Model Root** group.
- Test automation access variables of the bus simulation in the **Signal chain** group.
- Task information variables in the **Simulation and RTOS** group.

The name of the TRC file is the same as the name of the application process the bus simulation container is assigned to. For details on the contents of a TRC file, refer to [Information on Variable Description Files \(TRC Files\)](#) on page 722.

---

**EXPSWCFG file**

The EXPSWCFG file contains the bus configuration data. This file references the TRC file and can be used by an experiment software such as ControlDesk to access and manipulate bus communication.

---

**Related topics**

Basics

[Adding Bus Simulation Containers to a ConfigurationDesk Application](#)..... 596

# Working with Functional Mock-up Units

## Introduction

ConfigurationDesk lets you add model implementations of the Functional Mock-up Unit type (called FMUs below) to a ConfigurationDesk application to integrate them into your executable application, and to execute them on your real-time hardware.

## FMI standard

FMUs are based on the Functional Mock-up Interface standard (FMI standard). The FMI standard defines an interface standard for model exchange and co-simulation. It is supported by different tools, for example, Dymola, MapleSim, and SimulationX. An FMU implements a model using the interfaces defined by the FMI standard.

## Required knowledge

Working with FMUs in ConfigurationDesk requires a knowledge of the following topics:

- Modeling executable applications and tasks. Refer to [Modeling Executable Applications and Tasks](#) on page 471.
- The FMI standard. For more information, refer to the following website: [www.fmi-standard.org](http://www.fmi-standard.org).

## Where to go from here

## Information in this section

<a href="#">Introduction to the FMU Support</a> .....	607
<a href="#">Handling FMUs in a ConfigurationDesk Application</a> .....	610

# Introduction to the FMU Support

## Where to go from here

## Information in this section

<a href="#">Definition of the FMI Standard and FMUs</a> .....	608
<a href="#">Preconditions for Using FMUs in ConfigurationDesk</a> .....	609

## Definition of the FMI Standard and FMUs

### Functional Mock-up Interface

The Functional Mock-up Interface (FMI) is a tool-independent standard that supports both the co-simulation and the exchange of dynamic models by using archive files containing a combination of XML files and C functions provided in source and/or binary form. The FMI defines the interface to be implemented by a Functional Mock-up Unit (FMU).

**FMI for Co-Simulation interface** The FMI for Co-Simulation interface is designed both for coupling simulation tools (simulator coupling, tool coupling) and for coupling subsystem models which have been exported by their simulators together with their solvers as runnable code.

**FMI for Model Exchange interface** The FMI for Model Exchange interface defines an interface to the model of a dynamic system described by differential, algebraic and discrete-time equations. Via the FMI for Model Exchange interface, these equations can be evaluated as needed in different simulation environments.

#### Note

- ConfigurationDesk supports only the FMI for Co-Simulation interface, not the FMI for Model Exchange interface. For detailed and up-to-date compatibility information on FMI support in ConfigurationDesk, refer to the following website: <http://www.dspace.com/go/FMI-Compatibility>.
- ConfigurationDesk supports FMUs that comply with the FMI 2.0 standard. You might be able to add an FMU based on other versions of the FMI standard to your ConfigurationDesk application. If you do, errors might occur during the build process. For detailed and up-to-date compatibility information on FMI support in ConfigurationDesk, refer to the following website: <http://www.dspace.com/go/FMI-Compatibility>.

### Functional Mock-up Unit

A Functional Mock-up Unit (FMU) 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 (optional)
- Shared objects needed for simulation (optional)

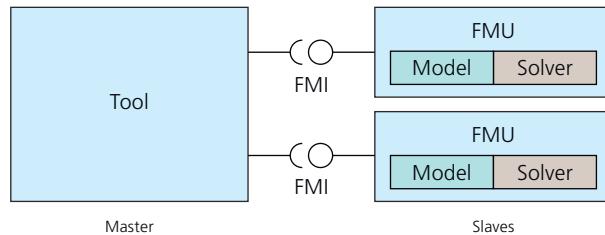
#### Note

There are limitations that apply to ConfigurationDesk's FMU support. Refer to [Limitations Concerning Functional Mock-up Units \(FMUs\)](#) on page 773.

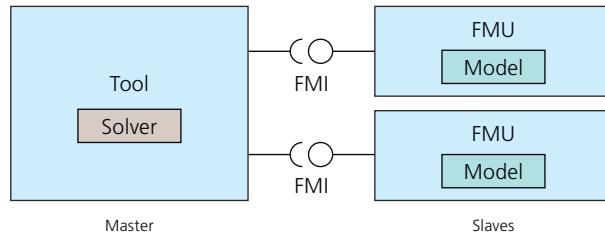
## Correlation between the FMI standard and FMUs

The illustrations below show how FMUs correlate with the FMI standard.

FMI for Co-Simulation interface:



FMI for Model Exchange interface:



## Preconditions for Using FMUs in ConfigurationDesk

### General preconditions

FMUs that you want to add to your ConfigurationDesk application must fulfill the following preconditions:

- They must comply with the FMI 2.0 for Co-Simulation standard.
- They must not require additional tools for simulation.
- They must not define a default start time unequal to 0.0.

#### Note

- FMUs must provide their C functions as source files or they must have been created with the PrecompileFMU command (refer to [Creating Precompiled FMUs](#) on page 618). You can still add an FMU that does not fulfill these preconditions to the ConfigurationDesk application. However, a warning is displayed and a build process is not possible in this case.
- If you try to import an FMU that does not fulfill these general preconditions, ConfigurationDesk aborts the import with an error message. No data is imported and no folder is created for the FMU in the ConfigurationDesk application folder.

### Preconditions for using FMUs with binaries

ConfigurationDesk also supports FMUs with source files that use functions and/or variables from compiled binaries, such as objects and static libraries. The

following preconditions have to be fulfilled to include the binaries in the real-time application during the build process:

- The binaries must be compiled with the GNU C/C++ Compiler.

The GNU C/C++ Compiler version used to compile the binary must be equal to or lower than that one installed with ConfigurationDesk. For information on the version that is automatically installed with the dSPACE software, refer to [Required C and C++ Compilers \(Installing dSPACE Software\)](#).

- The binaries must be stored in the following folder in an FMU:

- For SCALEXIO:

```
<FMU root>/binaries/SCALEXIO_LNX/gcc<x>.<y>
```

- For MicroAutoBox III:

```
<FMU root>/binaries/DS1403/gcc<x>.<y>
```

`gcc<x>.<y>` is a placeholder for the GNU C/C++ Compiler version, for example: `/MyFMUs/MyFMU_1.fmu/binaries/SCALEXIO_LNX/gcc5.2(Func01.o86)`.

An FMU can contain libraries for both SCALEXIO and MicroAutoBox III.

#### Note

- If the binaries are not located in the folder above or if the compiler version is higher than the version that is installed with ConfigurationDesk, the binaries are ignored during the build process. This can lead to error messages and the build process might be aborted.
- With dSPACE Release 2020-B, the base operating system on SCALEXIO platforms has changed from a QNX®-based to a Linux™-based distribution. Therefore, the folder where the binaries must be stored has changed. FMUs with libraries stored in the `<FMU root>/binaries/SCALEXIO/gcc<x>.<y>` folder are no longer supported by ConfigurationDesk.

## Handling FMUs in a ConfigurationDesk Application

### Introduction

ConfigurationDesk lets you add model implementations of the FMU type to your ConfigurationDesk application just like other model implementations. However, there are some specifics concerning their representation, runnable functions, build process and simulation behavior. There are also limitations concerning FMUs in ConfigurationDesk.

---

**Where to go from here****Information in this section**

Workflow for Integrating FMUs in Executable Applications.....	611
Updating FMUs in ConfigurationDesk.....	615
Special Aspects of Runnable Functions Resulting from FMUs.....	616
Creating Precompiled FMUs.....	618
Variable Description File Entries for Variables Provided by FMUs.....	620
Special Aspects of the Simulation Behavior.....	623

## Workflow for Integrating FMUs in Executable Applications

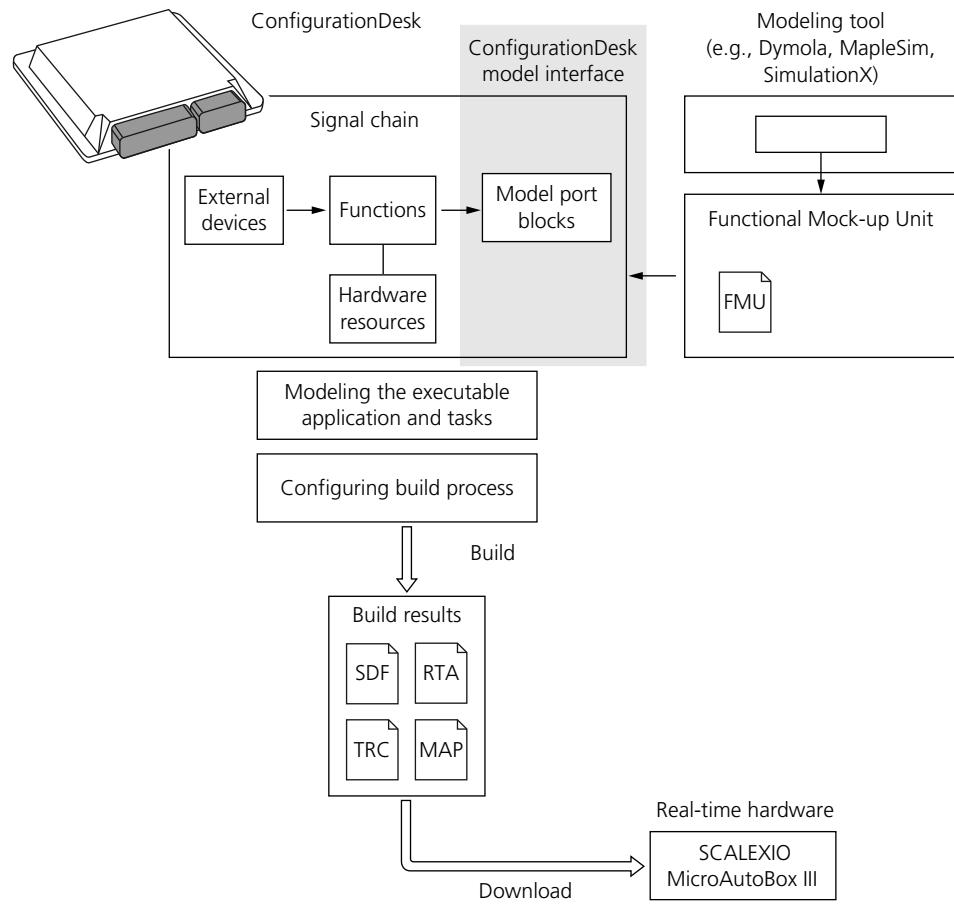
---

**Use scenario**

You can add an FMU to your active ConfigurationDesk application via the Project Manager in the same way as you add a Simulink model or a V-ECU implementation, for example.

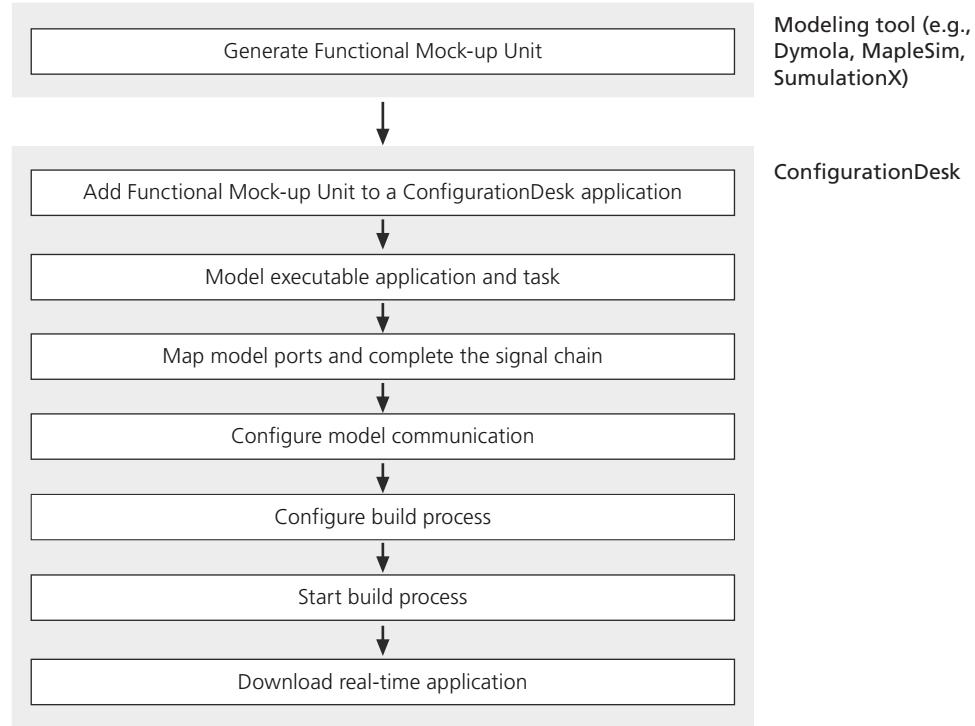
## Overview

The following illustration gives you an overview of the use scenario:



## Workflow

The workflow includes the steps that are specific to implementing a real-time application containing a Functional Mock-up Unit.

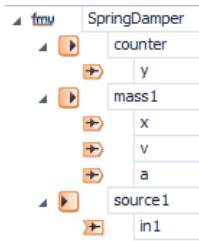


## Results of import

Adding an FMU to a ConfigurationDesk application provides the following results:

- ConfigurationDesk creates a new folder for the FMU within the application folder, and extracts the FMU file to the new folder. The new folder is named after the FMU.
- ConfigurationDesk displays the FMU's model topology in the **Model Browser**, marked with a **fmu** symbol.

The following illustration shows an example:



ConfigurationDesk creates a model port for each input and output whose data type is supported. The following data types are supported:

- Real
- Integer

- Boolean
- Enumeration

#### Note

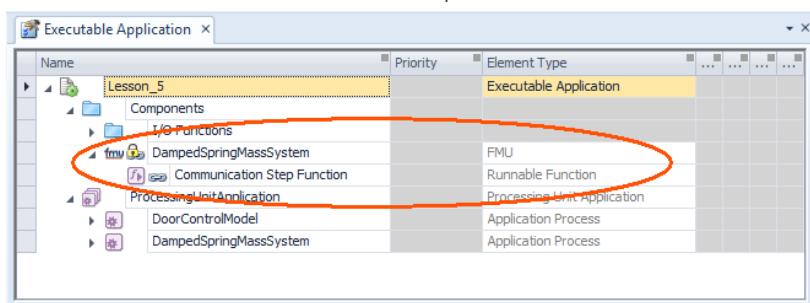
- In ConfigurationDesk, a data port with the enumeration data type is treated as a data port with the integer data type. The data type is displayed as int32.
  - FMU inputs and outputs with the string data type are not available in the model topology and thus cannot be used for model communication or I/O functionality.
- Variables with the **string** data type can be initialized via the start attribute in the `modelDescription.xml` file of an FMU. To change the initial value, you must edit the `modelDescription.xml` file, reload the FMU in ConfigurationDesk, and build the real-time application again. ConfigurationDesk does not generate TRC file entries for string variables. Therefore, you cannot access these variables during simulation in ControlDesk.

If the FMU provides information on the hierarchical structure of its inputs and outputs, ConfigurationDesk displays the model ports in the same hierarchy. In this case, the model topology can contain subsystems and model port blocks with more than one model port.

If the FMU does not provide information on the hierarchical structure of inputs and outputs, ConfigurationDesk creates one model port block for each input and each output.

- ConfigurationDesk creates a runnable function for the FMU in the Executable Application table.

The FMI standard specifies that an FMU must provide exactly one `fmiDoStep` function to calculate time steps. The runnable function created in ConfigurationDesk contains the call of this `fmiDoStep` function. This means that the calculation of an FMU cannot be spread across different tasks.



Name	Priority	Element Type
Lesson_5		Executable Application
Components		
I/O Functions		
DampedSpringMassSystem		FMU
Communication Step Function		Runnable Function
Processing Unit Application		Processing Unit Application
DoorControlModel		Application Process
DampedSpringMassSystem		Application Process

#### Note

If you remove an FMU from your ConfigurationDesk application, the related folder in the ConfigurationDesk application folder is deleted.

---

**Several FMUs with identical data port names within a ConfigurationDesk application**

ConfigurationDesk identifies each port of an FMU via the FMU's model name and the name of the port, including its hierarchical structure as defined in the FMU's model description file. Thus, ports in different FMUs have different identities, even if they have the same names. This means that:

- You can add different FMUs with identical port names to your ConfigurationDesk application without creating a Model port block: Duplicate ID conflict.
  - If you replace an FMU with an FMU that has the same port names, but a different model name, all ports that were provided by the original FMU and that are used in the ConfigurationDesk application become unresolved.
- 

**Using multiple FMUs in the same application process**

You can assign more than one FMU to the same application process. Refer to [Using Multiple Model Implementations in the Same Application Process](#) on page 491.

---

**Using FMUs providing shared objects**

During the build process, shared objects and resources that are provided by the FMU and match the registered platform are archived in the real-time application. Once the real-time application has been downloaded, the shared objects and resources are available on the real-time system so that the FMU can access them during simulation.

---

**Related topics**

HowTos

<a href="#">How to Add Model Implementation Containers to a ConfigurationDesk Application</a> .....	586
<a href="#">How to Import a Model Topology</a> .....	112

References

<a href="#">Add Model (ConfigurationDesk User Interface Reference)</a>	□
--	---

## Updating FMUs in ConfigurationDesk

---

**Introduction**

If there are any modifications in the FMU, you can update it in ConfigurationDesk. This is especially useful if a new FMU was generated because the model was changed in the modeling tool.

---

**Updating FMUs**

When you import an FMU, ConfigurationDesk generates a local copy of the FMU in its file system. Additionally, ConfigurationDesk saves both the absolute path to the FMU's file location and its path relative to the ConfigurationDesk application's root directory. To make modifications on the FMU known to

ConfigurationDesk, you must update the FMU in ConfigurationDesk by using the Reload command from the FMU's context menu, for example, in the Model Browser.

The Reload command tries to find the FMU at a file location relative to the ConfigurationDesk application's root directory. If that fails, ConfigurationDesk tries to find the FMU at the file location it was originally imported from.

ConfigurationDesk then updates the data in the file system and in the ConfigurationDesk application with the data from the FMU. If the FMU cannot be updated successfully, the model topology remains unchanged. In this case, information on the failure causes is displayed in the Message Viewer.

---

#### Resolved and unresolved FMUs

The handling of resolved and unresolved FMUs and their elements complies with the one for Simulink models. Refer to [Basics on Modeling Executable Applications](#) on page 475.

---

#### Related topics

#### References

[Reload \(ConfigurationDesk User Interface Reference\)](#)

## Special Aspects of Runnable Functions Resulting from FMUs

---

#### Introduction

If you create a [preconfigured application process](#) for an FMU, a task with a timer event is created, and the runnable function is assigned to that task. There are some facts to keep in mind when you trigger the runnable function resulting from an FMU import.

---

#### Different execution types

The following scenarios are possible concerning the execution types of an FMU:

- The FMU requires fixed step size execution
- The FMU allows both fixed step size execution and variable step size execution

The `canHandleVariableCommunicationStepSize` property of the `CoSimulation` element in the model description file specifies if the FMU requires a fixed step size execution or if it allows both fixed step size execution and variable step size execution.

**Fixed step size execution** An FMU can require a fixed step size for execution. The following scenarios are possible:

- The FMU provides a value for the fixed step size. In this case, the FMU's runnable function has a cycle time restriction indicated by the `Period` property of the runnable function.

- The FMU requires a fixed step size for execution but it does not provide a value for the fixed step size. In this case, ConfigurationDesk sets a cycle time restriction for the runnable function. This cycle time restriction is indicated by the Period property with the value -1. This means that the runnable function can be executed with any fixed step size  $> 0.0$ .

#### Note

If an FMU requires a fixed step size for execution and the expected fixed step size value is unknown, a conflict is shown if additionally one of the following is true:

- The event triggering the task that the runnable function is assigned to is not a timer event.
- The event triggering the task that the runnable function is assigned to is a timer event with Period = 0.0.

A warning is issued during the build process.

**Variable step size execution** An FMU can allow both fixed step size execution and variable step size execution. The following scenarios are possible:

- The FMU provides a preferred fixed step size. In this case, the FMU's runnable function has a cycle time restriction indicated by the Period property of the runnable function.
- The FMU does not have a preferred fixed step size. In this case, the FMU's runnable function does not have any cycle time restriction.

#### Note

If an FMU's runnable function provides a cycle time restriction and you set the Period property of the related timer event to a different value, a conflict is shown in the Conflicts Viewer. If the runnable function does not provide a cycle time restriction, the event triggering the related task can be one of these two elements:

- A timer event with any period
- An I/O event

#### Creating a preconfigured application process for an FMU

If you create a preconfigured application process for an FMU, a task with a timer event is created, and the FMU's runnable function is assigned to that task. The Period property of the timer event is set to the value indicated by the FMU. If the FMU does not provide any value for the period, the Period property of the timer event is set to the default value 0.001.

#### Automatic assignment of the runnable function

Suppose you delete a Simulink model from an application process and assign an FMU to that application process. The FMU's runnable function is then automatically assigned to the task to which the runnable function of the Simulink model's fastest periodic task had previously been assigned.

## Creating Precompiled FMUs

### Introduction

ConfigurationDesk provides a method for you to create an IP-protected FMU. For this purpose, ConfigurationDesk lets you convert FMUs with source files into FMUs without source files, but with a SCALEXIO-compatible or MicroAutoBox III-compatible library file. You can add the converted FMUs to your executable application.

ConfigurationDesk also supports the precompilation of FMUs with included binaries.

### Converting FMUs using the automation interface

You can use the `SetCustomInformation` method of ConfigurationDesk's automation interface to convert an FMU into an precompiled FMU. For information on the `SetCustomInformation` method, refer to [ICaApplicationMain <> \(ConfigurationDesk Automating Tool Handling\)](#).

The `SetCustomInformation` method has the following parameters:

- String
- Array

The following table shows the parameters and their values for converting FMUs into IP-protected FMUs.

Parameter	Possible Values
String	<code>PrecompileFMU</code>
Array	<p>The array must consist of 1...5 entries. The following values are possible:</p> <ol style="list-style-type: none"> <li>1. A string containing the path of an existing FMU file (*.fmu) or</li> <li>2. [optional] <ul style="list-style-type: none"> <li>▪ A string containing the path to the file to be created</li> <li>▪ A string containing a path to the folder that the precompiled FMU file must be copied to, or</li> <li>▪ Only the name of an FMU file (*.fmu)</li> </ul> </li> <li>3. [optional] A string containing compiler options</li> <li>4. [optional] <code>KeepSourceCode</code></li> </ol> <p>A flag indicating whether the precompiled FMU must keep the original source files. The following values are possible:</p> <ul style="list-style-type: none"> <li>▪ True: The created FMU contains the original source files and a precompiled SCALEXIO-compatible or MicroAutoBox III-compatible library.</li> </ul>

Parameter	Possible Values
	<ul style="list-style-type: none"> <li>▪ False (default): The precompiled FMU does not contain the original source files.</li> </ul> <p>5. [optional] <b>targetPlatformType</b> A string specifying the target platform. The following values are possible:</p> <ul style="list-style-type: none"> <li>▪ SCALEXIO_LNX</li> <li>▪ DS1403</li> </ul>

**Note**

- With dSPACE Release 2020-B, the operating system on SCALEXIO platforms has changed from a QNX®-based to a Linux™-based distribution. Precompiled FMUs that were built for dSPACE Release 2020-A and older are no longer compatible with the SCALEXIO system. You must precompile the FMUs from source code based on dSPACE Release 2020-B using the SCALEXIO\_LNX target platform identifier.
- If no platform is specified, SCALEXIO\_LNX is used as the default target platform.
- If you specify only the name of the target FMU file, the target FMU file is created in the path of the original FMU file.
- If you do not specify the name of the target FMU file, the target FMU file has the name of the original FMU file, extended with the \_precompiled suffix.

**Tip**

You can create precompiled FMUs without having to open a ConfigurationDesk project or application.

**Examples**

The following examples show how to convert FMU files into IP-protected FMU files:

- Application.SetCustomInformation(["PrecompileFMU"], [r"C:\MyExample.fmu"])
   
creates C:\MyExample\_precompiled.fmu
- Application.SetCustomInformation(["PrecompileFMU"], [r"C:\MyExample.fmu", r"WithoutSourceCode.fmu"])
   
creates C:\WithoutSourceCode.fmu
- Application.SetCustomInformation(["PrecompileFMU"], [r"C:\MyExample.fmu", r"C:\DestinationPath\WithoutSourceCode.fmu"])
   
creates C:\DestinationPath\WithoutSourceCode.fmu
- Application.SetCustomInformation(["PrecompileFMU"], [r"C:\MyExample.fmu", r"D:\DestinationPath"])
   
creates D:\DestinationPath\MyExample\_precompiled.fmu

- `Application.SetCustomInformation(["PrecompileFMU"], [r"C:\MyExample.fmu", r"D:\DestinationPath", r"-DMACRODEFINED"])`  
creates D:\DestinationPath\MyExample\_precompiled.fmu and during the compilation of the contained C Code the MACRODEFINED macro is defined.
- `Application.SetCustomInformation(["PrecompileFMU"], [r"C:\MyExample.fmu", None, None, True])`  
creates C:\MyExample\_precompiled.fmu. The created FMU contains the original source files and a precompiled SCALEXIO\_LNX-compatible library.

**FMUs precompiled for different dSPACE Releases**

FMUs can contain precompiled libraries for different dSPACE Releases. If you precompile an FMU that already contains a precompiled library, keep the following information in mind:

- If you precompile an FMU that already contains a platform-compatible library for the dSPACE Release you currently use, this precompiled library is replaced.
- If you precompile an FMU that already contains a platform-compatible library for a dSPACE Release that is older or newer than the currently used one, this precompiled library is kept in addition to the new precompiled library.

**Building real-time applications containing precompiled FMUs**

If your ConfigurationDesk application contains precompiled FMUs, keep the following information in mind if you start the build process:

- If an FMU contains both a precompiled library in a supported version and source code, the precompiled library is used for the build process. The source code is ignored.
- If an FMU contains only source code and not any precompiled library, the source code is used for the build process.
- If an FMU contains precompiled libraries from several supported ConfigurationDesk versions, ConfigurationDesk uses the precompiled library created with the newest compatible ConfigurationDesk version for the build process. Precompiled libraries created with ConfigurationDesk versions that are newer than the currently used version are ignored.

## Variable Description File Entries for Variables Provided by FMUs

**Variables provided by an FMU**

The FMU's model description file contains information on the following variable attributes for all the variables provided by an FMU:

- **variability**
- **causality**

The table below shows the attributes and their possible values:

Attribute	Possible Values
causality	▪ input

Attribute	Possible Values
	<ul style="list-style-type: none"> <li>▪ output</li> <li>▪ parameter</li> <li>▪ calculatedParameter</li> <li>▪ local</li> <li>▪ independent</li> </ul>
variability	<ul style="list-style-type: none"> <li>▪ constant</li> <li>▪ fixed</li> <li>▪ tunable</li> <li>▪ discrete</li> <li>▪ continuous</li> </ul>

The attributes' values determine the variables that ConfigurationDesk generates TRC entries for, and the properties of those variables, for example, if the variable is a parameter or a signal, or if the variable is read-only (see below).

#### Note

- For each variable, ConfigurationDesk generates information on the variability and the causality into the TRC file. The information is written to the Description property of the related variable.
- For more information on the variable attributes and their possible values, refer to the specification of the FMI standard, version 2.0 on the following website: <http://www.fmi-standard.org>.

---

#### Tunable parameters in the TRC file

ConfigurationDesk generates tunable parameters into the TRC file for variables with the following attributes:

- **variability = tunable**
- **causality = parameter**

This means that you can change the values of these tunable parameters during simulation. The new value takes effect in the next simulation step.

**Initial values of tunable parameters** If the start value of a tunable parameter is available in the FMU's model description file, the initial value of the variable is generated into the TRC file.

#### Note

During the build process, ConfigurationDesk generates TRC file entries for fixed parameters:

- **causality = parameter and variability = fixed**

The values of these parameters can be changed during simulation. However, changing these parameters does not effect the simulation results until the simulation is stopped and started again.

In the experiment software, fixed parameters are indicated by the following symbol: .

**Support of local variables provided by an FMU**

During the build process, ConfigurationDesk generates signal entries for local variables into the TRC file. In the FMU context, local variables are variables with the following combination of variability and causality:

- **constant local**
  - **fixed local**
  - **tunable local**
  - **discrete local**
  - **continuous local**
  - **continuous independent**
- 

**Input and output signals of the FMU**

ConfigurationDesk generates signal entries for input and output signals into the TRC file for all the variables whose causality = input or output: i.e., for all the variables that model ports are available for.

The TRC file entries for input and output signals are treated exactly like the TRC file entries for local variables (see above).

---

**Support of calculated parameters**

During the build process, ConfigurationDesk generates read-only parameters into the TRC file for variables with the following attributes:

`causality = calculatedParameter`

**Note**

The TRC file entries for these variables are read-only: i.e., the values of the variables cannot be changed during simulation irrespective of whether the application state is Running or Stopped.

---

**Hierarchical structure of the TRC file**

The FMI standard allows an FMU to follow different variable naming conventions (**structured** or **flat**). This means that an FMU can optionally provide hierarchy information on its variables. ConfigurationDesk considers this information for TRC file generation:

**Structured TRC file entries** If the variable naming convention is **structured**, ConfigurationDesk generates hierarchically structured TRC file entries that reflect the structure specified in the `modelDescription.xml` file.

**Flat TRC file entries** If the variable naming convention attribute is **flat**, ConfigurationDesk sorts the variables by causality within the TRC file. This means there is one group for each causality (outputs, inputs, parameters, local variables). Each group contains a flat variable list.

---

**Range of variables**

If the FMU provides the minimum and maximum value for a variable, the range of the variable is generated into the TRC file.

**Supported and unsupported data types**

The following data types are supported for variables provided by an FMU:

- Real
- Integer
- Boolean
- Enumeration

**Note**

ConfigurationDesk does not generate TRC file entries for variables with the **string** data type.

## Special Aspects of the Simulation Behavior

**Simulation behavior**

The special aspects of the simulation behavior of executable applications containing FMUs are as follows:

**Asynchronous step functions** ConfigurationDesk supports FMUs that allow a step function to be called asynchronously. In this case, the `canRunAsynchronously` property of the `CoSimulation` element is set to `True` in the `modelDescription.xml` file. ConfigurationDesk forces the step function of the associated FMU to be executed synchronously.

**Note**

In the FMI context, asynchronous means that the step function is called and returns to the calling function immediately. The calculation of the step function is executed in a separate thread. The calling function can poll the calculation state of the step function.

**Simulation stop time** If an FMU specifies a simulation stop time in the `stopTime` attribute of the `DefaultExperiment` element in the `modelDescription.xml` file, ConfigurationDesk ensures that the simulation stops when the specified stop time is reached.

**Note**

In a multicore application or multi-processing-unit application, the simulation is stopped on all cores and processing units, not only on the core that calculates the FMU.

**Debug logging** ConfigurationDesk allows the FMU's messages to be printed to the dSPACE Log. You can enable or disable debug logging in ConfigurationDesk. To enable debug logging, you must define the macro "`DSFMULogging=0|1`" (0=disabled, 1=enabled). This can be done via the

Macros to be defined property of the build configuration set that the related application process is assigned to. The default value is 0 (disabled).

**Model port start value for model communication** ConfigurationDesk uses the optionally defined start value of a variable provided by an FMU in a simulation. This means that all the variables (input and output variables, parameters, and local variables) are initialized with that start value. During simulation, the value of an FMU's input signal is equal to the defined start value, unless a connected port (e.g., another model port or a function port) provides a new value. For unconnected input ports, the defined start value is used.

# Working with V-ECU Implementations

## Where to go from here

## Information in this section

Handling V-ECU Implementations in ConfigurationDesk.....	625
Configuring Executable Applications Containing V-ECU Implementations.....	635
Details on the Build Process for Real-Time Applications Containing V-ECU Implementations.....	641

## Handling V-ECU Implementations in ConfigurationDesk

## Where to go from here

## Information in this section

Basics on V-ECU Implementations.....	625
Workflow for Integrating V-ECU Implementations in Executable Applications.....	629
Adding V-ECU Implementations to a ConfigurationDesk Application.....	632
Updating V-ECU Implementations Used in a ConfigurationDesk Application.....	633

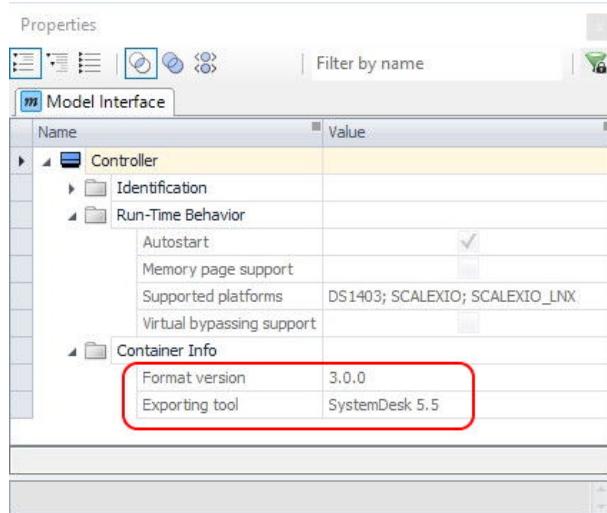
## Basics on V-ECU Implementations

### Introduction

With ConfigurationDesk, you can integrate V-ECU implementations in your executable applications (real-time applications). These V-ECU implementations can communicate with real ECUs via CAN bus or LIN bus. Thus, an ECU network consisting of real and virtual ECUs can be used for simulations and tests.

V-ECU implementations can be provided by TargetLink or SystemDesk.

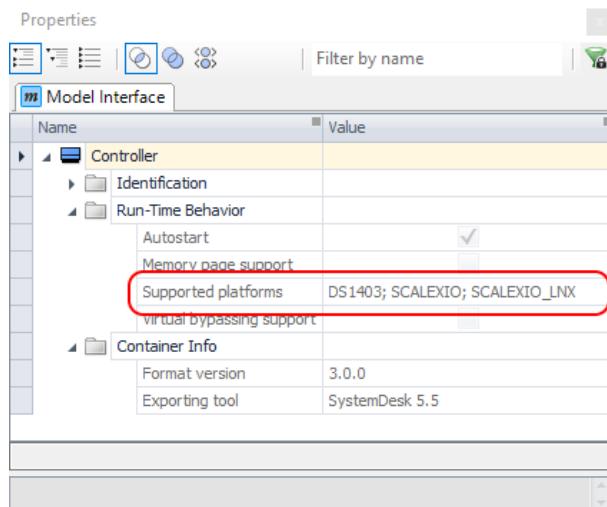
<b>Required licenses</b>	<p>Working with V-ECU implementations in ConfigurationDesk requires the following licenses:</p> <ul style="list-style-type: none"><li>▪ CFD_I_VECU<ul style="list-style-type: none"><li>If you want to work with V-ECU implementations in your ConfigurationDesk application, the V-ECU module license (CFD_I_VECU) is required.</li></ul></li><li>▪ CFD_I_XCP<ul style="list-style-type: none"><li>If a V-ECU provides XCP service access, the XCP module (CFD_I_XCP) license is required.</li></ul></li></ul>
	<p><b>Note</b></p> <p>If a required license is not available, the functionality of ConfigurationDesk is limited, and relevant error messages are displayed.</p>
	<p>For more information, refer to <a href="#">Overview of Licenses</a> on page 40.</p>
<b>Components of a V-ECU implementation</b>	<p>A V-ECU implementation is a software package consisting of:</p> <ul style="list-style-type: none"><li>▪ Hardware-independent software parts of an ECU defined by C or C++ code</li><li>▪ Description files for interfaces to hardware-dependent software parts</li><li>▪ An A2L file for the variables of the C/C++ code</li><li>▪ A catalog file listing all files and additional information (see below)</li></ul> <p>For ConfigurationDesk, V-ECU implementations are model implementations similar to Simulink behavior models.</p>
	<p><b>Note</b></p> <p>A V-ECU implementation either contains AUTOSAR code or it contains company-specific code that does not comply to the AUTOSAR standard. You can create V-ECU implementations with company-specific code via SystemDesk's V-ECU Manager. The following workflows refer to V-ECU implementations with AUTOSAR code, but they are also valid for V-ECU implementations with company-specific code.</p>
<b>V-ECU implementation container</b>	<p>A V-ECU implementation container is a ZIP file containing a V-ECU implementation. V-ECU implementation containers are exported by TargetLink or SystemDesk as VECU files (as of Release 2020-B) or CTLGZ files (Release 2020-A and older). You can add a V-ECU implementation container to your active ConfigurationDesk application just like adding a Simulink model (SLX or SIC file).</p> <p>In its Properties Browser, ConfigurationDesk displays the version of the V-ECU implementation container and the tool that the container was exported from. The following illustration shows an example for a V-ECU implementation container based on a VECU file:</p>



For an overview of the supported versions of V-ECU implementation containers, refer to [Compatibility of ConfigurationDesk 6.7](#) on page 36.

#### Support of different simulation platforms (V-ECU files only)

VECU files can contain data packages, such as binaries, or source files for different simulation platforms. This lets you simulate the same V-ECU implementation on different platforms without having to make platform-specific changes. ConfigurationDesk displays the platforms supported by a V-ECU implementation at the Supported platforms property in the Properties Browser. Refer to the following illustration:



The following table shows the supported platforms and their displayed names in the Properties Browser:

Platform	Platform Name in the Properties Browser
SCALEXIO	SCALEXIO
SCALEXIO with Linux® operating system	SCALEXIO_LNX
MicroAutoBox III	DS1403

#### V-ECU implementations containing AUTOSAR code

ConfigurationDesk supports the following types of V-ECU implementations that contain AUTOSAR code:

- V-ECU implementations that provide a reduced AUTOSAR stack. You can add your application code to these V-ECU implementations. ConfigurationDesk supports the following modules of the communication hardware abstraction:
  - CanIf
  - LinIf
- V-ECU implementations that provide only hardware-specific AUTOSAR modules of the microcontroller abstraction layer (MCAL). You can add your own AUTOSAR stack to these V-ECU implementations. ConfigurationDesk supports the following MCAL modules:
  - Sab module
  - Os module
  - Dap module
  - CanDrv module
  - LinDrv module

If you try to add V-ECU implementations that use other MCAL modules to a ConfigurationDesk application, the import is aborted and ConfigurationDesk displays an error message.

#### Note

- If you add a V-ECU implementation that contains both CanIf module configurations and CanDrv module configurations to a ConfigurationDesk application, only the configurations of the CanDrv module are imported into ConfigurationDesk.
- If you add a V-ECU implementation that contains both LinIf module configurations and LinDrv module configurations to a ConfigurationDesk application, only the configurations of the LinDrv module are imported into ConfigurationDesk.

---

**Related topics****Basics**

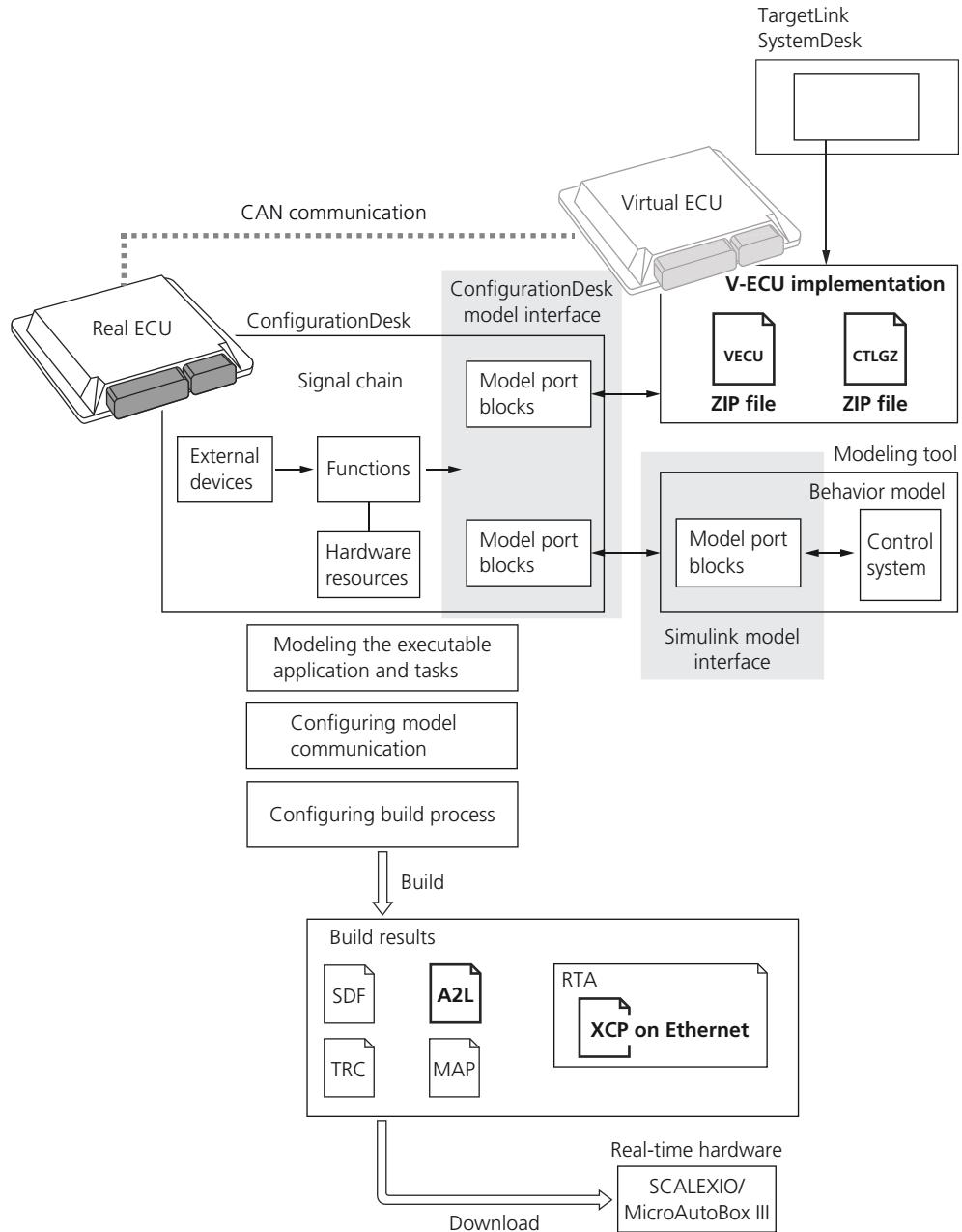
[Modeling Executable Applications and Tasks.....](#) 471

## Workflow for Integrating V-ECU Implementations in Executable Applications

---

**Overview**

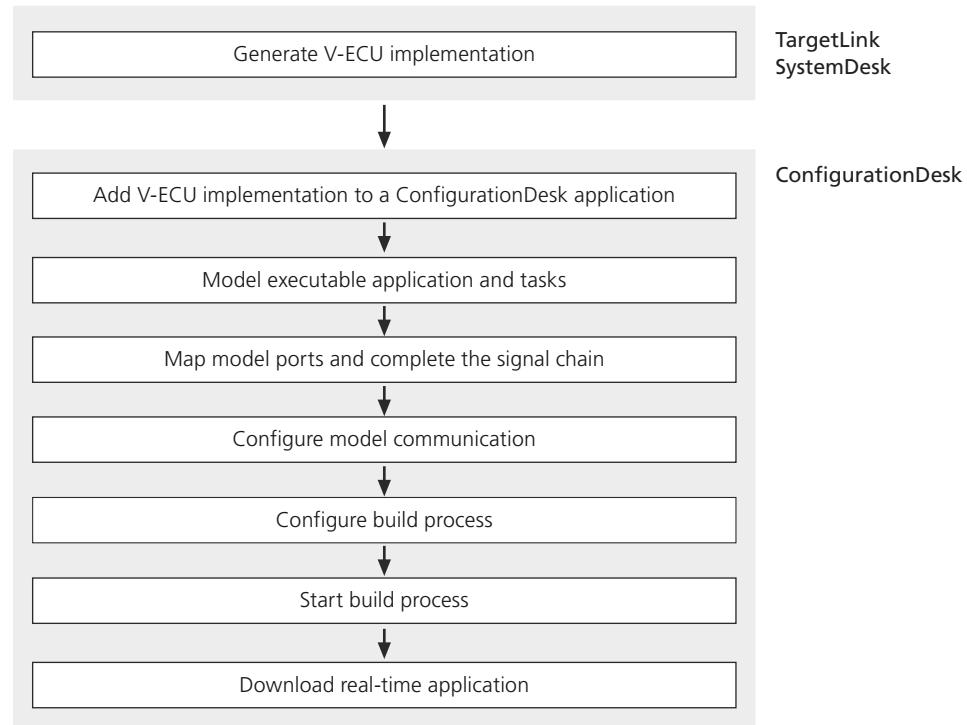
The following illustration gives you an overview of the use scenario for integrating V-ECU implementations in executable applications. Parts that are specific for V-ECU implementations are in bold letters.



