

ConfigurationDesk

Getting Started

For ConfigurationDesk 6.7

Release 2021-A – May 2021

How to Contact dSPACE

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

How to Contact dSPACE Support

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

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

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

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

Software Updates and Patches

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

Important Notice

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

© 2012 - 2021 by:
dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Germany

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

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

Contents

About This Document	5
Available Documentation	7
Documentation Overview.....	7
Basic Concepts of ConfigurationDesk	11
Separation of I/O Functionality in ConfigurationDesk and Behavior Model	11
Application Components and Their Changeability.....	15
Creating a Logical Signal Chain	18
Providing the Physical Signal Chain	22
Modeling Executable Applications and Tasks.....	24
Concepts of Building and Downloading a Real-Time Application.....	26
Use-Case-Specific User Interface.....	29
Typical Workflows for Beginners	31
Creating a Real-Time Application: From ECU to Model	32
Creating a Real-Time Application: Starting with a Simulink Behavior Model.....	39
Creating a Multicore Real-Time Application Using Multiple Behavior Models.....	45
Creating a Multicore Real-Time Application Using One Overall Behavior Model.....	47
Creating a Multi-PU Application Using Multiple Behavior Models.....	50
Creating a Multi-PU Application Using One Overall Behavior Model.....	52
Workflow for Integrating Simulink Implementation Containers in Executable Applications.....	55
Workflow for Integrating FMUs in Executable Applications.....	57
Workflow for Integrating Bus Simulation Containers in Executable Applications.....	59
Integrating V-ECUs in Executable Applications.....	61
Index	65





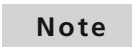



About This Document

Contents

Describes the basic concepts of ConfigurationDesk and different workflows to implement a real-time application in ConfigurationDesk. Provides an overview of the documents which are available for ConfigurationDesk and also for the SCALEXIO and MicroAutoBox III hardware.

Symbols

dSPACE user documentation uses the following symbols:

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

Naming conventions

dSPACE user documentation uses the following naming conventions:

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

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

Special folders

Some software products use the following special folders:

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

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

or

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

Documents folder A standard folder for user-specific documents.

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

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

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

Accessing dSPACE Help and PDF Files


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

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

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

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

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

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

Available 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.

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:
<http://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: <ul style="list-style-type: none"> 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	–
SCALEXIO LabBox Getting Started	Describes how to start handling with a SCALEXIO LabBox. The steps necessary for powering and	SCALEXIOLabBoxGettingStarted.pdf	–

Document	Contents	PDF File Name	Printed Version ¹⁾
	connecting are described as well as how to register a SCALEXIO LabBox to make it known to ConfigurationDesk.		
SCALEXIO Hardware Installation and Configuration	Introduces you to the SCALEXIO hardware: <ul style="list-style-type: none"> 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: <ul style="list-style-type: none"> 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. 	MicroAutoBoxIIIIHardwareInstallationAndConfiguration.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.	MicroAutoBoxIIIIHardwareAndSoftwareOverview.pdf	–
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 - Tutorial Starting with External Devices	The dSPACE software provides the <code>CfgStartingWithExternalDevices</code> 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 <code>CfgStartingWithSimulinkTutorial</code> 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 <code>CfgMicroAutoBoxIIITutorial</code> demo project. This tutorial helps you to use the demo project to	ConfigurationDeskTutorialMicroAutoBoxIII.pdf	X

Document	Contents	PDF File Name	Printed Version ¹⁾
	learn the basic steps in ConfigurationDesk when working with a MicroAutoBox III.		
ConfigurationDesk - Real-Time Implementation Guide	Introduces you to ConfigurationDesk and shows how to use it, for example: <ul style="list-style-type: none"> ▪ 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	-
ConfigurationDesk - Bus Manager Implementation Guide	Provides basic information and instructions on implementing bus communication via ConfigurationDesk's Bus Manager.	ConfigurationDeskBusManagerImplementationGuide.pdf	–
ConfigurationDesk - Demo Projects	Provides descriptions of the ConfigurationDesk demo projects or refers you to detailed descriptions in other documents.	ConfigurationDeskDemoProjects.pdf	–
ConfigurationDesk Glossary	Explains the most important expressions and naming conventions used in the ConfigurationDesk documentation.	ConfigurationDeskGlossary.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 - API Reference	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.	BusCustomCodeInterface.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	–

Document	Contents	PDF File Name	Printed Version ¹⁾
ConfigurationDesk - Hardware Resource Properties	Provides detailed information on the characteristics of the hardware resources that are supported by ConfigurationDesk.	ConfigurationDeskHardwareResourceProperties.pdf	–
ConfigurationDesk - Conflicts	Describes 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	X
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.

Basic Concepts of ConfigurationDesk

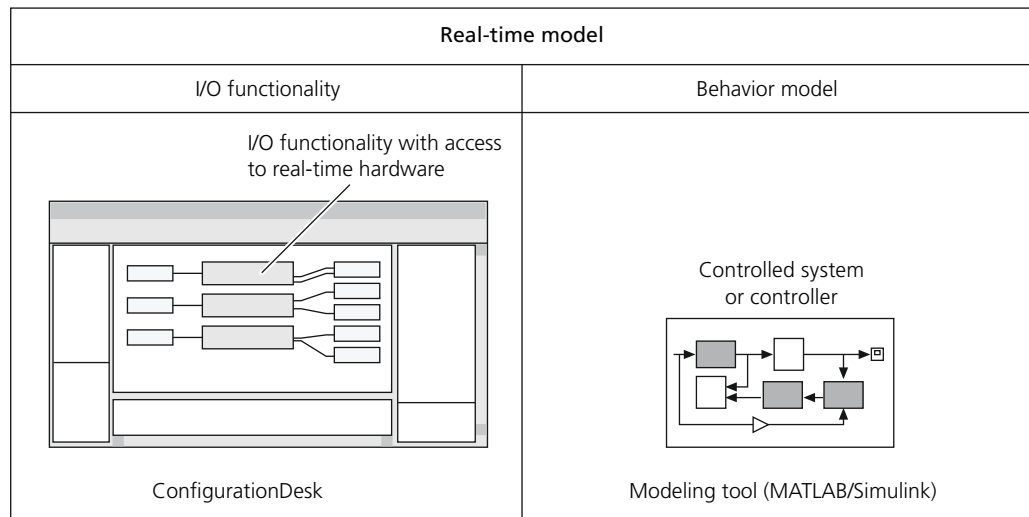
Objective	Knowing the basic concepts of ConfigurationDesk helps you to understand how to work with it.
------------------	--

Where to go from here	Information in this section
------------------------------	------------------------------------

Separation of I/O Functionality in ConfigurationDesk and Behavior Model	11
Application Components and Their Changeability.....	15
Creating a Logical Signal Chain	18
Providing the Physical Signal Chain	22
Modeling Executable Applications and Tasks.....	24
Concepts of Building and Downloading a Real-Time Application... ..	26
Use-Case-Specific User Interface.....	29

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.

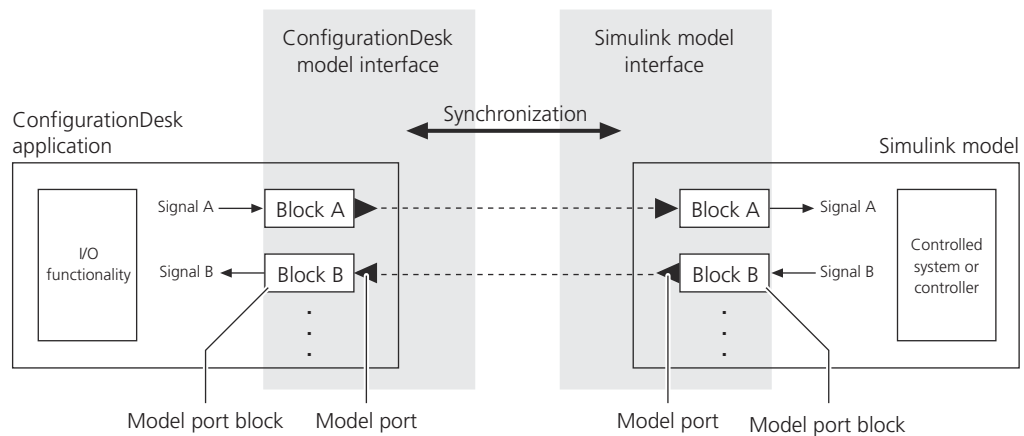
Implementing bus communication Bus communication (CAN, LIN, FlexRay) is an exception. This can be modeled in the behavior model via specific RTI blocksets (for example, the RTI CAN MultiMessage Blockset and RTI LIN MultiMessage Blockset). Then only access to the real-time hardware is added in ConfigurationDesk.

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.

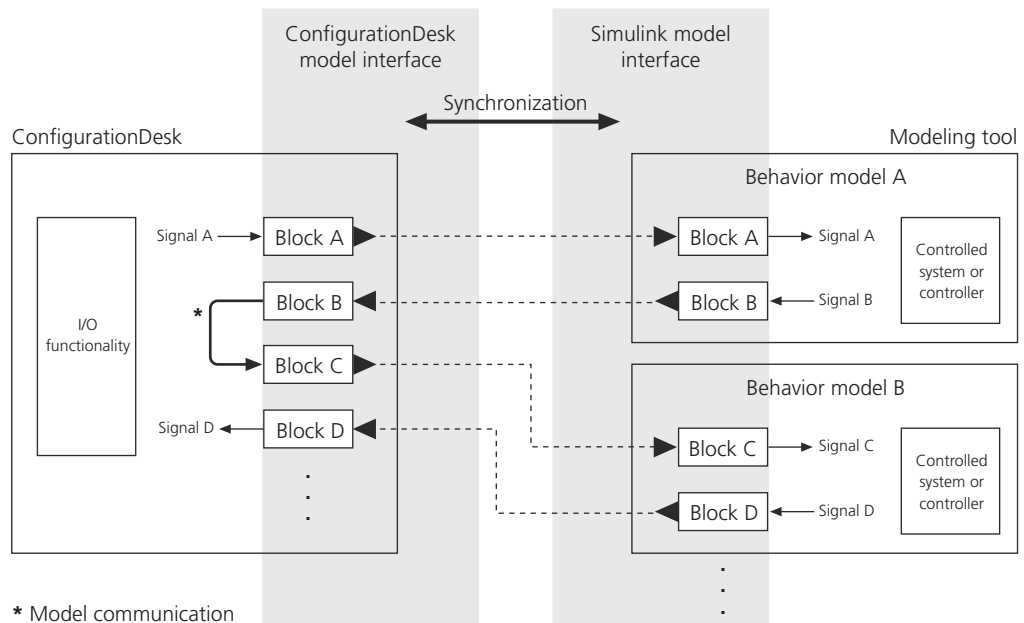
The model port blocks can be created in ConfigurationDesk or in the behavior model. You can synchronize the model interfaces from within ConfigurationDesk to make the blocks available in both models.



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). As 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 simplify 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.

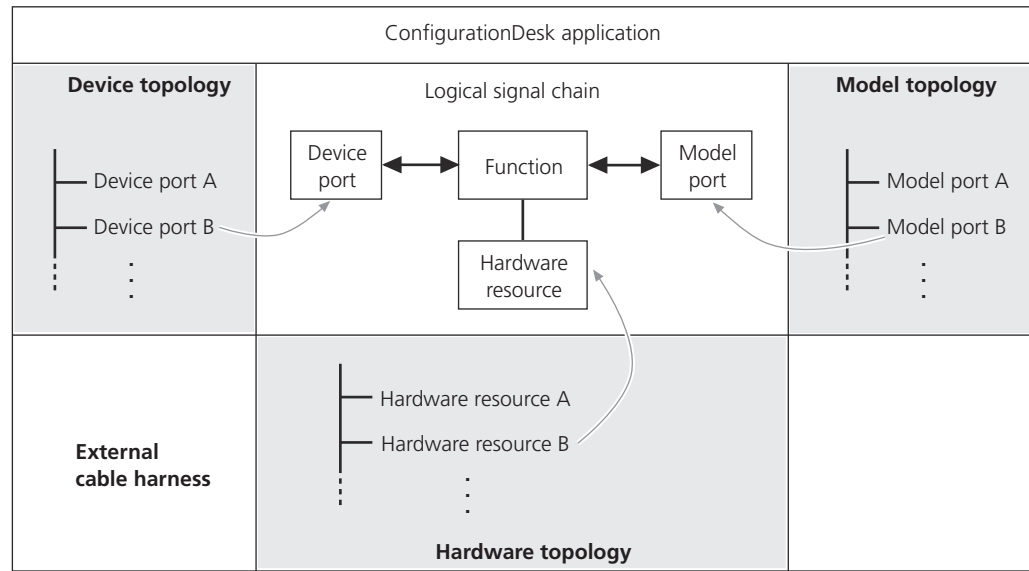
- **V-ECU implementation container files (VECU files)**

With ConfigurationDesk, you can integrate virtual ECUs (V-ECUs) in executable applications (real-time applications). These V-ECUs can communicate with real ECUs via CAN bus. Thus, an ECU network consisting of real and virtual ECUs can be used for simulations and tests. V-ECU implementations can be provided by TargetLink or SystemDesk.

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.

