ConfigurationDesk

Getting Started

For ConfigurationDesk 2024-B (24.3)

Release 2024-B – November 2024

dSPACE

How to Contact dSPACE

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

How to Contact dSPACE Support

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

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

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

If possible, always provide the serial number of the hardware, 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 https://www.dspace.com/go/patches for the software updates and patches themselves and for more information, such as how to receive an automatic notification when an update or a patch is available for your dSPACE software.

Important Notice

Contents

About This Document

Contents

- Describes the basic concepts of ConfigurationDesk and different workflows to implement a real-time application in ConfigurationDesk.
- Provides required information on new features, migration steps, and compatibility for the current ConfigurationDesk version.
- Provides an overview of the documents which are available for ConfigurationDesk and also for the SCALEXIO, MicroAutoBox III, and MicroLabBox II hardware.

Printed document

A printed copy of this document is available on demand.
You can order it free of charge by using the following link:
https://www.dspace.com/go/requestreleasematerial

Symbols

dSPACE user documentation uses the following symbols:

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

Windows-based software products use the following special folders:

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

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

or

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

Documents folder A standard folder for application-specific files that are used by the current user.

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

Local Program Data folder A standard folder for application-specific program data that is used by the current user.

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

Accessing dSPACE Help and PDF files

After you install and decrypt Windows-based 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**

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

dSPACE Help (Web) Independently of the software installation, you can access the Web version of dSPACE Help at https://www.dspace.com/go/help.

To access the Web version, you must have a *mydSPACE* account.
For more information on the mydSPACE registration process, refer to https://www.dspace.com/faq?097.

Features of ConfigurationDesk

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, MicroAutoBox III, and MicroLabBox II 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 applications are organized in projects.	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.	–
Implementing CAN and LIN communication by using the Bus Manager	The Bus Manager in ConfigurationDesk lets you configure CAN and LIN communication for simulation, inspection, and manipulation purposes. You work with communication matrices that serve as a library from which you can select the elements to work with.	–

Feature	Description	
	License-Protected Features	**License-Free Features**
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.	–
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.

Basic Concepts of ConfigurationDesk

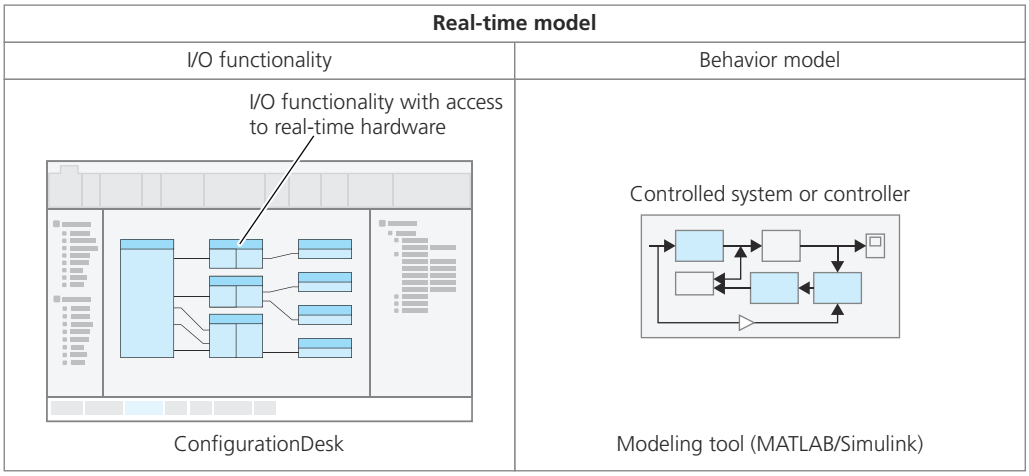
Where to go from here

Information in this section

Separation of I/O Functionality in ConfigurationDesk and Behavior Model

Implementing a real-time model

Your real-time application is based on a real-time model, which is strictly divided into the I/O functionality in ConfigurationDesk and the behavior model in a modeling tool (MATLAB/Simulink). This is shown below:



The functions for measuring and generating I/O signals, and access to the real-time hardware, are implemented in ConfigurationDesk. The behavior model contains only the algorithm of the controlled system (or the control algorithm). It does not contain I/O functionality and access to the hardware. For modeling in MATLAB/Simulink, you can use Simulink Blocksets and Toolboxes from the MathWorks®. At last Simulink behavior models must be prepared for the use in ConfigurationDesk with the features provided by the Model Interface Package for Simulink.

This concept gives you more flexibility. For example, you can change components easily while leaving other parts of the real-time model unchanged.

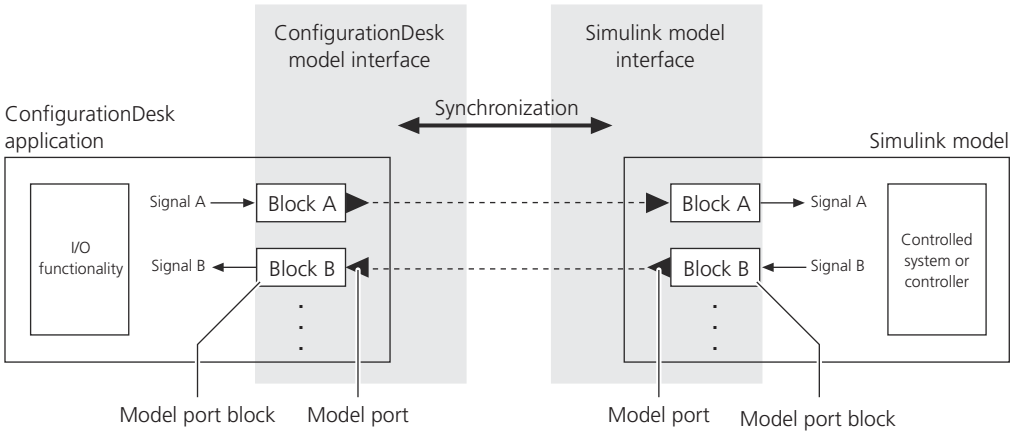
Implementing bus communication Bus communication (CAN, LIN, FlexRay) is an exception. This can be modeled in the behavior model via specific RTI blocksets (for example, the RTI CAN MultiMessage Blockset and RTI LIN MultiMessage Blockset). Then only access to the real-time hardware is added in ConfigurationDesk.

Alternatively, you can use the Bus Manager in ConfigurationDesk to configure and implement CAN and LIN communication. Note, that the Bus Manager does not support FlexRay communication.

Model interface

To connect the I/O functionality in ConfigurationDesk with the behavior model, you need a model interface. This interface is realized via model port blocks containing at least one model port.

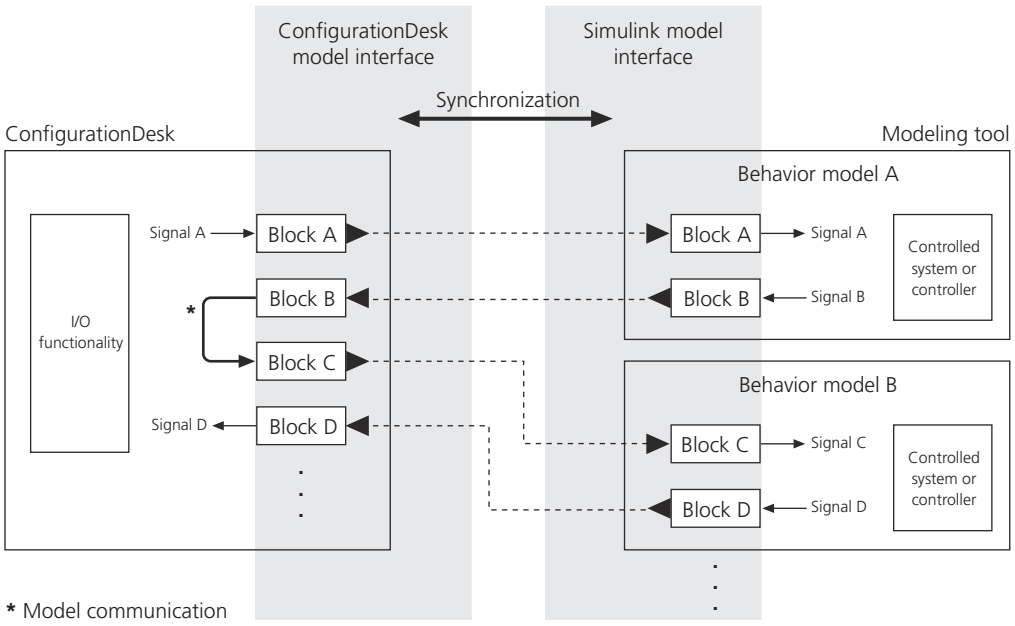
The model port blocks can be created in ConfigurationDesk or in the behavior model. You can synchronize the model interfaces from within ConfigurationDesk to make the blocks available in both models.



Each model port in ConfigurationDesk corresponds to exactly one model port in the behavior model. Connecting a signal to a model port in ConfigurationDesk makes the signal available at the corresponding model port in the behavior model and vice versa.

This concept allows you also to use interfaces which do not match completely. For example, you can work with model interfaces containing model port blocks which differ between ConfigurationDesk and the behavior model. Thus, the ConfigurationDesk model interface can have more model port blocks than the Simulink model interface and vice versa.

With ConfigurationDesk, you can implement a multimodel application where several behavior models are executed in parallel on the dSPACE real-time hardware (SCALEXIO, MicroAutoBox III). A shown in the illustration below, the Configurationdesk model interface is linked to several behavior models.



Synchronization of data

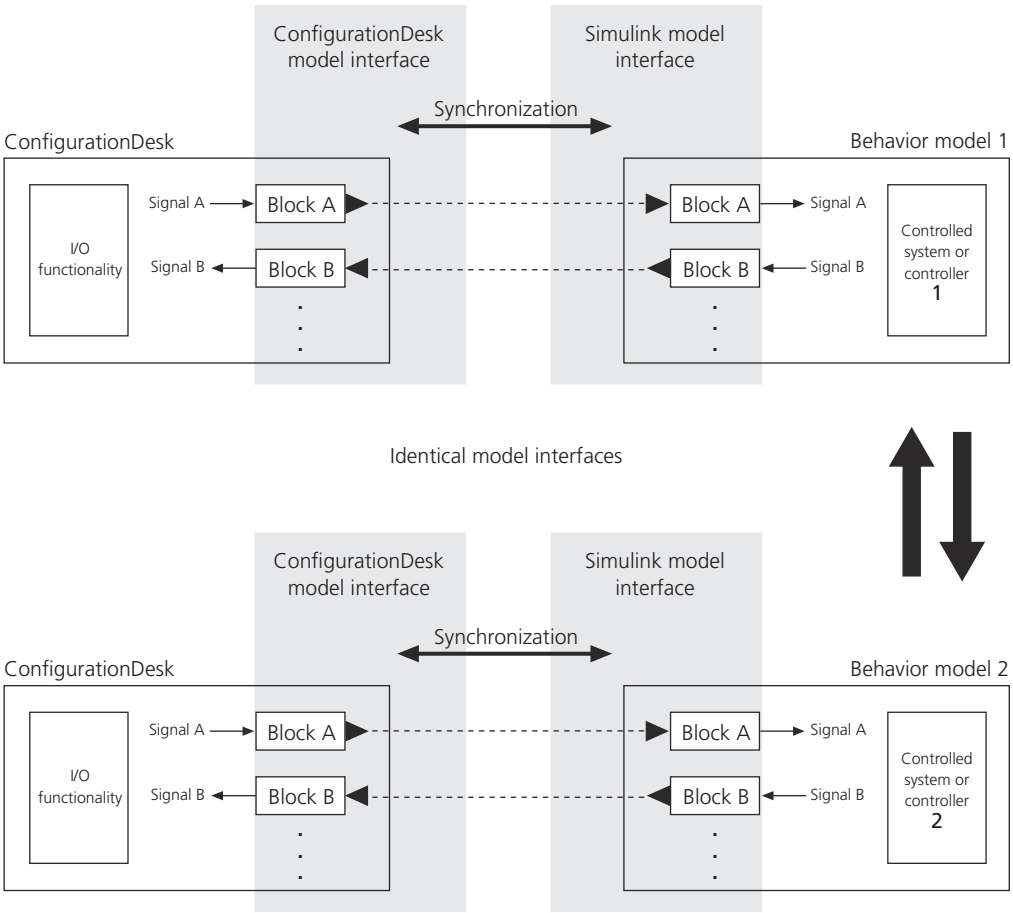
To equalize both model interfaces (ConfigurationDesk model interface and Simulink model interface), you have to synchronize them. This is also necessary after you have modified the I/O functionality in ConfigurationDesk or the behavior model, if the changes affect the model interfaces.

The following data is synchronized:

- Structure and elements of the model interfaces
- Properties of model port blocks and model ports

Using different controlled systems or controllers with the same model interface

To use different controlled systems or controllers with the same I/O functionality in ConfigurationDesk, you can create different behavior models with an identical model interface.



To support this use case, model port blocks (and their model ports) in the behavior model which are already a part of the model topology in ConfigurationDesk are given a unique identity by ConfigurationDesk. This allows you to move or rename the blocks in the behavior model without losing the connection to the corresponding model port block in ConfigurationDesk. The same applies, when you copy a model port block with its identity from one behavior model to another.

Note

Note that in a multimodel application, all model port blocks must have a unique identity. Thus, you cannot use a model port block with the same identity several times, for example, in several behavior models which are linked to the same ConfigurationDesk application.

Connections between ConfigurationDesk and Simulink behavior models

ConfigurationDesk as well as Simulink provide features for you to simpilfy the work with Simulink behavior models.

Remote access from Simulink model to ConfigurationDesk Useful ConfigurationDesk features are accessible directly in the Simulink behavior model, for example, creating a ConfigurationDesk project, analyzing the Simulink model in ConfigurationDesk or starting a ConfigurationDesk build process. In addition you can switch from a selected element in Simulink directly to its representative in ConfigurationDesk.

Synchronizing the model interfaces If there are any changes in the ConfigurationDesk model interface or in the Simulink model interface the model interfaces must be synchronized. There are several commands to simplify synchronization, for example, Propagate to Simulink Model or Analyze Simulink Model (Model Interface Only).

Remote start of MATLAB If you execute an operation in ConfigurationDesk that requires access to MATLAB, ConfigurationDesk starts MATLAB remotely, if it is not already running. Afterwards MATLAB opens the behavior model that is linked to the ConfigurationDesk application. However, the MATLAB session that is started from ConfigurationDesk runs independently of the ConfigurationDesk process, i.e., MATLAB is not closed when ConfigurationDesk is closed.

Working with container files as behavior model

ConfigurationDesk lets you work with different code container files containing a behavior model. You can integrate them into your executable application and execute them on your real-time hardware. The representation of container files is part of the real-time model and the interface to the I/O functionality in ConfigurationDesk is realized via model port blocks just as with other behavior models. The following container file types are supported:

- Simulink implementation container files (SIC files)

  Simulink implementation containers are code containers based on a Simulink behavior model. With a Simulink implementation container file, you can separate the model code generation from the build process.
- Bus simulation container files (BSC files)

  A bus simulation container lets you implement the bus communication of a real-time application. A bus simulation container file is configured and created with the Bus Manager.
- Functional Mock-up Units (FMU files)

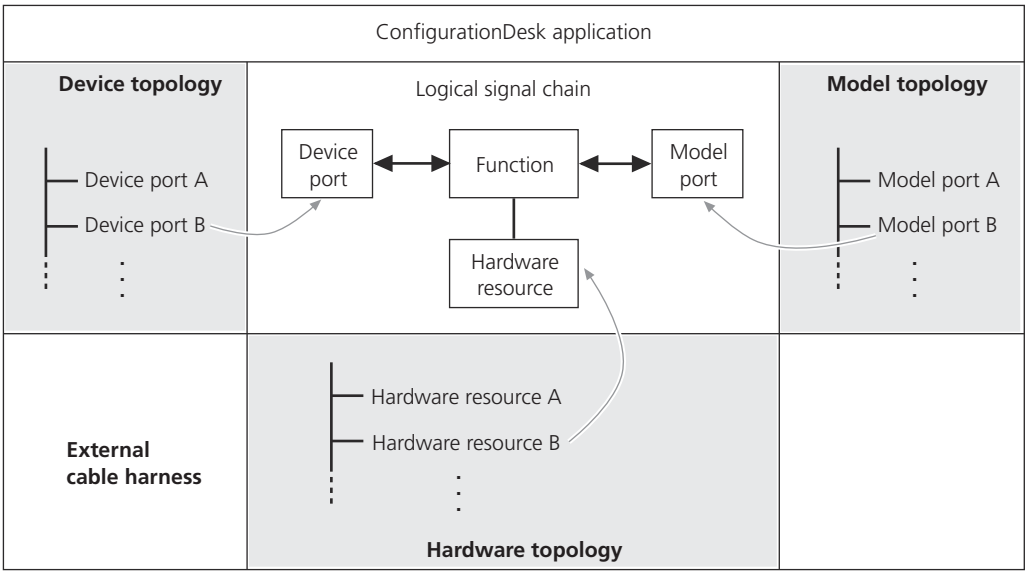
  FMUs are based on the Functional Mock-up Interface standard (FMI standard). The FMI standard defines an interface standard for model exchange and co-simulation. It is supported by different tools, for example, Dymola, MapleSim,

and SimulationX. An FMU implements a model using the interfaces defined by the FMI standard.

Application Components and Their Changeability

Overview of components

An application is the basis for carrying out tasks in ConfigurationDesk. A ConfigurationDesk application can contain the following exchangeable components (written in bold letters):



Definition of topologies

A topology 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.

Description of components

Device topology Represents the interface of your external devices in ConfigurationDesk, such as your ECU to be tested. The topology includes details of the available device pins and their characteristics.

The device topology can be created in ConfigurationDesk or imported from a Microsoft Excel™ file format.

Model topology Contains information on the interface to the behavior model, such as the specified model port blocks and their ports. In a multimodel application the model topology can contain interface information on several behavior models.

The model topology can be created in ConfigurationDesk or established by analyzing the model interfaces of a behavior model which is added to the ConfigurationDesk application.

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 channel types and slot numbers.

The hardware topology can be created in ConfigurationDesk or scanned automatically from the registered hardware.

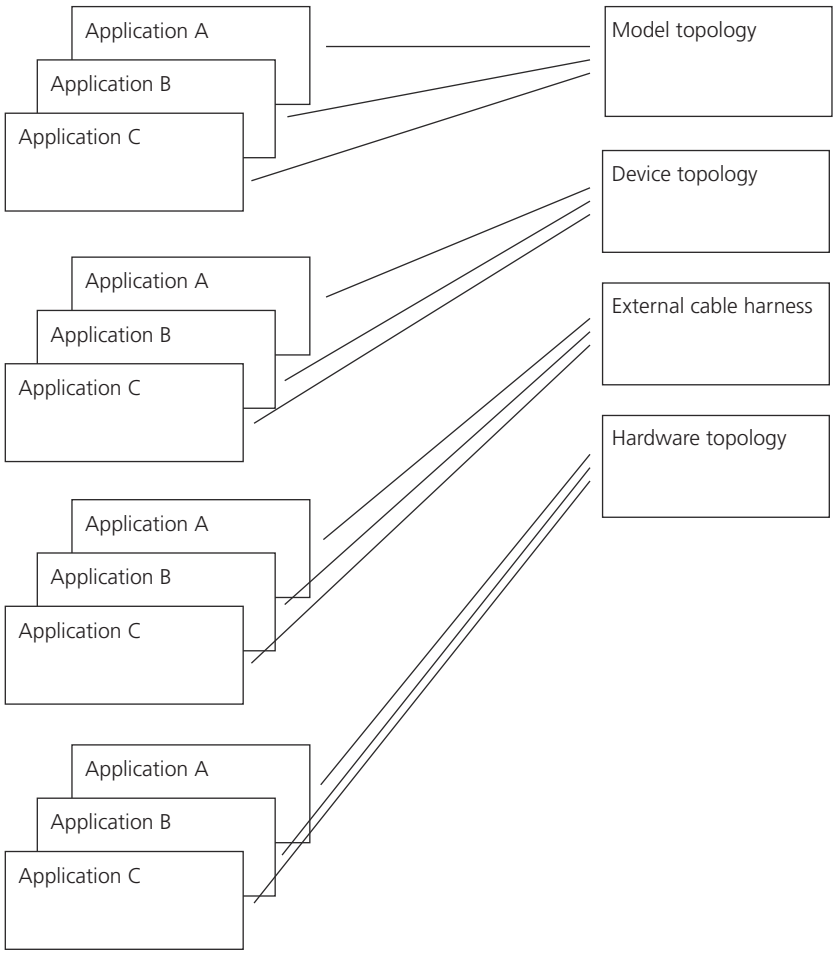
External cable harness Contains the wiring information for the external cable harness. The external cable harness is the connection between the I/O connectors of the real-time hardware and the devices, such as the ECUs to be tested.