The representation of a V-ECU in ConfigurationDesk is a specific model implementation. This so-called V-ECU implementation is part of the model topology, and the interface to the I/O functionality in ConfigurationDesk is realized via model port blocks just as with Simulink behavior models.

**Workflow**

The workflow includes the steps that are specific to implementing a real-time application containing a V-ECU implementation.

**Note**

Only one V-ECU implementation is allowed per application process. If you add more than one V-ECU implementation to an application process, a conflict is shown in the **Conflicts Viewer**. You must resolve the conflict before you start the build process. Otherwise, the build process is aborted.

**Tip**

You can export a ConfigurationDesk application containing V-ECU implementations as a ZIP file. If you import that ZIP file into ConfigurationDesk installed on another PC, all the operations you can perform in V-ECU implementations are available there.

**Limitations**

There are limitations that apply to working with V-ECU implementations in ConfigurationDesk. Refer to [Limitations Concerning V-ECU Implementations](#) on page 769.

## Adding V-ECU Implementations to a ConfigurationDesk Application

### Adding a VECU or CTLGZ file

ConfigurationDesk lets you add V-ECU implementations (VECU or CTLGZ files) to your ConfigurationDesk application by using the Add model command from the context menu of the Model Browser. Refer to [How to Add Model Implementation Containers to a ConfigurationDesk Application](#) on page 586. As an alternative, you can add V-ECU implementation to a ConfigurationDesk application via the Project Manager. Refer to [How to Import a Model Topology](#) on page 112.

### Result of adding a VECU or CTLGZ file to a ConfigurationDesk application

ConfigurationDesk generates a local copy of the selected V-ECU implementation container in its file system and displays its model topology in the Model Browser, marked with a  symbol.



If the V-ECU implementation cannot be added to your ConfigurationDesk application successfully, the Add Model dialog remains open for you to change the selected options, or to cancel the operation.

#### Note

ConfigurationDesk identifies each V-ECU implementation port via the V-ECU implementation name, the port name, and the path of the port within the V-ECU implementation. Thus, ports in different V-ECU implementations have different identities, even if they have the same name and the same path within the V-ECU implementation. This means that:

- You can add different V-ECU implementations with identical port names to your ConfigurationDesk application without creating a Model port block: Duplicate ID conflict.
- Suppose you replace a V-ECU implementation with a V-ECU implementation that has the same port names, but a different V-ECU implementation name by using one of the following methods:
  - You use the Replace Model Topology command.
  - You delete the V-ECU implementation using the Delete from Topology command and afterwards, add a new V-ECU implementation using the Add Model command.

In both cases, all ports that are provided by the original V-ECU implementation and that are used in the ConfigurationDesk application become unresolved. To keep the V-ECU implementation and ports resolved, replace one V-ECU implementation with another by using the Replace Model command. For more information, refer to [Replacing Model Implementations](#) on page 498.

## Updating V-ECU implementations

When you import a V-ECU implementation container, ConfigurationDesk generates a local copy of the V-ECU implementation container in its file system. If required, you must update this V-ECU implementation in ConfigurationDesk. Refer to [Updating V-ECU Implementations Used in a ConfigurationDesk Application](#) on page 633.

### Next steps

After you add the V-ECU implementation to your ConfigurationDesk application, you should perform the following steps:

- Model the executable application. For further information, see [Configuring Executable Applications Containing V-ECU Implementations](#) on page 635.
- Complete the signal chain.
- Configure and start the build and download process. Refer to [Details on the Build Process for Real-Time Applications Containing V-ECU Implementations](#) on page 641.

### Related topics

#### References

[Add Model \(ConfigurationDesk User Interface Reference\)](#)

## Updating V-ECU Implementations Used in a ConfigurationDesk Application

### Introduction

If there are any modifications in the V-ECU implementation container resulting from an export from SystemDesk or TargetLink, you can make these modifications known to ConfigurationDesk.

## Updating V-ECU implementations

When you import a V-ECU implementation container, ConfigurationDesk creates a local copy of the V-ECU implementation container in its file system. In addition, ConfigurationDesk saves both the absolute path to the V-ECU implementation's file location and its path relative to the ConfigurationDesk application's root directory.

To let ConfigurationDesk know about modifications on the V-ECU implementation container, you must update the V-ECU implementation in ConfigurationDesk by using the Reload command from the context menu of the V-ECU implementation in the Model Browser, for example. The Reload command tries to find the V-ECU implementation container at a file location relative to the ConfigurationDesk application's root directory. If that is not successful, ConfigurationDesk tries to find the V-ECU implementation container at the file location it was originally imported from. ConfigurationDesk then

updates the data in the file system and in the ConfigurationDesk application with the data from the V-ECU implementation container.

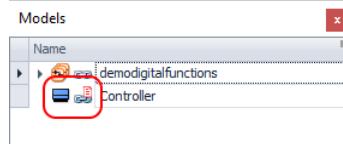
If the V-ECU implementation cannot be updated successfully, the model topology remains unchanged and information on the causes of the failure is displayed in the Message Viewer.

#### Possible data inconsistencies

In certain situations, data inconsistencies between the file system and the local copy of the V-ECU implementation container might occur. To avoid data loss, you must become familiar with the reasons for data inconsistencies and how you can correct them. Data inconsistencies can be caused by anyone of the following operations:

- You executed the Add Model, Reload or a delete command for a V-ECU implementation, and then closed ConfigurationDesk without saving your project.
- You executed the Add Model, Reload or a delete command for a V-ECU implementation, for example, in project A, and then opened project B without saving project A.
- You executed the Add Model, Reload or a delete command for a V-ECU implementation and then executed the Save as command without saving the current ConfigurationDesk project.

If you close your ConfigurationDesk project without saving it, ConfigurationDesk issues a warning which informs you of the possible data inconsistencies that can occur. In this case, you cannot perform the build process. If you load a ConfigurationDesk application containing data inconsistencies, ConfigurationDesk also issues a warning. V-ECU implementations containing inconsistent data are marked with a state icon:



#### Note

If you execute the Save as command without saving the current ConfigurationDesk project (see above), no warning is issued.

To fix data inconsistencies, use the Reload command to update the V-ECU implementation.

#### Resolved and unresolved V-ECU implementations

The handling of resolved and unresolved V-ECU implementations and their elements complies with the handling for Simulink models. Refer to [Basics on Modeling Executable Applications](#) on page 475.

**Related topics****References**

[Reload \(ConfigurationDesk User Interface Reference\)](#)

# Configuring Executable Applications Containing V-ECU Implementations

**Where to go from here****Information in this section**

Configuring Properties of Tasks and Events.....	635
Delaying the Start of V-ECU Implementations.....	638
Special Aspects of V-ECU Implementations Containing CAN Controllers.....	639
Special Aspects of V-ECU Implementations Containing LIN Controllers.....	639

## Configuring Properties of Tasks and Events

**Introduction**

There are some differences between tasks and events provided by a V-ECU implementation and tasks and events as described in [Modeling Executable Applications and Tasks](#) on page 471.

**Note**

You must not add user-defined tasks to an application process that has an assigned V-ECU implementation.

**Tasks of V-ECU implementations**

When you add a V-ECU implementation to your ConfigurationDesk application, the following tasks are displayed in the Executable Application table:

- The startup task
- AUTOSAR OS tasks provided by the V-ECU implementation
- AUTOSAR OS tasks created by ConfigurationDesk for CAN controllers

**Startup task of V-ECU implementations** Each V-ECU implementation contains a startup task that invokes the startup function of the V-ECU implementation (`Ecum_Init()` for an AUTOSAR ECU). You cannot change any of its properties.

**AUTOSAR OS tasks** AUTOSAR OS tasks have the following properties:

- Name
- Source type
- Source name
- DAQ raster name
- Has autostart configuration
- Preemptable
- Maximum number of task activations allowed
- Priority

#### Note

In the AUTOSAR standard, higher values indicate higher task priorities. For tasks provided by Simulink behavior models and Functional Mock-up Units (FMUs), it is the other way round: lower values indicate higher task priorities.

- AUTOSAR OS elements
- AUTOSAR OS resources

This property displays a list of all AUTOSAR OS resources referenced by the AUTOSAR OS task.

- AUTOSAR OS events

This property displays a list of all AUTOSAR OS events referenced by the AUTOSAR OS task.

Most of the above properties provided by the V-ECU implementation are read-only, because the V-ECU implementation code depends on these properties. This means that, later changes would lead to unexpected behavior. The DAQ raster is named automatically when you add the V-ECU implementation to your ConfigurationDesk application. You can change the DAQ raster name, or you can switch off the DAQ raster for a task by deleting the name in the DAQ raster name edit field.

#### Note

DAQ raster names must be unique within an application process. Otherwise, a conflict is shown in the Conflicts Viewer.

#### AUTOSAR OS tasks created by ConfigurationDesk for CAN controllers

A CAN controller task forwards CAN frames from/to a CAN board connected to a SCALEXIO system. You can specify the DAQ raster names for CAN controller tasks created by ConfigurationDesk, and also prioritize them

against the other tasks of the V-ECU implementation according to the AUTOSAR schematic.

#### Note

If you use RTE interventions without task references, you might find a task named Dap\_TestPoint\_Task in your ConfigurationDesk application after you add the V-ECU implementation. If this happens, the following limitation applies:

RTE intervention ports without task references must not be connected to I/O function ports. Otherwise, the build process is aborted.

---

## Events of V-ECU implementations

When working with V-ECU implementations, you need to distinguish between events in the ConfigurationDesk context and AUTOSAR OS events:

**ConfigurationDesk events** The following event types are available in ConfigurationDesk:

- Timer events
- I/O events
- Software events

A combination of an AUTOSAR OS counter and an AUTOSAR OS alarm that are configured to trigger a task is represented as a software event in ConfigurationDesk. For more information on events in ConfigurationDesk, refer to [Basics on Tasks, Events, and Runnable Functions](#) on page 480.

**AUTOSAR OS events** AUTOSAR OS events can release an AUTOSAR extended task from a waiting state when they are assigned to that task. For details refer to the AUTOSAR documentation.

---

## V-ECUs and environment model application processes: Task scheduling

In real-time simulation on SCALEXIO and a MicroAutoBox III, the clocks of V-ECUs and environment model application processes are not exactly synchronized. The operating system of the dSPACE real-time hardware might start the execution of V-ECU tasks and environment model tasks at slightly different times, depending on the initialization time.

As a consequence, the periodic tasks of V-ECUs and environment model application processes are not triggered synchronously. This effect is similar to real ECUs, which have their own local clocks. The resulting effects can be observed in a ControlDesk experiment when plotting signals from different V-ECUs and/or environment model application processes with a common time axis.

## Delaying the Start of V-ECU Implementations

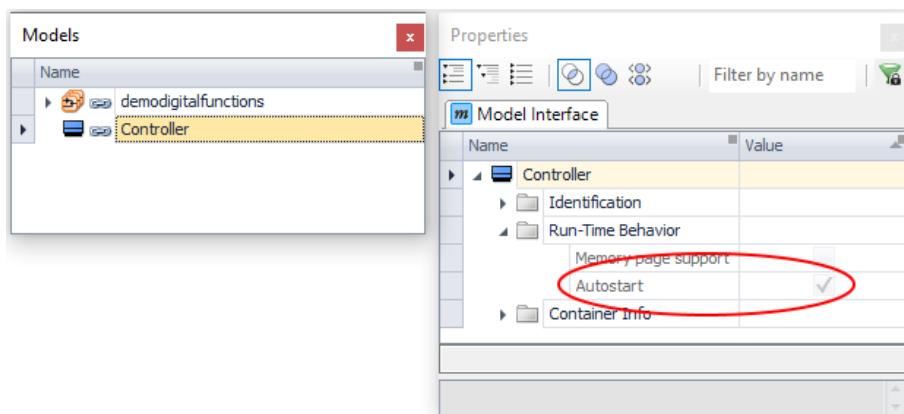
### Introduction

ConfigurationDesk lets you build real-time applications in which the start of the contained V-ECU implementations can be delayed.

### Autostart property of V-ECU implementations

V-ECU implementations are configured and exported with SystemDesk or TargetLink. In SystemDesk, you can configure the Autostart property of a V-ECU implementation to be enabled or disabled. If the Autostart property is enabled, the V-ECU implementation starts immediately when the real-time application starts. If the Autostart property is disabled, the V-ECU implementation waits when the real-time application starts. For details, refer to [Specific Variables for Simulation \(VEOS Manual\)](#).

In ConfigurationDesk, the Autostart property of a V-ECU implementation is displayed in the V-ECU implementation's Properties Browser. This property is read-only.



### Note

- If the Autostart property of a V-ECU implementation is not specified in SystemDesk, ConfigurationDesk enables the property.
  - The A2L file generated by ConfigurationDesk contains the following variables for delaying the startup of a V-ECU implementation:
    - `Sab_PowerOn` to enable (1 = ON) or disable (0 = OFF) KL30 (power)
    - `Sab_Ignition` to enable (1 = ON) or disable (0 = OFF) KL15 (ignition)
 You can set these variables in ControlDesk or using the XIL API.
- If Autostart is disabled, then the startup function of the V-ECU implementation is not called until both A2L variables `Sab_PowerOn` and `Sab_Ignition` are set to 1. Once the startup function is called and the V-ECU implementation is running, later changes of `Sab_PowerOn` and `Sab_Ignition` are ignored.

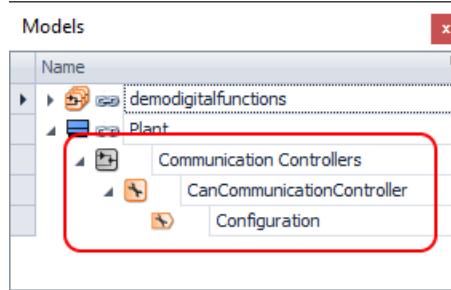
## Special Aspects of V-ECU Implementations Containing CAN Controllers

### Introduction

A V-ECU implementation can contain one or more CAN controllers. This lets you simulate the CAN bus communication of a V-ECU implementation on a real CAN bus.

### CAN controllers in V-ECU implementations

If you add a V-ECU implementation to your ConfigurationDesk application, the contained CAN controllers are displayed as Configuration Port blocks in the Model Browser.



You can drag the Configuration Port blocks to your working view and map the configuration ports to the Configuration port of a CAN function block.

#### Note

- If you do not map a configuration port of a Configuration Port block to a Configuration port of a CAN function block, the related CAN communication is not implemented in the real-time application when you start the build process.
- Properties for the SCALEXIO CAN channel which are not provided by the CAN controllers of the V-ECU implementation must be configured in the CAN function block. Termination, Transceiver type, and Feedthrough mode are examples.

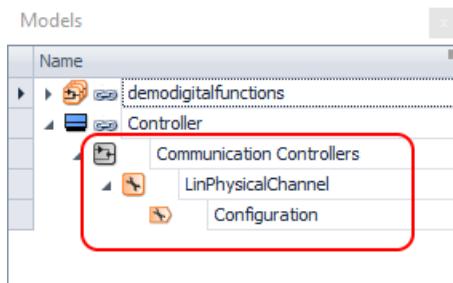
## Special Aspects of V-ECU Implementations Containing LIN Controllers

### Introduction

A V-ECU implementation can contain one or more LIN controllers. This lets you simulate the LIN bus communication of a V-ECU implementation on a real LIN bus.

### LIN controllers in V-ECU implementations

If you add a V-ECU implementation to your ConfigurationDesk application, the contained LIN controllers are displayed as Configuration Port blocks in the Model Browser.



You can drag the Configuration Port blocks to your working view and map the configuration ports to the Configuration port of a LIN function block.

#### Note

- If you do not map a configuration port of a Configuration Port block to a Configuration port of a LIN function block, the related LIN communication is not implemented in the real-time application when you start the build process.
- The following limitations apply to ConfigurationDesk's LIN communication support:
  - The LIN transport protocol is not supported.
  - LIN node configuration services are not supported.

### Specific A2L variables for switching LIN schedule tables

The A2L file of a V-ECU implementation configured as a LIN master for a specific channel contains the following variables that let you switch LIN schedule tables:

A2L Variable	Purpose
NetMgmt_<LinChannel>_ActiveScheduleTable	To display the currently active LIN schedule table.
NetMgmt_<LinChannel>_ChannelId	To display the ID of the selected LIN channel.
NetMgmt_<LinChannel>_ChannelState	To display the state of the selected LIN channel.
NetMgmt_<LinChannel>_SelectedScheduleTable	To specify the LIN schedule table.

#### Note

The LIN schedule table can be switched only before the startup of a V-ECU implementation, not during run time. Refer to [Delaying the Start of V-ECU Implementations](#) on page 638.

A2L Variable	Purpose
NetMgmt_<LinChannel>_SetBusAsleep	To set the LIN bus to sleep mode when the ECU is in sleep mode.  <div style="background-color: #f0f0f0; padding: 5px;"><b>Note</b> This is currently only supported for VEOS.</div>
NetMgmt_<LinChannel>_SwitchScheduleTableOnBusSleep	To switch to the next LIN schedule table before the LIN bus is set to sleep mode.  <div style="background-color: #f0f0f0; padding: 5px;"><b>Note</b> This is currently only supported for VEOS.</div>

You can set these variables in ControlDesk, for example.

## Details on the Build Process for Real-Time Applications Containing V-ECU Implementations

### Introduction

After you have modeled your executable application containing V-ECU implementations, you can configure and start the build process for your real-time application.

### Where to go from here

### Information in this section

Configuring the Build Process for ConfigurationDesk Applications Containing V-ECU Implementations.....	642
Build Result Files of ConfigurationDesk Applications Containing V-ECU Implementations.....	644
Working with V-ECU Implementations That Provide Memory Segments.....	646
Working with V-ECU Implementations That Provide Virtual Bypassing Support.....	647

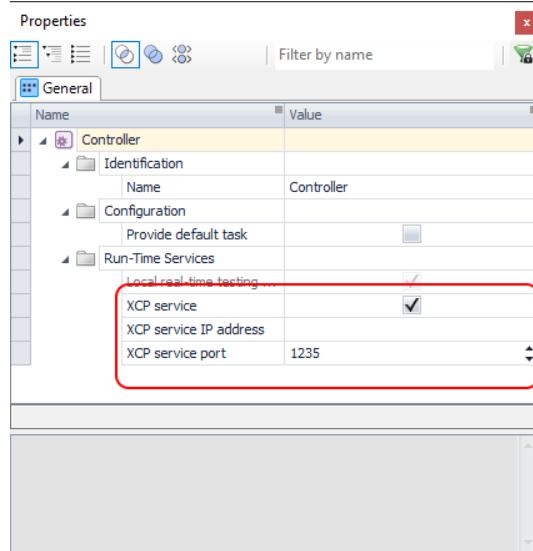
## Configuring the Build Process for ConfigurationDesk Applications Containing V-ECU Implementations

### Introduction

For executable applications containing V-ECU implementations, you must perform specific configurations for the build process.

### Configuring the XCP service

ConfigurationDesk provides properties that let you configure the XCP service for an application process to which a V-ECU implementation is assigned. Refer to the following illustration:



**XCP service** The XCP service checkbox lets you deactivate the XCP service for the application process to which the V-ECU implementation is assigned. If you clear the checkbox, no CFD\_I\_XCP license is required to use the V-ECU implementation in ConfigurationDesk. In this case, no A2L file and, if applicable, no HEX file is created for this application process during the build process.

**XCP service IP address** The XCP service IP address lets you specify the IP address of the XCP service in IPv4 notation: Each number of the local IP address must be in the range 0 ... 255, e.g., 192.168.140.6.

**XCP service port** If you want to configure the XCP service port number you have to specify it in the XCP service port property of the related application process. The value of the XCP port number must be in the range 1024 ... 65535. The default value is an integer value  $\geq 1024$ .

#### Note

The XCP service port number must be unique in your executable application. Otherwise, a conflict is shown in the Conflicts Viewer. If this happens, the build process is aborted with an error message.

After you configure the build process, you can start it. For details on the build results, refer to [Build Result Files of ConfigurationDesk Applications Containing V-ECU Implementations](#) on page 644.

#### Build settings for CTLGZ files containing TargetLink code

If your V-ECU implementation is based on a CTLGZ file and contains TargetLink code, you have to link the DSFXP library to your ConfigurationDesk application. The DSFXP library is exported from TargetLink as the **DsFxpLibScalexio.zip** file. It contains utility functions and macros used by TargetLink in the generated code.

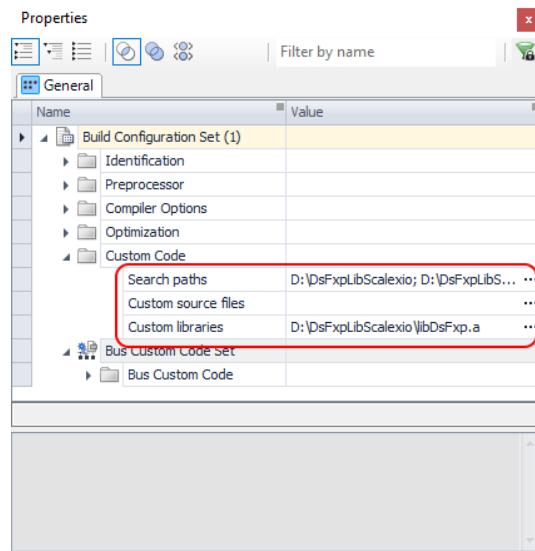
##### Note

- The following instructions apply only to V-ECU implementations that are based on CTLGZ files. In V-ECU implementations based on a VECU file, the DSFXP library is already included.
- The DSFXP library must be exported from the same TargetLink version that was used to generate the application code in the V-ECU implementation. If the V-ECU implementation contains components from different TargetLink versions, then the highest version must be used.

To link the DSFXP library to your ConfigurationDesk application, you have to perform the following steps:

1. Copy **DsFxpLibScalexio.zip** to your PC.
2. Extract the **DsFxpLibScalexio.zip** file to a user-defined folder (called DSFXP include folder below), for example, **D:\DsFxpLibScalexio**.
3. From that folder, run **build.bat** using the Command Prompt for dSPACE RCP and HIL <release number>, which is available in the Windows Start menu.
4. In the Build Configuration table in ConfigurationDesk, you have to reference the DSFXP include folder and its **SCALEXIO** subfolder. To do so, select the build configuration set to which the application process that is relevant for the V-ECU implementation is assigned.
5. In the Properties Browser, click the Search path browse button to open the Manage Search Paths dialog.
6. In the Manage Search Paths dialog, click  to create a new custom library entry and select the DSFXP include folder via the the Browse button.
7. Repeat step 6 to select the **SCALEXIO** subfolder of the DSFXP include folder.
8. Click the Custom libraries browse button to open the Manage Custom Libraries dialog.
9. In the Manage Custom Libraries dialog, click  to create a new search path entry and select the **libDsFxp.a** file via the the Browse button.

The following illustration shows an example configuration:



## Build Result Files of ConfigurationDesk Applications Containing V-ECU Implementations

### Files generated during the build process

The following files are generated during the build process:

File type	Description
Real-time application file (RTA file)	The executable object file for processor boards. After the build process, the RTA file can be downloaded to the real-time hardware.
System description file (SDF file)	Provides information on the real-time application that is needed by the experiment software, such as information on the variable description file(s) associated to the real-time application. The SDF file references the related RTA file. If you select to download an SDF file to the real-time hardware, the related RTA file is downloaded.
Variable description file (TRC file) (generated for application processes that do not have an assigned V-ECU implementation)	Contains the descriptions of all the variables (signals and parameters) which can be accessed via the experiment software. In multicore applications, one TRC file is generated for each application process.
Map file (MAP file) (generated for application processes that do not have	Maps symbolic names to physical addresses. It is used only by the experiment software. In

File type	Description
an assigned V-ECU implementation)	multicore applications, one MAP file is generated for each application process.
CANCFG file (generated only if your ConfigurationDesk application contains Simulink models with blocks from the RTI CAN MultiMessage Blockset)	Contains configuration data (in XML format) for CAN communication.
LINCFG file (generated only if your ConfigurationDesk application contains Simulink models with blocks from the RTI LIN MultiMessage Blockset)	Contains configuration data (in XML format) for LIN communication.
FLXCFG file (generated only if your ConfigurationDesk application contains Simulink models with blocks from the FlexRay Configuration Blockset)	Contains configuration data (in XML format) for FlexRay communication.
EXPSWCFG file	Contains configuration data (in XML format) for automotive fieldbus communication.
A2L file (only for application processes that have a V-ECU implementation or a Simulink model/SIC file with Variable description file type = A2L assigned)	<p>The A2L file contains</p> <ul style="list-style-type: none"> <li>▪ Model-specific variables provided by the A2L fragment of the related Simulink model, SIC file or V-ECU implementation container</li> <li>▪ Variables for I/O functions and test automation</li> <li>▪ DAQ raster names and the XCP service port number</li> <li>▪ Information required to perform virtual bypassing via the RTI Bypass Blockset (only for application processes that contain a V-ECU implementation configured for virtual bypassing)</li> </ul> <p>The A2L file is named after the related application process. You can use these A2L files to integrate each model implementation separately as <i>XCP on Ethernet device</i> in ControlDesk.</p>
HEX file (initial data file, generated only for V-ECU implementations that support memory pages)	The HEX file contains information about the initial state of calibration variables.

**Note**

Application processes containing V-ECU implementations are not generated into the SDF file. If your ConfigurationDesk application contains only application processes that have assigned V-ECU implementations, no SDF file is generated. If this happens, you must download the RTA file instead.

**Related topics****Basics**

[Build Result Files of the Build Process.....](#) 702

## Working with V-ECU Implementations That Provide Memory Segments

### V-ECU implementations with memory pages in ConfigurationDesk

A V-ECU implementation can provide memory segments. For application processes to which this V-ECU implementation is assigned, ConfigurationDesk creates an A2L file with a description of the existing memory segments during the build process. Each memory segment can have one or more pages. However, ConfigurationDesk only supports exactly one memory page per segment. The memory segment sections contain all variables with write access (= CHARACTERISTICS). You can access these variables with experiment software such as ControlDesk.

### Example of a MEMORY\_SEGMENT

In the A2L file, the memory segment sections are marked with MEMORY\_SEGMENT. Refer to the following example:

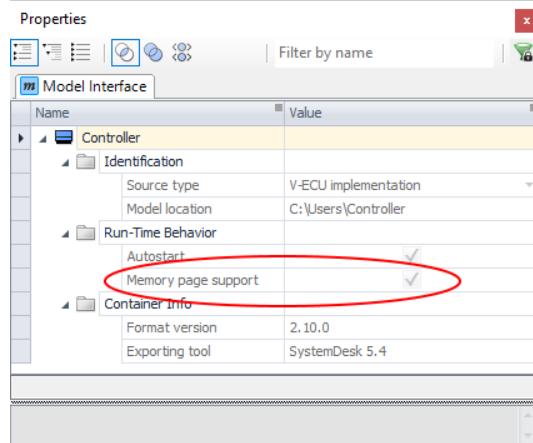
```
/begin MEMORY_SEGMENT
    /* Name */          CALIB
    /* LongIdentifier */ "Memory segments for calibration"
    /* PrgType */        DATA
    /* MemoryType */     FLASH
    /* Attribute */      EXTERN
    /* Address */        0x0805CC40
    /* Size in Byte */   0x00000AF2
    /* Offset */         -1 -1 -1 -1
    ...
/end MEMORY_SEGMENT
```

**Note**

Variables for test automation and for I/O functions are not created in the memory page entries.

### Display of memory page support in ConfigurationDesk

In ConfigurationDesk, you can see whether a V-ECU implementation offers memory page support by checking the Memory page support property of a V-ECU implementation. Refer to the following illustration:



### Initial data file

During the build process, an initial data file is created for the application process to which a V-ECU implementation with memory page support is assigned. The initial data file is created in the INTEL HEX file format. This file contains a separate section for each memory segment defined in the V-ECU implementation. Each section contains the memory image of the initial states of all CHARACTERISTIC variables of the respective memory segment. An additional section contains an EPK identifier for consistency checks of the V-ECU implementation. The EPK identifier changes with each build process.

The initial data file enables the experiment software to set the experiment into a well-defined state after multiple variables might have been modified. The file is named after the application process and has the HEX extension, for example, **controller.hex**. After the build process, the file is displayed in the **Build Results** folder in the Project Manager.

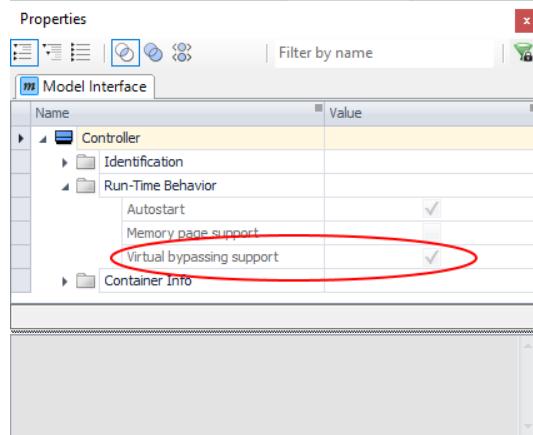
## Working with V-ECU Implementations That Provide Virtual Bypassing Support

### Introduction

ConfigurationDesk lets you use V-ECU implementations that support virtual bypassing. Virtual bypassing allows you to bypass variables or functions of the V-ECU implementation while the real-time application is running on the real-time hardware. You can bypass a variable by replacing the value of the variable with another value. A function can be bypassed by executing another function instead of the bypassed function.

### Display of virtual bypassing support in ConfigurationDesk

In ConfigurationDesk, you can see whether a V-ECU implementation supports virtual bypassing by checking the Virtual bypassing support property of a V-ECU implementation. Refer to the following illustration:



### A2L file entries for virtual bypassing

If a V-ECU implementation supports virtual bypassing, ConfigurationDesk creates additional information in the A2L file during the build process. This information is required by the RTI Bypass Blockset. The following additional entries are created in the A2L file:

- Target processor
- Target architecture
- A2L address extension
- Internal ID of the application process that contains the V-ECU implementation
- Information on the events provided by the V-ECU implementation
- A list of functions that can be bypassed including the return data type and a list of arguments and their data types

### Related topics

#### Basics

Dynamically Integrating a Simulink or TargetLink Model into a V-ECU Running on a SCALEXIO System or on a MicroAutoBox III (RTI Bypass Blockset Reference

# ECU Interfacing with SCALEXIO or MicroAutoBox III Systems

## Introduction

In ECU interfacing scenarios, you can access specific parts of an ECU application while the application is being executed on the ECU. To perform ECU interfacing with SCALEXIO or MicroAutoBox III systems, you must import ECU interface container (EIC) files to a ConfigurationDesk application. EIC files are generated with the ECU Interface Manager and describe an ECU application that is prepared for ECU interfacing. In ConfigurationDesk, you can integrate the prepared parts of the ECU application in the signal chain and build a real-time application for the SCALEXIO or MicroAutoBox III system.

## Where to go from here

### Information in this section

Introduction to ECU Interfacing with SCALEXIO or MicroAutoBox III Systems.....	650
Implementing ECU Interfacing in ConfigurationDesk.....	656
Updating ECU Interface Containers in ConfigurationDesk Applications.....	671

# Introduction to ECU Interfacing with SCALEXIO or MicroAutoBox III Systems

## Where to go from here

## Information in this section

[Introduction to ECU Interfacing](#)..... 650

[Basics on ECU Interfacing with SCALEXIO or MicroAutoBox III Systems](#)..... 652

## Introduction to ECU Interfacing

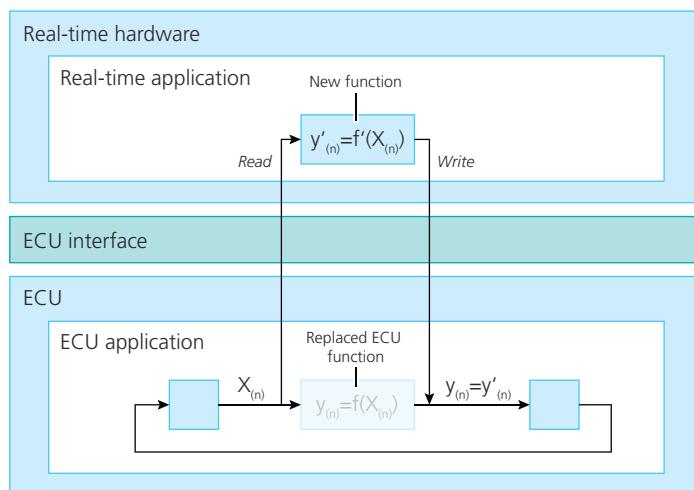
### Scope of ECU interfacing

ECU interfacing comprises various methods and tools to access specific parts of an ECU application for development and testing purposes while the application is executed on the ECU.

### Use scenarios for ECU interfacing

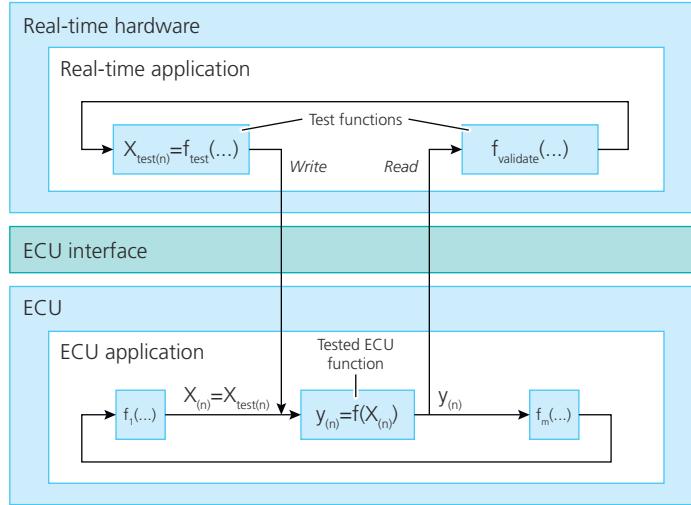
In ECU interfacing scenarios, you can access complete ECU functions or individual ECU variables to perform tasks such as replacing existing ECU functions with new functions (bypassing), testing existing ECU functions (white-box testing), or accessing the data of individual ECU variables.

**Replacing existing ECU functions with new functions** Replacing existing ECU functions with new functions (bypassing) lets you implement new or enhanced functions in the context of an existing ECU environment (i.e., an ECU including its I/O). The following illustration shows an example of this use scenario.



In the example above, the new function is part of a real-time application that is executed on real-time hardware. The ECU function to be replaced is part of an ECU application that is executed on the ECU. Both functions (i.e., the new function and the ECU function to be replaced) are executed synchronously. The real-time application reads the inputs of the ECU function, and overwrites its outputs with the outputs of the new function.

**Testing existing ECU functions** You can test existing ECU functions by replacing their input values and validating their output values (white-box testing). The following illustration shows an example of this use scenario.



In the example above, a real-time application is executed on real-time hardware and contains one test function to stimulate the inputs of an ECU function and one test function to validate the outputs of the ECU function. The ECU function to be tested is part of an ECU application that is executed on the ECU. The real-time application overwrites the inputs of the tested ECU function with stimulation values, and reads the outputs of the tested ECU function for validation.

**Accessing the data of individual ECU variables** Accessing the data of individual ECU variables lets you read and/or write variable values without accessing the related ECU functions. Via the real-time application, you can directly read or write variable values from or to an ECU variable independently from the ECU function algorithms. Depending on your requirements, you can read and write variable values synchronously or asynchronously to the execution of the ECU application.

## Further information

For more basic information on ECU interfacing (e.g., further use scenarios, methods, and tools), refer to [ECU Interfacing Overview](#).

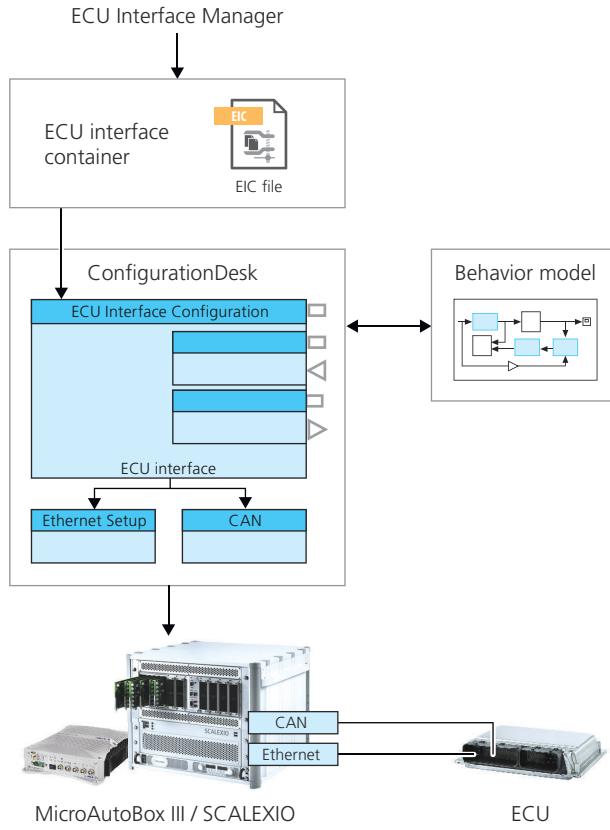
## Basics on ECU Interfacing with SCALEXIO or MicroAutoBox III Systems

### Introduction

You can use SCALEXIO or MicroAutoBox III systems to perform external ECU interfacing. For this purpose, the target ECU must be connected to a SCALEXIO simulator or a MicroAutoBox III and ECU interfacing must be implemented in ConfigurationDesk.

### Overview

The following illustration provides an overview of ECU interfacing with SCALEXIO or MicroAutoBox III systems.



To implement ECU interfacing in ConfigurationDesk, you need an ECU interface container (EIC) file. When you import an EIC file to an ECU Interface Configuration function block, the function block lets you integrate parts of an ECU application that are prepared for ECU interfacing in the [signal chain](#). Via an Ethernet Setup or CAN function block, you can configure the interface for the communication between SCALEXIO/MicroAutoBox III and the ECU.

### Basics on ECU interface container (EIC) files

An ECU interface container (EIC) file is a container file format that is generated with the ECU Interface Manager. The ECU Interface Manager is dSPACE software that lets you prepare ECU applications for ECU interfacing.

An EIC file describes an ECU application that is prepared for ECU interfacing. An EIC file can:

- Provide functions and variables of the ECU application that are prepared to be accessed by the real-time application.
- Provide the data direction (read or write) for accessing the input and/or output values of the ECU functions or ECU variables.
- Provide events that can trigger tasks after data was read from or written to an ECU function or ECU variable, for example.
- Provide specifications that let you enable or disable the read and/or write access of an ECU function or ECU variable, and provide status information on the access operations.
- Provide time stamps for ECU-synchronous read operations of the real-time application, e.g., when reading data of an ECU function.

Time stamps can be provided only if the ECU interface that connects the target ECU to SCALEXIO/MicroAutoBox III is based on an XCP service that supports time-stamping.

Additionally, the EIC file contains information on the ECU interface that is used to connect the target ECU to the SCALEXIO or MicroAutoBox III system. The EIC file provides the following information:

- The requirements for exchanging data via the ECU interface.
- The specifications that let you enable and disable the interface.
- The specifications that let you handle ECU calibration pages.

These specifications are available only if ECU calibration page handling is prepared with the ECU Interface Manager. ECU calibration pages can provide different parameter sets to the ECU application. If the specifications are available, you can switch the ECU calibration page that is accessed by the ECU application, for example. For more information, refer to [Preparing ECU Calibration Page Handling \(ECU Interface Manager Manual\)](#).

#### **Integrating parts of an ECU application prepared for ECU interfacing in the signal chain**

When you import an EIC file to an ECU Interface Configuration function block, the definitions of the EIC file are added to the function block and determine the function ports and properties that are available for the function block:

- The function ports let you exchange data between the ECU application that is executed on the target ECU and a behavior model (e.g., that contains the algorithms of new functions) that is part of the real-time application. Via the function ports, you can provide variable values of an ECU function to the behavior model, overwrite variable values of the ECU function with values from the behavior model, etc.
- The event ports of the function block let you map [Runnable Function blocks](#) to the available [I/O events](#).
- The properties of the function block let you reference function blocks that set up the connection between the target ECU and SCALEXIO/MicroAutoBox III, assign [tasks](#) to I/O events, etc.

For more information on the ECU Interface Configuration function block, refer to [ECU Interface Configuration \(ConfigurationDesk I/O Function Implementation Guide\)](#).

#### Exchanging data between the target ECU and SCALEXIO/MicroAutoBox III

To exchange data for ECU interfacing, the target ECU and SCALEXIO/MicroAutoBox III must be connected via Ethernet or CAN. To set up the Ethernet or CAN interface, you must specify a SCALEXIO/MicroAutoBox III hardware resource in an Ethernet Setup or CAN function block and reference this function block in the ECU Interface Configuration function block.

##### Note

To set up an Ethernet interface in ECU interfacing scenarios, you must use an Ethernet Setup function block. You cannot use custom Ethernet function blocks for this purpose.

**Supported ECU interfaces** To perform ECU interfacing with SCALEXIO or MicroAutoBox III, the target ECU must be connected to the SCALEXIO or MicroAutoBox III system via one of the following ECU interfaces:

- DCI-GS12 (only Ethernet)
- XCP on Ethernet
- XCP on CAN

For more information on connecting the ECU via these interfaces to a SCALEXIO or MicroAutoBox III system, refer to [Basics on Connecting the ECU \(ECU Interfacing Overview\)](#).

**Accessing ECU interfaces via function blocks** You can access Ethernet and CAN ECU interfaces via Ethernet Setup and CAN function blocks, respectively:

- The Ethernet Setup function block lets you initialize and configure the Ethernet controller of your SCALEXIO or MicroAutoBox III system to set up the connection to an Ethernet ECU interface. You can specify a local IP address, optimize the Ethernet controller, etc.

For more information on the Ethernet Setup function block, refer to [Ethernet Setup \(ConfigurationDesk I/O Function Implementation Guide\)](#).

- The CAN function block lets you access different bus channel types of the SCALEXIO or MicroAutoBox III system.

Depending on the channel type, you can configure various bus-related settings, such as the baud rate, CAN FD support, or partial networking.

- For more information on specifying a CAN ECU interface, refer to [Notes on Specifying CAN ECU Interfaces](#) on page 667
- For more information on the CAN function block and the characteristics of the supported bus channel types, refer to [CAN \(ConfigurationDesk I/O Function Implementation Guide\)](#)

#### Workflow for implementing ECU interfacing in ConfigurationDesk

For an overview of a typical workflow for implementing ECU interfacing in ConfigurationDesk, refer to [Typical Workflow for Implementing ECU Interfacing in ConfigurationDesk](#) on page 656.

**Updating the EIC file in an ECU Interface Configuration function block**

If you imported an EIC file to an ECU Interface Configuration function block and a new version of this file is available, you can update the EIC file in the function block. Refer to [Updating ECU Interface Containers in ConfigurationDesk Applications](#) on page 671.

---

**Limitations**

When you use SCALEXIO or MicroAutoBox III systems to perform ECU interfacing, some limitations apply. Refer to [Limitations Concerning ECU Interfacing with SCALEXIO or MicroAutoBox III](#) on page 779.

---

**Related topics**

**Basics**

[Compatibility of ConfigurationDesk 6.7](#).....36

# Implementing ECU Interfacing in ConfigurationDesk

Where to go from here	Information in this section
	<p>Typical Workflow for Implementing ECU Interfacing in ConfigurationDesk..... 656</p> <p>How to Import ECU Interface Container (EIC) Files..... 658</p> <p>Workflow for Modeling the Triggering Behavior in ECU Interfacing Scenarios via MATLAB/Simulink..... 659</p> <p>Examples for Modeling the Triggering Behavior in ECU Interfacing Scenarios via MATLAB/Simulink..... 662</p> <p>Notes on Model Interface Elements Generated for ECU Interface Configuration Function Blocks..... 667</p> <p>Notes on Specifying CAN ECU Interfaces..... 667</p> <p>Preconfigured Preprocessor Macros for ECU Interfacing..... 669</p>

## Typical Workflow for Implementing ECU Interfacing in ConfigurationDesk

<b>Overview</b>	To implement ECU interfacing in ConfigurationDesk, you need an ECU interface container (EIC) file that contains information on the ECU functions and/or ECU variables that are to be accessed by the real-time application. Additionally, you need a behavior model (e.g., that contains algorithms and functions to test or bypass the accessible ECU functions).
<b>Workflow</b>	<p>Implementing ECU interfacing in ConfigurationDesk comprises the following typical workflow steps:</p> <ol style="list-style-type: none"> <li>1. Import an ECU interface container (EIC) file to ConfigurationDesk. Refer to <a href="#">How to Import ECU Interface Container (EIC) Files</a> on page 658.</li> <li>2. Depending on the ECU interface (Ethernet or CAN), instantiate an Ethernet Setup or CAN function block</li> <li>3. Select the ECU Interface Configuration function block and assign the Ethernet Setup or CAN function block as the Referencing Configuration in the Properties Browser.</li> <li>4. Configure the Ethernet Setup or CAN function block: <ul style="list-style-type: none"> <li>▪ Assign a suitable hardware resource.</li> <li>▪ Specify Ethernet- or CAN-related settings to access the target ECU.</li> </ul> </li> </ol>

In case of a CAN ECU interface, the EIC file might provide requirements regarding the CAN bus. For more information, refer to [Notes on Specifying CAN ECU Interfaces](#) on page 667.