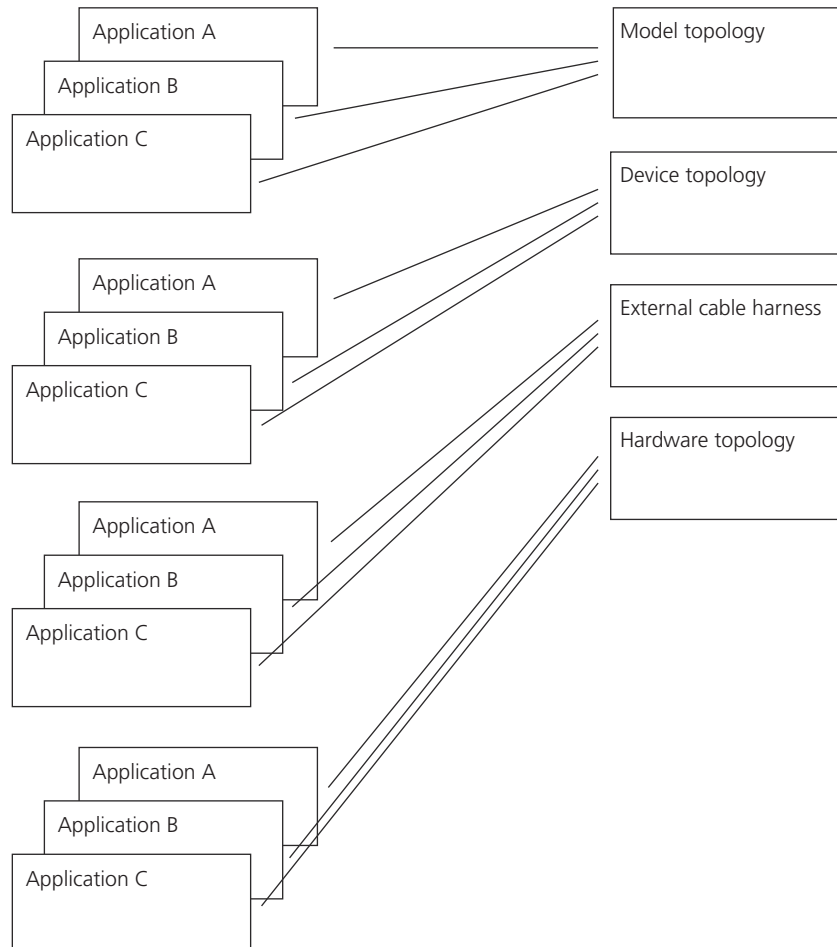
External cable harness Contains the wiring information for the external cable harness. The external cable harness is the connection between the I/O connectors of the real-time hardware and the devices, such as the ECUs to be tested.

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.

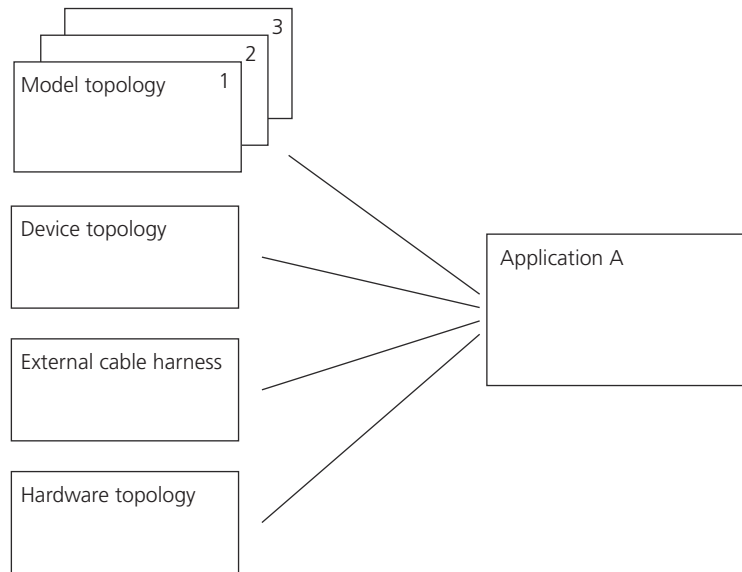
Interchangeability of components

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.

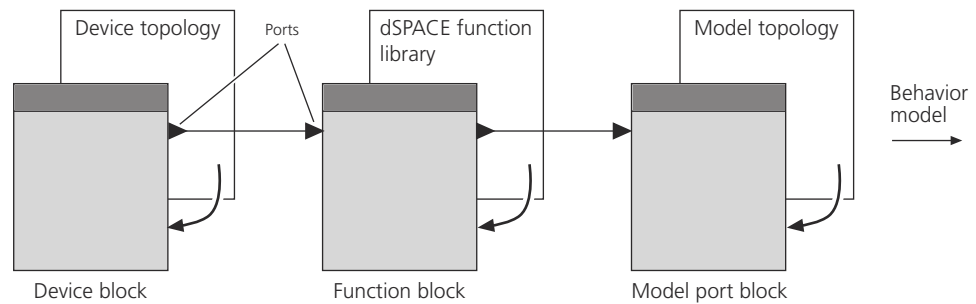
Logical signal chain

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 22.

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.

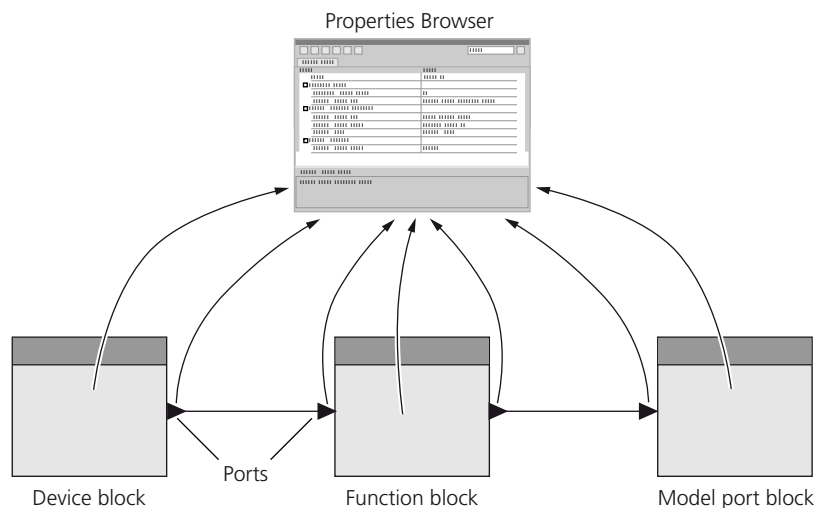
Model port block Each model port block is an element of the model topology of the active application. Model port blocks in ConfigurationDesk are the interface to the corresponding model port blocks in the behavior model. They provide the data flow between the function blocks and the behavior model. In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



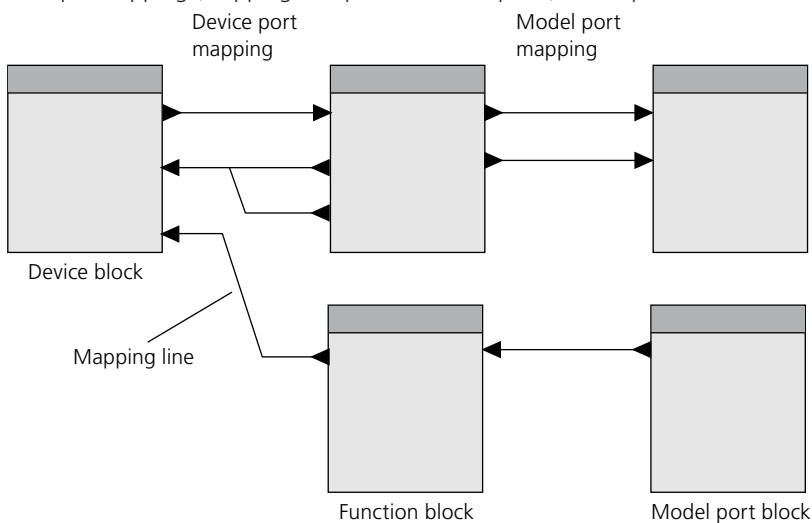
- Via tables

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

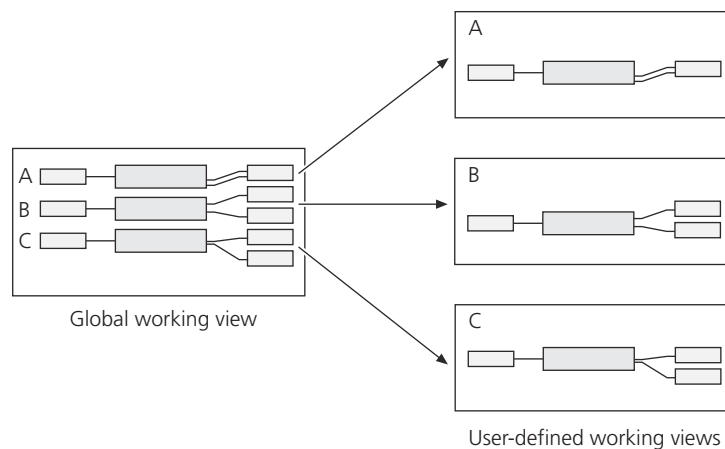
ConfigurationDesk determines the state of each element in the signal chain after every configuration change. States are displayed graphically and in tooltips. You can move the mouse cursor over an element to see its tooltip containing error and warning messages, etc.

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

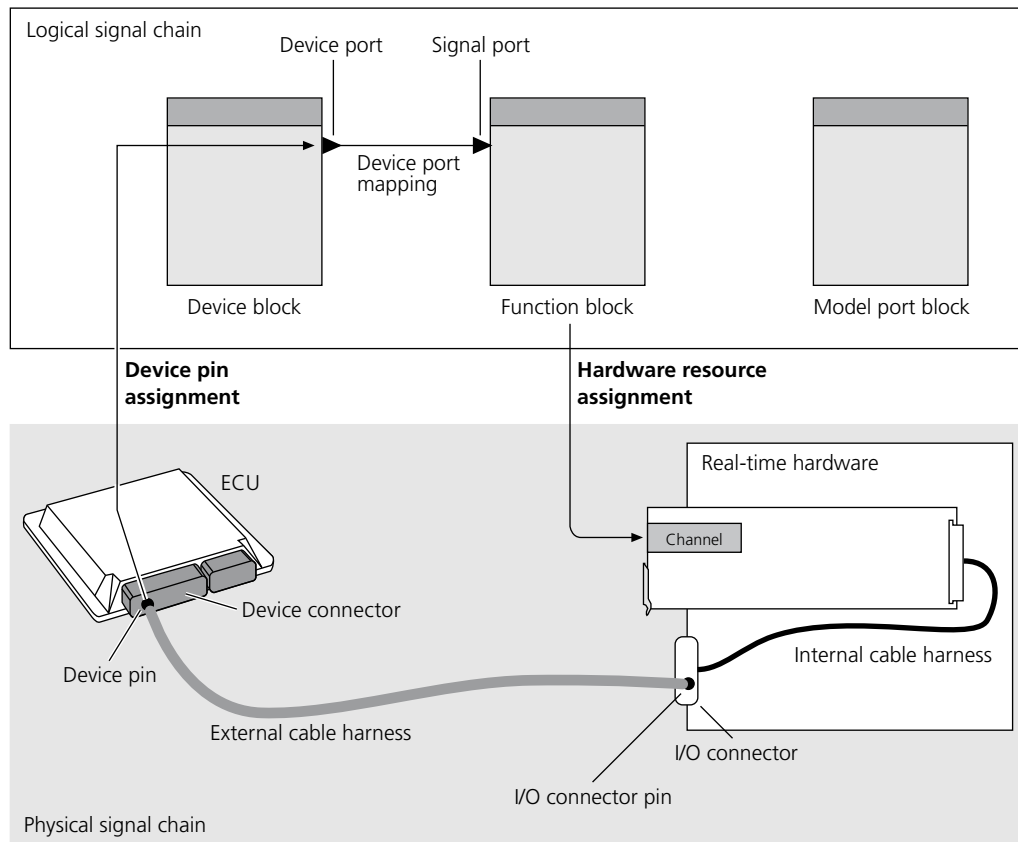
The external cable harness is the connection between the I/O connectors of the real-time hardware and the devices, such as the ECUs to be tested.

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 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:



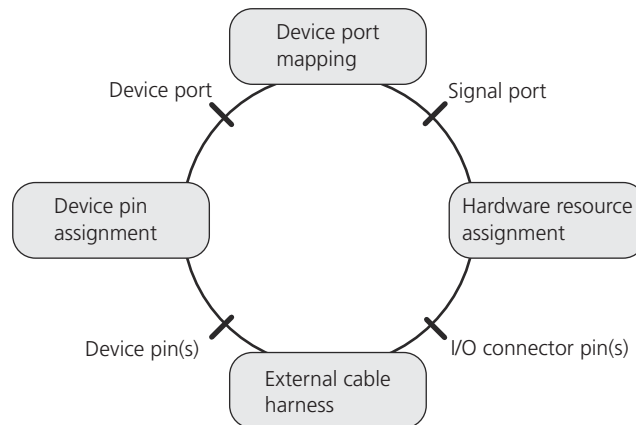
As shown above, model port blocks (= model interface) are not required and thus ignored in determining the physical signal chain.

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

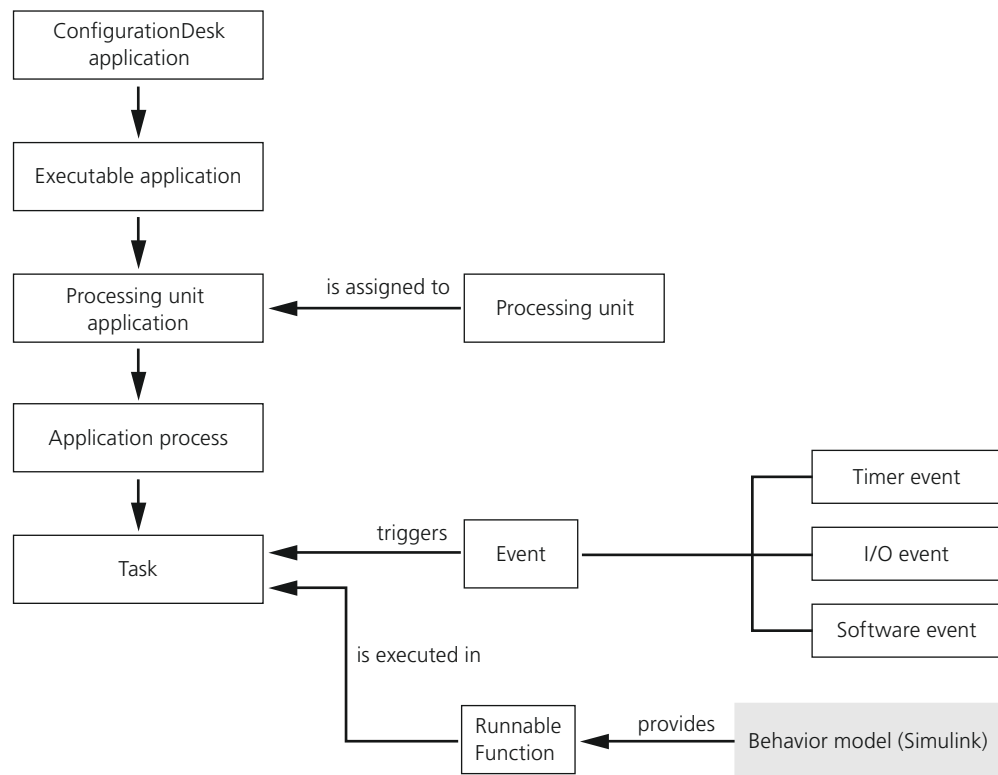
Objective

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.

