

SCALEXIO

Hardware and Software Overview

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.

© 2008 - 2021 by:
dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Germany

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

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

Contents

About This Document	7
Introduction to SCALEXIO Systems	9
Fields of Application for SCALEXIO Systems.....	10
Using a SCALEXIO System for Hardware-in-the-Loop Simulation.....	10
Using a SCALEXIO System for Rapid Control Prototyping.....	11
Hardware for a SCALEXIO System.....	13
Overview of the SCALEXIO Hardware.....	13
SCALEXIO Processing Hardware.....	17
SCALEXIO I/O Boards.....	18
HighFlex I/O Boards.....	21
MultiCompact I/O Units and Boards.....	22
I/O Boards for Extended I/O Slots.....	22
Software Tools for Working with a SCALEXIO System.....	24
Software Tool Chain.....	24
ConfigurationDesk.....	27
MATLAB/Simulink.....	28
Blocksets for Modeling in Simulink.....	28
dSPACE Automotive Simulation Models and ModelDesk.....	31
ControlDesk.....	33
MotionDesk.....	33
AutomationDesk.....	34
Real-Time Testing.....	35
Installing and Connecting a SCALEXIO System	37
Transporting, Installing, and Electrical Connection.....	37
Connecting a SCALEXIO System to a Host PC.....	39
Connecting Components of a SCALEXIO System.....	40
Configuring a SCALEXIO System	43
Basics on Configuring a SCALEXIO System.....	44
Basics on Configuring the Signal Chain.....	44
Behavior Model and I/O Model.....	45
Physical Signal Chain and Logical Signal Chain.....	46

Basics of External Devices.....	47
Basics of Function Blocks.....	48
Basics of Model Port Blocks.....	50
Basic Principles of Implementing the Real-Time Model.....	51
Overview of the Configuration Process.....	52
Implementing the Behavior Model.....	53
Creating a Project and a ConfigurationDesk Application.....	55
Building the Logical Signal Chain.....	57
Modeling Real-Time Applications and Tasks.....	58
Implementing FPGA Applications.....	58
Accessing the SCALEXIO System and Assigning Functions to Hardware Resources.....	59
Building and Downloading the Real-Time Application.....	60
Archiving the ConfigurationDesk Project and the Real-Time Model.....	61
Connecting Outputs of External Devices to Inputs of the SCALEXIO System.....	62
Hardware for Signal Measurement.....	62
Function Block Types for Signal Measurement.....	63
Connecting Loads to Outputs of External Devices.....	64
Connecting Inputs of External Devices to Outputs of the SCALEXIO System.....	66
Hardware for Signal Generation.....	66
Function Block Types for Signal Generation.....	67
Configuring the SCALEXIO System for Data Communication.....	70
Hardware for Communication.....	70
CAN Bus Connection.....	71
FlexRay Bus Connection.....	74
LIN Bus Connection.....	77
Serial Interface Connection.....	80
Waking up SCALEXIO AutoBoxes/LabBoxes via Bus Signals.....	82
Connecting External Devices to the Power Supply.....	85
Basics on Simulating the Vehicle Battery.....	85
Controlling the Battery Voltage Level.....	87
Switching the High Rails.....	88
Data Transmission Between dSPACE Systems.....	90
Basics of Data Transmission Between dSPACE Systems.....	90
Configuring a Gigalink Connection for PHS-Bus-Based Systems.....	92
Configuring a Gigalink Connection for SCALEXIO Systems.....	93
 Working with ASM and ModelDesk.....	 95
Basics on Automotive Simulation Models.....	95

Modeling with Automotive Simulation Models.....	97
Setting the Parameter Values of ASMs Using ModelDesk.....	97
Modeling the Environment (Roads and Scenarios).....	98
Accessing the SCALEXIO System for Downloading the Parameter Values with ModelDesk.....	101
Experimenting with a SCALEXIO System	103
Basics on Experimenting Using a SCALEXIO System.....	104
Basics on Experimenting With ControlDesk.....	107
Working with ControlDesk.....	108
Test Automation	111
Test Automation Using AutomationDesk and Python Scripts.....	112
Registering the Simulation Platform and Handling the Simulation Application.....	112
Preparing the Test.....	115
Testing the Simulation Application via XIL API Library.....	116
Test Automation Using Real-Time Testing.....	118
Basics on Real-Time Testing.....	118
Executing Automated Tests Using Real-Time Testing.....	119
Simulating Electrical Errors in the Wiring	123
Basics of Electrical Errors Simulation in a SCALEXIO System.....	124
Basics on Electrical Error Simulation with SCALEXIO Systems.....	124
Hardware for Electrical Error Simulation on SCALEXIO Systems.....	128
License Required for Electrical Error Simulation.....	132
Safety Precautions for Simulating Electrical Errors with a SCALEXIO System.....	133
Workflow for Simulating Electrical Errors.....	135
Allowing Electrical Error Simulation for Pins of the External Devices.....	137
Specifying Allowed Failure Classes for ECU Pins.....	137
Executing Electrical Error Simulation.....	141
Executing Electrical Error Simulation.....	141
Troubleshooting	143
No Battery Voltage After the Overcurrent Protection is Activated.....	143
Offset in Time of Signals of SCALEXIO and PHS-Bus-Based Systems.....	144

Glossary	145
Appendix	163
Abbreviations.....	163
Index	165

About This Document

Contents

This document introduces you to the SCALEXIO system. It provides an overview of the hardware system and the software tools which are used to implement the simulation model on the simulator and to use the system for experimenting. It describes the whole workflow for working with a SCALEXIO system.





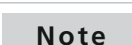


This document is primarily targeted at engineers who want to start working with a SCALEXIO system.


Graphical overview

A poster shows a graphical overview of a SCALEXIO system with I/O hardware, channel types, and I/O functions. Refer to **C:\Program Files <x86>\Common Files\dSPACE\HelpDesk <ReleaseVersion>\Print\SCALEXIO_IO_Overview.pdf**.

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.

Symbol	Description
	Precedes the document title in a link that refers to another document.

Naming conventions

dSPACE user documentation uses the following naming conventions:

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

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

Special folders

Some software products use the following special folders:

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

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

or

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

Documents folder A standard folder for user-specific documents.

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

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

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

Accessing dSPACE Help and PDF Files


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

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

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

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

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

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

Introduction to SCALEXIO Systems

Introduction

The following topics introduce the SCALEXIO systems, including the simulation hardware and the software tools for configuring and working with it.

Where to go from here

Information in this section

Fields of Application for SCALEXIO Systems.....	10
Provides an overview of the two main fields of application for SCALEXIO systems.	
Hardware for a SCALEXIO System.....	13
Provides basic information on the hardware of a SCALEXIO system.	
Software Tools for Working with a SCALEXIO System.....	24
Introduces the software tools for working with a SCALEXIO system in different development phases.	

Fields of Application for SCALEXIO Systems

Introduction

The following topics provides an overview of the two main fields of application for SCALEXIO systems.

Where to go from here

Information in this section

[Using a SCALEXIO System for Hardware-in-the-Loop Simulation..... 10](#)

SCALEXIO systems are suitable for testing electronic control units in a hardware-in-the-loop simulation.

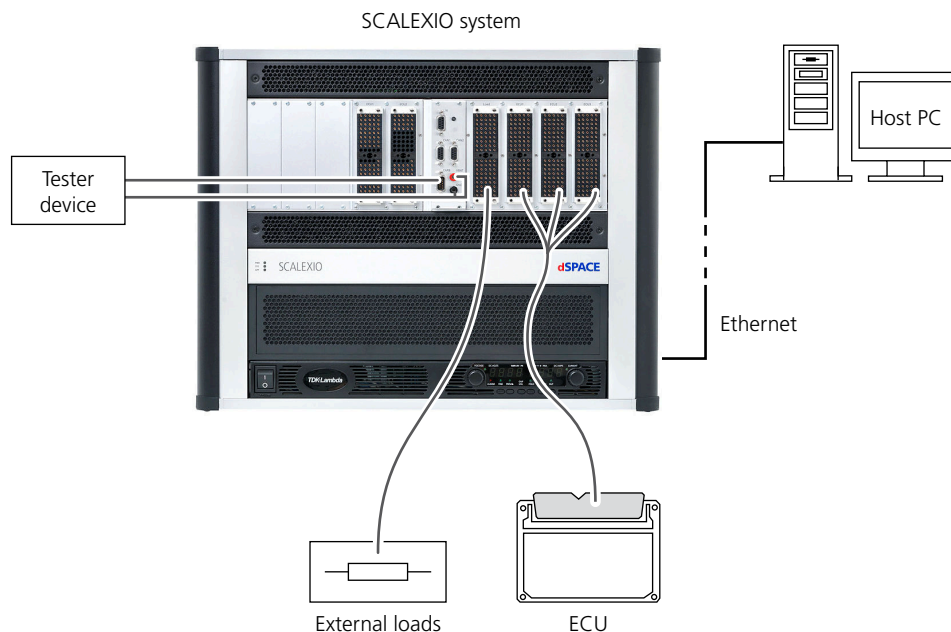
[Using a SCALEXIO System for Rapid Control Prototyping..... 11](#)

SCALEXIO systems are suitable for rapid control prototyping.

Using a SCALEXIO System for Hardware-in-the-Loop Simulation

Overview

You can use SCALEXIO systems for testing electronic control units (ECUs) in a hardware-in-the-loop (HIL) simulation. The following illustration shows an example of a SCALEXIO system that is connected to the ECU, external loads, a tester device, and the host PC.



SCALEXIO system used for HIL

When a SCALEXIO system is used as hardware-in-the-loop (HIL) simulator, it simulates the environment for the ECU. The SCALEXIO system can

- Generate the sensor signals which are required by the ECU
- Measure the signals which are generated by the ECU for actuator control
- Connect the output signals of the ECU to loads (internally in the rack or externally) to simulate the actuators
- Receive bus signals which are sent by the ECU and send bus signals to the ECU
- Simulate virtual ECUs (V-ECUs)
- Provide the battery voltage to the ECU
- Simulate electrical errors in the wiring of the ECU

The behavior of the controlled system is specified by a real-time application which runs on the SCALEXIO processing hardware. The real-time application is created on the host PC and downloaded to the SCALEXIO system via Ethernet.

Host PC

A host PC is connected to the SCALEXIO system via Ethernet. Software tools running on a host PC are used to work with the SCALEXIO system. Several tasks are performed on the host PC:

- Creating a real-time application that models the controlled system
- Implementing the real-time application on the SCALEXIO system
- Managing the real-time application on the SCALEXIO system
- Changing parameter values of the real-time application, for example, to simulate different car variants
- Measuring variables of the real-time application, for example, to analyze the signals for actuators
- Controlling the electrical error simulation in a SCALEXIO system

Related topics**Basics**

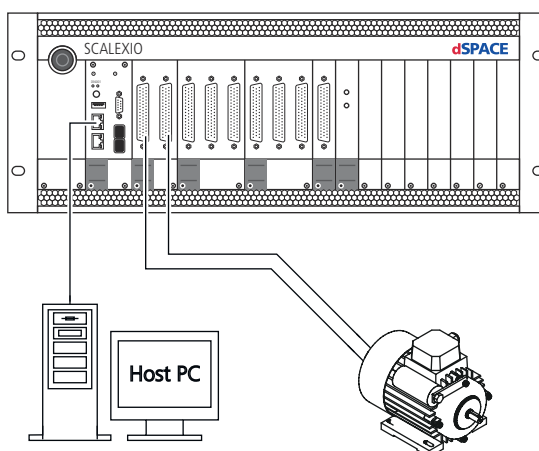
[Using a SCALEXIO System for Rapid Control Prototyping..... 11](#)

Using a SCALEXIO System for Rapid Control Prototyping

Overview

You can use SCALEXIO systems for rapid control prototyping (RCP).

The following example shows a SCALEXIO system that consists of SCALEXIO LabBox with a DS6001 Processor Board and I/O boards installed.



SCALEXIO system used for RCP

When a SCALEXIO system is used for rapid control prototyping, it controls a controlled system instead of an ECU. The SCALEXIO system must be able to

- Get the signals from sensors
- Generate the signals for the actuators
- Receive bus signals which are sent by other ECUs and send bus signals to other ECUs

The control algorithms are implemented in a real-time application which runs on the SCALEXIO processing hardware (processor board or processing unit). The real-time application is created on the host PC and downloaded to the SCALEXIO system via Ethernet.

Host PC

A host PC is connected to the SCALEXIO system via Ethernet. Software tools running on a host PC are used to work with the SCALEXIO system. Several tasks are performed on the host PC:

- Implementing a real-time application that includes the control algorithm
- Building the real-time application for the SCALEXIO system
- Managing the real-time application on the SCALEXIO system
- Measuring variables of the real-time application, for example, to analyze the signals
- Changing parameter values of the real-time application


Related topics

Basics

[Using a SCALEXIO System for Hardware-in-the-Loop Simulation..... 10](#)

Hardware for a SCALEXIO System

Introduction The hardware comprises the housings, such as racks and boxes, and the real-time hardware, such as processing units and I/O hardware.

Where to go from here	Information in this section
	Overview of the SCALEXIO Hardware..... 13 Provides an overview of the enclosures and the real-time hardware of a SCALEXIO system.
	SCALEXIO Processing Hardware..... 17 In a SCALEXIO system, two types of processing hardware are used.
	SCALEXIO I/O Boards..... 18 Provides an overview of the standard SCALEXIO I/O board.
	HighFlex I/O Boards..... 21 Provides information on HighFlex I/O boards.
	MultiCompact I/O Units and Boards..... 22 Provides information on MultiCompact I/O units and boards.
	I/O Boards for Extended I/O Slots..... 22 Provides information on I/O boards for extended I/O Slots.
	Information in other sections
	Overview of the SCALEXIO System (SCALEXIO Hardware Installation and Configuration  Introduces the SCALEXIO system, its main features and its basic working concepts.

Overview of the SCALEXIO Hardware

Introduction A SCALEXIO system consists of several components. This topic provides an overview of the enclosures and the real-time hardware.

Enclosures for SCALEXIO boards SCALEXIO systems are available in different enclosures for various application areas and test requirements. They range from off-the-shelf systems for function development and tests to huge test setups for virtual vehicle simulations and test benches, tailored individually by dSPACE Engineering Services.

The SCALEXIO family has three kinds of enclosures: SCALEXIO AutoBox, SCALEXIO LabBox, and the SCALEXIO rack.

SCALEXIO AutoBox SCALEXIO AutoBox is a shock- and vibration-resistant enclosure of the SCALEXIO family that can take up SCALEXIO boards for in-vehicle control experiments. SCALEXIO AutoBox offers space for up to seven SCALEXIO I/O boards to build a compact real-time system, which is ideally suited for function development and testing. It can be mounted in a vehicle and connected to a vehicle battery/power.



Each SCALEXIO AutoBox has one system slot to insert a DS6001 Processor Board as the processing hardware or an IOCNET router for adding SCALEXIO I/O boards to an existing SCALEXIO system.

A SCALEXIO AutoBox has seven slots to operate SCALEXIO I/O boards. They include five slots with PCIe support. The slots can be used to operate CompactPCI® Serial boards if they are qualified by dSPACE and a DS6001 Processor Board is located in the system slot.

Refer to [SCALEXIO AutoBox \(SCALEXIO Hardware Installation and Configuration\)](#).

SCALEXIO LabBox SCALEXIO LabBox is a compact enclosure of the SCALEXIO family for laboratory use. It comes in two sizes (8-slot and 19-slot) that offer space for up to 7 or up to 18 I/O boards to build a compact real-time system that is ideally suited for function development and testing. SCALEXIO LabBox can be mounted in a SCALEXIO rack or used as desktop version. The following example shows a desktop version of SCALEXIO LabBox (19-slot).



Each SCALEXIO LabBox has one system slot to insert a DS6001 Processor Board as processing hardware or an IOCNET router if it is used to add SCALEXIO I/O boards to an existing SCALEXIO system.

A SCALEXIO LabBox has 7 or 18 slots to operate SCALEXIO I/O boards. Among them are 5 slots with PCIe support. They can be used to operate CompactPCI® Serial boards if they are qualified by dSPACE and a DS6001 Processor Board is in the system slot.

Refer to [SCALEXIO LabBox \(SCALEXIO Hardware Installation and Configuration\)](#).

SCALEXIO rack A SCALEXIO rack is available in different sizes. The following illustration shows an example of the version with 9 height units (9 U).



Normally, a SCALEXIO rack contains a SCALEXIO Processing Unit as SCALEXIO processing hardware. In addition, it provides sufficient space for MultiCompact units, HighFlex I/O boards and SCALEXIO I/O boards. To insert I/O boards into the rack, slot units in two sizes are available (6 slots and 20 slots).

To simulate a car battery, a SCALEXIO rack can have a battery simulation power supply unit that can be controlled by the SCALEXIO system.

Refer to [SCALEXIO Racks \(SCALEXIO Hardware Installation and Configuration\)](#).

SCALEXIO processing hardware

Each SCALEXIO system must have at least one processing hardware component that executes the real-time application and provides an interface to the IOCNET for communication with I/O units, I/O boards, or other processing hardware. The processing hardware communicates with the host PC via Ethernet. In a SCALEXIO system, two types of processing hardware can be used, a DS6001 Processor Board or a SCALEXIO Processing Unit. For system configuration and support, each SCALEXIO processing hardware component has a web interface you can open in any Internet browser.

- The *DS6001 Processor Board* is a compact processor board with an integrated IOCNET interface. It must be inserted in the system slot of a SCALEXIO AutoBox/LabBox.

- A *SCALEXIO Processing Unit* consists of a SCALEXIO Real-Time PC with a DS2502 IOCNET Link Board. The SCALEXIO Real-Time PC is normally mounted in a SCALEXIO rack but is also available as a desktop version.

Refer to [SCALEXIO Processing Hardware](#) on page 17.

SCALEXIO I/O boards

SCALEXIO I/O boards provide a large number of I/O channels with dedicated channel types and a focus on I/O functions without a current-related functionality. SCALEXIO I/O boards can be inserted in SCALEXIO AutoBox/LabBox or in a slot unit of a SCALEXIO rack.

Refer to [SCALEXIO I/O Boards](#) on page 18.

HighFlex I/O boards

HighFlex I/O boards are versatile and finely scalable. You can use its channels for different I/O functions that you can select and configure in ConfigurationDesk. The channels are galvanically isolated from each other, so they can be used for channel multiplication (using several channels for one I/O function to enhance their physical characteristics, for example, to increase the maximum current). The channels have failure routing units, so they can be used for failure simulation. To use a HighFlex I/O board, you must insert it in a slot unit of a SCALEXIO rack.

Refer to [HighFlex I/O Boards](#) on page 21.

MultiCompact I/O hardware

MultiCompact I/O units have a great number of I/O channels for specific applications. The units have IOCNET routers to connect them to a SCALEXIO Processing Unit or further I/O units. They are installed in a SCALEXIO rack. Refer to [MultiCompact I/O Units and Boards](#) on page 22.

Boards for extended I/O slots

You can use up to five Ethernet boards and dSPACE-qualified CompactPCI® Serial boards in the extended I/O slots of a SCALEXIO AutoBox/LabBox if the AutoBox/LabBox has a DS6001 Processor Board.

Refer to [I/O Boards for Extended I/O Slots](#) on page 22.

Battery simulation power supply unit

The battery simulation power supply unit is used to simulate the vehicle battery. The battery simulation power supply unit is controlled via software by a battery simulation controller, such as the DS2907 Battery Simulation Controller or the onboard controller of the DS2680 I/O Unit. There are different battery simulation power supply units that can be installed in a SCALEXIO rack. Refer to [Battery Simulation Power Supply Unit \(SCALEXIO Hardware Installation and Configuration !\[\]\(2b17f17ebbacc911bb0ff784ab641779_img.jpg\)](#)).

SCALEXIO Processing Hardware

Introduction

Each SCALEXIO system must have at least one processing hardware component that executes the real-time application and provides an interface to the IOCNET for communication with I/O units, I/O boards, or other processing hardware. The processing hardware communicates with the host PC via Ethernet. In a SCALEXIO system, two types of processing hardware can be used, a DS6001 Processor Board or a SCALEXIO Processing Unit. For system configuration and support, each SCALEXIO processing hardware component has a web interface you can open in any Internet browser.

DS6001 Processor Board

The DS6001 Processor Board is a highly performant and compact two-slot processing hardware for SCALEXIO AutoBox/LabBox. It provides two optical and 17 electrical IOCNET interfaces and five PCIe interfaces for communication with other SCALEXIO components. The processor board can be connected to a host PC via Ethernet. Its operating system is a QNX real-time operating system.

The DS6001 Processor Board supports only standard SCALEXIO I/O boards and dSPACE-qualified CompactPCI® Serial boards with a PCIe interface as I/O hardware.

Refer to [DS6001 Processor Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(003082e50e3009141f59bd5df831749f_img.jpg\)](#)).

SCALEXIO Processing Unit

A SCALEXIO Processing Unit is based on a SCALEXIO Real-Time PC, a real-time operating system, and a DS2502 IOCNET Link Board for communication with other SCALEXIO components. The SCALEXIO Processing Unit comes in a 19" unit as a rack-mount version with front brackets to be inserted into a SCALEXIO rack or as a desktop version with feet to be used on a desktop.

SCALEXIO Real-Time PC The SCALEXIO Real-Time PC provides the calculation power to execute the real-time application. The SCALEXIO Real-Time PC consists of a high-standard industry ATX motherboard and a multicore main processor that is qualified by dSPACE for use with SCALEXIO. Its operating system is a Linux-based real-time operating system. One of the available processor cores is reserved for service jobs, the others are used for computing real-time models.

DS2502 IOCNET Link Board The DS2502 IOCNET Link Board is the interface of the SCALEXIO Processing Unit for connecting the related SCALEXIO Real-Time PC to IOCNET or Gigalink. It is installed in a specific PCIe slot of the SCALEXIO Real-Time PC. A DS2502 IOCNET Link Board has 4 or 8 optical ports. It provides also 6 angle clocks (APUs) as master.

Optional Ethernet boards A SCALEXIO Processing Unit can provide optional Ethernet boards. Refer to:

- [DS6331-PE Ethernet Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(05a3150ca7eafd44fce8deaa48838121_img.jpg\)](#))

- [DS6333-PE Automotive Ethernet Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(467d80e979964f7f8c752fb22248b5b7_img.jpg\)\)](#)
- [DS6334-PE Ethernet Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(b71552d33dbf62adf5e5199a70ee02bf_img.jpg\)\)](#)
- [DS6336-PE Ethernet Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(03134b765d1473836ff001925b1b0550_img.jpg\)\)](#)

A SCALEXIO Processing Unit supports MultiCompact I/O units and boards (refer to [MultiCompact I/O Units and Boards](#) on page 22), HighFlex I/O boards and standard SCALEXIO I/O boards as I/O hardware.

Refer to [SCALEXIO Processing Unit \(SCALEXIO Hardware Installation and Configuration !\[\]\(99f58673407353e96a019fbca558fd72_img.jpg\)\)](#).

Related topics

Basics

HighFlex I/O Boards.....	21
MultiCompact I/O Units and Boards.....	22
SCALEXIO I/O Boards.....	18

SCALEXIO I/O Boards

Introduction

SCALEXIO I/O boards provide a large number of I/O channels with dedicated channel types and a focus on I/O functions. You can use SCALEXIO I/O boards in a slot unit of a SCALEXIO rack or SCALEXIO LabBox.

SCALEXIO Hardware for FPGA applications

DS2655 FPGA Base Board The DS2655 FPGA Base Board provides a user-programmable FPGA platform. The board is designed for high-speed HIL applications that require the model to be computed, at least partly, on an FPGA. For details, refer to [DS2655 FPGA Base Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(6a9b39b98eb945faa14c645ec99e4eaa_img.jpg\)\)](#).

DS6601 FPGA Base Board The DS6601 FPGA Base Board provides a user-programmable FPGA platform. The board is the successor of the DS2655 FPGA Base Board and designed for applications that require very fast, high-resolution signal processing and must be computed, at least partly, on an FPGA. For details, refer to [DS6601 FPGA Base Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(9c2e8d1b5bd77cb5c9f83b7a9cff79fd_img.jpg\)\)](#).

DS6602 FPGA Base Board The DS6602 FPGA Base Board provides a user-programmable FPGA platform. The board is designed for eDrives and power electronics applications that require the model to be computed on a high-end

FPGA. For details, refer to [DS6602 FPGA Base Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(1d3a1175dd4902218e694b9c098adb83_img.jpg\)](#)

DS2655M1 Multi-I/O Module The DS2655M1 Multi-I/O Module is a single-slot I/O module for a SCALEXIO FPGA base board. The module provides five parallel 14-bit A/D channels, five parallel 14-bit D/A channels, and ten bidirectional digital I/O channels. Refer to [DS2655M1 Multi-I/O Module \(SCALEXIO Hardware Installation and Configuration !\[\]\(c507f772dba2b921f86777f01218e570_img.jpg\)](#)

DS2655M2 Digital I/O Module The DS2655M2 Digital I/O Module is a single-slot I/O module for a SCALEXIO FPGA base board. The module provides shared pins for up to 32 digital I/O channels. Some of these pins can also be used for up to 8 channels for RS232 or RS485 communication. Refer to [DS2655M2 Digital I/O Module \(SCALEXIO Hardware Installation and Configuration !\[\]\(4729e517bc6a7cd81c8025b9646574fb_img.jpg\)](#)

DS6651 Multi-I/O Module The DS6651 Multi-I/O Module is a single-slot I/O module for a SCALEXIO FPGA base board. The module provides 6 parallel A/D channels, 6 parallel D/A channels, 16 bidirectional digital I/O channels and 2 onboard sensor supplies. The module is ideally suited for control and simulation of standard electric drives and power electronic components. Refer to [DS6651 Multi-I/O Module \(SCALEXIO Hardware Installation and Configuration !\[\]\(cbe80b694ebd74fcfe136a095b608235_img.jpg\)](#)

DS6101 Multi-I/O Board

The DS6101 Multi-I/O Board is a 3-slot SCALEXIO I/O board. It provides 69 I/O channels for voltage-related functions, including analog, digital, resistance, and special input/output groups, for use cases such as lambda probe simulation. Refer to [DS6101 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(cbe2492b119e39e02a1dab2af4a4b296_img.jpg\)](#)

DS6121 Multi-I/O Board









The DS6121 Multi-I/O Board is a single-slot SCALEXIO I/O board. It provides the required interfaces for electric motor control applications and can be used for up to two electric motors. Refer to [DS6121 Multi-I/O Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(3e2231b1ad3ca8da8658228c00dd08e0_img.jpg\)](#)

DS6201 Digital I/O Board

The DS6201 Digital I/O Board is a 3-slot SCALEXIO I/O board. It provides 3 x 32 bidirectional digital I/O channels for generating or measuring typical automotive signals and TTL signals. Each channel can be configured individually as digital input or output. Refer to [DS6201 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(870f5d5e9c0d57485634be3ecf52f3ca_img.jpg\)](#)

DS6202 Digital I/O Board

The DS6202 Digital I/O Board is a single-slot SCALEXIO I/O board. It provides 32 fast bidirectional channels for advanced I/O functions. These channels can also be configured in pairs to establish up to 16 differential inputs. Refer to [DS6202 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(0d5ec72f61334709c3fc9450209b754f_img.jpg\)](#)

DS6221 A/D Board	The DS6221 A/D Board is a single-slot SCALEXIO I/O board. It is high-performance analog-to-digital converter board for digitizing analog input signals at high sampling rates. The board provides 16 differential inputs and 8 additional trigger inputs for connecting external trigger sources. Refer to DS6221 A/D Board (SCALEXIO Hardware Installation and Configuration ).
DS6241 D/A Board	The DS6241 D/A Board is a single-slot SCALEXIO I/O board. It provides 20 DAC channels with analog voltage output for signal generation and 4 additional trigger inputs for connecting external trigger sources. Refer to DS6241 D/A Board (SCALEXIO Hardware Installation and Configuration ).
DS6301 CAN/LIN Board	The DS6301 CAN/LIN Board is a single-slot bus board for CAN and LIN communication. It provides 4 CAN/CAN FD channels and 4 LIN channels. Refer to DS6301 CAN/LIN Board (SCALEXIO Hardware Installation and Configuration ).
DS6311 FlexRay Board	The DS6311 FlexRay Board a single-slot bus board for FlexRay communication. It provides four FlexRay controllers with two FlexRay channels (A and B) each. Refer to DS6311 FlexRay Board (SCALEXIO Hardware Installation and Configuration ).
DS6321 UART Board	The DS6321 UART Board is a single-slot bus board for serial communication. It provides 4 independent UART channels, each supporting RS232, RS422, RS485, or K-Line at a time. Refer to DS6321 UART Board (SCALEXIO Hardware Installation and Configuration ).
DS6341 CAN Board	The DS6341 CAN Board is a single-slot bus board for CAN communication. It provides 4 independent CAN/CAN FD channels. Refer to DS6341 CAN Board (SCALEXIO Hardware Installation and Configuration ).
DS6342 CAN Board	The DS6342 CAN Board is a single-slot bus board for CAN communication. It provides 8 independent CAN/CAN FD channels. Refer to DS6342 CAN Board (SCALEXIO Hardware Installation and Configuration ).
DS6351 LIN Board	The DS6351 LIN Board is a single-slot bus board for LIN communication. It provides 8 independent LIN channels. Refer to DS6351 LIN Board (SCALEXIO Hardware Installation and Configuration ).

HighFlex I/O Boards

Introduction

HighFlex I/O boards are versatile and finely scalable. You can use its channels for different I/O functions. You can select and configure the I/O functions for the channels in ConfigurationDesk. The channels are galvanically isolated from each other. They can be used for channel multiplication (using several channels for one I/O function to enhance their physical characteristics, for example, to increase the maximum current). The channels have failure routing units, so they can be used for electrical error simulation. You can use HighFlex I/O boards in slot units of a SCALEXIO rack.

A DS6001 Processor Board does not support HighFlex I/O boards.