For more information on the configurable settings of the function blocks, refer to [Ethernet Setup \(ConfigurationDesk I/O Function Implementation Guide\)](#) or [CAN \(ConfigurationDesk I/O Function Implementation Guide\)](#), respectively.

5. Add a suitable behavior model to the ConfigurationDesk application and map the ECU Interface Configuration function block to the model.

If you do not have a suitable behavior model yet, ConfigurationDesk can support you by generating the required model interface. For more information, refer to [Specifying the Model Interface](#) on page 417.

#### Note

Especially in bypassing scenarios, functions of the behavior model must be executed synchronously to the related functions of the ECU application. You must model this behavior manually in the behavior model. For more information, refer to [Workflow for Modeling the Triggering Behavior in ECU Interfacing Scenarios via MATLAB/Simulink](#) on page 659

6. If required, configure the available tasks.

Refer to [How to Configure Tasks in ConfigurationDesk](#) on page 515.

7. Complete the signal chain according to your requirements.

8. Configure and start the build process.

Refer to [Building Real-Time Applications](#) on page 697.

#### Tip

- When you configure the build process, you can specify preprocessor macros. For ECU interfacing, some preconfigured macros are available. Refer to [Preconfigured Preprocessor Macros for ECU Interfacing](#) on page 669.
- During the build process, ConfigurationDesk verifies the imported EIC file. If ConfigurationDesk finds an EIC file with an identical name but different generation time at the location from where you imported the EIC file, the Message Viewer displays a warning message. If required, you can update the EIC file in the ConfigurationDesk application. Refer to [Updating ECU Interface Containers in ConfigurationDesk Applications](#) on page 671.

**Related topics****Basics**

- [Timing Behavior in Service-Based ECU Interfacing Scenarios \(ECU Interfacing Overview\)](#)
- [Timing Behavior in Serviceless ECU Interfacing Scenarios \(ECU Interfacing Overview\)](#)

## How to Import ECU Interface Container (EIC) Files

**Objective**

To implement ECU interfacing in ConfigurationDesk, you must import ECU interface container (EIC) files to a ConfigurationDesk application.

**Preconditions**

- A ConfigurationDesk project with an application is open.
- An ECU interface container (EIC) file is available.  
Within one ConfigurationDesk application, you can import each EIC file only once. The EIC file version must be compatible to your ConfigurationDesk version. Refer to [Compatibility of ConfigurationDesk 6.7](#) on page 36.
- A working view is open.

**Method****To import ECU interface container (EIC) files**

- 1 From the Function Browser, drag an ECU Interface Configuration function block to the working view.  
ConfigurationDesk adds the instantiated ECU Interface Configuration function block to the signal chain.
- 2 Right-click the ECU Interface Configuration function block and select Import ECU Interface Container from the context menu.  
An Open dialog opens.
- 3 In the dialog, navigate to the EIC file you want to add and click Open.  
The EIC file is imported to the ECU Interface Configuration function block. The function block displays function ports, event ports, and the related structures hierarchically. The Properties Browser provides access to properties that let you assign tasks to I/O events etc.
- 4 Repeat steps 1 to 3 to import further EIC files.  
You can import exactly one EIC file to one ECU Interface Configuration function block.

**Result**

You imported EIC files to ConfigurationDesk. The imported EIC files are copied to the file system of ConfigurationDesk. These copies are used in the ConfigurationDesk application.

The functions and ports that are available for the ECU Interface Configuration function blocks depend on the specifications in the EIC files. For an overview of the functions and ports that can be available, refer to [Overview of Ports and Basic Properties \(ECU Interface Configuration\) \(ConfigurationDesk I/O Function Implementation Guide\)](#).

#### Next step

You can now continue to implement ECU interfacing in ConfigurationDesk. For an overview of a typical workflow, refer to [Typical Workflow for Implementing ECU Interfacing in ConfigurationDesk](#) on page 656.

## Workflow for Modeling the Triggering Behavior in ECU Interfacing Scenarios via MATLAB/Simulink

#### Overview

In ECU interfacing scenarios, the ECU application can provide [events](#). You can use these events to trigger functions in a behavior model. This is particularly important if functions of the behavior model must be executed synchronously to functions of the ECU application (e.g., as required in most bypassing scenarios).

You can access the events of an ECU application only if the access is configured in the EIC file that you import to an ECU Interface Configuration function block. If you can access the events of an ECU application, you must model the triggering behavior in the behavior model. You can model the triggering behavior separately for the following elements:

- Each ECU function that can be accessed by the ECU Interface Configuration function block
- Each data access that groups ECU variables that can be accessed by the ECU Interface Configuration function block

For an overview of the representation of ECU functions and data accesses in ECU Interface Configuration function blocks, refer to [Overview of Ports and Basic Properties \(ECU Interface Configuration\) \(ConfigurationDesk I/O Function Implementation Guide\)](#).

#### Modeling the triggering behavior in MATLAB/Simulink

If you work with a MATLAB/Simulink behavior model, you can model the triggering behavior via the following blocks and subsystems:

- Input and/or Output model port block of an ECU function
- Read and/or Write model port block of a data access
- Hardware-Triggered Runnable Function block of the AfterRead and/or AfterWrite event of an ECU function or data access
- Function-call subsystems

To model the triggering behavior, you must map the required Hardware-Triggered Runnable Function blocks to function-call subsystems and move the model port blocks to the function-call subsystems. Which Hardware-Triggered

Runnable Function blocks are required depends on your use scenario. For examples, refer to [Examples for Modeling the Triggering Behavior in ECU Interfacing Scenarios via MATLAB/Simulink](#) on page 662.

## Workflow

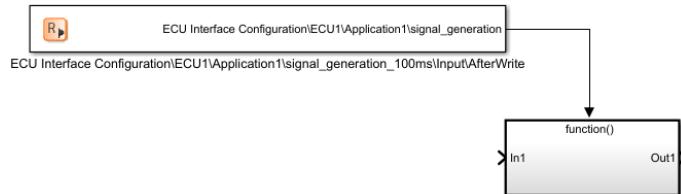
If you work with a MATLAB/Simulink behavior model, you can generate the required model port blocks and Hardware-Triggered Runnable Function blocks via ConfigurationDesk. If you do so, modeling the triggering behavior comprises the following workflow steps:

1. Select one or more ECU Interface Configuration function blocks that are available in the signal chain and generate the model port blocks and Runnable Function blocks in ConfigurationDesk and Simulink (e.g., via **Propagate - Generate New Simulink Model Interface**). Refer to [Transferring the ConfigurationDesk Model Interface to a New Simulink Interface Model](#) on page 542.
2. In the Simulink model, add one function-call subsystem for each Hardware-Triggered Runnable Function block that is required in your use scenario.

### Tip

You can easily add function-call subsystems by typing **function-call subsystem** in the Simulink Editor.

3. Map each required Hardware-Triggered Runnable Function block to the **function{}** port of one function-call subsystem.



4. Move the required model port blocks to the function-call subsystem:
  - For an ECU function, move the Input and/or Output model port blocks to the function-call subsystem that is mapped to the required AfterRead and/or AfterWrite Hardware-Triggered Runnable Function block.
  - For a data access, move the Read or Write model port block to the function-call subsystem that is mapped to the required AfterRead or AfterWrite Hardware-Triggered Runnable Function block.

**Note**

In most use scenarios, the **AfterWrite** event of a data access is generated after data of the behavior model is completely written to the Write model port block. In this case, do not move the Write model port block to the function-call subsystem that is mapped to the **AfterWrite Hardware-Triggered Runnable Function** block.

If you use this way to model the triggering behavior, data of the behavior model can be written only to the Write model port block after the **AfterWrite** event is generated. However, because the **AfterWrite** event is generated only after data has been written to the Write model port block, the event will never be generated.

Instead, move the Write model port block to a function-call subsystem that is triggered by another event, e.g., by an **AfterRead** event or a cyclic timer event.

**Tip**

You can identify the model port blocks and **Hardware-Triggered Runnable Function** blocks of an ECU function or data access by their names. For an overview of the default names, refer to [Notes on Model Interface Elements Generated for ECU Interface Configuration Function Blocks](#) on page 667.

5. Model the behavior model or copy all the blocks and subsystems with their identity to an existing behavior model. To do so, you must use the standard Copy command together with the Paste and Keep IDs command from the **Model Port Blocks** menu in the model window.
6. Make the behavior model available in the ConfigurationDesk application:
  - If the behavior model is not available in the ConfigurationDesk application yet, add it to the application. To do so, you can use the **Add This Model to ConfigurationDesk Project: <project name>** command from the ConfigurationDesk menu of the Simulink Editor, for example.
  - If the behavior model is already available in the ConfigurationDesk application, analyze the model including the task information in ConfigurationDesk. To do so, you can use the **Analyze Model in ConfigurationDesk (Including Task Information)** command from the ConfigurationDesk menu of the Simulink Editor, for example.

For more information, refer to [Remote Access to ConfigurationDesk \(Model Interface Package for Simulink - Modeling Guide](#)  ) and [Synchronizing the Model Interfaces of ConfigurationDesk and Simulink](#) on page 540, respectively.

**Related topics****Basics**

[Timing Behavior in Service-Based ECU Interfacing Scenarios \(ECU Interfacing Overview](#) 

## Examples for Modeling the Triggering Behavior in ECU Interfacing Scenarios via MATLAB/Simulink

### Overview

When you import an EIC file to an ECU Interface Configuration function block, the function block provides functions and events according to the settings of the EIC file. The available events depend on the use scenario. For example:

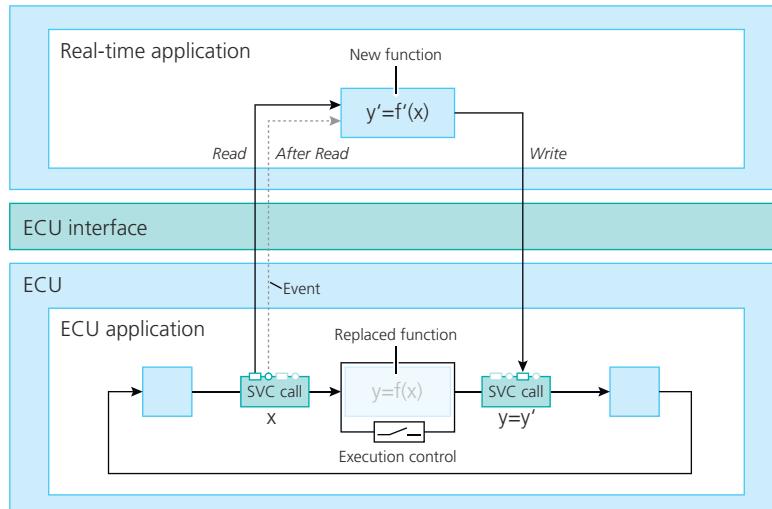
- For each ECU function that can be accessed in a bypassing scenario, one AfterRead event is available.
- For each ECU function that can be accessed in a function test scenario, one AfterRead event and one AfterWrite event are available.
- For each data access that is configured for reading data synchronously from the ECU application, one AfterRead event is available.
- For each data access that is configured for writing data synchronously to the ECU application, one AfterWrite event is available.

When you generate the model interface for an ECU Interface Configuration function block, model port blocks and Runnable Function blocks are created for the available functions and events. In a MATLAB/Simulink behavior model, you can use these blocks to model the triggering behavior in different ways.

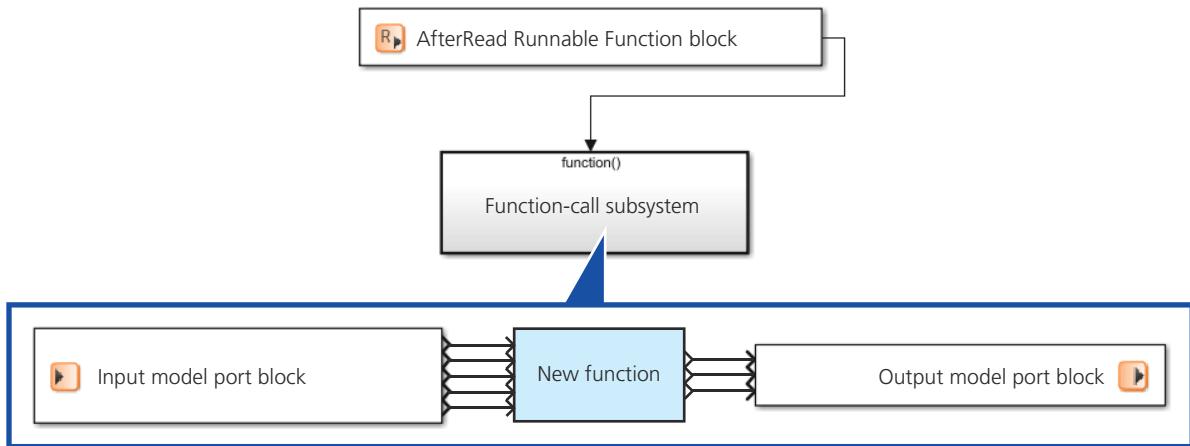
The following examples are typical ways to model the triggering behavior. For an overview of the actual workflow, refer to [Workflow for Modeling the Triggering Behavior in ECU Interfacing Scenarios via MATLAB/Simulink](#) on page 659.

### Modeling the triggering behavior for bypassing scenarios

In most bypassing scenarios, the ECU function and the related function of the behavior model are executed synchronously. The input values of the ECU function are read and used by the function of the behavior model. The calculated values of the behavior model's function are used to overwrite the output values of the ECU function:



To model this behavior in a MATLAB/Simulink behavior model, you must move the ECU function's Input and Output model port blocks to a function-call subsystem that is mapped to the ECU function's AfterRead Runnable Function block. In the subsystem, you can then map the inports and outports of the model port blocks to the related function of the behavior model:



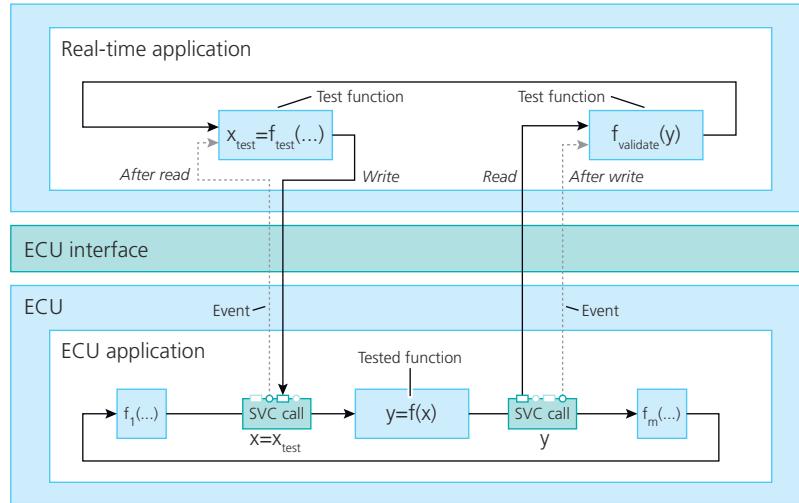
Doing this, you can achieve the following run-time behavior: The ECU function's AfterRead event is generated after all the input values are read and available at the Input model port block. This triggers the execution of the function-call subsystem. The values of the Input model port block can now be used by the function of the behavior model that replaces the ECU function. The output values of the behavior model's function are written to the Output model port block. These values overwrite the ECU function output values in the ECU application.

### **Modeling the triggering behavior for function test scenarios**

In function test scenarios, the input values of an ECU function are overwritten with values provided by a function of the behavior model. The output values of the ECU function are read and validated by a function of the behavior model. In the behavior model, the validated values can be used to calculate new input values for the ECU function.

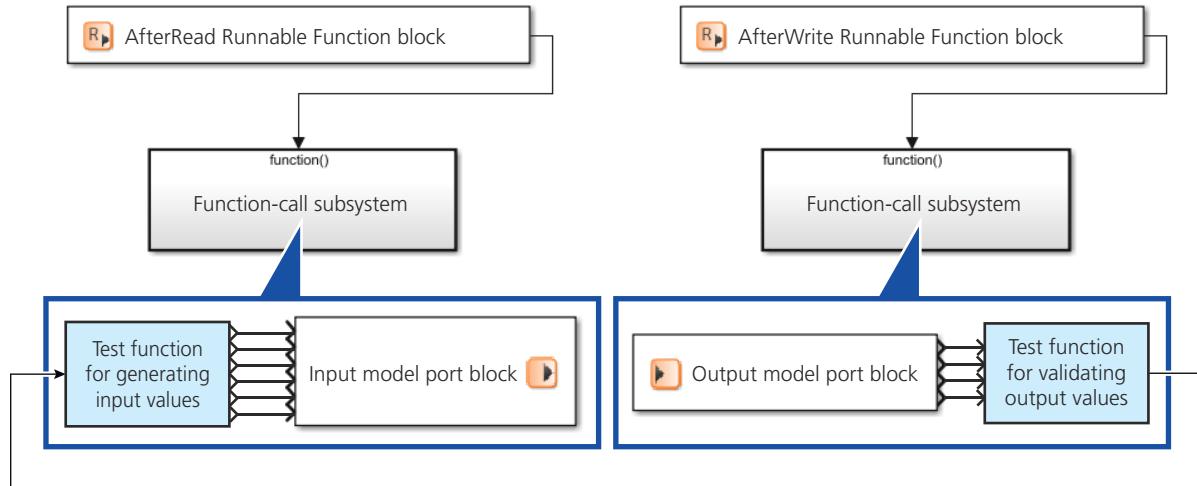
For each ECU function that can be accessed in a function test scenario, one AfterRead and one AfterWrite event are available. In your behavior model, you can model the triggering behavior by using both or only one event, as shown in the following examples.

**Testing an ECU function by using the AfterRead and AfterWrite events** You can test an ECU function by using the available AfterRead and AfterWrite events as shown in the following example.



To model this behavior in a MATLAB/Simulink behavior model, do the following:

- Move the ECU function's Input model port block to a function-call subsystem that is mapped to the ECU function's AfterRead Runnable Function block.
- Move the ECU function's Output model port block to a function-call subsystem that is mapped to the ECU function's AfterWrite Runnable Function block.
- In the subsystems, you can then map the imports and outports of the model port blocks to the related function of the behavior model.



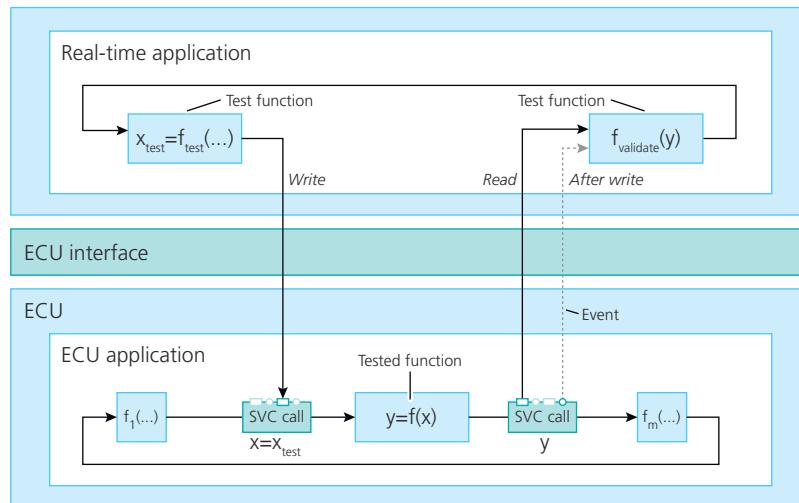
Doing this, you can achieve the following run-time behavior: The ECU function's AfterRead event triggers the execution of the function-call subsystem that is mapped to the AfterRead Runnable Function block. The values that are available at the Input model port block overwrite the input values of the ECU function in the ECU application. Then, the ECU function is executed. It uses the input values provided by the Input model port block.

The output values of the ECU function are read by the behavior model and buffered in the Output model port block. When all the output values are available at the Output model port block, the ECU function generates the AfterWrite event. This triggers the execution of the function-call subsystem that is mapped to the AfterWrite Runnable Function block. The values of the Output model port block can now be used in the behavior model for validation and calculating new input values for the ECU function.

#### Note

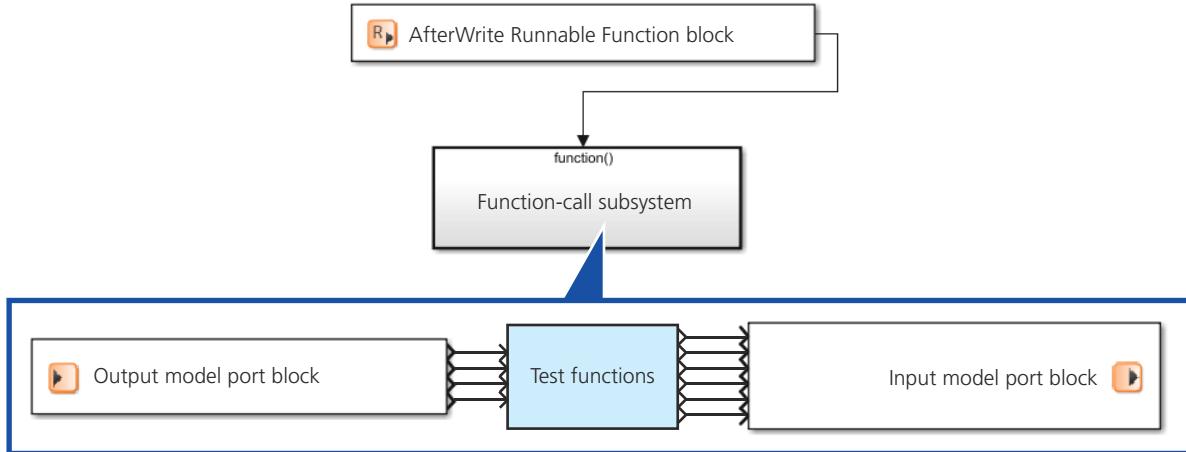
Modeling the triggering behavior this way might delay the ECU application because two events are processed by the real-time application in each sampling step.

**Testing an ECU function by using only the AfterWrite event** You can test an ECU function by using only the AfterWrite event as shown in the following example.



To model this behavior in a MATLAB/Simulink behavior model, you must move the ECU function's Input and Output model port blocks to a function-call subsystem that is mapped to the ECU function's AfterWrite Runnable Function

block. In the subsystem, you can then map the imports and outports of the model port blocks to the related functions of the behavior model:



Doing this, you can achieve the following run-time behavior: Since you do not use the ECU function's AfterRead event, it has no effect on the real-time application. In the first sampling step, the ECU function's input values are therefore not overwritten with values provided by the behavior model. However, the output values of the ECU function are read by the behavior model and buffered in the Output model port block.

When all the output values are available at the Output model port block, the ECU function generates the AfterWrite event. This triggers the execution of the function-call subsystem. The values of the Output model port block can now be used in the behavior model for validation and calculating new input values for the ECU function. The new input values are written directly to the ECU application and buffered in the ECU. In the next sampling step, the ECU function uses these values as the input values directly, without processing the AfterRead event.

#### Tip

Modeling the triggering behavior this way has only minor effects on the timing of the ECU application because only one event is processed by the real-time application in each sampling step.

## Notes on Model Interface Elements Generated for ECU Interface Configuration Function Blocks

### Generated elements

When you generate the model interface via ConfigurationDesk, the following elements can be generated for each ECU Interface Configuration function block:

- One model port block for each function of the ECU Interface Configuration function block (e.g., for Input, Output, Read, Write, and Control functions)
- One Runnable Function block for each I/O event

The available functions and I/O events depend on the EIC file you imported to an ECU Interface Configuration function block. For more information, refer to [Overview of Ports and Basic Properties \(ECU Interface Configuration\)](#) ([ConfigurationDesk I/O Function Implementation Guide](#) .

### Names of the generated elements

By default, the generated blocks and subsystems are named according to the following scheme:

- Names of model port blocks that are related to ECU functions or data accesses defined in the EIC file:  
`<function name> [<function block name>\<ECU name>\<application name>\<ECU function or data access name>]`
- Names of model port blocks that are related to the ECU interfaces:
  - Control [`<function block name>\<ECU name>\ECU Interfaces\<ECU interface name>`]
  - Status [`<function block name>\<ECU name>\ECU Interfaces\<ECU interface name>`]
- Names of Runnable Function blocks:  
`<function block name>\<ECU name>\<application name>\<ECU function or data access name>\<function name>\<event name>`

## Notes on Specifying CAN ECU Interfaces

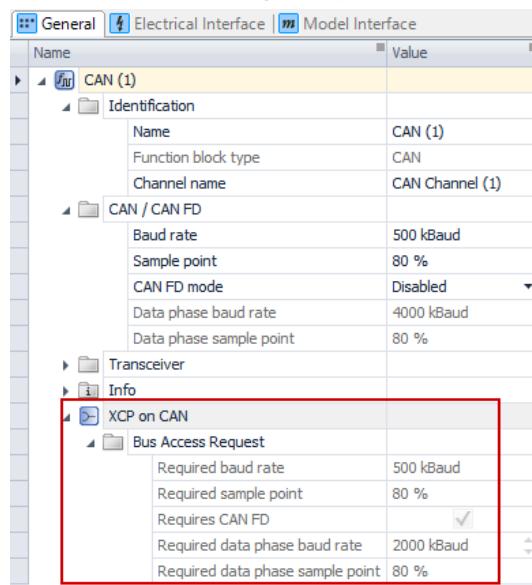
### Introduction

To set up a CAN ECU interface, you must reference a CAN function block in an ECU Interface Configuration function block. For an overview of a typical workflow, refer to [Typical Workflow for Implementing ECU Interfacing in ConfigurationDesk](#) on page 656.

Additionally, there are some points to note for specifying the CAN ECU interface in ConfigurationDesk.

### Requirements provided by EIC files

EIC files can provide requirements for accessing a CAN bus, e.g., a specific baud rate or required CAN FD support. When you select a CAN function block that is referenced in an ECU Interface Configuration function block, the General page of the Properties Browser provides access to the specified requirements, as shown in the following example.

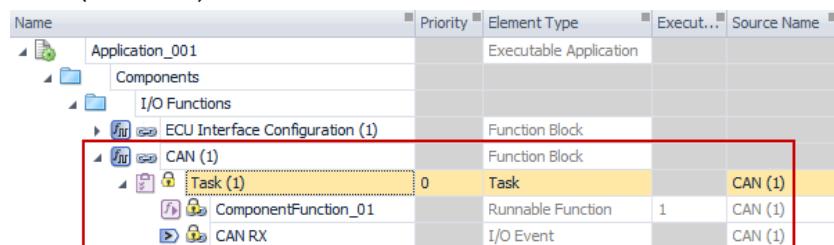


The screenshot shows the Properties Browser with the 'General' tab selected. Under the 'CAN (1)' section, there is a 'XCP on CAN' requirement block highlighted with a red box. This block contains properties such as 'Required baud rate' (500 kBaud), 'Required sample point' (80 %), 'Requires CAN FD' (checked), 'Required data phase baud rate' (2000 kBaud), and 'Required data phase sample point' (80 %).

If the settings that are specified for the related properties of the CAN function block differ from the requirements, a conflict occurs. However, the conflict does not affect the build process for building a real-time application. For example, this allows you to use a channel type of your SCALEXIO or MicroAutoBox III system that is sufficient for the ECU interfacing scenario but does not support all of the requirements.

### Tasks provided by CAN function blocks

When you reference a CAN function block in an ECU Interface Configuration function block, an asynchronous task with an assigned I/O event and runnable function is available for the CAN function block. The default name of the task is Task (<number>).



The screenshot shows a task list table with columns: Name, Priority, Element Type, Execut..., and Source Name. The table shows an executable application named 'Application\_001'. Under its components, there is an 'I/O Functions' section. An 'ECU Interface Configuration (1)' block is expanded, revealing a 'CAN (1)' block which in turn has a 'Task (1)' block. The 'Task (1)' block is highlighted with a red box. The table entries for the task are: Priority 0, Element Type Task, Source Name CAN (1). Below the task, there are two entries: 'ComponentFunction\_01' (Runnable Function, Priority 1, Source Name CAN (1)) and 'CAN RX' (I/O Event, Priority 1, Source Name CAN (1)).

At run time, the task is required to ensure that CAN messages are received by the real-time application with a low latency. For this purpose, the task requires a high priority. Therefore, the default priority of the task is 0, which is the highest priority.

**Note**

- To ensure optimum run-time behavior, do not change the default priority.
- To ensure that the ECU Interface Configuration function block provides only current data (e.g., to a mapped behavior model), all the tasks that are used to access the data of the function block must have a lower priority. A lower priority means a higher numerical value.

You can access and configure the task in the Task Configuration table, for example. However, the default settings of the task are sufficient in most use scenarios and it is recommended to make no manual changes.

For more information on tasks, refer to [Basics on Tasks, Events, and Runnable Functions](#) on page 480.

## Preconfigured Preprocessor Macros for ECU Interfacing

### Introduction

When you configure the build process, you can specify preprocessor macros. For ECU interfacing, preconfigured preprocessor macros are available, which automatically apply to each build configuration set. However, you can change the default value of the macros for each build configuration set.

For more information on build configuration sets, refer to [Specifying Options for the Build Process](#) on page 707.

### Available preconfigured macros

For ECU interfacing, the following preconfigured preprocessor macros are available.

**Macros for specifying the display of XCP service request messages** XCP service request messages of the SERV\_TEXT type are sent by an XCP slave (ECU) when it initializes the connection to the XCP master (measurement and calibration system). If the XCP service request message display is enabled, XCP service request messages of the SERV\_TEXT type are written to the dSPACE Log during run time of the real-time application. You can view them, for example, in the ControlDesk Messages controlbar.

The following macros specify the display of XCP service request messages:

Macro	Values	Purpose
DSBYPASS_GENERIC_DSXCP_DISPLAY_ECU_SERV_TEXT_DEFAULT	<ul style="list-style-type: none"> <li>▪ DSBYPASS_TRUE (default value)</li> <li>▪ DSBYPASS_FALSE</li> </ul>	To specify whether to display XCP service request messages of the SERV_TEXT type.
DSBYPASS_GENERIC_DSXCP_SERV_REQ_STRING_NUMBER	<ul style="list-style-type: none"> <li>▪ Value range: 1 ... 255</li> <li>▪ Default value: 50</li> </ul>	To specify the number of XCP service request messages of the SERV_TEXT type to display.

Macro	Values	Purpose
DSBYPASS_GENERIC_DSXCP_SERV_REQ_STRING_SIZE_MAX	<ul style="list-style-type: none"> <li>▪ Value range: 1 ... (MAX_CTO - 2), with MAX_CTO = maximum size of an XCP command transmission object (CTO)</li> <li>▪ Default value: (MAX_CTO - 2)</li> </ul>	To specify the maximum size (in characters) of XCP service request message strings to display.

**Macro for specifying the maximum number of ECU interface configurations that use the same interface type** By default, the maximum number of ECU interface configurations for each supported communication interface type is 16, i.e., 16 for CAN and 16 for Ethernet. The maximum number is independent of the number of available hardware interface channels. Suppose you want to perform ECU interfacing via Ethernet. Using the default maximum number of 16, you can access up to 16 ECUs, for example, 10 ECUs via XCP on Ethernet and 6 ECUs via DCI-GS12, which uses Ethernet as the transport layer.

The following macro specifies the maximum number:

Macro	Values	Purpose
DSBYPASS_MAX_NO_OF_SERVICES_PER_INTERFACE	<ul style="list-style-type: none"> <li>▪ Value range: 1 ... 255</li> <li>▪ Default value: 16</li> </ul>	To specify the maximum number of ECU interface configurations in a real-time application that use the same interface type.

#### Changing the default values of the macros

Generally, the preconfigured preprocessor macros apply to each build configuration set by default. Therefore, no additional steps are required to use the macros. However, you can change the default values of the macros via the Macros to be defined property of each build configuration set.

For example, to display 75 XCP service request messages of the SERV\_TEXT type with a maximum string size of 128, type the following in the Macros to be defined edit field:

```
DSBYPASS_GENERIC_DSXCP_DISPLAY_ECU_SERV_TEXT_DEFAULT=75 DSBYPASS_GENERIC_DSXCP_SERV_REQ_STRING_SIZE_MAX=128
```

# Updating ECU Interface Containers in ConfigurationDesk Applications

## Where to go from here

## Information in this section

Basics on Updating ECU Interface Containers in ConfigurationDesk Applications.....	671
How to Update an ECU Interface Container in the ConfigurationDesk Application.....	674

## Basics on Updating ECU Interface Containers in ConfigurationDesk Applications

### Introduction

If an ECU interface container changed after you imported the container to an ECU Interface Configuration function block and a new version of the related EIC file is available, you can update the EIC file in the function block.

### Updating an ECU interface container

For each ECU Interface Configuration function block that is available in the signal chain, you can update the imported ECU interface container via the Update ECU Interface Container command. You can select a new version of the EIC file from the file system and replace the originally imported EIC file with this file version. In this case, the function block is updated according to the new EIC file.

When you select an EIC file for the update, ConfigurationDesk compares the container identification of the selected EIC file and the EIC file that was originally imported to the ECU Interface Configuration function block. Differences in the container identification might indicate that the EIC files are not different versions of the same ECU interface container but completely different ECU interface containers. If the container identifications differ, you are therefore asked whether to continue the update.

### Update effects on function ports

When you update the EIC file in an ECU Interface Configuration function block, the structure of the function block's function groups, functions, and function ports is derived from the new EIC file. The update effects on function ports differ depending on whether their ID is found in both the old and the new EIC file.

**Function port ID found in both EIC files** Function ports whose ID is found in both the old and the new EIC file are updated:

- The settings of the following function port properties are derived from the new EIC file:
  - The position of the function port in the structure of the function block's model interface (i.e., the related higher-level function and function groups of the function port)
  - Function port name
  - Port type
  - Data type
  - Data width
- If the Initial substitute value was changed by the user, it is set to the default value 0. Other properties that were changed by the user (e.g., Model access, Activate test automation support) remain unchanged.
- If a function port is mapped to a model port, the mapping remains unchanged. If the Port type (i.e., the direction) of the function port changed, invalid mappings occur.

**Function port ID found only in the old EIC file** Function ports whose ID is found only in the old EIC file are obsolete:

- If the function ports are not mapped to model ports, they are deleted from the function block.
- If the function ports are mapped to model ports, they become unresolved. You can delete these function ports by deleting the mapping lines.

**Function port ID found only in the new EIC file** Function ports whose ID is found only in the new EIC file are added to the function block.

#### Update effects on I/O events and their event ports

Updating the EIC file in an ECU Interface Configuration function block affects the function block's I/O events and the related event ports. The update effects differ depending on whether the ID of an I/O event is found in both the old and the new EIC file.

**I/O event ID found in both EIC files** I/O events whose ID is found in both the old and the new EIC file are updated. This has the following effects on the I/O events and the related event ports:

- Settings specified for the I/O events, such as the event type, are derived from the new EIC file. The related event ports are updated accordingly.
- If the I/O events are assigned to tasks, the assignment remains unchanged. If the related event ports are mapped to Runnable Function blocks, the mapping remains unchanged as well.

**I/O event ID found only in the old EIC file** I/O events whose ID is found only in the old EIC file are obsolete. This has the following effects on the I/O events and their related event ports:

- If the I/O events are not assigned to tasks, they are deleted from the function block and the ConfigurationDesk application.

- If the I/O events are assigned to tasks, the assignment remains unchanged. The I/O events are still available in the ConfigurationDesk application but you cannot access them via the function block anymore. Instead, you can access them via the Executable Application table, for example. There, you can delete the events by deleting the assignment to the tasks, for example.
- I/O events of ECU Interface Configuration function blocks always belong to the Input, Output, Read, or Write functions of the function block. If an I/O event is assigned to a task but the related Input, Output, Read, or Write function is not available (e.g., because it was deleted during the update), the event becomes unresolved. To avoid conflicts, delete the event.
- Event ports of obsolete I/O events that are not mapped to Runnable Function blocks are deleted from the function block.
- Event ports of obsolete I/O events that are mapped to Runnable Function blocks become unresolved. You can delete these event ports by deleting the mapping lines.

**I/O event ID found only in the new EIC file** I/O events whose ID is found only in the new EIC file are added to the function block. For each newly added I/O event an event port is added to the function block as well.

#### Update effects on the requirements regarding the ECU interface

The requirements regarding the ECU interface (i.e., the interface for exchanging data between SCALEXIO/MicroAutoBox III and the target ECU) are derived from the new EIC file:

- If the name and the ID of an ECU interface are identical in both the old and the new EIC file, the requirements regarding the ECU interface are updated in the ECU Interface Configuration function block. If the ECU Interface Configuration function block references an Ethernet Setup or CAN function block that sets up the connection to the ECU interface, this reference remains unchanged.
- If the name and/or the ID of an ECU interface differs in the EIC files, the existing requirements regarding the ECU interface are deleted from the ECU Interface Configuration function block, including the reference to an Ethernet Setup or CAN function block. Instead, the requirements regarding the ECU interface specified in the new EIC file are added to the ECU Interface Configuration function block.

To set up the connection to the ECU interface, you must then specify an Ethernet Setup or CAN function block as the Referencing Configuration in the ECU Interface Configuration function block.

#### Limitations

When you update ECU interface containers in the ConfigurationDesk application, some limitations apply. For more information, refer to [Limitations Concerning ECU Interfacing with SCALEXIO or MicroAutoBox III](#) on page 779.

## How to Update an ECU Interface Container in the ConfigurationDesk Application

<b>Objective</b>	To adapt an ECU Interface Configuration function block to a new version of the imported ECU interface container (EIC) file, you can update the EIC file in the function block.
<b>Preconditions</b>	<ul style="list-style-type: none"><li>▪ An ECU Interface Configuration function block with an imported EIC file is available in the signal chain.</li><li>▪ A new version of the imported EIC file is available.</li></ul>
<b>Method</b>	<p><b>To update an ECU interface container in the ConfigurationDesk application</b></p> <ol style="list-style-type: none"><li>1 Select the ECU Interface Configuration function block in the signal chain.</li><li>2 Right-click the ECU Interface Configuration function block and select Update ECU Interface Container from the context menu. An Open dialog opens.</li><li>3 In the dialog, navigate to the new version of the EIC file.</li><li>4 Select the EIC file and click Open.</li></ol> <p>ConfigurationDesk compares the container identifications of the selected EIC file and the EIC file that was originally imported to the ECU Interface Configuration function block:</p> <ul style="list-style-type: none"><li>▪ If the container identifications are identical, the EIC file in the ECU Interface Configuration function block is updated.</li><li>▪ If the container identifications differ, an ECU Interface Container ID Mismatch dialog opens asking you whether to continue the update.</li></ul> <ol style="list-style-type: none"><li>5 If the ECU Interface Container ID Mismatch dialog opens, continue or abort the update:<ul style="list-style-type: none"><li>▪ In the dialog, click Yes to continue and update the EIC file in the ECU Interface Configuration function block.</li><li>▪ In the dialog, click No to abort the update. In this case, the dialog closes and the ECU Interface Configuration function block remains unchanged.</li></ul></li></ol>
<b>Result</b>	If you did not abort the update, the EIC file in the ECU Interface Configuration function block is updated with the new EIC file: <ul style="list-style-type: none"><li>▪ Elements that are unambiguously identified in both EIC files are updated in the function block.</li><li>▪ Elements that are available only in the new EIC file are added to the function block.</li><li>▪ Elements that are available only in the old EIC file become obsolete. Depending on the elements (e.g., function ports, I/O events) and their configuration in the signal chain, they are deleted or become unresolved.</li></ul>

**Tip**

If you work in the **Temporary** or a user-defined working view, not all update effects might be displayed. Open the **Global** working view to view all update effects.

For more information on the update effects, refer to [Basics on Updating ECU Interface Containers in ConfigurationDesk Applications](#) on page 671.



# Resolving Conflicts

## Objective

ConfigurationDesk allows for 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 to get proper build results.

## Where to go from here

### Information in this section

Basics on Resolving Conflicts.....	677
Details on Filtering Conflicts.....	679
How to Resolve Conflicts via the Conflicts Viewer.....	681

## Basics on Resolving Conflicts

### Effects of conflicts

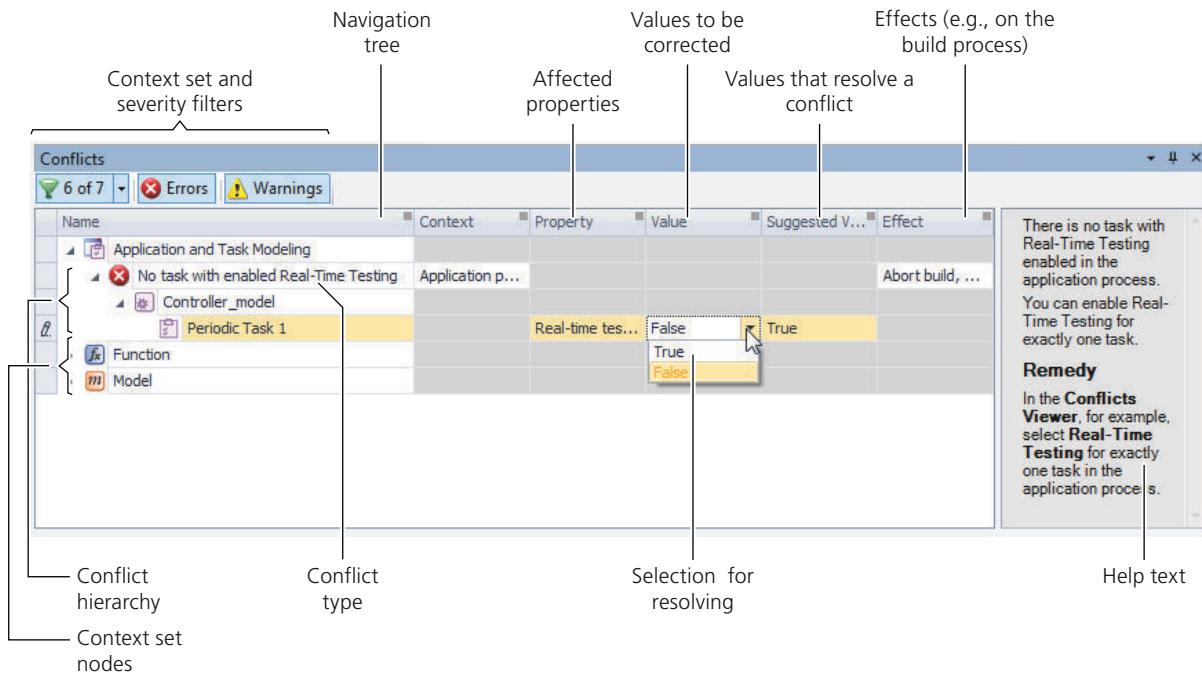
When you work with ConfigurationDesk, conflicts might occur that influence certain things, such as the results of the build process. For a proper build result, use the **Conflicts Viewer** to view and resolve at least the most severe conflicts. The conflicts are assigned to specific context sets and can be resolved selectively, according to the requirements.

#### Tip

Conflicts are also highlighted and displayed in a tooltip in working views with enabled highlighting (refer to [Highlighting Signal Chain Conflicts](#) on page 194).

### Principles of resolving conflicts

The conflicts are displayed in the Conflicts Viewer according to active context set and severity filters. For details, refer to [Details on Filtering Conflicts](#) on page 679.



The conflicts are sorted by the context sets, which you can navigate to by clicking on the context set nodes.

The relevant elements with regard to the conflict are displayed in an expandable conflict hierarchy. Most of the conflicts can be resolved directly in the Conflicts Viewer. If available, you can select the corresponding value from the list shown in the Value column of the conflict element or enter an appropriate value.

Changing the value of a property could cause new conflicts because the new value might mismatch the settings of other properties. Values with a colored markup (orange or red) will not resolve a conflict.

For some conflicts, the Suggested Values column contains values that will resolve this conflict.

New conflicts are shown immediately in the Conflicts Viewer. Resolved conflicts are cleared.

Some conflicts (such as mapping conflicts) cannot be resolved directly in the Conflicts Viewer. In those cases you can display the conflict elements in a working view or in a table by using context menu commands.

### Disabling conflict evaluation

If you want to speed up interface actions in a large application with many signal chain elements, you can disable the evaluation of conflicts by disabling all context set or severity filters.

**Logging conflicts**

The conflicts displayed in the Conflicts Viewer are not logged in the Message Viewer, with the following exceptions:

- Conflicts that influence the result of the build process. These are logged in the Message Viewer when a build is performed.
- Conflicts that affect an exported Excel file, such as an exported device topology Excel file.

**Exporting Conflicts Viewer entries**

If you want to document the entries in the Conflicts Viewer, you can export them to an XML file or a CSV (comma separated value) file. For details about XML or CSV file export, refer to [Exporting Data for Documentation Purposes](#) on page 757.