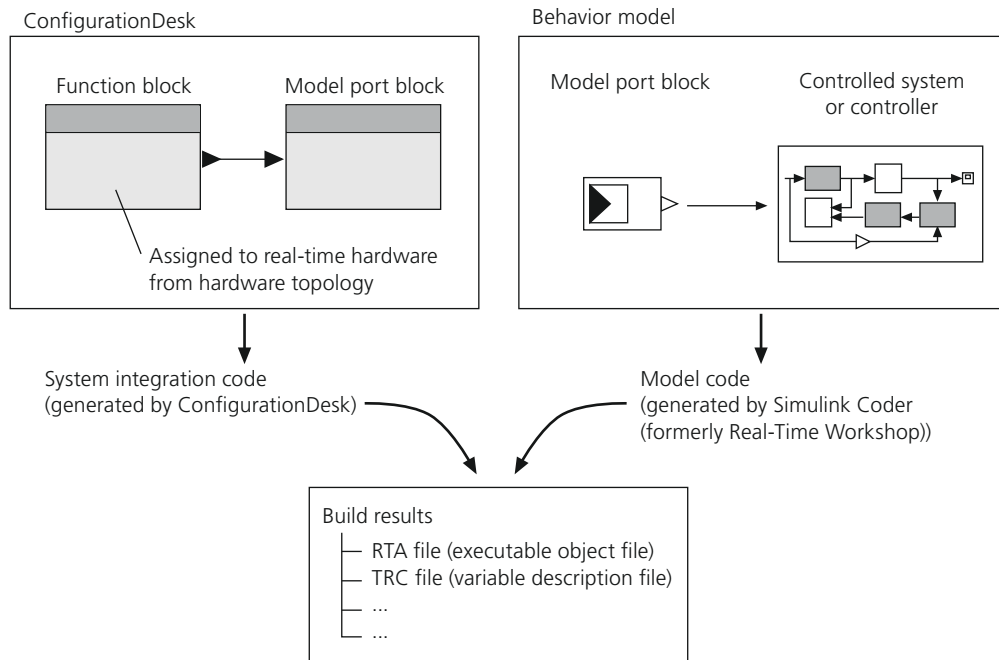
Software event Software events are specified in the behavior model. They are available in ConfigurationDesk after model analysis.

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 virtual ECUs (V-ECUs), 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. You cannot change this assignment.

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

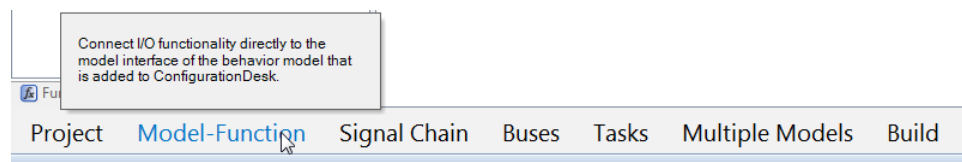
You can build a multi-PU application and download it to the dSPACE real-time hardware, where different processing unit applications (each containing at least one application process with one behavior model) are executed in parallel on several SCALEXIO Processing Units. The processing units are connected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

Use-Case-Specific User Interface

Different view sets for specific purposes

The user interface of ConfigurationDesk offers different view sets for specific purposes. You can switch between view sets by using the navigation bar as shown below:

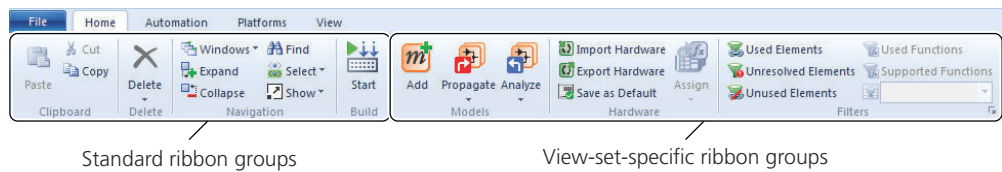


The available panes of each view set serve a specific purpose. For example, the Project view set contains the panes required to perform project and application management tasks.

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.

For each view set, the Home ribbon contains specific commands suitable for the purpose of the view set.

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



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.

Typical Workflows for Beginners

Where to go from here

Information in this section

Creating a Real-Time Application: From ECU to Model	32
Creating a Real-Time Application: Starting with a Simulink Behavior Model.....	39
Creating a Multicore Real-Time Application Using Multiple Behavior Models.....	45
Creating a Multicore Real-Time Application Using One Overall Behavior Model.....	47
Creating a Multi-PU Application Using Multiple Behavior Models.....	50
Creating a Multi-PU Application Using One Overall Behavior Model.....	52
Workflow for Integrating Simulink Implementation Containers in Executable Applications.....	55
Workflow for Integrating FMUs in Executable Applications.....	57
Workflow for Integrating Bus Simulation Containers in Executable Applications.....	59
Integrating V-ECUs in Executable Applications.....	61

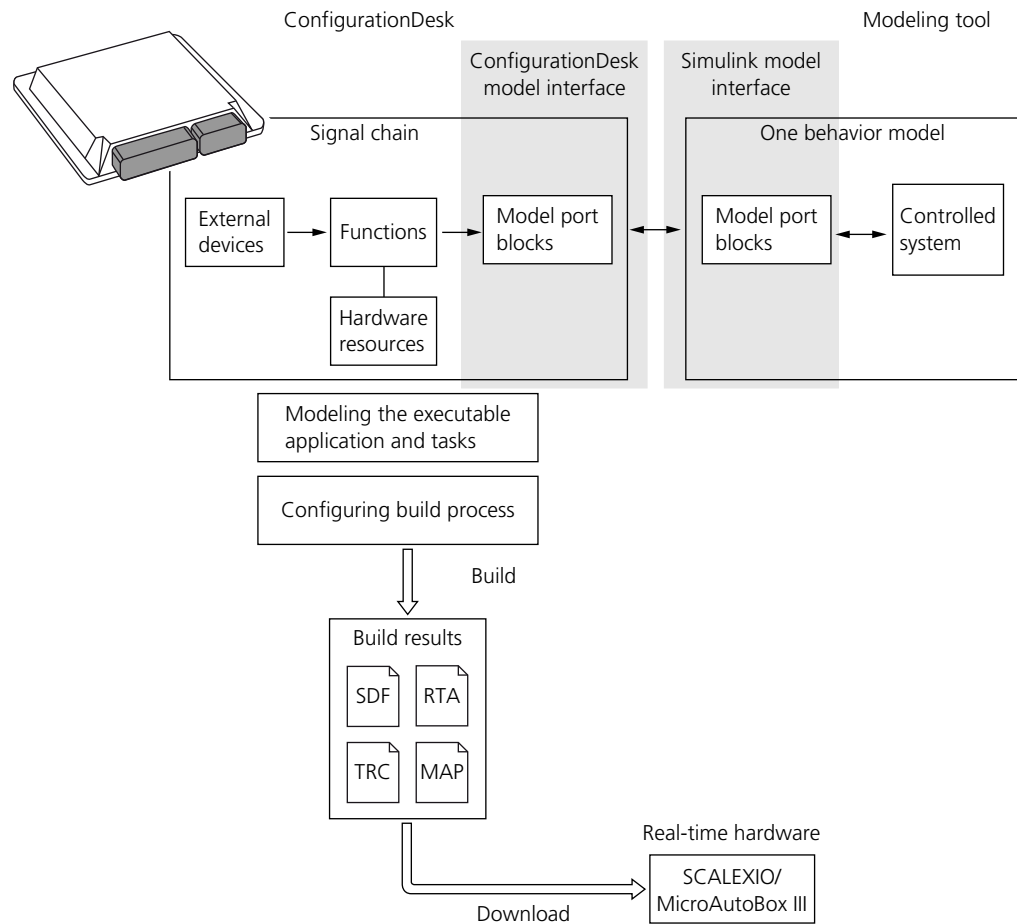
Creating a Real-Time Application: From ECU to Model

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.

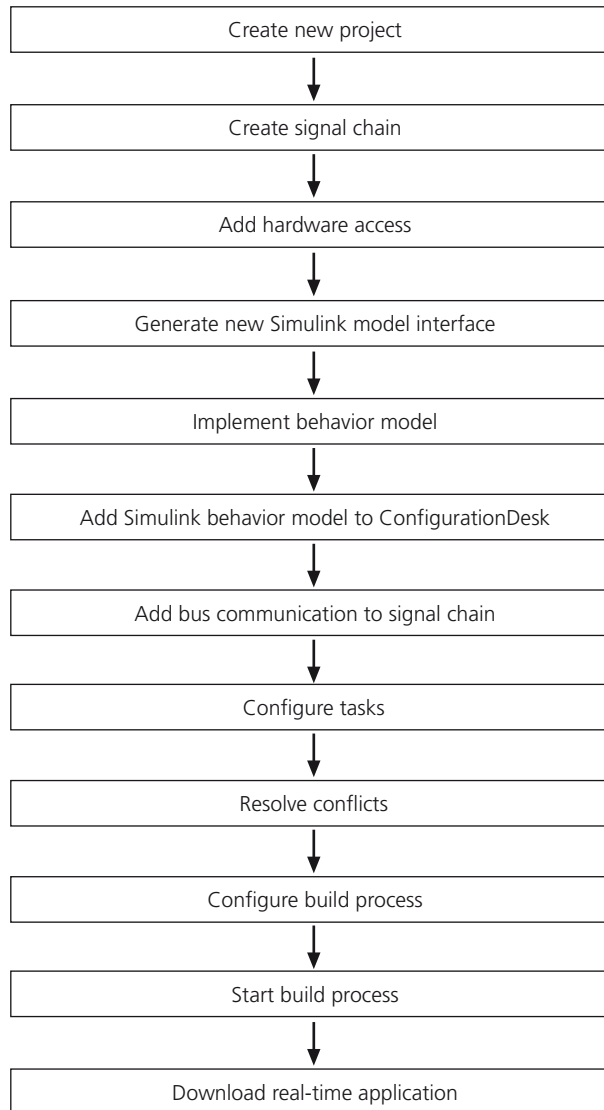
Overview

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



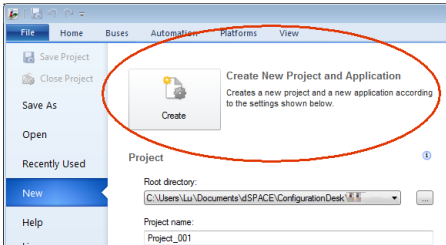
Workflow


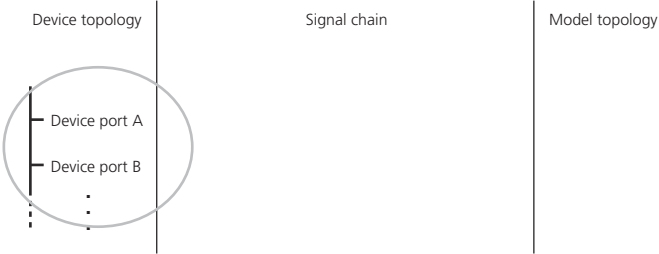

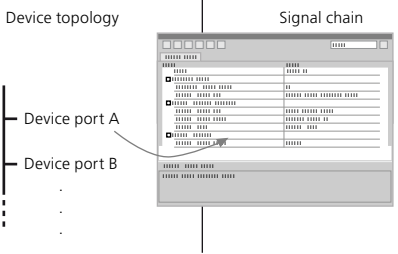

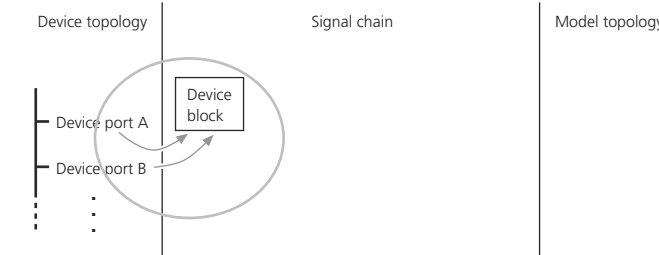

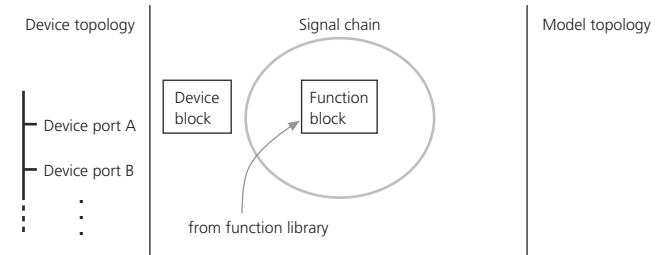
The workflow shows the steps of a standard workflow for implementing a ConfigurationDesk application with one behavior model.

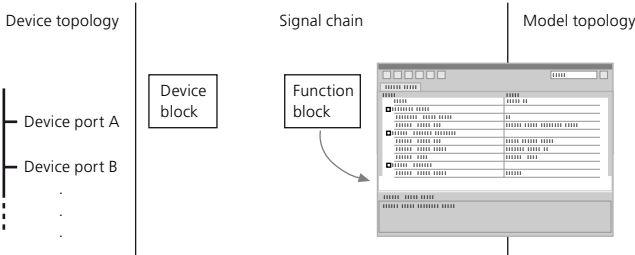

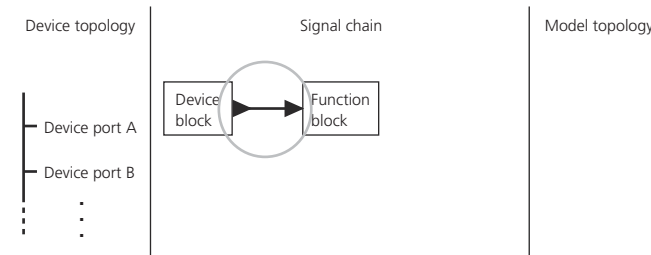

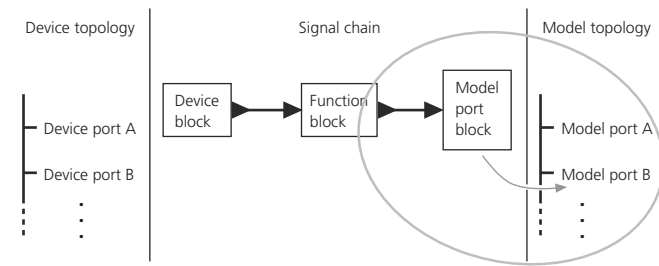

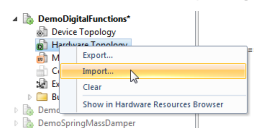
**Detailed steps**

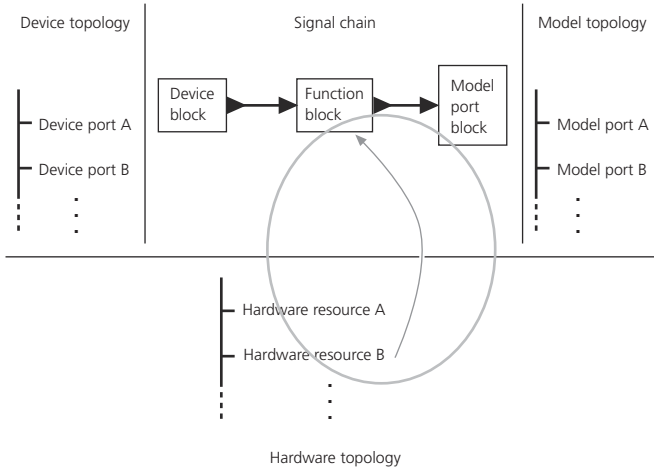
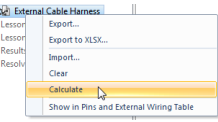
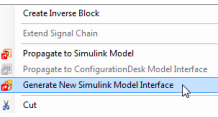
The following table shows detailed workflow steps and indicates the corresponding chapters, which give you detailed information (basic information, instructions etc.).