DS2601 Signal Measurement Board

DS2601 Signal Measurement Board is a HighFlex I/O board for SCALEXIO slot units. It measures signals for actuators coming from the ECU. Its channels can be connected to internal loads (loads mounted on the board) or external loads (loads connected to the SCALEXIO system's load connectors). It has 10 channels and requires 2 slots in a slot unit. Refer to [DS2601 Signal Measurement Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(0aff635c4179ba9e710b00f4b01d3b20_img.jpg\)](#)).

DS2621 Signal Generation Board

The DS2621 Signal Generation Board is a HighFlex I/O board for SCALEXIO slot units. It generates sensor signals for an ECU. It has 10 channels and requires 1 slot in a slot unit. Refer to [DS2621 Signal Generation Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(0b5e7e25e8775f7e7e80906ada4f0021_img.jpg\)](#)).

DS2642 FIU & Power Switch Board

The DS2642 FIU & Power Switch Board is a HighFlex I/O board that switches the power supply for a connected ECU. The board has 10 channels, which are used to switch the battery voltage that is provided by the Battery Simulation Power Supply Unit. The board also has a central FIU (failure insertion unit), which switches electrical failures for electrical error simulation. It requires 2 prepared slots in a SCALEXIO slot unit. Refer to [DS2642 FIU & Power Switch Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(6bb0e4f14c4133b37d2887cb37e67ddd_img.jpg\)](#)).

DS2671 Bus Board

The DS2671 Bus Board is a HighFlex I/O board for SCALEXIO slot units. It connects the ECU to a simulated bus (CAN, LIN, and FlexRay) or a serial interface. The board has 4 channels and requires 1 slot in a SCALEXIO slot unit. Refer to [DS2671 Bus Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(bd3b31712ad9bab5a241210fa6925cdd_img.jpg\)](#)).

MultiCompact I/O Units and Boards

Introduction

MultiCompact I/O units have a greater number of I/O channels for specific applications. The channels have failure routing units, so they can be used for failure simulation. Connecting real and substitute loads is possible.

MultiCompact I/O units provide ECU/load connectors with fixed pinouts. The unit have IOCNET interfaces, so they can be connected to the IOCNET. MultiCompact I/O units are installed in a SCALEXIO rack.

You can use MultiCompact I/O boards in slot units of a SCALEXIO rack.

A DS6001 Processor Board does not support MultiCompact I/O units and boards.

DS2680 I/O Unit

The DS2680 I/O Unit is a MultiCompact I/O unit for a SCALEXIO rack. The I/O unit provides all the I/O channels for the powertrain and vehicle dynamics. Internal loads can be connected to the actuator channels by the integrated DS2680-IL Load Board. To connect an ECU to communication buses (CAN, LIN, FlexRay), the DS2672 Bus Module can be optionally installed in the unit by dSPACE. Refer to [DS2680 I/O Unit \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2672 Bus Module The DS2672 Bus Module is an optional module for the DS2680 I/O Unit. The module provides six independent bus channels (2 × CAN, 2 × LIN, 2 × FlexRay). Refer to [DS2672 Bus Module \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2680-IL Load Board The DS2680-IL Load Board is a removable board of the DS2680 I/O Unit that can carry internal loads for each signal measurement channel and the special high-speed channels for lambda probes. It also provides some of the I/O unit's front connectors and elements. Refer to [DS2680-IL Load Board \(SCALEXIO Hardware Installation and Configuration\)](#).

DS2690 Digital I/O Board

The DS2690 Digital I/O Board is a 3-slot MultiCompact I/O board for SCALEXIO slot units. It provides 30 digital I/O channels for signal measurement and signal generation. Refer to [DS2690 Digital I/O Board \(SCALEXIO Hardware Installation and Configuration\)](#).

I/O Boards for Extended I/O Slots

Introduction

You can use the following Ethernet boards and dSPACE-qualified CompactPCI® Serial boards with PCIe interfaces in the extended I/O slots of a SCALEXIO AutoBox/LabBox if the AutoBox/LabBox has a DS6001 Processor Board.

DS6333-CS Automotive Ethernet Board

The DS6333-CS Automotive Ethernet Board is a PCIe x4 board that comprises two Ethernet modules with two LAN ports each and one additional onboard LAN port for connecting external devices to a SCALEXIO AutoBox/LabBox. Different Ethernet modules are available to support standard Ethernet (10/100/1000 Mbit/s) or automotive Ethernet (100/1000 Mbit/s). Refer to [DS6333-CS Automotive Ethernet Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(d84e7ea36f695d92cb39ec32c307ac93_img.jpg\)](#)).

DS6335-CS Ethernet Board

The DS6335-CS Ethernet Board is a cost-efficient PCIe x4 board that comprises two Ethernet modules with two LAN ports each and one additional onboard LAN port for connecting external devices to a SCALEXIO AutoBox/LabBox. Different Ethernet modules are available to support standard Ethernet (10/100/1000 Mbit/s) or automotive Ethernet (100/1000 Mbit/s). The board is not supported by the Ethernet Configuration Package. Refer to [DS6335-CS Ethernet Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(9dfdaff1d86ba3c1f8353b4d1b61b8c5_img.jpg\)](#)).

DS6336-CS Ethernet Board

The DS6336-CS Ethernet Board is a PCIe x4 board that provides two independent LAN ports for connecting external devices to a SCALEXIO AutoBox/LabBox via an Ethernet connection with up to 10 Gbit/s. Refer to [DS6336-CS Ethernet Board \(SCALEXIO Hardware Installation and Configuration !\[\]\(8d0f0e0fe25b320c33272c52aec1fbca_img.jpg\)](#)).

dSPACE-qualified CompactPCI Serial boards

You can use up to five Ethernet boards and dSPACE-qualified CompactPCI® Serial boards in the extended I/O slots of a SCALEXIO AutoBox/LabBox if the AutoBox/LabBox has a DS6001 Processor Board. dSPACE-qualified CompactPCI Serial boards can be used for I/O extensions or for dedicated customer-specific issues. The related slots must be equipped with a DS6411 CS Adapter each.

To operate dSPACE-qualified CompactPCI Serial boards with PCIe interfaces in SCALEXIO LabBox contact dSPACE Support.

Related topics**Basics**

[DS6411 CS Adapter \(SCALEXIO Hardware Installation and Configuration !\[\]\(c444627dab9fee9a1550c053ffaaaae2_img.jpg\)](#))
[SCALEXIO AutoBox \(SCALEXIO Hardware Installation and Configuration !\[\]\(e4a71fb14267cbc3c68a54ad33289c8f_img.jpg\)](#))
[SCALEXIO LabBox \(SCALEXIO Hardware Installation and Configuration !\[\]\(14c85d5bb83aa7451202bf95a5e535fd_img.jpg\)](#))
[Specific Hardware for SCALEXIO AutoBox/LabBox \(SCALEXIO Hardware Installation and Configuration !\[\]\(70b176afdd52e72e916a315f5ffd470c_img.jpg\)](#))

Software Tools for Working with a SCALEXIO System

Introduction

dSPACE provides several software tools for working with a SCALEXIO system in different development phases.

Where to go from here

Information in this section

Software Tool Chain.....	24
Provides an overview of the software tools which you can use to work with the SCALEXIO system.	
ConfigurationDesk.....	27
Provides information on the purpose and features of ConfigurationDesk.	
MATLAB/Simulink.....	28
Provides information on the simulation software.	
Blocksets for Modeling in Simulink.....	28
Provides information on the blocksets which can be used in Simulink for implementing the behavior model for a SCALEXIO system.	
dSPACE Automotive Simulation Models and ModelDesk.....	31
Provides information on the dSPACE models which can be used for modeling the controlled system and the software tool for parameterizing the models.	
ControlDesk.....	33
Provides information on the purpose and features of ControlDesk.	
MotionDesk.....	33
Provides information on the purpose and features of MotionDesk.	
AutomationDesk.....	34
Provides information on the purpose and features of AutomationDesk.	
Real-Time Testing.....	35
Provides information on the purpose and features of Real-Time Testing.	

Software Tool Chain

Introduction

For real-time simulation, a real-time application must be implemented for the SCALEXIO system. Furthermore, the SCALEXIO system must be controlled by a host PC. This topic gives you an overview of the software tools which are needed for this purpose.

Tool chain

The following list shows the software tools which are used to work with a SCALEXIO system. The software tools are grouped in three phases:

- Implementing: Software tools and blocksets for modeling and implementing the real-time application:
 - MATLAB/Simulink
 - ConfigurationDesk
 - Model Port Block Library
 - Model Separation Block Library
 - RTI CAN MultiMessage Blockset
 - RTI LIN MultiMessage Blockset
 - FlexRay Configuration Package
 - Automotive Simulation Models
 - RTI FPGA Programming Blockset
- Experimenting: Software tools for handling and controlling the real-time application and visualizing the simulation results:
 - ControlDesk
 - MotionDesk
 - ModelDesk
- Testing: Software tools for test automation:
 - AutomationDesk
 - Real-Time Testing

Compatibility for Working with a SCALEXIO system

The products for working with a SCALEXIO system must be compatible. This is only guaranteed for products delivered with the same dSPACE Release DVD.

Implementing

MATLAB/Simulink MATLAB®/Simulink® is used for modeling the simulation model (control algorithm or controlled system) which runs on the SCALEXIO system. The Simulink model is only the behavior model. The I/O functionality is modeled in an I/O model which is implemented using ConfigurationDesk. The Model Port blocks are used to connect the behavior model to the I/O model.

ConfigurationDesk ConfigurationDesk is the tool for implementing the I/O model on the SCALEXIO system. In ConfigurationDesk you can map the I/O signals of the simulation model to the external devices and vice versa. You can select and configure the functions which are supported by the SCALEXIO boards. When you have implemented the behavior and I/O model, ConfigurationDesk can build the real-time application containing both models. This is downloaded to the simulator for the ECU test.

Model Port Block Library The Model Port Block Library contains Simulink blocks that are used to read or write signals of the I/O model and to trigger asynchronous tasks in the behavior model.

Model Separation Block Library The Model Separation Block Library contains the Model Separation Setup block, which separates individual models

from an overall model in MATLAB/Simulink and generates a model communication description file (MCD file) that contains information on the separated models and their connections.

RTI CAN MultiMessage Blockset The RTI CAN MultiMessage Blockset contains Simulink blocks for implementing a CAN communication in the behavior model. You can implement the restbus simulation for the ECU to be tested. This means, you can generate all CAN messages which are received by the ECU and read all CAN messages which are send from the ECU.

RTI LIN MultiMessage Blockset The RTI LIN MultiMessage Blockset contains Simulink blocks for implementing a LIN communication in the behavior model. You can implement the restbus simulation for the ECU to be tested. This means, you can generate all LIN frames which are received by the ECU and read all LIN frames which are send from the ECU.

FlexRay Configuration Package The FlexRay Configuration Package consists of the FlexRay Configuration Tool and the FlexRay Configuration Blockset. To use the tools, the communication must be specified in a FIBEX or AUTOSAR file.

The FlexRay Configuration Tool reads the FIBEX or AUTOSAR file and lists all PDUs contained in the file in a tree. You select the PDUs which are necessary for a restbus simulation of the ECU to be tested and configure them.

The FlexRay Configuration Blockset contains the Simulink blocks for FlexRay communication. These blocks are configured using the settings done with the FlexRay Configuration Tool. The result is an automatically generated FlexRay model which contains Simulink blocks which are ready for the Simulink model to implement the FlexRay communication.

Automotive Simulation Models Automotive Simulation Models (ASM) are blocksets which you can use to create the behavior model in Simulink. The blocksets model different engines and vehicle dynamics in the automotive area.

RTI FPGA Programming Blockset The RTI FPGA Programming Blockset is a Simulink blockset that allows you to program an FPGA in a SCALEXIO system. This blockset is necessary if a SCALEXIO system has an FPGA board, for example, a DS2655 FPGA Base Board with I/O modules.

Experimenting

ControlDesk ControlDesk is the software tool for experimenting with a SCALEXIO system. It can be used for downloading the real-time application, calibrating parameters and measuring signals.

MotionDesk MotionDesk is the software tool for 3-D animation during experimenting with a SCALEXIO system. It can be used for visualizing the movement of objects, for example, for testing vehicle dynamics ECUs.

ModelDesk ModelDesk is a tool which can be used to parameterize Automotive Simulation Models. Usually a lot of parameters must be specified for Automotive Simulation Models. ModelDesk provides a graphical environment which makes it easy to specify the parameters. The parameter values can be changed and downloaded to the real-time application on SCALEXIO system.

Testing

AutomationDesk AutomationDesk is a universal tool for creating and managing automation tasks. A typical automotive automation task is testing a new electronic control unit (ECU). The test process can be made more efficient by using AutomationDesk.

Real-Time Testing Using Real-Time Testing, you can execute tests synchronously with the model, so all test actions are performed on a real-time basis. The tests for Real-Time Testing must be written in the Python programming language.

ConfigurationDesk

Purpose

ConfigurationDesk is the software tool for implementing the real-time model on the SCALEXIO system. Different versions of ConfigurationDesk are available. For implementing the real-time model on the SCALEXIO system, you need the Implementation Version.

Features

The Implementation Version of ConfigurationDesk has a bundle of main features that are helpful for implementing the real-time model:

- ConfigurationDesk's Project Manager allows you to organize all the relevant project information, such as configurations and application-specific data.
- ConfigurationDesk allows you to create and configure signal chains between the SCALEXIO system and external devices (ECU or controlled system).
- ConfigurationDesk allows you to select the functions for the SCALEXIO I/O channels.
- Real-time applications can be built and downloaded to the connected SCALEXIO system.
- ConfigurationDesk's Platform Manager gives you detailed information on the hardware topology, for example, the number and types of I/O units and I/O boards of the connected SCALEXIO system.
- ConfigurationDesk's Bus Manager lets you configure bus communication for simulation and inspection purposes (e.g., restbus simulation). You can work with multiple communication matrix files of various file formats and configure the communication of multiple communication clusters at a time.

Further information

For detailed information on using the Implementation Version of ConfigurationDesk, refer to [Introduction to ConfigurationDesk \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(4fe57c3593bf1b21d272ae7ac8dfaf77_img.jpg\)](#)).

MATLAB/Simulink

Purpose	MATLAB®/Simulink® is used for modeling the real-time application which runs on the SCALEXIO system.
MATLAB	MATLAB is an integrated environment for numeric computation that specializes in working with matrices. It combines a powerful user interface with 2-D and 3-D graphics and a comprehensive library of mathematical analysis techniques. MATLAB not only eliminates the need to program by hand, but also lets engineers analyze and visualize data and develop algorithms with exceptionally improved productivity and creativity.
Simulink	Simulink is an interactive environment integrated in MATLAB for modeling, analysis and simulation. It provides a graphical user interface for constructing block diagram models via drag & drop operations. Advanced features such as conditionally executed subsystems, If and Switch-Case subsystems, For and While iterator subsystems, data typing, and signal labeling make it ideally suited to complex control design tasks. Simulink's large block library is enhanced by specific dSPACE block libraries for I/O hardware support. You can use these blocks to build the entire real-time experiment setup, including I/O and initialization, without writing a single line of code.

Blocksets for Modeling in Simulink

Purpose	MATLAB®/Simulink® is used for modeling the behavior model. For implementing the I/O interface, blocksets for modeling in Simulink are necessary.
Model Port Block Library	<p>The Model Port Block Library is a library for Simulink. It provides Simulink blocks for interfacing the Simulink model to the I/O model implemented in ConfigurationDesk. The blockset contains three blocks:</p> <ul style="list-style-type: none"> ▪ The Data Inport block reads data from the I/O model (implemented in ConfigurationDesk) and makes it available in Simulink. ▪ The Data Outport block reads data from the Simulink model and makes them available in the I/O model in ConfigurationDesk. ▪ The Runnable Function block makes a ConfigurationDesk function event available as the trigger source for an asynchronous task in the Simulink model. <p>The model port blocks are represented in the model topology in ConfigurationDesk. Each model port block in Simulink corresponds to a model port block in ConfigurationDesk.</p>



For details on the model port blocks, refer to [Model Interface Package for Simulink Reference](#)

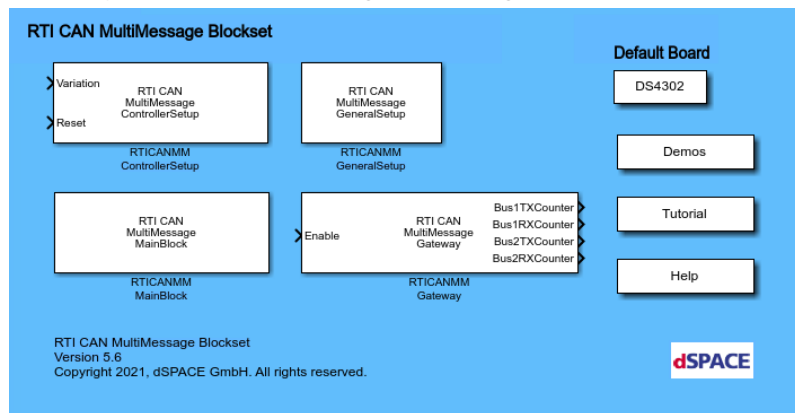
Model Separation Block Library

The Model Separation Block Library contains the Model Separation Setup block, which separates individual models from an overall model in MATLAB/Simulink. Additionally, the Model Separation Setup block generates a model communication description file (MCD file) that contains information on the separated models and their connections. You can import the MCD file (and thus the new model topology with different models) in ConfigurationDesk to build a multicore real-time application or a multi-processing-unit application and download it to the dSPACE real-time hardware, where the separated models are executed in parallel on the single cores of a single processing-unit or a multi-processing-unit system.

For details on the model separation blocks, refer to [Model Interface Package for Simulink Reference](#) .

RTI CAN MultiMessage Blockset

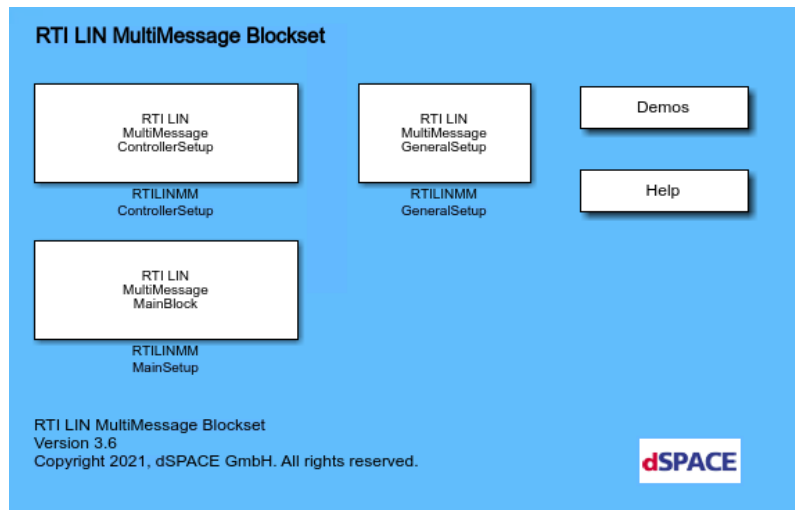
The CAN communication performed by the SCALEXIO system can be implemented in Simulink using the RTI CAN MultiMessage Blockset. The blockset allows you to handle a large number of CAN messages. You can control and manipulate all the CAN messages with a single Simulink block.



In ConfigurationDesk, the model must be analyzed to get the information needed for assigning a CAN controller to the CAN bus. For an overview of the workflow, refer to [CAN Bus Connection](#) on page 71. For details on the blocks, refer to [RTI CAN MultiMessage Blockset Reference](#) .

RTI LIN MultiMessage Blockset

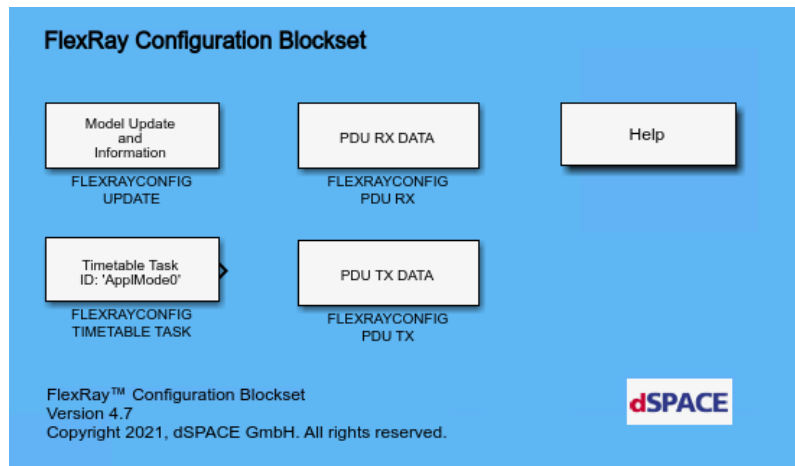
The LIN communication performed by the SCALEXIO system can be implemented in Simulink using the RTI LIN MultiMessage Blockset. The blockset allows you to handle a large number of LIN frames. You can control and manipulate all the LIN frames with a single Simulink block.



In ConfigurationDesk, the Simulink model must be analyzed to get the information needed for assigning a controller to the LIN bus. For an overview of the workflow, refer to [LIN Bus Connection](#) on page 77. For details on the blocks, refer to [RTI LIN MultiMessage Blockset Reference](#)

FlexRay Configuration Blockset

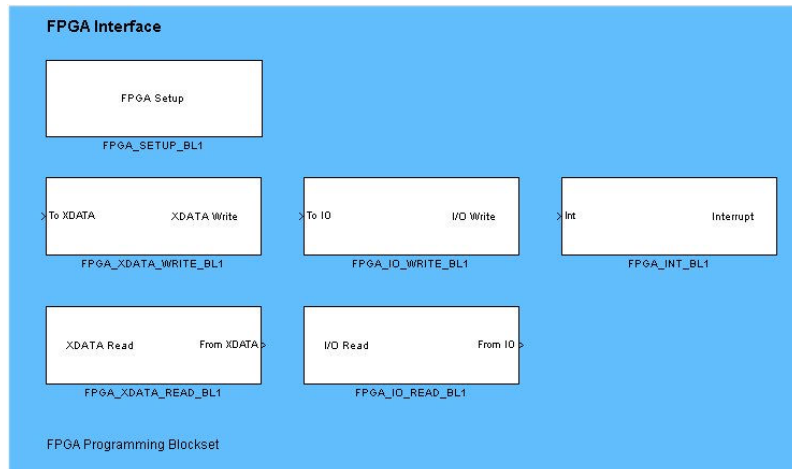
The FlexRay communication performed by the SCALEXIO system is implemented in Simulink using the FlexRay Configuration Blockset. The blockset allows you to handle a large number of FlexRay PDUs. You can control and manipulate each PDU with a single Simulink block. Additionally, the Simulink model must be analyzed to get the information needed for assigning controllers to the FlexRay bus in ConfigurationDesk.



In ConfigurationDesk, you can also configure further FlexRay communication properties such as controlling the assigned FlexRay controller and reading additional status information. For an overview of the workflow, refer to [FlexRay Bus Connection](#) on page 74. For details on the blocks, refer to [FlexRay Configuration Blockset Reference](#) .

RTI FPGA Programming Blockset

The RTI FPGA Programming Blockset is a Simulink® blockset that allows you to program an FPGA in a dSPACE system. If the SCALEXIO system has a DS2655 FPGA Base Board, you can use this blockset to implement an FPGA application for it.



Refer to [Implementing FPGA Applications](#) on page 58.

dSPACE Automotive Simulation Models and ModelDesk

Introduction

To implement a behavior model an appropriate Simulink model is necessary. dSPACE provides Automotive Simulation Models (ASM) for this purpose, and also ModelDesk, a software tool that can be used to parameterize the ASM.

Features of ASM

The ASM are simulation models for developing and testing automotive electronic control units (ECUs). They are implemented in Simulink and tailor-made for real-time execution on a dSPACE system. They can be used for modeling the behavior model for a SCALEXIO system. The ASMs are contained in simulation packages for engines and vehicle dynamics.

The ASM have the following features:

- An ASM consists of several components that can be combined to make an ASM via their standardized interfaces.
- ASM are open models. You can view and modify the models right down to the level of standard Simulink blocks.
- The vehicle dynamics model consists of components to simulate the dynamic behavior of the vehicle with its drivetrain, a complete environment with road and driver, and a basic engine. It can be extended by a brake hydraulics model or custom models.

- The engine models consist of components that comprise a complete full-featured engine with models for drivetrain, vehicle dynamics, and environment. The engine models can be extended by a complex turbocharger model or custom models.
- A whole virtual vehicle can be created by combining the engine and the vehicle dynamics model.
















Features of ModelDesk

ModelDesk is a graphical user interface for parameterizing and managing the parameter sets of the ASMs. It has the following features:





- Editing all the parameters of ASM and custom models
- Managing the parameter sets
- Handling variants of parameter sets
- Downloading the parameter values to the SCALEXIO system
- Calculating parameter values based on measurement data
- Road Generator to generate a road model for simulation and for visualization in MotionDesk
- Scenario Editor to specify scenarios with the movements of the ASM vehicle and for simulating other traffic participants around the simulated ASM vehicle

Further information

Several libraries are available for building an ASM. For detailed information on the blocksets, refer to the related reference:

- [ASM Brake Hydraulics Reference](#) 
- [ASM Diesel Engine Reference](#) 
- [ASM Diesel InCylinder Reference](#) 
- [ASM Diesel Exhaust Reference](#) 
- [ASM Drivetrain Basic Reference](#) 
- [ASM Electric Components Reference](#) 
- [ASM Environment Reference](#) 
- [ASM Gasoline Engine Reference](#) 
- [ASM Gasoline InCylinder Reference](#) 
- [ASM Pneumatics Reference](#) 
- [ASM Traffic Reference](#) 
- [ASM Trailer Reference](#) 
- [ASM Truck Reference](#) 
- [ASM Turbocharger Reference](#) 
- [ASM Vehicle Dynamics Reference](#) 

For detailed information on using ModelDesk, refer to the related document:

- [ModelDesk Basics](#) 
- [ModelDesk Parameterizing](#) 
- [ModelDesk Road Creation](#) 
- [ModelDesk Scenario Creation](#) 

ControlDesk

Purpose

ControlDesk is the software tool for experimenting with a SCALEXIO system. It can be used to download the real-time application and control the experiment.

Features

ControlDesk provides an intuitive environment for your experiment, calibration, measurement, and diagnostic tasks. Its wizard and template mechanisms guide you through the working steps, which are kept to a minimum, cutting the time needed for training and support. ControlDesk has integrated project and experiment management with many configuration options, including a folder structure the same as that in File Explorer, because of ControlDesk's file-based approach. A variety of features dedicated to data analysis ensure efficient measurement data handling.

For working with the SCALEXIO system, ControlDesk is extended:

- ControlDesk provides a graphical user interface (GUI) for electrical error simulation that is delivered with the Failure Simulation Package. The GUI bases on dSPACE XIL API .NET and its EESPort implementation. This GUI for electrical error simulation is targeted only at the initial operation of the HIL simulator's failure simulation hardware and to perform first manual tests.
- The Bus Navigator lets you handle CAN messages, LIN frames, and FlexRay PDUs, manipulate messages, frames, and PDUs before transmission, exclude them from being transmitted, etc.
- The Signal Editor lets you create, configure, display, and manage signals in signal description sets. You can use signal description sets as signal generators to stimulate model variables of real-time applications running on the simulator.

MotionDesk

Purpose

MotionDesk is the software tool for visualizing movements of a mechanical system, such as a car in a virtual world. It is useful if you want to create or test a controller for vehicle dynamics.

Features

MotionDesk has been developed as a complement to the dSPACE tool chain to visualize the movement of mechanical objects in the virtual world. MotionDesk can visualize mechanical parts like vehicles or robotic arms that move in a virtual world and are simulated with SCALEXIO systems. If your PC is fast enough, the latency time between simulation and visualization is low, which makes the system capable of "man-in-the-loop" simulation. For a presentation of simulation results, the simulation data can be recorded and replayed afterwards.

Creating a virtual world is simple. A wide range of 3-D objects for vehicle simulation is available in a 3-D object library. Library expansion is easy as the 3-D

object geometries are described in the COLLADA standard. The objects are assembled to a scene with MotionDesk's Scene Editor. You only have to drag them from the Library Browser to the scene and move them to the correct positions with the mouse or via numerical values. The scene looks more realistic through the use of modern rendering techniques such as texture mapping.

Once created, the 3-D scene comes to life in MotionDesk. MotionDesk gets the motion data from a SCALEXIO system and moves the movable objects in accordance with the data. To get the right view of the scene, observers can be defined with varied behaviors, for example, static in the scene or following a moving object.

Further information

For detailed information on using MotionDesk, refer to [Introduction to MotionDesk \(MotionDesk Basics !\[\]\(0f848bbd71cef6b345273b16f905912a_img.jpg\)](#)).

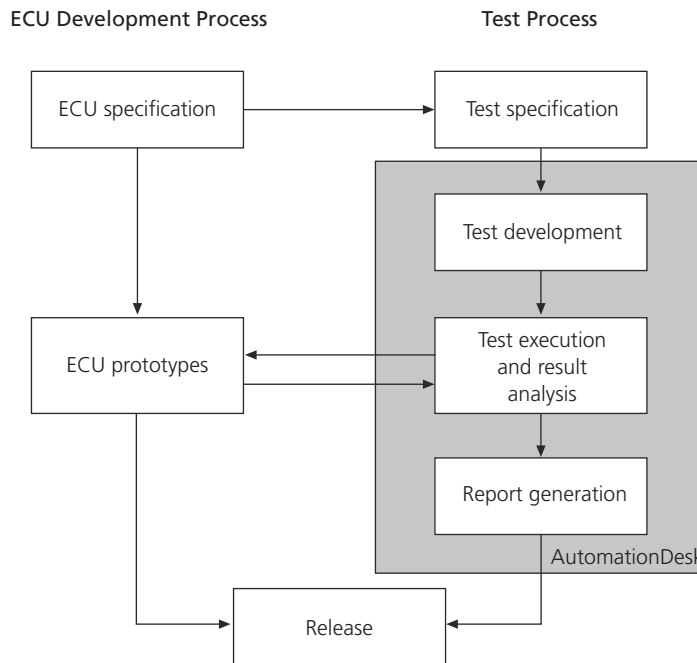
AutomationDesk

Purpose

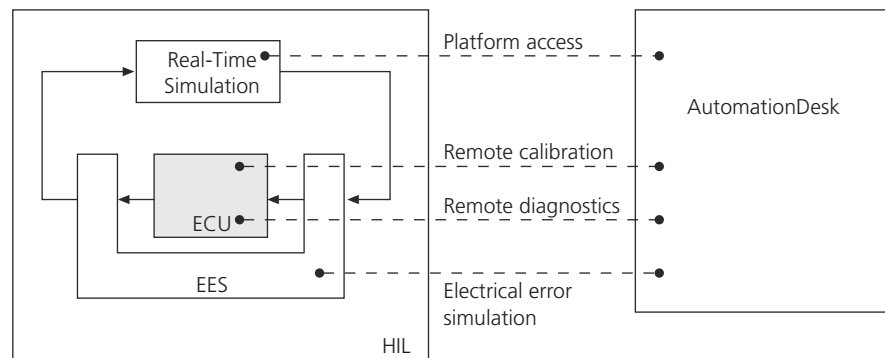
AutomationDesk is a software tool for creating and managing any kind of automation tasks.