**Related topics****Basics**

[Details on Filtering Conflicts](#)..... 679

**HowTos**

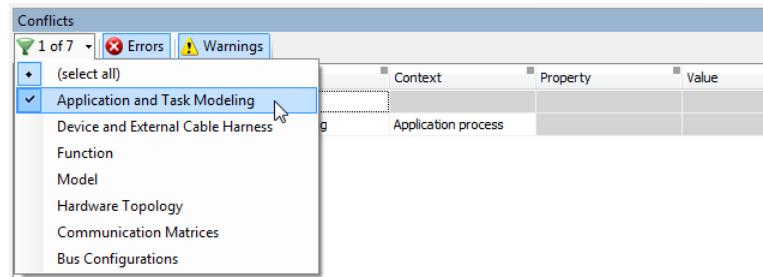
[How to Resolve Conflicts via the Conflicts Viewer](#)..... 681

## Details on Filtering Conflicts

**Filtering by context**

The Conflicts Viewer provides a context set filter to display only conflicts from a range of related contexts (e.g., conflicts regarding application and task modeling).

By default, most of the context sets are activated and their related conflicts are displayed in the Conflicts Viewer. You can disable context sets to reduce the number of displayed conflicts and to focus on specific contexts.



Clicking **(select all)** or the context set filter button activates all context sets.

The following context sets are available in the filter list:

Context Set	Description
Application and Task Modeling	Conflicts that occur if application processes and tasks are not configured correctly, e.g., if no behavior model is assigned to an application process.
Device and External Cable Harness	Conflicts that are caused by elements of the external device topology or the external cable harness, or differences between them.
Function	Conflicts regarding the configuration of function blocks, e.g., missing hardware resource assignments.
Model	Conflicts regarding the behavior model, e.g., unresolved model port blocks.
Hardware Topology	Conflicts regarding the hardware topology, e.g., the assignment of multiple boards to the same slot.
Communication Matrices	Conflicts regarding communication matrices, e.g., a missing cluster name.
Bus Configurations	Conflicts regarding bus configurations, e.g., missing bus accesses.

### Filtering by severity

The Conflicts Viewer also provides filters that let you focus on severe or less severe conflicts.

Name	Context	Property	Value	Suggested Values	Effect
No task with enabled Real-Time Testing	Application process				Abort build, Abort BS...
Duplicate name	Device port				Abort XLSX export

You can filter for the following severity levels:

Severity	Description
Error	Severe conflicts that cause an operation to be aborted, for example, the build process.
Warning	Conflicts that may lead to faulty build results and are recommended to be resolved, but which do not abort operations.

### Related topics

#### Basics

[Basics on Resolving Conflicts.....](#) 677

# How to Resolve Conflicts via the Conflicts Viewer

## Objective

The Conflicts Viewer is often the tool to use to resolve the displayed conflicts directly.

## Restriction

The following instructions refer to a conflict example which can be resolved in the Conflicts Viewer. This is possible for most conflicts. Some conflicts, for example, mapping conflicts cannot be resolved directly in the Conflicts Viewer.

### Tip

- You can display or select elements of a conflict hierarchy in a working view or in a table by using context menu commands such as [Show in Signal Chain Browser](#).
- You can highlight the state of signal chain elements to help you identify elements involved in a conflict. For details, refer to [Highlighting Signal Chain Conflicts](#) on page 194.

## Precondition

At least one conflict is displayed in the Conflicts Viewer.

## Method

### To resolve conflicts via the Conflicts Viewer

- 1 Select a conflict in the Conflicts Viewer.
- 2 Expand the hierarchy of the conflict and navigate to the Value column.
- 3 Open the list in the Value field to display the currently possible values.

Name	Context	Property	Value	Suggested Values	Effect
Device and External Cable Harness		Device port mapping			
Mismatching port type					
Mapping line (DoorLight - Signal)					
DoorLight		Port type	Bidirectional	In; Out; Reference; ...	
Signal					

- 4 Select a value from the list.

If you select a value with a colored markup, the conflict will not be resolved. If you select values without a markup, the conflict will be resolved. In some cases, conflicts of other conflict types might arise.

## Result

The conflict is resolved and cleared from the Conflicts Viewer.

**Related topics**

**Basics**

<a href="#">Basics on Resolving Conflicts.....</a>	677
--	-----

# Calculating an External Cable Harness

## Objective

You can calculate or import a representation of the external wiring information in ConfigurationDesk, which helps you build an external cable harness or reuse an existing cable harness.

### Note

If the calculated external cable harness represents an existing cable harness that you want to continue using, do not recalculate the external cable harness after applying changes affecting the wiring information.

## Where to go from here

## Information in this section

Basics on the External Cable Harness.....	683
How to (Re)Calculate the External Cable Harness.....	685
How to View the Wiring Information.....	687
How to Export Wiring Information for an External Cable Harness.....	689
Description of an Exported Cable Harness Microsoft Excel™ Sheet.....	690
Examples of Exported Wiring Information.....	693

## Basics on the External Cable Harness

### External cable harness

The external cable harness is the connection between the I/O connectors of the real-time hardware and the devices, such as the ECUs to be tested.

The external cable harness is represented by a specific component in a ConfigurationDesk application. This component contains the wiring information for the external cable harness. It contains only the logical connections and no further information like cable length, cable diameters, dimensions or the arrangement of connection points, etc.

The wiring information is stored in a specific file format (ECHX file). It can be calculated by ConfigurationDesk or imported from an ECHX file so that you can use an existing cable harness and do not have to build a new one.

ConfigurationDesk calculates the connection between device pins (for example, of your ECU) and the I/O connector pins of the dSPACE hardware. Bridges are also determined, for example, if channel multiplication is needed for current enhancement.

To get an overview of the wiring information, you can view it in the Pins and External Wiring table or export it to a Microsoft Excel™ file.

#### Requirements for the external cable harness file

The wiring information must match the application configuration. Depending on whether you are building a new external cable harness or reusing an existing one, you must calculate the ECHX file so that it matches the current application configuration, or import an existing ECHX file and configure the application according to it.

**Wiring information for a new external cable harness** You have to perform the following steps before you calculate the wiring information for the external cable harness. If you update one of the steps, the required cable harness might not match the calculated cable harness.

- Assigning device pins to device ports (device pin assignment).
- Adding devices to the signal chain.
- Mapping device blocks to function blocks.
- Assigning hardware resources to function blocks (hardware resource assignment).

**Wiring information for an existing external cable harness** You can use the representation of an existing cable harness in your ConfigurationDesk application to avoid building a new cable harness. In this case, the external cable harness component must not be changed.

Import the ECHX file that represents the existing cable harness. For instructions, refer to [How to Import an External Cable Harness](#) on page 117.

When you create the signal chain, wiring conflicts can occur, and these are displayed in ConfigurationDesk. If you want to use an existing cable harness, you must not use the calculate command to resolve the conflicts. You must solve wiring conflicts at other positions in the signal chain (hardware resource assignment, device port mapping or device pin assignment).

#### Related topics

##### Basics

Assigning Hardware Resources to Function Blocks.....	399
Device Port Mapping.....	312

Specifying the External Device Interface.....	251
---	-----

## HowTos

How to Export Wiring Information for an External Cable Harness.....	689
How to Import an External Cable Harness.....	117

## How to (Re)Calculate the External Cable Harness

### Objective

To get the wiring information that you need for building the cable harness between your external device and the I/O connector(s) of the dSPACE hardware, ConfigurationDesk lets you calculate the external cable harness.

#### Tip

Only calculate the external cable harness in your ConfigurationDesk application if you actually need the wiring information. If a calculated external cable harness exists, you can cause several conflicts during signal chain configuration since the settings affecting the wiring information are constantly compared to the external cable harness. You can remove the calculated external cable harness from your ConfigurationDesk application via a context menu command in the Project Manager.

### Using an existing cable harness

#### Note

If the calculated external cable harness represents an existing cable harness that you want to continue using, do not recalculate the external cable harness after applying changes affecting the wiring information.

### Preparation

To get a useful result, all working steps which influence the connection between your ECU and the dSPACE hardware should be finished. These are the device pin assignment, the device port mapping, and the hardware resource assignment.

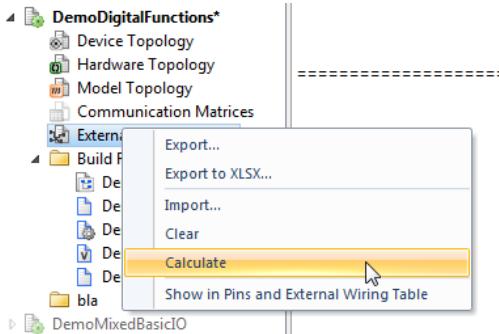
### Possible methods

You can calculate the external cable harness in two different ways:

- To calculate the external cable harness from the Project Manager, refer to [Method 1](#) on page 686.
- To calculate the external cable harness from the Pins and External Wiring table, refer to [Method 2](#) on page 686.

**Method 1****To calculate the external cable harness from the Project Manager**

- In the Project Manager, right-click the External Cable Harness component and select Calculate from the context menu.

**Method 2****To calculate the external cable harness from the Pins and External Wiring table**

- In the Pins and External Wiring table, right-click anywhere and select Calculate External Wiring from the context menu.

A screenshot of a table titled 'Pins and External Wiring'. The table has two columns: 'Name' and 'Connection ID'. It contains six rows with data: Pin 7 (Connection ID 2), Pin 8 (Connection ID 5), Pin 1 (Connection ID 1), Pin 2 (Connection ID 4), and Cut (Connection ID 6). A context menu is open at the bottom of the table, with the 'Calculate External Wiring' option highlighted in blue.

Name	Connection ID
Pin 7	2
Pin 8	5
Pin 1	1
Pin 2	4
Cut	6

**Result**

You calculated the external cable harness. If this is the first time you have calculated it for the active ConfigurationDesk application, the External Cable Harness component in the Project Manager is no longer grayed out. If you already calculated an external cable harness previously, it is automatically replaced.

**Next steps**

You can export the wiring information to an XLSX or ECHX file. The XLSX file contains information necessary for building the actual external cable harness (for instructions, refer to [How to Export Wiring Information for an External Cable Harness](#) on page 689). The ECHX file can be used to import the wiring information to a different ConfigurationDesk application (for instructions, refer to [How to Export Data of a Specific Application Component](#) on page 119).

**Related topics****Basics**

[Basics on Device Pin Assignment.....](#) 255

**HowTos**

[How to Export Data of a Specific Application Component.....](#) 119

[How to Export Wiring Information for an External Cable Harness.....](#) 689

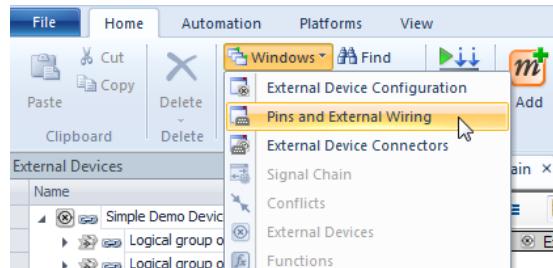
## How to View the Wiring Information

**Objective**

To get an overview of the external wiring information as well as the device connector pins, I/O connector pins, and the device pin assignment, you can use the Pins and External Wiring table.

**Method****To view the wiring information**

- 1 Switch to the Signal Chain view set if necessary.
- 2 On the Home ribbon, select Navigation – Windows – Pins and External Wiring.



The Pins and External Wiring table opens.

Name	Connection ID	Short ID	Connector	Signal	Port
Pin 7	2	1		Digital reference ...	
Pin 8	5	1		Digital reference ...	
Pin 1	1	1		Digital input sign...	
Pin 2	4	1		Digital output sig...	
Pin 21	6	1		Pulsed output sig...	
Pin 11	5	4		Pulse reference p...	
Pin 3	3	1		PWM input signal ...	
Pin 9	2	3		PWM reference p...	
Pin 22	7	1		PWM output sign...	
Pin 12	8	1		PWM reference p...	
Pin 201	9	1		Digital output sig...	
Pin 301	5	3		Digital reference ...	
Pin 211	10	1		Pull reference po...	
Pin 311	11	1		Digital input sign...	
Pin 212	12	1		Push reference p...	
Pin 221	13	1		Pull reference po...	
Pin 321	14	1		Digital input sign...	
Pin 222	15	1		Push reference p...	
A1				ECU1 [SCALEXIO ...	Analog In 1 Chan...
A2				ECU1 [SCALEXIO ...	Analog In 1 Chan...
A3				ECU1 [SCALEXIO ...	Analog In 1 Chan...
A4				ECU1 [SCALEXIO ...	Analog In 1 Chan...

Pins are displayed with different symbols:

Symbol	Description
	Indicates a device pin.
	Indicates an I/O connector pin (dSPACE real-time hardware).

The following table provides descriptions of the table columns:

Column	Description
Name	Name of the device pin or I/O connector pin. You can change the name of device pins here.
Connection ID	The Connection ID shows which device ports (each representing one or more device pins) and simulator signals (each available on one or more I/O connector pins) need to be connected. The external cable harness needs to be built in a way that pins with the same Connection ID hold the same electrical potential. For details on building the external cable harness, refer to: <ul style="list-style-type: none"> <li>▪ SCALEXIO: General Rules for Building the External Cable Harness (SCALEXIO Hardware Installation and Configuration </li> <li>▪ MicroAutoBox III: Workflow and Notes on Building the Cable Harness (MicroAutoBox III Hardware Installation and Configuration </li> </ul>
Short ID	The Short ID identifies different device ports or simulator signals belonging to the same signal (if they have the same Connection ID). Pins with the same Connection ID and different Short ID need to be connected by the external cable harness. Device pins or simulator pins with the same Short ID and Connection ID are internally connected, meaning that device pins were assigned to the same device port or I/O connector pins were part of the same simulator signal when the external cable harness information was calculated. In this case, you can choose how many pins of this signal are to be connected when building the external cable harness. This might depend, for example, on the maximum current that is to be supported.
Connector	Name of the connector, e.g., ECU2. You can group device pins in connectors that have been specified in the External Device Connectors table.

Column	Description
Signal	Name of the signal as shown in the simulator connector table.
Port	Name of the device or signal port. You can edit the device pin assignment.

- 3 You can filter the pins using the Filter for Used Elements, Filter for Unused Elements, and Filter for Unresolved Elements commands:
- Used pins are part of the calculated external cable harness. This is indicated by the  symbol.

**Tip**

To get an overview of the wiring information for the device and I/O connector pins in a printable file, you can export it to a Microsoft Excel™ file. For instructions, refer to [How to Export Wiring Information for an External Cable Harness](#) on page 689.

- Unresolved pins are referenced by the calculated external cable harness but not provided by the device topology or simulator hardware topology. This is indicated by the  symbol.

---

<b>Result</b>	You viewed the device pin assignment and external wiring information in the Pins and External Wiring table.
---------------	---

---

<b>Related topics</b>	<a href="#">Basics</a> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;"><a href="#">Basics on Device Pin Assignment.....</a></td><td style="padding: 2px; text-align: right;">255</td></tr> </table> <a href="#">HowTos</a> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;"><a href="#">How to Assign Device Ports to External Device Pins.....</a></td><td style="padding: 2px; text-align: right;">278</td></tr> <tr> <td style="padding: 2px;"><a href="#">How to Export Wiring Information for an External Cable Harness.....</a></td><td style="padding: 2px; text-align: right;">689</td></tr> <tr> <td style="padding: 2px;"><a href="#">How to Group Device Pins.....</a></td><td style="padding: 2px; text-align: right;">283</td></tr> </table>	<a href="#">Basics on Device Pin Assignment.....</a>	255	<a href="#">How to Assign Device Ports to External Device Pins.....</a>	278	<a href="#">How to Export Wiring Information for an External Cable Harness.....</a>	689	<a href="#">How to Group Device Pins.....</a>	283
<a href="#">Basics on Device Pin Assignment.....</a>	255								
<a href="#">How to Assign Device Ports to External Device Pins.....</a>	278								
<a href="#">How to Export Wiring Information for an External Cable Harness.....</a>	689								
<a href="#">How to Group Device Pins.....</a>	283								

## How to Export Wiring Information for an External Cable Harness

---

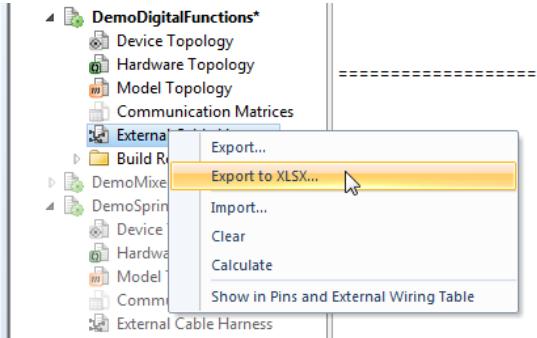
<b>Objective</b>	To prepare and build the external cable harness, you need wiring information. You can export the wiring information of a currently active ConfigurationDesk application to a Microsoft Excel™ (XLSX) file.
------------------	--

---

<b>Precondition</b>	<ul style="list-style-type: none"> <li>▪ An application which contains a calculated external cable harness is active.</li> </ul>
---------------------	--

**Method****To export wiring information for an external cable harness**

- 1 In the Project Manager, right-click External Cable Harness and select Export to XLSX from the context menu:



An Export External Cable Harness dialog opens.

- 2 In the Export External Cable Harness dialog, choose the folder you want to export the external cable harness file to.
- 3 Enter the file name, with or without the file name extension.
- 4 Click Save.

**Result**

You exported the wiring information for an external cable harness to an Microsoft Excel™ file (XLSX file). For details on the contents of the exported file, refer to [Description of an Exported Cable Harness Microsoft Excel™ Sheet](#) on page 690.

**Related topics****Basics**

[Basics on the External Cable Harness](#)..... 683

**Examples**

[Description of an Exported Cable Harness Microsoft Excel™ Sheet](#)..... 690

## Description of an Exported Cable Harness Microsoft Excel™ Sheet

**Objective**

An exported cable harness Microsoft Excel™ sheet contains information on the connection of pins, which helps you to build an external cable harness. You cannot import this file.

## Entries in the exported file

The wiring information is provided in a table format as shown in the example below:

A	B	C	D	E	F	G	H	I	J	K	L
1	2	3	4	5	6	7	8	9	10	11	12
Connection ID	Short ID	Device Block	Device Connec	Device Pin	Device Port G	Device Port	8	Unit Name	productName	Connector	Connector Pin
1	1	LeftDoorControl	Sub-D A	A1		DoorLight					
4	1							SCALEXIO Rock	DS2680 I/O Unit (1)		
5	1							SCALEXIO Rock	DS2680 I/O Unit (1)		
6	1							ECU1		A9	
7	2							ECU1		A10	
8	2	Potential	M	N	O	P	Q	R	S	T	U
9	2	Name	PotentialRoles	Board	Channel Set	Channel	Function Block	Function Block	Electrical Interface	Signal Group	Signal Port
10	2	Digital In 1 Channel 1 Signal	Signal	DS2680 I/O Module (Slot 1)	Digital In 1	Channel 1	Multi Bit In	Bit In DoorLight Left	Electrical Interface	Bit 1	Signal
11	2	Digital In 1 Channel 2 Signal	Signal	DS2680 I/O Module (Slot 1)	Digital In 1	Channel 2	Multi Bit In	Bit In DoorLight Left	Electrical Interface	Bit 1	Signal
12	2	Digital In 1 Channel 3 Signal	Signal	DS2680 I/O Module (Slot 1)	Digital In 1	Channel 3	Multi Bit In	Bit In DoorLight Left	Electrical Interface	Bit 1	Signal
13	2										
14	2	GND (Flexible In, Digital In)	Low reference	DS2680 I/O Module (Slot 1)			Multi Bit In	Bit In DoorLight Left	Electrical Interface	Bit 1	Reference
15	2	GND (Flexible In, Digital In)	Low reference	DS2680 I/O Module (Slot 1)			Multi Bit In	Bit In MirrorHeating Left	Electrical Interface	Bit 1	Reference
16	2	GND (Flexible In, Digital In)	Low reference	DS2680 I/O Module (Slot 1)			Multi Bit In	Bit In DoorRight	Electrical Interface	Bit 1	Reference
17	3	GND (Flexible In, Digital In)	Low reference	DS2680 I/O Module (Slot 1)			Multi Bit In	Bit In MirrorHeating Right	Electrical Interface	Bit 1	Reference
18	3	GND (Flexible In, Digital In)	Low reference	DS2680 I/O Module (Slot 1)			PWM/PFM In	PWM In MirrorMotor Right	Electrical Interface	Signal Group	Reference
19	3	GND (Flexible In, Digital In)	Low reference	DS2680 I/O Module (Slot 1)			PWM/PFM In	PWM In MirrorMotor Left	Electrical Interface	Signal Group	Reference
20	3										
21	4										
21	4										
		Digital In 1 Channel 4 Signal	Signal	DS2680 I/O Module (Slot 1)	Digital In 1	Channel 4	Multi Bit In	Bit In MirrorHeating Left	Electrical Interface	Bit 1	Signal
		Digital In 1 Channel 5 Signal	Signal	DS2680 I/O Module (Slot 1)	Digital In 1	Channel 5	Multi Bit In	Bit In MirrorHeating Left	Electrical Interface	Bit 1	Signal
		Digital In 2 Channel 4 Signal	Signal	DS2680 I/O Module (Slot 1)	Digital In 1	Channel 6	Multi Bit In	Bit In MirrorHeating Left	Electrical Interface	Bit 1	Signal

The sheet contains a row for each device pin and each dSPACE hardware pin.

## Note

The wiring information in the table represents the external cable harness as you last calculated it in ConfigurationDesk. Conflicts affecting the external cable harness might lead to device pins not being included in the exported Microsoft Excel™ file, for example, a pin from a wrong device (Invalid pin assignment conflict).

The table below provides descriptions of the table columns:

Column	Description
Connection ID	<p>The Connection ID shows which device ports (each representing one or more device pins) and simulator signals (each available on one or more I/O connector pins) need to be connected. The external cable harness needs to be built in a way that pins with the same Connection ID hold the same electrical potential. For details on building the external cable harness, refer to:</p> <ul style="list-style-type: none"><li data-bbox="663 1351 1273 1396">▪ <a href="#">SCALEXIO: General Rules for Building the External Cable Harness (SCALEXIO Hardware Installation and Configuration </a></li><li data-bbox="663 1404 1273 1493">▪ <a href="#">MicroAutoBox III: Workflow and Notes on Building the Cable Harness (MicroAutoBox III Hardware Installation and Configuration </a></li></ul>
Short ID	<p>For examples of how the Connection ID and Short ID columns identify the pins to be connected by the external cable harness, refer to <a href="#">Examples of Exported Wiring Information</a> on page 693.</p>

Column	Description
	external cable harness information was calculated. In this case, you can choose how many pins of this signal are to be connected when building the external cable harness. This might depend, for example, on the maximum current that is to be supported. For examples of how the Connection ID and Short ID columns identify the pins to be connected by the external cable harness, refer to <a href="#">Examples of Exported Wiring Information</a> on page 693.
Device Block	Name of the device.
Device Connector	The device connector(s) that the device pin from the same row is part of, as defined in the Device Connectors table.
Device Pin	A device pin that is assigned to a device port. For details, refer to <a href="#">Basics on Device Pin Assignment</a> on page 255.
Device Port Group	Name of the device port group.
Device Port	Name of the device port.
Rack	Name of the dSPACE hardware rack.
Unit – Name	Name of the hardware unit as defined in the Hardware Resource Browser.
Unit – ProductName	Displays the name of the unit type.
Connector	Name of the dSPACE hardware connector, e.g., ECU2.
Connector Pin	Pin of the dSPACE hardware connector.
Potential – Name	Name of the dSPACE hardware potential.
Potential – PotentialRoles	Role of the dSPACE hardware potential.
Board	Name of the dSPACE hardware board as defined in the Hardware Resource Browser.
Channel Set	The channel set that the channel from the same row (see Channel) belongs to. A channel set is 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	The name of the channel as it appears in the Hardware Resource Browser.
Function property columns	The Function Block Type, Function Block, Electrical Interface, Signal Group, and Signal Port column groups contain the names of instantiated function blocks and of their elements.
Bridge	Entries in this column indicate that the respective simulator pins are not represented by signal ports but there are connections needed to achieve specific electrical functionalities.

**Note**

The dSPACE hardware might provide signals on more pins than the exported wiring information contains.

The build process determines the I/O connector signals of the dSPACE hardware. In contrast to the calculation of the wiring information, the build process is independent of the device port mapping and the device pin assignment. For example, whether a signal of the signal chain is mapped to a device port does not influence the signals which are provided by the dSPACE hardware.

**Related topics****Basics**

Basics on Device Pin Assignment.....	255
General Rules for Building the External Cable Harness (SCALEXIO Hardware Installation and Configuration 	
Workflow and Notes on Building the Cable Harness (MicroAutoBox III Hardware Installation and Configuration 	

**HowTos**

How to Export Wiring Information for an External Cable Harness.....	689
---	-----

## Examples of Exported Wiring Information

**Objective**

You can use the wiring information from an exported external cable harness Microsoft Excel™ sheet to build an external cable harness. This is illustrated by three examples below.

**Identifying pin connections**

In the external cable harness Microsoft Excel™ sheet, the Connection ID and Short ID columns show you which device ports and dSPACE hardware signals must be connected by an external cable harness. Ports and signals with pins having the same Connection ID and different Short ID must be connected.

A device pin is uniquely identified by its position in the device topology structure, a dSPACE hardware pin by the connector/board/unit/rack it belongs to.

### Example 1: Connecting a dSPACE hardware pin to an external device pin

	A	B	C	F	G	H	J	Y	Z	AA	AC	AD	Potential			
1	Connection ID	Short ID	Device Block	Name	Device Conn.	Device Pin	Device Port Group	Name	Device Port	Name	Rack	Unit Name	ProductName	Connector	Connector Pin	Name
1	1	1	Simple Demo Device	Pin 1	Spring mass damper input	Position of the mass	SCALEXIO Rack	DS2680 I/O Unit	DS2680 I/O Unit	ECU3	A1				Analog Out 4	
5	2	1	Simple Demo Device	Pin 7	Spring mass damper input	Reference port	SCALEXIO Rack	DS2680 I/O Unit	DS2680 I/O Unit	ECU3	A2				Analog Out 4	
6	2	2	Simple Demo Device	Pin 2	Spring mass damper output	Mass displacement	SCALEXIO Rack	DS2680 I/O Unit	DS2680 I/O Unit	ECU1	A1				Analog In 1 Cl	
7	3	1	Simple Demo Device	Pin 8	Spring mass damper output	Reference port	SCALEXIO Rack	DS2680 I/O Unit	DS2680 I/O Unit	ECU1	A2				Analog In 1 Cl	
8	3	2														
9	4	1														
10	4	2														

The table contains two different Short IDs for each Connection ID. As described above, you have to connect pins with identical Connection ID and different Short ID via an external cable harness. According to the table above, you have to connect, for example, the A1 dSPACE hardware pin from the ECU3 connector of the DS2680 I/O Unit in the SCALEXIO Rack with the Pin 1 device pin which is assigned to the Position of the mass port in the Spring mass damper input port group of the Simple Demo Device.

### Example 2: Connecting several dSPACE hardware pins to an external device pin

	A	B	C	F	G	H	J	Y	Z	AA	AC	AD	Potential			
1	Connection ID	Short ID	Device Block	Name	Device Conn.	Device Pin	Device Port Group	Name	Device Port	Name	Rack	Unit Name	ProductName	Connector	Connector Pin	Name
1	1	1	LeftDoorControl	Sub-D A	A1			DoorLight			SCALEXIO Rack	DS2680 I/O Unit (1)	DS2680 I/O Unit	ECU1	A9	Digital In 1 C
4	1	2									SCALEXIO Rack	DS2680 I/O Unit (1)	DS2680 I/O Unit	ECU1	A10	Digital In 1 C
5	1	3									SCALEXIO Rack	DS2680 I/O Unit (1)	DS2680 I/O Unit	ECU1	B9	Digital In 1 C
7	2	1	LeftDoorControl	Sub-D A	A2			LightGND								

This table contains four identical Connection IDs with different Short IDs. Here, you have to connect the A9, A10, and B9 dSPACE hardware pins from the ECU1 connector of the DS2680 I/O Unit (1) in the SCALEXIO Rack with the A1 device pin from the Sub-D A connector, which is assigned to the DoorLight port in the LeftDoorControl device.

### Example 3: Connecting a dSPACE hardware pin to several external device pins

	A	B	C	F	G	H	J	Y	Z	AA	AC	AD	Potential			
1	Connection ID	Short ID	Device Block	Name	Device Conn.	Device Pin	Device Port Group	Name	Device Port	Name	Rack	Unit Name	ProductName	Connector	Connector Pin	Name
3	1	1	Simple Demo Device	Pin 1	Spring mass damper input	Position of the mass	SCALEXIO Rack	DS2680 I/O Unit	DS2680 I/O Unit	ECU3	A1				Analog Out 4 Cr	
4	1	2	Simple Demo Device	Pin 7	Spring mass damper input	Reference port	SCALEXIO Rack	DS2680 I/O Unit	DS2680 I/O Unit	ECU3	A2				Analog Out 4 Cr	
5	2	1	Simple Demo Device	Pin 8	Spring mass damper output	Reference port	SCALEXIO Rack	DS2680 I/O Unit	DS2680 I/O Unit	ECU1	A1				Analog In 1 Cha	
6	2	2	Simple Demo Device	Pin 2	Spring mass damper output	Mass displacement	SCALEXIO Rack	DS2680 I/O Unit	DS2680 I/O Unit	ECU1	A2				Analog In 1 Cha	
8	3	1	Simple Demo Device	Pin 8	Spring mass damper output	Reference port	SCALEXIO Rack	DS2680 I/O Unit	DS2680 I/O Unit	ECU1	A1				Analog In 1 Cha	
9	3	2	Simple Demo Device	Pin 2	Spring mass damper output	Mass displacement	SCALEXIO Rack	DS2680 I/O Unit	DS2680 I/O Unit	ECU1	A2				Analog In 1 Cha	
10	4	1														

Here, there are three different Short IDs for Connection ID 2. This means that you have to connect the Pin 7 and Pin 8 device pins which are assigned to a Reference port from the Spring mass damper input and Spring mass damper output port group of the Simple Demo Device to the A2 dSPACE

hardware pin from the ECU3 connector of the DS2680 I/O Unit in the SCALEXIO Rack.

---

**Related topics**

HowTos

[How to Export Wiring Information for an External Cable Harness.....](#) 689

Examples

[Description of an Exported Cable Harness Microsoft Excel™ Sheet.....](#) 690



# Building Real-Time Applications

---

## Introduction

After you have configured an executable application in ConfigurationDesk, you can start to build a real-time application.

---

## Where to go from here

## Information in this section

Understanding the Build Process.....	698
Configuring the Build Process.....	707
Starting the Build Process.....	716
Information on Variable Description Files (TRC Files).....	722

# Understanding the Build Process

<b>Introduction</b>	To ensure that your ConfigurationDesk applications are properly built, make yourself familiar with the build process and its build results.
---------------------	---

Where to go from here	Information in this section
	<a href="#">Introducing the Build Process.....</a> 698
	<a href="#">Build Result Files of the Build Process.....</a> 702

## Introducing the Build Process

<b>Building real-time applications via ConfigurationDesk</b>	Building real-time applications is a process which generates code as an executable file that can be run on the real-time hardware. You can manage the build process completely via ConfigurationDesk without switching to MATLAB/Simulink®.
--	---

### Note

If you use V-ECU implementations as behavior models, there are special aspects to note for building real-time applications. Refer to [Details on the Build Process for Real-Time Applications Containing V-ECU Implementations](#) on page 641.

<b>Overview of the build process</b>	Once you started the build procedure, the following process steps are executed automatically:
--------------------------------------	---

1. Model code generation including model analysis

This step is executed only for Simulink models that are assigned to an application process. This step is skipped if you use container files as behavior models (e.g., SIC files or FMU files).

If not already open, MATLAB and the behavior model linked to the ConfigurationDesk application are opened. The model interface of the behavior model is analyzed.

If necessary, ConfigurationDesk automatically configures the parameters of the behavior model as described in [Automatically configured parameters](#) (see below). Afterwards, the Simulink® Coder™ code generation for the behavior model is started.

2. System integration code generation

System integration code is generated, including code for I/O functionality and task handling.

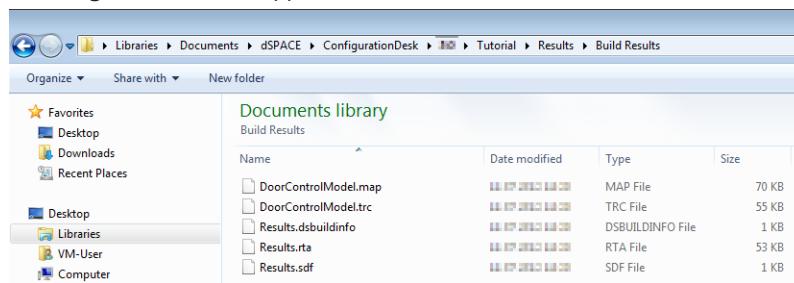
3. Make process

The code generated in the previous process steps is compiled and linked to the final real-time application `<application>.rta`, which can be downloaded to the real-time hardware.

The folder `<ConfigurationDesk Application>\Components` contains the generated model code from step 1.

The output files of the build process steps 2 & 3 can be found in `<ConfigurationDesk Application>\Build` and its subfolders.

After the build process has finished, its results can be found at `<ConfigurationDesk Application>\Build Results`.



Files which you created yourself and which are processed during the build process, for example, user variable description files, are not changed by the build process.

**Note**

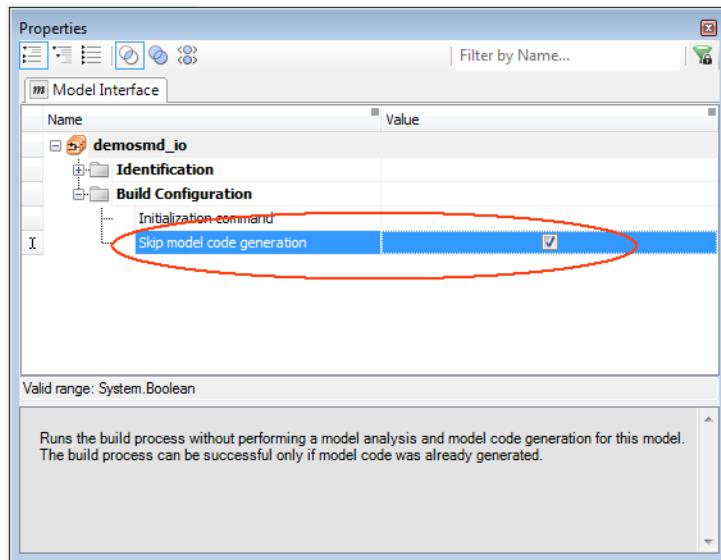
To generate code, each function block must be assigned to an application process. Otherwise, a conflict is shown. However, a function block cannot explicitly be assigned to application processes. Instead, it is assigned implicitly to an application process via:

- Its mapping lines to model port blocks of model implementations that are assigned to application processes.
- The assignment of its I/O events to tasks that are assigned to application processes.
- The hardware assignment: A function block must have a direct or indirect hardware connection to processing hardware.

**Build process without model code generation**

For Simulink models, ConfigurationDesk allows you to skip model code generation when building a real-time application. This reduces the time needed for the build process and is particularly useful if the behavior model has not been changed since the last build process. This feature also enables you to edit I/O configuration and task configuration, for example, without having to rebuild the behavior model completely.

To use this feature, the Skip model code generation checkbox must be selected in the model's properties.



Then no code is generated for the behavior model. The analysis data and the model code that were created during the last successful build process or model code generation are used instead.

#### Tip

In multimodel applications, you can skip model code generation for specific behavior models. If the model topology is complex, it is useful to generate code only for behavior models that have been modified.

During the build process, the Build Log Viewer informs you that the available model code is used. If no model code is available, the build process is aborted with an error message.

#### Note

When skipping model code generation, note the following limitations:

- ConfigurationDesk does not check whether your behavior model has been changed since the last model code generation. If you want new model code to be used during the next build process, you must ensure that the Skip model code generation option is not selected.
- The build process without model code generation cannot be performed without a MATLAB, Simulink, and Simulink Coder installation. It is needed for the make process. Thus, it is not possible to build model code on one PC and to use it on a different PC which has no Simulink Coder installed. The exchange of model code is possible only if the involved PCs use:
  - The same installation directories and the same versions of dSPACE software and modeling software.
  - The same path for the project and application to which the model code belongs.

**Tip**

To exchange model code between PCs that use different installations or project paths, you can use Simulink implementation container (SIC) files instead.

**Compilation of source code files**

Source code files are compiled as efficiently as possible. This means that during recurring build processes, source code files are recompiled only if one of the following conditions is fulfilled:

- The source file has been updated since the last build process.
- The compiler options have been changed since the last build process.
- The unchanged source file directly or indirectly includes files which have been updated since the last build process.

**Automatically configured parameters**

For your convenience, ConfigurationDesk automatically configures the model parameters and parameters of the model code generator, such as the **Make command** and the **Embedded hardware parameters** in Simulink's **Hardware Implementation** dialog. Automatic configuration is performed during the build process according to the values needed by ConfigurationDesk, but only if the configuration of the behavior model differs from the configuration required by ConfigurationDesk. Information on automatically configured parameters is displayed in MATLAB's command window during the build process.

**Note**

The original configuration is not restored after model code generation.

**Model referencing supported**

ConfigurationDesk supports the Simulink/Simulink Coder model referencing feature. Model referencing is a mechanism that enables you to develop complex models in a distributed way. It lets you include models in other models by using Simulink Model blocks.

For limitations with model referencing, refer to [General Information on the Model Interface Blockset \(Model Interface Package for Simulink - Modeling Guide](#) (book icon)) and [Limitations Concerning the Build and Download Process](#) on page 779.

**Using Simulink models containing an A2L file fragment**

You can add a Simulink model containing an A2L fragment to a ConfigurationDesk application. If you do so, an A2L file is created for the application process to which the Simulink model is assigned during the build process. The A2L file contains the following elements:

- Model-specific variables provided by the A2L fragment of the related SIC file

- Variables for I/O functions and test automation
- DAQ raster names and the XCP service port number

You can assign these Simulink models to separate application processes and configure an XCP service for them. For more information, refer to [Specifying Options for the Build Process](#) on page 707.

**Model implementations that provide shared objects**

ConfigurationDesk supports shared objects that are provided by the following model implementation types:

- Simulink models (SLX files)
- Simulink implementation containers (SIC files)
- Bus simulation containers (BSC files)
- Functional Mock-up Units (FMU files)

During the build process, shared objects that are provided by a model implementation and match the registered platform are archived in the real-time application. Once the real-time application has been downloaded, the shared objects are available on the real-time system so that the model implementation can access them.

**Note**

If a model implementation container that is assigned to a multimodel application process provides shared objects, the shared objects must not contain references to the model implementation container code.

**Related topics****Basics**

Build Result Files of the Build Process.....	702
Details on the Build Process.....	716
Specifying Options for the Build Process.....	707

**HowTos**

How to Start the Build Process.....	719
-------------------------------------	-----

## Build Result Files of the Build Process

**Files generated during the build process**

Many files are created during the build process of a ConfigurationDesk real-time application.

The following build result files can be distinguished:

File type	Description
Real-time application file (RTA file)	The executable object file for processor boards. After the build process, the RTA file can be downloaded to the real-time hardware.
System description file (SDF file)	Provides information on the real-time application that is needed by the experiment software, such as information on the variable description file(s) associated to the real-time application. The SDF file references the related RTA file. If you select to download an SDF file to the real-time hardware, the related RTA file is downloaded.
Variable description file (TRC file) (generated for application processes that do not have an assigned V-ECU implementation)	Contains the descriptions of all the variables (signals and parameters) which can be accessed via the experiment software. In multicore applications, one TRC file is generated for each application process.
Map file (MAP file) (generated for application processes that do not have an assigned V-ECU implementation)	Maps symbolic names to physical addresses. It is used only by the experiment software. In multicore applications, one MAP file is generated for each application process.
CANCFG file (generated only if your ConfigurationDesk application contains Simulink models with blocks from the RTI CAN MultiMessage Blockset)	Contains configuration data (in XML format) for CAN communication.
LINCFG file (generated only if your ConfigurationDesk application contains Simulink models with blocks from the RTI LIN MultiMessage Blockset)	Contains configuration data (in XML format) for LIN communication.
FLXCFG file (generated only if your ConfigurationDesk application contains Simulink models with blocks from the FlexRay Configuration Blockset)	Contains configuration data (in XML format) for FlexRay communication.
EXPSWCFG file	Contains configuration data (in XML format) for automotive fieldbus communication.

File type	Description
A2L file (only for application processes that have a V-ECU implementation or a Simulink model/SIC file with Variable description file type = A2L assigned)	<p>The A2L file contains</p> <ul style="list-style-type: none"> <li>▪ Model-specific variables provided by the A2L fragment of the related Simulink model, SIC file or V-ECU implementation container</li> <li>▪ Variables for I/O functions and test automation</li> <li>▪ DAQ raster names and the XCP service port number</li> <li>▪ Information required to perform virtual bypassing via the RTI Bypass Blockset (only for application processes that contain a V-ECU implementation configured for virtual bypassing)</li> </ul> <p>The A2L file is named after the related application process. You can use these A2L files to integrate each model implementation separately as <i>XCP on Ethernet device</i> in ControlDesk.</p>
HEX file (initial data file, generated only for V-ECU implementations that support memory pages)	The HEX file contains information about the initial state of calibration variables.

---

**Application processes containing V-ECU implementations**

Application processes containing V-ECU implementations are not generated into the SDF file. If your ConfigurationDesk application contains only application processes that have V-ECU implementations assigned, no SDF file is generated. In this case, you must download the RTA file instead.

---

**Build results for application processes containing BSC files**

During the build process, ConfigurationDesk generates the following files for each application process that has a bus simulation container assigned:

- A TRC file
- An EXPFWCFG file

**TRC file entries** The TRC file created during the build process contains the following variable entries:

- Model variables in the Model Root group.
- Test automation access variables of the bus simulation in the Signal chain group.
- Task information variables in the Simulation and RTOS group.

The name of the TRC file is the same as the name of the application process the bus simulation container is assigned to. For details on the contents of a TRC file, refer to [Information on Variable Description Files \(TRC Files\) on page 722](#).

**EXPSWCFG file** The EXPSWCFG file that was created contains the bus configuration data. This file references the TRC file and can be used by an experiment software such as ControlDesk to access and manipulate bus communication.

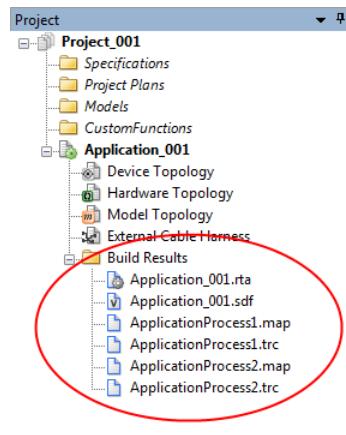
#### Naming of build results

Build results are named after the ConfigurationDesk application and the application process from which they originate. For example, if the name of your ConfigurationDesk application is Application\_001, and the application processes are ApplicationProcess1 and ApplicationProcess2, the build result files are named:

- ApplicationProcess1.trc
- ApplicationProcess2.trc
- ApplicationProcess1.map
- ApplicationProcess2.map
- Application\_001.rta
- Application\_001.sdf

#### Display and generation of build results

Build results are listed under the Build Results node of the respective application in the Project Manager and in the File Explorer. For example, in the Project Manager they are displayed as follows:



Regarding the display and generation of build results consider the following points:

- Build results are generated and displayed only if a build process was successful. An error during the download at the end of the build process does not affect the generation and display of the build results.
- Previous build results are overwritten each time a build process runs successfully.

- An unsuccessful build process does not affect previous build results. The build result files are not changed and are still displayed. This ensures that your build results are always consistent.
- If you rename a ConfigurationDesk application, its build results are still displayed with the old names until the next successful build process.
- You can remove the build results files from the active ConfigurationDesk application by using the Clear Build Results command from the context menu of the application or the Build Results folder. They are also removed from your file system.

---

**Saving and transferring build result files**

You can save build result files manually in a user-defined folder, for example, in a custom folder of the corresponding ConfigurationDesk project or application. RTA files can be loaded from user-defined folders in the same way as from the default folder.

You can also transfer build result files to other colleagues, for example, via e-mail. After the RTA file has been saved on the other PC, it can easily be loaded on the target hardware.

**Note**

To ensure consistency, you should always transfer the entire contents of the Build Results folder to your colleagues.

---

**Related topics****Basics**

<a href="#">Introducing the Build Process.....</a>	698
--	-----

**HowTos**

<a href="#">How to Start the Build Process.....</a>	719
---	-----

# Configuring the Build Process

## Introduction

To adapt the build process and its build results to your needs, you can customize it by configuring options.