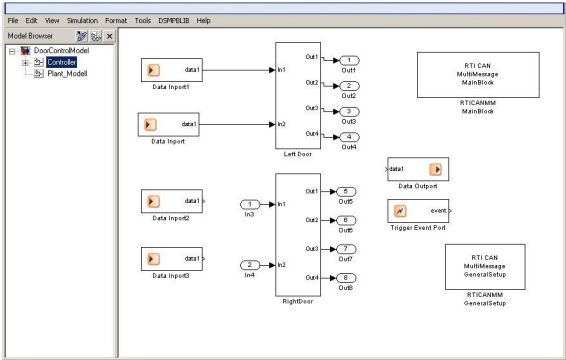
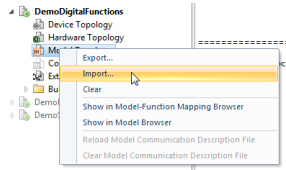

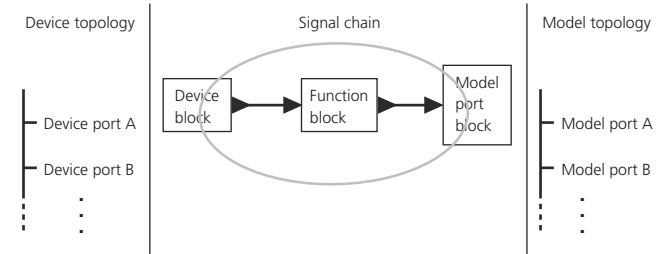

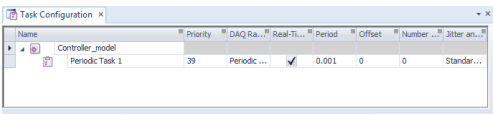

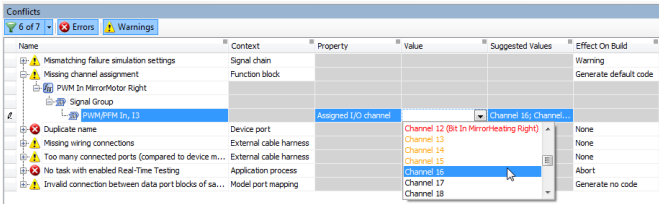

Step	Work	Refer to ...
1	Create a new project and a ConfigurationDesk application.	Managing ConfigurationDesk Projects and Applications (ConfigurationDesk Real-

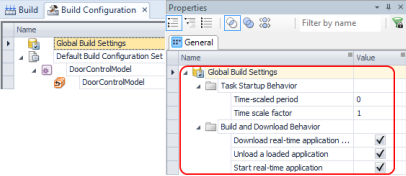

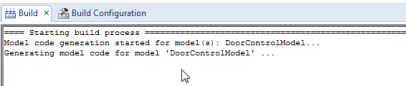

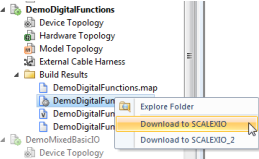

Step	Work	Refer to ...
		Time Implementation Guide (📖)

Step	Work	Refer to ...
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	<p>Configure the function block properties.</p> 	Configuring Function Blocks (ConfigurationDesk Real-Time Implementation Guide )
6	<p>Map device blocks to function blocks (= device port mapping).</p> 	Device Port Mapping (ConfigurationDesk Real-Time Implementation Guide )
7	<p>Add model port blocks to the signal chain.</p> 	Adding Model Port Blocks to the Signal Chain (ConfigurationDesk Real-Time Implementation Guide )
3	<p>Add hardware access.</p> <p>1 Add a hardware topology to the ConfigurationDesk application.</p> 	How to Import a Hardware Topology (ConfigurationDesk Real-

Step	Work	Refer to ...
		Time Implementation Guide (📖)
2	<p>Assign hardware resources to the function blocks.</p> 	Assigning Hardware Resources to Function Blocks (ConfigurationDesk Real-Time Implementation Guide) (📖)
3	<p>Calculate wiring information for the cable harness and export it for external usage.</p> 	How to (Re)Calculate the External Cable Harness (ConfigurationDesk Real-Time Implementation Guide) (📖)
4	<p>Generate the Simulink model interface and implement the behavior model.</p>	
1	<p>Generate new Simulink model interface.</p> 	How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model (ConfigurationDesk Real-Time Implementation Guide) (📖)
2	<p>Implement the behavior model:</p> <ul style="list-style-type: none"> 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	<p>Add the behavior model to the ConfigurationDesk application.</p> 	How to Import a Model Topology (ConfigurationDesk Real-Time Implementation Guide )
5	<p>Add bus communication to the signal chain (if necessary).</p> <p>1 Add specific function blocks for bus communication (CAN, LIN) to the signal chain.</p> <p>2 Map these function blocks to device blocks and specific model port blocks (= Configuration Port blocks).</p> 	Adding Bus and Gigalink Communication to the Signal Chain (ConfigurationDesk Real-Time Implementation Guide )
6	<p>Configure tasks.</p> 	Configuring Tasks in ConfigurationDesk (ConfigurationDesk Real-Time Implementation Guide )
7	<p>Resolve conflicts via the Conflicts Viewer.</p> 	Resolving Conflicts (ConfigurationDesk Real-Time Implementation Guide )

Step	Work	Refer to ...
8	<p>Configure and start the build process to build the real-time application.</p> <ol style="list-style-type: none"> Configure the build process. 	<p>Configuring the Build Process (ConfigurationDesk Real-Time Implementation Guide )</p>
	<ol style="list-style-type: none"> Start the build process. 	<p>Starting the Build Process (ConfigurationDesk Real-Time Implementation Guide )</p>
9	<p>Download the real-time application to a platform.</p> 	<p>Downloading and Executing Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide )</p>

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

Objective

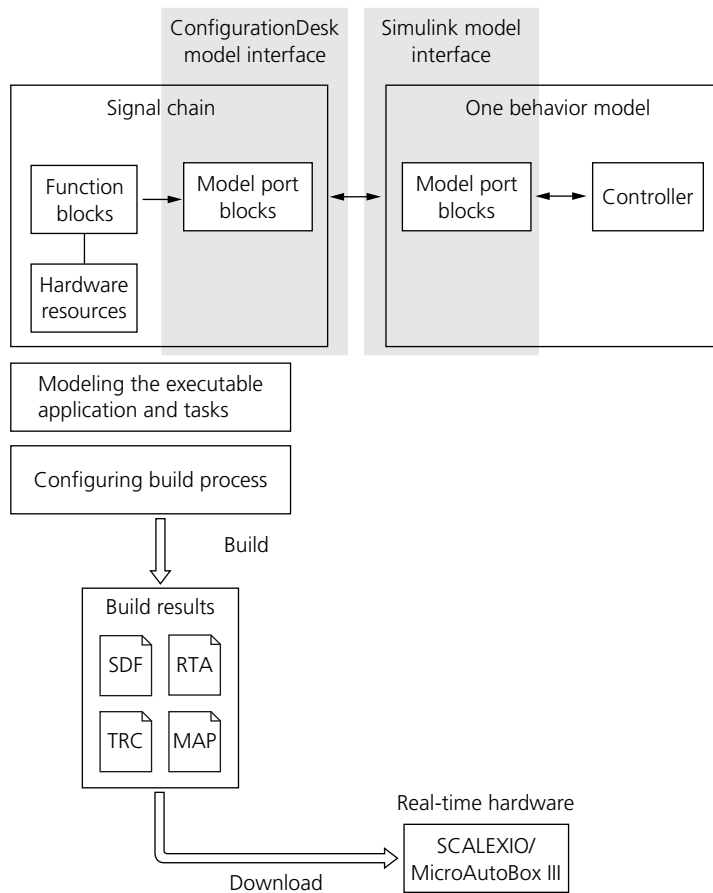
"Starting with a Simulink behavior model" is a workflow whose starting point is one behavior model in Simulink. This workflow is typically used in implementing real-time applications for rapid prototyping scenarios.

Overview

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

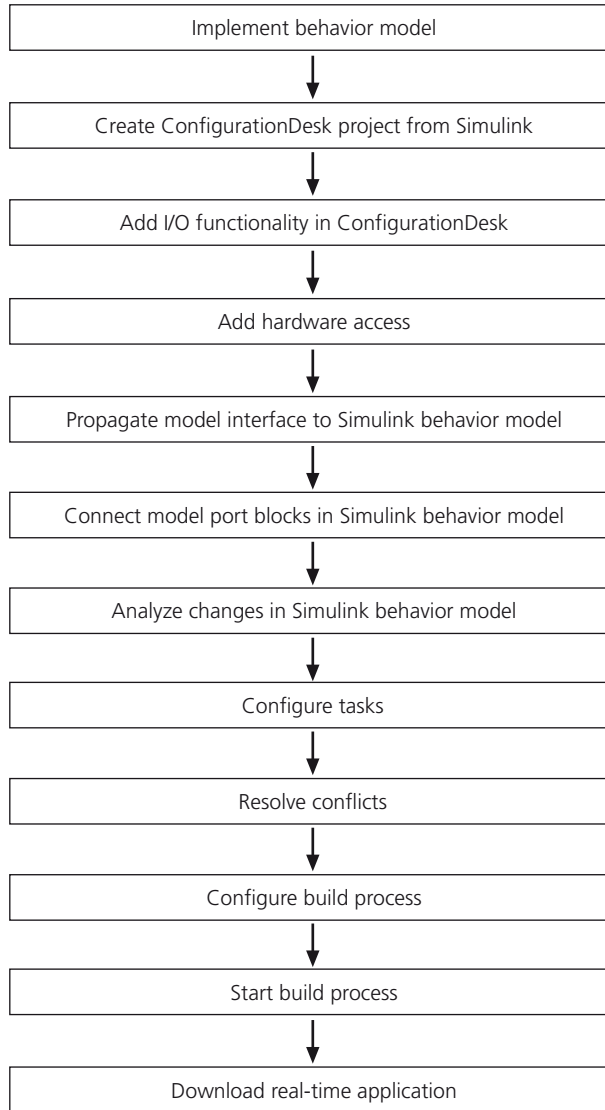
ConfigurationDesk

Modeling tool




Workflow

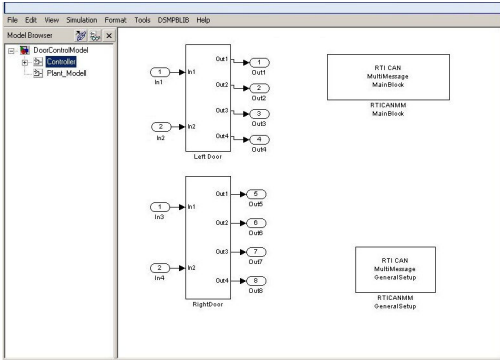
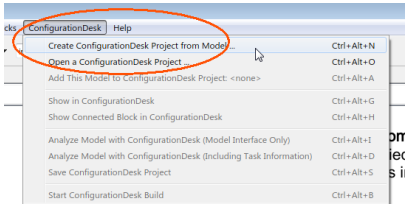
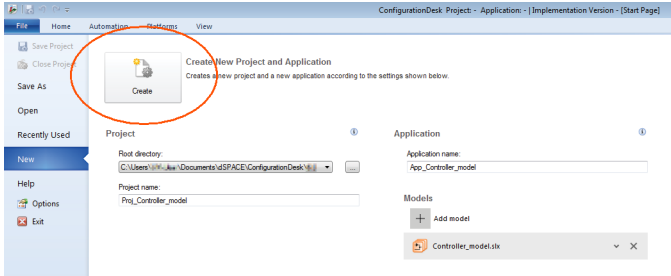


The workflow shows the steps of a standard workflow for implementing a ConfigurationDesk application with one behavior model.