Features

You can create and specify control flows and test parameters, execute tests and log the results. All execution results are stored as XML files. They can be exported to HTML or PDF reports. If you search for specific information, you will find it easily because of the structured view of your whole project. This ensures that you can identify and reproduce your tests.



AutomationDesk can be used for implementing automated ECU tests. Several AutomationDesk libraries are provided by dSPACE to access the HIL, real-time simulation, diagnostic tools, calibration tools and failure simulation.



Further information

For detailed information on using AutomationDesk, refer to [Introduction to AutomationDesk](#) (AutomationDesk Introduction And Overview ).

Real-Time Testing

Purpose

With Real-Time Testing, you can perform tests synchronously to the real-time application on the SCALEXIO system.

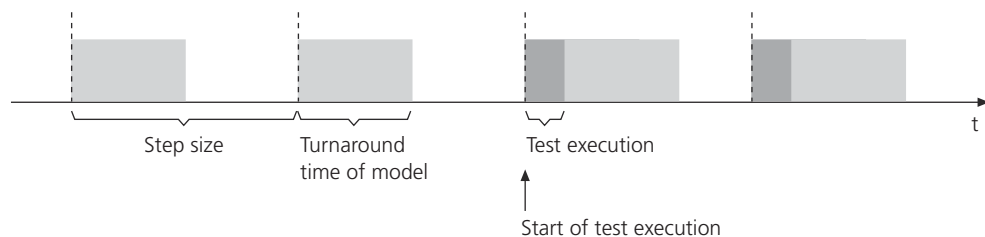
Features

The basis of Real-Time Testing is a Python interpreter running on the real-time processor. The Python interpreter contains special modifications which make it suitable for real-time environments. The real-time processor calls the Python interpreter and executes the real-time application in each sampling step.


Real-Time Testing provides the following features:

- Tests are programmed in Python, the object-oriented scripting language. Tests are called RTT sequences.
- RTT sequences are performed synchronously with the real-time application.
- RTT sequences are executed on the simulator in real time.
- RTT sequences can use standard Python modules, modules provided by dSPACE for real-time testing, and user Python modules.
- A single RTT sequence can contain concurrent elements. You can stimulate and monitor signals in parallel, for example.
- Several independent RTT sequences can be performed at the same time.
- Time measurements in RTT sequences are performed at the resolution of the simulation step size.
- RTT sequences have read/write access to all variables of the real-time application.

The RTT sequences are executed by a Python interpreter running on the real-time processor. The real-time processor calls the Python interpreter and executes the real-time application in each sampling step. The following illustration shows an example in which the test execution is performed before the model is calculated.



Further information

For detailed information on using Real-Time Testing, refer to [Real-Time Testing Guide](#) .

Installing and Connecting a SCALEXIO System

Where to go from here

Information in this section

[Transporting, Installing, and Electrical Connection.....](#) 37

There are some transportation conditions and specific environmental conditions you have to comply with.

[Connecting a SCALEXIO System to a Host PC.....](#) 39

The processing hardware of a SCALEXIO system is controlled by dSPACE software installed on a host PC. To communicate with the host PC, the SCALEXIO processing hardware must be connected via Ethernet.

[Connecting Components of a SCALEXIO System.....](#) 40

Provides information on the communication network of a SCALEXIO system.

Transporting, Installing, and Electrical Connection

Introduction

There are some transportation conditions and specific environmental conditions you have to comply with.

WARNING

Risk of electric shock and/or property damage

Working with the SCALEXIO hardware can be dangerous. To avoid electric shock and/or property damage, refer to [Safety Precautions \(SCALEXIO Hardware Installation and Configuration !\[\]\(41aea2746216b27a6939d696d8e035da_img.jpg\)](#)).

Requirements on the location

- Do not place the SCALEXIO system on an unstable cart, stand, or table. The cart, stand, or table must be able to carry the weight of the SCALEXIO system as well as the external cable harness, external devices, etc.
- Do not drop the SCALEXIO system or its components.
- Position the SCALEXIO system away from heat sources such as radiators, heat storage devices, power amplifiers, and other products producing heat.
- When positioning the SCALEXIO system, make sure that you can easily unplug the power cords if you have to disconnect the system from the power supply.
- The SCALEXIO system and its components are not waterproof. Do not expose them to water or other liquids.
- Route all the external cables so that they are not likely to be walked on or pinched by items placed on or against them.
- Do not block the ventilation inlets and outlets at the front and rear of the SCALEXIO rack. There must be a space of at least 50 mm (2.0 in) in front of these openings.
- Ensure that the required environment conditions are fulfilled:
 - For SCALEXIO AutoBox, refer to [Data Sheet of SCALEXIO AutoBox \(SCALEXIO Hardware Installation and Configuration !\[\]\(5ba1bc70d78f05c00988641e5e513c62_img.jpg\)](#)).
 - For SCALEXIO LabBox (8-slot), refer to [Data Sheet of SCALEXIO LabBox \(8-Slot\) \(SCALEXIO Hardware Installation and Configuration !\[\]\(0d3dd579ab24f8020cd6c2659f3acb8c_img.jpg\)](#)).
 - For SCALEXIO LabBox (19-slot), refer to [Data Sheet of SCALEXIO LabBox \(19-Slot\) \(SCALEXIO Hardware Installation and Configuration !\[\]\(77aacc67724f470ed5556217e9f1530a_img.jpg\)](#)).
 - For a SCALEXIO rack, refer to [Data Sheet of a Typical SCALEXIO Rack \(SCALEXIO Hardware Installation and Configuration !\[\]\(2f0a16d48331670e3ba1ef62cc117e02_img.jpg\)](#)).

Transporting the SCALEXIO system

You can transport a SCALEXIO system with all its boards installed. However, there are some transportation conditions you have to comply with.

To prevent personal injury or property damage, comply with the following transportation conditions:

- Disconnect the host PC and all the external devices (e.g., ECU, external loads, tester devices) from the SCALEXIO system.
- Disconnect all external cabling from the SCALEXIO system, for example, the external cable harness, the Ethernet connection cable, the power cords.
- If the SCALEXIO system has transport handles, use all the transport handles on both sides of the system to lift it. All the handles must bear an equal load.
- Depending on the weight of the SCALEXIO system, use a cart to transport it.
- If the SCALEXIO system or rack has wheels, move it only on a firm level surface. Do not move it on stairs or steps. Move it only when it is disconnected from the power source and external devices.
- Handle the SCALEXIO system with care and do not drop it. The SCALEXIO system and/or its components can be damaged if the system is dropped.
- The SCALEXIO system and its components are not waterproof. Do not expose them to water or other liquids.
- Ensure that the temperature is in the range -20 °C ... +80 °C (-4 °F ... 176 °F) while transporting the SCALEXIO system.

Connecting to mains

Note the following points when connecting the SCALEXIO system to the mains:

- Operate the SCALEXIO system only from the kind of power source indicated on the rear panel.
- Use appropriate miniature circuit breakers with a rated current of max. 16 A.
- It is recommended to connect the SCALEXIO system via residual current operated protective devices.
- Always use approved power cords with an appropriate cross-section, isolation, etc. for connection to the power source.
- You have to connect all the main connectors of the SCALEXIO system to the power source.
- The SCALEXIO system has protection class 1: The system must be operated with a protective earth/ground connection via the protective earth/grounding conductor of the power cord(s).
- Always ensure that the system is operated from properly grounded wall outlets only.
- Use multiple socket outlets only if they comply with the system's power requirements. Multiple socket outlets can cause hazardous touch currents due to the accumulation of earth leakage currents.
- It is recommended to connect each of the system's power source connectors to a separate power outlet with its own separate circuit breaker (16 A).

Connecting a SCALEXIO System to a Host PC

Introduction

The processing hardware of a SCALEXIO system is controlled by dSPACE software installed on a host PC. To communicate with the host PC, the SCALEXIO processing hardware must be connected via Ethernet.

Physical connection

You can connect the SCALEXIO system to the host PC directly (peer-to-peer) or via a network. If you use a network connection, it is recommended to use a separate network for the host PC and the SCALEXIO system, since their communication might result in a large amount of data traffic. For details, refer to [Connecting the SCALEXIO System to the Host PC \(SCALEXIO Hardware Installation and Configuration !\[\]\(2b376d1a92330ab09dad2665d2f89bf5_img.jpg\)](#)).

The Ethernet connector of the SCALEXIO Processing Unit is internally connected to a Ethernet connector on the back of the SCALEXIO rack.

The DS6001 Processor Board has the Ethernet connector for the host PC connection on the front.

Firewall settings

If a firewall is installed on the host PC, it must allow communication between the SCALEXIO system and the host PC. Windows firewalls are automatically adapted during the installation of dSPACE software, but you have to adapt other firewalls

manually. Refer to [Adapting Firewall Settings \(SCALEXIO Hardware Installation and Configuration !\[\]\(d263118e0bfd47dc6bc704167d936b83_img.jpg\)](#)).

IP address

To work with the SCALEXIO system, you have to define a unique IP address. The IP address is independent of the connection between SCALEXIO system and host PC, i.e., you have to define the IP address regardless of whether the SCALEXIO system and host PC are connected in a peer-to-peer connection or via network. You can either define a fixed IP address or let a DHCP server set the IP address.

By default, the Ethernet port for the host PC connection of SCALEXIO processing hardware is preconfigured as follows:

- IP address: 192.168.140.10
- Network mask: 255.255.255.0

To change the settings, refer to [Basics on the Network Configuration \(SCALEXIO Hardware Installation and Configuration !\[\]\(fa6f3af6bfa46c5d4a2d362681095beb_img.jpg\)](#)).

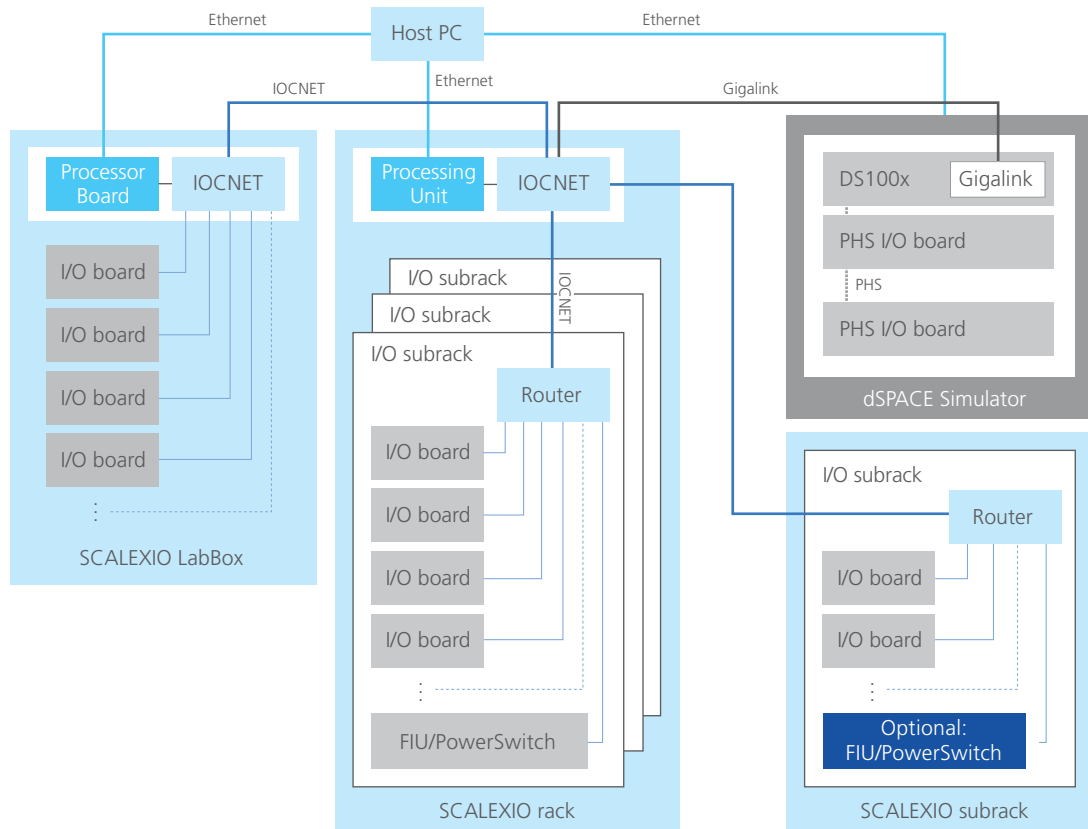
Connecting Components of a SCALEXIO System

Introduction

Provides information on the communication network of a SCALEXIO system.

Communication network of a SCALEXIO system

The illustration below shows an example of the communication network of a distributed SCALEXIO system.



Host PC communication The processing hardware of a SCALEXIO system is controlled by dSPACE software installed on a host PC. To communicate with the host PC, the SCALEXIO processing hardware must be connected via Ethernet. For details, refer to [Ethernet Connection \(SCALEXIO Hardware Installation and Configuration\)](#).

IOCNET communication IOCNET (I/O carrier network) is a dSPACE-specific high-speed serial communication bus that connects all the real-time hardware in a SCALEXIO system. IOCNET can also be used to build a multiprocessor system that consists of multiple SCALEXIO processor hardware components.

IOCNET is the default communication network for SCALEXIO systems. The connection to other members of a SCALEXIO system is performed via optical and electrical IOCNET connectors. By using IOCNET, you can add several I/O subracks (i.e., SCALEXIO slot units or SCALEXIO LabBoxes) and the installed I/O boards to your system. These decentralized components do not need their own SCALEXIO processing hardware.

- Optical IOCNET connections are used:
 - To connect multiple SCALEXIO processor hardware components in a multiprocessing system to distribute the real-time model and increase the computation power.
 - To connect SCALEXIO processing hardware to one or more external I/O racks or SCALEXIO LabBoxes to increase the number of I/O channels.
 - To connect I/O units and slot units (subracks) inside a SCALEXIO rack to increase the number of I/O channels.
- Electrical IOCNET connections are used for the backplanes inside of SCALEXIO slot units and SCALEXIO LabBoxes to connect the single SCALEXIO boards.

For details, refer to [Basics on IOCNET \(I/O Carrier Network\) \(SCALEXIO Hardware Installation and Configuration !\[\]\(3dfb8d66e81160ad61421a3452093d1b_img.jpg\)](#)).

Gigalink communication The optical IOCNET connectors of a SCALEXIO Processing Unit or DS6001 Processor Board can also be used for Gigalink communication.

Gigalink is a dSPACE-specific communication bus that is used in PHS-bus-based real-time systems from dSPACE. Via Gigalink, a SCALEXIO system can interact via Gigalink with all dSPACE systems, such as dSPACE Simulator Mid-Size and dSPACE Simulator Full-Size, which are based on the PHS-bus technology.

Gigalink can be used for the following purposes:

- Connecting a SCALEXIO system to a PHS-bus-based system (i.e., modular system with a DS1006 or DS1007) to exchange data between the two unsynchronized real-time systems.

Refer to [Connecting a PHS-Bus-Based System \(SCALEXIO Hardware Installation and Configuration !\[\]\(de95854c7ee024cfadc48187bbb781b2_img.jpg\)](#)).

- Connecting two or more SCALEXIO systems to exchange data between *unsynchronized* real-time applications.

Refer to [How to Connect Another SCALEXIO System via Gigalink \(SCALEXIO Hardware Installation and Configuration !\[\]\(6059a5aa8b4ca7bb793408023d6c6e42_img.jpg\)](#)).

Related topics

Basics

[Connecting a PHS-Bus-Based System \(SCALEXIO Hardware Installation and Configuration !\[\]\(9c2e8d1b5bd77cb5c9f83b7a9cff79fd_img.jpg\)](#))

HowTos

[How to Connect Another SCALEXIO System via Gigalink \(SCALEXIO Hardware Installation and Configuration !\[\]\(f60b7a900783ac3fd531bfd9c111be6d_img.jpg\)](#))

Configuring a SCALEXIO System

Introduction

Before you can start experimenting, the SCALEXIO system must be configured. In the configuration process you can specify the signals required by your ECU or controlled system, implement the real-time model and download it to the SCALEXIO system.

Where to go from here

Information in this section

[Basics on Configuring a SCALEXIO System..... 44](#)

Before you start configuring a SCALEXIO system and implementing the real-time model, you should be familiar with the basic principles.

[Connecting Outputs of External Devices to Inputs of the SCALEXIO System..... 62](#)

The SCALEXIO system measures the signals generated by the external devices.

[Connecting Inputs of External Devices to Outputs of the SCALEXIO System..... 66](#)

The SCALEXIO system generates the signals required by the external devices.

[Configuring the SCALEXIO System for Data Communication..... 70](#)

The SCALEXIO system provides the bus interfaces (CAN, LIN, FlexRay) and serial interfaces for the external devices.

[Connecting External Devices to the Power Supply..... 85](#)

The SCALEXIO system can provide the battery voltage via power switch channels for external devices.

[Data Transmission Between dSPACE Systems..... 90](#)

When a SCALEXIO system is connected to a PHS-bus-based system (modular hardware based on a DS1006 or DS1007) or another SCALEXIO system, you can transmit data between the systems.

Basics on Configuring a SCALEXIO System

Introduction

Before you start configuring a SCALEXIO system and implementing the real-time model, you should be familiar with the basic principles.

Where to go from here

Information in this section

Basics on Configuring the Signal Chain.....	44
Before you start configuring the signal chain, you should be familiar with the basic principles.	
Basic Principles of Implementing the Real-Time Model.....	51
Before you start implementing the real-time model, you should be familiar with the basic principles.	

Basics on Configuring the Signal Chain

Introduction

Before you start configuring the signal chain, you should be familiar with the basic principles.

Where to go from here

Information in this section

Behavior Model and I/O Model.....	45
The real-time model consists of two parts: the behavior model and the I/O model.	
Physical Signal Chain and Logical Signal Chain.....	46
Shows how the signal paths are described in ConfigurationDesk.	
Basics of External Devices.....	47
External devices are connected to a dSPACE system at the I/O connector.	
Basics of Function Blocks.....	48
The functionality of the I/O model is specified by function blocks provided by ConfigurationDesk.	
Basics of Model Port Blocks.....	50
Model port blocks represent the interface between the I/O model built in ConfigurationDesk and the behavior model modeled in Simulink.	

Behavior Model and I/O Model

Introduction

The real-time model for the SCALEXIO system consists of two parts: the behavior model and the I/O model. In the build process, both model parts are used to create the real-time application which can be loaded to the SCALEXIO system afterwards.

Behavior model

The behavior model is modeled in Simulink®. For an HIL simulator, it simulates the behavior of the controlled system, for example, the vehicle dynamics. The behavior model is connected to the I/O model via model port blocks. In an RCP system, the behavior model contains the control algorithms that control the controlled system.

I/O model

The I/O model is configured in ConfigurationDesk. The I/O model contains all the functions to measure and generate the required I/O signals. The I/O model is represented as a logical signal chain which shows how the signals from the behavior model are mapped to the external devices. Refer to [Physical Signal Chain and Logical Signal Chain](#) on page 46. The I/O model is connected to the behavior model via model port blocks.

Interface between behavior model and I/O model

The behavior model and the I/O model are connected via model port blocks. There are model port blocks in ConfigurationDesk and in the behavior model in Simulink which correspond to each other. Model port blocks (ConfigurationDesk) are part of the I/O model. Model port blocks (behavior model) make the I/O signals available to the behavior model. The following illustration shows an example of a model port block (ConfigurationDesk) on the left and the corresponding block in Simulink on the right for a signal which is generated by the SCALEXIO system.



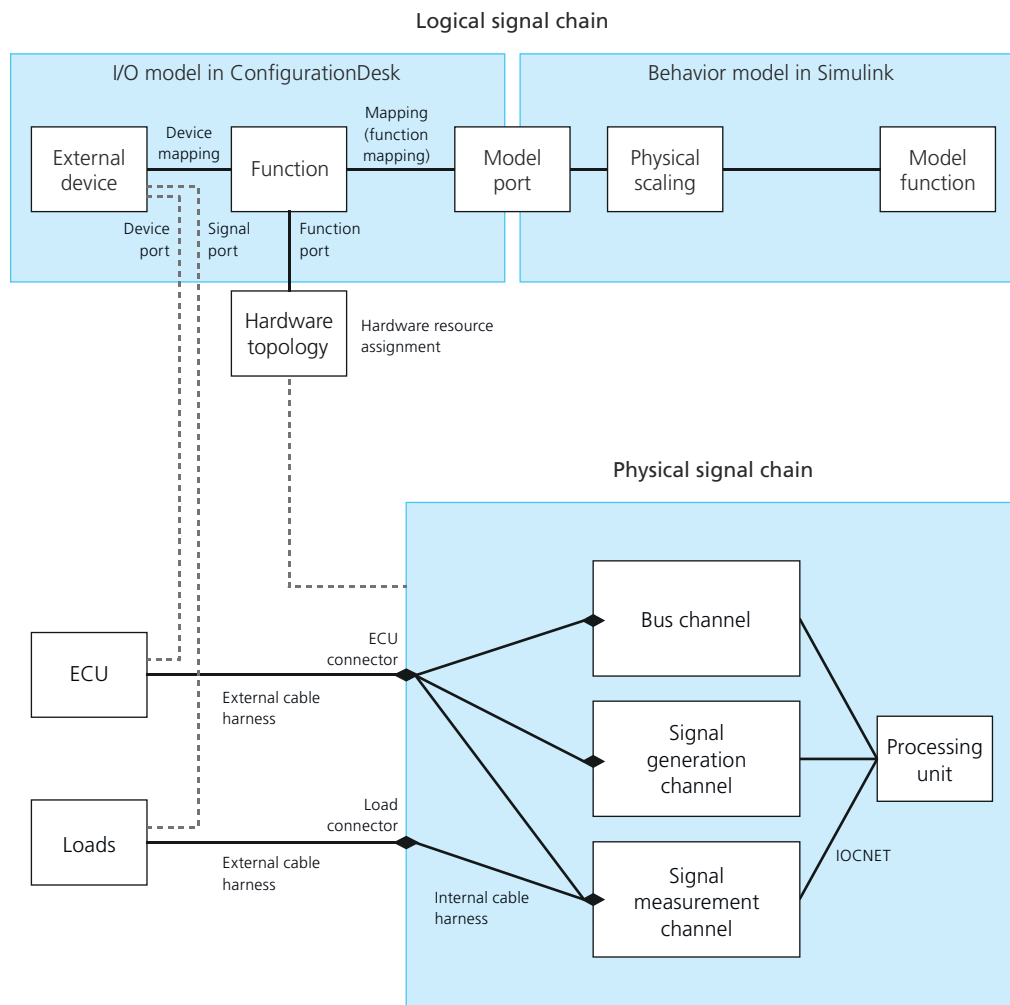
The following illustration shows an example of a model port block (ConfigurationDesk) on the left and the corresponding block in Simulink on the right for a signal which is measured by the SCALEXIO system.



Physical Signal Chain and Logical Signal Chain

Introduction

The I/O model is represented as a logical signal chain in ConfigurationDesk. The physical signal chain describes the actual connection between the external device and SCALEXIO system. The following illustration shows the logical signal chain when the SCALEXIO system is used to test an ECU.



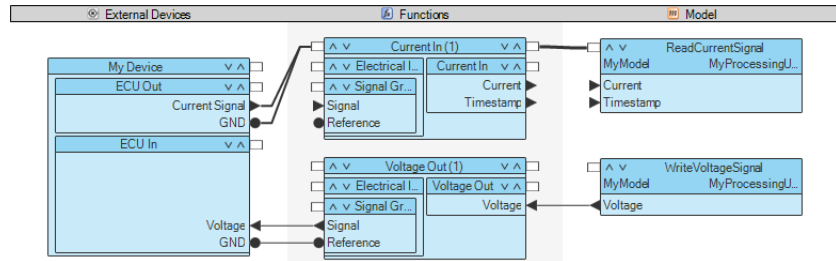
Physical signal chain

The physical signal chain describes the electrical wiring of external devices (ECU and loads, or the controlled system) to the I/O units and I/O boards of a SCALEXIO system. It includes the external cable harness, the I/O connector and the internal cable harness.

ConfigurationDesk can calculate the physical signal chain if the logical signal chain is specified and the real-time hardware is assigned to the function blocks.

Logical signal chain

The logical signal chain describes the complete chain of signals from the external device to the model I/O ports in the Simulink model in ConfigurationDesk. A logical signal chain consists of three parts, see the following illustration.



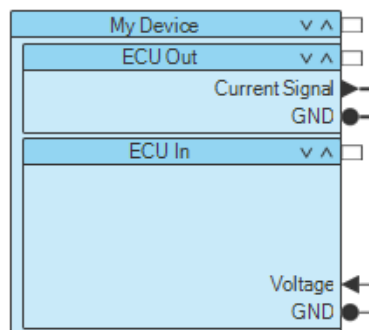
The three parts are:

- External device blocks representing the interfaces of the devices which are connected to the SCALEXIO system (ECU and external loads or the controlled system). Refer to [Basics of External Devices](#) on page 47.
- Function blocks representing the functions of the I/O model. Refer to [Basics of Function Blocks](#) on page 48.
- Model port blocks representing the interface from the I/O model to the behavior model. Refer to [Basics of Model Port Blocks](#) on page 50.

Basics of External Devices

Introduction

External devices are connected to a dSPACE system at I/O connectors. For an HIL simulator, external devices are the ECU to be tested and external loads. For an RCP system, external devices are the actuators and sensors of the controlled system.



External devices

External devices are represented as device blocks in the graphical window of ConfigurationDesk. External devices have ports which are the connection points for signal ports of function blocks when you build the logical signal chain.

Configuring device blocks

It is not required to configure device blocks for implementing the real-time model, but configuring device blocks will support your configuration process:

- You can structure the device in port groups in several hierarchies and ports, for example, ports can correspond to pins of the interface and port groups can correspond to connectors.
- You can specify the data direction and a physical attribute of a port to simplify the selection of signal ports when connecting the device port to function blocks.
- You can specify reference ports for a port.
- You can add a description to the port.
- You can specify expected loads for the out pins of an ECU device to simplify the selection of signal measurement channels.
- ConfigurationDesk can calculate the external cable harness according to the mapping of the device ports to the signal ports of the function blocks.
- You can specify your own ECU pin numbering for inclusion in the external cable harness which is calculated.
- You can enable or disable failure simulation for a device port and specify the allowed failure classes. You can transfer these settings to mapped function blocks.
- Configuration data of devices can be saved to a file and reused in other ConfigurationDesk applications.

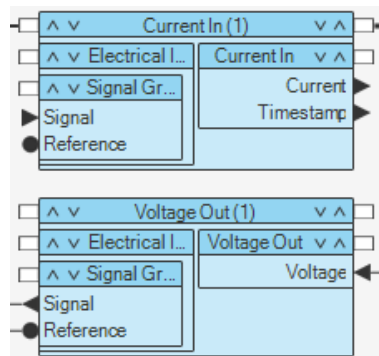
Detailed information

For details on implementing external devices, refer to [Specifying the External Device Interface \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(96cc62f861fdd6e50510c0224a756dff_img.jpg\)](#)).

Basics of Function Blocks

Introduction

The functionality of the I/O model is specified by function blocks provided by ConfigurationDesk.



Function blocks

Function blocks are used to implement the I/O model. The function blocks represent functions in ConfigurationDesk which are finally executed on the SCALEXIO system:

- Functions which generate signals
- Functions which measure signals
- Functions which connect the external device to simulated buses
- Functions which control the simulated battery voltage and power switches

For executing the function on the SCALEXIO system, suitable real-time hardware must be assigned to the function blocks (signal measurement channels, signal generation channels, bus channels, power switch channels, or battery simulation controller).

Function blocks have signal ports on the left side. The signal ports can be mapped to the external device. Function blocks have function ports on the right side. The function ports can be mapped to the model port blocks. The characteristics of the function blocks and their ports can be specified via properties.

Configuring function blocks

Several properties of a function block are user-configurable. The properties configure the model and electrical interface of the function block and also the code generation for the function. The properties which are available depend on the function block type.

The properties for the model interface describe the behavior of the function and its signals in the I/O model. You can specify:


- Parameters which set the behavior of the function.
- Initial values for the signals.
- When the initial values of the signals are used (at the first or at every start of the real-time application).
- Whether the values of the signals are saturated, and the saturation values.
- Which of the signals are used for automation.

The properties for the electrical interface describe the behavior of the SCALEXIO system. The values must be specified according to the signals of the external device:

- You can select a channel type which is used to execute the function. A channel type is a category of channels on a SCALEXIO I/O unit or I/O board that provide exactly the same physical characteristics.
- You can select channels of the SCALEXIO hardware which are used to execute the function. Several channels may be necessary depending on the function type and the maximum current.
- You can specify the maximum current and voltage required for the function.
- You can specify the rated current (trigger value) for the electronic fuses of the channels.
- You can enable or disable failure simulation for a channel and specify the allowed failure classes.