The wiring information can be calculated by ConfigurationDesk or imported from a specific file format (ECHX file) so that you can use an existing cable harness.

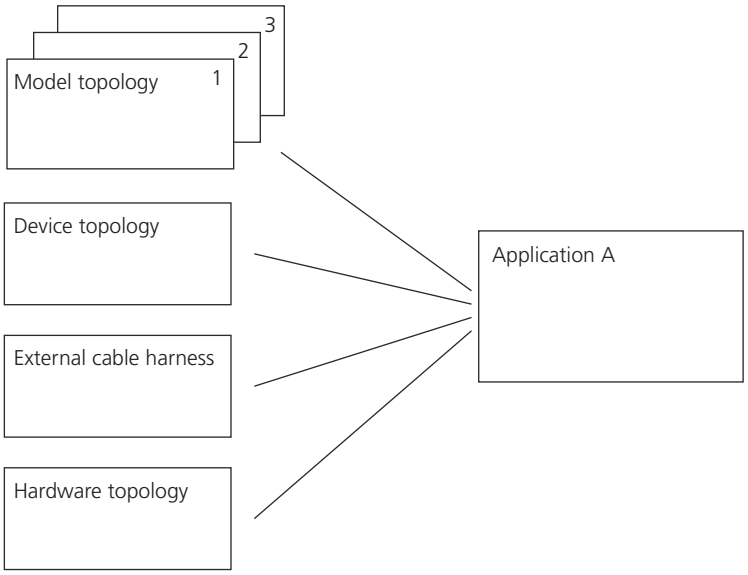
Interchangeability of components

Each component can be contained only once in an application. Each component type is stored in a specific file type and can be used separately in different ConfigurationDesk applications. ConfigurationDesk allows you to export, replace, add and remove each of them.

As shown below you can use each component in several applications. For example, you can use one ECU (device topology) in different applications.



As shown below you can use different components in one application. For example, you can use one application for different behavior models (represented by different model topologies).



Note that each component can be contained only once in an application at the same time.

Creating a Logical Signal Chain

Objective

In ConfigurationDesk, you implement the I/O functionality of your real-time application via a *logical* signal chain using graphical elements.

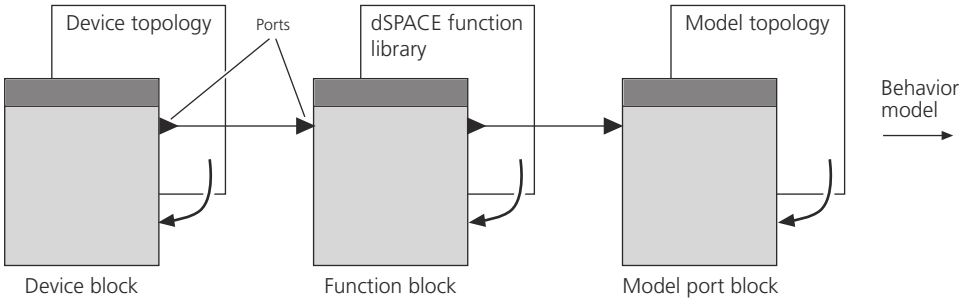
Logical signal chain

The logical signal chain describes the logical path of a signal between an external device and the behavior model. This lets you concentrate on the functionality in your application and not on the physical wirings and their requirements.

After you have assigned hardware (real-time hardware and external devices) to the logical signal chain, ConfigurationDesk also provides the physical path of the signal. For details, refer to Providing the Physical Signal Chain on page 19.

Main elements of the logical signal chain

The main elements of the logical signal chain are represented by different graphical blocks. Every block has ports to provide the mapping to neighboring blocks.



Device block A device block is a subset of the device topology. This subset is used in the signal chain. It represents the interface of your external device(s) and contains details on the input and output signals of the device, etc.

Function block Function block types are the basis for implementing I/O functionality. These are elements of the dSPACE function library. Each function block type has unique features. Instances of a function block type are called function blocks. You can instantiate as many function blocks as you want from a specific function block type.

Model port block Each model port block is an element of the model topology of the active application. Model port blocks in ConfigurationDesk are the interface to the corresponding model port blocks in the behavior model. They provide the data flow between the function blocks and the behavior model.

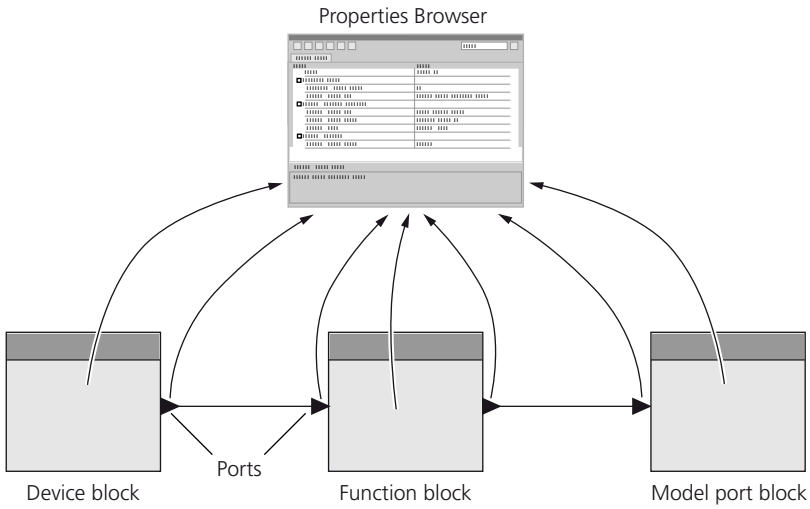
In a multimodel application, model port blocks also can be used to provide the data flow between two behavior models. This is called model communication.

Accessing properties of the signal chain elements

Every block and its components (for example, the ports) have properties. Some of them are user-configurable, others are only for information purposes. You can access the properties:

- Via Properties Browser

  The Properties Browser displays the properties of the selected element of the signal chain as shown below. If a property is configurable, you can change the setting there.



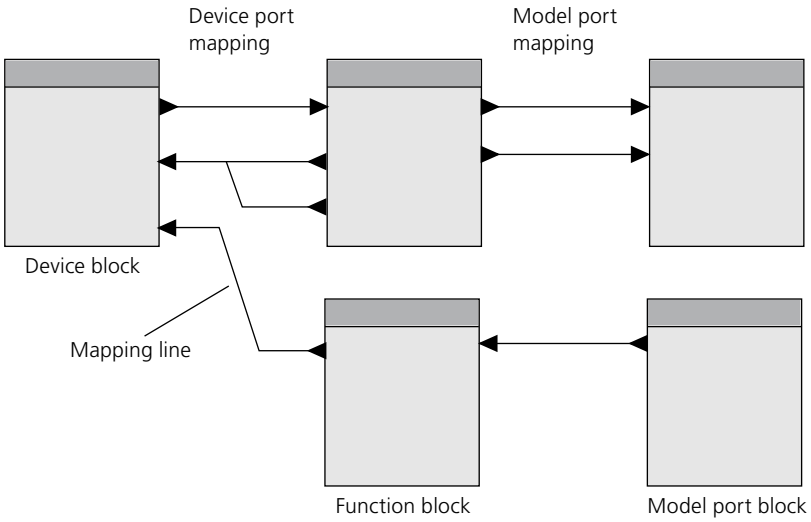
- Via tables

  If the signal chain consists of many blocks, displaying properties and property settings via **Properties Browser** can be time-consuming.

  As an alternative tables provide an overview of the property settings of signal chain elements used in your application. If a property is configurable, you can change the setting directly in the table.

Mapping ports

Mapping ports means drawing a mapping line between two ports. ConfigurationDesk allows only mapping lines which agree with specific rules. Multiple mapping (mapping one port to several ports) is also possible.



Creating the signal chain with the Model-Function Mapping Browser

The **Model-Function Mapping Browser** displays the structure of the behavior models including their subsystems. You can create signal chains by dragging and dropping hardware channels or function blocks to a Simulink behavior model or its subsystems. If you do this, ConfigurationDesk automatically creates model port blocks with suitable configurations for the connection to Simulink behavior models.

Conflicts caused by clashing configuration settings

When you create a signal chain and configure the elements in it conflicts might occur. The 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 allowed at first. However, before you build a real-time application, you have to resolve the conflicts to get proper build results. ConfigurationDesk provides the **Conflicts Viewer**, which displays all current conflicts and helps you to resolve them.

Display of states

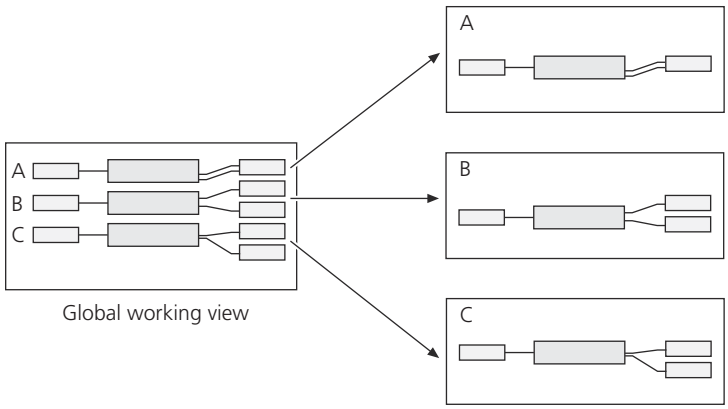
ConfigurationDesk determines the state of each element in the signal chain after every configuration change. States are displayed graphically and in tooltips. You can move the mouse cursor over an element to see its tooltip containing error and warning messages, etc.

Different views of the signal chain

A view of your logical signal chain in ConfigurationDesk is called a *working view*. ConfigurationDesk also provides a global working view, which contains all the elements of your application.

If you create a complex signal chain with a lot of elements, you can tailor your view to keep track of your work. You can define your own working views and add only specific signal chain elements to each of them. For example, one working view can contain all the elements which belong to a specific part of your vehicle, such as the brake system.

The illustration below shows an example where the elements of the global working view are also displayed in three user-defined working views.



Providing the Physical Signal Chain

Physical signal chain

The physical signal chain describes the electrical wiring of external devices (ECU and loads) to the I/O boards of the real-time hardware. It includes the external cable harness, the pinouts of the connectors and the internal cable harness.

External cable harness

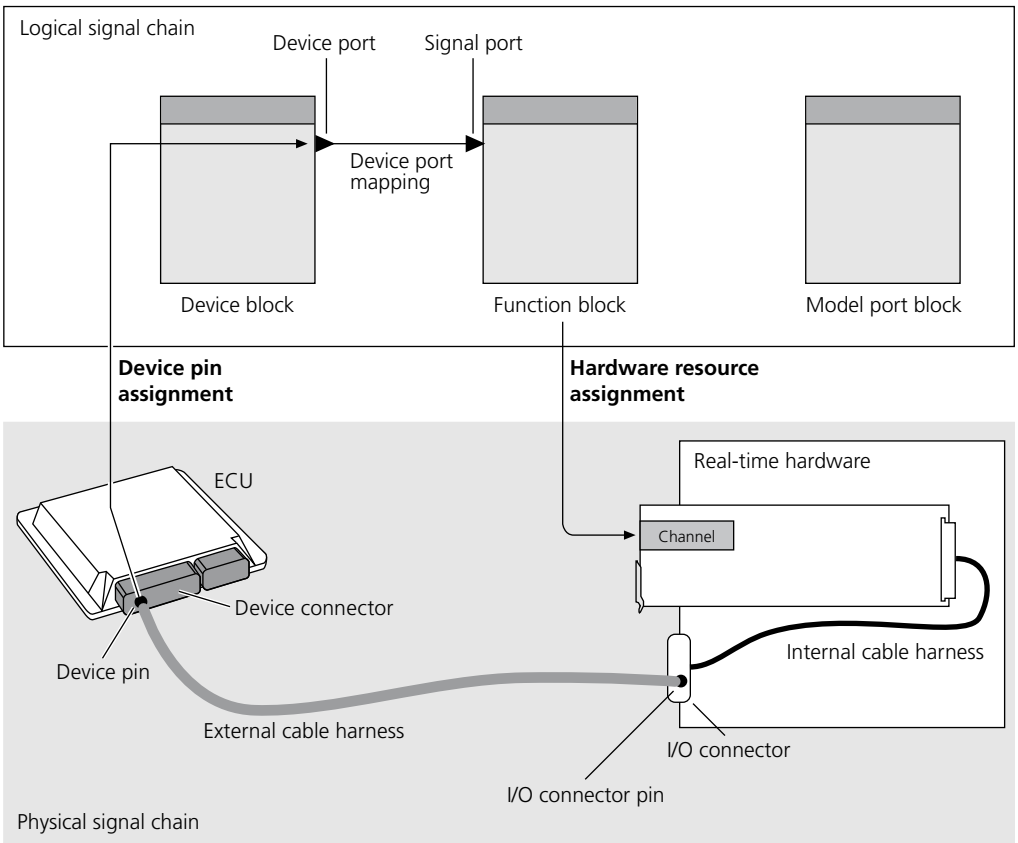
The external cable harness is the connection between the I/O connectors of the real-time hardware and the devices, such as the ECUs to be tested.

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 to have to build a new one.

Assigning hardware to the logical signal chain

ConfigurationDesk is only able to determine the physical signal chain if you have "assigned" the real-time hardware and the external devices to the logical signal chain, as shown below:



As shown above, model port blocks (= model interface) are not required and thus ignored in determining the physical signal chain.

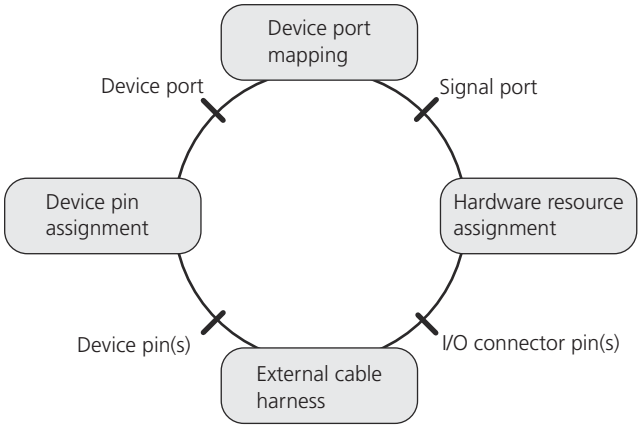
Device pin assignment Device pin assignment means assigning device pins to device ports. You have to do this manually.

Hardware resource assignment Each function block requires at least one hardware resource (channel) to perform the I/O functionality. With ConfigurationDesk and the dSPACE hardware architecture, 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.

You can choose between automatic assignment (done by ConfigurationDesk) and manual assignment (done by the user). To perform manual assignment, you must select a hardware resource from a list of suitable resources which are determined by ConfigurationDesk.

Dependencies in the signal chain

The following graphic illustrates the connections (in shaded frames) which are affected by determining the physical signal chain. You can influence these connections.



The circle illustrates some basic principles:

- If you change one of the connections, for example, the device port mapping, the path of the signal (in the circle) is disturbed and becomes conflicted. You have to change another connection in the circle to resolve the conflict.
- If ConfigurationDesk "knows" three connections in the circle, it supports you when you provide the fourth connection. For example, ConfigurationDesk checks the connection for conflicts and displays conflicted states.

Note

There is one exception: The device pin assignment must always be provided manually and checked by the user. It cannot be created and checked by ConfigurationDesk.

The following examples are based on the assumption, that you have finished the device pin assignment:

- When you have finished the device port mapping and the hardware resource assignment, ConfigurationDesk can calculate the wiring information for the external cable harness.
- If you use an existing cable harness in your ConfigurationDesk application, and you have finished the device port mapping, ConfigurationDesk only provides suitable channels for the hardware resource assignment, i.e., ones which match the wiring of the external cable harness.

Modeling Executable Applications and Tasks

Objective

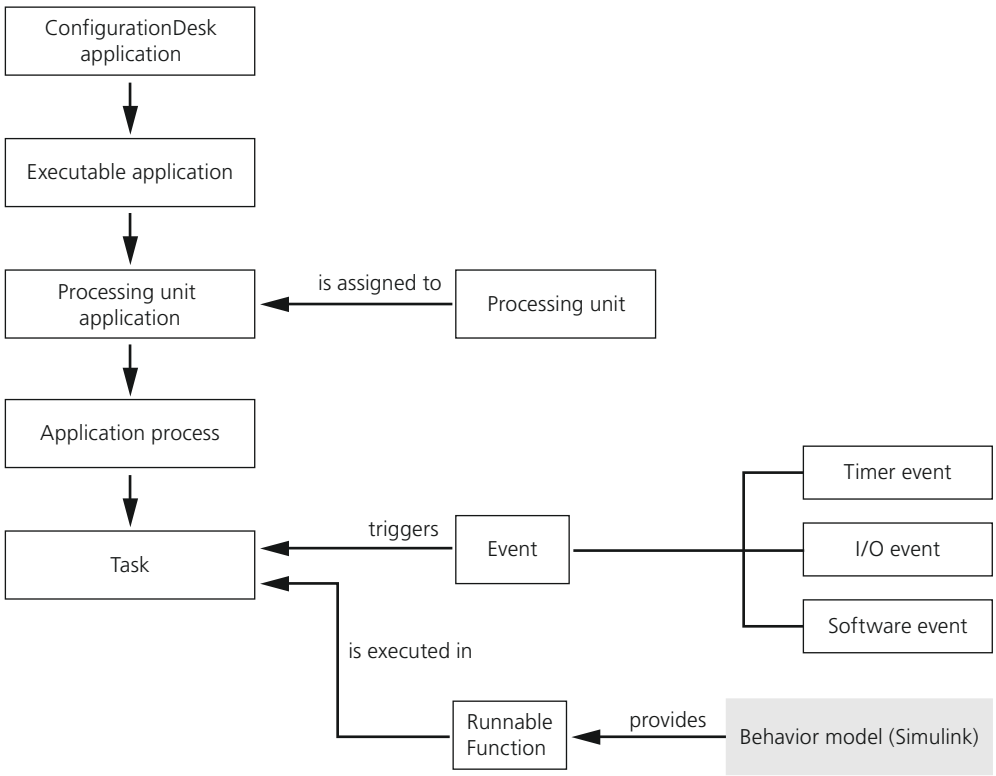
ConfigurationDesk lets you model executable applications (real-time applications) and the tasks used in them very flexibly to match your requirements.

Composition of an executable application

Each executable application consists of at least one processing unit application as its basic structural element. It contains at least one application process and at

least one task. You can define the elements of a processing unit application, the application process, and the priorities of the tasks in the process.

The following illustration shows the elements of an executable application and their relationships.



Task An application process must have one or more tasks. You can assign a component which contains a predefined task (including its runnable functions and events) to an application process. Additionally, ConfigurationDesk lets you create new periodic or asynchronous tasks to execute runnable functions. The execution of tasks is triggered by timer events, I/O events, or software events.

Runnable function 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. A Simulink behavior model provides runnable functions for the following elements:

- For the Simulink base rate task (no predefined task)
- For each **Runnable Function** block in the Simulink model. This block exports a function-call subsystem as a runnable function. Thus, a runnable function groups together all the behavior model parts that must be called to compute results in a specific task. A **Runnable Function** block can provide a runnable function without predefined task or a predefined task including a runnable function.

You can access runnable functions after model analysis. Then you can assign the function to a task (periodic or asynchronous).

Timer event You can create timer events in ConfigurationDesk, configure them (for example, their time periods) and assign them to a task to trigger it periodically.

I/O event Some function blocks provide event generation. When they are added to the signal chain, they can generate I/O events. You can use I/O events in an executable application by assigning them to a task to trigger it.

Software event Software events are specified in the behavior model. They are available in ConfigurationDesk after model analysis.