## Where to go from here

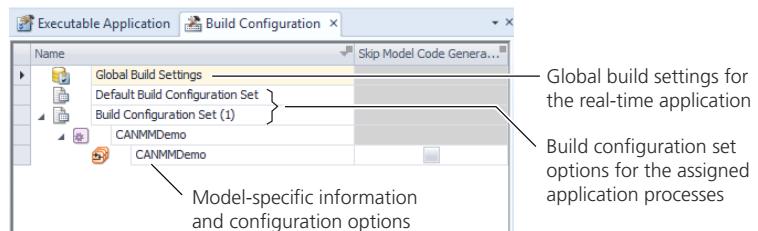
## Information in this section

<a href="#">Specifying Options for the Build Process</a>	707
<a href="#">Customizing the Build Process with User-Specific Files (For Simulink Behavior Models)</a>	713
<a href="#">How to Configure the Build Process</a>	714

## Specifying Options for the Build Process

### Available build options

ConfigurationDesk provides the Build Configuration table, which lets you specify build options in different subcategories. Refer to the following illustration:



### Global build settings

ConfigurationDesk lets you specify global build settings that affect the real-time application. The following categories of build options are available:

- Task startup behavior
- Build and download behavior
- TRC file options

**Task startup behavior** ConfigurationDesk lets you configure the task startup behavior of periodic tasks in real-time applications. This can be useful to avoid task overruns during initialization, which can occur due to a cold cache after the periodic task is started, for example. You can configure the following properties:

- Time-scaled period
- Time scale factor

**Build and download behavior** Before you start the build process, you must specify the build and download behavior for the real-time application of the active ConfigurationDesk application. The following options are available:

- Download real-time application after build
- Unload a loaded application
- Start real-time application

**Note**

- You can specify the default download behavior for real-time applications via the Configuration page of the ConfigurationDesk Options dialog. The specified settings apply to all newly added ConfigurationDesk applications. If you change the default download settings, existing ConfigurationDesk applications are not affected by the changes.
- The settings you specify for the build and download behavior in the Build Configuration table are saved with the ConfigurationDesk application and apply only to this application, i.e., other ConfigurationDesk applications and the default download behavior specified via the ConfigurationDesk Options dialog are not affected.

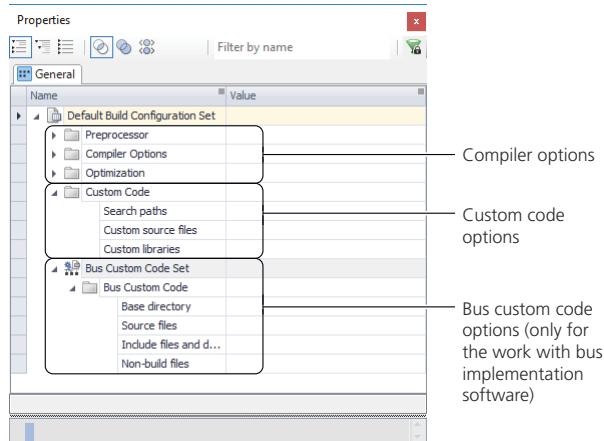
**TRC file options** If multiple models are assigned to the same application process, ConfigurationDesk automatically generates additional model groups in the variable description file (TRC file). If you want an additional model group in TRC files for application processes to which only one model is assigned, you can select the **Insert model name TRC file group** checkbox.

For more information on global build settings, refer to [Build Configuration Table \(ConfigurationDesk User Interface Reference\)](#).

---

## Build configuration set options

ConfigurationDesk provides build configuration sets for you to specify build options for specific application processes. You can create any number of build configuration sets with different build options and assign an application process to any one of them. This makes it easy to switch between different variants of compiler options, for example. Newly created application processes are automatically assigned to the default build configuration set. You can assign these application processes to a build configuration set that you created beforehand. Build configuration sets provide different categories of build options as shown in the following illustration:

**Tip**

In multicore applications, you can assign each application process (and thus the associated behavior model) to a different build configuration set. This is useful, for example, if you want to change the build options only for specific behavior models in a complex model topology, or if you want to specify different macros for different behavior models.

**Compiler options** Build configuration sets let you specify compiler options, such as preprocessor macros, C / C++ compiler options, and compiler optimization options. Depending on the platform you are using, specific optimization options are preset. For an overview of the available compiler options, refer to [Build Configuration Table \(ConfigurationDesk User Interface Reference\)](#).

**Note**

- To compile very large models in less time, it is sometimes necessary to reduce optimization options or to disable them completely.
  - ConfigurationDesk supports the use of Simulink Model Verification blocks. These blocks monitor individual signals of the real-time application and check them against a specified limit. When a signal crosses the limit, the Model Verification block either issues a warning or even stops the real-time application. ConfigurationDesk lets you specify how to treat Model Verification blocks in the behavior model during real-time simulation. Depending on the settings you make, one of the following actions is carried out:
    - You set the preprocessor option (macros to be defined) to **DOASSERTS**: If the assertion occurs, an error message is displayed and the real-time application is stopped.
    - You set the preprocessor option (macros to be defined) to **DOASSERTS** and **PRINT\_ASSERTS**: If the assertion occurs, a warning is displayed. The real-time application keeps running.
    - If none of the two preprocessor directives is set: Model Verification block assertions do not occur in the real-time application.
- For more information on Model Verification blocks, refer to the Simulink documentation.

**Custom code options** ConfigurationDesk lets you specify a list of search paths which are used by the make tool for searching source, header, and library files. In addition, ConfigurationDesk allows you to specify source files and libraries that must be linked to the real-time application. Search paths, source files, and libraries are lists separated by semicolons. You can enter them in the

edit field or click the  button to manage them in dialogs. For more information, refer to [Build Configuration Table \(ConfigurationDesk User Interface Reference\)](#).

The custom code folders are searched in the following order:

1. The folder of the currently processed file.
2. Custom source folders.
3. Model implementation folders.
4. System integration code.
5. System code folders.
6. Compiler folders.

If you work with Simulink behavior models, there are additional ways to customize the build process. Refer to [Customizing the Build Process with User-Specific Files \(For Simulink Behavior Models\)](#) on page 713.

**Bus custom code options (only for working with bus implementation software)** If you work with bus implementation software, such as the Bus Manager, the RTI CAN MultiMessage Blockset, or the FlexRay Configuration Package, you can use the bus custom code options to specify user code

implementations that must be included in the real-time application or in bus simulation containers. Refer to the following topics:

- [Working with User Code \(ConfigurationDesk Bus Manager Implementation Guide\)](#)
- [Secure Onboard Communication Page \(RTICANMM MainBlock\) \(RTI CAN MultiMessage Blockset Reference\)](#)
- [General Page \(FlexRay Configuration Tool Reference\)](#)

**Note**

If you do not work with bus implementation software, you can still specify bus custom code options. However, the specified options will be ignored during the build process.

For more information, refer to [Build Configuration Table \(ConfigurationDesk User Interface Reference\)](#).

---

**Information and configuration options for model implementations**

ConfigurationDesk lets you view properties for model implementations, such as the model location path. For Simulink behavior models, you can additionally specify a model initialization command or skip model code generation.

**Note**

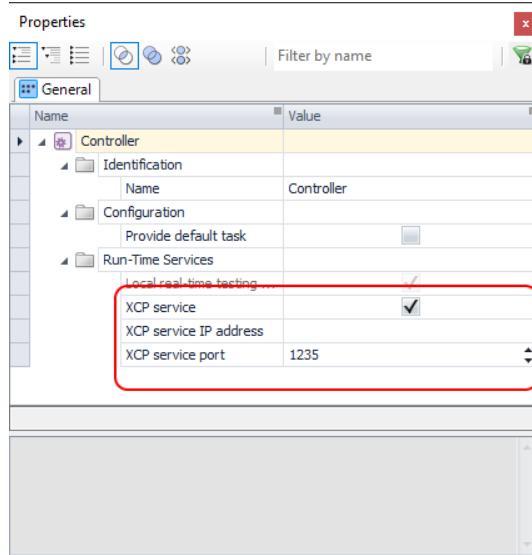
If you work with Simulink behavior models, you can also customize the build process and its build results by specifying variable description options in Simulink's Configuration Parameters dialog in the behavior model. For example, the variable description file (TRC file) can include initial parameter values that are needed for experiment tasks. These are not in the TRC file by default. In multimodel applications, you can specify the variable description options for each behavior model separately.

For more information, refer to [Build Configuration Table \(ConfigurationDesk User Interface Reference\)](#).

---

**Configuring the XCP service**

A Simulink model, SIC file, or V-ECU implementation added to an application can contain an A2L file fragment. You can assign these model implementations to separate application processes and configure an XCP service for them. Refer to the following illustration:



**XCP service** The XCP service checkbox lets you activate/deactivate the XCP service for the application process to which the model implementation is assigned. If you clear the checkbox, no CFD\_I\_XCP license is required to use the model implementation that provides an A2L file fragment in ConfigurationDesk. In this case, no A2L file is created for this application process during the build process.

**XCP service IP address** the XCP service IP address lets you specify the IP address of the XCP service in IPv4 notation: Each number of the local IP address must be in the range 0 ... 255, e.g., 192.168.140.6.

**XCP service port** If you want to configure the XCP service port number you have to specify it in the XCP service port property of the related application process. The default value is an integer value  $\geq 1024$ .

#### Note

The XCP service port number must be unique within your executable application. Otherwise, a conflict is shown in the Conflicts Viewer. If this happens, the build process is aborted with an error message.

#### Using C++11 custom code

You can use custom source code files that comply with the C++11 standard. For this purpose, you have to activate the C++11 standard for the relevant files. To do so, you must add the following line to the upper part of the respective source files or header files:

```
#pragma GCC diagnostic warning "-std=c++11"
```

**Note**

dSPACE has not fully quality-assured the C++11 functionality. Thus, dSPACE does not guarantee that all C++11 features are usable or work correctly. If you encounter any problems, contact dSPACE Support ([www.dspace.com/go/supportrequest](http://www.dspace.com/go/supportrequest)).

**Related topics**
**Basics**

Details on the Build Process.....	716
Using Multiple Model Implementations in the Same Application Process.....	491
Working with User Code (ConfigurationDesk Bus Manager Implementation Guide 	

**HowTos**

How to Configure the Build Process.....	714
---	-----

**References**

Build Configuration Table (ConfigurationDesk User Interface Reference 	
Configuration Page (ConfigurationDesk User Interface Reference 	

## Customizing the Build Process with User-Specific Files (For Simulink Behavior Models)

**User variable description files for Simulink behavior models**

ConfigurationDesk lets you add user variable description files to an application. The user-defined global variables contained in these files are considered during code generation. You can use them to adapt the build process and its build results to your requirements. The files are not changed during the build process. For details on user variable description files, refer to [Adapting the Generation of the Variable Description File](#) on page 736.

**Custom code features of Simulink Coder**

ConfigurationDesk supports the Simulink and Simulink® Coder™ features for custom code. You can customize the build process in ConfigurationDesk by adding your own C code (C and H files) and C++ code (CPP and HPP files) to an application.

**Note**

If you define a relative path for user C/C++ code files and user header files, they are searched for in the behavior model folder. For example, if you enter `./mySources` as the path, the files are searched for in `<behaviormodelfolder>/mySources`.

Custom code files are not changed during the build process. For details on custom code, refer to [Using Custom Code \(Model Interface Package for Simulink - Modeling Guide\)](#).

## How to Configure the Build Process

### Objective

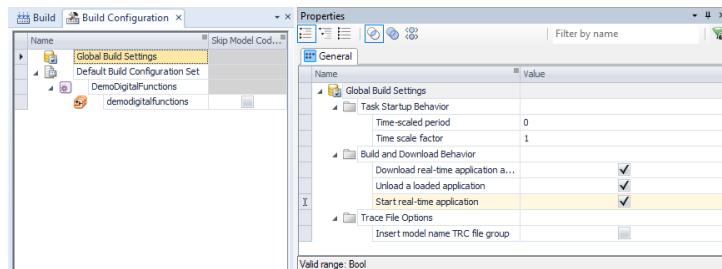
ConfigurationDesk allows you to specify options to configure the build process and its build results according to your requirements.

### Method

#### To configure the build process

- 1 Open the Build Configuration table.

The Build Configuration table lets you create build configuration sets, and specify compiler options and the task startup behavior.



- 2 From the context menu of the Build Configuration table, select New - Build Configuration Set.

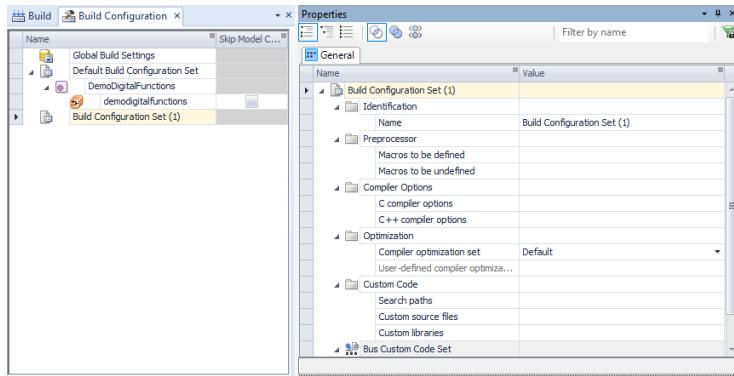
A new build configuration set named Build Configuration Set (1) is created.

**Tip**

- With the New - Multiple Build Configuration Sets context menu command, you can create several build configuration sets at once. New build configuration sets can be numbered consecutively. You can rename them. The new names must be unique. Otherwise, a conflict is generated and displayed in the Conflicts Viewer.
- You can change the settings of the default build configuration set.

**3** Select the Build Configuration Set (1) item.

The build configuration set options are available in the Properties Browser.



- 4** Configure the C/C++ compiler options of Build Configuration Set (1) to your needs.
- 5** Repeat steps 2 ... 4 to create additional build configuration sets, if needed.
- 6** Select the application process(es) from the Default Build Configuration Set node and drag it/them to the respective Build Configuration Set node.

#### Result

The options are set according to your entries and saved with your application. They are used in the next build process.

#### Next step

You can now start the build process. For instructions, refer to [How to Start the Build Process](#) on page 719.

#### Related topics

##### Basics

Details on the Build Process.....	716
Specifying Options for the Build Process.....	707

##### References

Build Configuration Table (ConfigurationDesk User Interface Reference	
---	--

# Starting the Build Process

**Introduction** You can build a real-time application with a single click in ConfigurationDesk, but there are some points you should note.

Where to go from here	Information in this section	
	Details on the Build Process.....	716
	How to Start the Build Process.....	719

## Details on the Build Process

**Starting the build process** You can start the build process only in ConfigurationDesk. You cannot start it in MATLAB. If MATLAB is not running when you start the build process, it is opened automatically, if required. For instructions, refer to [How to Start the Build Process](#) on page 719.

### Note

If you have connected several MATLAB installations to your dSPACE installation, you must select one of them as the preferred connection. For details, refer to [Introduction to Connecting a MATLAB Installation](#) ([Managing dSPACE Software Installations](#) 

**Stopping the build process** You can click the  button on the Home ribbon at any time. The current process step is completed and then the build process is aborted without new build result files being generated. Existing build result files are not overwritten.

**Configuring the download behavior** Before the build process, you can specify the download behavior in the Global Build Settings of the Build Configuration table:

- If you want to download a real-time application immediately after the build, the Download real-time application after build option must be selected. The application must be in the Matching platform connected state indicated by the green status bar. If the Unload a loaded application checkbox is selected, too, a running real-time application is automatically unloaded before the new application is downloaded. If the Start real-time application checkbox is selected, too, the real-time application is automatically started when the download is finished.

- If the real-time application does not have to be downloaded immediately after the build, the Download real-time application after build checkbox must be cleared. This disables the Unload a loaded application and Start real-time application options. For instructions on downloading a real-time application manually, refer to [How to Download a Real-Time Application](#) on page 748.

**Note**

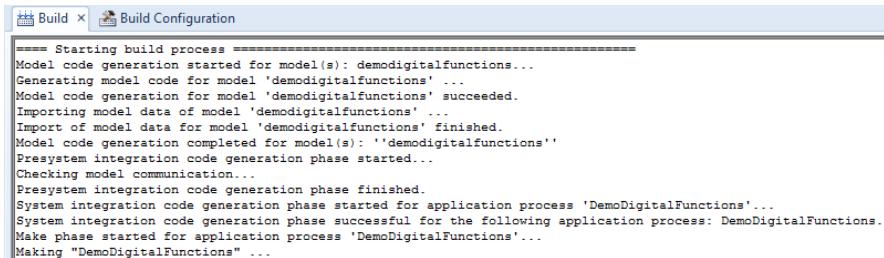
In multicore applications, ConfigurationDesk automatically assigns application processes to cores for execution during the download process. You cannot change this assignment. If your multicore application contains more application processes than the maximum allowed number (number of processor cores -1), the download is aborted with an error message.

**Tip**

The ConfigurationDesk Options dialog lets you specify the default behavior for downloading real-time applications. Refer to [Specifying Options for the Build Process](#) on page 707.

**Progress of the build process**

During the build process, messages in the Build Log Viewer indicate the progress of the build process in chronological order. This can look like this:



```
==== Starting build process =====
Model code generation started for model(s): demodigitalfunctions...
Generating model code for model 'demodigitalfunctions' ...
Model code generation for model 'demodigitalfunctions' succeeded.
Importing model data for model 'demodigitalfunctions' ...
Import of model data for model 'demodigitalfunctions' finished.
Model code generation completed for model(s): 'demodigitalfunctions'
Presystem integration code generation phase started...
Checking model communication...
Presystem integration code generation phase finished.
System integration code generation phase started for application process 'DemoDigitalFunctions'...
System integration code generation phase successful for the following application process: DemoDigitalFunctions.
Make phase started for application process 'DemoDigitalFunctions'...
Making "DemoDigitalFunctions" ...
```

In the Build Log Viewer, you can check out which step has been started and which was already performed, and which folders your build results are stored in. You can also see warnings and errors regarding the build process, and messages of the make tool and the compiler. Messages of the Simulink Coder are not displayed in the Build Log Viewer, but in MATLAB's command window.

**Conflicts in ConfigurationDesk applications**

It can happen that a ConfigurationDesk application involves conflicts such as missing hardware resources. Even in such cases, ConfigurationDesk always tries not to abort the build process, but to continue it whenever possible. The following table shows some possible conflicts and their effects on code generation. For a complete overview of possible conflicts, refer to [Conflicts \(ConfigurationDesk Conflicts\)](#).

Conflict	Description	Effects on code generation
Unassigned hardware resources or Missing hardware resources	Function blocks do not have channels assigned to them. or The missing channels are not contained in the hardware topology of your active application.	The I/O functionality is substituted, i.e. the initial values of the functions are used. Warnings are displayed during the build process.
Missing plug-ins	The function block types are not installed in the active dSPACE installation on your host PC.	The respective function blocks are not included in code generation. Warnings are displayed during the build process.
Unresolved model port blocks	The model port blocks are not contained in the behavior model which is linked to your active application.	The respective model port blocks are not included in code generation. This can also affect mapped function blocks. Warnings are displayed during the build process.
Missing configuration data	Configuration data is missing, for example, wavetable files are not configured.	For the respective function blocks, the missing data is substituted by default data during code generation, for example, a default wavetable is used. Warnings are displayed during the build process.
Inconsistent port width	The data widths of the mapped function and model ports do not match.	ConfigurationDesk considers only the cut-set of vector elements. The other elements are neglected.
Inconsistent data range	The data ranges of the mapped function and model ports are unequal.	ConfigurationDesk considers only the cut-range. The values that exceed this range are neglected.

**Note**

During the download process you are not informed about hardware resources that will be substituted when the real-time application is running.

---

**Copying messages for further use** ConfigurationDesk provides a context menu in the Build Log Viewer which lets you copy messages from the Build Log Viewer to the Clipboard for further use, for example, to display them in other tools.

**Related topics****Basics**

Introducing the Build Process.....	698
Specifying Options for the Build Process.....	707

**HowTos**

How to Configure the Build Process.....	714
---	-----

How to Start the Build Process..... 719

#### References

- [Build Log Viewer \(ConfigurationDesk User Interface Reference\)](#)
- [Start Build \(ConfigurationDesk User Interface Reference\)](#)
- [Stop Build \(ConfigurationDesk User Interface Reference\)](#)

## How to Start the Build Process

### Objective

To generate a real-time application, you must start a build process.

### Before starting

Before you start the build process, you can configure options. For details, refer to [Specifying Options for the Build Process](#) on page 707 and [Details on the Build Process](#) on page 716.

### Preconditions

The following preconditions must be fulfilled:

- The ConfigurationDesk application you want to build must be active.
- Conflicts of the Abort Build category must not exist.

### Method

#### To start the build process

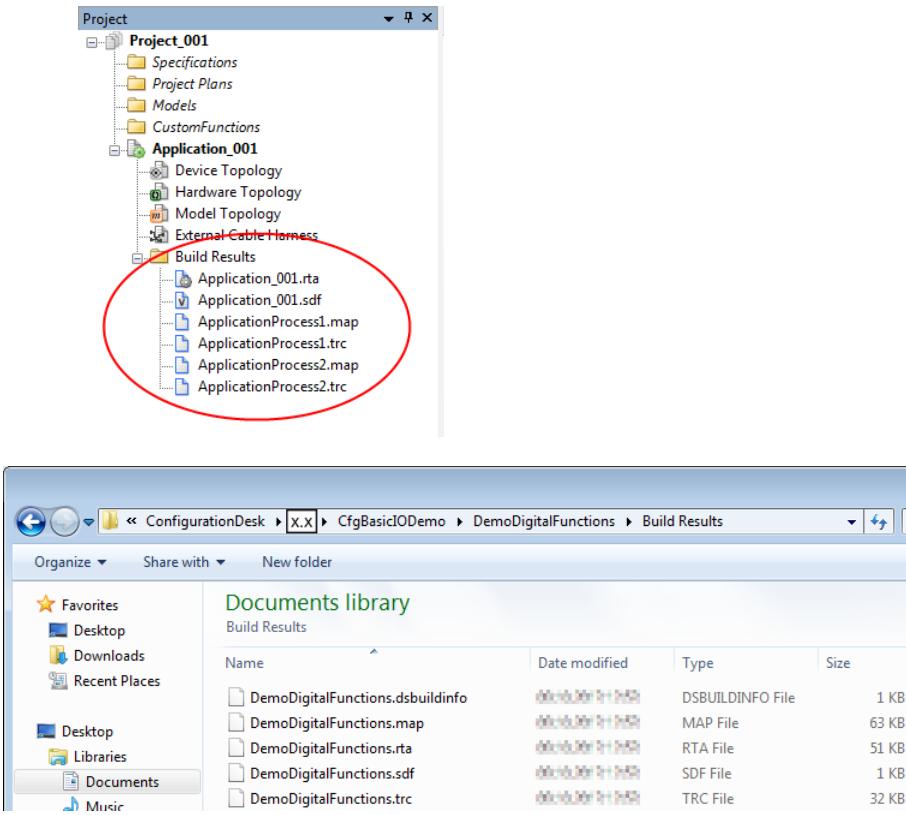
- 1 On the Home ribbon, click .

#### Tip

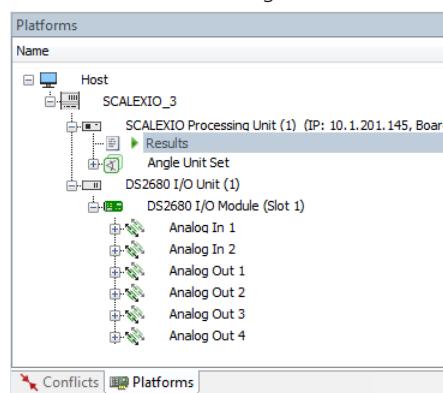
- You can also start the build process via the **Ctrl+Alt+B** shortcut key or via the Start Build command in the context menu of an active application in the Project Manager.
- If your ConfigurationDesk application is connected to a Simulink model, you can start the build process from Simulink using the Start ConfigurationDesk Build command from the ConfigurationDesk menu.

### Result

You have built a real-time application. The generated build result files are displayed in the Project Manager and in the File Explorer. In both cases they appear below the Build Results node of the active application. Refer to the following illustrations:



If the Download real-time application after build option is selected and your application is connected to hardware, the real-time application is also downloaded. The Platform Manager displays it below the Processing Unit item with the name of the respective RTA file. A red square next to the application's name in the Platform Manager indicates that it is stopped, a green arrow indicates that it is running.



**Note**

If ConfigurationDesk detects inconsistencies during the build or download process, you are informed about them in the Build Log Viewer, for example. The inconsistencies can cause ConfigurationDesk to abort the build or the download process.

**Next step**

If the real-time application was not automatically downloaded after the build, you can start the download process manually. For instructions, refer to [How to Download a Real-Time Application](#) on page 748.

**Related topics****Basics**

Application States of a Real-Time Application.....	753
Basics on Downloading Real-Time Applications.....	745
Introducing the Build Process.....	698

**HowTos**

How to Configure the Build Process.....	714
---	-----

**References**

<a href="#">Start Build (ConfigurationDesk User Interface Reference</a>	
<a href="#">Stop Build (ConfigurationDesk User Interface Reference</a>	

# Information on Variable Description Files (TRC Files)

**Introduction** The TRC file provides information on the variables of a real-time application that is required to connect variables to instruments in a layout of the experiment software, for example. It is an ASCII file that can either be generated automatically by ConfigurationDesk, or written manually. A variable description file is generated during the build process.

## Where to go from here

### Information in this section

Generating TRC Files.....	722
Structure of a TRC File for SCALEXIO or MicroAutoBox III Systems.....	724
Inclusion of Variables from the Simulink Model into the TRC File.....	731

# Generating TRC Files

## Basics on the Generation of TRC Files

### Introduction

A TRC file provides information on the variables that are used in an executable application. This variable description is a data interface between the executable application executed on dSPACE hardware or VEOS and the dSPACE experimentation software.

In the experimentation software, you get access to the variables by loading a system description file (SDF file). However, the variable description itself is stored in a trace file (TRC file), that is then also loaded. The following description refers to the contents of variable description files in TRC format.

### Generating and consuming software products

Build processes that include the generation of TRC files can be started via:

- RTI and RTI-MP
- ConfigurationDesk
- Model Interface Package for Simulink

The generated intermediate files are used by VEOS and ConfigurationDesk to create a platform-specific TRC file.

dSPACE products that need a generated TRC file to access variables in an executable application are, for example, ControlDesk, AutomationDesk, ModelDesk, and also custom applications based on dSPACE XIL API.

#### Data source of the variable description

When you build a simulation application, Simulink Coder provides information on the model's variables and parameters in an internal format. This data is used by dSPACE within the build process to generate a variable description in TRC file syntax.

The contents of a variable description depend on the following points:

- The way you have implemented and prepared your model. For example, variables in a subsystem can be excluded by using the subsystem omission tag (`DsVd0mit`).
- Simulink Coder version (represented by the MATLAB Release number).
- Configuration settings for code generation, refer to [Simulink configuration settings](#) on page 723.

#### Configuration settings for the TRC file generation

The contents of the TRC file can be configured by various settings. The most relevant settings are described below.

**Simulink configuration settings** The behavior of Simulink Coder can be influenced generally by the **Code Generation** options, that you can find in the model's **Configuration Parameters**. Also the optimization settings, such as the **Default parameter behavior** setting or the **Signal storage reuse** setting, have a direct effect on the available variables in the generated code and therefore on the TRC file generation.

##### Note

Not all configuration settings lead to the expected behavior, for example, variables can be missing in the TRC file as a result of the Simulink Coder internal optimizations.

**dSPACE configuration settings** If you use a system target file for a dSPACE platform, you can use the additional code generation options in the model's **Configuration Parameters**. Especially the **DSRT variable description file** options page provides settings to configure the contents of a generated TRC file. For further information, refer to [Adapting the Generation of the Variable Description File](#) on page 736.

# Structure of a TRC File for SCALEXIO or MicroAutoBox III Systems

## Introduction

The TRC file variables for SCALEXIO or MicroAutoBox III systems are structured in different groups.

## Where to go from here

## Information in this section

Available TRC File Variable Entries and Their Locations in the TRC File.....	724
Simulation and RTOS Group.....	726
Signal Chain Group.....	727
Diagnostics Group.....	729
XIL API/EESPort Group.....	730

## Available TRC File Variable Entries and Their Locations in the TRC File

## Introduction

To access the variables in your experiment or test software, it is helpful to know where you can find them in the variable description file (TRC file).

## TRC file structure

The TRC file of a real-time application is structured according to the following groups:

- Simulation and RTOS group
- Variable groups from the behavior model
- Signal Chain group
- Diagnostics group
- XIL API/EESPort group (available as of dSPACE Release 2016-A)

**Simulation and RTOS** The Simulation and RTOS group contains variables related to simulation and the real-time operating system. Refer to [Simulation and RTOS Group](#) on page 726.

**Variable groups from the behavior model** In the TRC file, the variables of the behavior model are organized in the following groups:

- Model Root
- Tunable Parameters
- State Machine Data

- Labels
- BusSystems
- User Variables from <model>\_usr.trc

For more information on the group contents, refer to [Inclusion of Variables from the Simulink Model into the TRC File](#) on page 731.

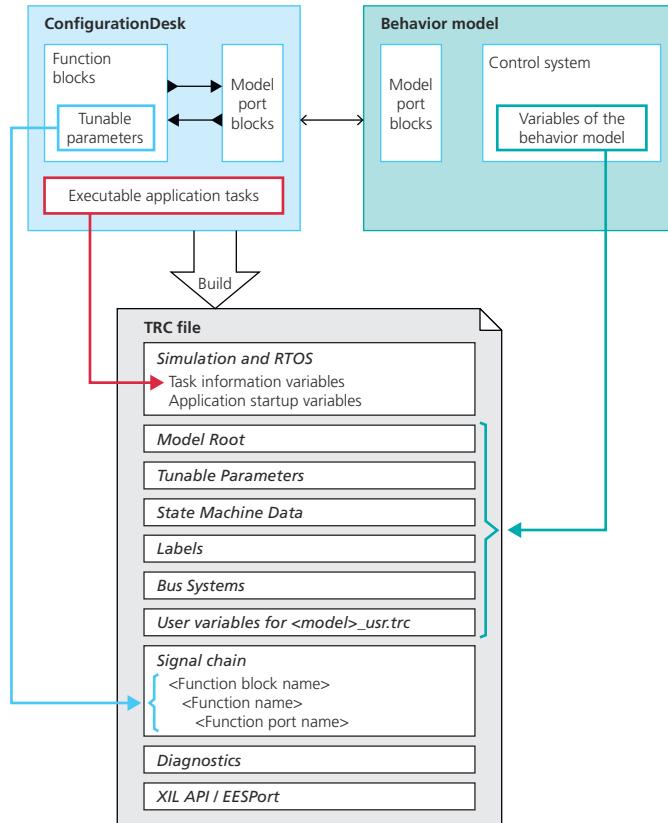
**Signal Chain** The Signal Chain group contains all the variables of the I/O functionality in ConfigurationDesk. It also contains variables of function ports that are configured for test automation support. If you enabled test automation support for a function port, the generated variable description file contains additional variables that can be accessed by the experiment and test software. These variables are generated in a subgroup of the affected function block. Refer to [Signal Chain Group](#) on page 727.

**Diagnostics** The Diagnostics group contains variables related to electronic fuses and the failure simulation components of the target system. Refer to [Diagnostics Group](#) on page 729.

**XIL API/EESPort** (Available as of dSPACE Release 2016-A) The XIL API/EESPort group contains variables for monitoring the switching behavior of electrical error simulation hardware. Refer to [XIL API/EESPort Group](#) on page 730.

**Graphical overview**

The following illustration gives an overview of the TRC file variables of a real-time application:



For multicore real-time applications and multiprocessing unit applications ConfigurationDesk creates one TRC file for each application process.

## Simulation and RTOS Group

### Introduction

The **Simulation and RTOS** group contains variables related to simulation and the real-time operating system.

### Task Information Variables

The **Task Information Variables** group contains the variables which you can use to obtain information on the tasks of the running real-time application. A real-time application can have several tasks structured in a group. Each task group has the following variables.

Variable	Description
Overrun Count	This read-only variable of a task counts the total number of task overruns that occurred for it.

Variable	Description
Task Call Counter	This read-only variable of a task is incremented whenever the task is called and executed.
Task Turnaround Time	This read-only variable shows the turnaround time, which is the time that passes between the triggering of the task and the end of its execution. It includes: <ul style="list-style-type: none"> <li>▪ The time it takes to execute the functional code of the task</li> <li>▪ The time it takes to trace variables with the data acquisition service of that task</li> <li>▪ The time the task's execution is interrupted by the execution of other tasks with higher priorities.</li> </ul> The turnaround time does not include the task switching time. The turnaround time is specified in seconds.

**Application Startup Variables**

The Application Startup Variables group contains the variables which you can use to specify parameters for the startup behavior of the real-time application.

Variable	Description
Scaled Time Interval	A period $[0; T]$ where $T \geq 0.0$ seconds. This period, including all the trigger times of periodic tasks within it, is scaled by the Scaling Factor, which must be $\geq 1.0$ . The result is that during the scaled time period $[0; T * \text{Scaling Factor}]$ the periodic tasks are not executed in real time, and have a longer time interval to complete their computations. This is useful to avoid task overruns in periodic tasks during the first simulation steps that result from cold cache, one-time initializations, etc.
Scaling Factor	The factor for scaling the time interval at the beginning of a simulation. The time scale factor must be greater or equal 1.0

## Signal Chain Group

**Introduction**

The Signal Chain group contains all the variables of the I/O functionality in ConfigurationDesk and can be accessed by the experiment and test software. These are the tunable parameters of the functions blocks and function ports, and the variables for test automation.

### Structure of the Signal Chain group

The **Signal Chain** group is structured on the basis of the function block types. This is the same structure as the Function Browser shows the function blocks.

Inside the group, you can find the function blocks used in your ConfigurationDesk application. A **Signal Chain** group exists only if you have instantiated a function block type in your ConfigurationDesk application, and the variables belong to the function block providing variables that can be accessed by the experiment and test software, for example, for test automation.

### Structure for a function block in the TRC file

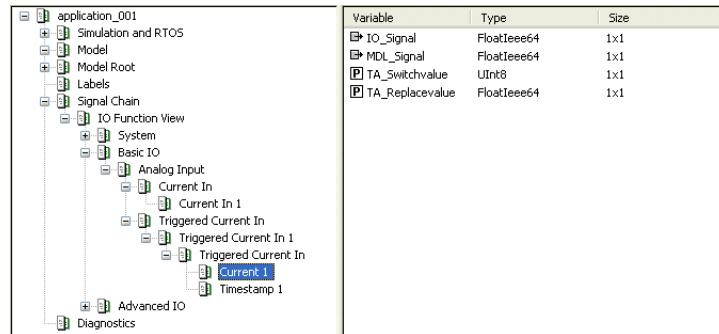
If the function block has variables which can be accessed by the experiment and test software, the TRC file contains a corresponding group. The name is equal to the name of the function block name specified in ConfigurationDesk. It is included under the function block type as described above.

The following shows the structure of a block group. The angle brackets (<>) mark placeholders. If subelements do not exist, the corresponding entries are omitted.

```
<block name>
  - <tunable and initialization parameters of the block>
  <function name>
    - <tunable and initialization parameters of the function>
    <function signal name>
      - <tunable and initialization parameters of the signal>
        IO_Signal
        MDL_Signal
        TA_Switchvalue
        TA_Replacevalue
```

If function ports of the function block provide variables for experiment and test software, for example, for test automation or tunable parameters, additional groups are inserted inside the group of the instantiated function. The names of the groups correspond to the signal names. For an overview of the variables, refer to the documentation of the function block in the [ConfigurationDesk I/O Function Implementation Guide](#).

**Example** The following illustration shows an example of an entry in a TRC file:



**Related topics****Basics**

Categories of Configurable Parameters.....	320
Configuring Test Automation Support (ConfigurationDesk I/O Function Implementation Guide 	

## Diagnostics Group

**Introduction**

The **Diagnostics** group contains all the variables necessary for a diagnostics of SCALEXIO or MicroAutoBox III systems.

**Fuse variables**

The **Diagnostics** group contains the following variables related to the electronic fuses of the SCALEXIO or MicroAutoBox III hardware.

Variable	Description
<b>Fuse Actual Status</b>	Displays the current status of all fuses of a SCALEXIO or MicroAutoBox III system: <ul style="list-style-type: none"> <li>▪ 0: All fuses are intact</li> <li>▪ 1: At least one fuse is open. This can happen, for example, caused by an erroneous wiring of the external cable harness or during failure simulation.</li> </ul>
<b>Fuse Restore Request</b>	Use the variable to restore the electronic fuses of a SCALEXIO or MicroAutoBox III system. If set to 1, all electronic fuses are restored.

**Failure simulation variables**

The **Failure Simulation** subgroup contains the following variables related to the electronic failure simulation of the SCALEXIO or MicroAutoBox III system. The variables allow the tracing of the internal states of the failure insertion unit (FIU):

Variable	Description
<b>Client connected</b>	Indicates if a failure simulation client (for example, ControlDesk) is connected to the FIU. <ul style="list-style-type: none"> <li>▪ 0: No failure simulation client is connected.</li> <li>▪ 1: A connection to a failure simulation client is established.</li> </ul>
<b>Failure activated</b>	Indicates if a failure is activated. <ul style="list-style-type: none"> <li>▪ 0: No failure is activated.</li> <li>▪ 1: Failure is activated.</li> </ul>

Variable	Description
Failure configured	Indicates the current failure configuration status of the FIU. <ul style="list-style-type: none"><li>▪ 0: No failure is configured.</li><li>▪ 1: At least one failure is configured.</li></ul>
Failure sequence count	Displays the number of state changes of the FIU. The failure sequence count is increased each time a failure is added or removed. This allows the tracing of multiple failures which are activated at the same time. <ul style="list-style-type: none"><li>▪ 0: No failure is configured.</li><li>▪ 1..65535: Failure sequence n is currently active.</li></ul>
Relay switching	The variable allows you to trace the switching times of the electromechanical relays of the FIU and failure routing units (FRU). It indicates if the status of the failure simulation system is stable or instable. <ul style="list-style-type: none"><li>▪ 0: The failure simulation system is stable.</li><li>▪ 1: The failure simulation system is currently busy.</li></ul>

## XIL API/EESPort Group

### Introduction

When you simulate electrical errors by using an EESPort implementation via the ControlDesk user interface or via a dSPACE XIL API application, you are able to monitor the switching behavior of your electrical error simulation hardware.

#### Note

The XIL API/EESPort group is available as of dSPACE Release 2016-A.

### XIL API/EESPort variables

The generated variable description file provides the following variables in the XIL API/EESPort group:

- **Active ErrorSet**
- **Error Activated**
- **Error Switching**
- **Flags**
- **Trigger**

For multiprocessor models, the XIL API group is generated into each submodel section separately. For more information, refer to [Monitoring the Switching Behavior of Electrical Error Simulation Hardware \(dSPACE XIL API Implementation Guide\)](#).

# Inclusion of Variables from the Simulink Model into the TRC File

## Simulink variables

The TRC file contains information on the variables and signals of a Simulink model.

## Where to go from here

## Information in this section

Details on the Variable Groups of the Behavior Model.....	731
Methods of Including Variables of the Behavior Model in the TRC File.....	734
Adapting the Generation of the Variable Description File.....	736
Conditions for the Inclusion of Variables from the Simulink Model into the TRC File.....	740
Conditions for the Inclusion of Variables From Referenced Models.....	742

## Details on the Variable Groups of the Behavior Model

### Introduction

The variables of the behavior model are organized in the following groups, which are described in detail below:

- Model Root
- Tunable Parameters
- State Machine Data
- Labels
- BusSystems
- User Variables from <model>\_usr.trc

### Note

The generation of groups and their contents are influenced by the settings on the DSRT variable description file options page of a model's Configuration Parameters dialog and the contents of a model. Note that a group might be empty or omitted.

<b>Model Root</b>	<p>This group contains the block and subsystem groups and the labeled signals for the top level of the Simulink model. The hierarchy of the block and subsystem groups in the Model Root group reflects the hierarchy of the Simulink model. Variables from the blocks and subsystems are located on lower hierarchical levels.</p> <ul style="list-style-type: none"> <li>▪ Each subsystem group contains the subsystem's outputs, and the block groups for the blocks within the subsystem. Masked subsystems might contain parameters.</li> <li>▪ Each block group contains the variables of one block, for example, outputs and parameters.</li> <li>▪ Block groups for Stateflow® charts contain the outputs to Simulink, Stateflow test points and parameters.</li> </ul> <p>All Stateflow charts together form the state machine, which can have global data. This data is available via the State Machine Data group (see below).</p>
<b>Tunable Parameters</b>	<p>This group contains the global parameters of the behavior model.</p> <div style="background-color: #f0f0f0; padding: 10px;"> <p><b>Note</b></p> <ul style="list-style-type: none"> <li>▪ MATLAB workspace variables and Simulink.Parameter objects, which are used as block parameters in the model, are generated as global variables in the Tunable Parameters group. Internal optimizations during code generation might be the reason that a variable will not be generated into the variable description.</li> <li>▪ Structured workspace variables and Simulink.Parameter objects that are used as block parameters in the model are generated as global structured parameters in the Tunable Parameters group. The structure has to fulfill the Simulink Coder conditions for a tunable structured parameter.</li> <li>▪ For model referencing hierarchies, a Tunable Parameters group is generated only for the top-level model. Global parameters referenced in referenced models are also generated into the Tunable Parameters group of the top-level model.</li> </ul> </div>
<b>State Machine Data</b>	<p>This group contains data objects of Exported scope defined at the State Machine level. The group is created only if the model contains Stateflow charts. Data objects of Output, Parameter or Local scopes defined for individual Stateflow charts are available via the Model Root group. Depending on the MATLAB Release used, Data objects of Local and Output scopes are generated into the TRC file only if the value attribute Test point is set.</p>
<b>Labels</b>	<p>This group contains all labeled signals. It appears only if labeled signals are found in the Simulink model. In this group, the model hierarchy is ignored to give quick access to important signals. The labeled signals are also available in the subsystem groups from which they originate.</p>

This group is only available if the **Include signal labels** checkbox on the **DSRT variable description file** options page of a model's Configuration Parameters dialog is selected.

**Multiple occurrences of labels with the same name** As of dSPACE Release 2019-B, multiple labels with the same name no longer have an entry in the TRC file by default. If required, you can temporarily activate TRC file entries for multiple labels.

Type the following commands in the MATLAB Command Window:

- To activate the option

```
ds_trc_multiplelabeloccurrence('set', 1)
```

- To reset the option to the default behavior

```
ds_trc_multiplelabeloccurrence('set', 0)
```

#### Note

- We strongly recommend not to activate this option.  
Using multiple labels with the same name in your variable description is on your own risk.
- The option to activate TRC file entries for multiple labels will no longer be available in one of the next dSPACE Releases. It will be available only temporarily for migrating existing models or scripts that handle multiple labels.

## BusSystems

This group contains all the bus systems modeled in the Simulink model. It contains all the signals which are sent or received by the dSPACE real-time hardware.

**CAN bus** For details on the BusSystems group structured for a CAN bus, refer to [TRC Options Page \(RTICANMM MainBlock\) \(RTI CAN MultiMessage Blockset Reference\)](#).

**FlexRay bus** For details on the BusSystems group structured for a FlexRay bus, refer to [Using the Generated TRC File of PDU-Based Modeling in ControlDesk \(Model Interface Package for Simulink - Modeling Guide\)](#).

**LIN bus** For details on the BusSystems group structured for a LIN bus, refer to [TRC Options Page \(RTILINMM MainSetup\) \(RTI LIN MultiMessage Blockset Reference\)](#).

## User Variables from <model>\_usr.trc

This group is filled with the user-specific contents of the user variable description file (<model>\_usr.trc) whenever code is generated from the Simulink model. It is created only if a user variable description file is available in the working folder of the model.

---

<b>Rules and Limitations</b>	<p>There are certain rules and limitations regarding the inclusion variables from the Simulink model. Refer to the following topics:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">Conditions for the Inclusion of Variables from the Simulink Model into the TRC File on page 740</a></li> <li>▪ <a href="#">Conditions for the Inclusion of Variables From Referenced Models on page 742</a></li> <li>▪ <a href="#">Limitations Regarding Simulink Variables in the TRC File on page 776</a></li> </ul>
------------------------------	--

## Methods of Including Variables of the Behavior Model in the TRC File

---

<b>Introduction</b>	<p>There are different options for adapting the contents of the TRC file.</p>
<b>Simulink options and their effects on the TRC file</b>	<p>Simulink provides the following code optimization options in the <b>Model Configuration Parameters</b> dialog to reduce memory consumption:</p> <p><b>Default parameter behavior</b> If you set the <b>Default parameter behavior</b> option in the <b>Code Generation</b> ⇒ <b>Optimization</b> dialog to <b>Inlined</b>, the numerical values of block parameters are used in the generated code. In this case, Simulink Coder does not generate any global variables for the block parameters, unless you specify them as tunable parameters or create <b>Simulink.Parameter</b> objects with a global storage class for them.</p> <p><b>Signal Storage Reuse</b> If you activate the <b>Signal Storage Reuse</b> option in the <b>Simulation Target</b> dialog, Simulink Coder does not always use a separate variable for each block output but attempts to assign a single buffer to several block outputs. This means that in general, the signals are not available in the TRC file unless you make specific settings for them.</p>
<b>Adapting parameter and signal entries in the TRC file</b>	<p>You can generate variables for block parameters or signals in the TRC file even if <b>Default parameter behavior</b> is set to <b>Inlined</b> and/or <b>Signal Storage Reuse</b> is activated.</p> <p><b>Specifying which parameters are available in the TRC file</b> When you set the <b>Default parameter behavior</b> optimization option to <b>Inlined</b> there are methods for you to specify block parameters that must nevertheless be available in the TRC file (including their names and properties). Set <b>Default parameter behavior</b> to <b>Inlined</b> and use one of the following methods:</p> <ul style="list-style-type: none"> <li>▪ Create variables in the MATLAB workspace, and declare them as tunable parameters via Simulink's <b>Model Parameter Configuration</b> dialog.</li> <li>▪ Create <b>Simulink.Parameter</b> objects with the <b>ExportedGlobal</b>, <b>ImportedExtern</b>, <b>ImportedExternPointer</b>, or <b>ModelDefault</b> storage class in the MATLAB workspace, and reference them in the model. For details, refer to <a href="#">Rules for Simulink.Signal and Simulink.Parameter objects on page 740</a>. You must use this method for referenced models.</li> </ul>

In both cases, the block parameter entries are displayed in the Tunable Parameters group of the generated TRC file. The complete model hierarchy is still available in the Model Root group.