- If a signal measurement channel is selected, you can configure settings for the load.
 - You can specify whether an external load is connected to this channel.
 - You can specify a description of the load connected to this channel. ConfigurationDesk can compare the description when the function block is mapped to a device.
 - You can specify load rejection for this channel. Load rejection means that the load is disconnected during failure simulation. This protects a sensitive load.

Detailed information

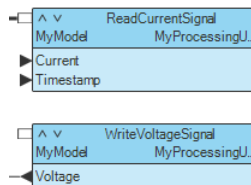
For details on the available functions, refer to [ConfigurationDesk I/O Function Implementation Guide](#) .

For details on using the function blocks, refer to [Implementing I/O Functionality \(ConfigurationDesk Real-Time Implementation Guide\)](#) .

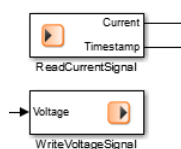
Basics of Model Port Blocks

Model port blocks

Model port blocks represent the interface between the I/O model built in ConfigurationDesk and the behavior model modeled in Simulink. The following illustration shows model port blocks in ConfigurationDesk.



The following illustration shows the corresponding model port blocks in Simulink.



Configuring model port blocks

Model port blocks are configured in Simulink. ConfigurationDesk can analyze the Simulink model to read the configuration data and update the model port blocks in ConfigurationDesk.

- You can configure the number of ports of the model port blocks.
- You can specify the names of the model port blocks and their ports.
- You can specify properties of the ports to adapt them to the function ports and the signals connected in the behavior model.

Detailed information

For details on configuring model port blocks in ConfigurationDesk, refer to [Specifying the Model Interface \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(c507f772dba2b921f86777f01218e570_img.jpg\)\)](#).

For details on configuring the model port blocks in Simulink, refer to [Creating the Interface of Behavior Models \(Model Interface Package for Simulink - Modeling Guide !\[\]\(4729e517bc6a7cd81c8025b9646574fb_img.jpg\)\)](#).

Basic Principles of Implementing the Real-Time Model

Introduction

Before you start implementing the real-time model, you should be familiar with the basic principles.

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

[Overview of the Configuration Process..... 52](#)

A real-time model of the system to simulate must be implemented using MATLAB/Simulink and ConfigurationDesk.

[Implementing the Behavior Model..... 53](#)

The behavior model is modeled in MATLAB/Simulink. It contains the model of the controlled system and bus communication.

[Creating a Project and a ConfigurationDesk Application..... 55](#)

All the configuration data is managed in projects and ConfigurationDesk applications.

[Building the Logical Signal Chain..... 57](#)

Gives an overview of building a logical signal chain in ConfigurationDesk

[Modeling Real-Time Applications and Tasks..... 58](#)

Gives an overview on modeling the tasks in your real-time application very flexibly. You can use predefined tasks provided by the behavior model, or you can create new tasks in ConfigurationDesk.

[Implementing FPGA Applications..... 58](#)

Gives an overview on how to implement FPGA applications for a SCALEXIO system. Using FPGA applications you can realize very fast closed feedback loops with low turnaround times because the simulation model is executed directly on the FPGA.

[Accessing the SCALEXIO System and Assigning Functions to Hardware Resources..... 59](#)

Every function requires resources of the SCALEXIO system for execution. To use functions for real-time simulation, they must be mapped to appropriate channels of SCALEXIO units or boards.

Building and Downloading the Real-Time Application.....	60
Describes how to build the real-time application for the SCALEXIO system	
Archiving the ConfigurationDesk Project and the Real-Time Model.....	61
A ConfigurationDesk project can be archived. This makes it possible to transfer the project to another PC or to add it to a version control system.	

Overview of the Configuration Process

Introduction

A real-time model of the system to simulate must be implemented. For a SCALEXIO system this is done in MATLAB/Simulink and in ConfigurationDesk. This topic provides an overview of the necessary steps.

Configuration process

The configuration process can be performed in several steps.

1. Implementing the behavior model
The behavior model is implemented in MATLAB/Simulink. The interface to the I/O model is implemented with the Model Port blocks. In addition, the behavior model can contain bus communication for a restbus simulation. ConfigurationDesk checks the Simulink model to get information on the model port blocks so that it can be appended to the logical signal chain.
2. Creating a project in ConfigurationDesk
ConfigurationDesk handles all the implementation data in projects and applications. A project can contain several applications, but only one can be active at the same time. An application contains all the data for an implementation task.
3. Building a logical signal chain
You can build the logical signal chain in an active ConfigurationDesk application. The logical signal chain contains device blocks, function blocks and model port blocks. The ports of the blocks are connected by mapping lines. The properties of the blocks can be set in the **Properties Browser**.
4. Configuring Real-Time Applications and Tasks (refer to [Modeling Real-Time Applications and Tasks](#) on page 58)
A real-time application contains periodic tasks and asynchronous tasks. ConfigurationDesk allows you to model the tasks in your real-time application very flexibly. You can use predefined tasks provided by the behavior model, or you can create new tasks in ConfigurationDesk.
5. Accessing the simulator
ConfigurationDesk has a Platform Manager to access the SCALEXIO system which you want to use. The hardware of the system is displayed in a

hardware topology. The hardware topology shows all the units and boards of the system including their channels.

6. Assigning functions to hardware resources

The channels can be assigned to the functions. When all the functions have been assigned to hardware channels, the real-time application can be built and downloaded. In addition, ConfigurationDesk can calculate the external cable harness (the connection of the SCALEXIO system to the external devices).

7. Building the real-time application

When the logical signal chain has been created and the function blocks have been assigned to channels of the hardware, you can build the real-time application. In the build process, the real-time application is generated from the behavior model and the I/O model. The real-time application can be downloaded to the SCALEXIO system for experimenting.

Related topics

Basics

Accessing the SCALEXIO System and Assigning Functions to Hardware Resources.....	59
Building and Downloading the Real-Time Application.....	60
Building the Logical Signal Chain.....	57
Creating a Project and a ConfigurationDesk Application.....	55
Implementing the Behavior Model.....	53
Modeling Real-Time Applications and Tasks.....	58

Implementing the Behavior Model

Introduction

The behavior model is modeled in MATLAB®/Simulink®. It contains the model of the controlled system and bus communication. This topic provides an overview of the modeling tasks in Simulink.

System target file


System target files determine the structure of code generation in MATLAB/Simulink. The system target file must fit the real-time processor you want to generate code for. For a SCALEXIO system, the system target file must be `dsrt.tlc`. The system target file is selected in the Configuration Parameters dialog on the Code Generation page.

Modeling input, outputs, and events

The interface between behavior and I/O model is modeled via Model Port blocks in Simulink. The model port blocks in your behavior model correspond to model port blocks in ConfigurationDesk. The Model Port Block Library for Simulink® Coder™ contains blocks for data inputs, data outputs and trigger asynchronous functions. For an overview on the implementation, refer to [Building the Logical Signal Chain](#) on page 57.


Modeling CAN communication

For making communication via the CAN bus available, several steps are necessary. One of them is modeling the CAN communication in Simulink using blocks from the RTI CAN MultiMessage Blockset. The RTI CAN MultiMessage Blockset lets you configure and handle a large number of CAN messages. All the incoming and outgoing messages of an entire CAN controller can be controlled by a single Simulink block. For an overview on the implementation, refer to [CAN Bus Connection](#) on page 71.

If you work without the RTI CAN MultiMessage Blockset, you can use ConfigurationDesk's Bus Manager to implement CAN communication in the I/O model. For details on the Bus Manager, refer to [ConfigurationDesk Bus Manager Implementation Guide](#) .

Modeling LIN communication

For making communication via the LIN bus available, several steps are necessary. One of them is modeling the LIN communication in Simulink using blocks from the RTI LIN MultiMessage Blockset. The RTI LIN MultiMessage Blockset lets you configure and handle a large number of LIN frames. All the incoming and outgoing frames of an entire LIN controller can be controlled by a single Simulink block. For an overview on the implementation, refer to [LIN Bus Connection](#) on page 77.

If you work without the RTI LIN MultiMessage Blockset, you can use ConfigurationDesk's Bus Manager to implement LIN communication in the I/O model. For details on the Bus Manager, refer to [ConfigurationDesk Bus Manager Implementation Guide](#) .


Modeling FlexRay communication

For making communication via the FlexRay bus available, you can connect its FlexRay channels to the SCALEXIO system. Blocks of the FlexRay Configuration Blockset must be configured and inserted in the Simulink model. For an overview on the implementation, refer to [FlexRay Bus Connection](#) on page 74.

Modeling with ASM libraries

You can use dSPACE Automotive Simulation Models (ASM libraries) for modeling the behavior model of an engine and/or vehicle dynamics in Simulink. For an overview, refer to [Working with ASM and ModelDesk](#) on page 95.

Generating motion data

MotionDesk can visualize the movement of objects, for example, a vehicle, in a graphical environment. To be able to move the objects, MotionDesk requires a data stream with motion data. The motion data is calculated in the Simulink model and sent to MotionDesk via the Ethernet network. Calculating and transferring the motion data is implemented using the MotionDesk Blockset. For details, refer to [MotionDesk Calculating and Streaming Motion Data](#) .

Creating a Project and a ConfigurationDesk Application

Introduction

All the configuration data is managed in projects and ConfigurationDesk applications. Before you can work with ConfigurationDesk, you must create a project and a ConfigurationDesk application.

ConfigurationDesk license

ConfigurationDesk - Implementation Version is license-protected. A Basic and a Function block license are required for installing and using the full-featured version of ConfigurationDesk - Implementation Version. The function block licenses are available in different sizes for different numbers of instantiated function blocks in your active ConfigurationDesk application. ConfigurationDesk counts the instantiated function blocks in the signal chain of your active application at run time and displays it in the Licenses dialog. For details on the licenses, refer to [Required Licenses \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(83f22ed94ec5517769dd76d702c6bfd8_img.jpg\)](#)).

Starting ConfigurationDesk

ConfigurationDesk is started in the same way as other Windows-based software. However, there are some points to note, refer to [Starting ConfigurationDesk \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(642aa997563f9a325b310230bb5078b7_img.jpg\)](#)).

Projects

A project centralizes all the items of a task in ConfigurationDesk. These can be:

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

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

Project 1

Project-specific files

Application 1.1

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

Application 1.2

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

Project 2

Project-specific files

Application 2.1

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

Application 2.2

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

Application 2.3

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

For details on managing projects in ConfigurationDesk, refer to [Managing ConfigurationDesk Projects \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(4146d17f71dced09c6ad789cacceaa6d_img.jpg\)](#)).

ConfigurationDesk application

A ConfigurationDesk application is the basis for carrying out tasks in ConfigurationDesk. It must always be included in a project, i.e. it cannot exist separately. A project can contain one or more ConfigurationDesk applications, each representing a specific task. You can work with only one application – the active one – at a time.

For details on working with ConfigurationDesk applications, refer to [Managing ConfigurationDesk Applications \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(9a795c4c0c43d0827b424565265fc8e6_img.jpg\)](#)).

Building the Logical Signal Chain

Introduction

The logical signal chain contains device blocks, function blocks and model port blocks. The ports of the blocks are connected via mapping lines.

Working in ConfigurationDesk

The logical signal chain is built in the working area of ConfigurationDesk. In the working area you can work with graphical windows or a table window. Graphical windows show the blocks and mapping lines, the table window lists all the data. Data which is displayed is specified in working views. The **Global** working view contains all the data of the configuration, which can be confusing in large configurations. You can specify your own working view containing only a subset of data for clearer focus. Refer to [Handling the Signal Chain in Working Views \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(d66ff64371a51729ac8c1cdaa685ba6f_img.jpg\)](#)).

Configuring device blocks

To be able to use device blocks, you must create a device topology or load an existing one. You can specify the properties of the device ports to adapt them to the characteristics of the ECU pins. You can drag the device ports from the External Device Browser to the graphical window for connecting them to the signal chain. Refer to [Specifying the External Device Interface \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(003082e50e3009141f59bd5df831749f_img.jpg\)](#)).

Configuring function blocks

The Function Browser in ConfigurationDesk provides all the function types which are available for the logical signal chain. Which function types can actually be used depends on the boards which are installed in the SCALEXIO system. From the Function Browser, you can drag function types to the graphical window to get instances of the function types, represented as function blocks. For details on using and configuring the function blocks, refer to [Implementing I/O Functionality \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(faf942dc3e59ce8eb64b4ac481eca7e0_img.jpg\)](#)).

Configuring model port blocks

There are two ways of obtaining model port blocks in ConfigurationDesk and the Simulink model.

- Starting from function blocks, ConfigurationDesk can extend the logical signal chain using suitable model port blocks. Afterwards ConfigurationDesk can create an interface model containing the corresponding model port blocks for Simulink. These blocks can be copied to the Simulink model and connected to the signals of the behavior model.
- Starting in Simulink, you can use model port blocks from the Model Port Block Library and connect them to the signals of your behavior model. ConfigurationDesk analyzes the Simulink model and creates the model topology containing all the model port blocks that were found. You can drag the model port blocks to the graphical window from the Model Browser.


Refer to [Specifying the Model Interface \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(95b425611cbd2b8716a140cf67c81822_img.jpg\)](#)).

Modeling Real-Time Applications and Tasks

Introduction

A real-time application contains periodic tasks and asynchronous tasks. ConfigurationDesk allows you to model the tasks in your real-time application very flexibly. You can use predefined tasks provided by the behavior model, or you can create new tasks in ConfigurationDesk.

Basics

Before you can configure the real-time application and tasks, you must be familiar with some basic information. Refer to [Terms and Definitions for Building Executable Applications](#) (ConfigurationDesk Real-Time Implementation Guide ).

Modeling real-time applications and tasks

In ConfigurationDesk, the term executable application is used for a real-time application.

ConfigurationDesk allows you to model the tasks in your executable application very flexibly. You can use predefined tasks provided by the behavior model, or you can create new tasks in ConfigurationDesk.

Modeling tasks comprises the following steps:


1. Creating a task
2. Assigning an event to the task
3. Assigning one or more runnable functions to the task
4. Configuring task properties

Refer to [Introduction to Modeling Executable Applications and Tasks](#) (ConfigurationDesk Real-Time Implementation Guide ).

Configuring tasks

Configuring tasks in ConfigurationDesk means specifying several task properties.

To avoid task overruns due to a cold cache during the first execution of tasks, you can configure the start-up behavior of periodic tasks.

Refer to [Configuring Tasks in ConfigurationDesk](#) (ConfigurationDesk Real-Time Implementation Guide ).

Implementing FPGA Applications

Introduction

You can implement FPGA applications for a SCALEXIO system. Using FPGA applications you can realize very fast closed feedback loops with low turnaround times because the simulation model is executed directly on the FPGA.

Required hardware

To execute FPGA applications, the SCALEXIO system must have an FPGA board. An FPGA board is inserted in a slot unit, an external I/O unit, or SCALEXIO LabBox. It consists of a base board (DS2655, DS6601, or DS6602 FPGA Base Board) and one or more DS2655M1 I/O Modules or DS2655M2 I/O Modules.

For details on the FPGA board, refer to [Hardware for FPGA Applications \(SCALEXIO Hardware Installation and Configuration !\[\]\(4729e517bc6a7cd81c8025b9646574fb_img.jpg\)](#)).

Required implementation software

To implement FPGA applications, the RTI FPGA Programming Blockset must be installed. This is a Simulink blockset for integrating an FPGA application into a dSPACE system. It provides RTI blocks for implementing the interface between the FPGA and the I/O of the FPGA module, and the interface between the FPGA board and the processing unit or processor board.

For details on working with the blockset, refer to [RTI FPGA Programming Blockset Guide !\[\]\(cbe2492b119e39e02a1dab2af4a4b296_img.jpg\)](#).

Workflow

For an overview of the workflow for implementing FPGA applications, refer to [Typical Workflow \(RTI FPGA Programming Blockset Guide !\[\]\(3e2231b1ad3ca8da8658228c00dd08e0_img.jpg\)](#)).

Accessing the SCALEXIO System and Assigning Functions to Hardware Resources

Introduction

Every function requires resources of the SCALEXIO system for execution. Different types of I/O units and I/O boards can be installed in a SCALEXIO system. The I/O units and I/O boards provide the resources for the function to be executed.

With the flexibility of ConfigurationDesk and the dSPACE hardware architecture, the execution of a function is not tied to a fixed wired channel. Function blocks can be assigned to any hardware resource which is suitable for the functionality.

Accessing the SCALEXIO system

Before you can assign the functions to hardware channels of the SCALEXIO system, ConfigurationDesk must access the SCALEXIO system to get information on its installed hardware. The information is stored in the hardware topology.

So that you can access the SCALEXIO system, it must be connected to the host PC via the Ethernet. Refer to [Setting up the Connection to the Host PC \(SCALEXIO Hardware Installation and Configuration !\[\]\(2bae76de5ebbd5c4d7d47162f1673734_img.jpg\)](#)).

For information on how to access the SCALEXIO system, refer to [Managing Real-Time Hardware \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(b64b40baaee5acddc1eab8538ba84754_img.jpg\)](#)).

If the real SCALEXIO system is not physically available, you can customize the hardware topologies of applications either by creating new hardware topologies from scratch or by extending existing ones. Thus, you can generate hardware topologies which do not depend on a particular hardware system or on an existing HTF file. You can then perform tasks such as implementing and building a real-time application for a SCALEXIO system which is not yet physically available. However, if you want to download and execute the real-time application, you need a real SCALEXIO system.

Assigning hardware resources

To execute functions of the I/O model, a channel type is selected in the function blocks. Channel types are categories of channels on a SCALEXIO I/O unit or I/O board that provide exactly the same characteristics.

When ConfigurationDesk has the information on the hardware of the SCALEXIO system, you can assign the function blocks to hardware channels of the selected channel type. The function blocks require one or more hardware channels depending on their configuration and type.

For details on assigning the hardware resources, refer to [Assigning Hardware Resources to Function Blocks \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(23d9fc146e83b5c3013cfa32c784f8d5_img.jpg\)](#)).

Connecting external devices to the SCALEXIO system

The wiring between a slot unit where the I/O boards are installed and the I/O connectors of the SCALEXIO system is fixed. The wiring of the channels of an I/O unit is also fixed. The assignment of functions to the hardware channels therefore determines the I/O mapping of the I/O connectors. The corresponding pin numbers are displayed in the Properties Browser when a function port of a function block is selected. ConfigurationDesk can calculate the wiring which connects the external devices to the SCALEXIO system (external cable harness). The information is exported as an Excel file.

For details on calculating and exporting the external cable harness, refer to [How to Export Wiring Information for an External Cable Harness \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(05be7c7a8995decd503647c99211f7c2_img.jpg\)](#)).

For details on configuring the external cable harness, refer to [Connecting an ECU to the SCALEXIO System \(SCALEXIO Hardware Installation and Configuration !\[\]\(aa53ad6fea213b8b2226d3077e30533a_img.jpg\)](#)).

Building and Downloading the Real-Time Application

Introduction

When the behavior model and I/O model are modeled, you can build the real-time application.

Building the real-time application

The build process must be started by ConfigurationDesk. You can specify several options to optimize the build results. Refer to [Building Real-Time Applications \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(2e897e890e69d81eae4503a8342c36b0_img.jpg\)](#)).

Downloading the real-time application

The real-time application can automatically be downloaded to the SCALEXIO system after the build process. If you have specified this option, the real-time application is downloaded to the SCALEXIO system after a successful build process. It is also possible to download the real-time application using the experiment or test software. Refer to [Experimenting with a SCALEXIO System](#) on page 103.

Archiving the ConfigurationDesk Project and the Real-Time Model

Introduction

A ConfigurationDesk project can be archived. This makes it possible to transfer the whole project to another PC or to add it to a version control system.

Creating and opening a ZIP archive

You can create a backup of the ConfigurationDesk project. When you backup the project, it is saved in a ZIP archive. Then you can transfer it easily to another PC or add it into a version control system. Refer to [How to Back up and Transfer a Project \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(8bba887393ca45b761e5cb49e755e762_img.jpg\)](#)).

Tip

Add the real-time model to the ConfigurationDesk project. When you create a backup project afterwards, the real-time model is also included in the ZIP archive. For details, refer to [Handling ConfigurationDesk Projects and Applications \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(47734e4656765d20df4fdbd5b7aff048_img.jpg\)](#)).

Connecting Outputs of External Devices to Inputs of the SCALEXIO System

Introduction

The SCALEXIO system measures the signals generated by the external devices. You can connect loads to the outputs of the external devices to obtain an authentic current.

Where to go from here

Information in this section

[Hardware for Signal Measurement..... 62](#)

To measure signals of the external devices, the SCALEXIO system must contain an I/O board or an I/O unit with signal measurement channels.

[Function Block Types for Signal Measurement..... 63](#)

To measure signals coming from the external devices, ConfigurationDesk provides different function blocks.

[Connecting Loads to Outputs of External Devices..... 64](#)

Some SCALEXIO I/O hardware allow to connect loads to the outputs of the external devices so that an authentic current can flow.

Hardware for Signal Measurement

Introduction

To measure signals of the external devices, the SCALEXIO system must contain an I/O board or an I/O unit with signal measurement channels.

SCALEXIO rack

You can use the following hardware components in a SCALEXIO rack:

- DS2680 I/O Unit
- DS2690 Digital I/O Board
- DS2601 Signal Measurement Board

SCALEXIO rack or SCALEXIO AutoBox/LabBox







You can use the following hardware components in a SCALEXIO rack or SCALEXIO AutoBox/LabBox:





- DS6101 Multi-I/O Board
- DS6121 Multi-I/O Board
- DS6201 Digital I/O Board
- DS6202 Digital I/O Board
- DS6221 A/D Board

Related topics**Basics**


HighFlex I/O Boards.....	21
MultiCompact I/O Units and Boards.....	22
SCALEXIO I/O Boards.....	18

Function Block Types for Signal Measurement

Introduction	To measure signals coming from the external devices, ConfigurationDesk provides different function blocks.
Current In	Allows you to measure and digitize analog currents. For details, refer to Current In (ConfigurationDesk I/O Function Implementation Guide ).
Digital Pulse Capture	Allows you to convert signals coming from the ECU to digital pulses. For details, refer to Digital Pulse Capture (ConfigurationDesk I/O Function Implementation Guide ).
Injection/Ignition Current In	Allows you to measure currents of injection and ignition signals generated by a real ECU. For details, refer to Injection/Ignition Voltage In (ConfigurationDesk I/O Function Implementation Guide ).
Injection/Ignition Voltage In	Allows you to measure voltages of injection and ignition signals generated by a real ECU. For details, refer to Injection/Ignition Voltage In (ConfigurationDesk I/O Function Implementation Guide ).
Multi Bit In	Allows you to measure digital signals (voltages and currents) coming from an ECU. The signals can be sent to the behavior model and you can use them, for example, to generate interrupts for the behavior model. For details, refer to Multi Bit In (ConfigurationDesk I/O Function Implementation Guide ).
PWM/PFM In	Allows you to measure one-phase square-wave input signals (voltages and currents). It lets you calculate the signal's frequency, the duty cycle, or both. For details, refer to PWM/PFM In (ConfigurationDesk I/O Function Implementation Guide ).

SENT In	Allows you to receive SENT (single edge nibble transmission) messages. For details, refer to SENT In (ConfigurationDesk I/O Function Implementation Guide ).
Triggered Current In	Allows you to measure and digitize analog currents according to a trigger event. For details, refer to Triggered Current In (ConfigurationDesk I/O Function Implementation Guide ).
Voltage In	Allows you to measure and digitize analog voltages. For details, refer to Voltage In (ConfigurationDesk I/O Function Implementation Guide ).
Related topics	<p>Basics</p> <div> Overview of SCALEXIO Channel Types (SCALEXIO Hardware Installation and Configuration ) </div>

Connecting Loads to Outputs of External Devices

Introduction	Some SCALEXIO I/O hardware allow to connect loads to the outputs of the external devices so that an authentic current can flow.
Loads for ECU test	ECUs monitor the current of the actuators, for example, to ensure that they work properly. Therefore, in a hardware-in-the-loop simulation loads must be connected to ECU outputs to obtain the necessary current. The loads are connected to the ECU output via the signal measurement channels of the SCALEXIO system so that the SCALEXIO system can measure the signal of the ECU. Loads can be installed inside the SCALEXIO system (internal loads) or connected to a LOAD connector on the front of the SCALEXIO system (external loads). The slots for internal loads and LOAD connectors are connected in parallel.
Internal load	<p>Internal loads are mounted on boards inside the SCALEXIO system. They are connected to the ECU outputs on the board. Because the available space is restricted, their physical dimensions and power dissipation are limited.</p> <p>For information on how to mount a load in a DS2680 I/O Unit, refer to How to Connect Internal Loads (DS2680-IL) (SCALEXIO Hardware Installation and Configuration ).</p>

For information on how to mount a load on a DS2601 Signal Measurement Board, refer to [How to Connect Internal Loads \(DS2601\) \(SCALEXIO Hardware Installation and Configuration !\[\]\(c8d96c8885d3000a912c2582004aed63_img.jpg\)](#)).

For information on how to handle internal loads in ConfigurationDesk, refer to [Details on Handling Internal Loads \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(919a2cb85b99741a73c0c31a427236a8_img.jpg\)](#)).

External loads

External loads are connected to LOAD connectors of the SCALEXIO system. This connects them to the signal measurement boards installed in the SCALEXIO system. Because the loads are outside of the SCALEXIO system, there are no restrictions on their dimensions. For example, you can use an original actuator. The power consumption of the loads is limited only by the power which the ECU can deliver via the SCALEXIO system.

For information on how to handle external loads in ConfigurationDesk, refer to [Details on Handling External Loads \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(d66ff64371a51729ac8c1cdaa685ba6f_img.jpg\)](#)).

Load rejection

You can disconnect internal and external loads during failure simulation. This can prevent sensitive loads from damage. Load rejection is not possible in some combinations, for example, if you want to simulate a failure between two signal measurement channels. If you specify load rejection and it is not possible to disconnect the load, failure simulation is not allowed for the channel concerned. For details, refer to [Basics on Load Rejection \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(003082e50e3009141f59bd5df831749f_img.jpg\)](#)).

Connecting Inputs of External Devices to Outputs of the SCALEXIO System

Introduction The SCALEXIO system generates the signals required by the external devices.

Where to go from here

Information in this section

[Hardware for Signal Generation.....](#) 66

To provide signals for the external devices, the SCALEXIO system must contain an I/O board or an I/O unit with signal generation channels.

[Function Block Types for Signal Generation.....](#) 67

To generate signals for the external devices, ConfigurationDesk provides different function blocks.

Hardware for Signal Generation

Introduction To provide signals for the external devices, the SCALEXIO system must contain an I/O board or an I/O unit with signal generation channels.

SCALEXIO rack

You can use the following hardware components in a SCALEXIO rack:

- DS2680 I/O Unit
- DS2690 Digital I/O Board
- DS2621 Signal Generation Board

SCALEXIO rack or SCALEXIO AutoBox/LabBox

You can use the following hardware components in a SCALEXIO rack or SCALEXIO AutoBox/LabBox:

- DS6101 Multi-I/O Board
- DS6121 Multi-I/O Board
- DS6201 Digital I/O Board
- DS6202 Digital I/O Board
- DS6241 D/A Board

Related topics**Basics**

HighFlex I/O Boards.....	21
MultiCompact I/O Units and Boards.....	22
SCALEXIO I/O Boards.....	18

Function Block Types for Signal Generation

Introduction

To generate sensor signals for the external devices, ConfigurationDesk provides different function blocks.

Crank/Cam Current Sink

Allows you to simulate current crankshaft and camshaft sensor signals of active sensors (including direction-sensitive (reverse) crankshaft sensors). For details, refer to [Crank/Cam Voltage Out \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(3e2231b1ad3ca8da8658228c00dd08e0_img.jpg\)](#)).

Crank/Cam Digital Out

Allows you to simulate digital crankshaft and camshaft sensor signals of active sensors (including direction-sensitive (reverse) crankshaft sensors). For details, refer to [Crank/Cam Voltage Out \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(870f5d5e9c0d57485634be3ecf52f3ca_img.jpg\)](#)).

Crank/Cam Voltage Out










Allows you to simulate crankshaft and camshaft sensor signals of passive sensors. For details, refer to [Crank/Cam Voltage Out \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(0d5ec72f61334709c3fc9450209b754f_img.jpg\)](#)).





Current Sink

Allows you to simulate sensors which provide a variable analog current. For details, refer to [Current Sink \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(7d1d6890825e83a6a4a51febe2dcc7f3_img.jpg\)](#)).

Digital Incremental Encoder Out

Allows you to simulate the signals provided by a rotary and a linear incremental encoder, for example, to analyze position changes. You can select whether the output should be in switch mode, push-pull mode, etc. The function requires at least 3 signal generation channels, in push-pull mode it requires at least 6 channels. For details, refer to [Digital Incremental Encoder Out \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(b64b40baaee5acddc1eab8538ba84754_img.jpg\)](#)).

Digital Pulse Out	Allows you to generate a digital pulse with each sampling step of the behavior model tasks or with each trigger event of another function block. For details, refer to Digital Pulse Out (ConfigurationDesk I/O Function Implementation Guide ).
Knock Signal Out	Allows you to simulate knock sensor signals. For details, refer to Knock Signal Out (ConfigurationDesk I/O Function Implementation Guide ).
Lambda DCR	Allows you to simulate wide-band lambda probes with direct current regulation (DCR). For details, refer to Lambda Probe Simulation (ConfigurationDesk I/O Function Implementation Guide ).
Lambda NCCR	Allows you to simulate wide-band lambda probes with Nernst-controlled current regulation (NCCR). For details, refer to Lambda Probe Simulation (ConfigurationDesk I/O Function Implementation Guide ).
Multi Bit Out	Allows you to output digital signals coming from the behavior model either bit-by-bit or in word format. You can select whether the output should be in switch mode, push-pull mode, etc. In push-pull mode the function requires at least 2 signal generation channels. For details, refer to Trigger In (ConfigurationDesk I/O Function Implementation Guide ).
Potentiometer Out	Allows you to simulate adjustable resistors such as sensors for angular and linear motion. It requires at least 2 signal generation channels. For details, refer to Potentiometer Out (ConfigurationDesk I/O Function Implementation Guide ).
PWM/PFM Out	Allows you to generate pulse-width modulated (PWM) or pulse-frequency modulated (PFM) signals, for example, to simulate electrical motors which deliver control signals as digital signal patterns. For details, refer to PWM/PFM Out (ConfigurationDesk I/O Function Implementation Guide ).
Resistance Out	Allows you to simulate resistance such as resistance temperature devices (RTD). For details, refer to Resistance Out (ConfigurationDesk I/O Function Implementation Guide ).
Voltage Out	Allows you to output analog voltages, for example, to simulate throttle position sensors. For details, refer to Voltage Out (ConfigurationDesk I/O Function Implementation Guide ).