Concepts of Building and Downloading a Real-Time Application

Objective

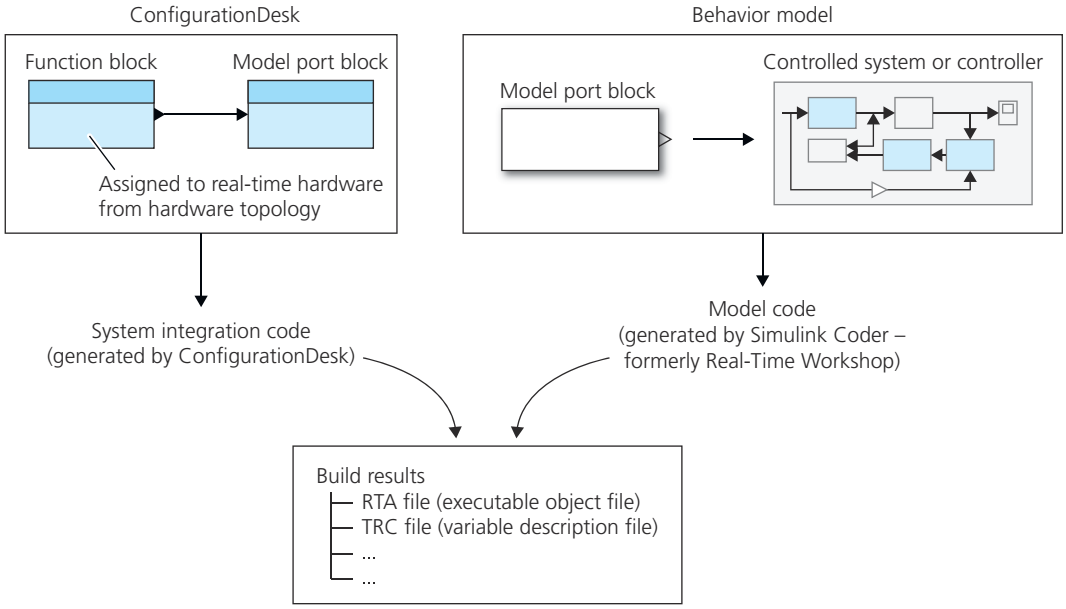
The main concepts for building and downloading a real-time application are described, using a Simulink model as a behavior model. For specifics in other use scenarios such as integrating Functional Mock-up Units (FMUs), refer to the relevant documentation.

Controlling the build process

You can start the build process in ConfigurationDesk or in the Simulink behavior model to generate the real-time application. The build process is completely controlled in ConfigurationDesk. This also includes the make process (for example, compiling and linking the sources, generating the object file and the variable description file).

Code generation

The illustration below shows the code generation for the I/O functionality in ConfigurationDesk and the behavior model:



- The C or C++ code for the behavior model is called model code. It is generated by the Simulink® Coder™.

- The C++ code for the I/O functionality, including the port mapping, the hardware resource assignment, and the task handling is called system integration code. It is generated by ConfigurationDesk.

The build results are copied to the **Build Results** folder of the ConfigurationDesk application.

The interface to your external devices (represented by the device topology and device blocks) is not required and thus ignored in code generation.

Build process without model code generation

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 model has not been changed since the last build process. Note that a build process without model code generation can only be performed successfully if model code of a prior build process exists.

Build options

All the required build options can be manually set via:

- **Build Configuration** table in ConfigurationDesk
- Simulink Coder

However, some Simulink Coder settings of the behavior model are overwritten by ConfigurationDesk if they are not suitable for model code generation.

The MATLAB Command Window displays all the build option changes which were automatically made by ConfigurationDesk with their old and new settings.

Handling errors during the build process

The build process is aborted only for serious errors. Most problems which occur during the build process are categorized as warnings. They are displayed in the **Build Log Viewer** in ConfigurationDesk.

While implementing the signal chain, ConfigurationDesk detects all conflicts which will affect the build process and code generation. The **Conflicts Viewer** displays these conflicts and provides information on the effects, for example, if they abort a build, a warning is generated during build, or no code or default code is generated.

Error messages which are generated by MATLAB, Simulink and Simulink Coder are only displayed in the MATLAB Command Window and not in ConfigurationDesk.

Simulation of I/O access

If there is a function block without hardware assigned to it, the build process is not aborted. In this case the I/O access is simulated, i.e., initial values configured at the function block are input to the behavior model.

This concept allows you to implement and build an executable real-time application without having access to a complete real-time hardware system.

Compatibility check during download

During the download, ConfigurationDesk checks if the hardware resources used during the build process match the platform to which the real-time application

is downloaded. The **Message Viewer** displays all inconsistencies, and in several cases ConfigurationDesk generates a message box which allows you to abort the download.

If there are inconsistencies, ConfigurationDesk tries to continue the download as follows:

- If at least one channel used by the application is missing on the platform, the function assigned to the channel is simulated by using user-configured initial values.
- If the signal available at a hardware resource is different to the one assigned to it, you can either accept or reject this reassignment. If you reject it, the corresponding functionality is missing in your real-time application.

In both cases you can continue the download process if you accept the conflicts.

This concept allows you to execute a real-time application although the platform does not exactly match the hardware topology of the application used during the build process. For example, you can perform tests without starting a new build process although several I/O boards are temporarily missing.

Building a multicore real-time application

You can build a multicore real-time application and download it to the dSPACE real-time hardware, where different application processes (each containing one behavior model) are executed in parallel on single processor cores. The maximum number of application processes is (number of processor cores) - 1 , since one core is reserved for the execution of system services.

You can configure several build options for each real-time model separately. However, ConfigurationDesk automatically assigns application processes to cores for execution. The maximum number of application processes running on a SCALEXIO Processing Unit is *number of processor cores - 1*, since one core is reserved for the execution of system services. For a DS6001 Processor Board, the maximum number of application processes is *number of processor cores*.

Building a multi-processing unit (multi-PU) application

You can build a multi-PU application and download it to the dSPACE real-time hardware, where different processing unit applications (each containing at least one application process with one behavior model) are executed in parallel on several SCALEXIO Processing Units. The processing units are connected via IOCNET and can be accessed from the same host PC.

You can configure several build options for each real-time model separately. Via processing unit assignment you have to assign a processing unit application to a specific processing unit for execution.

Overview of the User Interface of ConfigurationDesk

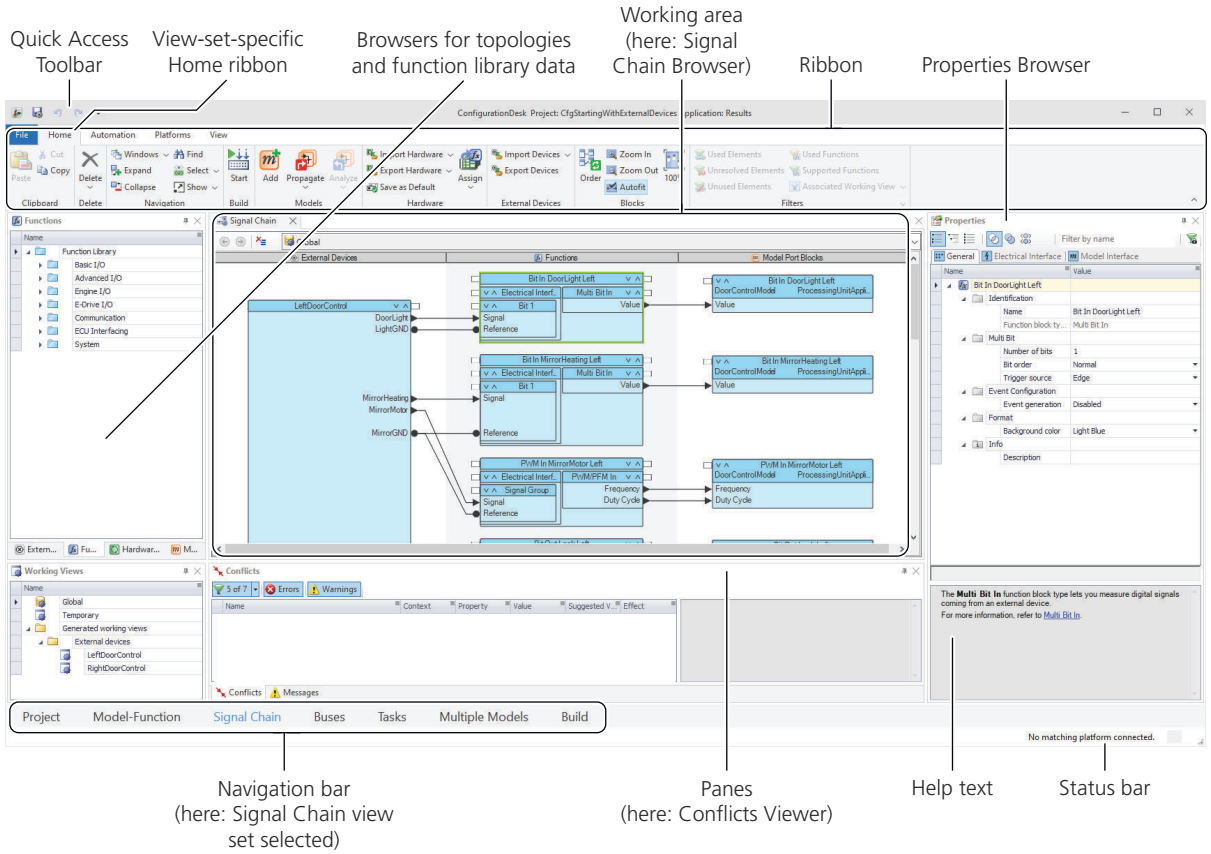
Switching between view sets for specific purposes

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 (ConfigurationDesk Real-Time Implementation Guide 🕮).

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 (ConfigurationDesk Real-Time Implementation Guide 🕮).

Available view sets and their purposes

View Set	Purpose	Panes[1)]	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).	▪ Project Manager ▪ *If no project is currently open:* Start Page	Managing ConfigurationDesk Projects and Applications (ConfigurationDesk Real-Time Implementation Guide 🕮)
Model-Function	To implement the connection between a behavior model and ConfigurationDesk's I/O functionality.	▪ Model-Function Mapping Browser ▪ Function Browser ▪ Hardware Resource Browser ▪ Properties Browser ▪ Conflicts Viewer	User-Friendly Connection of ConfigurationDesk and Simulink Models (ConfigurationDesk Real-Time Implementation Guide 🕮)
Signal Chain	To implement the complete logical signal chain between an external device and a behavior model.	▪ Signal Chain Browser ▪ Working View Manager ▪ External Device Browser ▪ Function Browser ▪ Hardware Resource Browser ▪ Model Browser ▪ Properties Browser ▪ Conflicts Viewer	Handling the Signal Chain in Working Views (ConfigurationDesk Real-Time Implementation Guide 🕮)
Buses	To implement bus communication based on communication matrices.	▪ Bus Configurations table ▪ Bus Simulation Features table ▪ Bus Inspection Features table ▪ Bus Manipulation Features table ▪ Bus Access Requests table ▪ Bus Configuration Ports table ▪ Buses Browser ▪ Hardware Resource Browser ▪ Model Browser ▪ Properties Browser ▪ Conflicts Viewer ▪ Message Viewer	ConfigurationDesk Bus Manager Implementation Guide 🕮
Tasks	To model and configure the tasks and application processes of your executable application.	▪ Task Configuration table ▪ Properties Browser ▪ Conflicts Viewer	Modeling Executable Applications and Tasks (ConfigurationDesk Real-Time Implementation Guide 🕮)
Multiple Models	To configure the communication between multiple behavior models.	▪ Model Communication Browser ▪ Model Communication Package table ▪ Processing Resource Assignment table	Setting Up Model Communication (ConfigurationDesk Real-Time Implementation Guide 🕮)

View Set	Purpose	Panes[1)]	Further Information
		▪ Working View Manager ▪ Model Browser ▪ Properties Browser ▪ Conflicts Viewer	
Build	▪ To configure the build process and build the real-time application. ▪ To download the real-time application to dSPACE hardware.	▪ Build Configuration table ▪ Build Log Viewer ▪ Properties Browser ▪ Platform Manager ▪ Conflicts Viewer	▪ Building Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 🕮) ▪ Loading and Executing Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 🕮)

1) Default state after installing ConfigurationDesk.

Common panes available in most view sets

Some panes are available and required in most view sets.

Properties Browser Lets you configure the properties of selected elements. Refer to Configuring Elements with the Properties Browser (ConfigurationDesk Real-Time Implementation Guide 🕮).

Conflicts Viewer Displays configuration conflicts in your ConfigurationDesk application. You can resolve most of the conflicts here. Refer to Resolving Conflicts (ConfigurationDesk Real-Time Implementation Guide 🕮).

Additional panes for specific purposes

Some panes are part of few or no view sets but may still be required for specific purposes.

Message Viewer 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 Message Viewer (ConfigurationDesk User Interface Reference 🕮).

Find Results Viewer Displays the results of searches you performed by using the Find command. Refer to How to Find Elements of a ConfigurationDesk Application (ConfigurationDesk Real-Time Implementation Guide 🕮).

Interpreter Handles Python commands and scripts for automating ConfigurationDesk. Refer to ConfigurationDesk Automating Tool Handling 🕮.

Customizing view sets

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 Customizing View Sets (ConfigurationDesk Real-Time Implementation Guide 🕮).

Related topics

Basics

Basics on Ribbons (ConfigurationDesk Real-Time Implementation Guide 🕮)
Customizing View Sets (ConfigurationDesk Real-Time Implementation Guide 🕮)

Typical Workflows for Beginners

Where to go from here

Information in this section

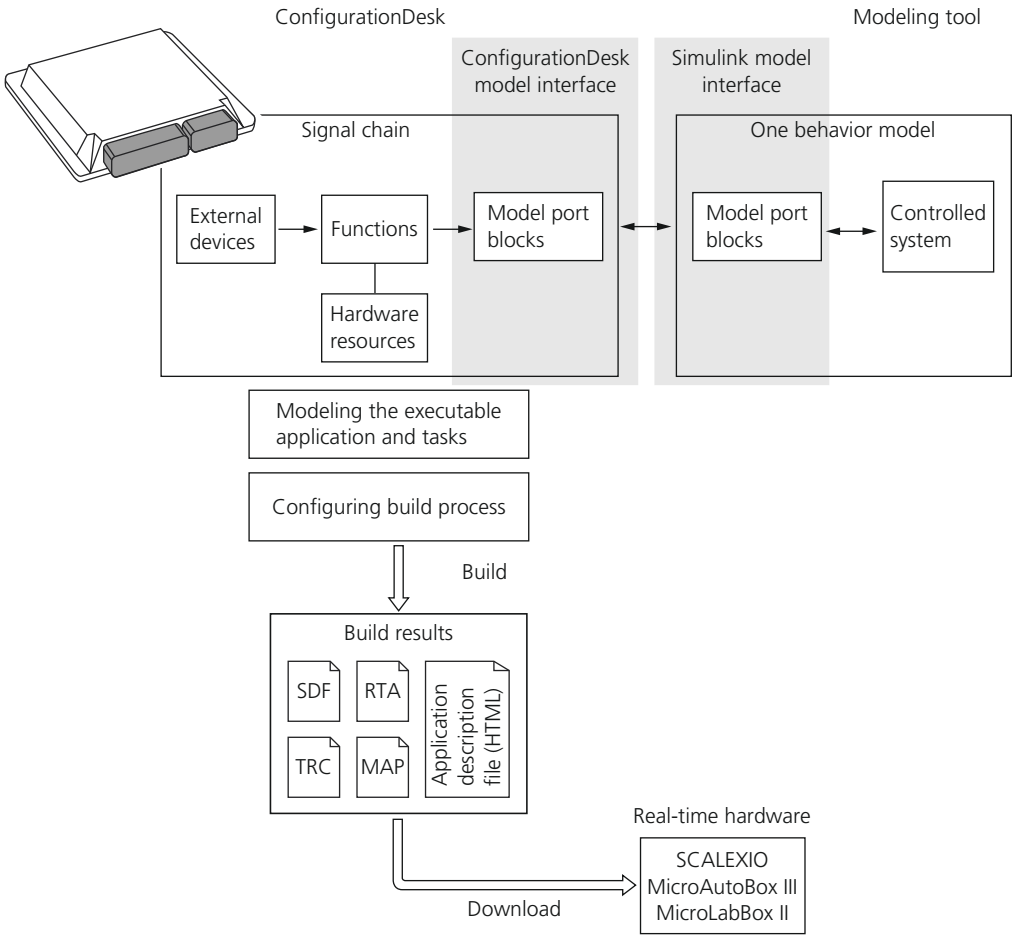
Creating a Real-Time Application: From ECU to Model

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.

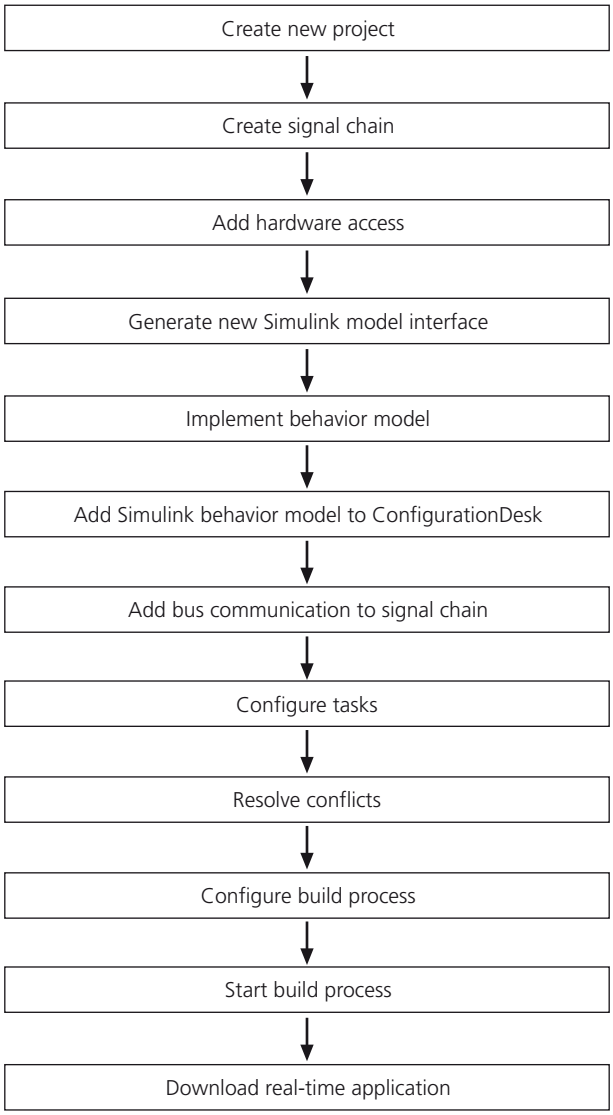
Overview

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



Workflow

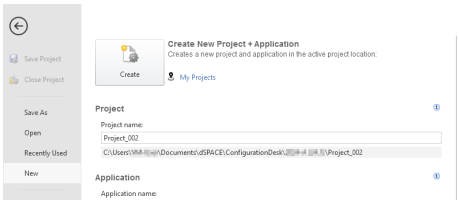
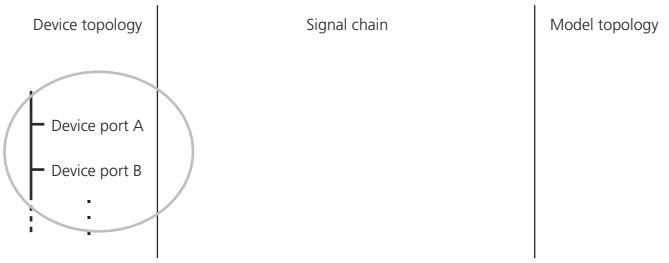
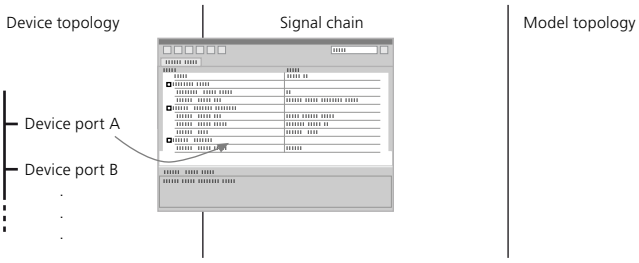
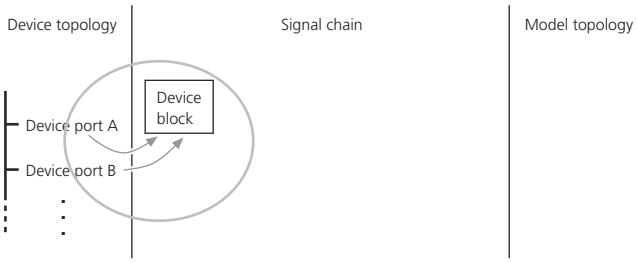
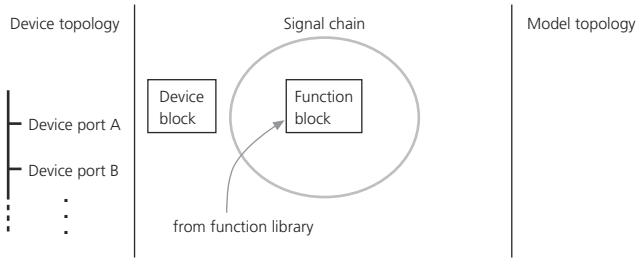
The workflow shows the steps of a standard workflow for implementing a ConfigurationDesk application with one behavior model.