The block groups within the Model Root group contain references to the global parameters in the Tunable Parameters group.

**Specifying which signals are available in the TRC file** To make sure that an entry for a block output signal is generated into the TRC file, even if the Signal Storage Reuse option is enabled, you can use one of the following methods:

- Enable the Test Point option for the signal.
- Specify a label and a global storage class for the signal.
- Specify a label for the signal and create an appropriate `Simulink.Signal` object with the `ExportedGlobal`, `ImportedExtern`, `ImportedExternPointer`, or `ModelDefault` storage class in the MATLAB workspace (for details, refer to [Rules for Simulink.Signal and Simulink.Parameter objects](#) on page 740). In addition, you must link the signal to the `Simulink.Signal` object. To do so, you must specify one of the following Simulink options:
  - Either select the `Signal must resolve to Simulink.Signal` object checkbox in the signal's `Signal Properties` dialog.
  - Or set the `Signal resolution` option to `Explicit` and `implicit` on the `Data Validity` page of the `Diagnostics` dialog.

---

#### Displaying TRC file entries as a flat list

With Simulink's code optimization options described above, the model hierarchy is still available in the Model Root group of the TRC file. If you do not need the model hierarchy, you can select the `Include only Simulink.Parameter and Simulink.Signal objects with global storage class` option (Configuration Parameters dialog, - Code Generation - DSRT variable description file options page). With this option activated, the TRC file provides only the Labels group showing the `Simulink.Signal` objects, and the Tunable Parameters group showing `Simulink.Parameter` objects with the `ExportedGlobal`, `ImportedExtern`, or `ImportedExternPointer` storage class as a flat list. It does not contain the entire model hierarchy. Selecting this option can reduce the time needed for the build process.

##### Note

If the `Include only Simulink.Parameter and Simulink.Signal objects with global storage class` option is enabled, `LookupTableData` entries are not available.

---

#### Content of the TRC file groups depending on the code option settings

You can use the DSRT variable description file options to control which variables are available in the TRC file.

Include only Signal.Parameter and Simulink.Signal objects with global storage class = ON:

- The Tunable Parameters group contains entries for all Simulink.Parameter objects with a global storage class.
- The Labels group contains entries for all Simulink.Signal objects with a global storage class.

Include only Signal.Parameter and Simulink.Signal objects with global storage class = OFF:

- The Model Root group contains the model hierarchy with:
  - Entries for the available signals
  - Entries for the available block parameters
  - Further entries such as mask parameters, states, and derivatives.
- The Tunable Parameters group contains entries for all the available global parameters.
- The Labels group contains entries for all the labeled signals if **Include signal labels** is selected.

#### Note

- Amongst other factors, it depends on the Simulink Coder optimization settings, such as Signal storage reuse, Inline invariant signals, and Enable local block outputs, if a signal entry is available in the TRC file (see below). To make sure that the Simulink Coder generates a separate entry for an output signal, you can designate any signal in a model as a test point, or specify a global storage class for the signal, either directly or via a Simulink.Parameter object.
- It depends on the setting of the Default parameter behavior option, if a block parameter is available in the TRC file. If Tunable is selected, the block parameters are generated as variables. If Inlined is selected, the numeric values of the block parameters are used in the generated C code, which makes them non-modifiable. However, you can use Simulink.Parameter objects or configure a variable as tunable in the Model Configuration Parameters dialog to make individual block parameters available in the TRC file.

For more information, refer to the Simulink and Simulink Coder documentation.

## Adapting the Generation of the Variable Description File

### Introduction

You can adapt the contents of the variable description file generated by ConfigurationDesk by adding and excluding variables using specific options. You can specify the variable description file options in the behavior model's Configuration Parameters dialog on the Code Generation - DSRT variable description file options page.

**Note**

The System target file parameter must be set to `dsrt.tlc` or `dsrt64.tlc` on the Code Generation page.

## Variable description file options in Simulink

The following options are available for the DSRT system target:

**Variable description file format** Lets you select a format for the variable description file that is created during the build process. The following formats are available:

- TRC
- A2L

**Note**

The following options only apply to the generation of TRC files.

**Include only Simulink.Parameter and Simulink.Signal objects with global storage class** Indicates that only parameters and signals are included in the variable description file which reference a `Simulink.Parameter` or a `Simulink.Signal` object in the MATLAB workspace. For more information, refer to [Methods of Including Variables of the Behavior Model in the TRC File](#) on page 734.

In the variable description file, Description, DocUnits, Min and Max entries of `Simulink.Signal` objects and `Simulink.Parameter` objects are available.

**Note**

When you use model referencing, this setting must be the same for all referenced models.

**Include signal labels** Indicates that the signal labels of the model are available in the variable description file.

- If selected, the signal labels of the model are generated into the variable description file.
- If cleared, the signal labels of the model are not generated into the variable description file. For large models especially, this might significantly reduce the time needed for the code generation process (default).

**Include virtual blocks** Indicates that the block outputs of virtual blocks (for example, From blocks) are available in the variable description file.

- If selected, the virtual blocks are available in the variable description file.
- If cleared, the virtual blocks are not available in the variable description file. This might significantly reduce the time needed for the code generation process (default).

If you just want to have the outputs of a few virtual blocks available in the variable description file, you can add test points to the relevant signals.

**Include states** Lets you make the states of blocks available in the variable description file.

- If selected, the block states are generated into the variable description file.
- If cleared, the block states are not generated into the variable description file (default).

**Include derivatives** Lets you make the derivatives of blocks available in the variable description file.

- If selected, the derivatives of blocks are generated into the variable description file.
- If cleared, the derivatives of blocks are not generated into the variable description file (default).

#### Note

The **Include states** and **Include Derivatives** options are not supported for model referencing, neither for top-level models nor for referenced ones.

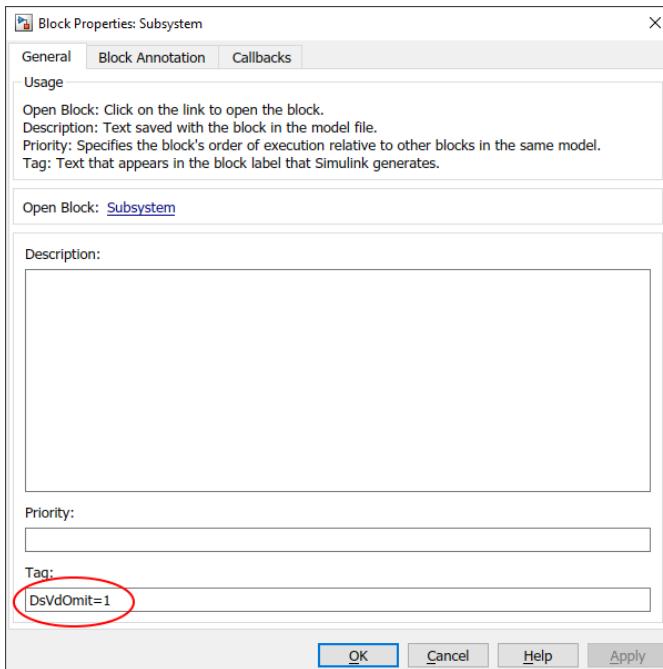
**Include initial parameter values** Lets you make the initial parameter values available in the variable description file. Initial parameter values are needed to carry out offline calibration tasks.

- If selected, the initial parameter values are generated into the variable description file.
- If cleared, the initial parameters are not generated into the variable description file.

#### Note

Deactivating this option might significantly reduce the time needed for code generation.

**Apply subsystem omission tags** Indicates that `DsVd0mit` tags are evaluated. These tags can be used to exclude the variables of specific subsystems from being generated into the variable description file.

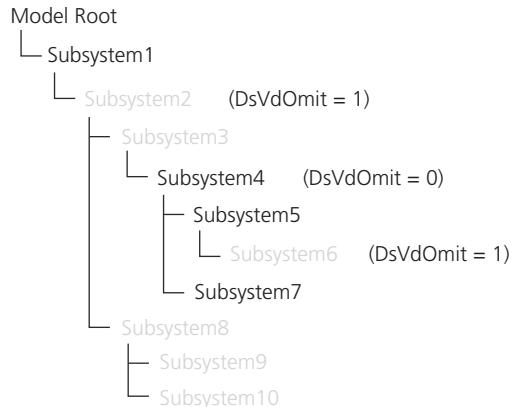


If the **Apply subsystem omission tags** checkbox is cleared, settings of the **DsVdOmit** tag are ignored for all subsystems in the model. All subsystems and their contents appear in the generated variable description file. If selected, **DsVdOmit** tags of subsystems have the following effects on the generated variable description file:

Setting	Description
<b>DsVdOmit</b> tag is set to 1	The subsystem contents including all blocks beneath this subsystem do not appear in the generated variable description file. Use <code>set_param(gcb, 'Tag', 'DsVdOmit=1')</code> .
<b>DsVdOmit</b> tag is set to 0	The subsystem contents including all blocks beneath this subsystem do appear in the generated variable description file, even if a subsystem above this subsystem has set the <b>DsVdOmit</b> tag to 1. Use <code>set_param(gcb, 'Tag', 'DsVdOmit=0')</code> .
<b>DsVdOmit</b> tag is set to -1	The subsystem contents including all blocks beneath this subsystem do not appear in the generated variable description file. <b>DsVdOmit</b> settings of subsystems included in this subsystem are ignored. Use <code>set_param(gcb, 'Tag', 'DsVdOmit=-1')</code> .

Exclusion is applied recursively through the model hierarchy. To re-include subsystems, you can enter **DsVdOmit=0**. You can set the Omit flag by using workspace variables. For example, if **WSVar1** is a workspace variable, you can use `set_param(subsystemHandle, 'Tag', 'DsVdOmit=$(WSVar1)')` to let the Omit tag value being evaluated during build process.

Example: If you set the **Apply subsystem omission tags** option, the variables of the grayed subsystems in the illustration below are not generated into the variable description file.



#### **Adding custom code variables to the variable description file**

If you use custom code, the variables in it are initially not accessible to the experiment software. To make the global variables accessible, you must provide an additional variable description file. When writing a user variable description file, use the syntax described in [ConfigurationDesk Syntax of the TRC File](#), and name it `<model>_usr.trc`. During the build process, the user file is inserted into the main variable description file. It must be created before the build process is started. It has to be located in the working folder of the behavior model. You can write variable description files also for each referenced model in your behavior model.

## Conditions for the Inclusion of Variables from the Simulink Model into the TRC File

### **Introduction**

Not all variables of a Simulink model are always available in the TRC file. The following conditions define whether a variable and its properties are included.

### **Rules for `Simulink.Signal` and `Simulink.Parameter` objects**

`Simulink.Parameter` and `Simulink.Signal` objects can be used to specify the characteristics of tunable parameters and labeled signals in a model.

Parameters and signals specified by `Simulink.Parameter` and `Simulink.Signal` objects are available in the TRC file by default.

`Simulink.Parameter` and `Simulink.Signal` objects are represented in the variable description file with the following properties:

- Description
- DocUnits  
Is displayed as *Unit* in your experiment software.
- Min and Max (represented by a range in the TRC file)

Note the following preconditions and restrictions:

- The properties for labels in the Labels group and the subsystem groups are used, if the labeled signal references a `Simulink.Signal` object of global storage class (`ExportedGlobal`, `ImportedExtern`, `ImportedExternPointer`). At the entries for the block output signals, these properties are omitted.

#### Note

The *Include Labels* checkbox must be selected. It is not selected per default.

- The properties for parameters in the Tunable Parameters group are used, if the parameters are specified by `Simulink.Parameter` objects of global storage class (`ExportedGlobal`, `ImportedExtern`, `ImportedExternPointer`).
- The range for Min and Max is generated only if both values are available.
- The range for Min and Max is not generated for fixed-point data.

---

#### Rules for virtual Simulink blocks

Virtual blocks are not available in the TRC file by default. If you want to include them, you can select the *Include virtual blocks* option. For details on this option refer to [Adapting the Generation of the Variable Description File](#) on page 736.

---

#### Rules for virtual signals

Entries for virtual signals, e.g., from a Mux block or a Bus Creator block, are not generated into the TRC file.

---

#### Rules for structured variables

Structured variables, such as non-virtual buses or tunable structured parameters, are generated into the code and represented in the variable description as a `struct` element.

---

#### Rules for complex variables

Code generated with Simulink Coder stores the real and imaginary parts of complex variables separately. The display of the variable dimensions in ControlDesk is different from the one displayed in MATLAB or Simulink, since the TRC file generator converts a complex number into a 2-dimensional vector: Twice as many rows or columns are displayed in comparison to the Simulink model. This applies also to complex data in the fields of a structured variable.

## Conditions for the Inclusion of Variables From Referenced Models

### Introduction

When you use model referencing there are important specifics regarding the inclusion of variables in the TRC file.

### Specifics for model referencing

The following list shows the specifics regarding model referencing:

- The TRC file contains the variables of the top-level model and of all the referenced models. The variables for the referenced models are stored in groups in a similar way to variables of subsystems.
- Parameters defined in the model workspace of referenced models can be accessed via the Model Parameters group of the referenced model. Block-level references to these parameters can be accessed via the referenced model group of the respective model.
- The TRC file information for a referenced model is rebuilt only if new code is generated for the model. Incremental code generation thus also includes incremental TRC file generation.
- Whether variables of a referenced model are written to the TRC file depends on the setting of the Total number of instances per top model option in the Model Referencing dialog. If the total number of instances is set to
  - "One", the contents of the referenced model appear in the TRC file.
  - "Multiple" (default), the contents of the referenced model do not appear in the TRC file. This means that such a model can be used to build real-time applications, but you cannot access internal variables for the model via the TRC file.

For details, refer to the Simulink online help by MathWorks.

- Variables of Input blocks located on the root level of a referenced model are never generated into the TRC file. This also applies to the outputs of virtual blocks (Mux, Demux, Goto, From, subsystems), which are directly connected to such an Input block.
- It can happen that block outputs of blocks which drive an Outport block located on the root level of a referenced model are not generated. To solve this problem, a test point can be set for the respective signal.
- Like subsystem outputs, Model block outputs are also contained in the TRC file.
- The following applies to the entries for block parameters of referenced models:

If the Default parameter behavior of the referenced model is set to Tunable, the tunable parameters are available:

- In the Tunable Parameters group, if they are defined globally.
- In the Model Parameters group, if they are defined in the model workspace.
- Directly at the block, if they are defined locally.

If the Default parameter behavior of the referenced model is set to Inlined, parameters are available in the TRC file, which are specified as tunable via *Simulink.Parameter* objects .

- Consider the following regarding the use of options on the DSRT variable description file options page:
  - The **Include states** and **Include Derivatives** options are not supported for model referencing, neither for top-level models nor for referenced ones. If you select these options you are informed during the build process that they are ignored.
  - All the other options on the DSRT variable description file options page are supported for top-level as well as for referenced models.
- For details on the above-mentioned options, refer to [Adapting the Generation of the Variable Description File](#) on page 736.
- The setting for the **Include only Simulink.Parameter** and **Simulink.Signal** objects with **global storage class** option must be the identical for all models in a Model Referencing hierarchy.



# Downloading and Executing Real-Time Applications

---

<b>Introduction</b>	To work with a real-time application, you must download it to the target hardware and execute it.
---------------------	---

---

Where to go from here	Information in this section										
	<table><tr><td>Basics on Downloading Real-Time Applications.....</td><td>745</td></tr><tr><td>How to Download a Real-Time Application.....</td><td>748</td></tr><tr><td>How to Start a Real-Time Application.....</td><td>751</td></tr><tr><td>Application States of a Real-Time Application.....</td><td>753</td></tr><tr><td>Handling Errors and Exceptions When Executing Real-Time Applications.....</td><td>755</td></tr></table>	Basics on Downloading Real-Time Applications.....	745	How to Download a Real-Time Application.....	748	How to Start a Real-Time Application.....	751	Application States of a Real-Time Application.....	753	Handling Errors and Exceptions When Executing Real-Time Applications.....	755
Basics on Downloading Real-Time Applications.....	745										
How to Download a Real-Time Application.....	748										
How to Start a Real-Time Application.....	751										
Application States of a Real-Time Application.....	753										
Handling Errors and Exceptions When Executing Real-Time Applications.....	755										

---

## Basics on Downloading Real-Time Applications

---

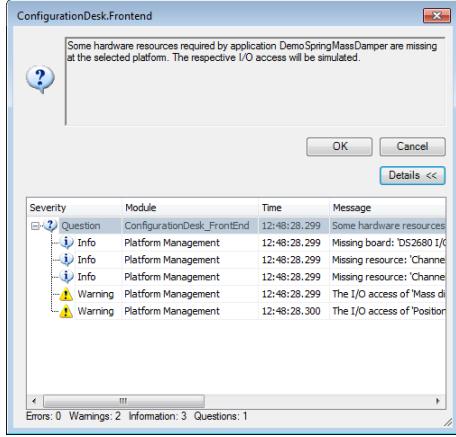
<b>Download process</b>	The work step which usually follows the build of real-time applications is the download process. This is run from ConfigurationDesk and enables you to analyze and test your application in a real-time hardware environment.
-------------------------	---

---

<b>Starting the download</b>	ConfigurationDesk allows you to start the download process in two different ways: <ul style="list-style-type: none"><li>▪ You can specify to start the download automatically after a successful build process was performed. For details, refer to <a href="#">Details on the Build Process</a> on page 716.</li><li>▪ You can start the download manually whenever you need it. For instructions, refer to <a href="#">How to Download a Real-Time Application</a> on page 748.</li></ul>
------------------------------	---

**Checking compatibility**

During the download process, ConfigurationDesk checks if the hardware topology and the configuration of the real-time application are compatible with the target hardware. This compatibility check can have the following results:

Result	Description																												
The application is compatible with the connected target hardware.	Application properties such as the processor architecture and the memory capacity match the corresponding properties of the target hardware. All the channels used by the application are identical to the channels on the target hardware regarding channel number, Ethernet adapter name, DS number of the I/O boards, slot number, unit name, rack name, and connected dSPACE pins. The application can be executed on the target hardware.																												
The application is conditionally compatible with the connected target hardware.	This case occurs if one of the following conditions applies: <ul style="list-style-type: none"> <li>▪ At least one channel used by the application is missing on the target hardware.</li> <li>▪ A channel property is configured differently, for example, the Load description property or connector pins.</li> <li>▪ The variant of the DS2655 FPGA Base Board does not match to the FPGA custom function block. Refer to <a href="#">How to Check Hardware Resources Required for FPGA Custom Function Blocks (SCALEXIO) (ConfigurationDesk I/O Function Implementation Guide</a> (</li> </ul> In these cases a message dialog appears which informs you about the inconsistencies and asks you how to continue. A message dialog can, for example, look like this:  <p>The screenshot shows a message dialog from ConfigurationDesk.Frontend. The title bar says 'ConfigurationDesk.Frontend'. The main message area contains the text: 'Some hardware resources required by application DemoSpringMassDamper are missing at the selected platform. The respective I/O access will be simulated.' Below this is a table with the following data:</p> <table border="1"> <thead> <tr> <th>Severity</th> <th>Module</th> <th>Time</th> <th>Message</th> </tr> </thead> <tbody> <tr> <td>Question</td> <td>ConfigurationDesk_FrontEnd</td> <td>12:48:28,299</td> <td>Some hardware resources</td> </tr> <tr> <td>Info</td> <td>Platform Management</td> <td>12:48:28,299</td> <td>Missing board: DS2680 1/C</td> </tr> <tr> <td>Info</td> <td>Platform Management</td> <td>12:48:28,299</td> <td>Missing resource: 'Channe</td> </tr> <tr> <td>Info</td> <td>Platform Management</td> <td>12:48:28,299</td> <td>Missing resource: 'Channe</td> </tr> <tr> <td>Warning</td> <td>Platform Management</td> <td>12:48:28,299</td> <td>The I/O access of 'Mass di</td> </tr> <tr> <td>Warning</td> <td>Platform Management</td> <td>12:48:28,300</td> <td>The I/O access of 'Position</td> </tr> </tbody> </table> <p>At the bottom of the dialog, there are buttons for 'OK' and 'Cancel', and a 'Details &lt;&gt;' button. Below the dialog, a status bar shows 'Errors: 0 Warnings: 2 Information: 3 Questions: 1'.</p>	Severity	Module	Time	Message	Question	ConfigurationDesk_FrontEnd	12:48:28,299	Some hardware resources	Info	Platform Management	12:48:28,299	Missing board: DS2680 1/C	Info	Platform Management	12:48:28,299	Missing resource: 'Channe	Info	Platform Management	12:48:28,299	Missing resource: 'Channe	Warning	Platform Management	12:48:28,299	The I/O access of 'Mass di	Warning	Platform Management	12:48:28,300	The I/O access of 'Position
Severity	Module	Time	Message																										
Question	ConfigurationDesk_FrontEnd	12:48:28,299	Some hardware resources																										
Info	Platform Management	12:48:28,299	Missing board: DS2680 1/C																										
Info	Platform Management	12:48:28,299	Missing resource: 'Channe																										
Info	Platform Management	12:48:28,299	Missing resource: 'Channe																										
Warning	Platform Management	12:48:28,299	The I/O access of 'Mass di																										
Warning	Platform Management	12:48:28,300	The I/O access of 'Position																										

**Tip**

You can also download a real-time application via ControlDesk. ControlDesk performs the same compatibility check during the download process as ConfigurationDesk.

**Note**

- The compatibility check is always performed entirely, i.e., it does not stop after the first inconsistency is detected, but continues till the end to give you a complete overview of all the inconsistencies in the Message Viewer.
- In multimodel applications, ConfigurationDesk automatically assigns models to cores for execution during the download process. You cannot change this assignment. If your multimodel application contains more models than the maximum allowed number, the download is aborted with an error message. For a SCALEXIO Real-Time PC, the maximum allowed number of models is (number of processor cores - 1). For the DS6001 Processor Board, it is the number of processor cores.

**Loading a real-time application to the flash memory**

ConfigurationDesk provides the following commands for loading a real-time application to the flash memory of a platform:

- **Real-Time Application - Load to Flash**

This command loads the selected real-time application to the flash memory of a registered platform. The real-time application is not started. You have to start it manually. Refer to [How to Start a Real-Time Application](#) on page 751.

- **Real-Time Application - Load to Flash and Start**

This command loads the selected real-time application to the flash memory of a registered platform, and starts it automatically.

If you reboot the platform, the real-time application in the flash memory is loaded to the RAM and started automatically.

**Note**

If you download a real-time application to a DS6001 Processor Board, the processor board must be connected only to standard SCALEXIO I/O boards. Otherwise, an warning is displayed during the download.

**Integrity check of the real-time application after download**

To check the integrity of the real-time integration and to guarantee functional safety, ConfigurationDesk ensures that the RTA file downloaded to the real-time system and the RTA file generated by ConfigurationDesk are bit-identical. If ConfigurationDesk detects any changes, the real-time application is not started and an error message is displayed.

**Channel behavior during download**

When a real-time application is downloaded, the channels are initialized first. They are set to a secure state during initialization, which means that the channels on the real-time hardware are cut off from the ECU electrically, and no signals are available at the I/O connectors of the dSPACE hardware. All channels which are obsolete or not assigned remain "secure" during the execution of the real-time application.

**Related topics**

**Basics**

[Application States of a Real-Time Application](#)..... 753

**HowTos**

[How to Download a Real-Time Application](#)..... 748

**References**

[Message Viewer \(ConfigurationDesk User Interface Reference\)](#)

[Real-Time Application - Load \(ConfigurationDesk User Interface Reference\)](#)

[Real-Time Application - Load and Start \(ConfigurationDesk User Interface Reference\)](#)

[Real-Time Application - Load to Flash \(ConfigurationDesk User Interface Reference\)](#)

[Real-Time Application - Load to Flash and Start \(ConfigurationDesk User Interface Reference\)](#)

## How to Download a Real-Time Application

---

**Objective**

Once built, a real-time application can be manually downloaded via ConfigurationDesk at any time, for example, if a real-time application was not downloaded immediately after the build.

---

**Preconditions**

To download a real-time application, a platform must be connected to ConfigurationDesk and displayed in the Platform Manager.

---

**Possible methods**

You can download a real-time application in three different ways:

- To download an RTA file to a platform via the Home ribbon, refer to Method 1.
- To download a selected RTA file from the Project Manager, refer to Method 2.
- To download an RTA file to a selected platform in the Platform Manager, refer to Method 3.

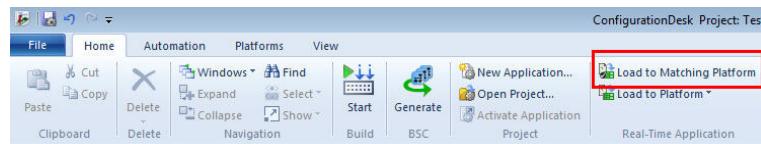
**Method 1****To download a real-time application via the Home ribbon**

- 1 Switch to the Build view set.

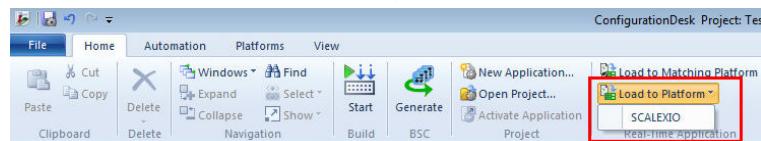
**Tip**

Alternatively, you can switch to the Project view set.

- 2 Download the real-time application to a platform:
  - If the active ConfigurationDesk application is in Matching platform connected state (displayed in the status bar), you can download the real-time application directly to the matching platform. To do this, click Load to Matching Platform on the Home ribbon.

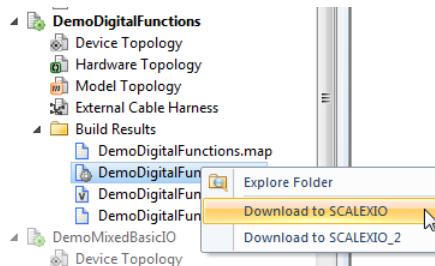


- If you want to select a registered platform, click Load to Platform on the Home ribbon and select an available platform from the submenu.

**Method 2****To download a real-time application from the Project Manager**

- 1 Open the Project Manager.
- 2 In the active application, right-click the RTA file which you want to download to a platform.

A context menu displaying the system names of the available registered platforms opens.

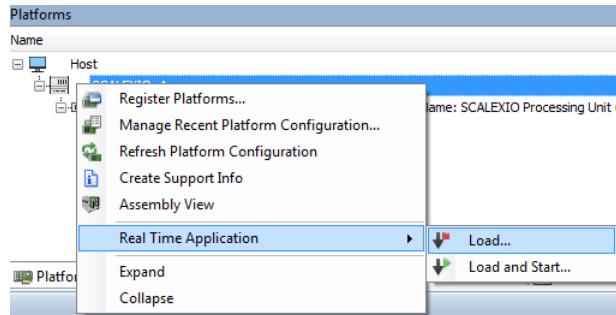


- 3 In the context menu, click the platform you want to download the real-time application to.

**Method 3****To download a real-time application in the Platform Manager**

- 1 In the Platform Manager, select the platform you want to download the real-time application to.

**2** From its context menu, select Real Time Application - Load.



The Select Real-Time Application dialog opens.

**3** In the Select Real-Time Application dialog, browse to the Build Results folder and select the RTA or SDF file you want to download.

The SDF file contains a reference to the related RTA file. If you select to download an SDF file to the real-time hardware, the related RTA file is actually downloaded to dSPACE real-time hardware.

**4** Click Open.

### Result

Depending on the compatibility of your real-time application with the selected hardware, the following results can occur:

- If the real-time application is fully compatible with the selected hardware, it is immediately downloaded and started. The Platform Manager displays it below the platform item with the name of the respective RTA file.
- If the real-time application is not or not fully compatible, you are informed about inconsistencies via dialog messages or in the Message Viewer. Depending on the kind of inconsistencies, the download is aborted automatically or you are asked if you want to continue despite the inconsistencies. For details on inconsistencies, refer to [Basics on Downloading Real-Time Applications](#) on page 745.

### Note

- If a real-time application is running on the selected hardware, ConfigurationDesk asks whether it is to be unloaded. If you click No, the download process is aborted.
- If you want to load a real-time application to the flash memory and the RAM of a registered platform, you have to use the Real-Time Application - Load to Flash command or the Real-Time Application - Load to Flash and Start command.

### Related topics

#### Basics

[Application States of a Real-Time Application.....](#) 753

Basics on Downloading Real-Time Applications..... 745

#### References

- [Clear Complete Flash Memory \(ConfigurationDesk User Interface Reference\)](#)
- [Real-Time Application - Load \(ConfigurationDesk User Interface Reference\)](#)
- [Real-Time Application - Load and Start \(ConfigurationDesk User Interface Reference\)](#)
- [Real-Time Application - Load to Flash \(ConfigurationDesk User Interface Reference\)](#)
- [Real-Time Application - Load to Flash and Start \(ConfigurationDesk User Interface Reference\)](#)

## How to Start a Real-Time Application

### Objective

You can start real-time applications manually.

#### Tip

Real-time applications can also start running automatically after download. Refer to the **Start real-time application** option in the **Global Build Settings** of the **Build Configuration** table.

### Precondition

The following preconditions must be fulfilled:

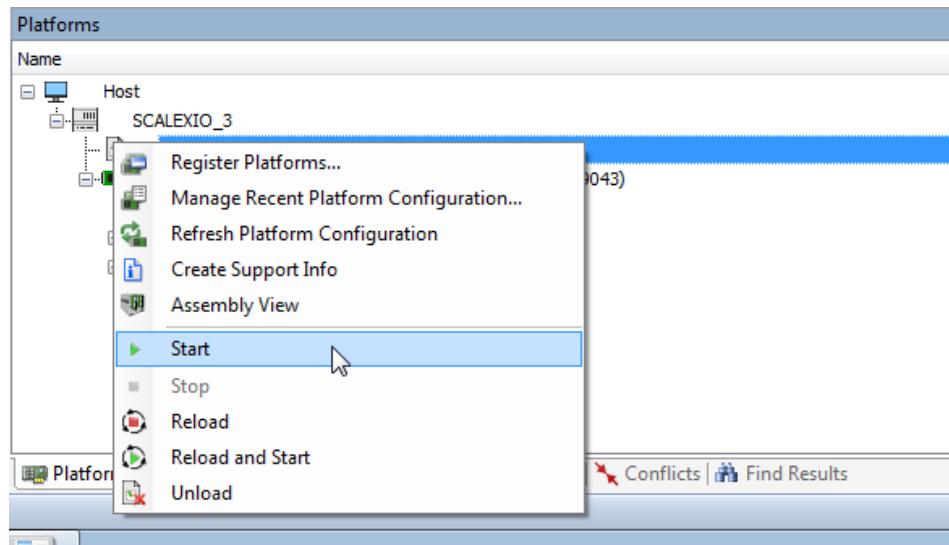
- A real-time system must be registered in ConfigurationDesk.
- A real-time application must be available on the real-time hardware.

### Method

#### To start a real-time application

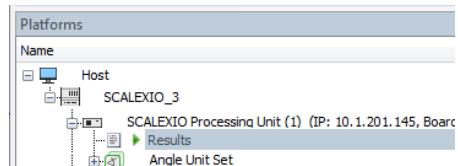
- 1 In the Platform Manager, select the real-time application you want to start.

2 From its context menu, select Start.



## Result

The selected real-time application has been started and is now running on the real-time hardware. The Platform Manager indicates this by a green arrow next to the application's name.



### Note

An application with errors can immediately switch to the application state "terminated", refer to [Application States of a Real-Time Application](#) on page 753.

### Tip

You can stop the real-time application by right-clicking it in the Platform Manager and selecting Stop from the context menu.

**Related topics****Basics**

[Application States of a Real-Time Application](#)..... 753

**HowTos**

[How to Download a Real-Time Application](#)..... 748

**References**

[Real-Time Application - Load \(ConfigurationDesk User Interface Reference\)](#)

[Start \(Real-Time Application\) \(ConfigurationDesk User Interface Reference\)](#)

[Stop \(Real-Time Application\) \(ConfigurationDesk User Interface Reference\)](#)

[Unload \(Real-Time Application\) \(ConfigurationDesk User Interface Reference\)](#)

## Application States of a Real-Time Application

**Introduction**

After a real-time application has been downloaded and started, it is executed on the real-time hardware.

**Application states of a real-time application**

A real-time application can have various application states on the real-time hardware, as shown below.

State	Symbol	Description
Running		<ul style="list-style-type: none"> <li>▪ All the tasks of the real-time application are executed.</li> <li>▪ The simulation time is incremented.</li> </ul>
Running		<ul style="list-style-type: none"> <li>▪ All the tasks of the real-time application loaded to the flash memory of the platform are executed.</li> <li>▪ The simulation time is incremented.</li> </ul>
Stopped		<ul style="list-style-type: none"> <li>▪ Tasks are not executed.</li> <li>▪ Simulation time is not incremented.</li> </ul>
Stopped		<ul style="list-style-type: none"> <li>▪ Tasks of the real-time application loaded to the flash memory are not executed.</li> <li>▪ Simulation time is not incremented.</li> </ul>
Terminated		This is a final state. No more actions are performed, and the application can be restarted only by downloading it again via the Platform Manager or via the Project Manager.

The states are visualized next to the application's name in the Platform Manager.

---

**Application state switches** Real-time applications can switch their application state on the real-time hardware as follows:

State Switch	Description
running → stopped	The implemented functions output stop values. Depending on the function block's configuration, a function outputs a user-defined stop value or keeps and outputs the last run-time value.
stopped → running	<ul style="list-style-type: none"> <li>▪ The implemented functions output initial values, until measurement values are available. Depending on the function block's configuration, a function outputs a user-defined initial value or keeps and outputs the last run-time value.</li> <li>▪ Parameters of the model are not reset.</li> <li>▪ Simulation time is reset to 0.</li> </ul>
running/stopped → terminated	No more actions are performed.

---

**Triggering application switches**

The "running to stopped" switch can be triggered manually by right-clicking the real-time application in the Platform Manager and selecting Stop from its context menu. You can also make running real-time applications switch themselves to the application state "stopped". To make this possible, you must ensure that one of the following Simulink conditions is fulfilled:

- In the behavior model's Configuration Parameters dialog, a stop time other than inf is specified.
- The model executes a Simulink Stop Simulation block.
- An S-function of the model executes the Simulink function `ssSetStopRequested()`.

The "stopped to running" switch can also be triggered manually by right-clicking the real-time application in the Platform Manager and selecting Start from its context menu.

The "running to terminated" or "stopped to terminated" switch occurs after serious errors. It takes place automatically. You cannot trigger it yourself.

---

**Related topics**

Basics

<a href="#">Basics on Downloading Real-Time Applications.....</a>	745
---	-----

HowTos

<a href="#">How to Download a Real-Time Application.....</a>	748
--	-----

## Handling Errors and Exceptions When Executing Real-Time Applications

### Handling errors with real-time applications

When a real-time application is running, errors can occur in different time phases.

**Errors during the execution phase** Errors which occur during the execution phase of a real-time application are displayed in the Message Viewer. There are two types:

- Recoverable errors

These do not violate the integrity of the real-time application, but they cause the application state to change from "running to stopped" as soon as all the tasks already triggered have been executed. No new tasks are triggered.

- Unrecoverable errors

These result from serious software or hardware problems. If they occur, all the tasks already triggered are terminated as fast as possible. The state of the real-time application switches to "terminated". An error message is displayed in the Message Viewer. You can restart the application only by loading it again.

**Errors during the initialization phase** During the initialization phase, all errors are treated as unrecoverable. This means they terminate the real-time application immediately. An error message is displayed in the Message Viewer.

### Handling exceptions with real-time applications

When a real-time application is running, different kinds of exceptions can occur:

- Recoverable exceptions

These arise from arithmetic calculations which have invalid operands or which have results such as NaN, +Inf or -Inf. Recoverable exceptions are ignored and not corrected. The real-time application continues running.

- Unrecoverable exceptions

These are caused by serious problems in the code of the running application or in the real-time hardware. Unrecoverable exceptions violate the integrity of the application and result in immediate termination. A message is displayed in the Message Viewer.



# Exporting Data for Documentation Purposes

## Objective

You can export the complete configuration data of a ConfigurationDesk application to a Microsoft Excel™ file for documentation purposes. If you want to document the data of a specific ConfigurationDesk component, for example, the Working View Manager or the Conflicts Viewer, you can export the data to an XML file or a CSV (comma separated value) file.

## Where to go from here

### Information in this section

<a href="#">How to Export the Configuration of an Application</a> .....	757
<a href="#">Description of the Exported Configuration</a> .....	758
<a href="#">How to Export Data for Documentation Purposes</a> .....	760

## How to Export the Configuration of an Application

### Objective

To document your configuration of a specific ConfigurationDesk application, you can export it to a Microsoft Excel™ file (XLSX file).

### Precondition

The application whose configuration you want to export is active.

### Method

#### **To export the configuration of an application**

- 1 In the Project Manager, right-click the currently active application.
- 2 From the context menu, select Export Configuration.  
An Export Configuration dialog with preset file format opens.
- 3 Select a folder, enter a file name, and click Save.

**Result** You exported the configuration of a ConfigurationDesk application for documentation purposes. For details on the contents of the exported file, refer to [Description of the Exported Configuration](#) on page 758.

<b>Related topics</b>	<b>Basics</b>
	Description of the Exported Configuration..... 758

## Description of the Exported Configuration

**Objective** An exported configuration Microsoft Excel™ file contains information on several areas of a ConfigurationDesk application. You cannot import this file.

**Entries in the exported file** The configuration data is provided in a table format as shown in the example below:

	A	D	F	U	V	AA	AP
1	Device Block	Device Port Group	Device Port	Logical Electrical	Function Block Type	Function Block Name	Electrical Interface
2	Name	Name	Name	Name	Name	Name	Signal Name
3	LeftDoorControl		DoorLight	1	Multi Bit In	Bit In DoorLight Left	Electrical Interface
4	LeftDoorControl		LightGND	2	Multi Bit In	Bit In DoorLight Left	Electrical Interface
5					Multi Bit In	Bit In DoorLight Left	Electrical Interface
6					Multi Bit In	Bit In DoorLight Left	Electrical Interface
7					Multi Bit In	Bit In DoorLight Left	Electrical Interface
8					Multi Bit In	Bit In DoorLight Left	Electrical Interface
9	LeftDoorControl		MirrorHeating	3	Multi Bit In	Bit In MirrorHeating Left	Electrical Interface
10				4	Multi Bit In	Bit In MirrorHeating Left	Electrical Interface
11					Multi Bit In	Bit In MirrorHeating Left	Electrical Interface
12					Multi Bit In	Bit In MirrorHeating Left	Electrical Interface
13					Multi Bit In	Bit In MirrorHeating Left	Electrical Interface
14					Multi Bit In	Bit In MirrorHeating Left	Electrical Interface
15	LeftDoorControl		MirrorMotor	5	PWM/PFM In	PWM In MirrorMotor Left	Electrical Interface
16	LeftDoorControl		MirrorGND	6	PWM/PFM In	PWM In MirrorMotor Left	Electrical Interface
17					PWM/PFM In	PWM In MirrorMotor Left	Electrical Interface
18					PWM/PFM In	PWM In MirrorMotor Left	Electrical Interface
19					PWM/PFM In	PWM In MirrorMotor Left	Electrical Interface
20					PWM/PFM In	PWM In MirrorMotor Left	Electrical Interface
21	LeftDoorControl		Lock	7	Multi Bit Out	Bit Out Lock Left	Electrical Interface
22				8	Multi Bit Out	Bit Out Lock Left	Electrical Interface
23	LeftDoorControl		Unlock	9	Multi Bit Out	Bit Out Unlock Left	Electrical Interface
24	LeftDoorControl		GND	10	Multi Bit Out	Bit Out Unlock Left	Electrical Interface
25	RightDoorControl		DoorLight	11	Multi Bit In	Bit In DoorLight Right	Electrical Interface
26	RightDoorControl		LightGND	12	Multi Bit In	Bit In DoorLight Right	Electrical Interface
27					Multi Bit In	Bit In DoorLight Right	Electrical Interface

You can expand further columns to view more details on specific areas of the configuration:

1	F	U	V	W	X	Y		
1	Device Port	Logical Electrical Link	Function Block Type	Name	Path	Physical Attributes	Port Type	Classification
3	DoorLight	1	Multi Bit In		Current, Voltage	In	Basic I/O; D	
4	LightGND	2	Multi Bit In		Current, Voltage	In	Basic I/O; D	
5			Multi Bit In		Current, Voltage	In	Basic I/O; D	
6			Multi Bit In		Current, Voltage	In	Basic I/O; D	
7			Multi Bit In		Current, Voltage	In	Basic I/O; D	
8			Multi Bit In		Current, Voltage	In	Basic I/O; D	
9	MirrorHeating	3	Multi Bit In		Current, Voltage	In	Basic I/O; D	
10		4	Multi Bit In		Current, Voltage	In	Basic I/O; D	
11			Multi Bit In		Current, Voltage	In	Basic I/O; D	
12			Multi Bit In		Current, Voltage	In	Basic I/O; D	
13			Multi Bit In		Current, Voltage	In	Basic I/O; D	
14			Multi Bit In		Current, Voltage	In	Basic I/O; D	

In most cases, the column header represents a property of a ConfigurationDesk application element and the table cells contain the respective property values.

## Different sheets

The table contains different sheets containing data from different areas of the ConfigurationDesk application.

21	Lock	7	Multi Bit Out	Bit Out Lock Left	Electrical Interface Bit 1
22		8	Multi Bit Out	Bit Out Lock Left	Electrical Interface Bit 1
23	Unlock	9	Multi Bit Out	Bit Out Unlock Left	Electrical Interface Bit 1

You can find a description for each of the sheets below. The unique and important columns from each sheet are also described. For descriptions of the general properties of different ConfigurationDesk application elements, refer to the related reference documentation.

**Signal Chain** The Signal Chain sheet is the main sheet containing most columns from the other sheets as well. It contains the properties of different signal chain elements that are grouped and highlighted according to their positions in the signal chain and the mappings between them. The function block properties, for example, are indicated by blue column headers.

### Tip

ConfigurationDesk collects the configuration data for each element of the signal chain and then groups connected elements in the same rows. For a better overview and to reduce the total number of rows, channel requests from a signal group (logical signal) are displayed in parallel to the signal ports from the same signal group instead of creating a new row for each channel request and signal port. This means that channel request and channel information do not necessarily belong to a signal port in the same row.

Mappings are represented by the following columns:

- **Logical Electrical Link:** Unique identifier for a mapping between device port(s) and/or signal port(s).

- Software Link: Unique identifier for a mapping between a function port and a model port or between two model ports.

**Device Configuration** Contains the properties of device topology elements used in the signal chain. For property descriptions, refer to [External Device Properties \(ConfigurationDesk User Interface Reference\)](#).

**Hardware Resource Assignment** Contains the properties relevant for the assignment of hardware resources to function blocks. The sheet contains a row for each channel request. For details on hardware resource assignment and channel requests, refer to [Assigning Hardware Resources to Function Blocks](#) on page 399.

**IO Functions** Contains the properties of instantiated function blocks and their elements. For property descriptions, refer to [ConfigurationDesk Function Block Properties](#).

The Function Block Type – Path column contains the custom functions directory (only for custom function blocks). Custom functions must be located in the project-specific custom functions directory or in the global custom functions directory. Refer to [ConfigurationDesk Custom I/O Function Implementation Guide](#).

**External Cable Harness** Contains the wiring information for an external cable harness.

For descriptions of the table columns refer to [Description of an Exported Cable Harness Microsoft Excel™ Sheet](#) on page 690.

---

#### Related topics

#### HowTos

[How to Export the Configuration of an Application](#)..... 757

## How to Export Data for Documentation Purposes

---

#### Objective

You can export data from different panes into an XML or CSV file for documentation purposes.

---

#### Method

#### To export data for documentation purposes

- 1 Right-click in the pane from which you want to export the data.
- 2 In the context menu, click Export View – Export View to XML or Export View – Export View to CSV.  
A dialog with preset file format opens.
- 3 Enter a file name and click Save.

**Result**

You exported data for documentation purposes.