Wavetable Current Sink	Allows you to output analog currents whose values are specified in a table, for example, to simulate signals of analog wheel speed sensors. For details, refer to Wavetable Current Sink (ConfigurationDesk I/O Function Implementation Guide ).
Wavetable Digital Out	Allows you to simulate digital signals whose values are specified in a table, for example, to simulate signals of digital wheel speed sensors. You can select whether the output should be in switch mode, push-pull mode, etc. In push-pull mode the function requires at least 2 signal generation channels. For details, refer to Wavetable Current Sink (ConfigurationDesk I/O Function Implementation Guide ).
Wavetable Voltage Out	Allows you to output analog voltages whose values are specified in a table, for example, to simulate signals of inductive engine speed sensors. For details, refer to Wavetable Current Sink (ConfigurationDesk I/O Function Implementation Guide ).
Wheelspeed Out	Allows you to simulate the signals provided by active wheel speed sensors, for example, to simulate sensors supporting very low rotational speed. You can simulate different sensor types and simulate failures for the output signal. For details, refer to Wheelspeed Out (ConfigurationDesk I/O Function Implementation Guide ).

Related topics**Basics**

[Overview of SCALEXIO Channel Types \(SCALEXIO Hardware Installation and Configuration !\[\]\(e2376d476d06eb31946dc01a69a4403a_img.jpg\)](#))

Configuring the SCALEXIO System for Data Communication

Introduction

The SCALEXIO system provides the bus interfaces (CAN, LIN, FlexRay) and serial interfaces for the external devices.

Where to go from here

Information in this section

[Hardware for Communication..... 70](#)

To be able to transfer data via a bus connection, serial interface, or Ethernet, the SCALEXIO system must contain appropriate hardware.

[CAN Bus Connection..... 71](#)

To communicate to and from the external devices via a CAN bus, you can connect their CAN channels to the SCALEXIO system.

[FlexRay Bus Connection..... 74](#)

To communicate to and from the external devices via a FlexRay bus, you can connect their FlexRay channels to the SCALEXIO system.

[LIN Bus Connection..... 77](#)

To communicate to and from the external devices via a LIN bus, you can connect their LIN channels to the SCALEXIO system.

[Serial Interface Connection..... 80](#)

To connect external devices to the SCALEXIO system via a serial interface, you must implement the corresponding communication in the real-time model.

[Waking up SCALEXIO AutoBoxes/LabBoxes via Bus Signals..... 82](#)

A SCALEXIO AutoBox/LabBox with a DS6001 Processor Board can change its operating mode from standby to normal operating mode when it receives a dedicated wake-up message or frame via a CAN or LIN bus.

Hardware for Communication

Introduction

To be able to transfer data via a bus connection, serial interface, or Ethernet, the SCALEXIO system must contain appropriate hardware.

SCALEXIO rack

You can use the following hardware components in a SCALEXIO rack:

- DS2672 Bus Module (only in a DS2680 I/O Unit)
- DS2671 Bus Board

SCALEXIO Processing Unit

You can use the following hardware components in a processing unit:

- DS6331-PE Ethernet Board
- DS6333-PE Automotive Ethernet Board
- DS6334-PE Ethernet Board
- DS6336-PE Ethernet Board

SCALEXIO rack or SCALEXIO AutoBox/LabBox

You can use the following hardware components in a SCALEXIO rack or SCALEXIO AutoBox/LabBox:

- DS6301 CAN/LIN Board
- DS6311 FlexRay Board
- DS6321 UART Board
- DS6341 CAN Board
- DS6342 CAN Board
- DS6351 LIN Board

SCALEXIO AutoBox/LabBox

You can use the following hardware components in an extended I/O slot of SCALEXIO AutoBox/LabBox:

- DS6333-CS Automotive Ethernet Board
- DS6335-CS Ethernet Board


Related topics**Basics**

HighFlex I/O Boards.....	21
I/O Boards for Extended I/O Slots.....	22
MultiCompact I/O Units and Boards.....	22
SCALEXIO I/O Boards.....	18
SCALEXIO Processing Hardware.....	17

CAN Bus Connection

Introduction

To communicate to and from the external devices via a CAN bus, you can connect their CAN channels to the SCALEXIO system.

The information in this topic applies only if you model CAN communication using the RTI CAN MultiMessage Blockset. For information on implementing CAN communication using ConfigurationDesk's Bus Manager, refer to [ConfigurationDesk Bus Manager Implementation Guide](#) .

Restbus simulation

External devices can be members of a CAN bus. The SCALEXIO system can replace all the other bus members which are normally connected to them. Thus, there is no difference in the CAN communication from the point of view of the external devices. This functionality is called "restbus simulation".

For specific purposes, it is useful to replace a simulated bus member by the real bus member. Either a bus member is simulated by the SCALEXIO system or the real bus member is connected to the bus. To minimize the number of real-time applications required for the simulation, it is possible to switch off a simulated bus member without compiling the real-time application again. Otherwise the SCALEXIO system would require hundreds of different real-time applications.

Simulating errors

You can simulate message errors and test the reaction of the external devices. You can manipulate message IDs and lengths, detect checksums and cycle time misbehavior, and observe message behavior.

You can also simulate errors in the physical layer of the CAN bus. The bus lines can be broken or switched to the lines of other signals, battery voltage or ground (see [Simulating Electrical Errors in the Wiring](#) on page 123).

Preconditions

Hardware: The SCALEXIO system must contain a bus module or board with one of the following channel types:

- *CAN 1* (available on a DS2672 Bus Module)
- *CAN 2* (available on a DS6301 CAN/LIN Board, a DS6341 CAN Board, and a DS6342 CAN Board)
- *Bus 1* (available on a DS2671 Bus Board)

Software: The RTI CAN MultiMessage Blockset for implementing a CAN communication must be installed.

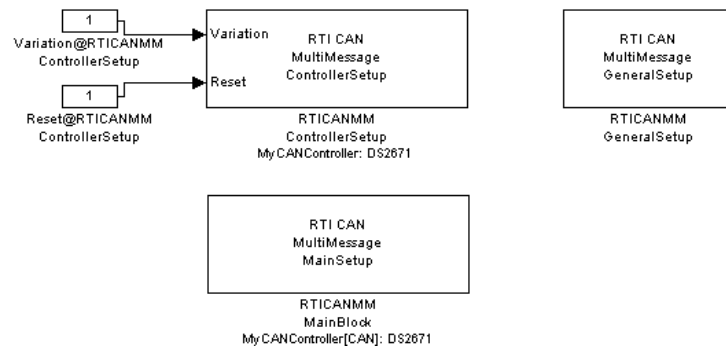
Workflow

This workflow shows all the steps necessary to connect the ECU to a simulated CAN communication.

1. Modeling the CAN communication in Simulink.

The CAN communication must be modeled in Simulink using the RTI CAN MultiMessage Blockset. For details on modeling the CAN communication, refer to [Modeling a CAN Bus Interface \(Model Interface Package for Simulink - Modeling Guide !\[\]\(e3f255517d37bb309a3a931ec4849e6a_img.jpg\)](#)).

The following illustration shows an example of a Simulink model containing the blocks of the RTI CAN MultiMessage Blockset.



2. Implementing the CAN communication in ConfigurationDesk.

To execute the modeled CAN communication on the SCALEXIO system, you must implement the CAN communication in ConfigurationDesk. In general, this comprises the following steps:

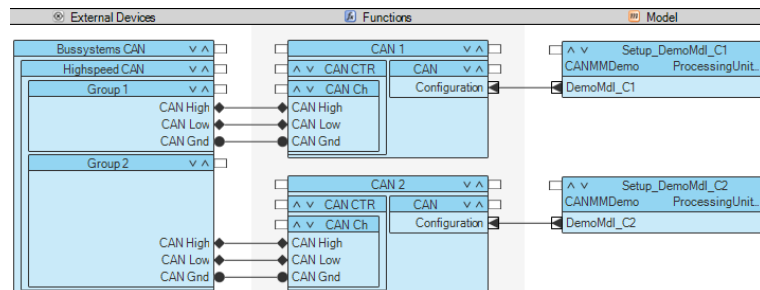
1. Analyzing the Simulink model in ConfigurationDesk.

When you do this, Configuration Port blocks (i.e., specific model port blocks) are created that provide the CAN bus settings which you modeled in the Simulink model to ConfigurationDesk.

2. Building the signal chain in ConfigurationDesk.

You must map the Configuration Port blocks to CAN function blocks. If you have specified a device for your ECU, you can additionally map the CAN function blocks to the related device blocks.

The following illustration shows an example of a signal chain with one device block, two CAN function blocks, and two Configuration Port blocks.



3. Assigning suitable hardware resources to the CAN function blocks to access the required bus modules or boards of the SCALEXIO system.

4. Configuring the CAN function blocks according to your requirements and depending on the assigned hardware resources.

Depending on the configuration, function ports are added to the CAN function blocks. To use the function ports in the Simulink model, you must map them to model port blocks, and synchronize the model interfaces in ConfigurationDesk and Simulink.

For basic information and details on the individual steps, refer to [Building the Signal Chain for CAN Bus Communication \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(3dfb8d66e81160ad61421a3452093d1b_img.jpg\)](#)).

3. Connecting the CAN bus lines of the external devices and the SCALEXIO system.

Assigning the function to a hardware resource specifies the ECU connector pins that the signals are routed to. You can get the pin numbers in ConfigurationDesk in the Properties Browser when the signal port is selected, refer to [How to Access Hardware Properties \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(0f848bbd71cef6b345273b16f905912a_img.jpg\)](#)).

FlexRay Bus Connection

Introduction

To communicate to and from the external devices via a FlexRay bus, you can connect their FlexRay channels to the SCALEXIO system.

Restbus simulation

External devices can be members of a FlexRay bus. The SCALEXIO system can replace all the other bus members which are normally connected to them. Thus, there is no difference in the FlexRay communication from the point of view of the external devices. This functionality is called “restbus simulation”.

Simulating errors

You can simulate PDU errors and test the reaction of your ECU. For example, you can manipulate the payload length of the TX PDU, or manipulate the values of update bits.

You can also simulate errors in the physical layer of the FlexRay bus. The bus lines can be broken or switched to the lines of other signals, battery voltage or ground (see [Simulating Electrical Errors in the Wiring](#) on page 123).

Preconditions

Hardware: The SCALEXIO system must contain a bus module or board with *FlexRay 1* channel types (available on a DS2672 Bus Module), *FlexRay 2* channel types (DS6311 FlexRay Board), or *Bus 1* channel types (DS2671 Bus Board).

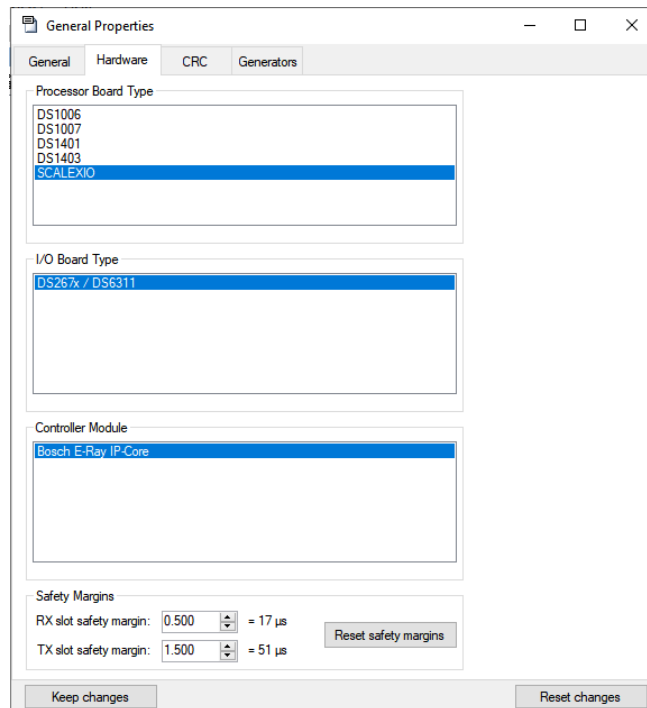
Software:

- FlexRay Configuration Tool must be installed.
- FlexRay Configuration Blockset for implementing a FlexRay communication must be available.
- A FIBEX file or an AUTOSAR system description file must be available.

Workflow

This workflow shows all the steps necessary to connect the external devices to a simulated FlexRay communication.

1. Creating Simulink configuration data in the FlexRay Configuration Tool.
 - In the FlexRay Configuration Tool, create a project and import the FIBEX file or AUTOSAR system description file that contains the definition of the FlexRay bus system. Refer to [Handling Configuration Projects \(FlexRay Configuration Tool Guide\)](#).
 - Create configurations, for example, select the PDUs and signals for simulation. Then select the SCALEXIO hardware in the hardware configuration.



For details, refer to [Creating Configurations \(FlexRay Configuration Tool Guide\)](#).

- Configure and create communication tasks, application tasks, and synchronization tasks. Refer to [Creating Tasks \(FlexRay Configuration Tool Guide\)](#).
- Generate code and Simulink configuration data for modeling and simulating with the FlexRay Configuration Blockset. Refer to [Generating Code \(FlexRay Configuration Tool Guide\)](#).

The Simulink configuration data is stored in an M file.

2. Generating the FlexRay blocks for the Simulink model.

In MATLAB, you must execute a command to generate the FlexRay blocks based on the M file created by the FlexRay Configuration Tool. The command creates the automatically generated FlexRay model, which contains the configured FlexRay blocks which you can use in your Simulink model. Refer to [How to Generate Blocks for Modeling a FlexRay Communication \(Model Interface Package for Simulink - Modeling Guide\)](#).

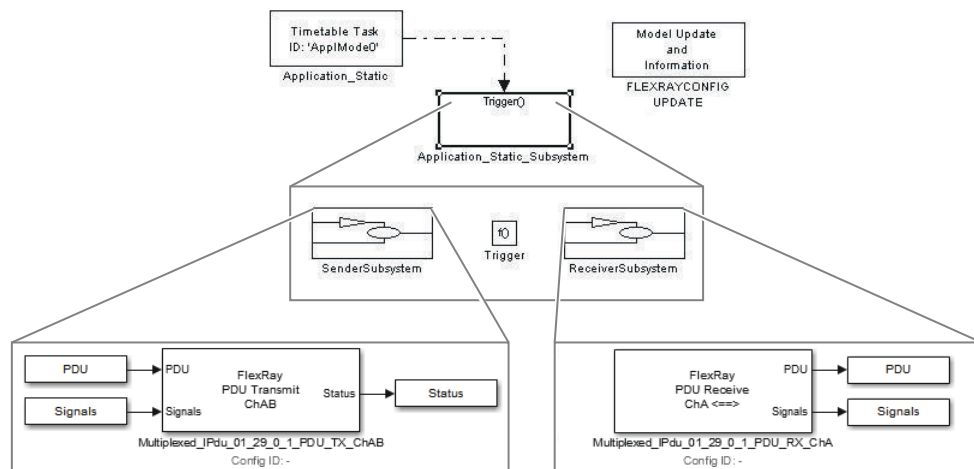
3. Modeling the FlexRay communication in Simulink.

The FlexRay communication must be modeled in Simulink using the automatically generated FlexRay model. Refer to [Modeling a FlexRay Bus Interface \(Model Interface Package for Simulink - Modeling Guide\)](#).

The FLEXRAYCONFIG Update block is mandatory in the model. You can also model PDUs for sending and receiving, manipulate them, and control the FlexRay network. Refer to:

- [Modeling FlexRay Communication \(Model Interface Package for Simulink - Modeling Guide\)](#)
- [How to Manipulate the Payload Length of a PDU \(Model Interface Package for Simulink - Modeling Guide\)](#)
- [How to Manipulate the Update Bit of a PDU \(Model Interface Package for Simulink - Modeling Guide\)](#)
- [Configuring a FlexRay Network \(Model Interface Package for Simulink - Modeling Guide\)](#)

The following illustration shows an example of a Simulink model containing some blocks of the FlexRay Configuration Blockset.

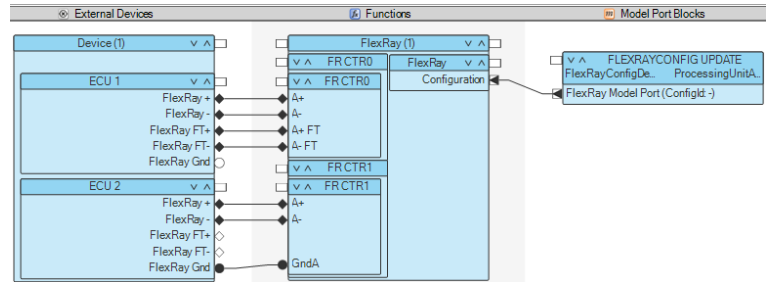


4. Implementing the FlexRay communication in ConfigurationDesk.

To execute the modeled FlexRay communication on the SCALEXIO system, you must implement the FlexRay communication in ConfigurationDesk. In general, this comprises the following steps:

1. Analyzing the Simulink model in ConfigurationDesk.
When you do this, Configuration Port blocks (i.e., specific model port blocks) are created that provide the FlexRay bus settings which you modeled in the Simulink model to ConfigurationDesk.
2. Building the signal chain in ConfigurationDesk.
You must map the Configuration Port blocks to FlexRay function blocks. If you have specified a device for your ECU, you can additionally map the FlexRay function blocks to the related device blocks.

The following illustration shows an example of a signal chain with one device block, one FlexRay function block, and one Configuration Port block.



3. Assigning suitable hardware resources to the FlexRay function blocks to access the required bus modules or boards of the SCALEXIO system.
4. Configuring the FlexRay function blocks according to your requirements and depending on the assigned hardware resources.

Depending on the configuration, function ports and/or event ports are added to the FlexRay function blocks. To use these ports in the Simulink model, you must map the ports to model port blocks and/or Runnable Function blocks, and synchronize the model interfaces in ConfigurationDesk and Simulink.

5. Configuring FlexRay tasks according to your requirements, e.g., prioritize the FlexRay tasks over other tasks of the real-time application.

Depending on the configuration, you must synchronize the model interfaces in ConfigurationDesk and Simulink.

For basic information and details on the individual steps, refer to [Building the Signal Chain for FlexRay Communication \(ConfigurationDesk Real-Time Implementation Guide\)](#).

5. Connecting the FlexRay bus lines of ECU and SCALEXIO system
Assigning the FlexRay function to a hardware resource specifies the ECU connector pins that the signals are routed to. You can get the pin numbers in ConfigurationDesk in the Properties Browser when the signal port is selected. Refer to [How to Access Hardware Properties \(ConfigurationDesk Real-Time Implementation Guide\)](#).

LIN Bus Connection

Introduction

To communicate to and from the external devices via a LIN bus, you can connect their LIN channels to the SCALEXIO system.

The information in this topic applies only if you model LIN communication using the RTI LIN MultiMessage Blockset. For information on implementing LIN communication using ConfigurationDesk's Bus Manager, refer to [ConfigurationDesk Bus Manager Implementation Guide](#).

Restbus simulation

External devices can be members of a LIN bus. The SCALEXIO system can replace all the other bus members which are normally connected to them. Thus, there is no difference in the LIN communication from the point of view of the external devices. This functionality is called "restbus simulation".

For specific purposes, it is useful to replace a simulated bus member by the real bus member. Either a bus member is simulated by the SCALEXIO system or the real bus member is connected to the bus. To minimize the number of real-time applications required for the simulation, it is possible to switch off a simulated bus member without compiling the real-time application again. Otherwise the SCALEXIO system would require hundreds of different real-time applications.

Simulating errors

You can simulate frame errors and test the reaction of the external devices. You can manipulate frame IDs and lengths, detect checksums, and observe frames behavior.

You can also simulate errors in the physical layer of the LIN bus. The bus lines can be broken or switched to the lines of other signals, battery voltage or ground. Refer to [Simulating Electrical Errors in the Wiring](#) on page 123.

Preconditions

Hardware: The SCALEXIO system must contain a bus module or board with one of the following channel types:

- *LIN 1* (available on a DS2672 Bus Module)
- *LIN 2* (available on a DS6301 CAN/LIN Board and a DS6351 LIN Board)
- *Bus 1* (available on a DS2671 Bus Board)

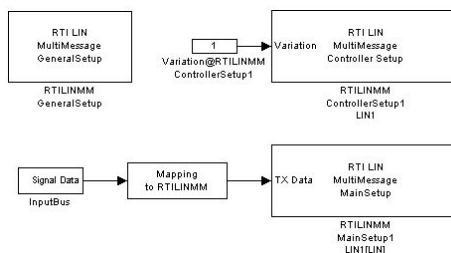
Software: The RTI LIN MultiMessage Blockset for implementing a LIN communication must be installed.

Workflow

This workflow shows all the steps necessary to connect the ECU to a simulated LIN communication.

1. Modeling the LIN communication in Simulink.

The LIN communication must be modeled in Simulink using the RTI LIN MultiMessage Blockset. For instructions on modeling the LIN communication, refer to [Modeling a LIN Bus Interface \(Model Interface Package for Simulink - Modeling Guide\)](#).



2. Implementing the LIN communication in ConfigurationDesk.

To execute the modeled LIN communication on the SCALEXIO system, you must implement the LIN communication in ConfigurationDesk. In general, this comprises the following steps:

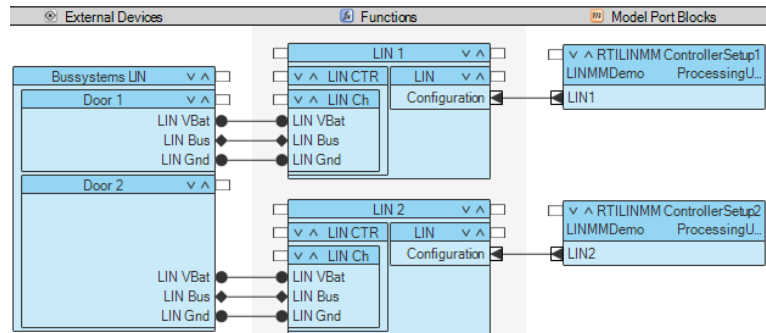
1. Analyzing the Simulink model in ConfigurationDesk.

When you do this, Configuration Port blocks (i.e., specific model port blocks) are created that provide the LIN bus settings which you modeled in the Simulink model to ConfigurationDesk.

2. Building the signal chain in ConfigurationDesk.

You must map the Configuration Port blocks to LIN function blocks. If you have specified a device for your ECU, you can additionally map the LIN function blocks to the related device blocks.

The following illustration shows an example of a signal chain with one device block, two LIN function blocks, and two Configuration Port blocks.



3. Assigning suitable hardware resources to the LIN function blocks to access the required bus modules or boards of the SCALEXIO system.
4. Configuring the LIN function blocks according to your requirements and depending on the assigned hardware resources.
5. Configuring LIN tasks according to your requirements, e.g., prioritized the LIN tasks over other tasks of the real-time application.

For basic information and details on the individual steps, refer to [Building the Signal Chain for LIN Bus Communication \(ConfigurationDesk Real-Time Implementation Guide\)](#).

3. Connecting the LIN bus lines of the external devices and the SCALEXIO system.

Assigning the function to a hardware resource specifies the ECU connector pins that the signals are routed to. You can get the pin numbers in ConfigurationDesk in the Properties Browser when the signal port is selected, refer to [How to Access Hardware Properties \(ConfigurationDesk Real-Time Implementation Guide\)](#).

Serial Interface Connection

Introduction

To connect external devices to the SCALEXIO system via a serial interface, you must implement the corresponding communication in the real-time model.

Custom function for implementing a UART interface

ConfigurationDesk does not provide a standard function for implementing a UART interface. You must create a custom function for this purpose.

Preconditions


The SCALEXIO system must contain a bus module or board with one of the following channel types:

- *LIN 1* (available on a DS2672 Bus Module)
- *Bus 1* (available on a DS2671 Bus Board)
- *UART 1* (available on a DS6321 UART Board)

Workflow

This workflow shows all the steps necessary to connect the ECU to a UART interface of the SCALEXIO system.

1. Creating a custom function


A custom function consists of a C++ source code module and a custom function block type. In the C++ source code module, you must create and configure a driver object which controls the UART channel. The instances of the custom function block type are required for building the signal chain in ConfigurationDesk, assigning a hardware resource to the UART driver object and integrating the UART driver object into the real-time application. For detailed information on how to create the custom function, refer to [ConfigurationDesk UART Implementation](#) .

2. Modeling the communication in Simulink (behavior model)

In your behavior model you can manage the signals that are sent or received via the UART interface or handle the events of the UART interface. All the signals which are used outside the behavior model (for example, in the I/O model) must be connected to model port blocks. These are the interface between the Simulink model and the signal chain in ConfigurationDesk.

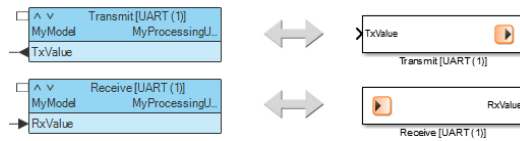
3. Creating the model port blocks

There are two ways to create the model port blocks:

1. In Simulink, use the model port blocks of the dSPACE Model Port Block library. Drag the necessary blocks to your Simulink model, configure them and connect them to the signals. Afterwards, you can analyze your Simulink model in ConfigurationDesk. This makes the model port blocks available in the model topology. For details, refer to [Analyzing Simulink Behavior Models \(ConfigurationDesk Real-Time Implementation Guide\)](#) .
2. In ConfigurationDesk, you can extend the signal chain starting from the function ports of custom function blocks. ConfigurationDesk can generate

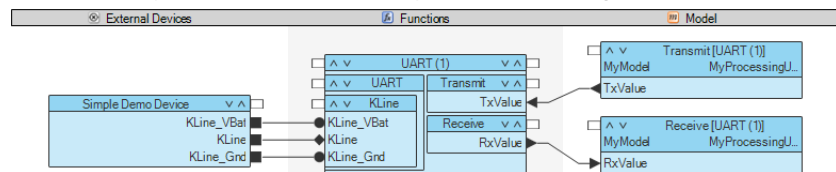
suitable model port blocks in the Model section and can connect them to the function block. Afterwards, ConfigurationDesk can generate the model interfaces of all the unresolved model blocks. A temporary Simulink model is created which contains all the necessary model ports blocks. These blocks can be used in your Simulink model for interfacing the signals. For details, refer to [How to Transfer Unresolved Model Port Blocks to a Simulink Behavior Model via an Interface Model \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(c8d96c8885d3000a912c2582004aed63_img.jpg\)\)](#).

The following illustration shows a model port block in Simulink (right) and the corresponding model port block in ConfigurationDesk (left).



4. Building the signal chain

You can build the signal chain in ConfigurationDesk. The custom function block of the UART interface can be connected to the external device and to the model port blocks, see the example in the following illustration.

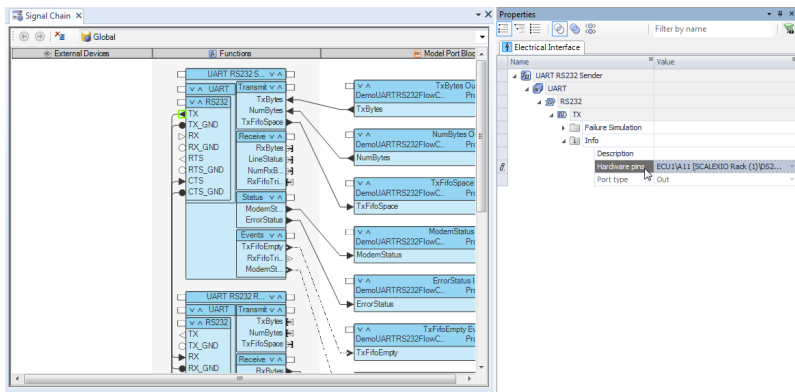


5. Assigning hardware resources

To execute the modeled serial communication, the custom function must be assigned to channels of a suitable channel type. The number of channels required depends on the implemented driver object (see [Creating and Adjusting the Source Code Files \(Serial Communication\) \(ConfigurationDesk Custom I/O Function Implementation Guide !\[\]\(17413706fd4997a1a4bdf85c6864eee1_img.jpg\)\)](#)). For details on assigning hardware resources, refer to [Assigning Hardware Resources to Function Blocks \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(f419710cbe076aa30a9c6c031b5cbe84_img.jpg\)\)](#).

6. Connecting the bus lines

Assigning the function to a hardware resource specifies the ECU connector pins that the signals are routed to. You can get the pin numbers in ConfigurationDesk in the Properties Browser when the signal port is selected. Refer to the following illustration and [How to Access Hardware Properties \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(cf531ed27e91483460120fcc057b3901_img.jpg\)\)](#).



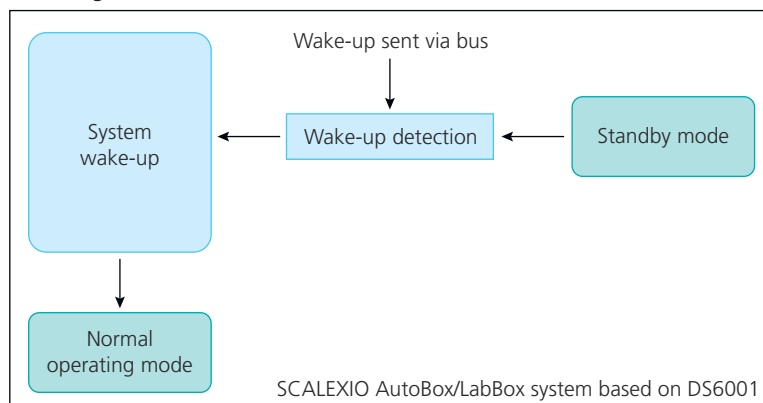
Waking up SCALEXIO AutoBoxes/LabBoxes via Bus Signals

Introduction

A SCALEXIO AutoBox/LabBox with a DS6001 Processor Board can change its operating mode from standby to normal operating mode when it receives a dedicated wake-up message or frame via a CAN or LIN bus.