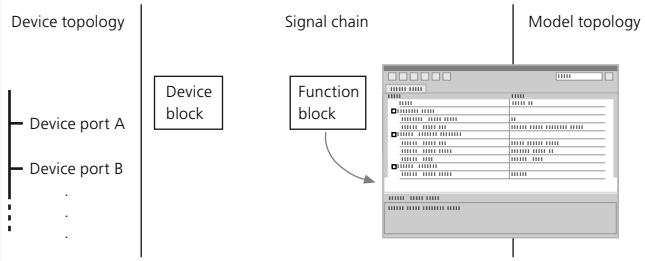
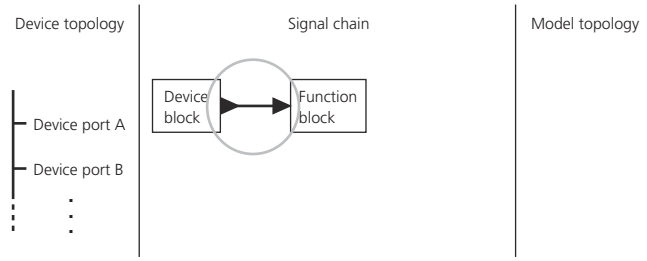
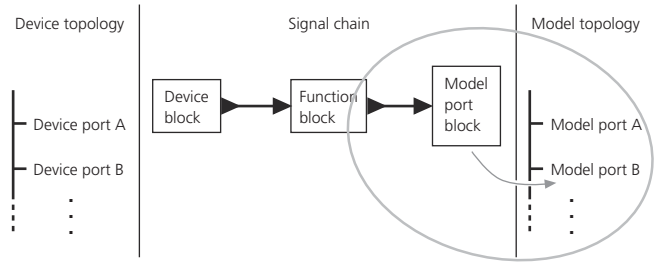


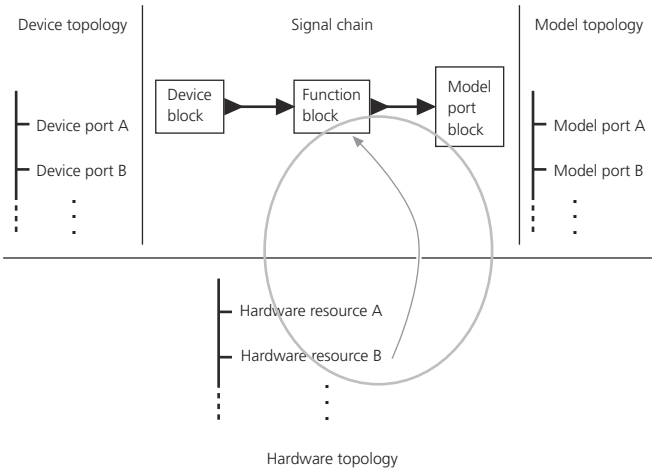
Detailed steps

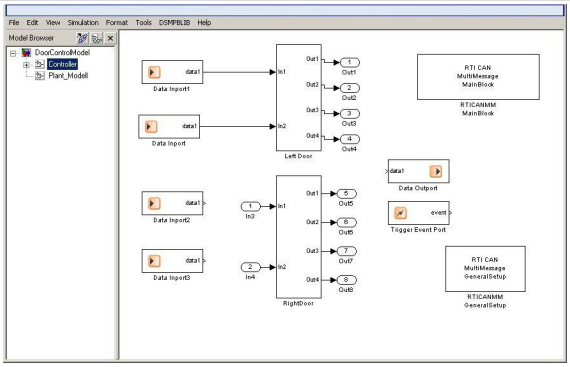
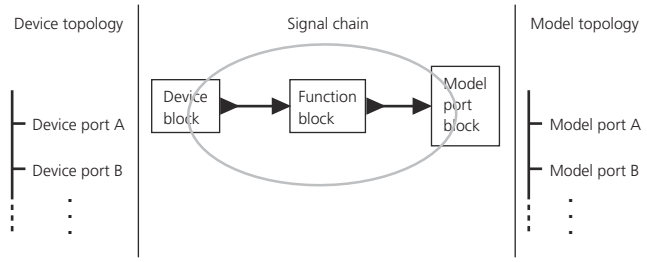
The following table shows detailed workflow steps and indicates the corresponding chapters, which give you detailed information (basic information, instructions etc.).

Step	Work	Refer to ...
1	Create a new project and a ConfigurationDesk application.	Managing ConfigurationDesk Projects and Applications (ConfigurationDesk Real-

Step	Work	Refer to ...
		Time Implementation Guide 🕮)
2	Create a signal chain in ConfigurationDesk.	
	1 Create a device topology. 	Creating and Extending Device Topologies (ConfigurationDesk Real-Time Implementation Guide 🕮)
	2 Configure device ports. 	Configuring External Devices (ConfigurationDesk Real-Time Implementation Guide 🕮)
	3 Add device blocks to the signal chain. 	Adding Device Topology Elements to the Signal Chain (ConfigurationDesk Real-Time Implementation Guide 🕮)
	4 Add function blocks to the signal chain. 	Adding Function Blocks to the Signal Chain (ConfigurationDesk Real-Time Implementation Guide 🕮)

Step	Work	Refer to ...
	5 Configure the function block properties. 	Configuring Function Blocks (ConfigurationDesk Real-Time Implementation Guide 🕮)
	6 Map device blocks to function blocks (= device port mapping). 	Device Port Mapping (ConfigurationDesk Real-Time Implementation Guide 🕮)
	7 Add model port blocks to the signal chain. 	Adding Model Port Blocks to the Signal Chain (ConfigurationDesk Real-Time Implementation Guide 🕮)

Step	Work		Refer to ...
3	Add hardware access.		
	1	Add a hardware topology to the ConfigurationDesk application.	How to Import a Hardware Topology (ConfigurationDesk Real-Time Implementation Guide 🕮)
	2	Assign hardware resources to the function blocks.  	Assigning Hardware Resources to Function Blocks (ConfigurationDesk Real-Time Implementation Guide 🕮)
	3	Calculate wiring information for the cable harness and export it for external usage.	How to (Re)Calculate the External Cable Harness (ConfigurationDesk Real-Time Implementation Guide 🕮)
4	Generate the Simulink model interface and implement the behavior model.		
	1	Generate new Simulink model interface.	How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model (ConfigurationDesk Real-Time Implementation Guide 🕮)
	2	Implement the behavior model: ▪ Model the control algorithm. Use Simulink Blocksets and Toolboxes from MathWorks®. ▪ Specify the model interface of the behavior model. ▪ Configure tasks. ▪ Implement bus communication (CAN, LIN, FlexRay) via specific dSPACE RTI blocksets (if necessary).	Introduction to the Model Interface Package for Simulink (Model Interface Package for Simulink - Modeling Guide 🕮)

Step	Work	Refer to ...
		
	3 Add the behavior model to the ConfigurationDesk application.	How to Import a Model Topology (ConfigurationDesk Real-Time Implementation Guide 🕮)
5	Add bus communication to the signal chain (if necessary).	
	1 Add specific function blocks for bus communication (CAN, LIN) to the signal chain.	Adding Bus and Gigalink Communication to the Signal Chain (ConfigurationDesk Real-Time Implementation Guide 🕮)
	2 Map these function blocks to device blocks and specific model port blocks (= Configuration Port blocks). 	
6	Configure tasks.	Configuring Tasks in ConfigurationDesk (ConfigurationDesk Real-Time Implementation Guide 🕮)
7	Resolve conflicts via the **Conflicts Viewer**.	Resolving Conflicts (ConfigurationDesk Real-Time Implementation Guide 🕮)
8	Configure and start the build process to build the real-time application.	
	1 Configure the build process.	Configuring the Build Process (ConfigurationDesk Real-Time Implementation Guide 🕮)
	2 Start the build process.	Starting the Build Process (ConfigurationDesk

Step	Work	Refer to ...
		Real-Time Implementation Guide 🕮)
9	Download the real-time application to a platform.	Loading and Executing Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 🕮)

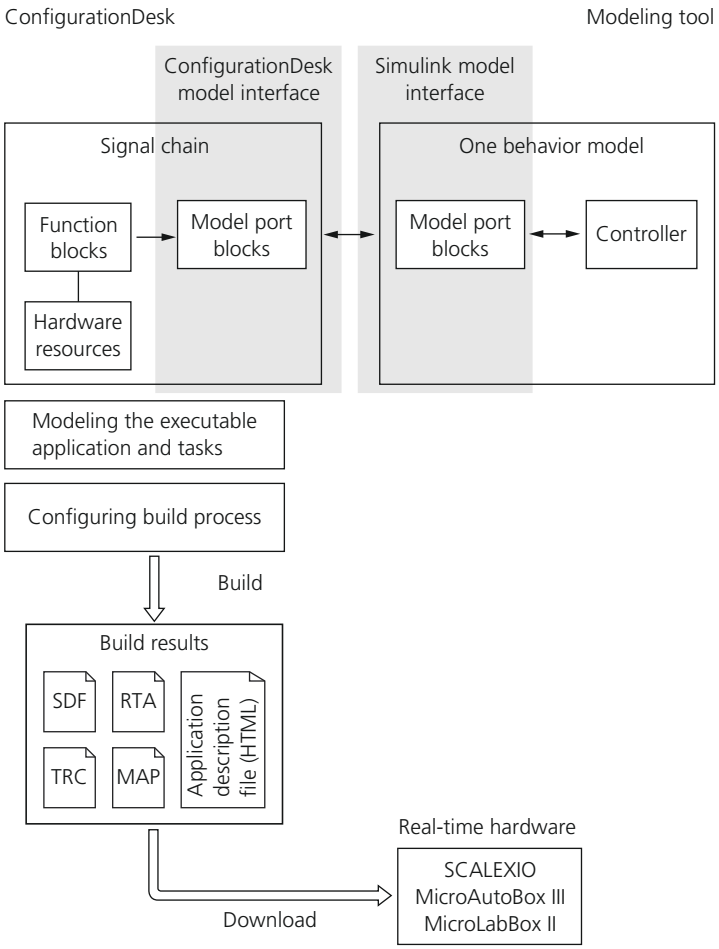
Creating a Real-Time Application: Starting with a Simulink Behavior Model

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.

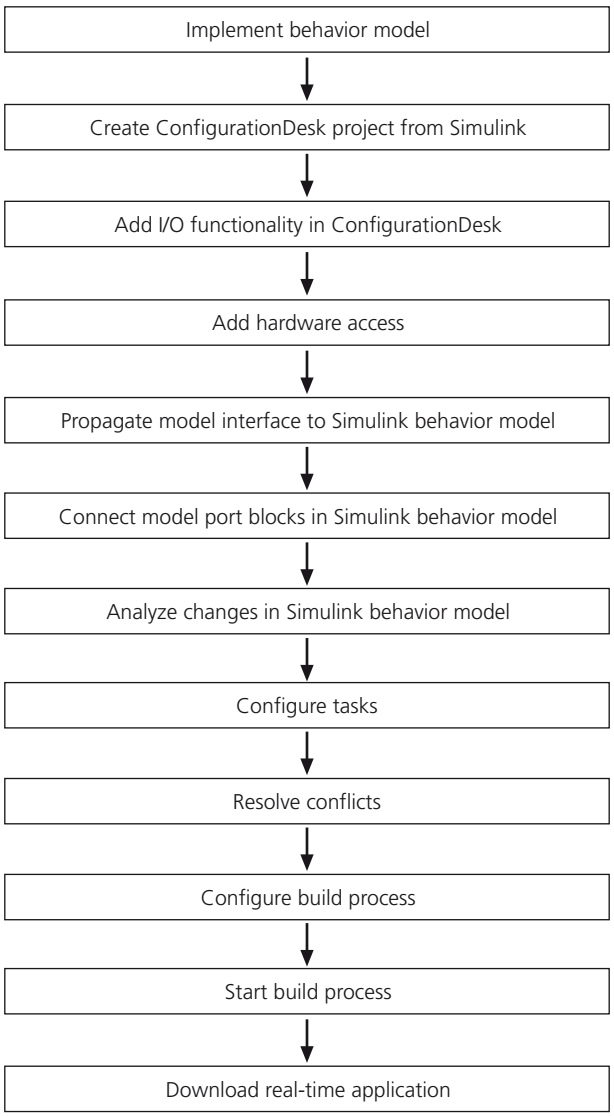
Overview

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



Workflow

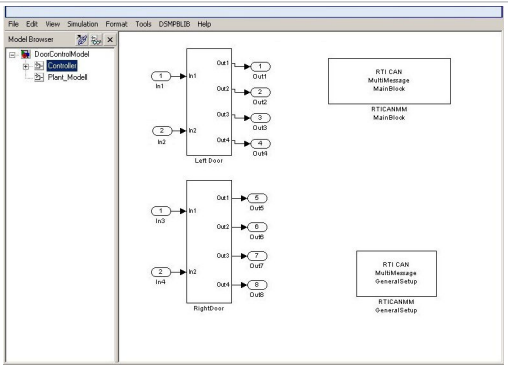
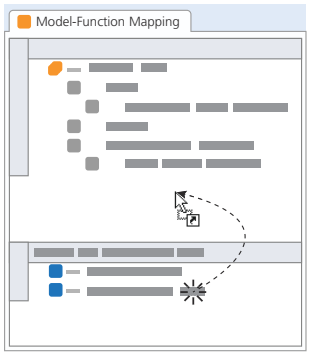
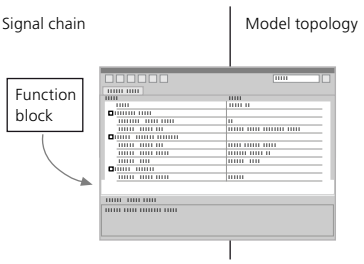
The workflow shows the steps of a standard workflow for implementing a ConfigurationDesk application with one behavior model.

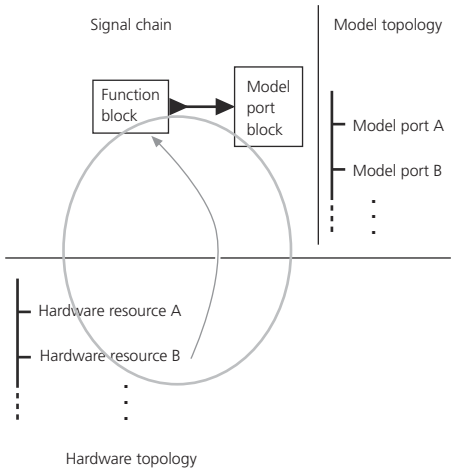
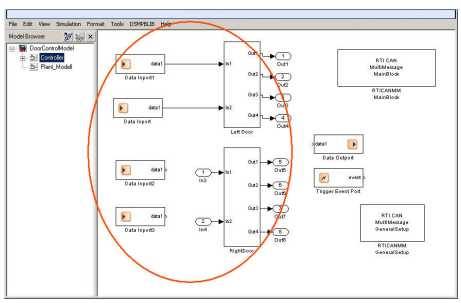


Detailed steps

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: ▪ Model the control algorithm. Use Simulink Blocksets and Toolboxes from MathWorks®. ▪ Implement bus communication (CAN, LIN, FlexRay) via specific dSPACE RTI blocksets (if necessary).	Introduction to the Model Interface Package for Simulink (Model Interface Package for Simulink - Modeling Guide 🕮)

Step	Work		Refer to ...
			
2	Create a new ConfigurationDesk project.		
	1	Create ConfigurationDesk project from the Simulink behavior model.	Remote Access to ConfigurationDesk (Model Interface Package for Simulink - Modeling Guide 🕮)
	2	Add the behavior model to the ConfigurationDesk application.	Managing ConfigurationDesk Projects and Applications (ConfigurationDesk Real-Time Implementation Guide 🕮)
3	Add I/O functionality to the ConfigurationDesk application.		
	1	Create signal chain using the **Model-Function Mapping Browser**. 	Creating Signal Chains via the Model-Function Mapping Browser (ConfigurationDesk Real-Time Implementation Guide 🕮)
	2	Configure the function block properties. 	Configuring Function Blocks (ConfigurationDesk Real-Time Implementation Guide 🕮)

Step	Work		Refer to ...
4	Add hardware access.		
	1	Add a hardware topology to the ConfigurationDesk application.	How to Import a Hardware Topology (ConfigurationDesk Real-Time Implementation Guide 🕮)
	2	Assign hardware resources to the function blocks.  	Assigning Hardware Resources to Function Blocks (ConfigurationDesk Real-Time Implementation Guide 🕮)
5	Propagate changes from ConfigurationDesk to the Simulink behavior model.		Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model (ConfigurationDesk Real-Time Implementation Guide 🕮)
6	Connect new model port blocks in Simulink behavior model. 		Introduction to the Model Interface Package for Simulink (Model Interface Package for Simulink - Modeling Guide 🕮)
7	Analyze changes in Simulink behavior model from ConfigurationDesk.		Analyzing Simulink Behavior Models (ConfigurationDesk Real-Time Implementation Guide 🕮)
8	Configure tasks.		Configuring Tasks in ConfigurationDesk (ConfigurationDesk Real-Time Implementation Guide 🕮)

Step	Work		Refer to ...
9	Resolve conflicts via the **Conflicts Viewer**.		Resolving Conflicts (ConfigurationDesk Real-Time Implementation Guide 🕮)
10	Configure and start the build process to build the real-time application.		
	1	Configure the build process.	Configuring the Build Process (ConfigurationDesk Real-Time Implementation Guide 🕮)
	2	Start the build process.	Starting the Build Process (ConfigurationDesk Real-Time Implementation Guide 🕮)
11	Download the real-time application to a platform.		Loading and Executing Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 🕮)

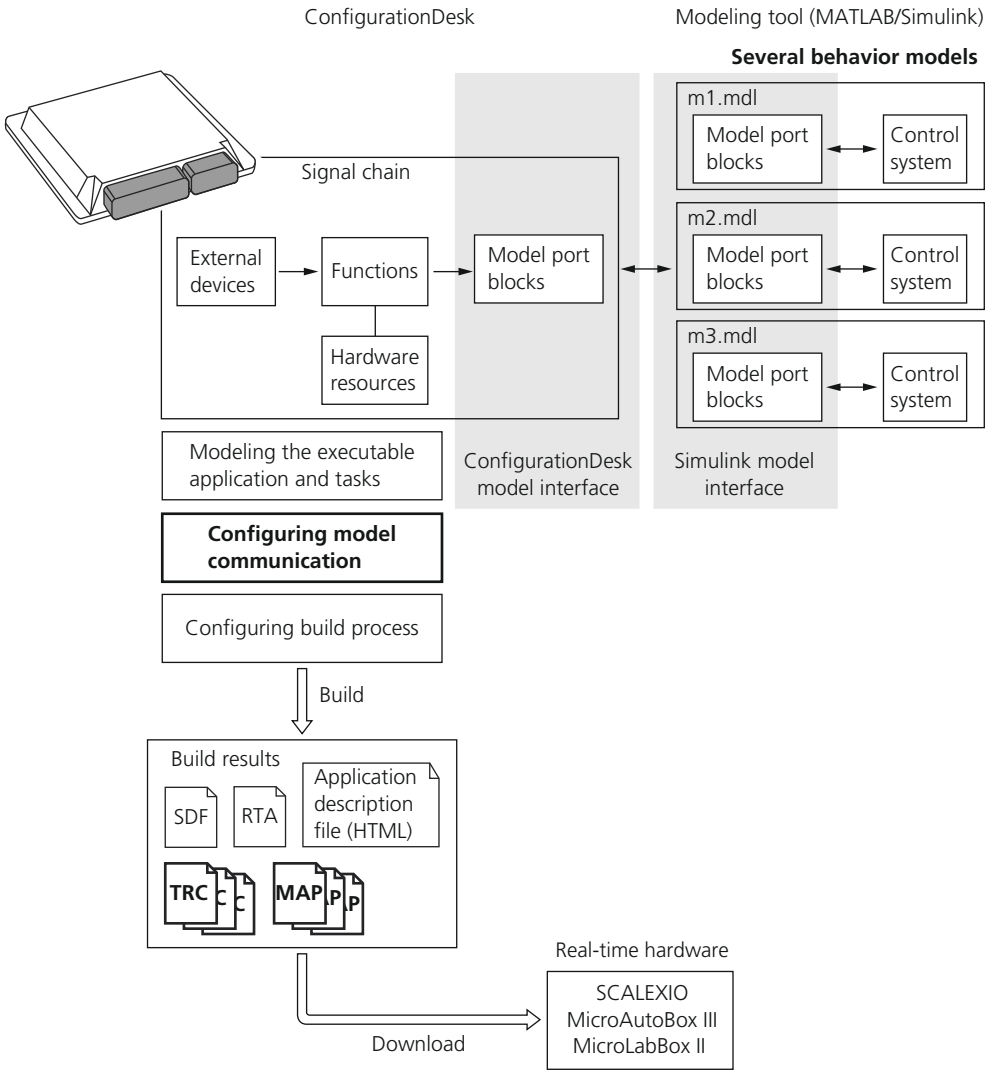
Creating a Multicore Real-Time Application Using Multiple Behavior Models

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.