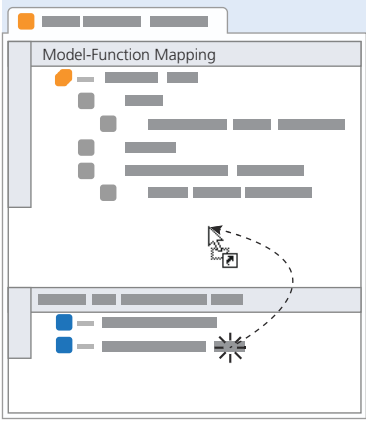
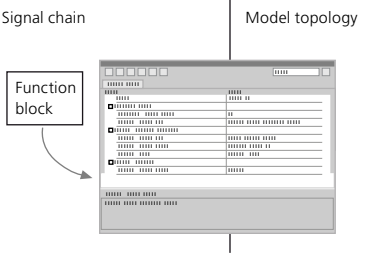


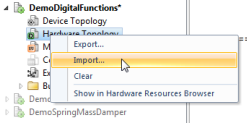


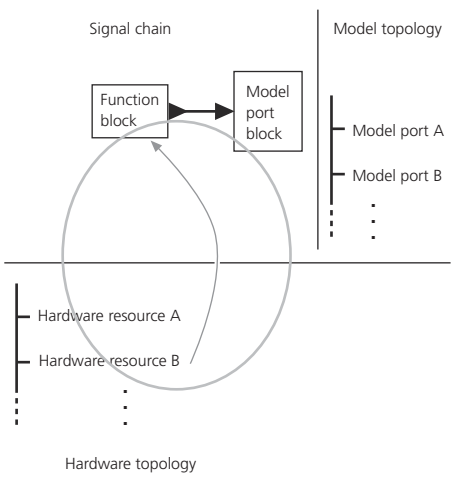
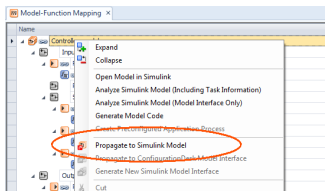
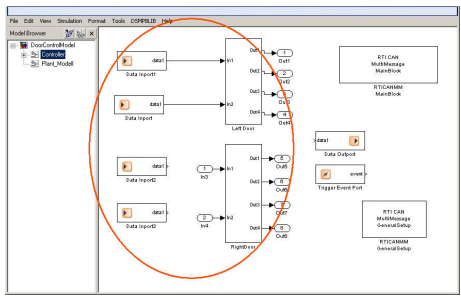
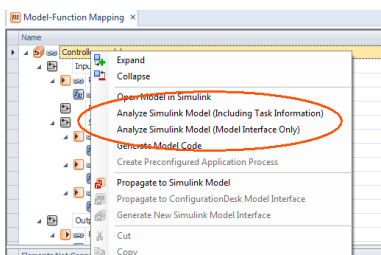
Detailed steps

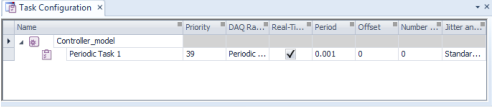

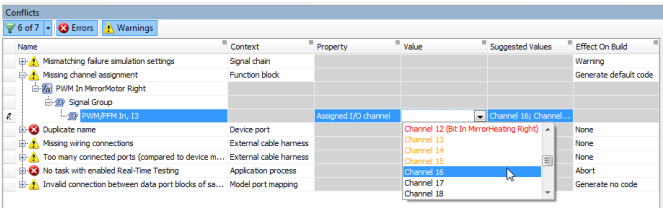

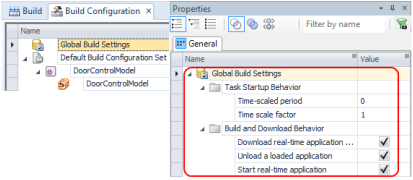
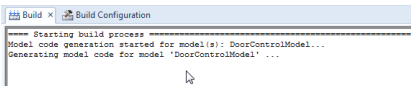


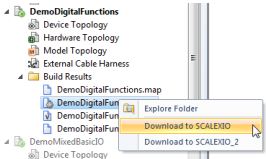

The following table shows detailed workflow steps and indicates the corresponding chapters, which give you detailed information (basic information, instructions etc.).

Step	Work	Refer to ...
1	Implement the behavior model: <ul style="list-style-type: none"> 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	<p>Create a new ConfigurationDesk project.</p> <p>1 Create ConfigurationDesk project from the Simulink behavior model.</p>  <p>2 Add the behavior model to the ConfigurationDesk application.</p> 	<p>Remote Access to ConfigurationDesk (Model Interface Package for Simulink - Modeling Guide )</p> <p>Managing ConfigurationDesk Projects and Applications (ConfigurationDesk Real-Time Implementation Guide )</p>

Step	Work	Refer to ...
3	<p>Add I/O functionality to the ConfigurationDesk application.</p> <p>1 Create signal chain using the Model-Function Mapping Browser.</p>  <p>2 Configure the function block properties.</p> 	<p>Creating Signal Chains via the Model-Function Mapping Browser (ConfigurationDesk Real-Time Implementation Guide )</p> <p>Configuring Function Blocks (ConfigurationDesk Real-Time Implementation Guide )</p>
4	<p>Add hardware access.</p> <p>1 Add a hardware topology to the ConfigurationDesk application.</p> 	<p>How to Import a Hardware Topology (ConfigurationDesk Real-</p>

Step	Work	Refer to ...
		Time Implementation Guide (📖)
2	<p>Assign hardware resources to the function blocks.</p> 	Assigning Hardware Resources to Function Blocks (ConfigurationDesk Real-Time Implementation Guide) (📖)
5	<p>Propagate changes from ConfigurationDesk to the Simulink behavior model.</p> 	Propagating Changes in the ConfigurationDesk Model Interface Directly to a Simulink Model (ConfigurationDesk Real-Time Implementation Guide) (📖)
6	<p>Connect new model port blocks in Simulink behavior model.</p> 	Introduction to the Model Interface Package for Simulink (Model Interface Package for Simulink - Modeling Guide) (📖)
7	<p>Analyze changes in Simulink behavior model from ConfigurationDesk.</p> 	Analyzing Simulink Behavior Models (ConfigurationDesk Real-Time Implementation Guide) (📖)

Step	Work	Refer to ...
8	<p>Configure tasks.</p> 	<p>Configuring Tasks in ConfigurationDesk (ConfigurationDesk Real-Time Implementation Guide )</p>
9	<p>Resolve conflicts via the Conflicts Viewer.</p> 	<p>Resolving Conflicts (ConfigurationDesk Real-Time Implementation Guide )</p>
10	<p>Configure and start the build process to build the real-time application.</p> <div> <div> <p>1 Configure the build process.</p>  </div> <div> <p>2 Start the build process.</p>  </div> </div>	<p>Configuring the Build Process (ConfigurationDesk Real-Time Implementation Guide )</p> <p>Starting the Build Process (ConfigurationDesk Real-Time Implementation Guide )</p>
11	<p>Download the real-time application to a platform.</p> 	<p>Downloading and Executing Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide )</p>

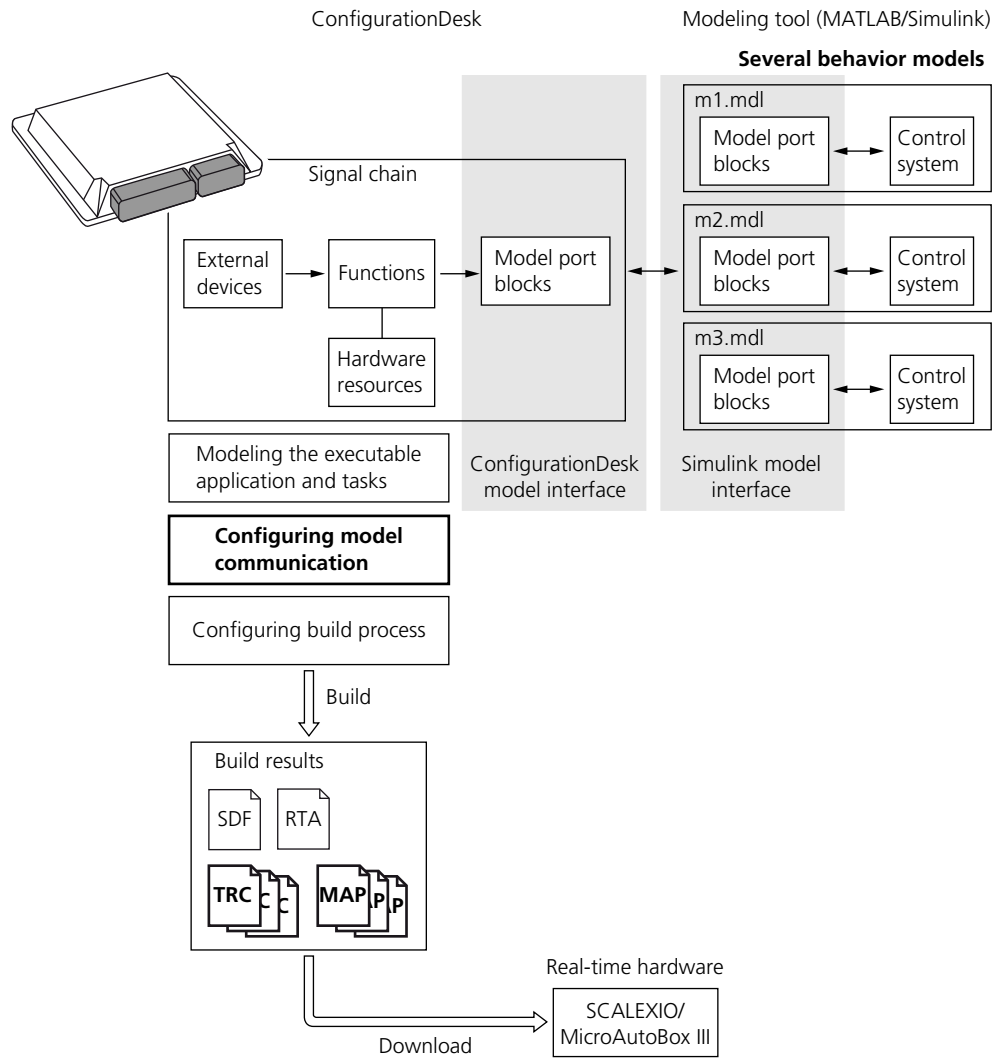
Creating a Multicore Real-Time Application Using Multiple Behavior Models

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.