Basics on waking up

SCALEXIO AutoBoxes/LabBoxes with DS6001 Processor Boards can be woken up by a CAN or LIN wake-up. This means a AutoBox/LabBox can change its operating mode from standby to normal operating mode when it receives a dedicated wake-up message or frame via CAN or LIN bus, as shown in the following illustration.



Hardware requirements

To use the wake-up feature, you must have a SCALEXIO system that consists of the following hardware components:

- DS6001 Processor Board
- One of the following SCALEXIO I/O boards:
 - DS6301 CAN/LIN Board

- DS6341 CAN Board
- DS6342 CAN Board
- DS6351 LIN Board
- One or more SCALEXIO AutoBoxes or SCALEXIO LabBoxes (19-Slot or 8-Slot)

The I/O board must be installed in a AutoBox/LabBox. If the SCALEXIO system consists of more than one AutoBox/LabBox, the AutoBoxes/LabBoxes must be connected via their power control bus connectors. Refer to:

- [Powering Up and Shutting Down SCALEXIO AutoBox \(SCALEXIO Hardware Installation and Configuration !\[\]\(0551a83d441798e532995956b603f604_img.jpg\)\)](#)
- [Powering Up and Shutting Down the LabBox \(SCALEXIO Hardware Installation and Configuration !\[\]\(54ee180c0037b66a36ce2219a481afde_img.jpg\)\)](#)

Implementing the wake-up

To use the wake-up feature, you must implement it in ConfigurationDesk via CAN or LIN function blocks as follows:

- Assign the required CAN or LIN channel of the I/O board to the respective CAN or LIN function block as a hardware resource.
- Set the Power wake-up property of the function block to Enabled.
- Only for CAN function blocks: Map the Mode function port of the Transceiver function to a model port and/or enable test automation support for the function port.

To use wake-up, the AutoBox/LabBox must be in standby mode. It is recommended to use the System Shutdown function block for this purpose. Refer to [Basics on Using the System Shutdown Functionality \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(e474458956c9a37fbf9586ddb60a7fa1_img.jpg\)\)](#)

For more information, refer to [Configuring the Basic Functionality \(CAN\) \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(3e2231b1ad3ca8da8658228c00dd08e0_img.jpg\)\)](#) or [Configuring the Basic Functionality \(LIN\) \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(96a82dd1250f57fd139c5f3b80c9d977_img.jpg\)\)](#).

Using wake-up

To wake up a AutoBox/LabBox via a CAN or LIN wake-up, the AutoBox/LabBox must be in standby mode. If the AutoBox/LabBox is in any other mode and a wake-up is sent via the bus, the wake-up has no effect.

The following workflow is recommended for setting the AutoBox/LabBox to standby mode so you can wake it up via CAN or LIN:

1. Implement the wake-up feature in ConfigurationDesk as described above.
2. Load the real-time application to the flash memory of the DS6001 Processor Board.

This step is optional. However, if you do this and the AutoBox/LabBox is woken up later on, the real-time application starts automatically from the flash memory. Otherwise, no real-time application is executed after wake-up. Refer to [Real-Time Application - Load to Flash \(ConfigurationDesk User Interface Reference !\[\]\(b792654f2cef9719eabeb6c5be00811e_img.jpg\)\)](#)

3. Only relevant for wake-up via CAN: Set the transceiver of the related CAN channel to sleep mode.

To wake up the AutoBox/LabBox via a CAN wake up, you must set the transceiver of the related CAN channel to sleep mode *before* you set the AutoBox/LabBox to standby mode. You must explicitly set the transceiver to sleep mode via the Mode function port of the CAN function block's Transceiver function. Refer to [Overview of Ports and Basic Properties \(CAN\) \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(21199eb166cc97331a0c54c649195dcc_img.jpg\)\)](#).

4. Set the AutoBox/LabBox to standby mode.

Effects on the power control bus

If you wake up a SCALEXIO AutoBox/LabBox via CAN or LIN and the AutoBox/LabBox is connected to other AutoBoxes/LabBoxes via the power control bus, all AutoBoxes/LabBoxes that are connected to the power control bus wake up. This also applies to the following AutoBoxes/LabBoxes:

- AutoBoxes/LabBoxes without a DS6001 Processor Board
- AutoBoxes/LabBoxes without a DS6301 CAN/LIN Board, DS6341 CAN Board, or DS6351 LIN Board

Related topics

Basics

[Basics on Using the System Shutdown Functionality \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(05be7c7a8995decd503647c99211f7c2_img.jpg\)\)](#)
[Configuring the Basic Functionality \(CAN\) \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(16cd6e1a39784ecf52b4db09f4865f40_img.jpg\)\)](#)
[Configuring the Basic Functionality \(LIN\) \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(64f85e895c86bd992221df2da6f33c1f_img.jpg\)\)](#)
[Overview of Ports and Basic Properties \(CAN\) \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(a60e44a6ea673209b24ab1016c50184b_img.jpg\)\)](#)
[Powering Up and Shutting Down the LabBox \(SCALEXIO Hardware Installation and Configuration !\[\]\(112e9b265c82c58da0f0ab95fe70ecde_img.jpg\)\)](#)

References

[Real-Time Application - Load to Flash \(ConfigurationDesk User Interface Reference !\[\]\(dd161862f9164df98f62b726e9846241_img.jpg\)\)](#)

Connecting External Devices to the Power Supply

Introduction The simulator can provide the battery voltage via power switch channels for the external devices.

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

Basics on Simulating the Vehicle Battery.....	85
Gives basic information on simulating the vehicle battery.	
Controlling the Battery Voltage Level.....	87
You can control the battery voltage to simulate overvoltage, undervoltage or voltage fluctuation during engine start.	
Switching the High Rails.....	88
You can switch the battery voltage to simulate different switched battery outputs, for example, KL 15 (battery+ through ignition switch) or KL 30 (battery+ direct). In addition, you can measure the currents flowing through the high rails.	

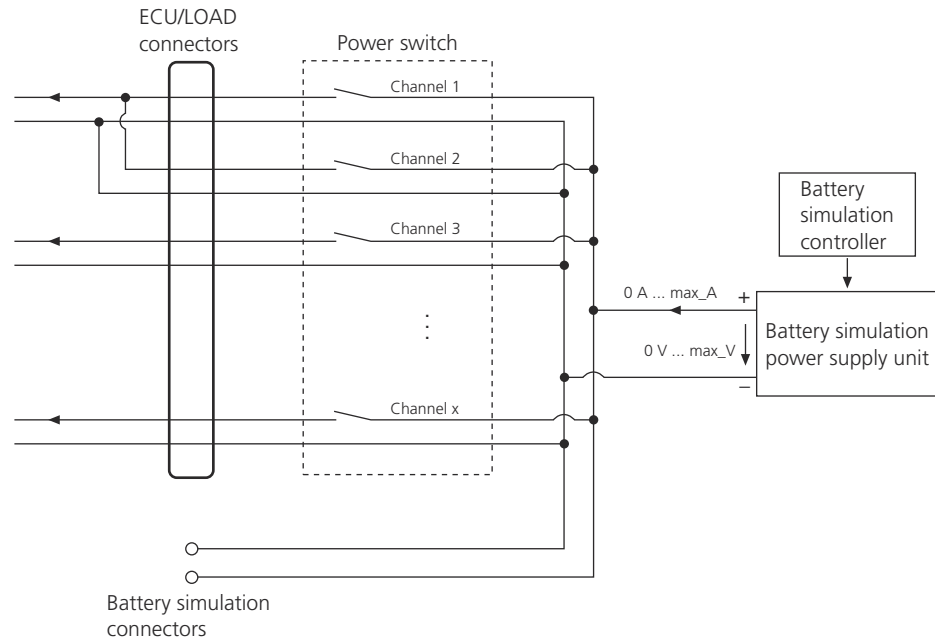
Basics on Simulating the Vehicle Battery

Introduction The SCALEXIO system can be be equipped with a power supply that can be used to provide the battery voltage.

Overview The SCALEXIO system can have a power supply unit to simulate the vehicle battery. The unit consists of three components:

- A battery simulation power supply unit
- A battery simulation controller
- Power switches

The following illustration shows how the components cooperate.



The battery simulation power supply unit generates the simulated battery voltage. It is controlled via the battery simulation controller which can be controlled by software to set the level of the battery voltage and the current limit.

Power switches are used to switch the battery voltage. They are controlled by software to simulate different switched battery outputs, for example, KL 15 (battery+ through ignition switch) or KL 30 (battery+ direct). Power switches are part of power switch channels. The channels can be used in parallel if currents higher than the maximum of one channel are required. In this case you must connect the additional channels at the ECU connector (see the example above).

Hardware

To simulate the battery power of a vehicle battery, for example, in a hardware-in-the-loop simulation, a SCALEXIO system requires the following components:

- A *battery simulation power supply unit* which generates the required voltage and current to simulate a vehicle battery.
- A *battery simulation controller* which sets the voltage and current limit used to simulate a vehicle battery.
- A *power switch* which switches the outputs of the battery simulation power supply unit to pins on the SCALEXIO system's front connector(s) or internal power supply rails.

Software

The battery simulation controller and the power switches are controlled via software. ConfigurationDesk provides function blocks which you can use for this purpose:

- To set the value of the battery voltage and the current limit, use the Power Supply Control function. Refer to [Power Supply Control \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(a88007b249b36c75dcbde101f514cec3_img.jpg\)](#).
- To switch the high rails, use the Power Switch function. Refer to [Power Switch \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(800628c068083563f747129d8b339031_img.jpg\)](#).

Controlling the Battery Voltage Level

Introduction

You can control the battery voltage to simulate overvoltage, undervoltage or voltage fluctuation during engine start.

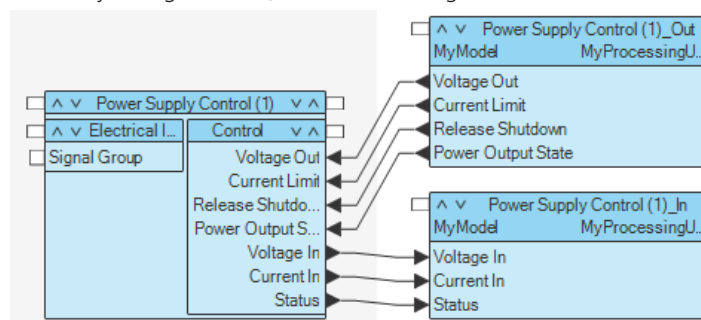
Basics

You can set the value of the battery voltage to simulate different operating voltages. The following table shows some examples of battery voltages of a typical passenger car.

For Testing ...	Set the Battery Voltage To ...
Undervoltage	under 9 V
Engine cranking	8 ... 11 V
Engine running	13.8 ... 14 V
Overvoltage	over 17 V

Extending the signal chain

To control the battery voltage, you must use one Power Supply Control function block in your signal chain, see the following illustration.



Model interface The Power Supply Control Switch function block has four function inputs for controlling the power supply and three function outputs to provide measured values to the behavior model. You can use the function port to specify the value of the battery voltage and to specify a maximum current. The

values of the voltage and the total current supplied by the power supply are measured and can be read in the behavior model.

Electrical interface A Power Supply Control function has no interface to the ECU, so the function block has no signal ports.

Assigned hardware For executing the Power Supply Control function, the function block must be assigned to the battery simulation controller which is installed in the simulator to control the battery simulation power supply unit.

Related topics

Basics

[Power Supply Control \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(fa6f3af6bfa46c5d4a2d362681095beb_img.jpg\)\)](#)

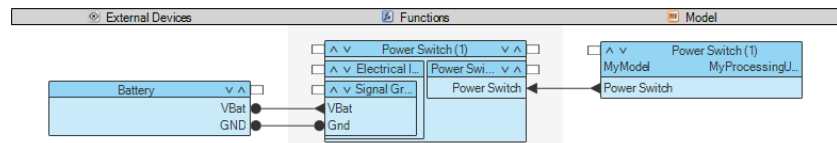
Switching the High Rails

Introduction

You can switch the battery voltage to simulate different switched battery outputs, for example, KL 15 (battery+ through ignition switch) or KL 30 (battery+ direct). In addition, you can measure the currents flowing through the high rails.

Extending the signal chain

To switch the battery voltage, you must use the Power Switch function block in your signal chain, see the following illustration.



Model interface The Power Switch function block has one or two function ports. You can use the Power Switch function port in your behavior model to switch the battery voltage. If you connect the port to a Constant block, for example, you can switch the assigned channels via the experiment and test software.

A DS2642 FIU & Power Switch Board can measure the current flowing through the assigned channels. If the function block is assigned to such a board, the model interface has a Current signal port that provides the measured value.

Electrical interface The electrical interface is the signal ports where you connect your ECU. The VBat port provides the switched battery voltage. The Gnd port provides the ground. The ECU connector pins providing the signal and reference depend on the assigned hardware or channels.

Assigned hardware For executing the Power Switch function, the function block must be assigned to a channel with the *Power Switch 2* channel type (available on a DS2680 I/O Unit) or *Power Switch 1* channel type (DS2642 FIU & Power Switch Board). If the current of one channel exceeds the maximum

allowed value, several channels can be used in parallel. If you use several channels in parallel, the maximum usable current of the channels are reduced to avoid an overcurrent on one channel caused by different internal resistances.

Related topics

Basics

[Power Switch \(ConfigurationDesk I/O Function Implementation Guide !\[\]\(c3d993ca47bfe2a953c700506ce31fa0_img.jpg\)\)](#)

Data Transmission Between dSPACE Systems

Introduction

When a SCALEXIO system is connected to a PHS-bus-based system (modular hardware based on a DS1006 or DS1007) or another SCALEXIO system, you can transmit data between the systems.

Where to go from here

Information in this section

[Basics of Data Transmission Between dSPACE Systems](#)..... 90

You can transmit data between a SCALEXIO system and a PHS-bus-based system or another SCALEXIO system.

[Configuring a Gigalink Connection for PHS-Bus-Based Systems](#)..... 92

To send or receive data of PHS-bus-based systems, the real-time model must be extended.

[Configuring a Gigalink Connection for SCALEXIO Systems](#)..... 93

To send or receive data of SCALEXIO systems, the behavior and I/O model must be extended.

Information in other sections

[Connecting a PHS-Bus-Based System \(SCALEXIO Hardware Installation and Configuration](#)

You can connect a SCALEXIO system to a PHS-bus-based system (i.e., a modular system with a DS1006 or DS1007) via Gigalink to exchange data.

Basics of Data Transmission Between dSPACE Systems

Introduction

You can transmit data between a SCALEXIO system and a PHS-bus-based system or another SCALEXIO system.

Data transmission

Data transmission allows to transmit data between dSPACE systems even if they base on different technologies (SCALEXIO system or PHS-bus-based system). You can only transmit model signals. Signal transmission is not synchronized. As a result, there is no way to check if new data is available. Sending signal data on the same channel again overrides previously sent data, and repeated reading signal data on the same channel without sending new data reads out already received data.

Modeling

You must create a real-time model for each dSPACE system. The real-time models are independent from each other. This means:

- If the real-time models use the same parameters, the real-time application will have independent copies of them.
- Interrupts cannot be forwarded from one real-time model to another. However, you can realize interrupts using I/O boards.
- You can not access I/O boards of the connected dSPACE system.
- The real-time models can be opened in Simulink at the same time. This is not allowed if the real-time models are modeled with RTI CAN MultiMessage or RTI LIN MultiMessage blockset. In this case only one model containing these blocks must be opened at the same time.

To be able to transmit data, the real-time models must be extended. The implementation depends on the system type.

In a SCALEXIO system, Gigalink function blocks must be added to the signal chain, see [Configuring a Gigalink Connection for SCALEXIO Systems](#) on page 93.

In a PHS-bus-based system, the Gigalink blocks must be added to the real-time model, see [Configuring a Gigalink Connection for PHS-Bus-Based Systems](#) on page 92.

When the real-time models are extended for data transmission, you can start code generation for each dSPACE system independent from each other.

Simulation

The dSPACE systems run independently. This means you must handle the real-time applications for each dSPACE system separately.

Although the simulations run independently, the plotter instruments in ControlDesk can plot variables in the same window. So it is possible to compare the plots of variables.

However, there can be a time offset when signals are triggered at the SCALEXIO system and PHS-bus-based system at the same time, refer to [Offset in Time of Signals of SCALEXIO and PHS-Bus-Based Systems](#) on page 144.

Related topics

Basics

[Connecting a PHS-Bus-Based System \(SCALEXIO Hardware Installation and Configuration !\[\]\(7d1d6890825e83a6a4a51febe2dcc7f3_img.jpg\)](#))

Configuring a Gigalink Connection for PHS-Bus-Based Systems

Introduction

To send or receive data of PHS-bus-based systems, the real-time model must be extended.

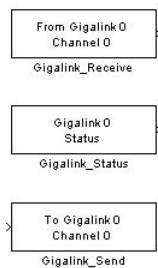
Software requirements

Data transmission can be realized via Gigalink blocks. They are contained in the RTI Gigalink Blockset, which is part of the RTI-MP library. The RTI Gigalink Blockset is available for the DS1006 and DS1007 platform and allows you to transmit signals between real-time applications via a Gigalink connection.

Extending the real-time model

The RTI Gigalink Blockset contains the following blocks:

- The **Gigalink_Send** block sends data to a specified Gigalink channel.
- The **Gigalink_Receive** block receives data from a specified Gigalink channel.
- The **Gigalink_Status** block checks the synchronization status of the specified Gigalink number.



In the block dialog of the RTI blocks, specify the Gigalink number, channel number, and signal width. The Gigalink number must be the port being used for data transfer. In PHS-bus-based system, the ports are number from 0 to 3. The channel number and signal width of the sender and receiver must be the same.

In Simulink connect the signals being sent or received to the Gigalink blocks.

It is not automatically checked whether the Gigalink connection is operable during run-time, but you can get the information using **Gigalink_Status** block.

For details on the Gigalink blocks, refer to [RTI Gigalink Blockset Reference \(RTI and RTI-MP Implementation Reference\)](#).

For details on using the Gigalink blocks, refer to [Implementing Interprocessor Communication Using Gigalink Blocks \(RTI and RTI-MP Implementation Guide\)](#).

Related topics

Basics

[Basics of Data Transmission Between dSPACE Systems](#)..... 90

Offset in Time of Signals of SCALEXIO and PHS-Bus-Based Systems.....	144
--	-----

References

RTI Gigalink Blockset Reference (RTI and RTI-MP Implementation Reference )

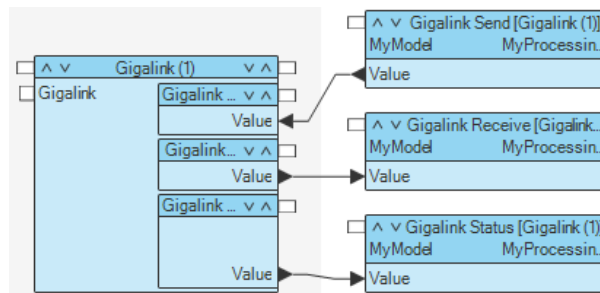
Configuring a Gigalink Connection for SCALEXIO Systems

Introduction

To send or receive data of SCALEXIO systems, the behavior and I/O model must be extended.

Extending the signal chain

To send or receive data, you must use a Gigalink function block in your signal chain, see the following illustration.



At the model interface of the Gigalink function block, specify the data width of the signals being sent and/or received. The data width of the sender and receiver must be the same.

At the electrical interface, specify the port and channel used for data transfer. In a SCALEXIO system the ports are numbered from 1 to 4 or 8 (depending on the version of the used DS2502 Link Board). In the standard configuration 4 ports are available and port 4 is connected to the back of SCALEXIO system. This port can usually be used for data transfer. The channel number of the sender and receiver must be the same.

For details on configuring the Gigalink function block, refer to [Basics on Gigalink Communication \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(bd3b31712ad9bab5a241210fa6925cdd_img.jpg\)\)](#).

Extending the behavior model

The signals to be sent or received must be connected to the Model Port blocks which corresponds to the Gigalink function block in MATLAB/Simulink, see the following illustration.



In Simulink connect the signals being sent to Model Port blocks connected to Value port of the Gigalink Send function.

Model Port blocks connected to Value port of the Gigalink Receive function provide the signals which are received via the Gigalink connection.

It is not automatically checked whether the Gigalink connection is operable during run-time, but you can get the information using Model Port blocks connected to the Value port of the Gigalink Status function.

Related topics

Basics

[Basics of Data Transmission Between dSPACE Systems..... 90](#)

HowTos

[How to Connect Another SCALEXIO System via Gigalink \(SCALEXIO Hardware Installation and Configuration !\[\]\(73002692dd5e7a64e60946be3158e719_img.jpg\)](#))

Working with ASM and ModelDesk

Introduction	You can use dSPACE Automotive Simulation Models (ASM) for the behavior model running on a SCALEXIO system. These are simulation models for developing and testing automotive ECUs.
--------------	--

Where to go from here	Information in this section
	Basics on Automotive Simulation Models..... 95 You can use ASMs to model the behavior of the controlled system.
	Modeling with Automotive Simulation Models..... 97 You can use the ASM blocksets to model the behavior model for your SCALEXIO system.
	Setting the Parameter Values of ASMs Using ModelDesk..... 97 If your behavior model contains blocks of the ASM library and/or Simulink blocks that are built like ASM blocks, you can parameterize the model using ModelDesk.
	Modeling the Environment (Roads and Scenarios)..... 98 If you are testing an ECU for vehicle dynamics you may want to simulate different driving situations. The environment must be modeled for vehicle dynamics simulations.
	Accessing the SCALEXIO System for Downloading the Parameter Values with ModelDesk..... 101 To download the parameter values to the real-time application, you must access the SCALEXIO system where the real-time application is loaded.

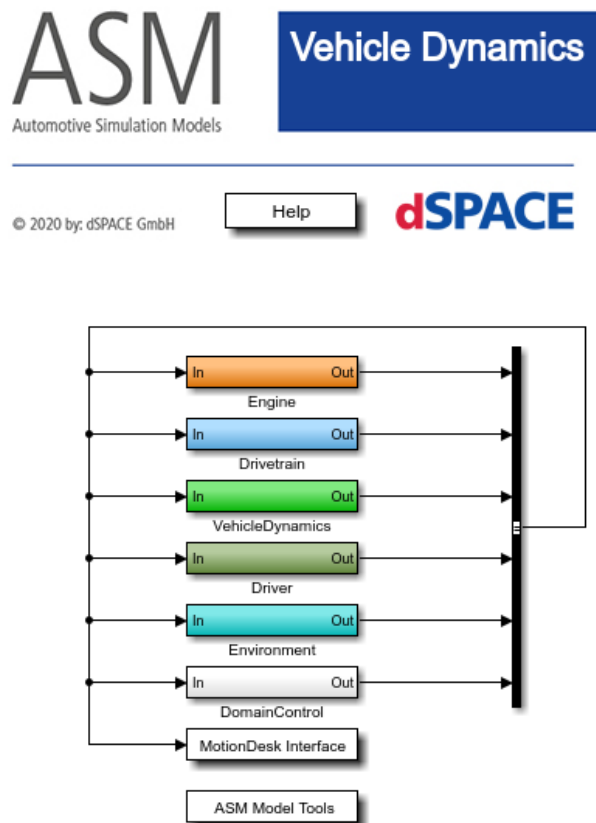
Basics on Automotive Simulation Models

Introduction	You can use ASMs to model the behavior of the controlled system.
--------------	--

Model Overview

ASMs are modeled in Simulink. The ASMs are shipped with various demo models. All demo models have a similar structure.









When you open an ASM demo model in MATLAB/Simulink, it looks something like this:




The demo model contains the following components:

- ASM modules that are autonomous and constitute different parts of an ASM, such as the Engine module and the Environment module. Refer to [ASM Module \(ASM User Guide\)](#).
- ASM Model Tools for access to different common functionalities in ASM demo models. Refer to [ASM Model Tools \(ASM User Guide\)](#).
- (optionally) A MotionDesk Interface for animation in MotioDesk. Refer to [MotionDesk Interface \(ASM User Guide\)](#).

Modeling with Automotive Simulation Models

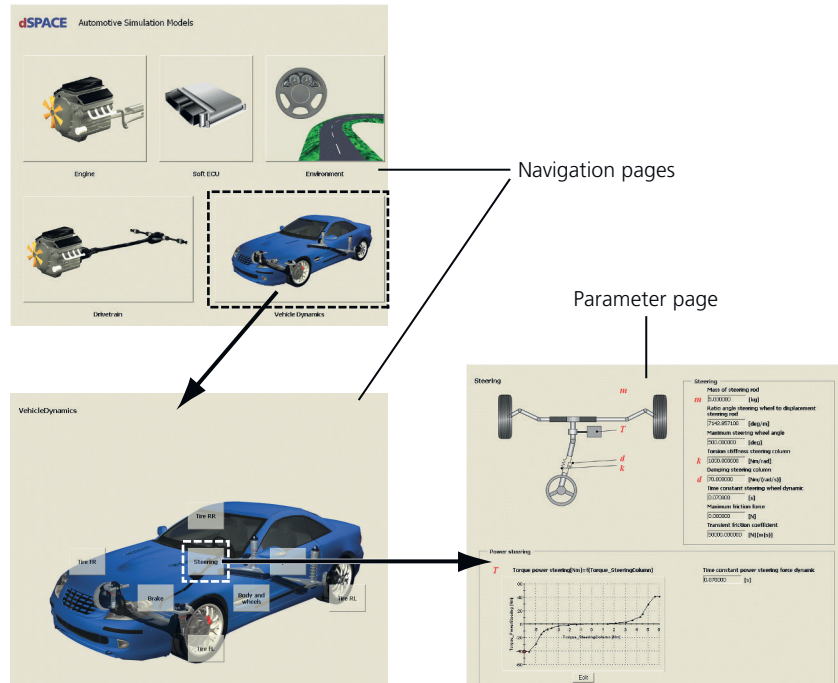
Introduction	You can use the ASM blocksets to model the behavior model for your SCALEXIO system.
Tutorials	<p>When ASM libraries are installed, the related documentation are also installed. The following documents contains tutorials that guides you to your first steps with ASM:</p> <ul style="list-style-type: none"> ▪ Tutorials (ASM Vehicle Dynamics Model Description ) ▪ Tutorials (ASM Trailer Model Description ) ▪ Tutorials (ASM Gasoline Engine Model Description ) ▪ Tutorials (ASM Diesel Engine Model Description ) ▪ Tutorials (ASM Gasoline Engine InCylinder Model Description ) ▪ Tutorials (ASM Diesel Engine InCylinder Model Description ) ▪ Tutorials (ASM Drivetrain Basic Model Description ) ▪ Tutorials (ASM Electric Components Model Description )

Setting the Parameter Values of ASMs Using ModelDesk

Introduction	If your behavior model contains blocks of the ASM library and/or Simulink blocks that are built like ASM blocks, you can parameterize the model using ModelDesk. ModelDesk provides a user interface for parameterizing these blocks.
Managing parameters	<p>ModelDesk manages the parameters for the ASM in projects, experiments, and parameter sets.</p> <p>Project In a project, you can manage different parameterization tasks that belong together. For example, you can group several ModelDesk experiments in one project.</p> <p>Experiments Defining an experiment is the basis for carrying out parameterization and allows you to manage all the files and parameter sets for it. You can add one ASM to the experiment. Defining different experiments allows you to manage different ASMs in one project.</p> <p>Parameter sets Parameter sets contain a set of all the parameters of one ASM vehicle. You can specify several variants of one ASM by using parameter sets. For example, you can specify parameter sets for an ASM for manual or automatic transmission.</p> <p>For details on managing the parameters in ModelDesk, refer to Creating Projects and Experiments (ModelDesk Project and Experiment Management ).</p>

Parameterizing ASM blocks

ModelDesk displays the components of a vehicle in parameter pages for you to specify parameter values in a graphical environment. To navigate to the parameter pages, you can use the car component illustrations on the navigation pages.



By specifying the parameters on the parameter pages, you specify the parameters of the ASM blocks.

For details on setting the parameters values managing the parameters in ModelDesk, refer to [Setting Parameters \(ModelDesk Parameterizing\)](#).

Parameterizing custom blocks

If you use blocks from your own libraries in the Simulink model, these custom blocks can also be parameterized via ModelDesk. To parameterize custom blocks, they must be collected in custom libraries. These libraries must be registered in ModelDesk.

For details on preparing and working with custom blocks, refer to [Integrating and Using Custom Libraries \(ModelDesk Parameterizing\)](#).

Modeling the Environment (Roads and Scenarios)

Introduction

If you are testing an ECU for vehicle dynamics, for example, ESP, you may want to simulate different driving situations. You can use ModelDesk to specify roads and scenarios for this.