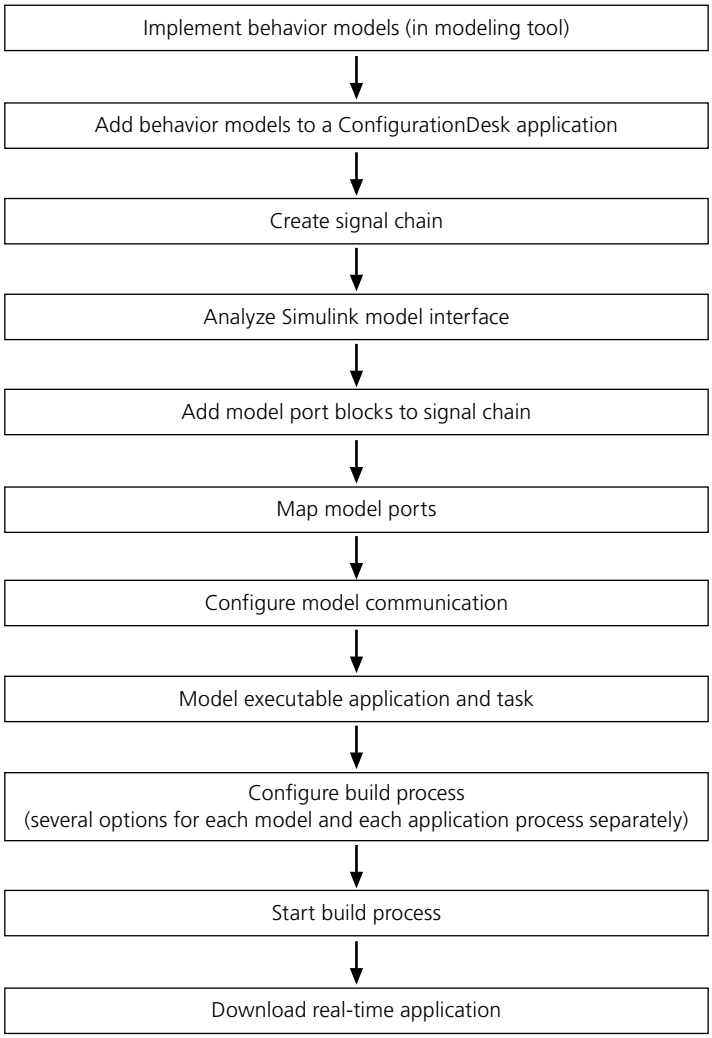
Overview

The following illustration gives you an overview of the use scenario. Parts that are specific to this scenario are in bold letters.



Workflow

The workflow includes the steps that are specific for this use scenario.



Creating a Multicore Real-Time Application Using One Overall Behavior Model

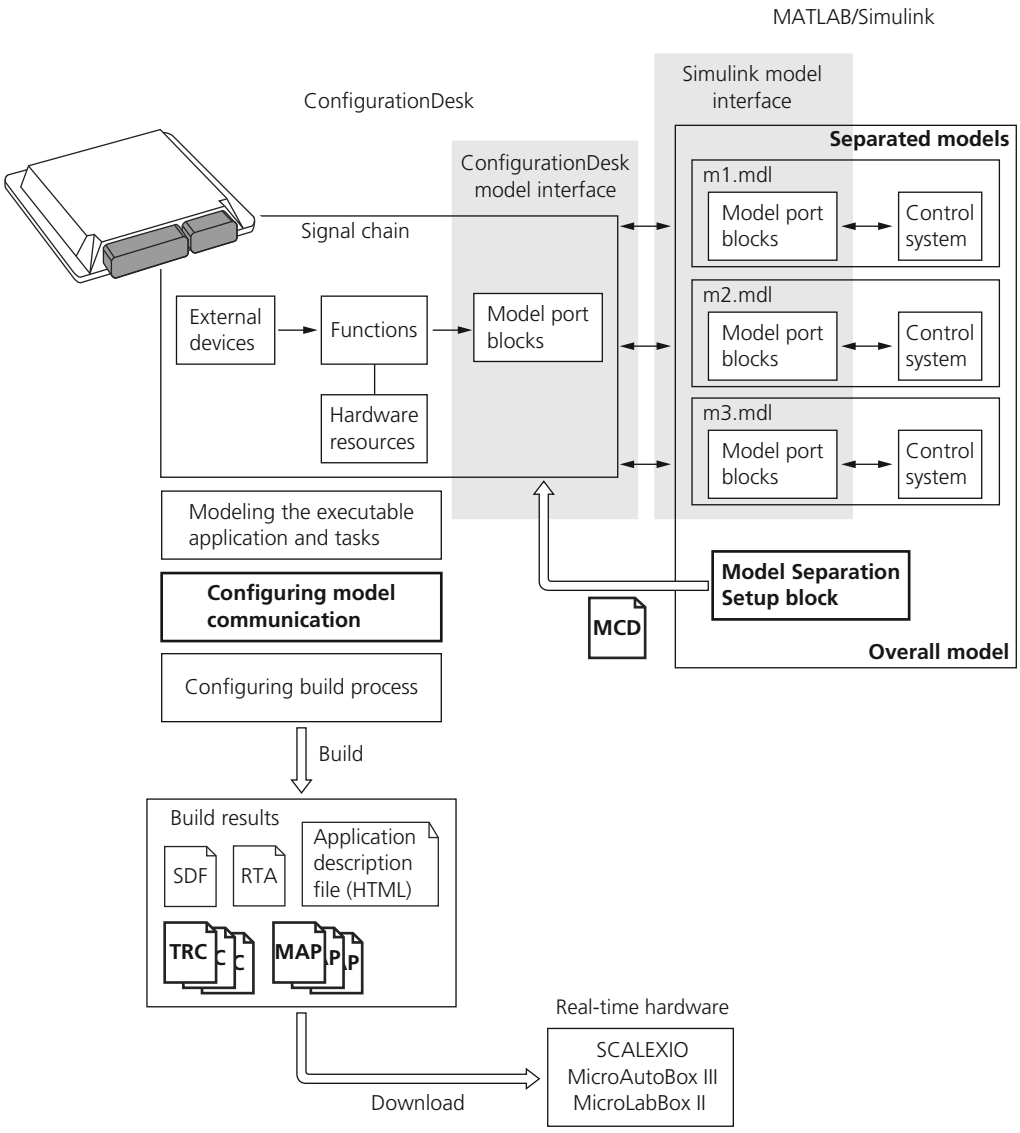
Use scenario

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

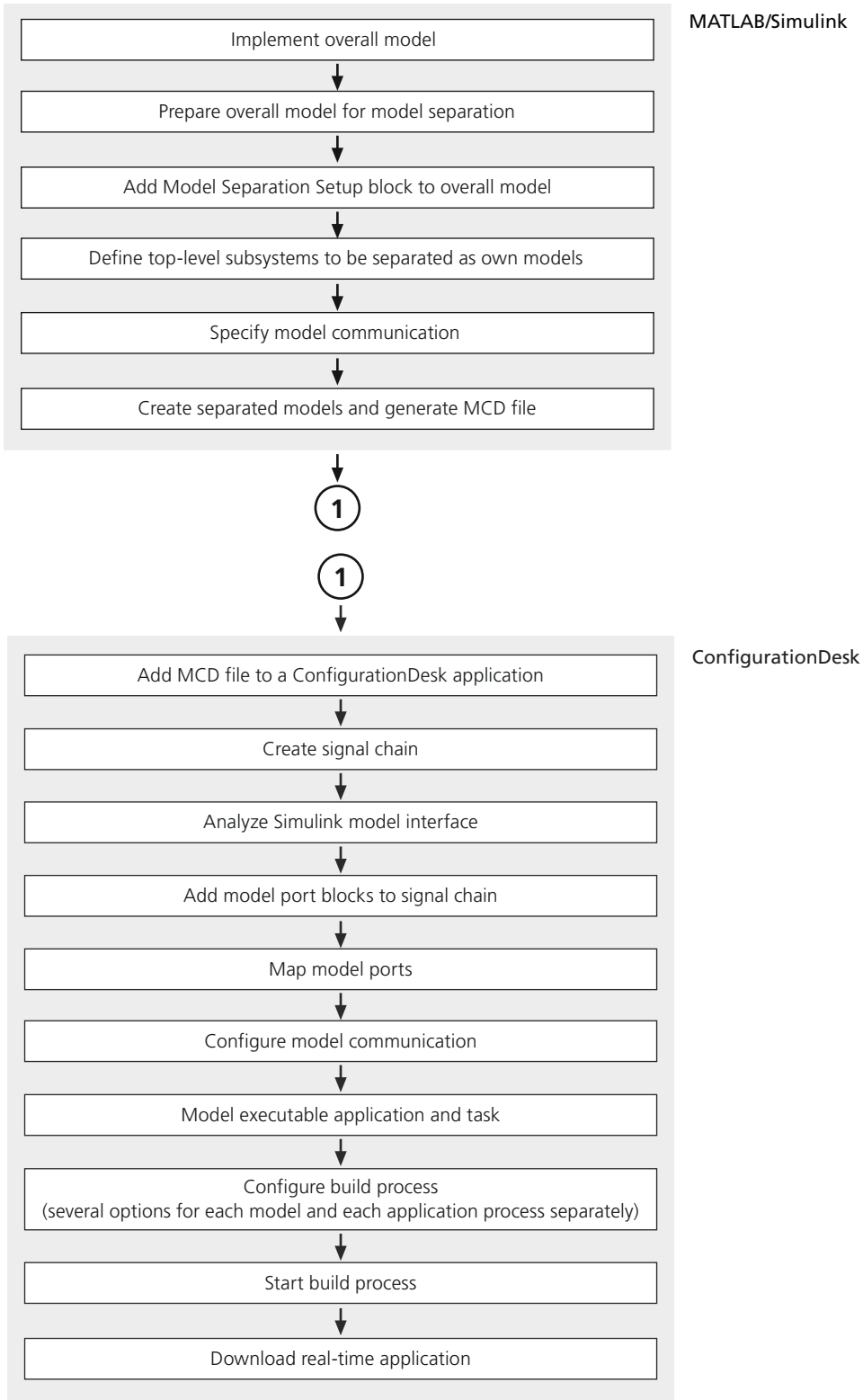
The following illustration gives you an overview of the use scenario. Parts that are specific to this scenario are in bold letters.



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.



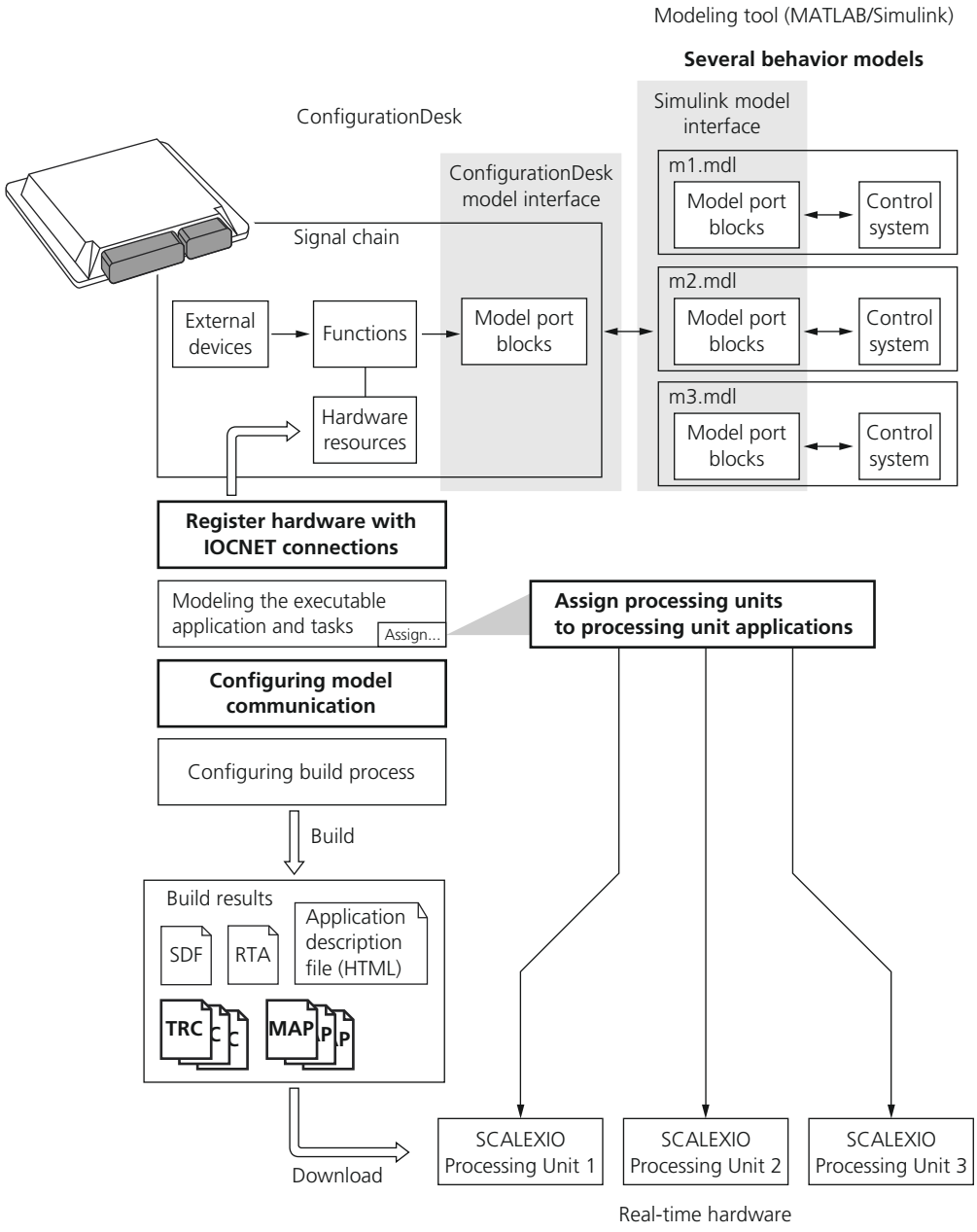
Creating a Multi-PU Application Using Multiple Behavior Models

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.

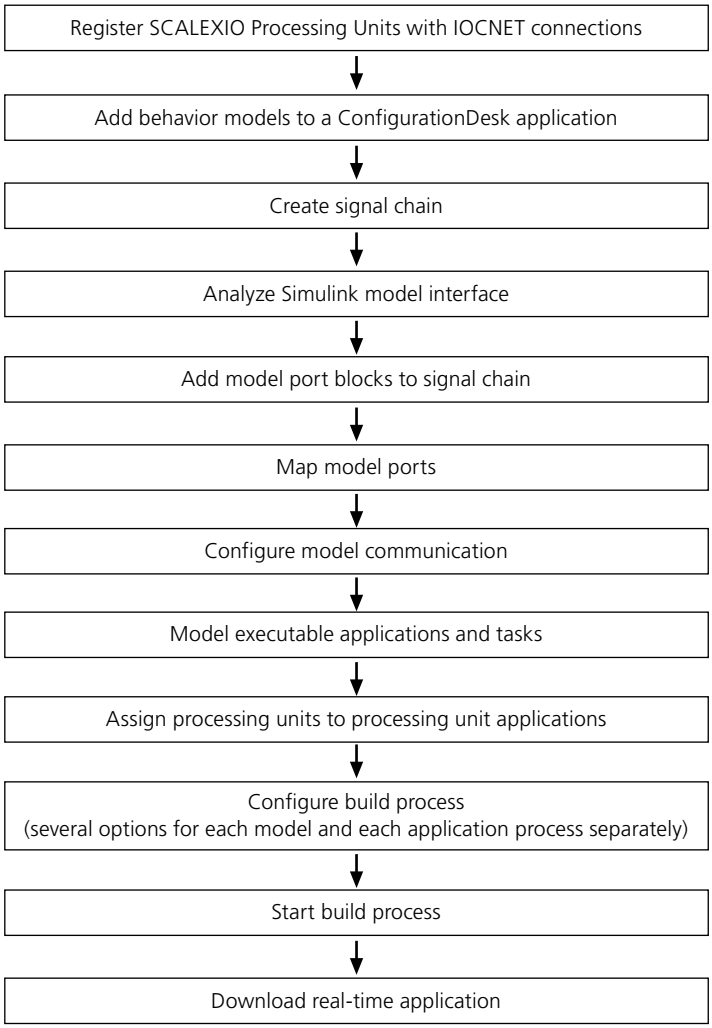
Overview

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.



Creating a Multi-PU Application Using One Overall Behavior Model

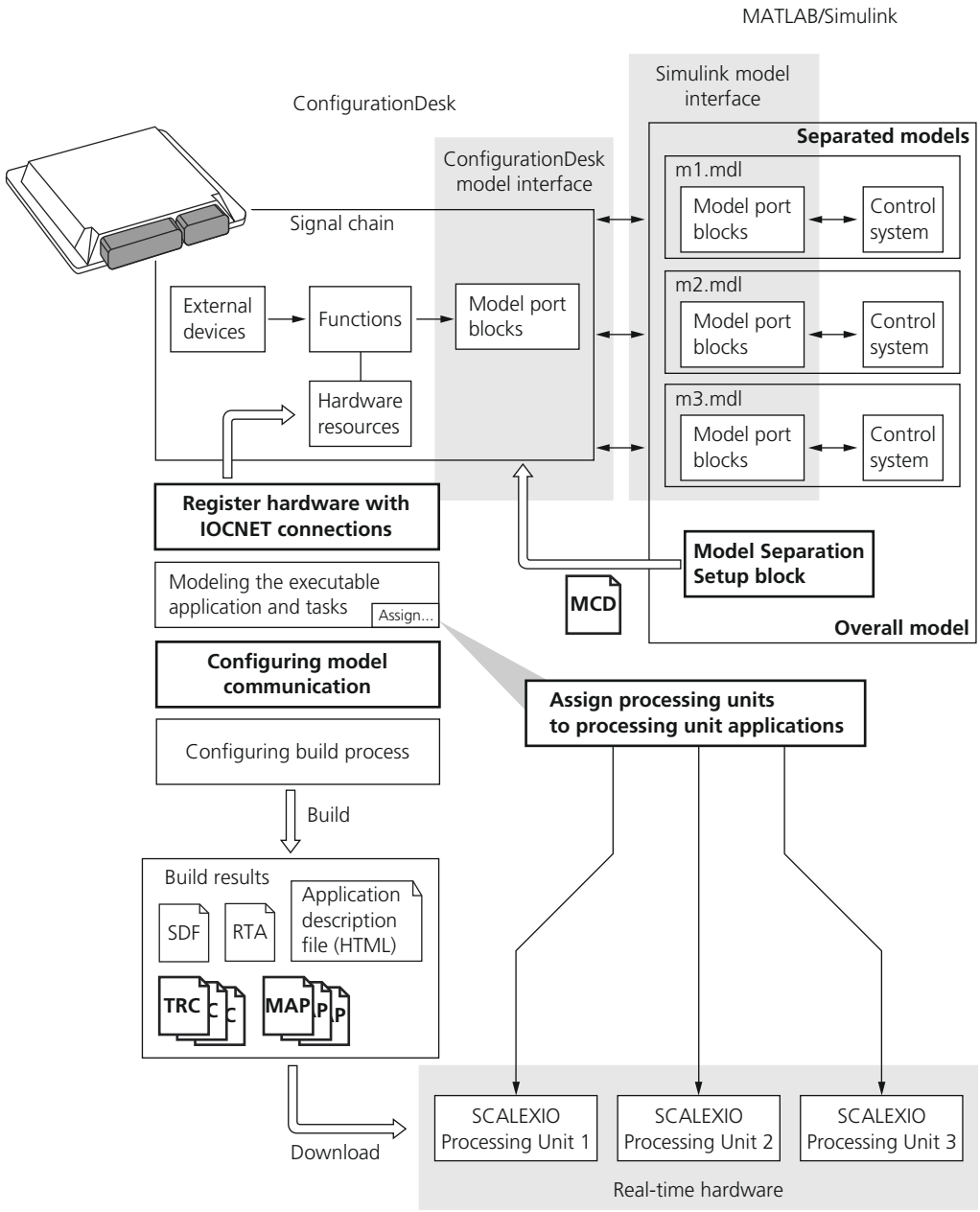
Use scenario

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

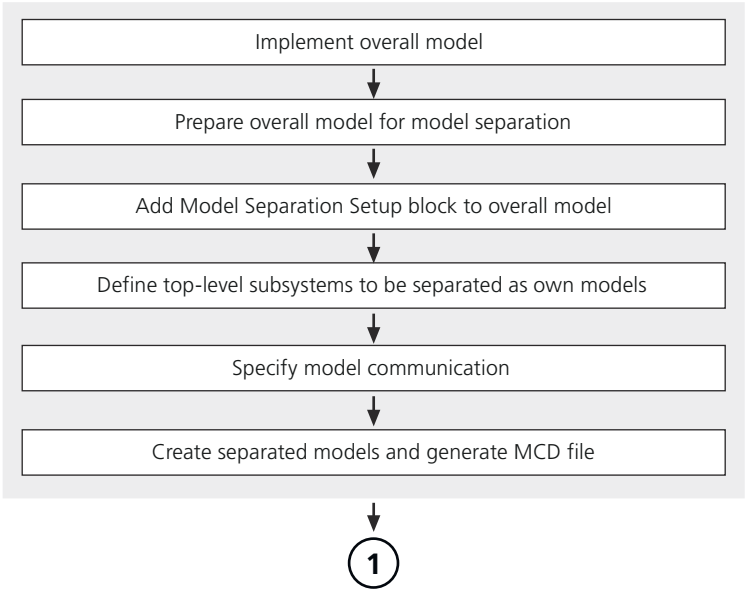
The following illustration gives you an overview of the use scenario. Parts that are specific to this scenario are in bold letters.