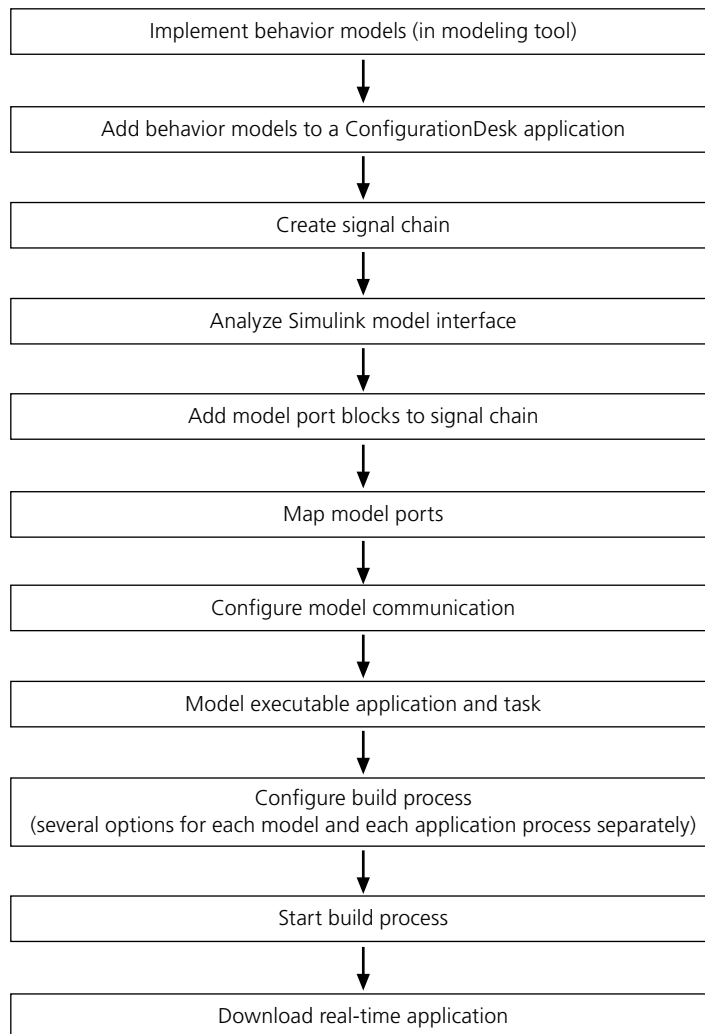
Overview

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

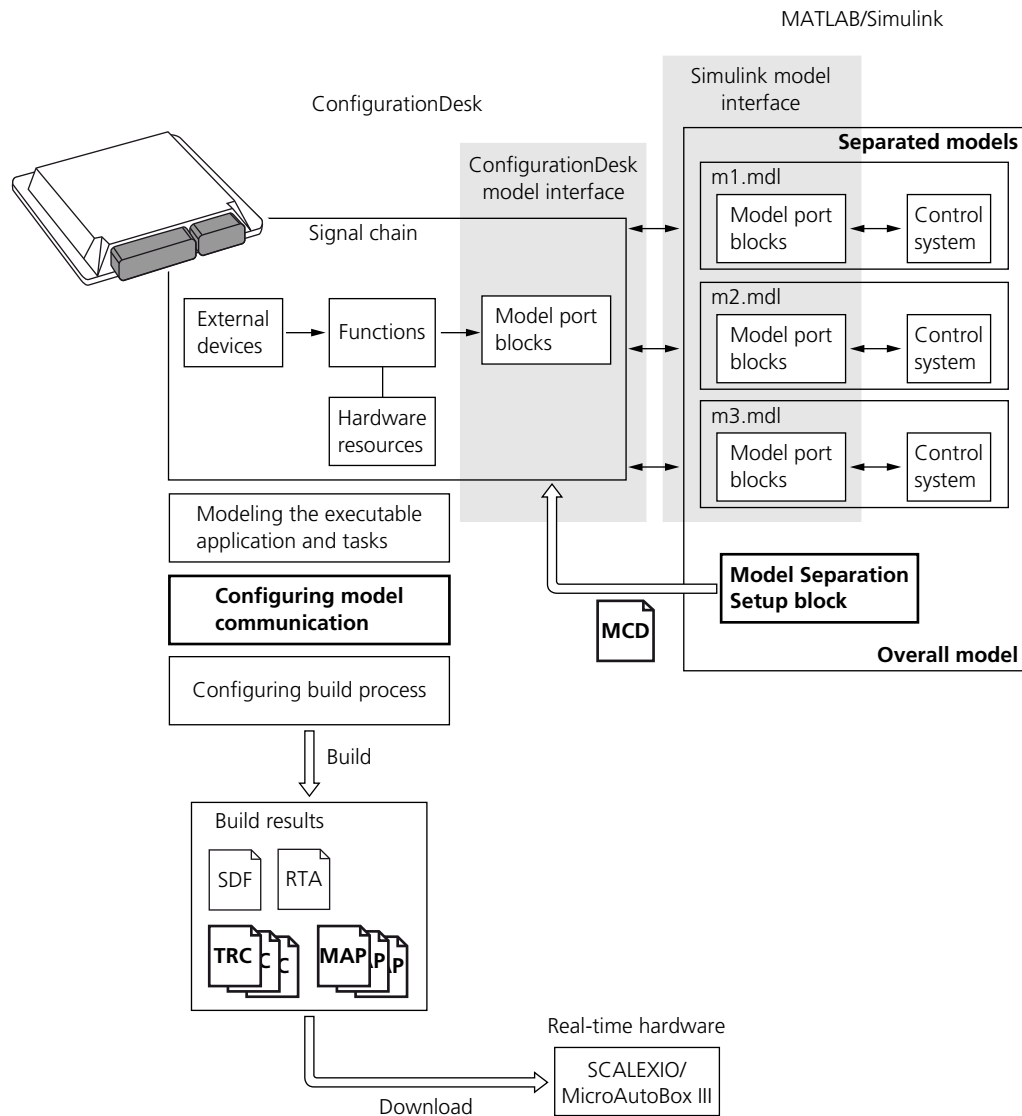
Use scenario

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

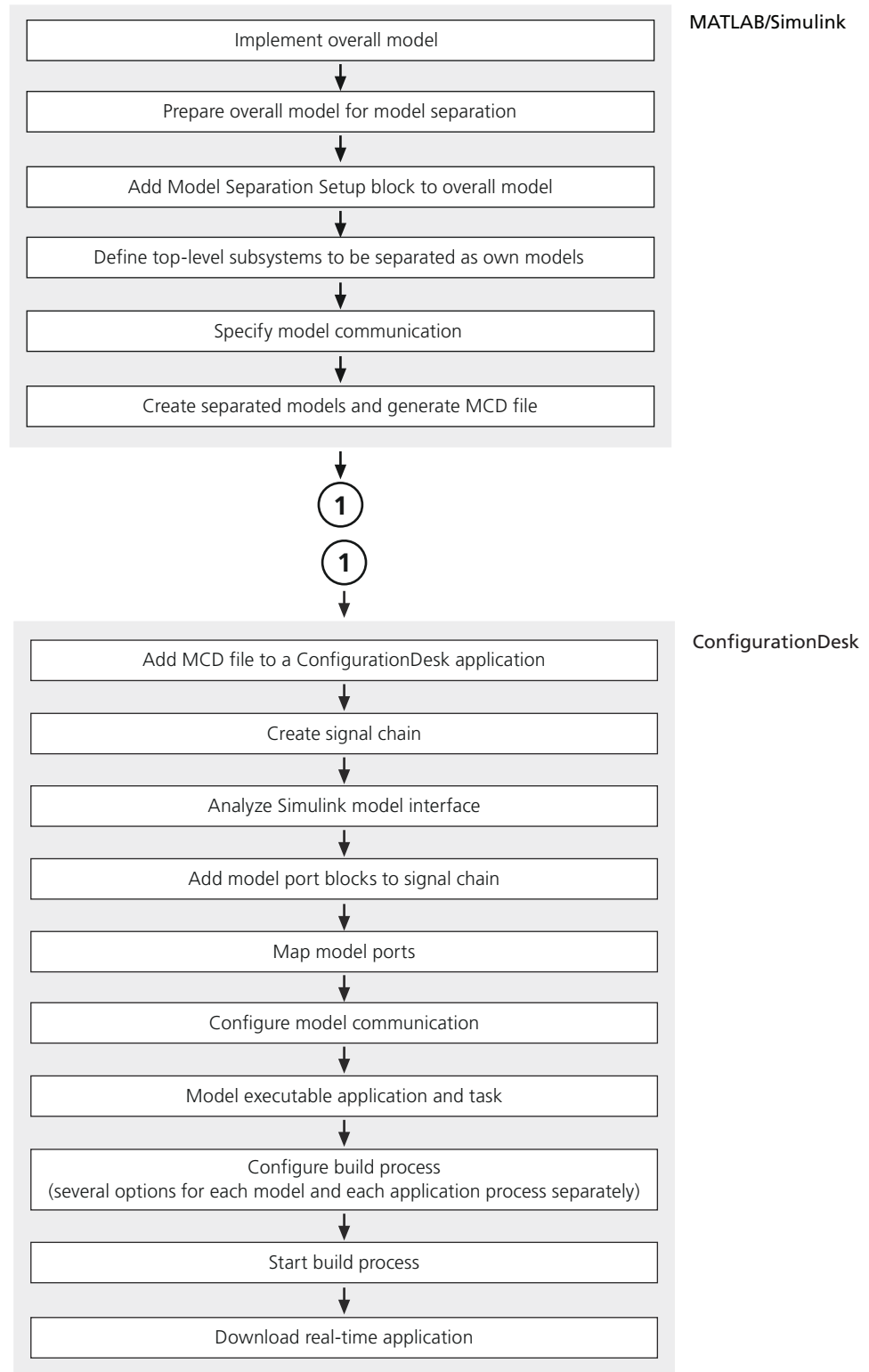
The following illustration gives you an overview of the use scenario. Parts that are specific to this scenario are in bold letters.



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.



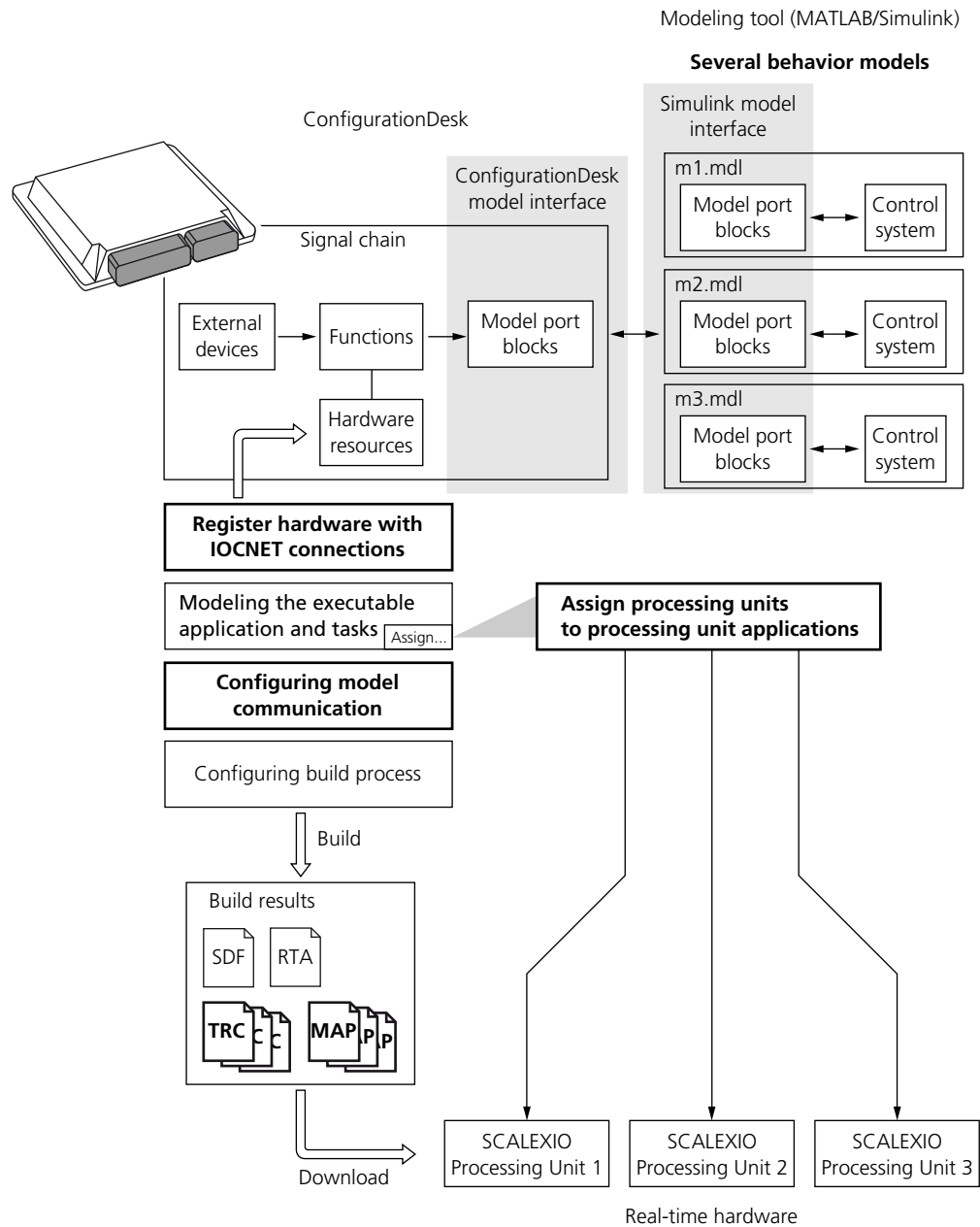
Creating a Multi-PU Application Using Multiple Behavior Models

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.

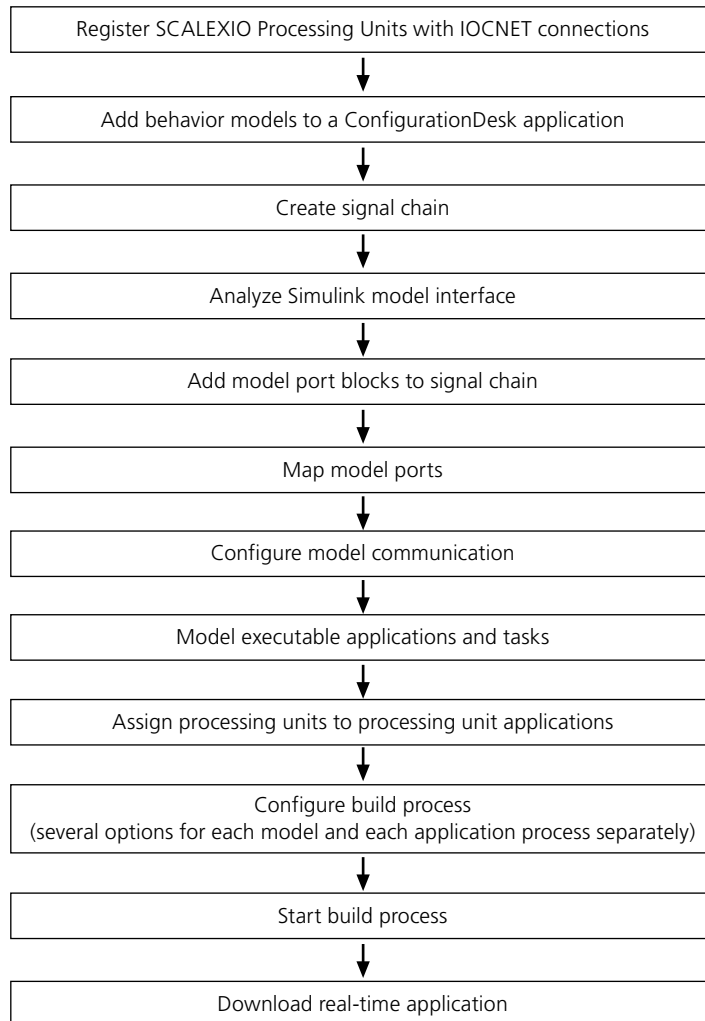
Overview

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

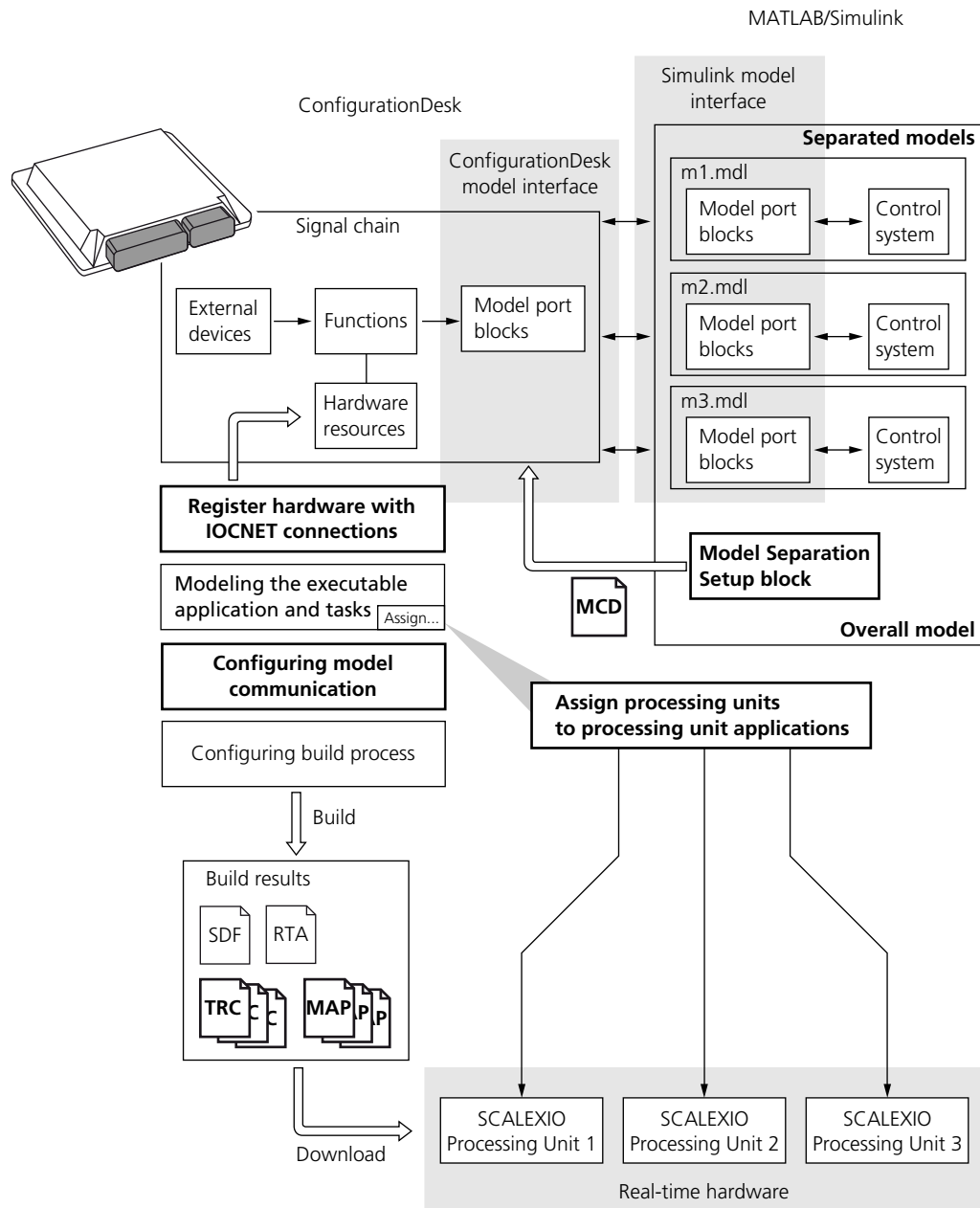
Use scenario

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

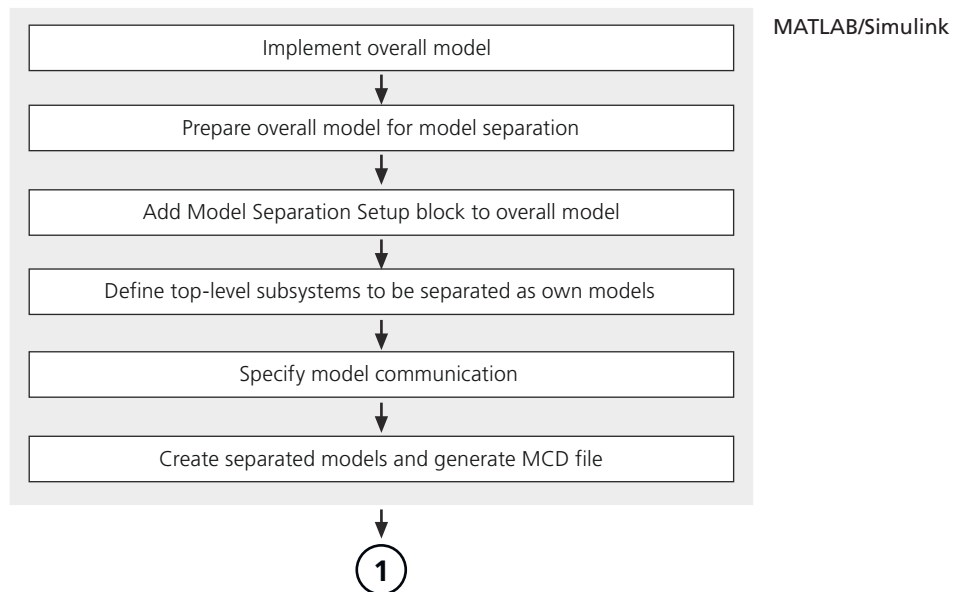
The following illustration gives you an overview of the use scenario. Parts that are specific to this scenario are in bold letters.