# Limitations

## Where to go from here

## Information in this section

Limitations for Starting ConfigurationDesk.....	764
Limitations Concerning Hardware.....	764
Limitations Concerning Projects and Applications.....	765
Limitations Concerning the Signal Chain.....	765
Limitations Concerning Function Blocks.....	767
Limitations Concerning Multi-Processing-Unit Systems.....	767
Limitations Concerning Multiple Model Implementations Assigned to the Same Application Process.....	768
Limitations Concerning V-ECU Implementations.....	769
Limitations Concerning Simulink Implementation Containers.....	772
Limitations Concerning Functional Mock-up Units (FMUs).....	773
Limitations Concerning Bus Simulation Containers.....	774
Limitations Regarding Simulink Variables in the TRC File.....	776
Limitations for Ethernet Communication.....	777
Limitations for FlexRay Communication.....	778
Limitations for SENT Communication.....	778
Limitations for Gigalink Communication.....	778
Limitations Concerning ECU Interfacing with SCALEXIO or MicroAutoBox III.....	779
Limitations Concerning the Build and Download Process.....	779
Limitations Concerning MATLAB Compatibility.....	780

### Information in other sections

[Limitations for Using the Bus Manager \(ConfigurationDesk Bus Manager Implementation Guide !\[\]\(c47b73203714532ae45300f0bc03e668\_img.jpg\)](#)

[Limitations for Automating ConfigurationDesk \(ConfigurationDesk Automating Tool Handling !\[\]\(f27fa31d25923ebacde41fb746401dc7\_img.jpg\)](#)

## Limitations for Starting ConfigurationDesk

### Multiple instances not supported

In some cases, it is possible to start multiple instances of ConfigurationDesk on the same host PC. This is not supported and can cause unexpected behavior. Make sure that you do not start ConfigurationDesk when it is already running under a different user.

## Limitations Concerning Hardware

### Limitations for updating the firmware of SCALEXIO systems

Note the following restrictions when you want to update the firmware of your SCALEXIO system:

- The Update Firmware Wizard supports SCALEXIO systems as of dSPACE Release 2015-B. If you want to update the firmware version on a SCALEXIO system to an earlier version, you have to use ConfigurationDesk from an earlier dSPACE Release.
- If your SCALEXIO system contains a DS2655M2 Digital I/O Module, a firmware update from firmware version 3.2 or earlier to a firmware version 3.3 or later, might lead to an error and the update process is then stopped.

To finish the firmware update you have to do the following steps:

- Restart the SCALEXIO system.
  - Call the [Refresh Interface Connections \(ConfigurationDesk User Interface Reference !\[\]\(d3cdbf8b1aefd508ed11d8487cf3c888\_img.jpg\)](#)) command in the Platform Manager.
  - Repeat the firmware update process.
  - If your SCALEXIO system contains one or more new hardware components that are not already supported by the currently active firmware on the SCALEXIO Processing Unit or DS6001 Processor Board, a firmware update might lead to an error and the update process is then aborted.
- To finish the firmware update, perform the following steps:
- Restart the SCALEXIO system.
  - Call the [Refresh Interface Connections \(ConfigurationDesk User Interface Reference !\[\]\(477d2522eaf770e16e8ad8d16cd9813d\_img.jpg\)](#)) command in the Platform Manager.
  - Repeat the firmware update process.

---

<b>Limitation on reusing an external cable harness</b>	If you reuse an external cable harness, the automatic hardware resource assignment analyzes the relations between the device port mapping and the external cable harness before it assigns the hardware. In exceptional cases, this analysis might not cover all complex relations. This could lead to a hardware resource assignment that results in a conflict.
<b>Limited task access to DS2680 I/O Unit</b>	A DS2680 I/O Unit must not be accessed by more than 14 function modules. This means that: <ul style="list-style-type: none"> <li>▪ Simulink tasks that access a DS2680 I/O Unit must be partitioned into 14 function modules at the most.</li> <li>▪ The number of Simulink tasks that can access a DS2680 I/O Unit is limited to a maximum of 14.</li> </ul>

## Limitations Concerning Projects and Applications

---

<b>Character limit for project and application names</b>	The Windows maximum path length limit of 260 characters applies to ConfigurationDesk projects and applications. ConfigurationDesk does not check if this limit is exceeded by projects, applications, or project and application elements.
--	--

### Note

Project and application element paths exceeding the limit of 260 characters are corrupted. You have to make sure that project paths do not exceed the limit.

## Limitations Concerning the Signal Chain

---

<b>Handling large applications</b>	Large applications (more than approx. 4500 function blocks or 4000 device ports) cause operations to take more time: <ul style="list-style-type: none"> <li>▪ Using working views that contain a large number of elements, such as the Global working view, slows down performance, for example, when creating a device port or a function block.</li> <li>▪ Migrating projects from earlier releases with large applications takes several minutes. During the migration process, ConfigurationDesk does not respond to user interaction. You have to wait until the migration is finished.</li> </ul>
------------------------------------	---

<b>Mapping indicators for parent ports</b>	When you use parent ports for model port mapping, indicators notifying you that the (auto-)mapping is complete or may result in conflicts are never shown while dragging the mapping line. That means: <ul style="list-style-type: none"><li>▪ If you drag a mapping line starting at a parent port, no indicators are shown at the other parent ports. All individual ports are marked red.</li><li>▪ If you drag a mapping line starting at an individual port, indicators are shown only at other ports, but not at the parent ports.</li></ul>
<b>Limitations for importing working views that contain Bus Configuration function blocks</b>	<p>You can import working views to a ConfigurationDesk application via CAFX (ConfigurationDesk application fragment) files. When a CAFX file contains Bus Configuration function blocks, the following limitations apply.</p> <p><b>No support of CAFX files exported with Release 2020-B or earlier</b> You cannot import CAFX files that were exported with dSPACE Release 2020-B or earlier. If you try to do so, the import is aborted and a warning message is displayed.</p> <p><b>No support of merging or replacing signal chains in working views</b> You cannot import the CAFX file via the Merge Signal Chain into Working View or Replace Signal Chain in Working View commands. If you try to do so, the import is aborted and a warning message is displayed.</p> <p>You can import the CAFX file only via the Add Signal Chain to Working View command.</p> <p><b>Limitations for communication matrices</b> For the used communication matrices, the following limitations apply:</p> <ul style="list-style-type: none"><li>▪ The communication matrices that were used to configure the Bus Configuration function blocks must be available in the Buses Browser. If one of the required communication matrices is missing, the import is aborted and a warning message is displayed.</li><li>▪ If user-defined ISignal IPDUs and/or ISignals were used to configure the Bus Configuration function blocks, you must add the same user-defined ISignal IPDUs and ISignals to the related communication matrix in the Buses Browser, i.e., the following settings must match:<ul style="list-style-type: none"><li>▪ Related higher-level elements</li><li>▪ Names</li><li>▪ Direction (TX or RX)</li></ul>If a required user-defined ISignal IPDU or ISignal is missing, the import is aborted and a warning message is displayed.</li><li>▪ User-defined settings of communication matrix elements are not included in CAFX files. If communication matrix elements with user-defined settings were used to configure Bus Configuration function blocks, e.g., PDUs with user-defined cyclic timings or ISignals with modified base data types, the user-defined settings are lost when you import a CAFX file. Instead, the original communication matrix settings are used.</li></ul> <p><b>Limitations for the target ConfigurationDesk application</b> Importing a CAFX file that contains Bus Configuration function blocks to a ConfigurationDesk application might cause problems when you build a real-time</p>

application or generate bus simulation containers later on. To avoid these problems, adhere to the following limitations:

- Do not import the CAFX file to the ConfigurationDesk application from which it was exported.
- Do not import the CAFX file to the same ConfigurationDesk application more than once.

## Limitations Concerning Function Blocks

### **Problem with using special characters in function block names**

If you use a special character such as a semicolon in a function block name, it is not written to the FIU configuration data required for experiment software. Special characters are replaced by blanks in the configuration data, even though they are still displayed in ConfigurationDesk. This causes errors if you manually provide the names containing special characters in other dSPACE products used in your failure configuration scenario.

## Limitations Concerning Multi-Processing-Unit Systems

### **Only one real-time application per processing unit**

Only one single-core real-time application, multicore real-time application, or multi-processing-unit application is allowed per processing unit. For example, if real-time application RTA1 runs on some of the cores of a processing unit, it is not possible to run real-time application RTA2 on the other (unused) cores.

### **Limitations concerning connections to the host PC**

- All the processing units in a multi-processing-unit system must be connected via Ethernet network to the host PC. You cannot have only one processing unit connected to the host PC via Ethernet while all the other processing units are just connected via IOCNET.
- All the processing units in a multi-processing-unit system must be connected to the same Ethernet network. You cannot connect processing units to the host PC via separate network cards.

### **IOCNET connections required in a multi-PU system**

SCALEXIO processing units of a multi-processing-unit system must be connected via IOCNET. The IOCNET ports must be configured for IOCNET communication. For details, refer to [Communication Network of a SCALEXIO System \(SCALEXIO Hardware Installation and Configuration\)](#).

### **No access to I/O channels of I/O boards of other local systems**

Application processes that are executed on a processing unit can use only the I/O channels of I/O boards of their local system. They cannot access I/O channels of I/O boards of other local systems. That means, an I/O function that is assigned to

an I/O resource of one processing unit must not be mapped to the data port of a model that is executed on another processing unit. Otherwise, a conflict is shown in the Conflicts Viewer. Refer to [Function Block Conflicts](#) ([ConfigurationDesk Conflicts](#) ).

**Note**

The above limitation does not apply to angular processing units (APUs) or clocks which are accessible throughout the network.

## Limitations Concerning Multiple Model Implementations Assigned to the Same Application Process

<b>Limited support of models with blocks from the MotionDesk Blockset</b>	Only one Simulink model or SIC file with blocks from the MotionDesk Blockset is supported per application process.
<b>Unsupported blocks in Simulink models assigned to multimodel application processes</b>	ConfigurationDesk does not support Simulink models or SIC files in multimodel application processes that contain blocks from the following blocksets: <ul style="list-style-type: none"><li>▪ RTI CAN MultiMessage Blockset</li><li>▪ RTI LIN MultiMessage Blockset</li><li>▪ FlexRay Configuration Package</li><li>▪ Ethernet Configuration Package</li></ul>
<b>TRC file limitation</b>	If two or more model implementations are assigned to the same application process, and the related TRC files contain the same global C variables, the variables cannot be accessed during real-time simulation.
<b>Build options not configurable for model implementations</b>	You cannot configure different build options for model implementations that are assigned to the same application process.
<b>Name restrictions of SIC files contained in BSC files</b>	When you use BSC files in multimodel applications, you must observe the following name restrictions: <ul style="list-style-type: none"><li>▪ If a BSC file contains a Simulink implementation container (SIC file), the name of the model implementation that the SIC file describes must be unique in the</li></ul>

application process to which the BSC file is assigned. This includes the following model implementations:

- All model implementations assigned to the application process.
- All model implementations described by SIC files that are included in other BSC files assigned to the application process.
- If you assign two or more BSC files to an application process, the names of the bus configurations that are included in the BSC files must be unique across all BSC files.

---

<b>Name restrictions of bus configurations in BSC files</b>	Multiple BSC files that contain bus configurations with the same name are not supported in a multimodel application process.
<b>No support of shared objects referencing container code</b>	If a model implementation container that is assigned to a multimodel application process provides shared objects, the shared objects must not contain references to the model implementation container code.

## Limitations Concerning V-ECU Implementations

---

<b>No Real-Time Testing for V-ECU implementation tasks</b>	You cannot specify Real-Time Testing for tasks provided by V-ECU implementations.
<b>RTE interventions without task references</b>	RTE interventions without task references can be used only for model communication. You must not connect RTE intervention ports without task references to function ports. If you do, the build process will be aborted with an error message.
<b>No multiple starts for a real-time application containing V-ECU implementations</b>	You cannot restart a real-time application containing V-ECU implementations after stopping it. The SCALEXIO system issues an error message in this case.
<b>I/O access within extended tasks</b>	I/O ports of V-ECU implementations which are accessed by extended tasks must not be connected to I/O function ports.
<b>No scaled time interval and scaling factor properties</b>	Scaled time interval and scaling factor properties are not supported for application processes with assigned V-ECU implementations.
<b>No A2L variables for CAN controllers</b>	The real-time application built by ConfigurationDesk does not contain A2L variables for the CAN controllers of a V-ECU implementation.

<b>Limitations concerning LIN controllers</b>	<p><b>No slave configurations for the LinIf module</b> ConfigurationDesk does not support slave configurations defined for the LinIf module.</p> <p><b>No LIN transport protocol</b> The LIN transport protocol is not supported.</p> <p><b>No LIN node configuration services</b> LIN node configuration services are not supported.</p>
<b>No automatic layout generation in ControlDesk Bus Navigator for CAN or LIN parts</b>	<p>During the build process, ConfigurationDesk does not generate EXPSWCFG files for the CAN or LIN parts of a V-ECU implementation. EXPSWCFG files are required for automatic layout generation in ControlDesk's Bus Navigator. Thus, automatic layout generation in the Bus Navigator is not supported for elements related to CAN or LIN parts of V-ECU implementations.</p>
<b>No validation of binary files in V-ECU implementations</b>	<p>When you add or update V-ECU implementations that contain binary files, ConfigurationDesk does not check if the binary files:</p> <ul style="list-style-type: none"><li>▪ Have a valid file name extension</li><li>▪ Are valid as binary files</li><li>▪ Are suitable for the current target platform (only valid for V-ECU implementations based on CTLGZ files)</li></ul> <p>You must ensure that the binary files provided by V-ECU implementations meet the above requirements.</p>
<b>Restricted support of MIME types for binary files</b>	<p>ConfigurationDesk supports only the following MIME types for binary files in V-ECU implementations:</p> <ul style="list-style-type: none"><li>▪ <code>application/obj</code></li><li>▪ <code>application/lib</code></li></ul> <p>ConfigurationDesk does not support the <code>application/octet-stream</code> MIME type in binary files in V-ECU implementations. If a V-ECU implementation contains unsupported MIME types, a warning is displayed when you add or update the V-ECU implementation, and when you start the build process.</p>
<b>Import/export of MTFX files</b>	<p>If you export model topologies that contain V-ECU implementations, the V-ECU implementation parts are not saved in the MTFX file.</p>
<b>Restricted support of AUTOSAR OS elements</b>	<p>ConfigurationDesk does not support V-ECU implementations that provide specific AUTOSAR OS elements. The following applies:</p> <ul style="list-style-type: none"><li>▪ V-ECU implementations must not use the following AUTOSAR OS elements:<ul style="list-style-type: none"><li>▪ AUTOSAR OS software counters</li><li>▪ AUTOSAR OS schedule tables</li><li>▪ AUTOSAR OS ISRs</li><li>▪ AUTOSAR OS task timing protections</li></ul></li></ul>

- V-ECU implementations must contain exactly one AUTOSAR OS application.
- Only one application mode is allowed in a V-ECU implementation.
- The incrementation of AUTOSAR OS counters is always triggered by a timer.
- The AUTOSAR OS alarm action of an AUTOSAR OS alarm can only activate tasks and set AUTOSAR OS events.
- Only 32 different task priorities are allowed in a V-ECU implementation.

---

**Restricted support of microcontroller abstraction layer (MCAL) modules**

ConfigurationDesk supports only V-ECU implementation containers that use the following modules for the microcontroller abstraction layer (MCAL):

- Sab module
- Os module
- Dap module
- CanDrv module
- LinDrv module

If you try to add V-ECU implementation containers that use other MCAL modules to a ConfigurationDesk application, the import is aborted and ConfigurationDesk displays an error message.

**Note**

- If you add V-ECU implementations that contain both CanIf module configurations and CanDrv module configurations to a ConfigurationDesk application, only the configurations from the CanDrv module are imported into ConfigurationDesk.
- If you add V-ECU implementations that contain both LinIf module configurations and LinDrv module configurations to a ConfigurationDesk application, only the configurations from the LinDrv module are imported into ConfigurationDesk.

---

**No consideration of unmapped LIN function blocks and associated LIN tasks**

If you remove the mapping line between the Configuration port of LIN function block and the Configuration port of a Configuration Port block of a V-ECU implementation, the LIN task configuration is not updated. Unmapped LIN function blocks and the related LIN tasks are not included in the real-time application during the build process.

---

**No support of V-ECU implementations for an adaptive platform**

ConfigurationDesk does not support V-ECU implementation containers providing a V-ECU implementation for an adaptive platform that requires a Linux environment on the simulator.

---

**No support of DsIDBusIf**

ConfigurationDesk does not support the DsIDBusIf software module, which is a dSPACE-specific module for idealized bus interfaces, e.g., to FlexRay controllers. For an overview of the basic software module support for AUTOSAR classic V-

ECUs and non-AUTOSAR V-ECUs, refer to [Basic Software Module Support for V-ECUs \(SIL Testing Overview\)](#).

<b>No A2L variable access for static variables</b>	ConfigurationDesk does not support A2L variable access for static variables of V-ECU implementations.
--	---

<b>Maximum number of variables that can be measured</b>	The number of variables that can be measured is limited.		
Data Type	Maximum Number of Variables That Can Be Measured ...	... A2L Variables	... TRC Variables <sup>1)</sup>
Float, int, short, or byte	Up to 1,245	Up to 1,245	Up to 1,048,576
Double	Up to 1,244	Up to 1,244	Up to 131,072

<sup>1)</sup> The maximum numbers apply to measurements with only a single measurement raster. If you use multiple measurement rasters, higher maximum numbers apply. However, these numbers depend on the specific measurement scenario.

<b>No support of page switching</b>	Real-time applications generated by ConfigurationDesk do not support the page switching commands of the XCP protocol.
-------------------------------------	---

## Limitations Concerning Simulink Implementation Containers

<b>No export of SIC files into MTFX files</b>	ConfigurationDesk does not support exporting a complete model topology containing an SIC file into an MTFX file. All parts of the model topology related to the SIC file, i.e., all subsystems and all model port blocks of the SIC file, are ignored during an MTFX export.
---	--

<b>Limitations for TargetLink-created SIC files</b>	The following limitations apply to TargetLink-created SIC files: <ul style="list-style-type: none"> <li>▪ SIC files generated with TargetLink 4.4 (dSPACE Release R2018-B) are not supported in application processes that have multiple model implementations assigned.</li> <li>▪ SIC files generated with TargetLink 4.4 (dSPACE Release R2018-B) cannot be used in real-time applications for the MicroAutoBox III.</li> <li>▪ You cannot use SIC files that are generated with TargetLink for BSC file generation.</li> </ul>
---	--

<b>Restricted support of SIC files generated for dsrt64.tlc</b>	You can add SIC files generated for the <code>dsrt64.tlc</code> target system file to a ConfigurationDesk application. However, the SIC files can only be used to
---	---

generate BSC files for VEOS simulations on Linux. If you start a build process for a ConfigurationDesk application that contains an SIC file generated for the `dsrt64.tlc` target system file, ConfigurationDesk creates a conflict and the build process is aborted.

## Limitations Concerning Functional Mock-up Units (FMUs)

### Preconditions for adding FMUs to the ConfigurationDesk application

FMUs that you want to add to your ConfigurationDesk application must fulfill the preconditions listed below.

- They must comply with FMI 2.0 for Co-Simulation.
- They must not require additional tools for simulation.
- They must not define a default start time other than 0.0.

#### Note

If you try to import an FMU that does not fulfill the above preconditions, ConfigurationDesk aborts the import with an error message. No data is imported and no folder is created for the FMU in the ConfigurationDesk application's folder.

### FMUs without source files or supported libraries

FMUs must provide their C functions as source files or they must have been created with the PrecompileFMU command (refer to [Creating Precompiled FMUs](#) on page 618). You can still add an FMU that does not fulfill these preconditions to the ConfigurationDesk application. However, a warning is displayed and a build is not possible in this case.

### Unsupported FMI versions

ConfigurationDesk supports FMUs complying with the FMI 2.0 standard for Co-Simulation. For detailed and up-to-date compatibility information on FMI support in ConfigurationDesk, refer to the following website:  
<http://www.dspace.com/go/FMI-Compatibility>.

#### Note

You might be able to add an FMU based on other versions of the FMI standard to your ConfigurationDesk application. If you do this, errors might occur during the build process.

### No support of FMI for Model Exchange interface FMUs

ConfigurationDesk supports only the FMI for Co-Simulation interface, but not the FMI for Model Exchange interface.

<b>Restricted support of the string data type</b>	ConfigurationDesk does not create model ports for inputs and outputs with this data type. Variables of the <b>string</b> data type can be initialized via the <code>start</code> attribute in the <code>modelDescription.xml</code> file of the FMU. However, ConfigurationDesk does not generate TRC file entries for these variables.
<b>Restricted support of fixed parameters</b>	<p>During the build process, ConfigurationDesk generates TRC file entries for fixed parameters:</p> <ul style="list-style-type: none"><li>▪ causality = parameter <i>and</i> variability = fixed</li></ul> <p>The values of these parameters can be changed during simulation. However, changing these parameters has no effect on the simulation results until the simulation is stopped and started again.</p> <p>In the experiment software, fixed parameters are indicated by the following symbol: .</p>
<b>Unit information is ignored</b>	Unit information provided for FMU variables in the <code>modelDescription.xml</code> file is ignored.
<b>Unique FMU model names</b>	The names of models described by the FMUs must be unique within a ConfigurationDesk application. Explanation: The name of the FMU file and the name of the model that is described by the FMU file might differ. ConfigurationDesk uses the value of the <code>modelName</code> attribute that is defined in the FMU's model description file as the model name. For this limitation, the model name is relevant, not the name of the FMU file.
<b>No export of FMUs into MTFX files</b>	ConfigurationDesk does not support exporting a complete model topology containing an FMU into an MTFX file. All parts of the model topology related to the FMU are ignored during an MTFX export.
<b>No additional tools for simulation</b>	FMUs requiring additional tools for simulation are not supported.

## Limitations Concerning Bus Simulation Containers

<b>Introduction</b>	The following limitations apply when you work with bus simulation containers in ConfigurationDesk.
---------------------	--

**MTFX export**

ConfigurationDesk does not support exporting a complete model topology containing a bus simulation container (BSC) file into an MTFX file. All parts of the model topology related to the BSC file, i.e., all subsystems and all model port blocks of the BSC file, are ignored during an MTFX export.

**BSC files and ConfigurationDesk from different dSPACE Releases**

You can add bus simulation container (BSC) files to the Model Browser only if the BSC files are generated with the same dSPACE Release as ConfigurationDesk.

**Naming restrictions for SIC files and bus configurations**

When you use BSC files in a ConfigurationDesk application, you must observe the following naming restrictions:

- If a BSC file contains a Simulink implementation container (SIC file), the name of the [model implementation](#) that the SIC file describes must be unique in the application process to which the BSC file is assigned. This applies to the following:
  - All model implementations assigned to the application process.
  - All model implementations described by SIC files that are included in other BSC files assigned to the application process.
- The names of bus configurations that are included in a BSC file must be unique in the application process to which the BSC file is assigned. This applies to all BSC files and Bus Configuration function blocks that are assigned to the application process.

**No support of BSC files containing SIC files generated for dsrt64.tlc**

BSC files that include SIC files generated for the `dsrt64.tlc` system target file are not supported by ConfigurationDesk. You can use the BSC files in VEOS to build an offline simulation application for execution on a Linux operating system.

**Implementing the bus communication of BSC files in real-time applications**

When implementing the bus communication of a BSC file in a real-time application, the following restrictions apply:

- You must map all the Configuration ports of a BSC file to suitable bus function blocks (CAN, LIN) to implement the bus communication in a real-time application.
- You must not map the Configuration ports of a BSC file to bus function blocks that are used in one of the following ways:
  - The bus function block specifies the [bus access](#) for [bus access requests](#) of bus configurations.
  - The bus function block is referenced by one or more ECU Interface Configuration function blocks.
- If a BSC file contains Configuration ports resulting from LIN bus access requests, at least one LIN function block must be assigned to the same application process as the BSC file. This applies even if you do not want to implement LIN communication in the real-time application.

## Limitations Regarding Simulink Variables in the TRC File

<b>Block output signals</b>	In some cases, Simulink Coder does not generate separate variables for block output signals. To make sure that Simulink Coder generates a separate variable for an output signal, you can designate any signal in a model as a test point.
<b>Handling of the Selector block and the Bus Selector block</b>	<p>The outports of the Selector block, the Bus Selector block, and other virtual blocks that are connected to them are not generated into the variable description file, if the import of the Selector block is connected to a contiguous array, or if the import of the Bus Selector block is connected to a non-virtual bus.</p> <p>To access a signal of a bus, do the following:</p> <ul style="list-style-type: none"><li>▪ Access the signal directly from the bus/array or</li><li>▪ Connect a non-virtual block to the signal, for example, a Signal Conversion block whose Output property is set to <b>Signal Copy</b>. Its outport is available in the variable description file.</li></ul> <p>If the import of the Bus Selector block is connected via virtual bus, the outports are generated into the variable description file.</p>
<b>No TRC file entries for unsupported data types</b>	<p>The Model Interface Package for Simulink does not generate any TRC file entries for the following elements:</p> <ul style="list-style-type: none"><li>▪ String signals</li><li>▪ Fixed-point variables that are longer than 64 bits</li></ul> <div style="background-color: #f0f0f0; padding: 10px; border-radius: 5px;"><p><b>Note</b></p><p>To use 64 bit integer variables in the generated code, the SupportLongLong flag must be set in the Simulink model.</p></div> <ul style="list-style-type: none"><li>▪ Structured variables, such as bus signals or structured parameters whose elements have an unsupported data type</li><li>▪ Simulink messages</li></ul>
<b>No TRC file entries for virtual signals</b>	Entries for virtual signals, e.g., from a Mux block or a Bus Creator block, are not generated into the TRC file.
<b>Naming conflicts for signals and parameters</b>	A Simulink block can have output signals and parameters with the same name, for example, in the case of masked subsystems or Stateflow charts. For this block, multiple entries with the same name are generated at the same level in the TRC file. To avoid this, you must specify different names for output signals and parameters of a Simulink block.

## Limitations for Ethernet Communication

<b>SCALEXIO Real-Time PC HPP 2.0: Using the 10-Gbit Ethernet controller</b>	<p>The 10-Gbit Ethernet controller (Eth0_3) of the SCALEXIO Real-Time PC Version HPP 2.0 has the following limitations:</p> <ul style="list-style-type: none"> <li>▪ The controller supports Ethernet connections only with a link speed of 1 Gbit/s and 10 Gbit/s.</li> <li>▪ The link speed used is determined exclusively by means of autonegotiation. The setting of the Link speed property of the Ethernet Setup function block is ignored.</li> <li>▪ PTP frames with a VLAN tag are not supported.</li> </ul>
<b>MicroAutoBox III: Receiving broadcast messages</b>	<p>The physical and virtual Ethernet controllers can receive broadcast messages from subnetworks that do not match the IP settings of the controllers. This affects real-time applications for the MicroAutoBox III with virtual Ethernet controllers. However, if you use only physical Ethernet controllers, each controller receives only the broadcast messages that match its IP settings.</p>
<b>Suppression of IGMP messages</b>	<p>ConfigurationDesk lets you disable the sending of Internet group management protocol (IGMP) messages. The MicroAutoBox III and SCALEXIO with a Linux®-based operating system (introduced with firmware 5.0, dSPACE Release 2020-B) do not support the suppression of IGMP messages.</p>
<b>Using the same port of virtual and physical Ethernet controllers</b>	<p>The following limitation affects only SCALEXIO systems with QNX, the standard operating system up to dSPACE Release 2020-A. SCALEXIO systems with a Linux®-based operating system that is introduced with firmware 5.0 (dSPACE Release 2020-B) and the MicroAutoBox III are not affected by this limitation.</p> <p>If a UDP Receive function block references a Virtual Ethernet Setup function block and enables the reception of multicast/broadcast messages, unicast UDP messages cannot be received at the port that is used by the UDP Receive function block. This affects the following Ethernet controllers that receive UDP messages via the same port:</p> <ul style="list-style-type: none"> <li>▪ The physical Ethernet controller that is assigned to the Virtual Ethernet Setup function block.</li> <li>▪ All other virtual Ethernet controllers that use the same physical Ethernet controller.</li> </ul> <p>However, other ports of the Ethernet controllers are not affected and can be used to receive unicast messages.</p>

## Limitations for FlexRay Communication

---

<b>Build process aborts due to unresolved Configuration Port block</b>	If the signal chain for FlexRay communication contains an unresolved Configuration Port block, the build process aborts with a code generation error.  Workaround: Delete the unresolved Configuration Port block or the mapping line between it and the FlexRay function block from the application.
--	---

<b>Error hook handler calls for the first FlexRay block in an application process only</b>	This limitation applies if you work with the multiple bus option.  The Last error type and Error counter function ports are updated only for the first FlexRay function block (assigned to the FlexRay bus with configuration ID '-') in the application process, but not for the other FlexRay function blocks (assigned to FlexRay configurations 1, 2, and 3) in the application process. The Any Error Hook event is also triggered for the first FlexRay function block in the application process only.
--	---

## Limitations for SENT Communication

---

<b>Specifying stop behavior</b>	The support of stop values is limited to the output of the last run time value for the SENT Out function block. Stop values are output when the application status changes from running to stopped. <ul style="list-style-type: none"><li>▪ Leave the setting of the Stopped status output property at Keep last run-time values.</li></ul>
---------------------------------	---

## Limitations for Gigalink Communication

---

<b>Limitation for multicore and multi-processing-unit applications</b>	In a multicore or a multi-PU application with Gigalink communication all the channels of a specific Gigalink port must be mapped to data ports from model implementations that are assigned to the same application process.  If you map the channels of one Gigalink port to data ports of model implementations that are assigned to different application processes, this error is detected when the real-time application is executed on the hardware. The real-time application is then terminated and an error message is generated.  Note that ConfigurationDesk does not generate conflicts during signal chain implementation, Gigalink function block configuration, or the build process.
--	--

<b>Synchronized communication</b>	Synchronized communication via Gigalink is supported between SCALEXIO systems and between a SCALEXIO system and a PHS-bus-based system. The PHS-
-----------------------------------	--

bus-based system can only act as an event receiver, while the SCALEXIO systems can be both event receiver and event sender.

Synchronized communication via Gigalink is only supported for the timer task by means of sending the event of a timer task via Gigalink. To achieve synchronized communication, you have to ensure that the timer task configured to be sent via Gigalink has no offset. If an offset is used, this can lead to the two applications having different simulation times.

## Limitations Concerning ECU Interfacing with SCALEXIO or MicroAutoBox III

---

<b>Limitations for ECU interfacing with MicroAutoBox III</b>	ECU interfacing with a MicroAutoBox III is supported only as of EIC file version 4.0.0.
<b>Working view export excludes ECU Interface Configuration function block</b>	When you export the contents of a working view via the Export Signal Chain from Working View command, ECU Interface Configuration function blocks are not exported.
<b>Instantiating ECU Interface Configuration function blocks via copy &amp; paste not supported</b>	You cannot instantiate ECU Interface Configuration function blocks via the Copy and Paste or Paste Multiple commands.
<b>Limitation for updating EIC files in the ConfigurationDesk application</b>	When you update an EIC file in an ECU Interface Configuration function block, neither the model port blocks that are mapped to the function block, nor the related behavior model are updated. If required, you must adapt the model port blocks and the behavior model to the updated ECU Interface Configuration function block manually (e.g., by deleting obsolete mapping lines or creating new model port blocks).

## Limitations Concerning the Build and Download Process

---

<b>Messages of the Simulink® Coder™</b>	Messages of the Simulink® Coder™ are not displayed in ConfigurationDesk. However, these messages are displayed in the MATLAB Command window or in the respective diagnostics window of Simulink.
---	--

<b>Compiler options</b>	ConfigurationDesk does not support compiler options set in the Simulink model. They are ignored during the build process.
<b>Build process aborts after changing the MATLAB version</b>	If you change the MATLAB version connected to ConfigurationDesk, the build process may abort with the following error message in the MATLAB Command Window:  <pre>The existing Simulink project (slprj) folder in the current folder was created for a different version of MathWorks products. This project folder is not compatible with the current version. To continue you must manually remove the slprj folder and any generated code files it contains.</pre> In this case, you must delete the <code>&lt;ProjectRootFolder&gt;\&lt;Project&gt;\&lt;Application&gt;\Components\&lt;ModelName&gt;</code> folder.

## Limitations Concerning MATLAB Compatibility

<b>Introduction</b>	The following limitations apply when you work with MATLAB®.
<b>User S-functions must provide source code</b>	For a <code>&lt;sfunname&gt;</code> user S-function, a <code>&lt;sfunname&gt;.c</code> or <code>&lt;sfunname&gt;.cpp</code> source file has to be present during model code generation. Such a file can be empty if the S-function implementation is provided otherwise, e.g., by a static library.
<b>Model referencing</b>	If you work with model referencing, the following limitations apply: <ul style="list-style-type: none"><li>▪ The build process does not support referenced models which contain model port blocks. It is aborted with an error message. Model port blocks must be used only on the top level of a model referencing hierarchy.</li><li>▪ To access variables of a referenced model during simulation, the Total number of instances allowed per top model option on the Model Referencing page of the Configuration Parameters dialog must be set to One. This means that the referenced model cannot be included multiple times. If the Total number of instances allowed per top model option of a referenced model is set to Multiple, no entries for variables of this model are generated into the TRC file.</li><li>▪ In Simulink, protected models can contain other protected referenced models. Nested protected models are not supported.</li></ul>
<b>Naming conventions</b>	You have to follow the MATLAB naming conventions for files and directories.

<b>Restricted character encoding</b>	<ul style="list-style-type: none"> <li>▪ Block names, signal labels and annotations using the full range of UTF-16 encoding are not supported by ConfigurationDesk. Such characters might lead to problems when you generate code. Use only ASCII characters to avoid this.</li> <li>▪ In addition, to avoid conflicts in the TRC file generation, do not use the following characters in block names and signals:           <ul style="list-style-type: none"> <li>▪ "</li> <li>▪ /</li> <li>▪ \</li> </ul> </li> <li>▪ Variable names, e.g., parameter names, must not contain double underscores. The names also must not start with an underscore followed by an uppercase letter.</li> </ul>
<b>Restricted support of model templates</b>	You cannot use model port blocks from the Model Interface Blockset, in a model template.
<b>Restricted support of incremental code generation for top-level models</b>	<p>The MATLAB Incremental code generation for top-level models feature is supported by ConfigurationDesk with the following restriction:</p> <p>If you perform a build process in ConfigurationDesk, including model code generation, and then just change labels in the Simulink model, Simulink Coder will not recognize this as a change in the model. Thus, Simulink Coder does not trigger a new code generation. As a result, ConfigurationDesk does not generate a new variable description file. The variable description file will contain the old label names.</p>
<b>No locked Simulink models</b>	Locked Simulink models are not supported.
<b>No support for Emulation hardware option</b>	The Simulink® Coder™ option called Emulation hardware on the Hardware Implementation dialog is not supported.
<b>No support of unit checks</b>	Model port blocks do not support automatic unit checks.
<b>Unsupported statements in custom code</b>	Functions that access host hardware, such as file I/O, network access, etc., are not supported in custom code.
<b>No code generation for recursive functions</b>	You are not recommended to use recursive functions when generating real-time applications executed on real-time hardware.
<b>Limitation for Upgrade Advisor API</b>	The <code>upgradeadvisor</code> function lets you analyze and perform required migration steps for your model when you upgrade to a newer MATLAB version. The

Simulink libraries of third-party blocksets that you use, such as the Model Interface Blockset, cannot be upgraded by this function. Do not change the default value of the `SkipBlocksets` argument from `true` to `false`.

<b>Unsupported Simulink blocks</b>	The following new Simulink blocks are not supported: <ul style="list-style-type: none"><li>▪ Initialize Function block</li><li>▪ Reset Function block</li><li>▪ Terminate Function block</li></ul>
<b>No custom folder settings for code generation and simulation fragments</b>	Custom folder settings for code generation and simulation fragments are not supported.
<b>No concurrent execution behavior</b>	Simulink Coder provides the <code>Allow tasks to execute concurrently on target</code> option. The Model Interface Package for Simulink does not support this feature.
<b>Restricted support of message signals</b>	The following applies to message signals provided by the blocks of the Messages & Events library: <ul style="list-style-type: none"><li>▪ Message signals do not appear in the TRC file, or they do not point to readable variables.</li><li>▪ You cannot connect message signals to blocks from dSPACE blocksets.</li><li>▪ Model separation is not supported if the separated models are connected via message signals.</li></ul>
<b>No TRC file entries for string signals</b>	Simulink signals support the String data type. A string signal can be used for modeling, but it does not appear in the generated TRC file, and can therefore not be accessed, e.g., for experimenting in ControlDesk.
<b>No Simulink parameter dependency</b>	You can define a Simulink parameter with a dependency to another Simulink parameter. For example, you can define the Simulink parameter <code>myParamPlusOne</code> with the value <code>myParam + 1</code> . However, due to a Simulink Coder limitation, parameter dependencies are not propagated to the generated code.
<b>Code obfuscation not supported</b>	Simulink Coder provides the <code>ObfuscateCode</code> option. This option is not supported by Model Interface Package for Simulink.

**No support of matrices generated in row major format**

You can specify whether the layout of matrices with two or more dimensions must be *column major* or *row major*. You can specify this for each model. If you select the *column major* option, the first index indicates the column. If you select the *row major* option, the first index indicates the row.

A2L file generation for Simulink models with matrices generated in *row major* format is not supported by Model Interface Package for Simulink.

**No support of overriding parameters of referenced configuration sets**

Overriding parameters of referenced configuration sets is not supported. The Model Interface Package for Simulink does not check if you have specified parameter overrides for your model.

**Note**

The use of parameter overrides can lead to misconfiguration of the behavior model and thus of the generated code.

**No 64-bit integers for A2L file generation**

64-bit integers are not supported for A2L file generation.

**No modification of model port block configurations in subsystem references**

The configuration of model port blocks in subsystem reference instances is read-only. The Model Interface Package for Simulink intercepts attempts to modify the data of model port blocks using an API command or block dialog.

**Limitations when using MATLAB R2020a and newer**

The following limitation applies when you work with MATLAB R2020a and newer:

- The 16-bit floating-point data type can be used with the Model Interface Package for Simulink. However, signals of this data type cannot be connected to model port blocks. Variables of this type are not generated into the variable description file.

**Limitations when using MATLAB R2020b and later**

The following limitations apply when you work with MATLAB R2020b:

- As of MATLAB R2020b, code generation settings for Simulink signals and Data Store Memory blocks can be configured only via the Code Mappings editor. Settings of the Code Mappings editor are not copied to the target models created during model separation. You must configure the code generation settings via the Code Mappings editor also in the target models.
- In Variant Subsystem blocks, Variant Sink blocks, and Variant Source blocks, you can specify variants for Simulink simulation and code generation. If model port blocks are used in such variants, only the active variant of a model port block is recognized during model analysis. The code generation variant is recognized during code generation and the creation of SIC files. Therefore, use model port blocks in such variants only for generating SIC files.

**Limitations when using  
MATLAB R2021a**

As of MATLAB R2021a, Simulink libraries can reference data dictionaries as data sources. A Simulink model that references blocks from these libraries inherits access to the referenced data dictionaries. The Model Interface Package for Simulink does not support model interfaces with data types, for example, `Simulink.Bus` objects, defined in the data dictionaries referenced by libraries.

# 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.....	786
B.....	786
C.....	789
D.....	792
E.....	793
F.....	795
G.....	796
H.....	796
I.....	797
L.....	798
M.....	799
N.....	802
O.....	802
P.....	802
R.....	804
S.....	805
T.....	806
U.....	807

V.....	808
W.....	808
X.....	809

## 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 PDUs](#) 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.

**BSCL 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 Import blocks which are connected to Data Outport blocks. Hence, it also bundles the signals that are received by these blocks. If Data Import blocks are executed within the same [task](#) and belong to the same [communication package](#), their data imports are read simultaneously. If Data Outport blocks that are connected to the Data Import 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 import** A port that supplies data from ConfigurationDesk's function outports to the behavior model.

In a multimodel application, data imports 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 imports.

In a multimodel application, data outports also can be used to supply data to a data import 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 import** 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 (dsmpplib) 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 output (sending model) to a data import

(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 Import 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 import
- Data output
- 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 imports 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 imports](#) 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 import** 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 Import 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.

---

**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.



# Appendix

---

## Where to go from here

## Information in this section

File Types and Keyboard Shortcuts.....	812
Version-Specific Migration Steps.....	820

# File Types and Keyboard Shortcuts

## Where to go from here

## Information in this section

File Types.....	812
Available Keyboard Shortcuts.....	814

## File Types

### Objective

When you work with ConfigurationDesk, you will work with different file types.

#### Note

Do not change file name extensions. This leads to conflicts later on because ConfigurationDesk uses fixed file name extensions for file types. Do not modify the contents of files manually (except for files that let you edit components outside of ConfigurationDesk, such as XLSX files containing a device topology).

### File Types

The following list describes the most common ConfigurationDesk-specific file types:

Extension	Description
ARXML, DBC, LDF, XML	These communication matrix file formats can describe automotive fieldbus communication. You can import such files via the Buses Browser or the Buses ribbon to configure bus communication in a ConfigurationDesk application. For details, refer to <a href="#">Working with Communication Matrices (ConfigurationDesk Bus Manager Implementation Guide)</a> .
BSC	A BSC file is a bus simulation container file that contains the configured bus communication of one application process. BSC files are generated with the Bus Manager.
CAFX	The ConfigurationDesk application fragment file 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	The ConfigurationDesk application file contains links to all the documents related to an application.
CDP	The ConfigurationDesk project file contains links to all the documents related to a project (for example, the hardware topology file, the configuration file, or all existing applications).
CTLGZ	CTLGZ files are ZIP files containing a V-ECU implementation. CTLGZ files are exported from TargetLink or SystemDesk. You can add a V-ECU implementation based on a CTLGZ file to the model topology in the same way as adding a Simulink model based on an SLX file.

Extension	Description
DSA	A dSPACE archive contains an application and all the files belonging to it as one unit. It can later be loaded into another ConfigurationDesk project.
DTFX	The device topology file 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.
ECHX	The external cable harness file 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.
EIC	An ECU interface container file describes an ECU application that is prepared for ECU interfacing. EIC files are generated with the ECU Interface Manager. You can import EIC files to ConfigurationDesk to perform ECU interfacing with SCALEXIO or MicroAutoBox III systems.
EXPSWCFG	The experiment software configuration file contains configuration data for automotive fieldbus communication. It is created during the build process and contains the data in XML format.
FMU	FMU files are used to add model implementations of the Functional Mock-up Unit type (called FMUs below) to the model topology. FMUs are based on the Functional Mock-up Interface standard (FMI standard). The FMI standard defines an interface standard for model exchange and co-simulation. It is supported by different tools, for example, Dymola, MapleSim, and SimulationX. An FMU implements a model using the interfaces defined by the FMI standard.
HTFX	The hardware topology file contains information on a specific hardware system which can be used with ConfigurationDesk. It provides information on the components of the system and on the channel properties, such as board and channel types and slot numbers.
MAP	The MAP file maps symbolic names to physical addresses.
MCD	The model communication description file 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 from dSPACE.
SLX (or MDL)	The Simulink model file contains the behavior model. You can add an SLX file to your ConfigurationDesk application. With MATLAB® R2012a, The MathWorks® file name extension for the Simulink model file was changed from MDL to SLX.
MTFX	The model topology 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.
RTA	The real-time application file is the 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.
SDF	The system description file contains information on the CPU names, the corresponding object files to be downloaded and the corresponding variable description files. It is created during the build process.
SIC	An SIC file is a Simulink implementation container file containing the model code of a Simulink behavior model. An SIC file is generated from a Simulink behavior model using the Model Interface Package for Simulink.
TRC	The variable description file contains all variables (signals and properties) that can be accessed via the experiment software. It is created during the build process. For more information, refer to <a href="#">Information on Variable Description Files (TRC Files)</a> on page 722.

Extension	Description
VECU	VECU files are ZIP files containing a V-ECU implementation. VECU files can contain data packages for different platforms. VECU files are exported from 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.
VSET	A VSET file contains all view sets and their settings of a ConfigurationDesk installation. A VSET file can be exported and imported via the View Sets page of the Customize dialog (refer to <a href="#">Customize Quick Access Toolbar/Customize/More Commands (ConfigurationDesk User Interface Reference)</a> ).
XLSX	The Microsoft Excel™ file formats are used for the following purposes: <ul style="list-style-type: none"> <li>▪ Creating or configuring a device topology outside of ConfigurationDesk. Refer to <a href="#">Importing a Device Topology from a Microsoft Excel Sheet</a> on page 268.</li> <li>▪ Exporting the wiring information for the external cable harness. Refer to <a href="#">How to Export Wiring Information for an External Cable Harness</a> on page 689.</li> <li>▪ Exporting the configuration data of the active ConfigurationDesk application for documentation purposes. Refer to <a href="#">How to Export the Configuration of an Application</a> on page 757.</li> </ul>
XML	The Extensible Markup Language format is used for the ConfigurationDesk project template file: %DSPACE_CONFIG%\ConfigurationDeskTemplate.xml

## Available Keyboard Shortcuts

---

<b>Shortcuts for different areas</b>	Available shortcuts depend on the area you are currently operating in: <ul style="list-style-type: none"> <li>▪ Global operations</li> <li>▪ Navigating ribbons</li> <li>▪ Navigating hierarchies</li> <li>▪ Handling context menus</li> <li>▪ Handling dialogs</li> <li>▪ Switching to panes</li> <li>▪ Operating the backstage view</li> <li>▪ Properties Browser</li> <li>▪ Project management</li> <li>▪ Customizing the display of working views</li> <li>▪ Controlling the build process</li> <li>▪ Executing user functions</li> <li>▪ Automating tool handling</li> </ul>
--------------------------------------	---

---

### Global operations

Shortcut	Purpose
<b>Ctrl+Z</b>	Undoes your most recent action or change in ConfigurationDesk.
<b>Ctrl+Y</b>	Redoes your most recently undone action or change in ConfigurationDesk.