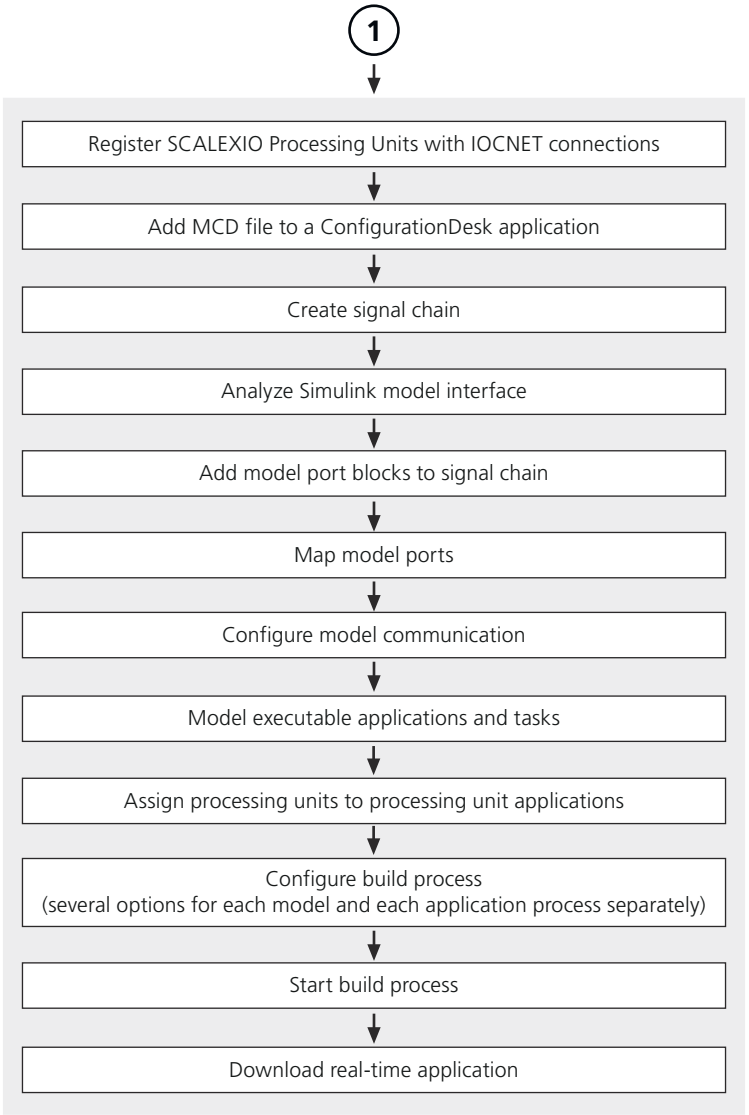


To separate models from an overall model in MATLAB/Simulink, you can use the **Model Separation Setup** block. This also generates an MCD file, which contains information on the separated models and their interconnections in the overall model.

Workflow

The workflow includes the steps that are specific for this use scenario.





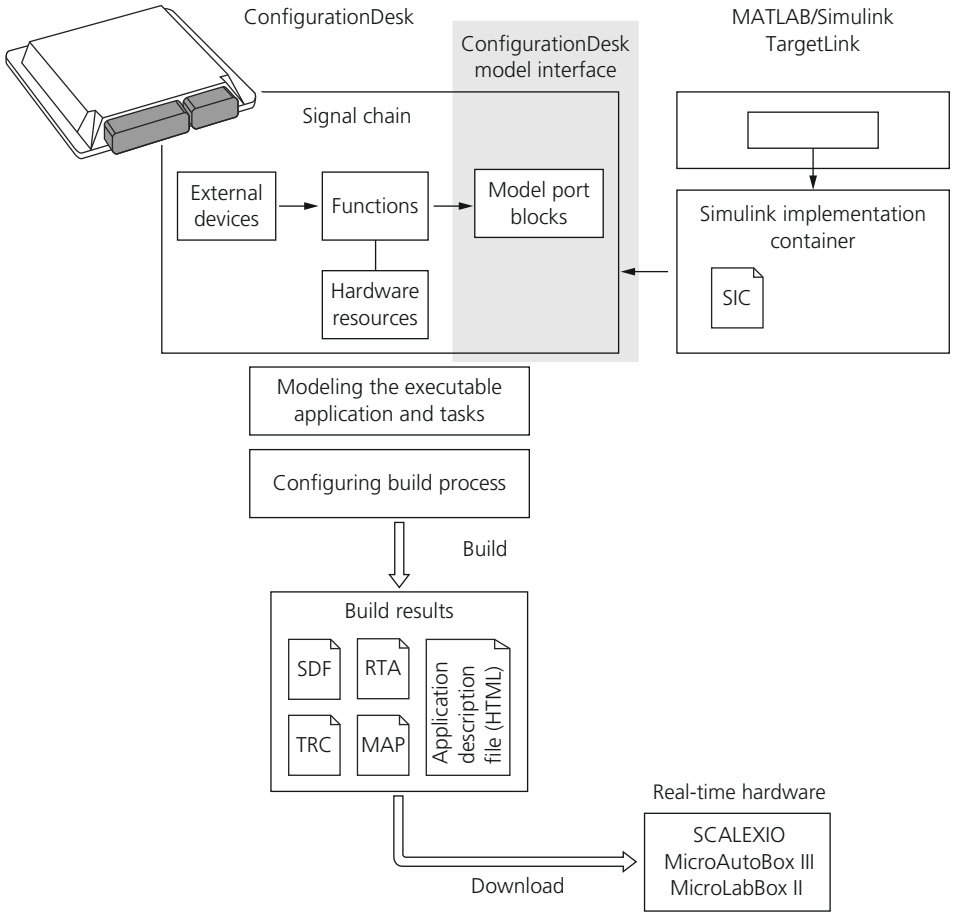
Workflow for Integrating Simulink Implementation Containers in Executable Applications

Use scenario

With ConfigurationDesk, you can integrate Simulink® implementation containers in your executable application.

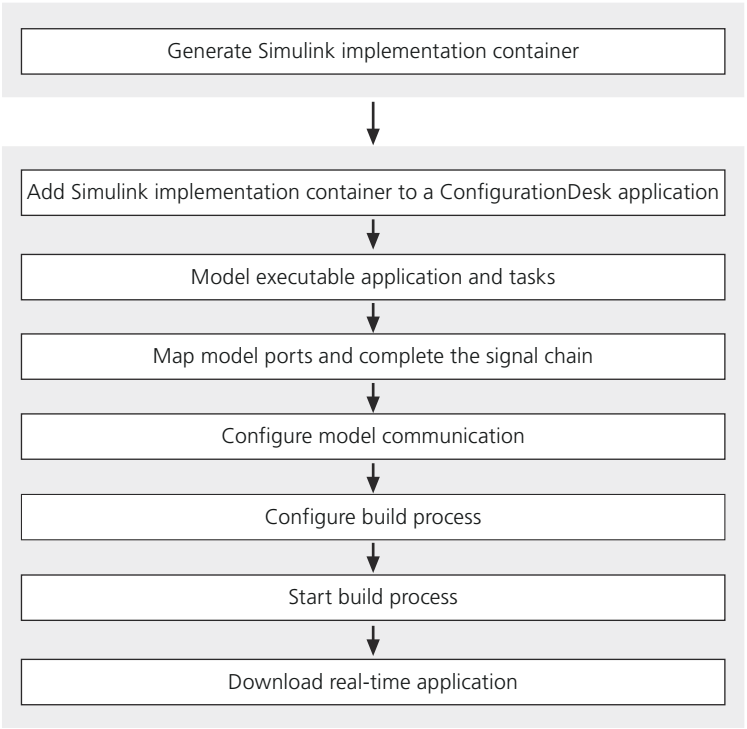
Overview

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.



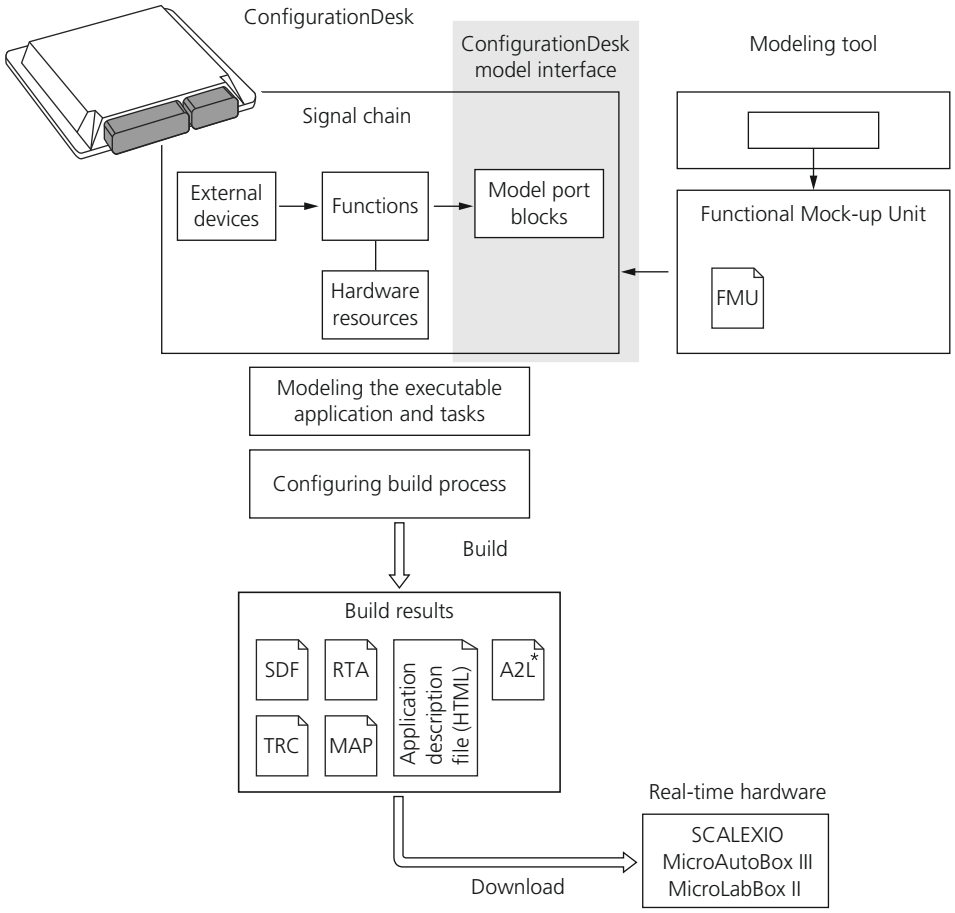
Workflow for Integrating FMUs in Executable Applications

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 Simulink implementation container, for example.

Overview

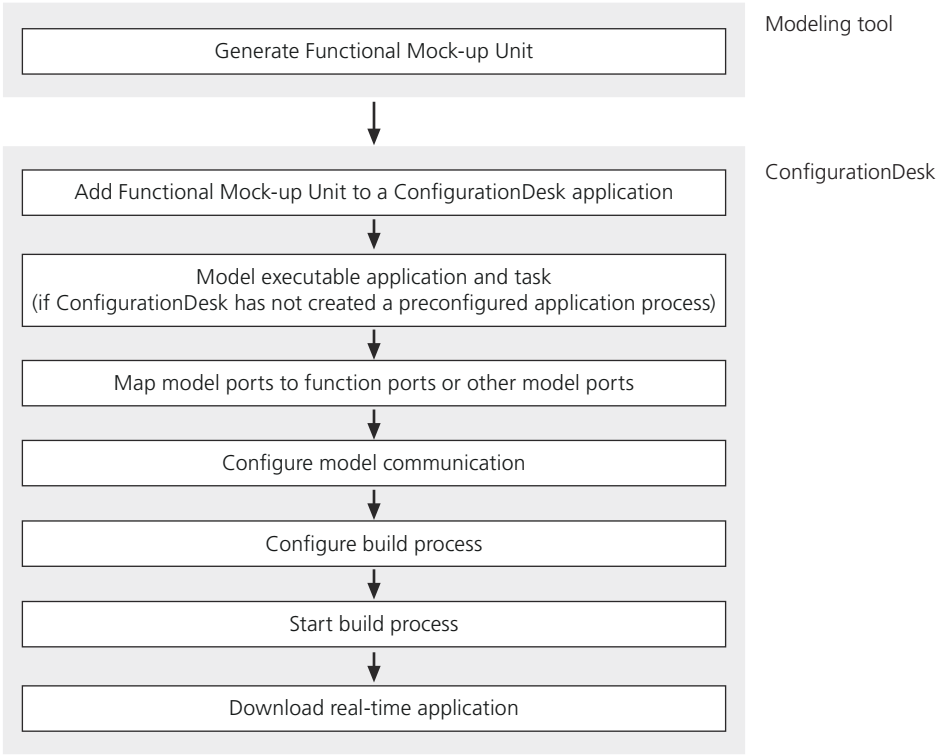
The following illustration gives you an overview of the use scenario:



*Only generated for binary FMUs that include A2L files and an XCP service implementation according to the fmi-ls-xcp standard

Workflow

The workflow includes the steps that are specific to implementing a real-time application containing a Functional Mock-up Unit.



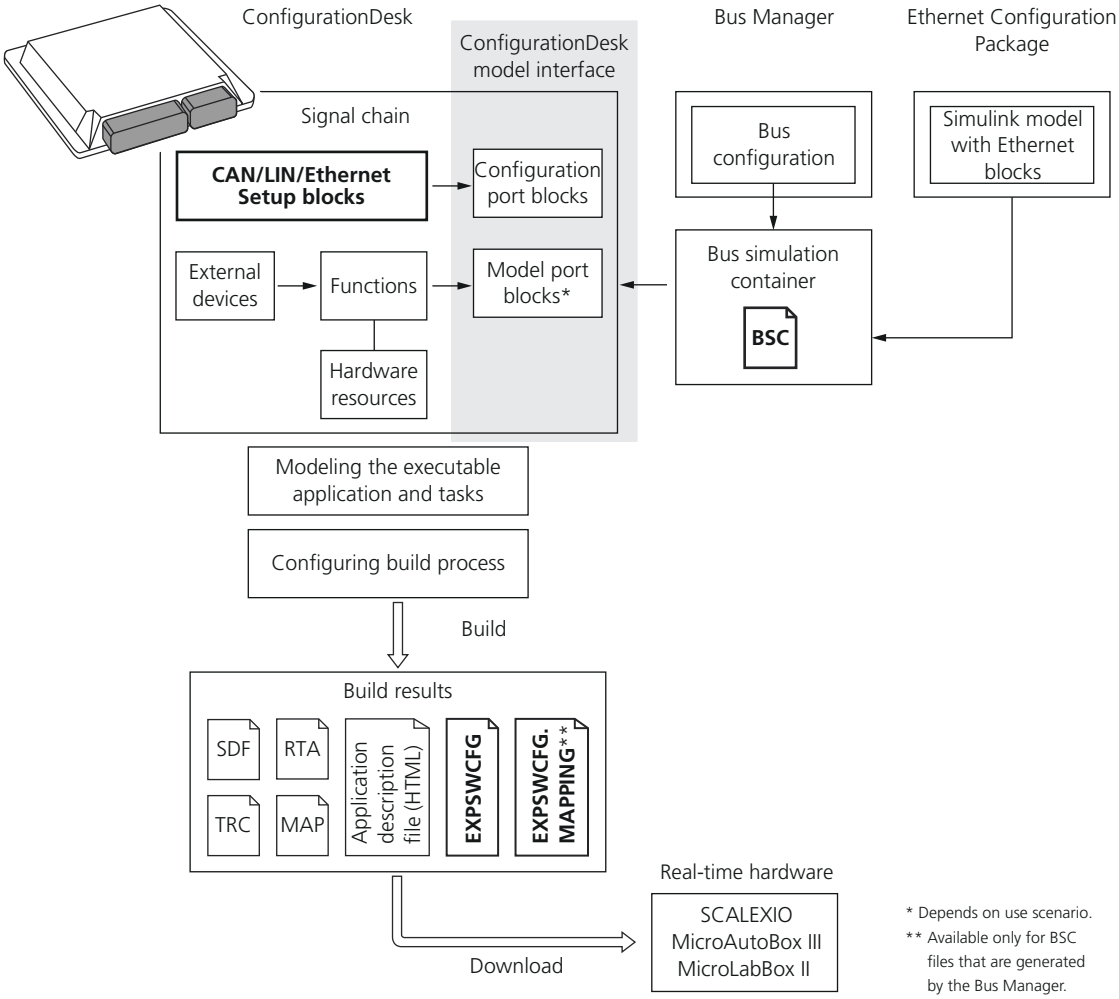
Workflow for Integrating Bus Simulation Containers in Executable Applications

Use scenario

With ConfigurationDesk, you can integrate bus simulation containers generated by the Bus Manager or the Ethernet Configuration Package in your executable application.

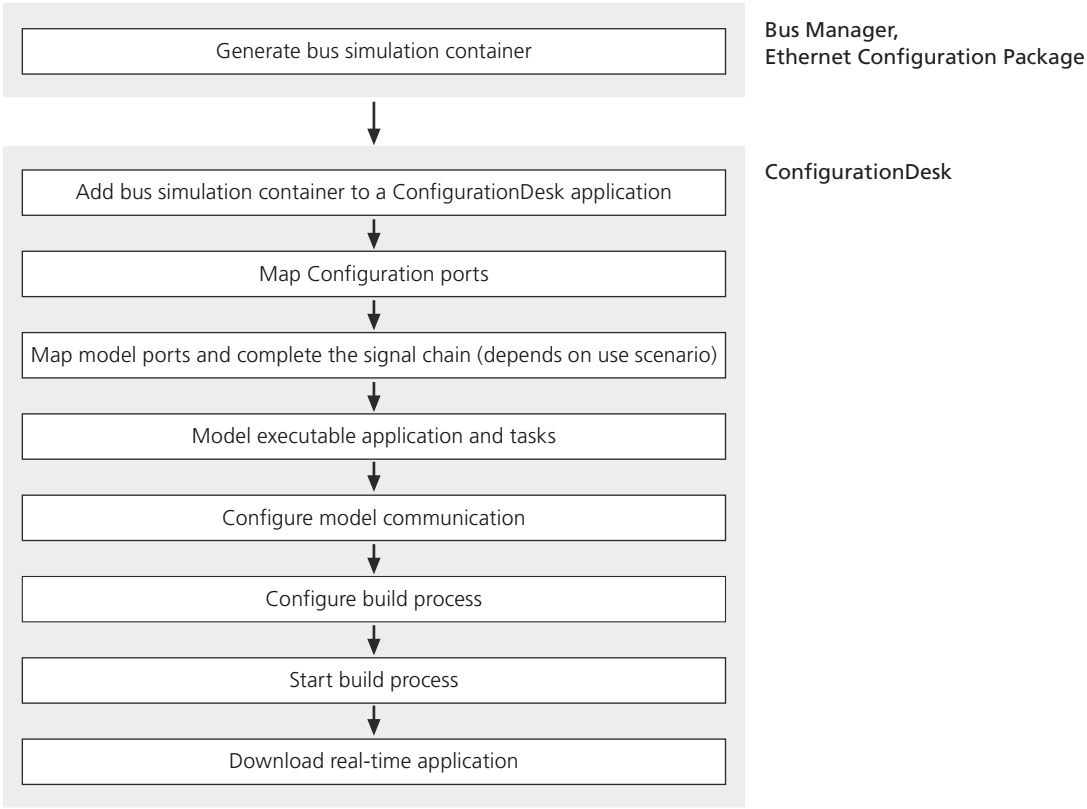
Overview

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 containing a bus simulation container.