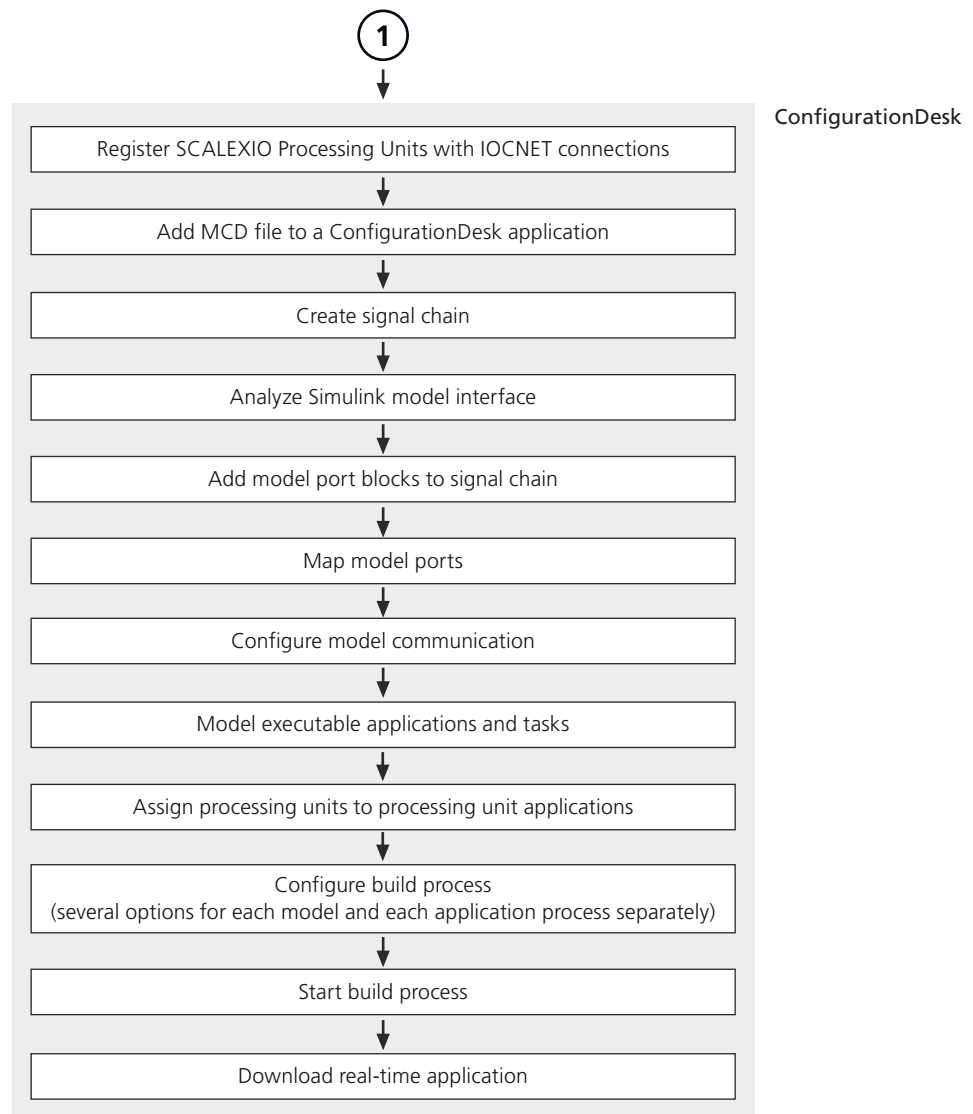


To separate models from an overall model in MATLAB/Simulink, you can use the **Model Separation Setup block**. This also generates an MCD file, which contains information on the separated models and their interconnections in the overall model.

Workflow

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





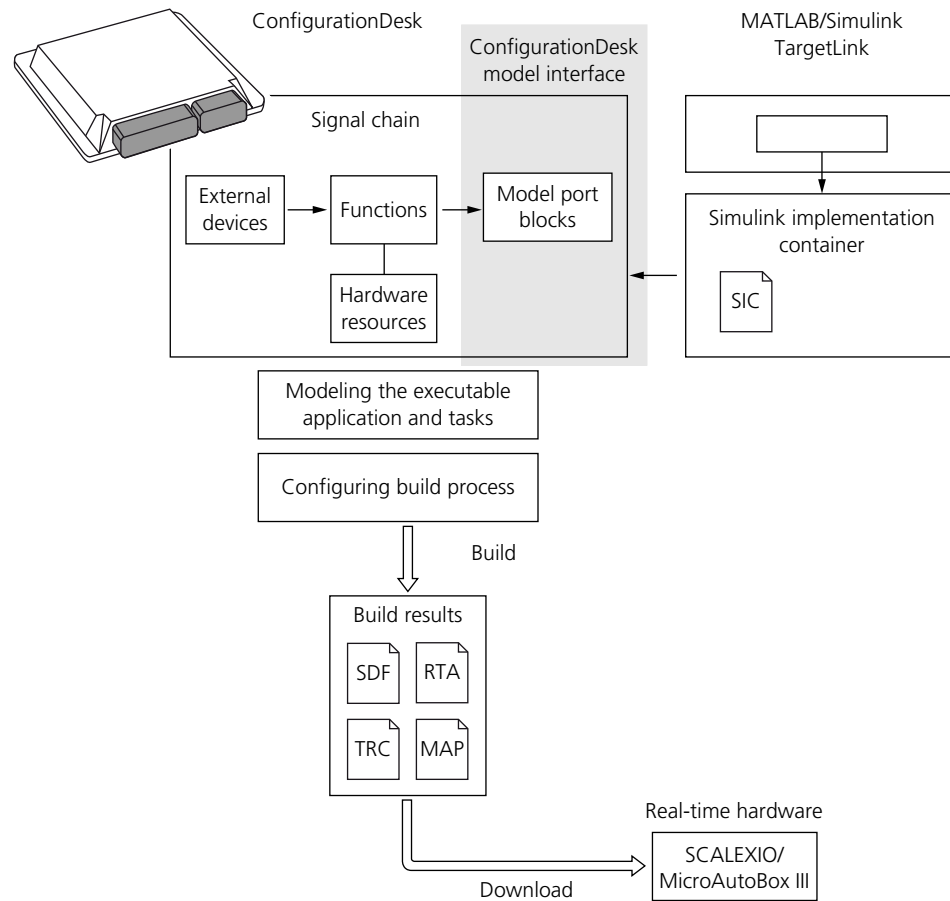
Workflow for Integrating Simulink Implementation Containers in Executable Applications

Use scenario

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

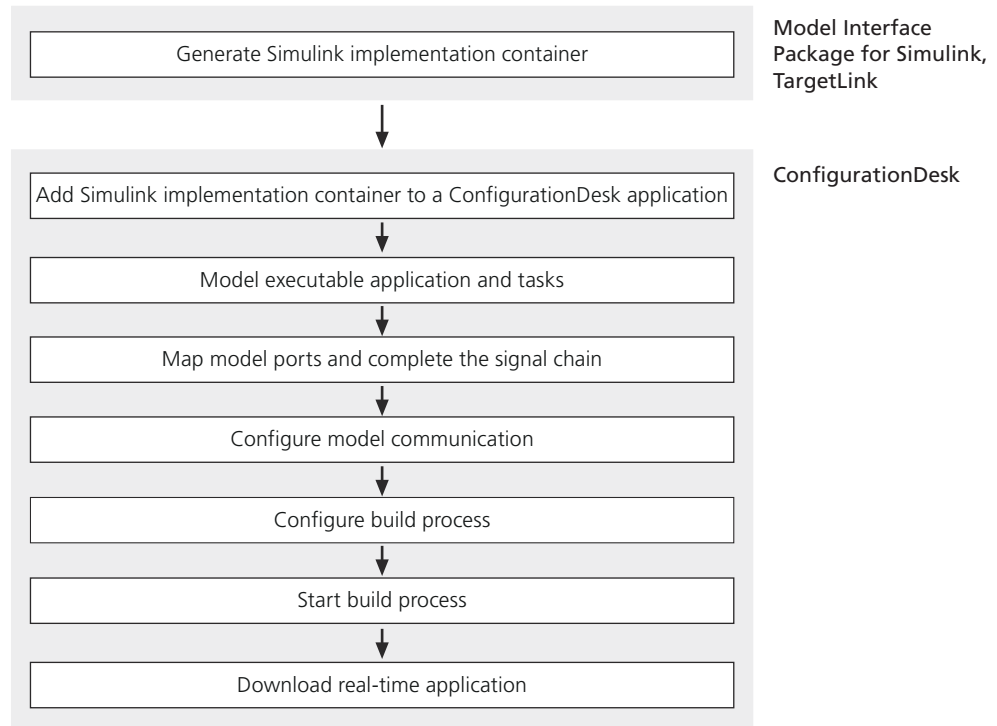
Overview

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.



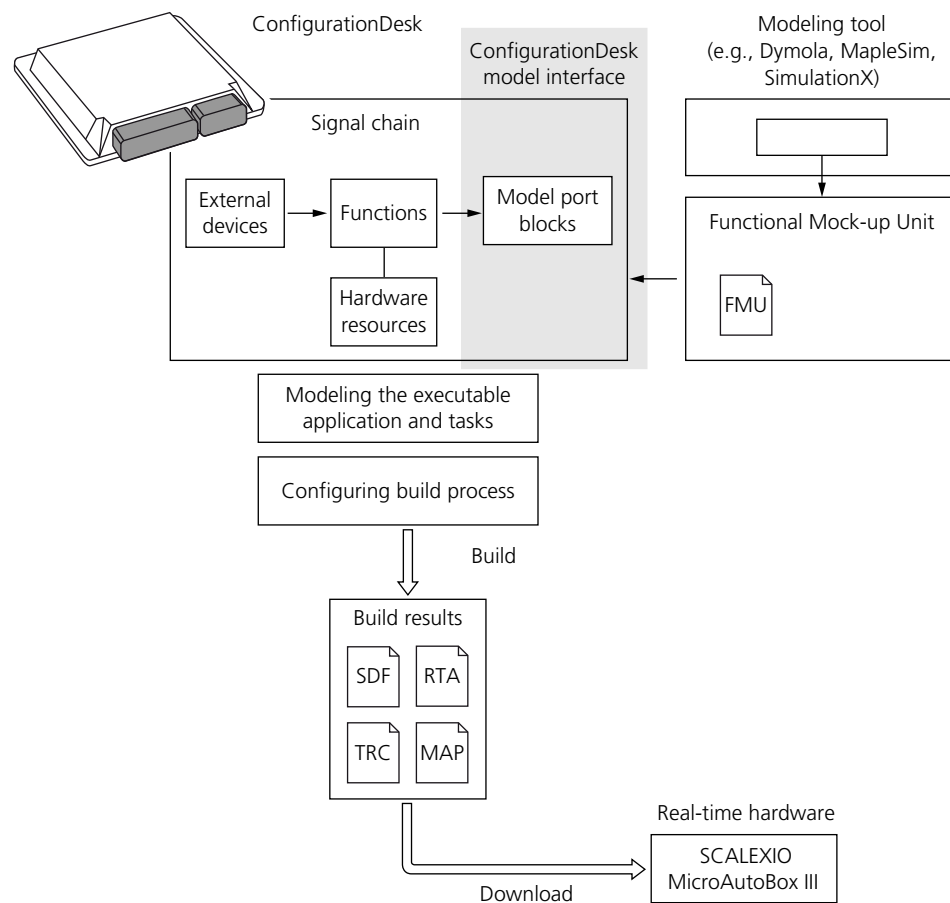
Workflow for Integrating FMUs in Executable Applications

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 V-ECU implementation, for example.