Shortcut	Purpose
<b>Ctrl+C</b> (if applicable)	Copies a selection to the Clipboard.
<b>Ctrl+X</b> (if applicable)	Cuts a selection to the Clipboard.
<b>Ctrl+V</b> (if applicable)	Pastes the content of the Clipboard.
<b>Ctrl+A</b> (if applicable)	Selects all contents of the current pane.
<b>Ctrl+F</b> or <b>Ctrl+Shift+F</b>	Opens the Find (Global) dialog that allows you to search for specific elements of the currently active ConfigurationDesk application.
<b>F1</b>	Opens the dSPACE Help to get help on the currently active context.
<b>F2</b>	Lets you make the name of selected elements of a ConfigurationDesk application editable.
<b>Ctrl+F1</b>	Opens a list of available shortcut keys.
<b>Alt+F4</b>	Exits the current ConfigurationDesk session. If you made any changes to the currently open ConfigurationDesk application, you are prompted to save them.

### Navigating ribbons

Shortcut	Purpose
<b>Alt</b>	Marks each command in the Quick Access Toolbar and each ribbon tab by an access key. Press <b>Alt</b> again to remove the access keys and quit ribbon navigation.
Displayed access keys	Use the access keys to navigate to the desired ribbon command.
<b>Esc</b>	Returns to the first level of ribbon navigation if you have already used an access key.

### Navigating hierarchies

Shortcut	Purpose
Up and down arrow keys	Navigate the currently displayed elements of a hierarchy. Navigating to an element usually selects it.
+ (numeric keypad)	Expands the subelements of the currently selected element (if available). Subelements containing further subelements are not expanded. However, the expand state of sub-hierarchies is preserved.
- (numeric keypad)	Collapses the subelements of the currently selected element (if available). The expand state of sub-hierarchies is preserved.
* (numeric keypad)	Completely expands the hierarchy of subelements of the currently selected element.
<b>Ctrl+Space</b>	Undoes a selection of elements in a pane. Required to access context menus of empty areas.

---

### Handling context menus

Shortcut	Purpose
Menu key	Opens the context menu of the currently active area or selected element.
Arrow keys	Navigate to the command you want to execute (up/down for command selection, left/right for navigating submenus).
<b>Enter</b>	Execute the selected command.
<b>Ctrl</b>	If you press <b>Ctrl</b> while right-clicking an element, the context menu additionally contains the commands that are usually only available after right-clicking an empty area.

---

### Handling dialogs

Shortcut	Purpose
<b>Tab</b>	Access the different elements of a dialog such as edit fields, checkboxes, buttons etc.
Arrow keys	Used for different purposes such as: <ul style="list-style-type: none"> <li>▪ Navigating lists.</li> <li>▪ Increasing/decreasing numeric values.</li> <li>▪ Accessing the different elements of a dialog such as edit fields, checkboxes, buttons etc.</li> </ul>
<b>Space</b>	Used for different purposes such as: <ul style="list-style-type: none"> <li>▪ Selecting or clearing checkboxes.</li> <li>▪ Selecting elements from a list.</li> <li>▪ Using buttons.</li> </ul>
<b>Enter</b>	Use a selected button. In most dialogs, the OK button is preselected so that you can confirm your settings at any time by pressing the <b>Enter</b> key.
<b>Alt</b>	Underlines the characters whose keys you can press to use a button. Using these keys also works without pressing <b>Alt</b> first.

---

### Switching to panes

Shortcut	Pane
<b>Alt+Shift+1</b>	Message Viewer
<b>Alt+Shift+2</b>	Project Manager
<b>Alt+Shift+5</b>	Platform Manager
<b>Alt+Shift+6</b>	Interpreter
<b>Ctrl+Alt+Z</b>	Register Platforms dialog
<b>Ctrl+0</b>	Switches to the top window in the working area (if a window is open).
<b>Alt+F6</b>	Switches to the next pane.

Shortcut	Pane
<b>Alt+Shift+F6</b>	Switches to the previous pane.
<b>Alt+F7</b>	Opens a dialog to select a pane. Hold and use arrow keys for selection.

---

### Operating the backstage view

Shortcut	Purpose
Up/down arrow keys	Navigate the commands and ribbon groups.
<b>Tab</b>	Navigate the commands and elements of a ribbon group.
<b>Space</b> or <b>Enter</b>	Execute the selected command.

---

### Properties Browser

Shortcut	Purpose
<b>Tab</b>	Switches to a different tabbed property group.
Arrow keys	<ul style="list-style-type: none"> <li>▪ Navigate between the Name and Value column (right/left arrow key).</li> <li>▪ Navigate the rows and columns in a table.</li> </ul>
<b>Enter</b>	<ul style="list-style-type: none"> <li>▪ Confirms entered or selected values.</li> <li>▪ Opens a list of existing values already uses for the property in the currently active ConfigurationDesk application.</li> </ul>
<b>Space</b>	<ul style="list-style-type: none"> <li>▪ Opens a list of allowed values.</li> <li>▪ Selects or clears a checkbox.</li> </ul>
<b>Esc</b>	Unlocks a value field without making any changes.

---

### Project management

Shortcut	Purpose
<b>Ctrl+O</b>	Opens a dialog to select a Python script file in the Source Code Editor or a ConfigurationDesk project (CDP) file.
<b>Ctrl+Shift+O</b>	Opens the Select a Project dialog to open a project.
<b>Ctrl+Shift+N</b>	Opens the File – New backstage view, where you can create a new project or add a ConfigurationDesk application to the current project.
<b>Ctrl+S</b>	Saves the current project and/or active ConfigurationDesk application and modified external documents that are part of the project (e.g., script files).
<b>Ctrl+Shift+S</b>	Saves the current project and/or active ConfigurationDesk application.

---

### Customizing the display of working views

Shortcut	Purpose
+ (numeric keypad)	Zoom in.
- (numeric keypad)	Zoom out.
<b>Shift++</b> (numeric keypad)	Increase width of the working view.
<b>Shift+-</b> (numeric keypad)	Decrease width of the working view.
<b>Ctrl+Shift+A</b>	Switch to align blocks mode.
<b>Ctrl+Shift+P</b>	Switch to push blocks up mode.
<b>Ctrl+F4</b>	Closes the current window in the working area. Also works for tables.
<b>Ctrl+F6</b>	Switches to the next window in the working area.
<b>Ctrl+Shift+F6</b>	Switches to the previous window in the working area.

---

### Controlling the build process

Shortcut	Purpose
<b>Ctrl+Alt+B</b>	Starts the build process.
<b>Ctrl+Alt+A</b>	Stops the build process.

---

### Executing user functions

Shortcut	Purpose
<b>Ctrl+Shift+1 – Ctrl+Shift+9</b>	Executes a user function that you added via the Customize command.

---

### Automating tool handling

Shortcut	Purpose
<b>Ctrl+W</b>	Toggles the display of whitespaces and tabulators in the source code.
<b>Ctrl+F</b>	Opens the Find dialog for you to locate text in the source code.
<b>Ctrl+R</b> (Interpreter only)	Opens the Run Script dialog for you to select a Python script to run.
<b>Ctrl+I</b> (Interpreter only)	Opens the Import Module dialog for you to select a Python script to import into the namespace of the Interpreter.
<b>Ctrl+G</b> (Source Code Editor only)	Opens the Go To Line dialog for you to enter a line number you want the cursor to go to.
<b>Alt+3</b> (Source Code Editor only)	Inserts a double comment character (##) at the beginning of all selected lines.

Shortcut	Purpose
<b>Alt+4</b> (Source Code Editor only)	Deletes the comment character(s) (## or #) at the beginning of all selected lines.
<b>Ctrl+Shift+C</b> (Source Code Editor only)	Checks the syntax of the code in the current Python script.

---

**Related topics****Basics**

[Using Keyboard Shortcuts.....](#) 63

# Version-Specific Migration Steps

Where to go from here	Information in this section
	Migrating from ConfigurationDesk 6.5 to 6.6.....820
	Migrating from ConfigurationDesk 6.3 to 6.4.....821
	Migrating from ConfigurationDesk 6.1 to 6.2.....822
	Migrating from ConfigurationDesk 6.0 to 6.1.....823
	Migrating from ConfigurationDesk 5.7 to 6.0.....823
	Migrating from ConfigurationDesk 5.6 to 5.7.....825
	Migrating from ConfigurationDesk 5.5 to 5.6.....825
	Migrating from ConfigurationDesk 5.4 to 5.5.....825
	Migrating from ConfigurationDesk 5.3 to 5.4.....826
	Migrating from ConfigurationDesk 5.2 to 5.3.....826
	Migrating from ConfigurationDesk 5.1 to 5.2.....827
	Migrating from ConfigurationDesk 4.4 to 5.0.....829
	Migrating Projects from ConfigurationDesk 4.2 (on dSPACE Release 7.3) or Earlier to 4.3 or 4.4.....830

## Migrating from ConfigurationDesk 6.5 to 6.6

### **Migrating to the Linux-based operating system on SCALEXIO**

With dSPACE Release 2020-B, dSPACE changes the SCALEXIO firmware from a QNX-based to a Linux-based distribution. The following items built for dSPACE Release 2020-A and earlier are no longer compatible with the SCALEXIO system and must be (re-)built from source code based on dSPACE Release 2020-B:

- Real-time applications
- Binary libraries contained in model containers (i.e., SIC, BSC, FMU, and CTLGZ files)
- Binary libraries referenced by Simulink models
- Binary libraries referenced by ConfigurationDesk applications via custom code settings or custom I/O functions

Source code is expected to be reusable in most cases. Cases that might require an adaptation of C code are related to custom code that uses special features (e.g., OS-specific functions) created by your company or third-party suppliers.

BSC, FMU, and SIC files that were precompiled using the SCALEXIO platform identifier must be precompiled using the SCALEXIO\_LNX identifier. As of dSPACE Release 2020-B, you must always use the SCALEXIO\_LNX identifier to create precompiled BSC, FMU, and SIC files.

---

<b>Project import restricted as of Release 2021-A</b>	As of the dSPACE Release 2021-A, ConfigurationDesk will support the direct import only of projects last saved with one of the previous seven ConfigurationDesk versions.
---	--

---

<b>Changes to the I/O Ethernet communication</b>	<p>The following changes in dSPACE Release 2020-B involve the I/O Ethernet communication.</p> <p><b>Receiving UDP messages</b> The Received Bytes function port of the UDP Receive function block now continuously provides the number of valid data bytes that are provided by the Data Vector function port. Until Release 2020-A, the Received Bytes function port provides the number of valid data bytes only during the model step in which the function block provides new data.</p> <p><b>Unsupported I/O Ethernet feature</b> A SCALEXIO real-time application can suppress Internet group management protocol (IGMP) messages for I/O Ethernet communication.</p> <p>As of dSPACE Release 2020-B, dSPACE introduces a Linux-based operating system for the SCALEXIO Processing Unit and DS6001 Processor Board. The Linux-based operating system does not support the suppression of IGMP messages.</p>
--	---

## Migrating from ConfigurationDesk 6.3 to 6.4

---

<b>Changed naming scheme for bus access requests (available for Bus Manager elements)</b>	The naming scheme for bus access requests has changed from ConfigurationDesk 6.3 to ConfigurationDesk 6.4. The change is the following:
---	---

Old Naming Scheme	New Naming Scheme
<Bus configuration name>:<communication cluster name> (<bus configuration part name>)	<p>Bus access requests available for:</p> <ul style="list-style-type: none"> <li>▪ Simulated ECUs, Inspection, or Manipulation bus configuration part: Bus Access Request [&lt;bus configuration name&gt;\&lt;bus configuration part name&gt;\&lt;communication cluster name&gt;\name of communication matrix in bus configuration]</li> <li>▪ Gateways bus configuration part: Bus Access Request [&lt;bus configuration name&gt;\&lt;frame gateway name&gt;\&lt;communication cluster name&gt;]</li> </ul>

If you work with automation scripts that access bus access requests via their names, you have to adapt the automation scripts if you use them with new ConfigurationDesk projects.

## Migrating from ConfigurationDesk 6.1 to 6.2

### Discontinuation of Python 2.7

The support of Python 2.7 is discontinued with dSPACE Release 2018-B. Python 3.6 is now supported.

#### Note

Python scripts that have been added to a ConfigurationDesk project in a previous ConfigurationDesk version via **Insert Script** or **Import Script** are automatically converted to Python 3.6 when you open the project. The script migration cannot be reverted.

### Discontinuation of SCALEXIO Ethernet Solution

The SCALEXIO Ethernet Solution is discontinued as follows:

- The end-of-life date is January 31, 2021. You can still buy the product up to and including January 31, 2019.
- New Releases of the SCALEXIO Ethernet Solution will still be available for customers with a Software Maintenance Service contract until at least January 31, 2020.
- Customers with a Software Maintenance Service contract who work with dSPACE Release 2018-B will be automatically migrated to the new ConfigurationDesk UDP/TCP function blocks.

For new projects (using dSPACE Release 2018-A and later), we recommend that you use the new UDP/TCP function blocks that are natively integrated in ConfigurationDesk. They provide additional and new options such as IPv6, UDP Multicast support, and enhanced TCP status information.

Note: The dedicated license is required for using the new UDP/TCP function blocks in ConfigurationDesk.

### FPGA custom function blocks with APU functionality

With dSPACE Release 2018-B, the angle range handling of the angular processing unit (APU) changed. FPGA custom function blocks that use the APU in the 360° angle range are incompatible if they are built with the FPGA Programming Blockset 3.5 or earlier.

To resolve the incompatibility, use the FPGA model/code of the incompatible FPGA custom function block and build a new FPGA custom function block with the RTI FPGA Programming Blockset 3.6 or later. The RTI FPGA Programming Blockset automatically migrates the framework of the FPGA model/code to the current version.

## Migrating from ConfigurationDesk 6.0 to 6.1

### Changes regarding the Bus Manager

**Changes to the tool automation interface** Changes have been made to the tool automation interface. Some of these changes affect the data model regarding Bus Manager elements and can cause code from previous Releases to malfunction. For more information, refer to [Version-Specific Migration Steps for ConfigurationDesk Tool Automation \(ConfigurationDesk Automating Tool Handling\)](#).

**Changes in the TRC file** The paths of Bus Manager elements in the TRC file have changed from ConfigurationDesk 6.0 to ConfigurationDesk 6.1. When you build a real-time application or generate bus simulation containers with ConfigurationDesk 6.1, you might have to adapt projects that use the generated TRC file (e.g., generate new instrument layouts in ControlDesk).

## Migrating from ConfigurationDesk 5.7 to 6.0

### Runnable functions from projects created with ConfigurationDesk 5.7 or earlier

If you work with ConfigurationDesk projects created with ConfigurationDesk 5.7 or earlier, the following applies:

If the ConfigurationDesk project contains runnable functions from the Simulink model, the related Runnable Function blocks are displayed on the model root level in the Model-Function Mapping Browser, even if the Runnable Function blocks are not located on the root level of the Simulink behavior model. The name you specified for the runnable function in the Runnable Function block dialog in Simulink is displayed as the Runnable Function block name in the Model-Function Mapping Browser. You have to perform a model analysis of the Simulink model. The Model-Function Mapping Browser then displays these Runnable Function blocks with the correct name and with the correct location in the model hierarchy.

#### Note

This also applies to Simulink implementation container files (SIC files) that have been created with the Model Interface Package for Simulink 3.4 or earlier.

### Propagating changes to Simulink

Propagating changes to a Simulink model overwrites specific configurations in the Simulink model, such as the port configuration of Data Import blocks and Data Outport blocks, and the settings on the Runnable Function page of the Runnable Function block dialog. These parameters are configured based on the configuration of the function blocks to which they are mapped.

If you work with ConfigurationDesk projects created with ConfigurationDesk 5.7 or earlier, keep in mind that propagating changes to Simulink affects all the signal chains, including signal chains that have been created manually with an older ConfigurationDesk version.

**Tip**

To prevent parts of a Simulink model from being accidentally changed by a propagate operation, you can protect specific subsystems by setting them to read-only in Simulink. When analyzed, these subsystems are marked with a lock symbol in ConfigurationDesk.

---

**ConfigurationDesk projects containing FlexRay communication**

As of ConfigurationDesk 6.0, function blocks have event ports for I/O events. If you work with ConfigurationDesk projects that contain FlexRay communication and that were created with ConfigurationDesk 5.7 or earlier, the following applies: Each instantiated FlexRay function block provides event ports, regardless of whether you use the related I/O events in your Simulink model:

- Event ports related to I/O events that you use in your Simulink model are automatically mapped to Runnable Function blocks. In this case, you must analyze your Simulink model.
- Event ports related to I/O events that you do not use in your Simulink model are unmapped. You cannot disable or delete event ports from FlexRay function blocks. Instead, you can leave the event ports unmapped. Keep in mind that each time you propagate changes to Simulink, Runnable Function blocks are automatically created for these event ports.

---

**Inconsistency with Ethernet adapters**

When you migrate a project of dSPACE Release 2016-A or earlier, the migration process adds the Ethernet adapter of the SCALEXIO Real-Time PC to the hardware topologies of the migrated project. The default name of the added Ethernet adapter might not match the Ethernet adapter name of the accessible platforms. Therefore, the status bar shows the status "No matching platform connected". This status prevents the automatic download of the real-time application after the build. However, you can build the real-time application and manually download it to the hardware.

To resolve the inconsistency, specify an identical name for the Ethernet adapter of the SCALEXIO Real-Time PC in the Hardware Resource Browser and in the Platform Manager, e.g., by replacing the hardware topology.

---

**Changes in TRC file for Bus Manager elements**

The paths of Bus Manager elements in the TRC file have changed from ConfigurationDesk 5.7 to ConfigurationDesk 6.0. When you build a real-time application or generate bus simulation containers with ConfigurationDesk 6.0, you might have to adapt projects that use the generated TRC file (e.g., generate new instrument layouts in ControlDesk).

## Migrating from ConfigurationDesk 5.6 to 5.7

<b>Changes in TRC file for Bus Manager elements</b>	The paths of Bus Manager elements in the TRC file have changed from ConfigurationDesk 5.6 to ConfigurationDesk 5.7. When you build a real-time application with ConfigurationDesk 5.7, you might have to adapt projects that use the generated TRC file (e.g., generate new instrument layouts in ControlDesk).
<b>Discontinuation of SYNECT server connection</b>	As of dSPACE Release 2017-A, ConfigurationDesk can no longer connect to the SYNECT server. As a result, exchanging data, such as build results required for ControlDesk, with the SYNECT server is no longer possible.

## Migrating from ConfigurationDesk 5.5 to 5.6

<b>Inconsistency Ethernet adapters</b>	When you migrate a project to dSPACE Release 2016-B, the migration process adds the Ethernet adapter of the SCALEXIO Real-Time PC to the hardware topologies of the migrated project. The default name of the added Ethernet adapters is "[MAC address 00:00:00:00:00] no name assigned". This name does not match the Ethernet adapter name of the accessible platforms. Therefore, the status bar visualizes the status "No matching platform connected". This status prevents the immediate download of the real-time application after the build.  To resolve the inconsistency, specify an identical name for the Ethernet adapter of the SCALEXIO Real-Time PC in the Hardware Resource Browser and in the Platform Manager.
--	--

## Migrating from ConfigurationDesk 5.4 to 5.5

<b>Possible M script adjustments for automation</b>	<b>Value returned for ICaApplicationMain: SetCustomInformation and ICaComponent: Configure</b> <b>ICaApplicationMain: SetCustomInformation</b> and <b>ICaComponent: Configure</b> now return a value. In most cases, this value is <code>None</code> (e.g., in Python).
	<p><b>Note</b></p> <p>Even if the returned value is not used, M script clients should be aware that a printout follows after calling one of the methods if no semicolon ends the statement. This can cause unexpected output from existing scripts.</p>

<b>Hardware topology containing a DS2671 Bus Board</b>	If you migrate to dSPACE Release 2016-A and your hardware topology contains a DS2671 Bus Board, note the following: After migration, the DS2671 does not support the features and configuration settings introduced with former dSPACE Releases (e.g., CAN FD mode). Workaround: Replace the hardware topology after migration.
<b>Discontinuation of platform management automation API version 1.0 as of dSPACE Release 2016-B</b>	The platform management automation API version 1.0 was supported for the last time with ConfigurationDesk 5.5 of dSPACE Release 2016-A.

## Migrating from ConfigurationDesk 5.3 to 5.4

---

<b>Changes in TRC file generation</b>	You have to note some modifications on TRC file generation in ConfigurationDesk. For details, refer to the <a href="#">New Features and Migration document of dSPACE Release 2015-B</a> .
---------------------------------------	---

## Migrating from ConfigurationDesk 5.2 to 5.3

---

<b>New XML schema for custom function blocks</b>	A new XML schema was implemented for custom function blocks (see <code>&lt;InstallationFolder&gt;\ConfigurationDesk\Implementation\UserFile s\CustomFunctionTypeSchema_V3_1.xsd</code> ). ConfigurationDesk lets you update custom function files to the new schema via the <b>Create Updated Custom Function Type XML</b> command.
--	---

### Note

You cannot downgrade an updated custom function file to the previous schema version. Make sure to back up the existing files before the update.

<b>Using parallel build for referenced models</b>	If you work with MATLAB R2015a, the following applies:  If you build referenced models with activated MATLAB workers, the build might fail or the results might be erroneous. You have to remove all build results from modified referenced models in the <code>s1prj</code> folder beforehand. To solve this problem, you can install the following MathWorks® patch: <a href="http://www.mathworks.com/support/bugreports/1199590">http://www.mathworks.com/support/bugreports/1199590</a> .
---	---

## Migrating from ConfigurationDesk 5.1 to 5.2

<b>Pre-4.3 projects and component files no longer supported</b>	ConfigurationDesk projects from ConfigurationDesk 4.2 or earlier cannot be migrated to ConfigurationDesk 5.2. Pre-4.3 Application component files (DTF, ECH, HTF, MTF) can also no longer be migrated.
<b>Modifications concerning the TRC file generation</b>	<p>The following settings in the Code Generation - DSRT variable description file options page of the Configuration Parameters dialog have been changed:</p> <ul style="list-style-type: none"> <li>▪ The Apply subsystem read/write permissions setting is no longer available.</li> <li>▪ The following settings are now cleared by default:           <ul style="list-style-type: none"> <li>▪ Include signal labels</li> <li>▪ Include virtual blocks</li> </ul> </li> </ul> <p>The new default values are only relevant for new models or if you switch your model to another platform.</p> <ul style="list-style-type: none"> <li>▪ The Include only Simulink.Parameter and Simulink.Signal objects with global storage class setting now also considers the BusSystems group of the RTI CAN MultiMessage Blockset and the RTI LIN MultiMessage Blockset.</li> </ul>
<b>Handling of Bus Selector block changed</b>	<p>With MATLAB R2014a, the Simulink Bus Selector block has been virtualized. In combination with dSPACE Release 2014-B, the outports of this block and other virtual blocks that are connected to it are not generated into the variable description file if the inport of the Bus Selector block is connected to a non-virtual bus.</p> <p>To access a signal of a bus do the following:</p> <ul style="list-style-type: none"> <li>▪ Access the signal directly from the bus,</li> <li>or</li> <li>▪ Connect a non-virtual block to the signal, for example, a Gain block with 1 as the factor. Its outport is available in the variable description file.</li> </ul> <p>If the inport of the Bus Selector block is connected via a virtual bus, the outports are generated into the variable description file without applying special methods.</p>
<b>Generated code from Simulink Coder has changed</b>	<p><b>Using MATLAB R2014a</b> With the MATLAB/Simulink R2014a release, code generation by Simulink Coder has changed. Therefore, also the generation of variable descriptions (TRC files) in ConfigurationDesk has changed.</p> <p>Using dSPACE Release 2014-B in combination with MATLAB R2014a requires to use the <code>revertInlineParametersOffToR2013b</code> command before you start a build process. This command enables the Simulink Coder behavior and dSPACE TRC file generation in MATLAB R2014a similar to that in MATLAB R2013b and</p>

before. The `revertInlineParametersOffToR2013b` command is shipped with MATLAB R2014a as part of the Simulink Coder product.

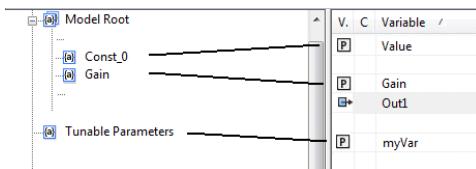
**Using MATLAB R2014b** The modification in the Simulink Coder affects the entries in the generated TRC file only if you have worked with the Inline Parameters optimization option cleared. Workspace variables that are referenced by one or more block parameters are now handled as global parameters in the model and not as local parameters of the block. They are stored to the Tunable Parameters group in the generated TRC file. If you now change the value of such a global parameter, any block parameter referencing this global parameter changes as well. The behavior of your simulation will change.

Example:

Workspace variables are generated as local block parameters in the following cases:

- The parameter is a structure or a structure item, for example, `myStruct.Value`.
- The parameter is used in an expression or in a function, for example, `myValue + 1` or `sin(myValue)`.

Local block parameters are also generated if references to mask parameters are used. The mask parameter itself is not generated into the variable description.



In the model, you have configured the `Const_0` block parameter and the `Gain` block parameter by using the workspace variable `myVar`. Independently of the Inline parameters option on the Optimization page, the workspace variable is generated into the variable description. The variables can be used in ControlDesk Next Generation as before by connecting `Const_0/Value` and `Gain/Gain` to instruments, but internally, the parameter values depend on the value of `myVar`. If you modify one of the parameters, all the other parameters are changed too.

---

#### Changed custom function path in variable description file

The path to the variables of custom functions in the variable description file has been changed. For example, when you reload a variable description file from ConfigurationDesk 5.2 in ControlDesk Next Generation the connections of layouts and signal generators to these variables are lost. You must reconnect these variables.

## Migrating from ConfigurationDesk 4.4 to 5.0

### Migrating a custom function block from ConfigurationDesk 4.4

With ConfigurationDesk 4.4, a project-specific search path for custom functions was added. Projects using this search path do not require any adjustments.

If some or all custom functions types are unresolved after loading a project, you can perform one of the following actions:

- Copy the XML file of each custom function type and, if available, the header file (`<Function_block_type_name/CModule_name>.h`), the C++ source code file (`<Function_block_type_name/CModule_name>.cpp`), and the type definition file (`<Function_block_type_name/CModule_name>_TypeDef.h`) either to the project-specific search path or to the global search path.

The project-specific custom function directory is:

```
<DocumentsFolder>\<Project>\CustomFunctions
```

The default global search path for custom functions is:

```
<InstallationFolder>\ConfigurationDesk\Implementation\UserFiles
```

Since this directory is normally read-only, you should change the path in the Settings dialog, which you can open via File – Settings (see [Configuration Page \(ConfigurationDesk User Interface Reference\)](#)).

Afterwards, the function block type can be resolved via Reload Definition from its context menu.

- Change the global search path for custom functions to a folder containing the missing XML files. The corresponding custom function types are resolved automatically.

If an existing header or source file is not found, open the corresponding XML file of the custom function type with a suitable editor and make sure that in the `<CModule Name="XXX">` tag, XXX is the same as the `<Function_block_type_name/CModule_name>`.

If you only have the XML file, you can create the header file and the C++ source code file via Create Custom Function Code and the type definition file via Create Custom Function Type Definition from the context menu of a custom function block type.

### Migration of SYNECT database

If you work with the SYNECT database, ConfigurationDesk 5.0 supports only SYNECT 1.2. You have to migrate a SYNECT database with ConfigurationDesk data from a previous SYNECT database version.

The SYNECT server provides a central database. Migrating SYNECT's database therefore affects all of the client users that connect to the SYNECT server. This means you have to migrate the database in a central process.

## Migrating Projects from ConfigurationDesk 4.2 (on dSPACE Release 7.3) or Earlier to 4.3 or 4.4

<b>Reasons for migration</b>	As of version 4.3, ConfigurationDesk has a new software architecture with improved software handling and also some new features. To use projects and applications from ConfigurationDesk 4.2 and earlier with ConfigurationDesk 4.3 or 4.4, you have to migrate them beforehand.
<b>Running the migration</b>	<p>You can migrate a project created with ConfigurationDesk 4.2 or earlier automatically by opening it with ConfigurationDesk as usual. ConfigurationDesk detects projects which need to be migrated and opens the migration dialog. When you click <b>OK</b>, migration starts and converts all the data and applications belonging to the project.</p> <p>At the end of the migration process, the project opens automatically and you can work with it in ConfigurationDesk 4.3 or 4.4.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"><p><b>Note</b></p><p>You cannot open and use a migrated project with ConfigurationDesk 4.2 or earlier.</p><p>During the migration process, a backup ZIP file is created for each application in the project. The ZIP files contain the old data and can be used for later restoration. Each <code>&lt;Application name&gt;.migrationbackup.zip</code> file is saved to the application root level.</p></div>
<b>Changes after migration</b>	<p><b>Migration of topology files</b> You can separately migrate topology files (DTF, HTF, or MTF) from ConfigurationDesk 4.2 or earlier by selecting the <b>Import xxx topology</b> from file option in the Replace XXX dialog and browsing for the respective file extension.</p> <p><b>Migration of Simulink models</b> Simulink models which are added to the ConfigurationDesk application are migrated the first time you open the model from ConfigurationDesk 4.3 or 4.4, for example, if you perform a model analysis.</p> <p><b>Changes related to the Project Manager</b> The following changes affect projects and ConfigurationDesk applications:</p> <ul style="list-style-type: none"><li>▪ A ConfigurationDesk application no longer contains the Configuration application component. The CDS file which holds the data of the Configuration application component is no longer provided or supported. As of ConfigurationDesk 4.3, all application data (topology data and signal chain data) is stored in one internal zip file. You can import and export the data to reuse it in other projects.</li></ul>

- The topology files now are used only to exchange single topology data between different applications. The file name extensions of topology files have changed, for example, the file name extension of the hardware topology file has changed from HTF to HTFX.

**Assessing properties of elements** A central Properties Browser is now always available on the screen to display and configure properties, for example, of the signal chain elements. The Properties Browser replaces the properties dialogs of earlier software versions, where you had to open and close the dialogs for each element separately.

**Handling conflicts** The new concept of ConfigurationDesk allows very flexible configuration settings, with the effect that one setting might not match a setting at another place in the signal chain. These configuration conflicts are initially allowed, but before you build a real-time application, you have to resolve the conflicts to get proper build results. The new Conflicts Viewer shows all the conflicts and helps you to resolve them.

**Implementing tasks** Up to ConfigurationDesk 4.2, the implementation of tasks and components that influence the timing behavior of the executable application (= real-time application) is adopted from the behavior model (Simulink). ConfigurationDesk 4.3 and later allows you to model executable applications and the tasks used within them very flexibly.

**Changes related to the model interface** The changes after migration are as follows:

- In the Model Port Block Library, the Trigger Event Port block has been replaced with the Runnable Function block, which exports a function-call subsystem as a runnable function. A runnable function groups together all the behavior model components that must be called for computing results in a specific task.
- In ConfigurationDesk:
  - You can access runnable functions (provided by the Runnable Function block in the behavior model) after model analysis. With the executable application and the task modeling feature, you can assign the runnable functions to (periodic or asynchronous) tasks for execution.

**Note**

Runnable functions are not part of the model topology and therefore you cannot access them via the Model Browser.

- Trigger Event Port blocks that allow the transfer of asynchronous ConfigurationDesk function events to a Simulink behavior model no longer exist.
- I/O function event ports (at function blocks) no longer exist. Access to the I/O events now is realized via the executable application and task modeling feature, where you can assign I/O events to tasks to trigger them.

**Undo/Redo** As of ConfigurationDesk 4.3, the software has undo/redo functionality. You can undo/redo actions or changes that you carried out beforehand.

**Rebuild detection for ControlDesk Next Generation** ConfigurationDesk provides specific build information so that ControlDesk Next Generation can detect whether new build results are available.

**Limitations**

The following limitations exist for migrating projects:

**Obsolete and unresolved elements** Obsolete elements in a signal chain are not migrated. Unresolved elements of the hardware topology and the model topology are not migrated. However, unresolved elements of the device topology are migrated.

**Working view layouts** User-defined working view layouts for customized layout settings are not migrated and not supported with ConfigurationDesk 4.3 and later.

**More than one DS2907 Battery Simulation Controller cannot be differentiated** If a project contains a hardware topology with more than one DS2907 Battery Simulation Controller inside the same rack, it will contain only one DS2907 with the settings of the last controller after migration.

**Properties of the Gigalink electrical interface unit not available** If your project contains Gigalink function blocks, the properties of the electrical interface unit initially are not available in user-defined working views after migration.

To access the properties of the electrical interface unit, use the global working view or re-add the function block to the user-defined working view via copy and paste.

**Migrating a custom function block from ConfigurationDesk 4.2****Note**

Do not open your projects which contain custom function blocks from ConfigurationDesk 4.2 before you have migrated your custom function blocks as follows.

1. Change the global search path for custom functions via Tools – Options and Settings .The default path is:

```
<InstallationFolder>\ConfigurationDesk\Implementation\UserFiles
```

Since this folder is normally read-only, you must change the path to a folder that you have write access to.

2. Copy the XML file of each custom function block and, if available, the header file (`<Function_block_type_name>.h`) and the C++ source code file (`<Function_block_type_name>.cpp`) to the global search path.

#### Note

The `<Function_block_type_name>` must be identical to the XML file name to avoid malfunction.

#### Tip

The type definition file (`<Function_block_type_name>_TypeDef.h`) is not available for custom functions from ConfigurationDesk 4.2. It will be automatically created during a build process.

If you require the type definition file prior to the build process, you can manually create it via **Create Custom Function Type Definition** in the context menu of an instantiated custom function block.

3. Start ConfigurationDesk and open your project.

If you only have the XML file, you can create the header file and the C++ source code file via **Create Custom Function Code** from the context menu of an instantiated custom function block.

### Migrating a custom function block from ConfigurationDesk 4.3 to 4.4

1. Change the global search path for custom functions via Tools – Options and Settings .The default path is:

```
<InstallationFolder>\ConfigurationDesk\Implementation\UserFiles
```

Since this folder is normally read-only, you must change the path to a folder that you have write access to.

2. Copy the XML file of each custom function block and, if available, the header file (`<Function_block_type_name>.h`), the C++ source code file (`<Function_block_type_name>.cpp`), and the type definition file (`<Function_block_type_name>_TypeDef.h`) to the global search path.
3. Open the XML file of each function block with a suitable editor and make sure that in the `<CModule Name="XXX">` tag XXX equals the `<Function_block_type_name>`. Save your changes if required.
4. Start ConfigurationDesk and open your project.

If you only have the XML file, you can create the header file and the C++ source code file via **Create Custom Function Code** and the type definition file via **Create Custom Function Type Definition** from the context menu of an instantiated custom function block.



**A**

accessing hardware properties 222  
 accessing hardware simultaneously  
     access levels 233  
     error messages 235  
 accessing model port block and model port  
 data 466  
 activating a ConfigurationDesk application 101  
 adding  
     custom device properties 281  
     device topology elements to the signal  
     chain 288  
     function blocks via copy and paste 309  
     model port blocks to the signal chain via  
     function blocks 435  
     model port blocks to the signal chain via the  
     Model Browser 436  
 adding a ConfigurationDesk application 100  
 adding function block to the signal chain via  
 function browser 307  
 adding function blocks to the signal chain via  
 device ports 308  
 adding variables to the TRC file 736  
 analyzing the interface of the Simulink behavior  
 model 561  
 application 86  
 application components 105  
 application process 472, 476  
 application processes  
     providing default tasks 521  
 application states of a real-time application 753  
 assigning  
     device ports to pins 255  
     reference ports 274  
 assigning device ports to pins 278  
 assigning hardware resources  
     methods 404  
 assigning reference ports 276  
 asynchronous tasks  
     modeling 502  
 automatic hardware resource assignment 406

**B**

backup of projects 98  
 BSC  
     precompiled 602  
 BSC file 596  
 build result files 702  
 build results 105  
 building signal chain  
     for CAN communication 336  
     for communication via ethernet  
         interface 362  
     for FlexRay communication 351  
     for FPGA applications 369  
     for Gigalink communication 363  
     for LIN communication 344  
 bus simulation container 596

**C**

CAN communication  
     building signal chain 336  
 cfusr.cpp 371  
 cfusr.h 371  
 changing a system name 223  
 channel request 401  
 channel set 322  
 characteristics of device blocks 258  
 Common Program Data folder 20, 790  
 communication between behavior models 448  
 communication via ethernet interface  
     building signal chain 362  
 compatibility check 746  
 configuring  
     tasks 513  
 configuring external devices  
     basics on configuring 272  
 configuring tasks 513  
 copying device topology elements 263  
 creating a project 92  
 creating device topologies  
     creating via external device browser 260  
     editing in Microsoft Excel™ 268  
 creating hardware topologies from scratch 241  
 cycle time restriction 488

**D**

data acquisition raster 482  
 Delay 638  
 delayed task 517  
 deleting custom properties 282  
 device blocks 257  
 device pin assignment 255, 278  
 device port mapping  
     conflicts and effects 315  
     methods 312  
 device port type  
     bidirectional port 273  
     inport 273  
     outport 273  
     reference port 273  
     undefined port 273  
 device topology 252  
 device topology (DTFX file) 105  
 device topology elements  
     copying 263  
     moving 265  
     rename 276  
 device type  
     ECU 273  
     Load 273  
     specifying 273  
 displaying accessible and used licenses 43  
 Documents folder 20, 792  
 downloading a real-time application 748  
 DS1403FwArchive 216  
 DsVdOmit 738  
 DTFX file 105

**E**

ECHX file 105  
 elements  
     filtering 167  
 event 480  
 event port 305  
     characteristics 305  
     possible states 305  
 excluding variables from the TRC file 736  
 executable application 472  
     components 475  
     modeling 475  
 export  
     a ConfigurationDesk application 102  
     a device topology 267  
     wiring information 689  
 extending device topologies  
     extending via external device browser 260  
     merging device topologies 265  
 extending existing hardware topologies 243  
 external cable harness 683  
 external cable harness (ECHX file) 105  
 external loads  
     definition 385  
     handling with 395  
 workflow for specifying 395

**F**

failure simulation  
     introduction 373  
     load rejection and signal dropping 378  
     providing data to experiment software 381  
     specifying failure simulation settings 374  
 file types 812  
 filtering  
     elements 167  
 filters 158  
 finding  
     elements of a ConfigurationDesk  
     application 129  
 firmware  
     limitation for SCALEXIO systems 218, 764  
     update instructions 219  
 FlexRay communication  
     building signal chain 351  
 FMI 608  
 FMU 608  
     Import 611  
     integrating 607  
     IP protected 618  
     limitations 773  
     precompiled 618  
     update 615  
 FPGA applications  
     building signal chain 369  
     port-specific user functions 371  
 function block 301  
     adding to the signal chain via device  
     ports 308

- adding to the signal chain via function  
 browser 307  
 configuration 318  
 deleting 297  
 event port 305  
 function block properties 318  
 function port 303  
 functions 298  
 logical signals 299  
 managing configuration data 320  
 methods for instantiating 293  
 possible states 300  
 purpose 297  
 rename 324  
 signal port 301  
 function block types 292  
 function blocks  
   adding via copy and paste 309  
 Function Browser 292  
 function library 292  
 function port 303  
   characteristics 304  
   possible states 304  
 function triggers 328  
 Functional Mock-up Unit 608  
   integrating 607
- G**  
 Gigalink communication  
   building signal chain 363  
 Global working view 171
- H**  
 handling errors with real-time applications 755  
 handling exceptions with real-time  
   applications 755  
 hardware resource  
   basics on assigning 399  
   definition 321  
 hardware resource assignment  
   automatic 406  
   basics 399  
   channel request 401  
   logical signal 401  
   manual 404  
   methods 404  
   possible conflicts 409  
   via drag & drop 405  
 hardware resource assignment via drag &  
 drop 405  
 hardware topology (HTFX file) 105  
 HTFX file 105
- I**  
 importing  
   a device topology 110  
   a hardware topology 115  
   a model topology 112  
   an external cable harness 117  
 Importing
- FMUs 611  
 importing a ConfigurationDesk application 102  
 importing a device topology 268  
 initializing Simulink behavior models 546  
 inline parameter 320  
 internal loads  
   definition 385  
   handling with 392  
   load compare check 394  
   providing data to hardware topology 397  
   workflow for specifying 392
- K**  
 keyboard shortcuts 63
- L**  
 LIN communication  
   building signal chain 344  
 load rejection 389  
   enforcing 390  
 Local Program Data folder 20, 798  
 logical signal 401
- M**  
 manual hardware resource assignment 404  
 MAP file 702  
 mapping conflicts  
   device port mapping 315  
 mapping methods  
   device port mapping 312  
   model port mapping 438  
 mapping rules  
   model port mapping 443  
 mapping states  
   model port mapping 445  
 MCD file 105  
 MDL/MTFX file 105  
 merging device topologies 265  
 missing hardware resources 239  
 model communication 448  
 Model Communication Browser 173  
 model port block  
   rename 468  
 model port blocks 426  
   duplicate identities 578  
 model port mapping  
   mapping rules 443  
   methods 438  
   states and effects 445  
 model ports 429  
 model topology (MDL/MTFX file) 105  
 modeling  
   asynchronous tasks 502  
 modeling asynchronous tasks 502  
 modeling executable application from  
 scratch 489  
 moving device topology elements 265  
 multicore real-time application 472  
 multi-processing-unit application 472  
 multi-processing-unit applications 525
- multi-processing-unit system 472  
 multi-PU application 472  
 multi-PU applications 525  
 multi-PU system 472  
 multitasking 488
- N**  
 navigation bar 50
- O**  
 opening a project 93  
 overview of the build process 698
- P**  
 parameter categories  
   inline parameter 320  
   tunable parameter 321  
 pin address 257  
   absolute address 257  
   relative address 257  
 port group address 253  
   absolute address 253  
   relative address 253  
 processing unit 472  
 processing unit application 472  
 project 86  
 Properties Browser 131  
 PU 472
- Q**  
 Quick Access Toolbar 65
- R**  
 real-time application 472  
 real-time hardware registering 202  
 reference ports  
   assigning 276  
 registering real-time hardware 208  
 replace model 498  
 ribbon 54  
 RTA file 702  
 rules for editing device topologies 268  
 runnable function 480  
 Run-Time-Tunable parameter 321
- S**  
 SCALEXIOFWArchive 215  
 SDF file 702  
 searching  
   a ConfigurationDesk application 129  
 Set Column Filter 167  
 SIC  
   IP protected 592  
   precompiled 592  
 Signal Chain Browser 173  
 signal port  
   characteristics 302  
   possible states 302

Simulink tasking modes 488  
single-core real-time application 472  
single-processing-unit system 472  
single-PU system 472  
single-tasking 488  
specifying a project root folder 96  
specifying device port types 273  
starting a real-time application 751  
starting the build process 719  
state of device blocks 259  
state of device ports 259  
state of port groups 259  
Stop-Run-Tunable parameter 321  
structure of device blocks 258  
structure of device topologies 253  
SubArray 741

**T**

Table 151  
task 480  
  delayed 517  
  modeling 486  
task priority restriction 487  
Temporary working view 171  
tracing a model port back to Simulink 566  
transferring port settings 317  
TRC file 702, 722  
  Diagnostics group 729  
  Signal Chain group 727  
  Simulation and RTOS group 726  
  structure for MicroAutoBox III systems 724  
  structure for SCALEXIO systems 724  
  tunable parameter 727  
  XIL API/EESPort group 730  
trigger  
  avoiding multiple function triggers 332  
  function triggers 328  
triggering runnable functions 511  
tunable parameter 321  
  in variable description file 727

**U**

updating  
  FMU 615  
user TRC file 736

**V**

variable description  
  configuration settings 723  
  consuming products 722  
  data source 723  
  dSPACE configuration settings 723  
  generating products 722  
  Simulink configuration settings 723  
variable description file 722  
  Diagnostics group 729  
  Signal Chain group 727  
  Simulation and RTOS group 726  
  structure for MicroAutoBox III systems 724  
  structure for SCALEXIO systems 724

tunable parameter 727  
XIL API/EESPort group 730  
variant handling 578  
variants  
  behavior model 578  
V-ECU  
  implementation 625  
  implementation container 625  
V-ECU implementation 625  
V-ECU implementations  
  delay 638  
view set 50  
  customize 57  
viewing circuit diagrams 325

**W**

working view 170