Generating road models

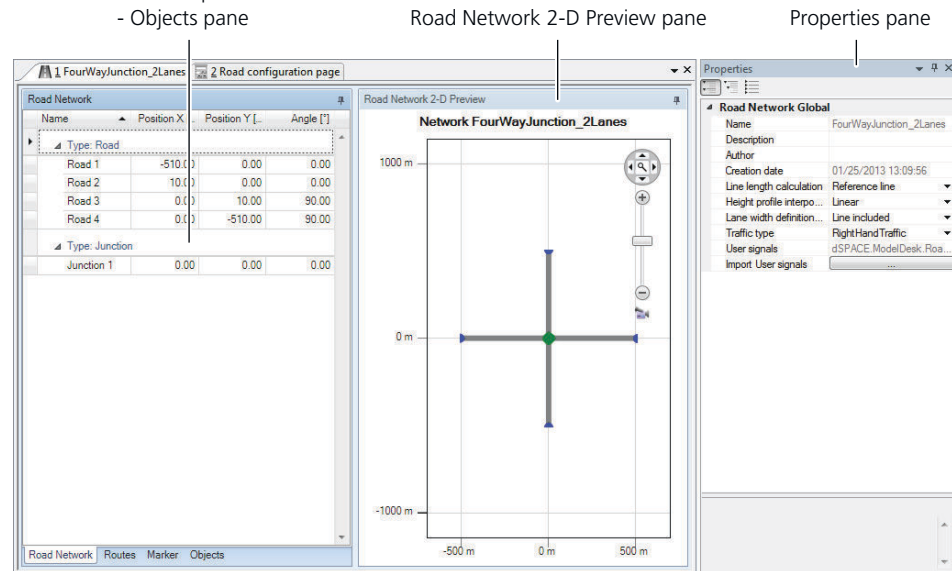
ModelDesk has the Road Generator to generate road models for simulation in the simulator and to visualize the road in MotionDesk.

With the Road Generator, you can:

- Create roads and specify their characteristics:
 - Horizontal profile
 - Height profile
 - Lateral slope profile
 - Lane sections and lanes (e.g., for modeling different numbers of lanes or road markings)
 - Special surface conditions (e.g., for modeling low μ areas)
 - Additional height profiles (e.g., for modeling a rough road)
 - Surface textures (e.g., arrows or crosswalks)
 - Scenery (e.g., for specifying the environment of a road)
 - Road length is limited only by the available memory
- Create junctions and connect them with roads to build a road network.
- Place static objects on roads or junctions (e.g., traffic signs)
- View roads, junctions, and road networks in a preview.
- Place and define shapes on roads or junctions (e.g., for roadworks):
 - Define additional road markings (e.g., parking lots)
 - Define continuous objects (e.g., concrete barriers)
 - Place repeating traffic objects (e.g., pylons)
- Define routes on road networks for the ASM and fellow vehicles.
- Define the traffic type for a road network.
- Define position markers as trigger points for the movement of the ASM and fellow vehicles on the road network.
- Inspect elements or areas of the generated road network scene in MotionDesk.
- Create the files:
 - For the simulation model of a road network to download them to a real-time application or Simulink
 - For visualization of a road network in MotionDesk

The following illustration shows the graphical user interface of the Road Generator in the road network view.

- Road Network pane
- Routes pane
- Marker pane
- Objects pane



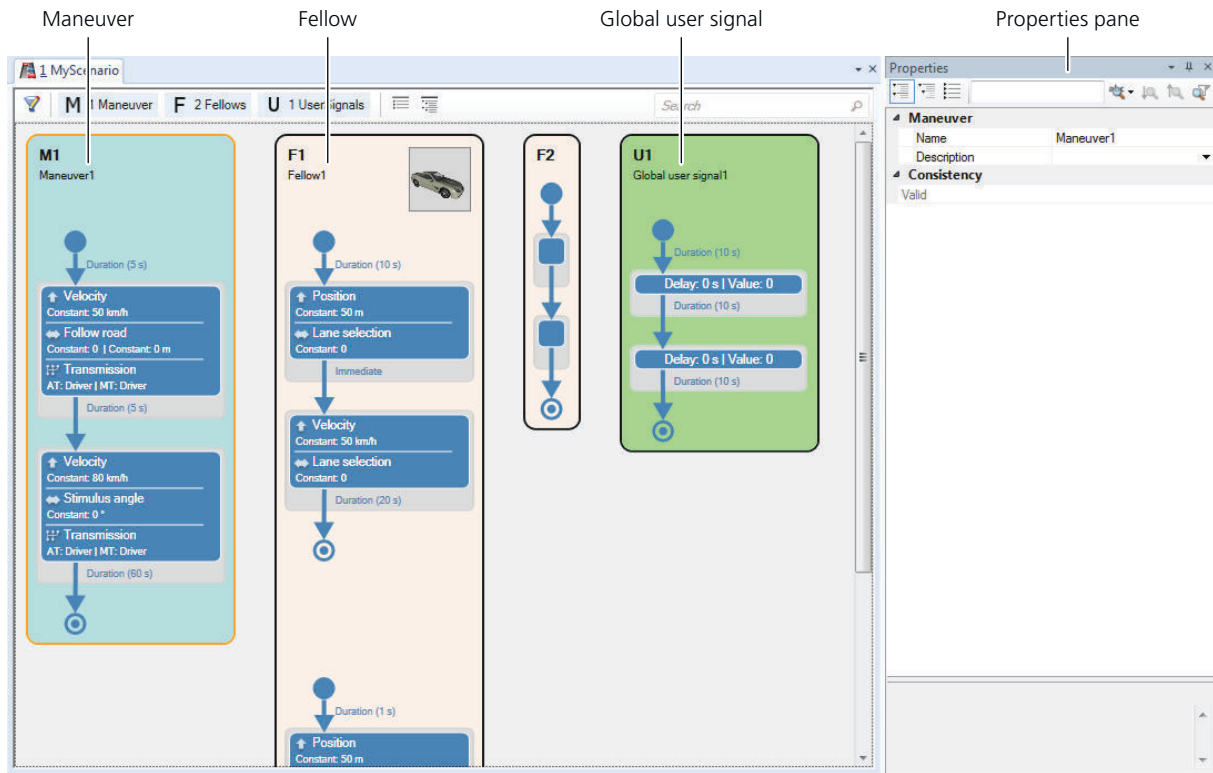
For details on generating and working with roads, refer to [Introduction to the Road Generator \(ModelDesk Road Creation\)](#).

Specifying scenarios

When you create a scenario with the Scenario Editor, you can:

- Specify the movement of the ASM vehicle.
- Specify the movements of the traffic participants, such as fellow vehicles or pedestrians
- Specify the road the scenario is linked to
- Specify the transition from one activity to the next
- Specify user signals for the maneuver, fellows, or globally to communicate with the simulation model
- Download the scenario parameters to Simulink or generate a parameter file to be loaded to the simulation platform via an experiment software

The following illustration shows the graphical user interface of the Scenario Editor.



For details on specifying scenarios, refer to [Introduction to the Scenario Editor \(ModelDesk Scenario Creation\)](#).

Related topics

Basics

[Introduction to the Road Generator \(ModelDesk Road Creation\)](#)
[Introduction to the Scenario Editor \(ModelDesk Scenario Creation\)](#)

Accessing the SCALEXIO System for Downloading the Parameter Values with ModelDesk

Introduction

To download the parameter values to the real-time application, you must access the SCALEXIO system where the real-time application is loaded.

Accessing the SCALEXIO system

To download the parameter values to the SCALEXIO system, you must register it. The host PC where ModelDesk is running and the SCALEXIO system are connected via Ethernet, so you must specify the IP address or the alias name of the SCALEXIO system in ModelDesk. It is sufficient to specify one of these parameters. If you specify both parameters and they do not match the entry in the `hosts` file, a warning message appears.

For details on registering, refer to [How to Register a Platform \(ModelDesk Platform Management !\[\]\(eafc244b53721dd1ec133f0772f70fc7_img.jpg\)](#)).

Downloading parameter values

When you have access to the SCALEXIO system, you can download the parameter values of one parameter page or the complete parameter set for the real-time application. If the real-time application for which parameters have been set is not already loaded, it is downloaded first. If another real-time application is loaded on the SCALEXIO system, a warning appears and you can cancel the download or download the correct real-time application and the parameter values.

For details on downloading, refer to [Working with Parameter Sets \(ModelDesk Parameterizing !\[\]\(5a132f13505a6571904d622757b7a8f0_img.jpg\)](#)).

Related topics**Basics**

[Using an Alias Name for the SCALEXIO System \(SCALEXIO Hardware Installation and Configuration !\[\]\(73002692dd5e7a64e60946be3158e719_img.jpg\)](#))

HowTos

[How to Choose a Model and Initialize the Consistency Check \(ModelDesk Parameterizing !\[\]\(35dc653d59570f8f891c312eeece91a2_img.jpg\)](#))

Experimenting with a SCALEXIO System

Introduction When a real-time application is created for the SCALEXIO system, you can start experimenting.

Where to go from here

Information in this section

[Basics on Experimenting Using a SCALEXIO System](#)..... 104

When the real-time application is built, you can download it to the SCALEXIO system.

[Basics on Experimenting With ControlDesk](#)..... 107

You can organize and perform your experimentation with ControlDesk. Before you start your work you should familiarize yourself with some basic terms.

[Working with ControlDesk](#)..... 108

When you have access to the SCALEXIO system, you can start experimenting.

Information in other sections

[ControlDesk](#)..... 33

Provides information on the purpose and features of ControlDesk.

[Configuring a SCALEXIO System](#)..... 43

Before you can start experimenting, the SCALEXIO system must be configured. In the configuration process you can specify the signals required by your ECU or controlled system, implement the real-time model and download it to the SCALEXIO system.


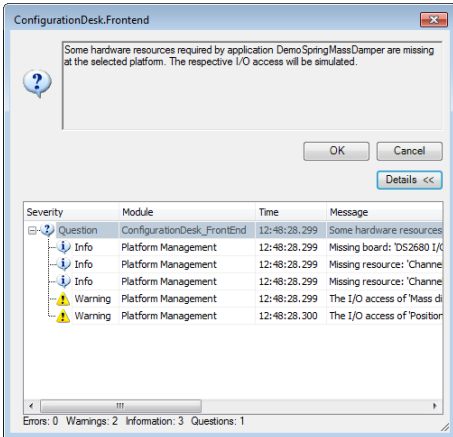
Basics on Experimenting Using a SCALEXIO System

Introduction

When the real-time application is built, you can download it to the SCALEXIO system using dSPACE tools, such as, ConfigurationDesk or ControlDesk.

Checking compatibility

During the download process, the dSPACE tools check if the hardware topology and the configuration of the real-time application are compatible with the target hardware. This compatibility check can have the following results:

Result	Description
The application is compatible with the connected target hardware.	Application properties such as the processor architecture and the memory capacity match the corresponding properties of the target hardware. All the channels used by the application are identical to the channels on the target hardware regarding channel number, Ethernet adapter name, DS number of the I/O boards, slot number, unit name, rack name, and connected dSPACE pins. The application can be executed on the target hardware.
The application is conditionally compatible with the connected target hardware.	<p>This case occurs if one of the following conditions applies:</p> <ul style="list-style-type: none"> At least one channel used by the application is missing on the target hardware. A channel property is configured differently, for example, the Load description property or connector pins. The variant of the DS2655 FPGA Base Board does not match to the FPGA custom function block. Refer to How to Check Hardware Resources Required for FPGA Custom Function Blocks (SCALEXIO) (ConfigurationDesk I/O Function Implementation Guide ). <p>In these cases a message dialog appears which informs you about the inconsistencies and asks you how to continue. A message dialog can, for example, look like this:</p> 

Note

- The compatibility check is always performed entirely, i.e., it does not stop after the first inconsistency is detected, but continues till the end to give you a complete overview of all the inconsistencies in the Message Viewer.
- In multimodel applications, the dSPACE tools automatically assign models to cores for execution during the download process. You cannot change this assignment. If your multimodel application contains more models than the maximum allowed number, the download is aborted with an error message. For a SCALEXIO Real-Time PC, the maximum allowed number of models is (number of processor cores - 1). For the DS6001 Processor Board, it is the number of processor cores.

For details of the compatibility check, refer to [Basics on Downloading Real-Time Applications \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(666e09182d4cd268646ea700ea60dcdf_img.jpg\)](#)).

Notes on firmware compatibility for building real-time applications

A dSPACE installation provides the files that are necessary to build the real-time application and the firmware files for the hardware. These files must be compatible with each other. If the real-time application is built with the same dSPACE installation that the firmware version of the hardware comes from, the real-time application is fully compatible with the firmware. You can download the real-time application without restriction.

If the real-time application was built with an earlier dSPACE installation and the firmware version of the hardware comes from a later dSPACE installation, it is possible to download the real-time application to the hardware. Later firmware versions are downward-compatible, so they are always compatible with earlier real-time applications.

If the real-time application was built with a later dSPACE installation and the firmware version of the hardware comes from an earlier dSPACE installation, it is not possible to download the real-time application. dSPACE tools prevent the download as compatibility is not guaranteed. To download the real-time application, you must first update the firmware with ConfigurationDesk (refer to [How to Update or Repair SCALEXIO or MicroAutoBox III Firmware via ConfigurationDesk \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(003082e50e3009141f59bd5df831749f_img.jpg\)](#)) or ControlDesk (refer to [Update Firmware \(ControlDesk Platform Management !\[\]\(f439ede8735757e3190eab35e168f1de_img.jpg\)](#))).

Tip

For information on the firmware version of the hardware, you can read its properties in ConfigurationDesk or ControlDesk.

- ConfigurationDesk: Refer to [How to Access Hardware Properties \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(039cd6b2e7148ba5690aa619b922c426_img.jpg\)](#)).
- ControlDesk: Refer to [How to Assign dSPACE Real-Time Hardware or VEOS to a Platform \(ControlDesk Platform Management !\[\]\(8b9db310e3bd56ffa44f3d5130ea99e2_img.jpg\)](#)).

SCALEXIO_RTLIB license

If a real-time application was built with Release 2017-B or later, loading it to a SCALEXIO system does not require a SCALEXIO_RTLIB license on the host PC as it was required in previous releases.

If a real-time application was built with Release 2017-A or earlier, loading it to a SCALEXIO system is not possible using an experiment software of Release 2017-B or later.

Channel behavior during download

When a real-time application is downloaded, the channels are initialized first. They are set to a secure state during initialization, which means that the channels are cut off from the ECU electrically, and no signals are available at the I/O connectors of the SCALEXIO system.

Setting the SCALEXIO system time

The system time (i.e. the real-time clock) of the SCALEXIO processing hardware is set automatically to the system clock of the connected host PC when the first real-time application is downloaded to the processing hardware after it was powered up.

Starting the real-time application

The real-time application can be started automatically or manually in ConfigurationDesk. Refer to [How to Download a Real-Time Application \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(6059a5aa8b4ca7bb793408023d6c6e42_img.jpg\)](#)).

For handling the real-time application in ControlDesk, refer to [Basics on Experimenting With ControlDesk](#) on page 107.

Related topics

HowTos

- [How to Access Hardware Properties \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(adb0331d22f78481623cc605df40612a_img.jpg\)](#))
- [How to Assign dSPACE Real-Time Hardware or VEOS to a Platform \(ControlDesk Platform Management !\[\]\(7e3a264c08e10137510d1aa76522412b_img.jpg\)](#))
- [How to Download a Real-Time Application \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(13ab9bea7a2b6465d20b6fafd4770e28_img.jpg\)](#))
- [How to Update or Repair SCALEXIO or MicroAutoBox III Firmware via ConfigurationDesk \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(fdbd4f3e18d391808e42202d652ce159_img.jpg\)](#))

References

- [Update Firmware \(ControlDesk Platform Management !\[\]\(5a0d662075632df1b39c9e3427a70093_img.jpg\)](#))

Basics on Experimenting With ControlDesk

Introduction

You can organize and perform your experimentation with ControlDesk. Before you start your work you should familiarize yourself with some basic terms.

Basic terms in ControlDesk

The following list contains brief definitions of terms used in ControlDesk.

Project A project manages different experiments that belong together, such as the ECU testing tasks for verifying a specific air condition control system. It holds the experiments relating to these tasks, and documents relevant to the project.

Experiment An experiment is the basis for carrying out a specific task, for example, adjusting the values of inside/outside temperature or solar intensity. An experiment allows you to manage all the items relating to the task, such as: platforms/devices, layouts, variable descriptions, data sets.

Platform A software component representing an environment where a simulation model is computed in real-time (dSPACE real-time hardware). For a SCALEXIO system, this is the SCALEXIO platform.

Variable description file The variable description specifies the parameters and measurement variables that are available on the real-time hardware. The variable description of a SCALEXIO system is stored as a system description (SDF) file.

Data set A data set contains a complete set of parameters. The SCALEXIO platform has a working data set for modifying parameter values.

Layout Layouts are used for visualizing variables: To change parameter values, or measure and record data, variables can be placed on a layout. In the layout, graphical instruments are used for adjusting and recording the variables.

Workflow


To access the SCALEXIO system and prepare an experiment in ControlDesk, you have to perform the following steps:

1. Define a project.
2. Add an experiment to the project.
3. Add the SCALEXIO platform to the experiment and configure its settings.
4. Select a variable description for access to the variables of the real-time application.
5. Create layouts with instruments to visualize the values of parameters and measurement variables.

Project wizard

In ControlDesk, a project wizard guides you through the first steps of accessing the SCALEXIO system. For a description of this fast and easy way of starting experimenting, refer to [Defining Projects and Experiments \(ControlDesk Project and Experiment Management !\[\]\(b64b40baaee5acddc1eab8538ba84754_img.jpg\)](#)).

Accessing the SCALEXIO

When you have defined your project and experiment, you can access the SCALEXIO system. In ControlDesk you need a SCALEXIO platform which represents a SCALEXIO system used as real-time hardware. Refer to [ControlDesk Platform Management](#) .

Working with ControlDesk

For an overview of typical work steps in ControlDesk, refer to [Working with ControlDesk](#) on page 108.

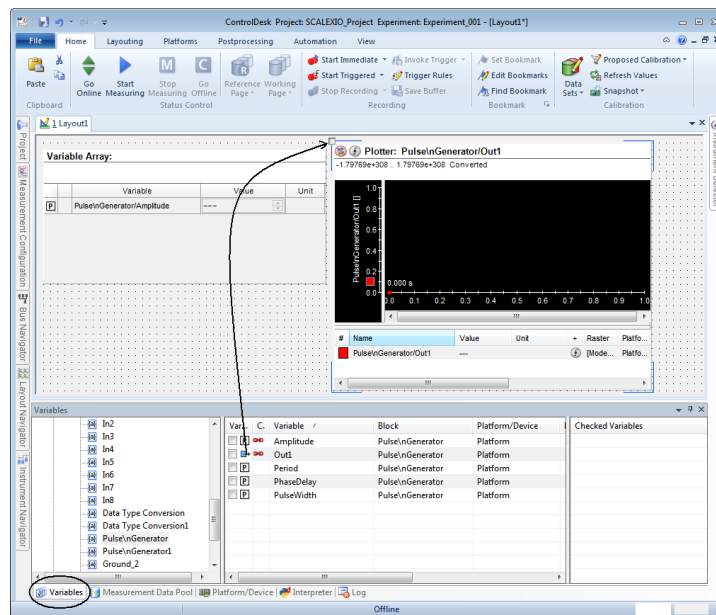
Working with ControlDesk

Introduction

When you have access to the SCALEXIO system, you can start experimenting. Some of the steps you can now take in ControlDesk are shown below.

Selecting variables for experimenting

From ControlDesk's Variables controlbar, drag variables to the layout to visualize them in instruments.



Starting and stopping measurement

To start measuring, go to the Home ribbon and click Status Control – Start Measuring or press F5.

ControlDesk starts online calibration and starts measuring.

To stop measuring, go to the Home ribbon and click Status Control – Stop Measuring or press F6.

Displaying the parameter values of a data set

On the Home ribbon, click **Calibration – Data Sets – Show/List Data Sets** to open the Data Set Manager. Click a data set to list its values.

When you have added the SCALEXIO platform to an experiment, it has a writable working data set. This data set is used for changing parameter values in the experiment.

An experiment can contain a lot of data sets for a platform but only one can be defined as the working data set. You can create a new data set by copying an existing data set.

Creating signals for stimulation









ControlDesk has the Signal Editor which you can use to specify signals and use them for stimulation. You can create and arrange arbitrary signal shapes with the Signal Editor. The signals can be used to stimulate variables of the variable description file. Refer to [Introduction to the Signal Editor \(ControlDesk Signal Editor !\[\]\(0aff635c4179ba9e710b00f4b01d3b20_img.jpg\)](#)).

Bus Navigator

ControlDesk has the Bus Navigator which you can use to handle and manipulate CAN messages, LIN frames, and FlexRay PDUs. The Bus Navigator displays the bus communication in simulation models configured with the CAN MultiMessage blockset, LIN MultiMessage blockset, and FlexRay Configuration blockset. It offers easy access to the various communication configurations of entire CAN, LIN, and FlexRay networks (including all hierarchies and variants) and is always synchronous with the model. For details on the Bus Navigator, refer to [Introduction to the Bus Navigator \(ControlDesk Bus Navigator !\[\]\(0b5e7e25e8775f7e7e80906ada4f0021_img.jpg\)](#)).

Information in the ControlDesk documentation

The following table guides you to more information on basic steps in working with ControlDesk.

Steps in ControlDesk	Refer to ...
Select the variables (parameters and measurement variables) you want to work with.	Basics of the Variables Controlbar (ControlDesk Variable Management )
Visualize the variables in instruments on a layout.	Basics of Handling Layouts (ControlDesk Layouting )
Change the values of parameters.	Variable Types That can be Calibrated (ControlDesk Calibration and Data Set Management )
Configure a measurement or recording.	Basics on Configuring Measurement (ControlDesk Measurement and Recording ) and Basics on Configuring Recording (ControlDesk Measurement and Recording )
Start a measurement or recording.	How to Start Measuring (ControlDesk Measurement and Recording ) and How to Perform an Immediate Recording (ControlDesk Measurement and Recording )
Analyze the results of a measurement or recording.	Introduction to Instrument Handling (ControlDesk Instrument Handling )

Archiving the project and experiment

In ControlDesk you can backup the whole project and experiment. This generates a ZIP archive which you can add to a version control system or transfer to another host PC.

To generate a backup, refer to [Backup Project \(ControlDesk Project and Experiment Management !\[\]\(eafc244b53721dd1ec133f0772f70fc7_img.jpg\)](#)).

To open a ZIP archive, refer to [Open Project + Experiment from Backup \(ControlDesk Project and Experiment Management !\[\]\(d3fb9f94af8b26d1c844efa9a98805b0_img.jpg\)](#)).

Test Automation

Introduction

You can perform the tests automatically. There are two methods to access the real-time application running on the SCALEXIO system.

Where to go from here

Information in this section

[Test Automation Using AutomationDesk and Python Scripts..... 112](#)

You can access the SCALEXIO system using AutomationDesk or Python scripts running on the host PC for test automation. As both access the SCALEXIO system from the host PC, there may be some latency caused by the data transfer between host PC and SCALEXIO system, for example.

[Test Automation Using Real-Time Testing..... 118](#)

You can use Real-Time Testing for test automation. In Real-Time Testing, test scripts are performed synchronously to the real-time application on the SCALEXIO system. There are no latency in relation to the execution of the real-time application and no latencies caused by the host PC.

Information in other sections

[AutomationDesk..... 34](#)

Provides information on the purpose and features of AutomationDesk.

[Real-Time Testing..... 35](#)

Provides information on the purpose and features of Real-Time Testing.

[Experimenting with a SCALEXIO System..... 103](#)

When a real-time application is created for the SCALEXIO system, you can start experimenting.

Test Automation Using AutomationDesk and Python Scripts

Introduction

You can access the SCALEXIO system using AutomationDesk or Python scripts running on the host PC for test automation.

Where to go from here

Information in this section

[Registering the Simulation Platform and Handling the Simulation Application..... 112](#)

To test on a VEOS, SCALEXIO, or MicroAutoBox III system using AutomationDesk, you first have to register the system as the simulation platform. After you load the simulation application to the registered simulation platform, you can start and stop the application.

[Preparing the Test..... 115](#)

Having registered a simulation platform, loaded and started the simulation application, you can create test sequences in AutomationDesk.

[Testing the Simulation Application via XIL API Library..... 116](#)

AutomationDesk lets you perform automated tests. You can use AutomationDesk's XIL API and XIL API Convenience libraries to automate access to the simulation platforms for SIL testing purposes.

Registering the Simulation Platform and Handling the Simulation Application

Introduction

To test on a VEOS, SCALEXIO, or MicroAutoBox III system using AutomationDesk, you first have to register the system as the simulation platform. After you load the simulation application to the registered simulation platform, you can start and stop the application.

This topic shows the steps to register the simulation platform and handle the simulation application.

Registering the simulation platform

To register a simulation platform (if it is not already registered with another dSPACE tool), use the Register Platforms command.

For instructions, refer to [How to Register a dSPACE Platform \(AutomationDesk Accessing Simulation Platforms !\[\]\(bcece9a353e60caece619217f5c1ea39_img.jpg\)](#)).

Loading the simulation application

VEOS To perform an offline simulation on VEOS, you need an *offline simulation application (OSA) file* for the simulation system and variable description file(s) for the V-ECUs and environment models contained in it. To load an OSA file to VEOS ...


- ... *manually*:
Use the Load command.
- ... *via automation*:
Use the LoadSimulationApplication automation block.

SCALEXIO and MicroAutoBox III To perform a real-time simulation on SCALEXIO or a MicroAutoBox III, you need a *real-time application (RTA) file* and variable description file(s) for the V-ECUs and environment models contained in it.

You can load an RTA file to a SCALEXIO or MicroAutoBox III system manually or via automation:

- *Manually*:
Use the Load command.
- *Via automation*:
Use the LoadSimulationApplication automation block.


The real-time application is started after it is loaded to the system, depending on the configuration of the build process.

In AutomationDesk's Platform Manager, the  icon next to the simulation platform indicates that an application is loaded but not running.

Starting the simulation application

To start the simulation application loaded to the simulation platform ...

- ... *manually*:
Use the Start command.
- ... *via automation*:
Use the StartSimulation automation block.


In AutomationDesk's Platform Manager, the  icon next to the simulation platform indicates that a simulation application has been loaded to the platform and is running.

Pausing the simulation application

VEOS To pause the offline simulation application loaded to VEOS ...

- ... *manually*:
Use the Pause command.
- ... *via automation*:
Currently, there is no related automation block.

SCALEXIO and MicroAutoBox III Pausing the real-time application loaded to a SCALEXIO or MicroAutoBox III system is not possible.

AutomationDesk's Platform Manager displays the  icon next to the VEOS platform after pausing an offline simulation application.

Running a paused simulation application stepwise

VEOS To run a paused offline simulation application loaded to VEOS stepwise ...

- ... *manually*:
Use the **Single Step** command.
- ... *via automation*:
Currently, there is no related automation block.

SCALEXIO and MicroAutoBox III Pausing the real-time application loaded to a SCALEXIO or MicroAutoBox III system is not possible.


Stopping the simulation application

To stop the simulation application loaded to the simulation platform ...

- ... *manually*:
Use the **Stop** command.
- ... *via automation*:
Use the **StopSimulation** automation block.

Note

After you stop a real-time application with at least one V-ECU, you cannot restart it. You have to download it to the real-time hardware again first and then restart it.

AutomationDesk's Platform Manager displays the  icon next to the platform after stopping the simulation application.

Unloading the simulation application

To unload the simulation application from the memory of the simulation platform ...

- ... *manually*:
Use the **Unload** command.
- ... *via automation*:
Use the **UnloadSimulationApplication** automation block.

Related topics

Basics

[AutomationDesk.....](#) 34

Preparing the Test

Introduction

Having registered a simulation platform, loaded and started the simulation application, you can create test sequences in AutomationDesk.

This topic shows the steps to prepare a test sequence.

Creating an AutomationDesk project

In AutomationDesk, a *project* is the central element containing all relevant parts and information of an automation task.

To create an AutomationDesk project, use the **New Project** command.

For instructions, refer to [How to Create a New Project \(AutomationDesk Basic Practices !\[\]\(3e2231b1ad3ca8da8658228c00dd08e0_img.jpg\)](#)).

Creating an automation sequence

An *automation sequence* includes the control flow of an automation task and the parameterization of its data objects.

To add a sequence to a project, use the **New Sequence** command.

For instructions, refer to [How to Create Automation Sequences \(AutomationDesk Basic Practices !\[\]\(0d5ec72f61334709c3fc9450209b754f_img.jpg\)](#)).

Accessing environment models via XIL API library

AutomationDesk's *XIL API library* provides automation blocks that let you automate the access to simulation platforms such as the SCALEXIO system.

To create automation sequences using the XIL API library, the project requires some data objects:

- To specify the platform and variable description file (SDF) of the simulation application, you have to add a **MAPortConfiguration** data object to the project. You can use ControlDesk's **Variables** controlbar to add variables in the SDF file as **Variable** data objects to your sequence. Refer to [MAPortConfiguration \(Data Object\) \(AutomationDesk Accessing Simulation Platforms !\[\]\(a16a19bbc0e991a431a3f945e52ea4ee_img.jpg\)](#)).
- To access the simulation application that is running on a simulation platform, you have to add a **MAPort** data object to the project. Refer to [MAPort \(Data Object\) \(AutomationDesk Accessing Simulation Platforms !\[\]\(84adebc4a9e78c4c1c7cf356a810b3d7_img.jpg\)](#)).

For instructions, refer to [How to Prepare a Project for Using the XIL API Library \(AutomationDesk Accessing Simulation Platforms !\[\]\(84f47badaad7772cd95667a7c387a639_img.jpg\)](#)).

Related topics

References

[New Project \(AutomationDesk Basic Practices !\[\]\(aff7c69c44a5e015f18c35867ef3f5c3_img.jpg\)](#))
[New Sequence \(AutomationDesk Basic Practices !\[\]\(0008e7ee33c76c8da181b6f52bf57cf7_img.jpg\)](#))

Testing the Simulation Application via XIL API Library

Introduction

AutomationDesk lets you perform automated tests. You can use AutomationDesk's *XIL API* and *XIL API Convenience* libraries to automate access to the simulation platforms for SIL testing purposes.

Initializing and releasing the MAPort

Initializing the MAPort To use the XIL API library in an automation sequence, the MAPort must be initialized at the beginning of the sequence. For instructions, refer to [How to Initialize a Model Access Port \(AutomationDesk Accessing Simulation Platforms !\[\]\(23d9fc146e83b5c3013cfa32c784f8d5_img.jpg\)\)](#). For reference documentation, refer to [InitMAPort \(AutomationDesk Accessing Simulation Platforms !\[\]\(f5c463b8c1554ac5049d611bd8e33a51_img.jpg\)\)](#).