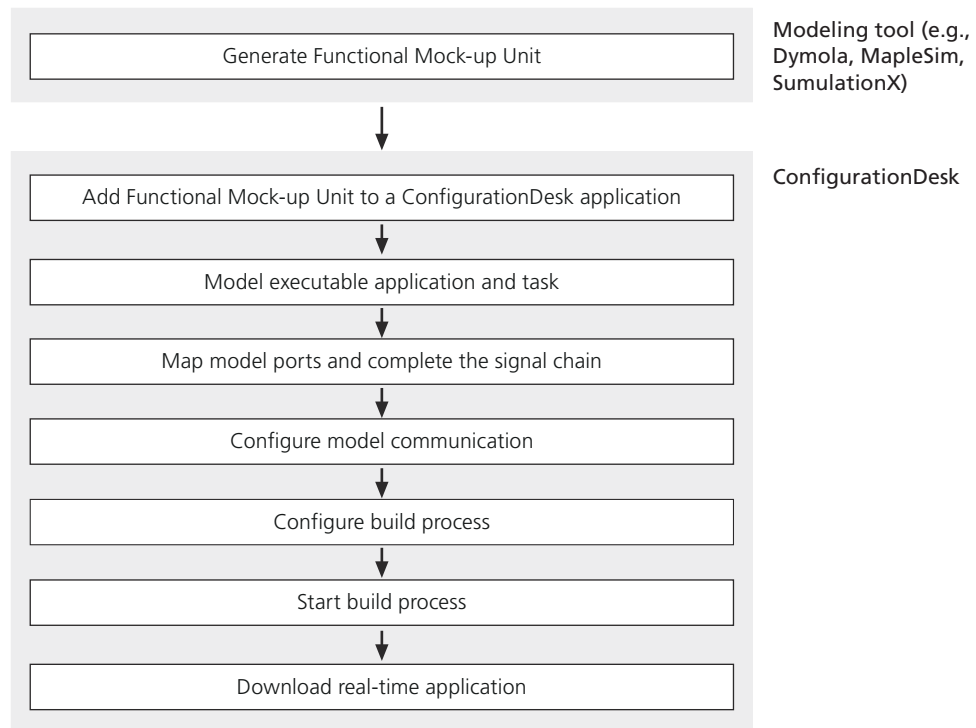
Overview

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



Workflow

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



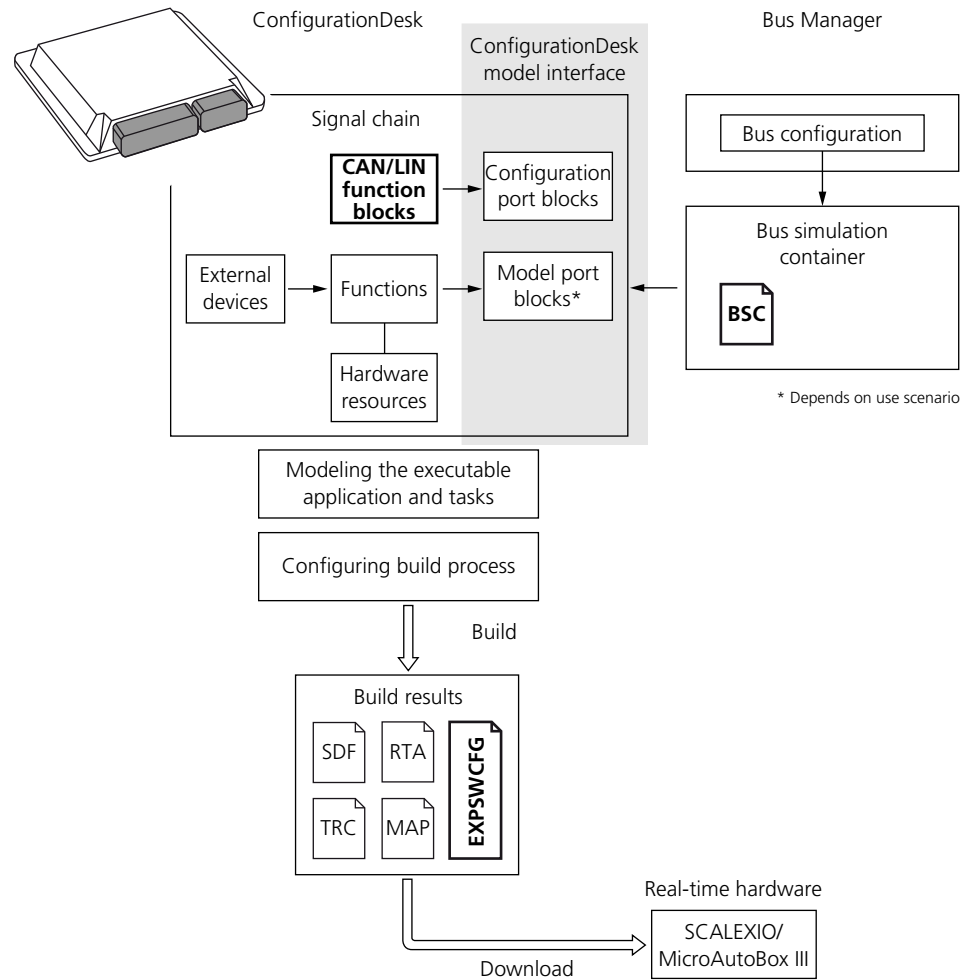
Workflow for Integrating Bus Simulation Containers in Executable Applications

Use scenario

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

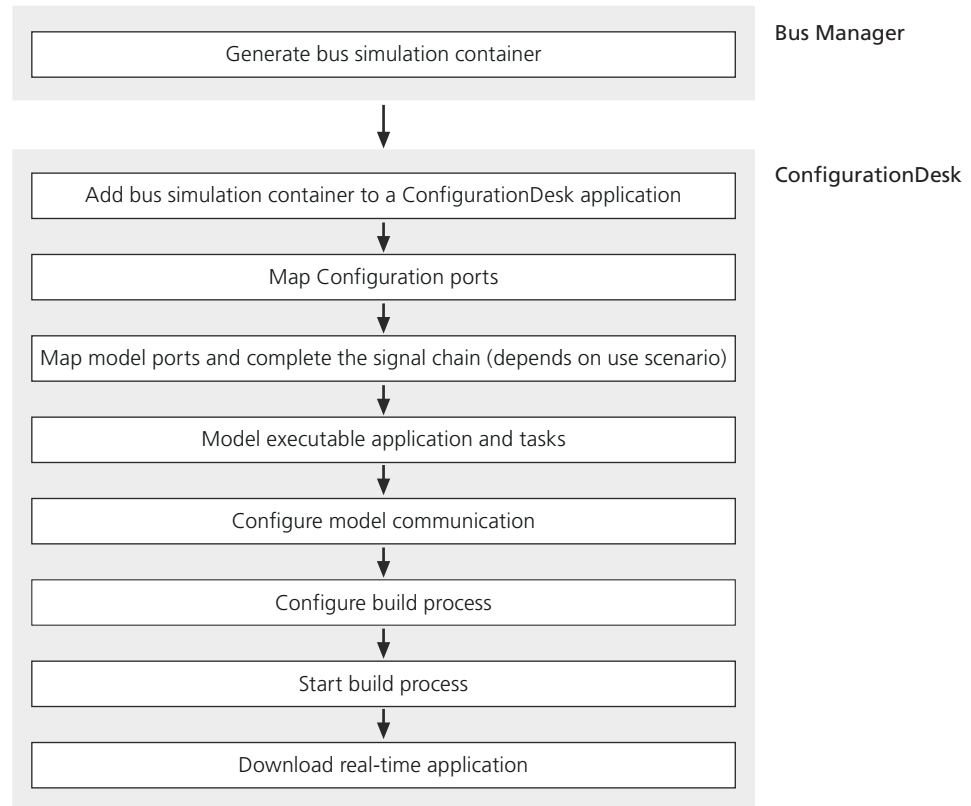
Overview

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



Workflow

The workflow includes the steps that are specific to implementing a real-time application container a bus simulation container.



Integrating V-ECUs in Executable Applications

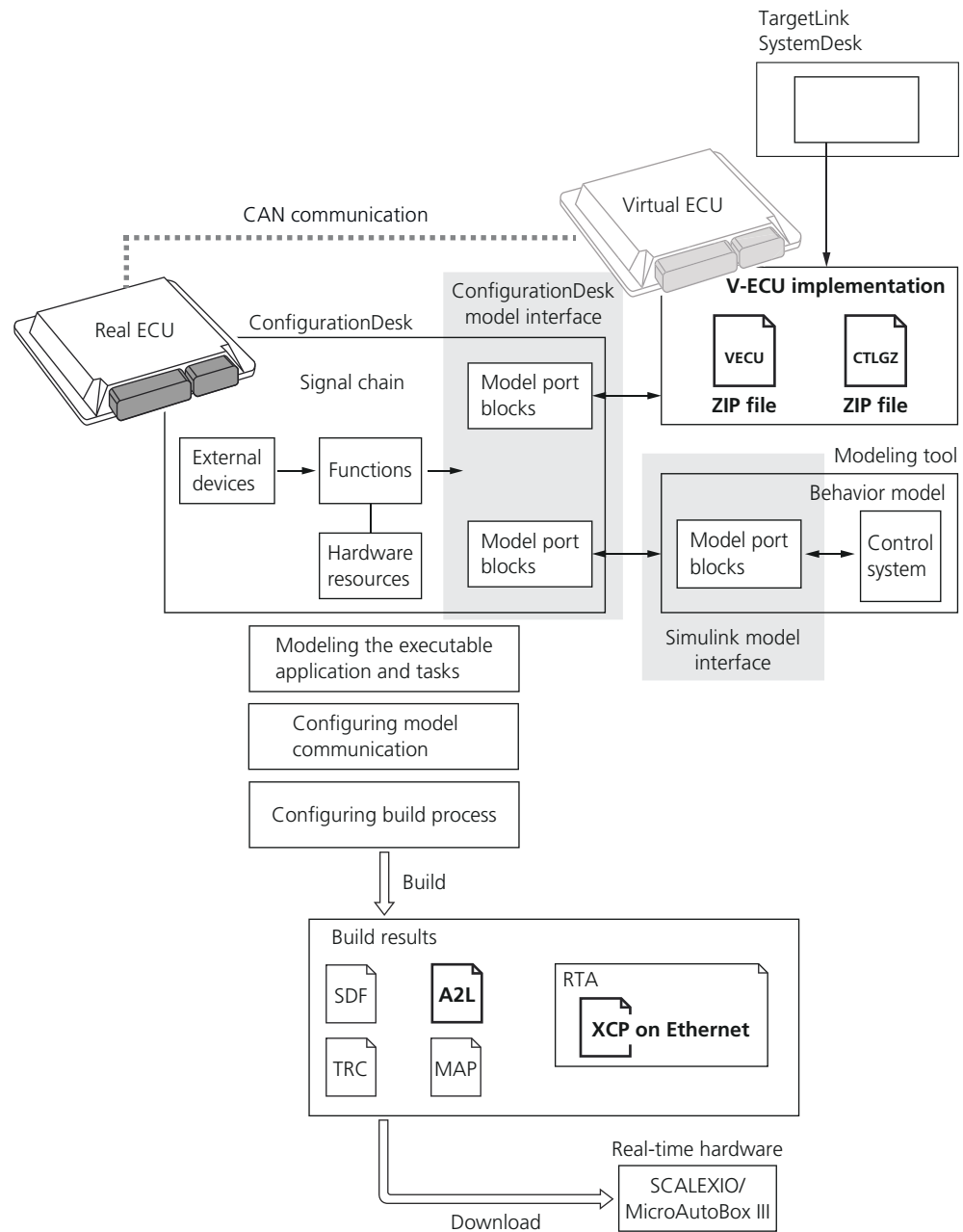
Use scenario

With ConfigurationDesk, you can integrate V-ECUs in your executable applications (real-time applications). These V-ECUs can communicate with real ECUs via CAN bus. Thus, an ECU network consisting of real and virtual ECUs can be used for simulations and tests.

V-ECU implementations can be provided by TargetLink or SystemDesk.

Overview

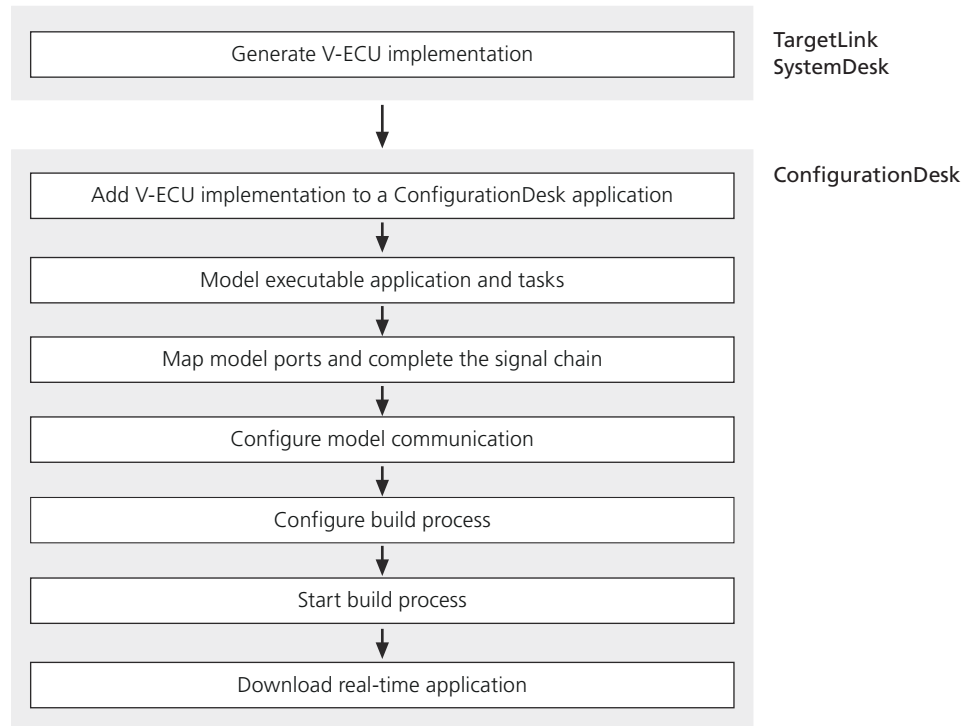
The following illustration gives you an overview of the use scenario. Parts that are specific for V-ECU implementations are in bold letters.



The representation of a V-ECU in ConfigurationDesk is a specific model implementation. This so-called V-ECU implementation is part of the model topology, and the interface to the I/O functionality in ConfigurationDesk is realized via model port blocks just as with Simulink behavior models.

Workflow

The workflow includes the steps that are specific to implementing a real-time application containing a V-ECU implementation.

**Further information**

For details on how to integrate V-ECUs in executable applications, refer to [Working with V-ECU Implementations \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(83f22ed94ec5517769dd76d702c6bfd8_img.jpg\)](#)).

B

behavior model 11

C

Common Program Data folder 6

D

documentation, available 7

Documents folder 6

E

external cable harness 22

I

integration of V-ECUs 61

L

Local Program Data folder 6

logical signal chain 18

M

model interface 12

P

physical signal chain 22

printed documents 7

R

real-time model 11

S

signal chain

 logical 18

 physical 22