Required Licenses

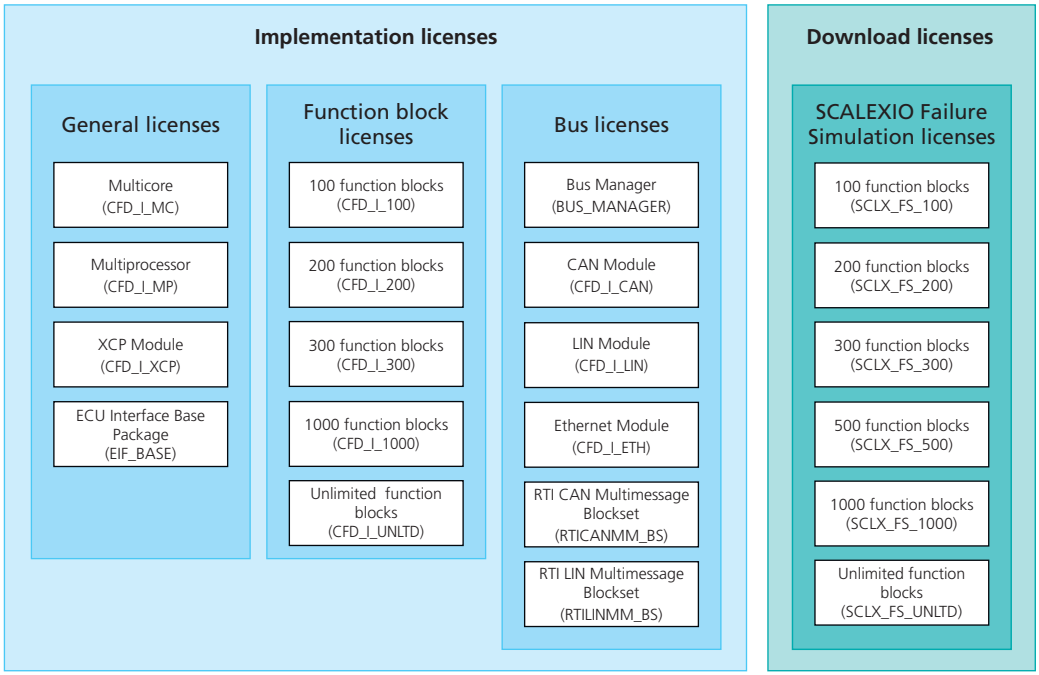
Where to go from here

Information in this section

Overview of Licenses

Overview

These are the available licenses to use the license-protected features of ConfigurationDesk:



General licenses

The following general licenses are available:

- ConfigurationDesk MultiCore

  This license (CFD_I_MC) is required if you want to implement an application that employs two or more processor cores. This applies in the following cases:

  - You want to implement a multimodel application with several application processes.
  - Your application contains an application process that employs two or more processor cores (**Number of required processor cores** property ≥ 2).

  If you want to implement a multimodel application with one application process that employs only one processor core, i.e., you do not want to build a multicore or multi-processing-unit (multi-PU) application, one of the function block licenses (CFD_I_xxx) is sufficient.

- ConfigurationDesk MultiProcessor

  This license (CFD_I_MP) is required if you want to implement a multimodel application and build a multi-processing-unit (multi-PU) application. Note, that the CFD_I_MC license is also required for multi-PU applications. If you purchase the CFD_I_MP license, the shipment automatically includes the CFD_I_MC license.

- ConfigurationDesk XCP Module

  This license (CFD_I_XCP) is required if you want to configure an XCP service for accessing Simulink models via A2L/XCP.

- **ECU Interface Base Package**

  This license (EIF_Base) is required if you want to implement applications which contain at least one **ECU Interface Configuration** function block to import an ECU interface container (EIC) file.

Function block licenses

At least one function block license (CFD_I_xxx) is required to implement and use function blocks in your application. A function block license works as a basic license and enables you 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 61.

Bus licenses

The following licenses for bus communication are available:

- **Bus Manager**

  This license (BUS_MANAGER) is required, if you want to implement bus communication via ConfigurationDesk's Bus Manager.
- **ConfigurationDesk - CAN Module**

  This license (CFD_I_CAN) is required when you use a **CAN** function block, for example, in the following cases:
  - You want to implement CAN communication via the RTI CAN MultiMessage Blockset.
  - You want to implement CAN communication via imported ECU interface container (EIC) files or imported BSC files.
  - You use ConfigurationDesk's Bus Manager and you want to implement CAN communication.
- **ConfigurationDesk - Ethernet Module**

  This license (CFD_I_ETH) is required, if you want to implement Ethernet communication via an Ethernet function block (for example, the **Ethernet Setup** function block).
- **ConfigurationDesk - LIN Module**

  This license (CFD_I_LIN) is required when you use a **LIN** function block, for example, in the following cases:
  - You want to implement LIN communication via the RTI CAN MultiMessage Blockset.
  - You want to implement LIN communication via imported BSC files.
  - You use ConfigurationDesk's Bus Manager and you want to implement LIN communication.
- **RTI CAN Multimessage Blockset**

  This license (RTICANMM_BS) is required if you want to use at least one Simulink model that contains blocks of the RTI CAN MultiMessage Blockset.
- **RTI LIN Multimessage Blockset**

  This license (RTILINMM_BS) is required if you want to use at least one Simulink model that contains blocks of the RTI LIN MultiMessage Blockset.

FIU licenses

The FIU licenses are download licenses and 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 62.

Solution licenses

Add-ons (dSPACE Solutions) are available for ConfigurationDesk. If these solutions are license-protected, the corresponding licenses must be available on your host PC to work with the solution.

dSPACE Solutions is a collective term for various dSPACE software products that can be installed and used as add-ons to the standard products such as ConfigurationDesk. Solutions cover special use scenarios that go beyond the standard use of dSPACE products.

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.

Invalid licenses

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.

How to Show Accessible and Used Licenses

Objective

ConfigurationDesk provides a license overview, which shows the available licenses to use the license-protected features of ConfigurationDesk. The overview also shows the license status, for example, if a license is accessible, or in use in your active ConfigurationDesk application.

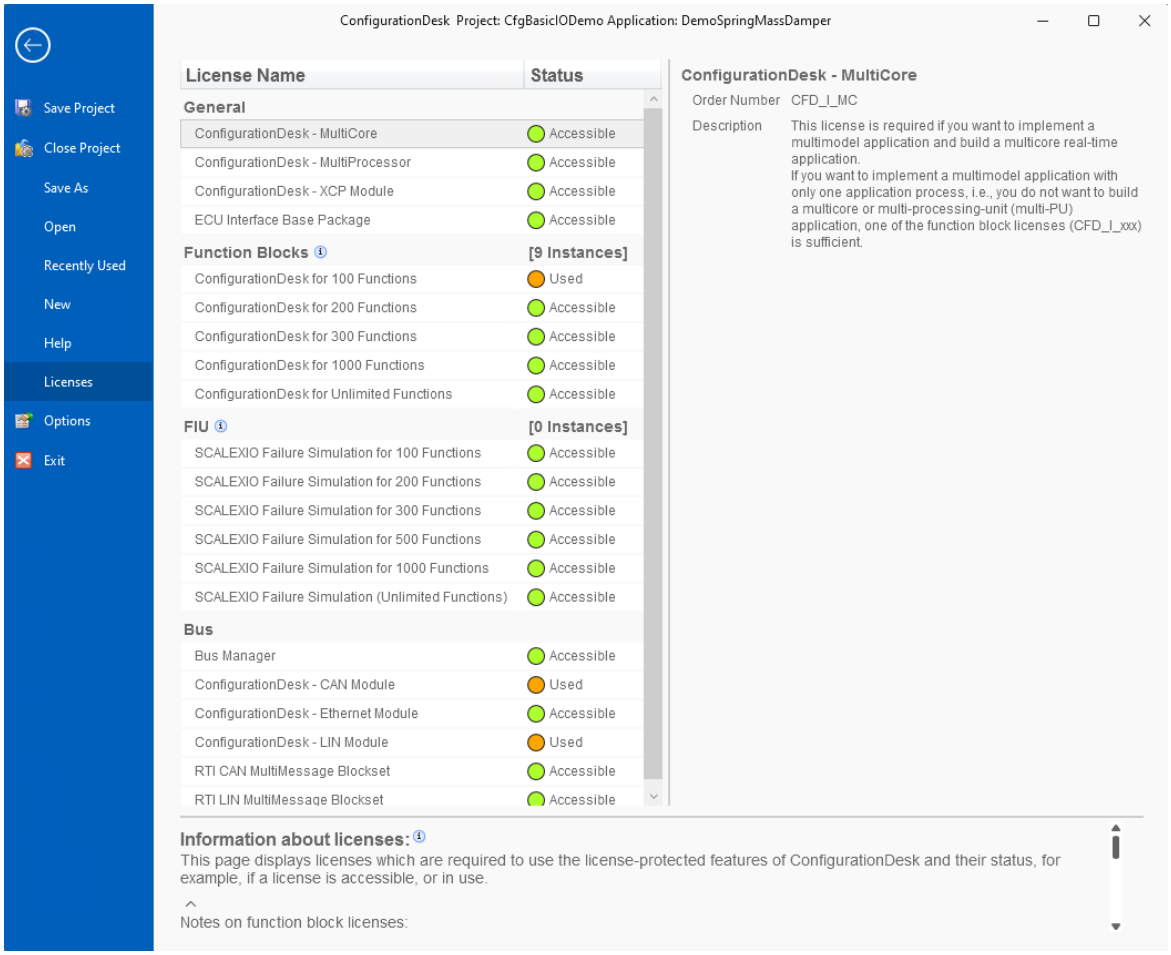
Method

To display accessible and used licenses

1 On the File ribbon, click Licenses.

Result

ConfigurationDesk opens the Licenses page.



- 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 Function Block Licenses

Objective

At least one function block license is required to implement and use function blocks in a ConfigurationDesk application.

Handling function block licenses

At least one function block license is required to implement and use function blocks in your ConfigurationDesk application. They are available in different sizes for different numbers (100, 200, 300, 1000, or unlimited) of instantiated function blocks in your application. 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, it is possible to delete elements of the application and save the application.

Tip

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

Not affected function block types

The following function block types providing bus communication are not protected by this function block licenses and therefore not counted when used:

- CAN function block
- LIN function block
- FlexRay function block
- Bus Configuration function block
- Ethernet function blocks, for example, the Ethernet Setup function block

Note

However, you need other specific bus communication licenses to implement and use the above listed function blocks in your ConfigurationDesk application. Refer to Overview of Licenses on page 58.

Details on SCALEXIO FIU Licenses

Objective

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. For example, this also can be a PC on which ControlDesk is installed.

Notes on using SCALEXIO FIU licenses

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.

The licenses are available in different sizes for different numbers of instantiated function blocks in your ConfigurationDesk application.

Note

For FIU licenses only those function blocks are counted that are assigned to a hardware resource providing failure simulation support. For example, function blocks that are assigned to channel sets of the DS2601, DS2680, and DS2690 boards are counted.

The affected function blocks are counted in the same way as this is done for function block licenses.

Notes on Using Floating Network Licenses

Objective

When you use ConfigurationDesk with floating network licenses, the dSPACE License Client requests the required licenses from the connected dSPACE License Server.

Details on the license behavior

One function block 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.

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.

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

Further Documentation

Documentation Overview

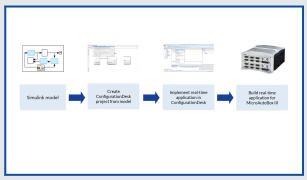
Introduction

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

Video introducing ConfigurationDesk for rapid control prototyping

Introduction to ConfigurationDesk for Rapid Control Prototyping

This video shows you how to get your model to run on a dSPACE prototyping system, using the MicroAutoBox III as an example. The video starts with your Simulink model and shows the implementation, build, and download workflow in ConfigurationDesk.



To watch this video, click the following link or scan the QR code:
https://www.dspace.com/dspace-help/wi62w

Printed Documents

A subset of the entire user documentation is available on demand as printed documents. If you wish to receive printed user documentation, you can order it free of charge by using the following link: https://www.dspace.com/go/requestreleasematerial.

Available documentation

The table shows the documents which are available for the SCALEXIO hardware, the MicroAutoBox III hardware, ConfigurationDesk, and the Model Interface Package for Simulink.

Document	Contents	PDF File Name	Printed Version[1)]
SCALEXIO System			
SCALEXIO System Overview	Introduces you to the SCALEXIO system: ▪ Provides an overview of the simulator and the software tools which are used to implement the simulation model on the simulator and to perform ECU tests. ▪ Describes the whole workflow for working with a SCALEXIO system.	SCALEXIOSystemOverview.pdf	–
SCALEXIO Rack Getting Started	Describes how to start handling with a SCALEXIO rack. The steps necessary for powering and connecting are described as well as how to register a SCALEXIO rack to make it known to ConfigurationDesk.	SCALEXIORackGettingStarted.pdf	X
SCALEXIO LabBox Getting Started	Describes how to start handling with a SCALEXIO LabBox. The steps necessary for powering and connecting are described as well as how to register a SCALEXIO LabBox to make it known to ConfigurationDesk.	SCALEXIOLabBoxGettingStarted.pdf	X
SCALEXIO Hardware Installation and Configuration	Introduces you to the SCALEXIO hardware: ▪ Describes the simulator components. ▪ Provides all information relevant to installing and connecting the SCALEXIO hardware. ▪ Provides hardware-related reference information on the components of the SCALEXIO hardware, such as technical data.	SCALEXIOHardwareInstallationConfiguration.pdf	X
MicroAutoBox III			
MicroAutoBox III - Getting Started	Describes how to start handling with the MicroAutoBox III. The steps necessary for powering and connecting are described as well as how to register the MicroAutoBox III to make it known to ConfigurationDesk. Furthermore, this guide provides information for MicroAutoBox II users.	MicroAutoBoxIIIGettingStarted.pdf	X
MicroAutoBox III Hardware Installation and Configuration	Introduces you to the MicroAutoBox III hardware: ▪ Describes the MicroAutoBox III components. ▪ Provides all information relevant to installing and connecting the MicroAutoBox III hardware. ▪ Provides hardware-related reference information on the components of the MicroAutoBox III hardware, such as technical data.	MicroAutoBoxIIIHardwareInstallationAndConfiguration.pdf	X
MicroAutoBox III - Hardware and Software Overview	Gives a brief introduction on the MicroAutoBox III and an overview on the dSPACE hardware and software products to support different use scenarios.	MicroAutoBoxIIIHardwareAndSoftwareOverview.pdf	–
MicroLabBox II			
MicroLabBox II - Getting Started	Introduces you to the MicroLabBox II. It describes how to connect and register a MicroLabBox II to make it known to ConfigurationDesk and to run a demo experiment.	MicroLabBoxIIGettingStarted.pdf	X

Document	Contents	PDF File Name	Printed Version[1)]
	In addition, this guide provides information for migrating MicroLabBox RTI models.		
MicroLabBox II Hardware Installation and Configuration	This document describes how to install and configure the MicroLabBox II: ▪ Setting up the Host communication. ▪ Configuring features of the MicroLabBox II. ▪ Information to connect external devices, such as pinouts and signal descriptions. ▪ Self-help on hardware-related problems.	MicroLabBoxIIHardwareInstallationAndConfiguration.pdf	X
Implementing in ConfigurationDesk			
ConfigurationDesk - Getting Started	Describes the basic concepts of ConfigurationDesk and different workflows to implement a real-time application in ConfigurationDesk.	ConfigurationDeskGettingStarted.pdf	X
ConfigurationDesk - New Features and Migration	Provides information on new features, a feature history, migration steps to the current version, a migration history, and information on compatibility and discontinuations of ConfigurationDesk.	ConfigurationDeskNewFeaturesAndMigration.pdf	–
ConfigurationDesk - Tutorial Starting with External Devices	The dSPACE software provides the `CfgStartingWithExternalDevices` demo project. This tutorial helps you to use the demo project to learn the basic steps in ConfigurationDesk when you start out with an external device such as an ECU.	ConfigurationDeskTutorialStartingwithExternalDevices.pdf	X
ConfigurationDesk - Tutorial Starting with Simulink	The dSPACE software provides the `CfgStartingWithSimulinkTutorial` demo project. This tutorial helps you to use the demo project to learn the basic steps in ConfigurationDesk when you start out with a Simulink behavior model.	ConfigurationDeskTutorialStartingwithSimulink.pdf	X
ConfigurationDesk - Tutorial MicroAutoBox III	The dSPACE software provides the `CfgMicroAutoBoxIIITutorial` demo project. This tutorial helps you to use the demo project to learn the basic steps in ConfigurationDesk when working with a MicroAutoBox III.	ConfigurationDeskTutorialMicroAutoBoxIII.pdf	X
ConfigurationDesk - Real-Time Implementation Guide	Introduces you to ConfigurationDesk and shows how to use it, for example: ▪ Setting up projects and applications ▪ Managing real-time hardware ▪ Specifying the model interface ▪ Building and downloading real-time applications to dSPACE real-time hardware	ConfigurationDeskImplementationGuide.pdf	X
ConfigurationDesk - I/O Function Implementation Guide	Gives you feature-related information on implementing and using I/O functionality in ConfigurationDesk.	ConfigurationDeskIOFunctionImplementationGuide.pdf	–
Bus Manager Tutorial	The dSPACE software provides the `BusManagerTutorial` demo project. This tutorial helps you to use the demo project to learn the basic steps when working with the Bus Manager.	BusManagerTutorial.pdf	–

Document	Contents	PDF File Name	Printed Version[1)]
ConfigurationDesk - Bus Manager Implementation Guide	Provides basic information and instructions on implementing bus communication via ConfigurationDesk's Bus Manager.	ConfigurationDeskBusManagerImplementationGuide.pdf	–
ConfigurationDesk - Using Demo Projects	Provides descriptions of the ConfigurationDesk demo projects or refers you to detailed descriptions in other documents.	ConfigurationDeskUsingDemoProjects.pdf	–
ConfigurationDesk Glossary	Explains the most important expressions and naming conventions used in the ConfigurationDesk documentation.	ConfigurationDeskGlossary.pdf	–
ConfigurationDesk - Tutorials Automating Tool Handling	Guides you through projects to learn the basic steps in automating ConfigurationDesk using Python.	ConfigurationDeskTutorialsAutomatingToolHandling.pdf	–
ConfigurationDesk - Automating Tool Handling	Provides detailed information on ConfigurationDesk's automation interface including a graphical representation of ConfigurationDesk's object model and the API description of all elements that are accessible via ConfigurationDesk's automation interface.	ConfigurationDeskAutomatingToolHandling.pdf	–
ConfigurationDesk - Custom I/O Function Implementation Guide	Shows you how to create custom function blocks for ConfigurationDesk.	ConfigurationDeskCustomIOFunctionImplementationGuide.pdf	–
ConfigurationDesk - Syntax of the TRC File	Provides information on the syntax you must adhere to if you want to write a TRC file manually.	ConfigurationDeskSyntaxTRCfile.pdf	–
ConfigurationDesk - UART Implementation	Shows you how to implement a protocol for UART serial communication in ConfigurationDesk.	ConfigurationDeskUARTImplementation.pdf	–
Bus Custom Code Interface Handling	Introduces you to the Bus Custom Code interface provided by dSPACE and shows how to integrate user-specific bus functionality in bus implementation software, for example, ConfigurationDesk.	BusCustomCodeInterfaceHandling.pdf	–
ConfigurationDesk - User Interface Reference	Provides detailed information on the ribbons, context menus, and dialogs contained in ConfigurationDesk.	ConfigurationDeskUserInterfaceReference.pdf	–
ConfigurationDesk - Function Block Properties	Provides detailed reference information on the properties of ConfigurationDesk's function blocks.	ConfigurationDeskFunctionBlockProperties.pdf	–
ConfigurationDesk - Hardware Resource Properties	Provides detailed information on the characteristics of the hardware resources that are supported by ConfigurationDesk.	ConfigurationDeskHardwareResourceProperties.pdf	–
ConfigurationDesk - Conflicts	Decribes the conflicts that can occur while you are implementing a ConfigurationDesk application. The conflicts are displayed in the Conflicts Viewer.	ConfigurationDeskConflicts.pdf	–
Modeling in Simulink			
Model Interface Package for Simulink - Modeling Guide	Introduces you to implementing behavior models that you can use with ConfigurationDesk. The main focus is on modeling aspects in MATLAB®/Simulink®, but you can also find information on modeling-specific aspects in ConfigurationDesk.	ModelInterfacePackageforSimulinkGuide.pdf	–

Document	Contents	PDF File Name	Printed Version[1)]
Model Interface Package for Simulink - Reference	Provides all the reference information you need for working with the Simulink Model Block Libraries which are available for ConfigurationDesk.	ModelInterfacePackageForSimulinkReference.pdf	–
Model Interface Package for Simulink - API Reference	Provides reference information on the API functions of the Model Interface Package for Simulink.	ModelInterfacePackageForSimulinkAPIReference.pdf	–
Model Interface Package for Simulink - Message Reference	Provides reference information on error messages that might be displayed when working with the Model Interface Package for Simulink.	ModelInterfacePackageForSimulinkMessageReference.pdf	–

1) Available on demand.

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

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 description file An HTML file that is created in the **Build Results** folder during the build process. It contains the following information:

- Project name
- Application name
- Build time
- Project description taken from the **Description** field of the ConfigurationDesk project
- Application description taken from the **Description** field of the ConfigurationDesk application
- List of names and paths of all contained model implementations

  For BSC files, the content of the **Content** property is also displayed.
- Structure/hierarchy of the application (processing unit applications - application processes - models)

Application description files are named according to the following naming scheme:

```
<application name>_description.html
```

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. Unless stated otherwise, the Bus Manager provides the same functionalities for all basic PDUs, such as ISignal IPDUs or NMPDUs.

Behavior model A model that contains the control algorithm for a controller (function prototyping system) or the algorithm of the controlled system (hardware-in-the-loop system). It does not contain I/O functionality nor access to the hardware. Behavior models can be modeled, for example, in MATLAB®/Simulink® by using Simulink Blocksets and Toolboxes from the MathWorks®.

You can add Simulink behavior models to a ConfigurationDesk application. You can also add code container files containing a behavior model such as Functional Mock-up Units, or Simulink implementation containers to a ConfigurationDesk application.

Bidirectional signal port A signal port that is independent of a data direction or current flow. This port is used, for example, to implement bus communication.

BSC file A bus simulation container file that is generated with the Bus Manager or the Ethernet Configuration Package. A BSC file that is generated with the Bus Manager contains the configured bus communication of one application process. A BSC file that is generated with the Ethernet Configuration Package contains the configured bus communication of one project.

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 dSPACE real-time hardware. 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 Ports table A pane that lets you access and configure function ports and event ports of bus configurations.

Bus Configurations table A pane that lets you access and configure bus configurations of a ConfigurationDesk application.

Bus Custom Code interface A C-based API interface provided by dSPACE. It is required to implement user code and its functionalities in executable applications. In general, the Bus Custom Code interface lets you do the following:

- Specify the unique user code ID for each user code implementation. The ID is required to unambiguously reference the user code implementation in bus configurations.
- Initialize the functionalities provided by user code in the executable application.
- Exchange data between the user code and the Bus Manager, for example, to access properties of secured IPDUs or write calculated checksum values to ISignals of an IPDU.

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 or the Ethernet Configuration Package. Bus simulation container (BSC) files that are generated with the Bus Manager contain CAN and/or LIN communication, BSC files generated with the Ethernet Configuration Package contain Ethernet communication.

Depending on the contained bus communication, BSC files can be used in VEOS, in ConfigurationDesk, and/or in RTMaps:

- In VEOS, they let you implement the bus communication in an offline simulation application to perform SIL simulation on VEOS.
- In ConfigurationDesk, they let you implement the bus communication in a real-time application for SCALEXIO, MicroAutoBox III, or MicroLabBox II.
- In RTMaps, they let you implement the bus communication in an RTMaps-based application for the AUTERA AutoBox, for example.

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.

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 A short form of communication cluster.

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

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

or

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

Communication cluster A communication network of network nodes that are connected to the same physical channels and share the same bus protocol and address range.

Communication matrix A file that defines the communication of a bus network. It can describe the bus communication of one communication cluster or a bus network consisting of different bus systems and clusters. Files of various file formats can be used as a communication matrix: For example, AUTOSAR system description files, DBC files, LDF files, and FIBEX files.

Communication package A package that bundles Data Inport blocks which are connected to Data Outport blocks. Hence, it also bundles the signals that are received by these blocks. If Data Inport blocks are executed within the same task and belong to the same communication package, their data inports are read simultaneously. If Data Outport blocks that are connected to the Data Inport blocks are executed in the same task, their output signals are sent

simultaneously in one data package. Thus, communication packages guarantee simultaneous signal updates within a running task (data snapshot).

Configuration port A port that lets you create the signal chain for the bus communication implemented in a Simulink behavior model. The following configuration ports are available:

- The configuration port of a Configuration Port block.
- The **Configuration** port of a **CAN**, **LIN**, or **FlexRay** function block.

To create the signal chain for bus communication, the configuration port of a **Configuration Port** block must be mapped to the **Configuration** port of a **CAN**, **LIN**, or **FlexRay** function block.

Configuration Port block A model port block that is created in ConfigurationDesk during model analysis for each of the following blocks found in the Simulink behavior model:

- **RTICANMM ControllerSetup** block
- **RTILINMM ControllerSetup** block
- **FLEXRAYCONFIG UPDATE** block

Configuration Port blocks are also created for bus simulation containers. A Configuration Port block provides a configuration port that must be mapped to the Configuration port of a **CAN**, **LIN**, or **FlexRay** function block to create the signal chain for bus communication.

ConfigurationDesk application A part of a ConfigurationDesk project that represents a specific implementation. You can work with only one application at a time, and that application must be activated.

An application can contain:

- Device topology
- Hardware topology
- Model topology
- Communication matrices
- External cable harness
- Build results (after a successful build process has finished)

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 smaller IPDUs (i.e., contained IPDUs). Contained IPDUs can be multiplexed IPDUs, secured IPDUs, and ISignal IPDUs, for example.

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.

Countdown event An event that triggers a task exactly once, after a specified time (offset) has elapsed. Countdown events are provided by bus simulation containers (BSC files).

Custom code Custom source, header, and library files that must be compiled and linked to real-time applications. Custom code files can be specified via the **Build Configuration Set** options in the Build Configuration table. Do not confuse custom code with user code, which is a term used in the Bus Manager context.

Cycle time restriction A value of a runnable function that indicates the sample time the runnable function requires to achieve correct results. The cycle time restriction is indicated by the **Period** property of the runnable function in the Properties Browser.

D

Data inport A port that supplies data from ConfigurationDesk's function outports to the behavior model.

In a multimodel application, data inports also can be used to provide data from a data outport associated to another behavior model (model communication).

Data outport A port that supplies data from behavior model signals to ConfigurationDesk's function inports.

In a multimodel application, data outports also can be used to supply data to a data inport associated to another behavior model (model communication).

Data.CfgDeskApp file A ConfigurationDesk application file that contains links to all the documents related to an application.

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.

Delayed event An event that triggers a task when the specified delay time has expired after the execution of the source task started.

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 application-specific files that are used by the current user.

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

dSPACE Help The dSPACE online help that contains all the relevant user documentation for dSPACE products. Via the F1 key, 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, MicroAutoBox III, or MicroLabBox II 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
- Delayed event
- Countdown event
- Synchronized timer 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 a ZIP archive.

EXPSWCFG.MAPPING file An experiment software configuration mapping file that contains the mapping between CAN and/or LIN function blocks and the bus access requests of the bus configurations or BSC files for which the function blocks specify the bus access. It is created during the build process.

Extended multiplexed IPDU An IPDU that is used for extended signal multiplexing, which can be specified only in DBC communication matrices. The IPDU contains *multiplexer signals* and *multiplexed signals*: The value of a multiplexer signal can determine that specific multiplexed signals are included in the IPDU. However, with extended signal multiplexing, a multiplexed signal itself can serve as a multiplexer signal.

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, e.g., CAN or LIN.

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 that are different from other function block types.

To use a function block type in your ConfigurationDesk application, you have to create an instance of it. This instance is called a function block. Instances of function block types can be used multiple times in a ConfigurationDesk application. The types and their instantiated function blocks are displayed in the function library of the Function Browser.

Function Browser A pane that displays the function library in a hierarchical tree structure. Function block types are grouped in function classes. Instantiated function blocks are added below the corresponding function block type.

Function inport A function port that inputs the values from the behavior model to the function block to be processed by the function.

Function library A collection of function block types that allows access to the I/O functionality in ConfigurationDesk. The I/O functionality is based on function block types. The function library provides a structured tree view on the available function block types. It is displayed in the Function Browser.

Function outport A function port that outputs the value of a function to be used in the behavior model.

Function port An element of a function block that provides the interface to the behavior model via model port blocks.

Functional Mock-up Unit An archive file that describes and implements the functionality of a model based on the Functional Mock-up Interface (FMI) standard.

G

Global working view The default working view that always contains all signal chain elements.

H

Hardware resource A hardware element (normally a channel on an I/O board or I/O unit) that 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 that 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

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 program data that is used by the current user.

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

Logical signal An element of a function block that combines all the signal ports which belong together to provide the functionality of the signal. Each logical signal causes one or more channel requests. Channel requests are available after you have assigned a channel set to the logical signal.

Logical signal chain A term that describes the logical path of a signal between an external device and the behavior model. The main elements of the logical signal chain are represented by different graphical blocks (device blocks, function blocks and model port blocks). Every block has ports to provide the mapping to neighboring blocks.

In the documentation, usually the short form 'signal chain' is used instead.

M

MAP file A file that maps symbolic names to physical addresses.

Mapping line A graphical representation of a connection between two ports in the signal chain. You can draw mapping lines in a working view.

MCD file A model communication description file that is used to implement a multimodel application. It lets you add several behavior models that were separated from an overall model to the model topology.

The MCD file contains information on the separated models and information on the connections between them. The file is generated with the Model Separation Setup Block in MATLAB/Simulink. The block resides in the Model Interface Blockset (dsmpblib) from dSPACE.

MDL file A Simulink model file that contains the behavior model. You can add an MDL file to your ConfigurationDesk application.

As of MATLAB® R2012a, the file name extension for the Simulink model file has been changed from MDL to SLX by The MathWorks®.

Message Viewer A pane that displays a history of all error and warning messages that occur during work with ConfigurationDesk.

Model analysis A process that analyzes the model to determine the interface of a behavior model. You can select one of the following commands:

- Analyze Simulink Model (Model Interface Only)

  Analyzes the interface of a behavior model. The model topology of your active ConfigurationDesk application is updated with the properties of the analyzed behavior model.
- Analyze Simulink Model (Including Task Information)

  Analyzes the model interface and the elements of the behavior model that are relevant for the task configuration. The task configuration in ConfigurationDesk is then updated accordingly.

Model Browser A pane that lets you display and access the model topology of an active ConfigurationDesk application. The **Model Browser** provides access to all the model port blocks available in the behavior models that are linked to a ConfigurationDesk application. The model elements are displayed in a hierarchy, starting with the model roots. Below the model root, all the subsystems containing model port blocks are displayed as well as the associated model port blocks.

Model communication The exchange of signal data between the models within a multimodel application. To set up model communication, you must use a mapping line to connect a data outport (sending model) to a data inport (receiving model). The best way to set up model communication is using the Model Communication Browser.

Model Communication Browser A pane that lets you open and browse working views like the **Signal Chain Browser**, but shows only the Data Outport and Data Inport blocks and the mapping lines between them.

Model Communication Package table A pane that lets you create and configure model communication packages which are used for model communication in multimodel applications.

Model implementation An implementation of a behavior model. It can consist of source code files, precompiled objects or libraries, variable description files, and a description of the model's interface. Specific model implementation types are, for example, model implementation containers, such as Functional Mock-up Units or Simulink implementation containers.

Model implementation container A file archive that contains a model implementation. Examples are FMUs and SIC 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 and Functional Mock-up Units.

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.

Model port An element of a model port block. Model ports provide the interface to the function ports and to other model ports (in multimodel applications).

These are the types of model ports:

- Data inport
- Data outport
- Runnable function port
- Configuration port

Model port block A graphical representation of the ConfigurationDesk model interface in the signal chain. Model port blocks contain model ports that can be mapped to function blocks to provide the data flow between the function blocks in ConfigurationDesk and the behavior model. The model ports can also be mapped to the model ports of other model port blocks with data inports or data outports to set up model communication. Model port blocks are available in different types and can provide different port types:

- Data port blocks with data inports and data outports
- Runnable Function blocks with runnable function ports
- Configuration Port blocks with configuration ports. Configuration Port blocks are created during model analysis for each of the following blocks found in the Simulink behavior model:
  - RTICANMM ControllerSetup block
  - RTILINMM ControllerSetup block
  - FLEXRAYCONFIG UPDATE block

  Configuration Port blocks are also created for bus simulation containers.

Model Separation Setup Block A block that is contained in the Model Interface Package for Simulink. It is used to separate individual models from an overall model in MATLAB/Simulink. Additionally, model separation generates a model communication description file (MCD file) that contains information on the separated models and their connections. You can use this MCD file in ConfigurationDesk.

Model topology A component of a ConfigurationDesk application that contains information on the subsystems and the associated model port blocks of all the behavior models that have been added to a ConfigurationDesk application.

Model-Function Mapping Browser A pane that lets you create and update signal chains for Simulink behavior models. It directly connects them to I/O functionality in ConfigurationDesk.

MTFX file A file containing a model topology when explicitly exported. The file contains information on the interface to the behavior model, such as the implemented model port blocks including their subsystems and where they are used in the model.

Multicore real-time application A real-time application that is executed on several cores of one PU of the real-time hardware.

Multimodel application A real-time application that executes several behavior models in parallel on dSPACE real-time hardware.

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, i.e., as dynamic-part IPDUs. One of these alternatives is transmitted at a time.
- The selector field value indicates which ISignal IPDU is transmitted in the dynamic part during run time. For each ISignal IPDU that is configured as a dynamic-part alternative, there must be a unique selector field value. The selector field value is evaluated by the receiver of the multiplexed IPDU.
- The static part is one ISignal IPDU, i.e., the static-part 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 application An application that runs on VEOS to perform SIL simulation. For example, you can use the VEOS Player to build an offline simulation application and load the resulting OSA file to VEOS.

OSA file An offline simulation application file that is built with VEOS to perform SIL 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 that 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, or 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 dSPACE real-time hardware.

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 that 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. You must define a project or open an existing one to work with ConfigurationDesk. Projects are stored in a project location.

Project location A designated folder in your file system where ConfigurationDesk stores project and application folders. Several projects can use the same project location. ConfigurationDesk uses the Documents folder as the default project location. You can specify further project locations, but only one can be active at a time.

Project Manager A pane that provides access to ConfigurationDesk projects and applications and all the components and files they contain.

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, e.g., a SCALEXIO Processing Unit.

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, i.e., it is a non-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.

Signal chain A term used in the documentation as a short form for logical signal chain. Do not confuse it with the physical signal chain.

Signal Chain Browser A pane that lets you open and browse working views such as the Global working view or user-defined working views.

Signal inport A signal port that represents an electrical connection point of a function block which provides signal measurement (= input) functionality.

Signal outport A signal port that represents an electrical connection point of a function block which provides signal generation (= output) functionality.

Signal port An element of a function block that provides the interface to external devices (e.g., ECUs) via device blocks. It represents an electrical connection point of a function block.

Signal reference port A signal port that represents a connection point for the reference potential of inports, outports and bidirectional ports. For example: With differential signals, this is a reference signal, with single-ended signals, it is the ground signal (GND).

SIL simulation Abbreviation of *software-in-the-loop 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. SIL simulations are independent from real time and can run on VEOS.

Simulink implementation container A container that contains the model code of a Simulink behavior model. A Simulink implementation container is generated from a Simulink behavior model by using the Model Interface Package

for Simulink. The file name extension of a Simulink implementation container is SIC.

Simulink model interface The part of the model interface that is available in the connected Simulink behavior model.

Single-core real-time application An executable application that is executed on only one core of the real-time hardware.

Single-PU system Abbreviation of single-processing-unit system.

A system consisting of exactly one PU and the directly connected I/O units and I/O routers.

SLX file A Simulink model file that contains the behavior model. You can add an SLX file to your ConfigurationDesk application.

As of MATLAB® R2012a, the file name extension for the Simulink model file has been changed from MDL to SLX by The MathWorks®.

Software event An event that is activated from within a task to trigger another subordinate task. Consider the following example: A multi-tasking Simulink behavior model has a base rate task with a sample time = 1 ms and a periodic task with a sample time = 4 ms. In this case, the periodic task is triggered on every fourth execution of the base rate task via a software event. Software events are available in ConfigurationDesk after model analysis.

Source Code Editor A Python editor that lets you open and edit Python scripts that you open from or create in a ConfigurationDesk project in a window in the working area. You cannot run a Python script in a **Source Code Editor** window. To run a Python script you can use the **Run Script** command in the Interpreter or on the **Automation** ribbon or the **Run** context menu command in the Project Manager.

Structured data port A hierarchically structured port of a Data Inport block or a Data Outport block. Each structured data port consists of more structured and/or unstructured data ports. The structured data ports can consist of signals with different data types (single, double, int8, int16, int32, int64, uint8, uint16, uint32, uint64, Boolean).

Synchronized timer event A periodic event that lets you trigger a task periodically and synchronously to a global time. You can specify a period, a global time domain identifier, and an optional offset for a synchronized timer event.

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 Simulink implementation container (SIC file) to be used in ConfigurationDesk.

Task A piece of code whose execution is controlled by a real-time operating system (RTOS). A task is usually triggered by an event, and executes one or more runnable functions. In a ConfigurationDesk application, there are predefined tasks that are provided by executable application components. In addition, you can create user-defined tasks that are triggered, for example, by I/O events. Regardless of the type of task, you can configure the tasks by specifying the priority, the DAQ raster name, etc.

Task Configuration table A pane that lets you configure the tasks of an executable application.

Temporary working view A working view that can be used for drafting a signal chain segment, like a notepad.

Timer event A periodic event with a sample rate and an optional offset.

Topology A hierarchy that serves as a repository for creating a signal chain. All the elements of a topology can be used in the signal chain, but not every element needs to be used. You can export each topology from and import it to a ConfigurationDesk application. Therefore a topology can be used in several applications.

A ConfigurationDesk application can contain the following topologies:

- Device topology
- Hardware topology
- Model topology

TRC file A variable description file that contains all variables (signals and parameters) which can be accessed via the experiment software. It is created during the build process.

TX Communication data that is transmitted by a bus member.

U

User code A term used in the Bus Manager context. User code is C code or C++ code containing user-specific algorithms. The algorithms provide additional functionality to the Bus Manager, such as calculating checksum values or generating authentication information for secured IPDUs. To implement user code in executable applications, you have to use functions of the Bus Custom Code interface and add the user code implementation to ConfigurationDesk. Do not confuse user code with custom code.

User code implementation A term used in the Bus Manager context. An implementation of user code in ConfigurationDesk. A user code implementation consists of one or more source files (*.c, *.cpp) and optional include files (*.h, *.hpp), such as header files.

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

VEOS The dSPACE software product for building offline simulation applications and performing SIL simulation. VEOS is a PC-based simulation platform that allows SIL simulation independently from real time. VEOS can run on Windows or Linux and consists of different components, such as the VEOS Player.

VEOS Player A component of VEOS. The VEOS Player is the graphical user interface for VEOS on Windows. For example, you can use the VEOS Player to import bus simulation containers to VEOS, integrate the respective bus communication in an offline simulation application, and load the application to VEOS.

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.

You can export the signal chain elements from a working view to a CAFX file and import them to a working view in a different ConfigurationDesk application.

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

XIL mapping file Contains the mapping between the TRC file variables of a ConfigurationDesk application component and the related test bench variables. XIL mapping files are XML files based on the ASAM XIL standard. They have the following content:

- Framework labels generated for the TRC file variables of the component.
- Testbench labels containing the paths to the related TRC file variables in the TRC file.
- The mapping between the framework labels and the testbench labels.

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.