Releasing the MAPort The MAPort must be released at the end of the sequence. For reference documentation, refer to [ReleaseMAPort \(AutomationDesk Accessing Simulation Platforms !\[\]\(c694a3ff3b077d76910920a6a1593ab4_img.jpg\)\)](#).

Reading and writing variables

AutomationDesk's XIL API library lets you read and write to scalar and vector type variables.

For instructions, refer to [How to Write and Read Variables \(AutomationDesk Accessing Simulation Platforms !\[\]\(aa53ad6fea213b8b2226d3077e30533a_img.jpg\)\)](#).

Data capturing

AutomationDesk's XIL API library lets you capture data from the simulation platform by using complex trigger conditions.

For instructions, refer to [How to Capture Data \(AutomationDesk Accessing Simulation Platforms !\[\]\(fe3aebe81acea8d45108cd2768939da7_img.jpg\)\)](#).

Stimulating variables

AutomationDesk's XIL API library lets you stimulate environment model variables of simulation applications running on VEOS, SCALEXIO, and a MicroAutoBox III.



For stimulus generation, you need an STZ file to be used as a signal generator. The STZ file contains stimulus signals and information on the mapping of these signals to variables of the simulation application to be stimulated.

Tip

You can create an STZ file with ControlDesk's Signal Editor.

For details, refer to [Basics on Stimulating Variables via AutomationDesk \(AutomationDesk Accessing Simulation Platforms !\[\]\(cbd8541a32dfc32f356f5c6c994b0a21_img.jpg\)\)](#).

Refer to [SignalGenerator \(Data Object\) \(AutomationDesk Accessing Simulation Platforms !\[\]\(d3e32d099174a7c248ec1f564ee4f69c_img.jpg\)\)](#).

Executing automation sequences	AutomationDesk lets you execute automation sequences. Refer to Executing Automation Sequences (AutomationDesk Basic Practices ).
Evaluating test results	AutomationDesk lets you evaluate test results. Refer to Implementing Result Evaluation (AutomationDesk Basic Practices ).
Related topics	<p>Basics</p> <div> AutomationDesk..... 34 </div>

Test Automation Using Real-Time Testing

Introduction

You can use Real-Time Testing for test automation. In Real-Time Testing, test scripts are performed synchronously to the real-time application on the SCALEXIO system.

Where to go from here

Information in this section

[Basics on Real-Time Testing..... 118](#)

With Real-Time Testing, you can perform tests synchronously to the real-time application on the SCALEXIO system.

[Executing Automated Tests Using Real-Time Testing..... 119](#)

You can use Real-Time Testing for tests when you want to perform tests synchronously to the real-time application.

Basics on Real-Time Testing

Introduction

With Real-Time Testing, you can perform tests synchronously to the real-time application on the SCALEXIO system.

Overview

Real-Time Testing consists of different parts:

- A library for the RTT sequences that are executed on the SCALEXIO system.
- A Python interpreter that executes the RTT sequences on the SCALEXIO system.
- A Real-Time Test Manager Server that provides functions for managing RTT sequences.

Precondition

You must have experience with Python because the RTT sequences and the scripts for managing them must be implemented in Python.

Components

RTT sequences RTT sequences are the tests which are implemented in Python for real-time testing. The RTT sequences are downloaded to the SCALEXIO system. The RTT sequences and the real-time application are executed

synchronously with the real-time simulation. Refer to [Basics on RTT Sequences \(Real-Time Testing Guide !\[\]\(35e4f762fc1cfea5610d92e2d225d5b4_img.jpg\)](#)).

Python interpreter The Python interpreter is the test engine running on the simulator in parallel to the simulation model. It processes the RTT sequences.

Real-Time Test Manager Server The Real-Time Test Manager Server is a dSPACE Python module. The module is used in host scripts which manage the RTT sequences. It has all the required functions to download, start, pause, continue, or stop RTT sequences. Additionally, it can be used to implement an event handling on the host PC. Refer to [Basics on Running RTT Sequences \(Real-Time Testing Guide !\[\]\(feabb98897b440bc8695a03336a6e2df_img.jpg\)](#)).

Implementing RTT sequences

An RTT sequence is a Python script which is executed on the real-time platform. The scripts must have generator functions that can be paused in one sampling step and resumed in the next sampling step. In the implementation you can use Python modules which are specially implemented for real-time testing and standard Python modules which are suitable for real-time testing. You can also use your own Python modules if they are suitable for real-time testing. Refer to [Implementing RTT Sequences \(Real-Time Testing Guide !\[\]\(83f22ed94ec5517769dd76d702c6bfd8_img.jpg\)](#)).

Limitation The following Real-Time Testing modules are not supported for a SCALEXIO system with a SCALEXIO Processing Unit or DS6001 Processor Board:

- rttlib.rs232lib (sending and receiving data via an RS232 interface)

Managing RTT sequences

RTT sequences are managed using Python scripts running on the host PC or the Real-Time Test Manager.

Python scripts The Python scripts use the Real-Time Test Manager Server. In the Python scripts, you can create the RTT sequences on the SCALEXIO system and start, stop, pause, resume one or all RTT sequences. Refer to [Managing RTT Sequences in Python Scripts \(Real-Time Testing Guide !\[\]\(d0262bbe9d2356661a2e89321dfcc781_img.jpg\)](#)).

Real-Time Test Manager The Real-Time Test Manager is a graphical user interface for managing RTT sequences. You can create the RTT sequences on the SCALEXIO system and start, stop, pause, resume one or all RTT sequences. Refer to [Managing RTT Sequences Using the Real-Time Test Manager \(Real-Time Testing Guide !\[\]\(51514032c8ca341817228f39f1307b05_img.jpg\)](#)).

Executing Automated Tests Using Real-Time Testing

Introduction

You can use Real-Time Testing for tests when you want to perform tests synchronously to the real-time application.

Workflow

Real-time testing is performed in several steps:

1. Prepare the real-time application

A real-time model does not need to be changed for real-time testing. However, the real-time application must contain the service call to the Python interpreter. For a SCALEXIO system, Real-Time Testing is enabled for the fastest periodic task of the real-time application by default. You can change the settings in ConfigurationDesk. Refer to [How to Configure Tasks in ConfigurationDesk \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(6605b201d6f14d9b3bcb8ab5f274d107_img.jpg\)](#)).

2. Create the RTT sequence

An RTT sequence is a Python script which is executed on the real-time platform. For information on how to implement an RTT sequence. Refer to [Implementing RTT Sequences \(Real-Time Testing Guide !\[\]\(fa6f3af6bfa46c5d4a2d362681095beb_img.jpg\)](#)).

Note that RTT sequences implemented for a SCALEXIO system cannot use all the features of Real-Time Testing.

3. Generate the BCG file

A BCG file contains the RTT sequence and all the modules which are imported by the RTT sequence. BCG files must be signed. Only signed BCG files can be downloaded to the SCALEXIO system. The Real-Time Test Manager generates the BCG file automatically when you download the RTT sequence.

4. Executing the real-time test

When the BCG file is generated, you can download it to the SCALEXIO system for executing the test. You can control the execution using a Python script on the host PC or the Real-Time Test Manager. For information on accessing a SCALEXIO system, see [Accessing the SCALEXIO system](#) on page 120. For general information on managing the tests, refer to [Managing RTT Sequences \(Real-Time Testing Guide !\[\]\(d8ab143e904bfa3467271eec5af75a9b_img.jpg\)](#)).

5. Error management and troubleshooting

If your real-time test does not work in the desired way, you can stop the execution and change the implementation of the RTT sequence.

- For tips and tricks for implementing, refer to [Writing Effective RTT Sequences \(Real-Time Testing Guide !\[\]\(f0543fe51acd79be3858008749d93a88_img.jpg\)](#)).
- For information on how you can debug an RTT sequence, refer to [Debugging RTT Sequences \(Real-Time Testing Guide !\[\]\(b452a1210835992e25e075124622531b_img.jpg\)](#)).
- For information on general errors, refer to [Troubleshooting \(Real-Time Testing Guide !\[\]\(7bc2b99ff222bd0a25e1cf77d692b0e7_img.jpg\)](#)).

Accessing the SCALEXIO system

A SCALEXIO system is connected to the host PC via Ethernet. You must therefore specify its IP address and the name of the real-time application when you want to access a SCALEXIO system for real-time testing.

Python scripts For example, if the SCALEXIO system has the IP address 100.200.3.4 and the name of the real-time application is "myApplication", you access it as follows in a Python script:


```
Board = rtm.AccessBoard("100.200.3.4/myApplication")
```

Real-Time Test Manager In the Real-Time Test Manager, you access a SCALEXIO system as follows:

1. Choose Tools – Register Platform.
2. In the dialog, select SCALEXIO.
3. Enter the IP address and click Register.

Simulating Electrical Errors in the Wiring

Introduction You can simulate electrical errors in the wiring of the external devices that is connected to the SCALEXIO system.

Where to go from here

Information in this section

[Basics of Electrical Errors Simulation in a SCALEXIO System.....](#) 124
The HighFlex I/O Boards and MultiCompact I/O units of a SCALEXIO system are prepared for simulating electrical errors in the wiring from an ECU to sensors, actuators, and buses and between these components.

[Allowing Electrical Error Simulation for Pins of the External Devices.....](#) 137
You must specify which failure class is allowed for the pins when configuring the SCALEXIO system in ConfigurationDesk.

[Executing Electrical Error Simulation.....](#) 141
Performs information on how to perform electrical error simulation.

Basics of Electrical Errors Simulation in a SCALEXIO System

Introduction

The HighFlex I/O Boards and MultiCompact I/O units of a SCALEXIO system are prepared for simulating electrical errors in the wiring from an ECU to sensors, actuators, and buses and between these components.

Where to go from here

Information in this section

[Basics on Electrical Error Simulation with SCALEXIO Systems..... 124](#)
Gives an overview of the integrated failure simulation hardware of a SCALEXIO System.

[Hardware for Electrical Error Simulation on SCALEXIO Systems..... 128](#)
To perform electrical error simulation, your SCALEXIO system must provide the required failure simulation hardware and be operated by a SCALEXIO Processing Unit.

[License Required for Electrical Error Simulation..... 132](#)
Gives information on the license required for electrical error simulation.

[Safety Precautions for Simulating Electrical Errors with a SCALEXIO System..... 133](#)
Provides safety precautions for simulating electrical errors with a SCALEXIO system.

[Workflow for Simulating Electrical Errors..... 135](#)
Describes the process of simulating electrical errors with SCALEXIO systems.

Information in other sections

[Executing Electrical Error Simulation..... 141](#)
Performs information on how to perform electrical error simulation.

[Allowing Electrical Error Simulation for Pins of the External Devices..... 137](#)
You must specify which failure class is allowed for the pins when configuring the SCALEXIO system in ConfigurationDesk.

Basics on Electrical Error Simulation with SCALEXIO Systems

Introduction

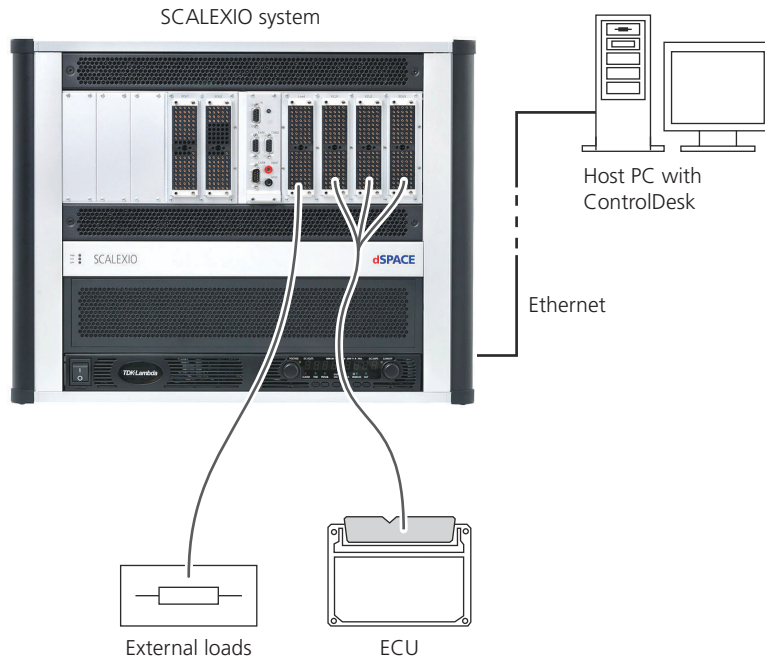
To perform electrical error simulation with a SCALEXIO system, you can use the integrated failure simulation hardware of SCALEXIO components.

Overview

The important components for electrical error simulation are:

- Failure simulation hardware for inserting electrical errors. Refer to [Hardware for Electrical Error Simulation on SCALEXIO Systems](#) on page 128.
- Host PC with ControlDesk and the Failure Simulation Package

The following illustration shows an example of a rack-based SCALEXIO system that is connected to an ECU, external loads, and the host PC.



Required licenses

To simulate electrical errors, two kinds of licenses are required:

- A license for using the Failure Simulation Package of ControlDesk
- A license for using the failure simulation hardware provided by a SCALEXIO system

For more information, refer to [License Required for Electrical Error Simulation](#) on page 132.

Electrical errors

The integrated SCALEXIO failure simulation hardware lets you switch the following electrical errors:

- *Open circuit* to simulate a broken wire or loose contact
- *Short circuit*
 - To ground (KL31)
 - To power switch channels with battery voltage (KL15, KL30)
 - Between channels of the same signal category (e.g., a short circuit between signal measurement channels)
 - Between channels of different signal categories (e.g., a short circuit between signal measurement channels and signal generation channels)

- Between signal measurement channels or signal generation channels and bus channels
- Pulsed switching for signals of signal measurement channels and signal generation channels (e.g., to simulate loose contacts or relay contact bouncing)

Note

You cannot use pulsed switching for bus signals or when simulating multiple electrical errors.

Except for the *Analog In 2* and *Analog Out 2* channel types, switching electrical errors is performed only for the signal/bus lines, not for reference lines.

Load or signal disconnection

In most cases, you can disconnect the loads or signals during electrical error simulation (*load rejection*). For example, to protect sensitive loads of signal measurement channels. Limitations apply only when you work with multi-pin errors. Refer to [Switching short circuits between multiple signals and/or bus channels](#) on page 128.

Loads and signals can be disconnected by switching semiconductor switches or relays. For technical reasons, semiconductor switches cannot be used in all cases. Relays must switch the load or signal disconnection in the following cases:

- Short circuit between two signal generation or signal measurement channels
- Short circuit of a bus channel to any other channel type
- Multiple electrical errors when activation by an FRU relay is necessary

NOTICE

The FRU relays might be carrying a current when you are using them to disconnect the loads or signals. Note the related warnings listed in [Safety Precautions for Simulating Electrical Errors with a SCALEXIO System](#) on page 133.

To switch an FRU relay that might be carrying a current, you must set the corresponding property of the channel in ConfigurationDesk. Otherwise you cannot switch this kind of error in combination with this channel.

Pulsed switching

The central FIU uses semiconductor switches for switching the electrical errors. It is able to switch very fast (pulsed switching). This makes it possible to simulate loose contacts or defined switch bouncing. However, pulsed switching is not possible in all combinations of channel types.

Pulsed switching is not supported for

- Electrical errors when bus channels are involved
- Multiple electrical errors when activation by an FRU relay is necessary

Switching multiple electrical errors

The integrated SCALEXIO failure simulation hardware supports switching multiple electrical errors at the same time or in succession. For example, you can simulate an open circuit for one channel and a short circuit for another channel at the same time, without deactivating the first error.

Note

When multiple electrical errors are simulated, switching can be done by relays that are carrying a current. Note the related warnings listed in [Safety Precautions for Simulating Electrical Errors with a SCALEXIO System](#) on page 133.

Allowed electrical errors You can simulate the following electrical errors at the same time:

- Any number of open circuits (interrupts)
- Multiple short circuits to one power switch channel (short to battery voltage or another potential)
 - Up to 10 short circuits with a DS2642 FIU & Power Switch Board
 - Up to 6 short circuits with a DS2680 I/O Unit
- Multiple short circuits to ground (short to GND)
 - Up to 10 short circuits with a DS2642 FIU & Power Switch Board
 - Up to 6 short circuits with a DS2680 I/O Unit
- Multiple short circuits between signal measurement, signal generation and bus channels (short to pins, multi-pin errors (refer to [Switching short circuits between multiple signals and/or bus channels](#) on page 128))
 - Short circuits between up to 10 channels (one multi-pin error)
 - Two multi-pin errors in parallel (2×10 channels)
- Combinations of any number of open circuits (interrupts) with:
 - Multiple short circuits to one power switch channel (short to VBAT)
 - Multiple short circuits to ground (short to GND)
 - Multiple short circuits between channels (short to pins)
 - One error (short circuit to any potential or open circuit) of a signal that uses current enhancement
 - One error (short circuit to any potential or open circuit) with pulsed switching

Limitations

- In general, switching multiple electrical errors is limited by the allowed maximum current of the channels and failrails involved.
- Switching multiple electrical errors with channels that use current enhancement is not supported (except for channel multiplication of the Power Switch 1 channel type of the DS2642 FIU & Power Switch Board).
- Pulsed switching is supported for one signal only.

Switching short circuits between multiple signals and/or bus channels

Multi-pin errors Multi-pin errors let you simulate a short circuit between three or more signal channels and/or bus channels. The channels can be located on the same or different boards or I/O units. You can simulate a short circuit between:

- Channels of the same signal category (e.g., four signal generation channels)
- Channels of different signal categories (e.g., three signal generation channels and two signal measurement channels)
- Signal channels and bus channels (e.g., two signal generation channels, one signal measurement channel, and one bus channel)

Switching multi-pin errors The both failrails of a SCALEXIO system are used automatically by the XIL API EESPort according to the specified error configuration.


Multi-pin errors can be switched only by relays. Therefore, you must use ConfigurationDesk to allow the activation by FRU relays for each involved channel.

Load or signal disconnection You can disconnect loads or signals from channels that are used for multi-pin errors. To disconnect loads or signals, you must switch the channels to failrail 1. You cannot use failrail 2 in this case. Loads or signals can be disconnected only by relays.

Note

You can switch multi-pin errors and disconnect loads or signals by using relays that might be carrying a current. Note the related warnings listed in [Safety Precautions for Simulating Electrical Errors with a SCALEXIO System](#) on page 133.

Monitoring the switching behavior

With an experiment software such as ControlDesk, you can monitor and trace the switching behavior of the SCALEXIO failure simulation hardware. For more information, refer to [Monitoring the Switching Behavior of the Failure Simulation Hardware](#) (ControlDesk Electrical Error Simulation via XIL API EESPort ).

Hardware for Electrical Error Simulation on SCALEXIO Systems

Introduction

To perform electrical error simulation, your SCALEXIO system must provide the required hardware and be operated by a SCALEXIO Processing Unit.

Electrical error simulation concept of a SCALEXIO system

To simulate failures (electrical errors) in an ECU wiring, the following hardware components are required in a SCALEXIO system:

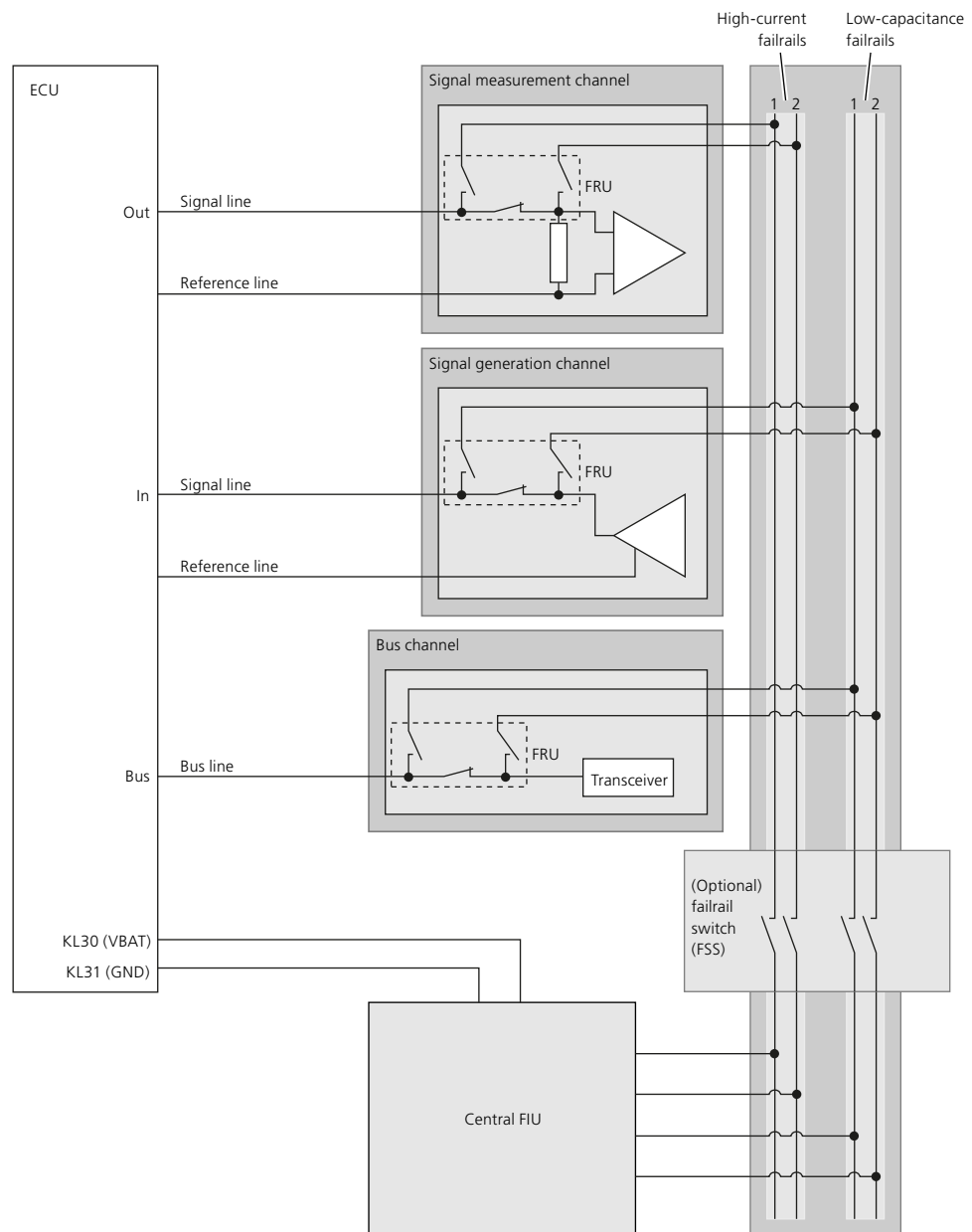
Central failure insertion unit A central failure insertion unit (central FIU), which is located, for example, on an DS2642 FIU & Power Switch Board or a DS2680 I/O Unit. Refer to [Central FIU](#) on page 130.

Failure routing units Single failure routing units (FRUs), which are located, for example, in the signal or bus lines of the single channels. Refer to [Failure routing units](#) on page 131.

Failrails The failrails which connect the central FIU to the single failure routing units (FRUs) via the backplanes of the SCALEXIO slot or I/O units. Refer to [Failrails](#) on page 131.

Failrail segment switches (optional) One or more (optional) failrail segment switches which are used for selectively connecting/disconnecting SCALEXIO slot or I/O units to/from the common failrails of the SCALEXIO system. Refer to [Failrail segment switches](#) on page 131.

The following illustration shows the hardware components that are required for electrical error simulation.



SCALEXIO Processing Unit The SCALEXIO system must consist of a SCALEXIO rack and be operated by a SCALEXIO Processing Unit. (A SCALEXIO system with SCALEXIO LabBox and a DS6001 Processor Board does not support electrical error simulation.)

Central FIU

The *central FIU* is the major component of the SCALEXIO failure simulation hardware.

Usually, the central FIU activates a single electrical failure which is preconfigured by a channel's *failure routing unit* (FRU). For failure simulation on a bus channel

and for simulating multiple failures, the central FIU does not activate a failure but only preconfigures it. The failure is activated by the FRU(s) of the channel(s) involved.

The central FIU is connected to the single FRUs via the *failrails* located on the backplane of a SCALEXIO slot unit or I/O unit. The central FIU is located on a DS2642 FIU & Power Switch Board or a DS2680 I/O Unit.

The central FIU has semiconductor switches. In comparison to the relays used for the FRUs, these switches are able to switch very fast without bouncing (pulsed switching). It is therefore possible to simulate loose contacts or defined switch bouncing with the semiconductor switches of the central FIU.

Failrails

The *failrails* connect the single *failure routing units* (FRUs) to the *central FIU*. As signals for signal measurement and signal generation have different requirements, a SCALEXIO system can have different failrails:

- *Low-capacitance failrails* to connect the FRUs of signal generation channels and bus channels to the central FIU. These failrails have a low capacitance related to signal ground (KL 31) to disturb the signals as little as possible.
- *High-current failrails* to connect the FRUs of signal measurement channels to the central FIU. These failrails are able to carry a high current.

Failrail segment switches

The SCALEXIO failrail system can contain optional *failrail segment switches*. These switches are used for selectively connecting/disconnecting slot units and I/O units to/from the common failrails of the SCALEXIO system. Failrail segment switches can also switch the inter-rack connections of the failrail system in a multi-rack SCALEXIO system.

The selective disconnection of large sections of the failrails by using failrail segment switches minimizes system-inherent parasitic effects (e.g., parasitic capacitances of semiconductor switches).

Failure routing units

For SCALEXIO HighFlex I/O boards and MultiCompact I/O units and boards, each signal channel and each bus channel provide one or more *failure routing units* (FRUs) to connect the channel to the failrails. (Standard SCALEXIO I/O boards do not have failure routing units and do not support electrical error simulation.)

An FRU has *FRU relays* for switching. That is why FRUs cannot be used to simulate pulsed switching. Switching an FRU relay can be accompanied by contact chatter.

The FRU relays are used and switched as follows:

- *To simulate an electrical error of a single signal line*, the FRU relays preconfigure the error first. The error is then actually activated by the semiconductor switches of the central FIU.

If parallel channels are used, for example, for current enhancement, they are switched synchronously.

- *To simulate an electrical error of a single bus line*, the semiconductor switches of the central FIU do not activate the error but only preconfigure it. The error is activated by the FRU relays, because they disturb the bus signals as little as possible.
- *To simulate multiple electrical errors*, the first of the errors – except for an error in a bus line – is switched the same way as when switching a single error. This means that the first error is preconfigured by switching the channel's FRU relays first and then actually activated by the semiconductor switches of the central FIU. Depending on the selected error type, the second and all other errors can be activated by the FRU relays without preconfiguring the error.
- *To disconnect the loads or signals* during electrical error simulation (load rejection). When simulating errors:
 - A load can be disconnected from a signal measurement channel.
 - A generated signal can be disconnected from a signal generation or bus channel.

Short-circuit detection units

Software-controlled short-circuit detection units are implemented in the connection between each power switch channel and the central FIU. They protect the semiconductors of the central FIU against overcurrent if the fuses implemented in the signal measurement, signal generation, and/or bus channels react too slowly.

If the signal measurement, signal generation, or bus board detects an overcurrent, the semiconductor of the central FIU are opened. The status LEDs of the boards display the error state, and the short-circuit detection units report the overcurrent event to the SCALEXIO Processing Unit.

Related topics

Basics

[Basics on Electrical Error Simulation with SCALEXIO Systems..... 124](#)

License Required for Electrical Error Simulation

Introduction

A license is required to use the failure simulation hardware provided by a SCALEXIO system.

SCALEXIO_FIU_<n> license

The integrated failure simulation hardware of a SCALEXIO system is license-protected. The failure routing units (FRUs) are already connected to the I/O channels, so the hardware is prepared for electrical error simulation. To enable electrical error simulation of a SCALEXIO system, a SCALEXIO_FIU_<n> license must be installed.

The licenses are available in different sizes for different numbers of instantiated function blocks in your implemented ConfigurationDesk application, see the following table.

License	Description
SCALEXIO_FIU_100	License for simulating electrical errors with a real-time application implemented with up to 100 function blocks in ConfigurationDesk.
SCALEXIO_FIU_200	License for simulating electrical errors with a real-time application implemented with up to 200 function blocks in ConfigurationDesk.
SCALEXIO_FIU_300	License for simulating electrical errors with a real-time application implemented with up to 300 function blocks in ConfigurationDesk.
SCALEXIO_FIU_500	License for simulating electrical errors with a real-time application implemented with up to 500 function blocks in ConfigurationDesk.
SCALEXIO_FIU_1000	License for simulating electrical errors with a real-time application implemented with up to 1000 function blocks in ConfigurationDesk.
SCALEXIO_FIU_FULL	License for simulating electrical errors with a real-time application implemented with an unlimited number of function blocks in ConfigurationDesk.

ConfigurationDesk counts the instantiated function blocks in the same way as for function block licenses. For details on the function block licenses, refer to [Details on Function Block Licenses \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(bd1a142de767a21e5362c595f844a4ff_img.jpg\)](#)).

The SCALEXIO_FIU_<n> licenses are available only as dongle licenses. They must be installed on the host PC which you use for downloading the real-time application.

The license is checked when the real-time application is downloaded. If no license is available, a warning is given and subsequent electrical error 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 (provided the SCALEXIO_RTLIB license is installed, see [SCALEXIO_RTLIB license](#) on page 106).

Safety Precautions for Simulating Electrical Errors with a SCALEXIO System

Introduction

You must consider some safety precautions to perform electrical error simulation with a SCALEXIO Systems.

General warning

WARNING

Risk of unexpected high currents and voltages due to electrical error simulation

During electrical error simulation, high currents and voltages might be present on board channels and/or connector pins, which is not expected. This can result in death, personal injury, fire, and/or damage to the SCALEXIO system and connected external devices.

Note

Especially signal measurement channels (such as channels connected in parallel or interconnected reference lines) can carry high currents.

You must ensure that no voltages or currents outside the specified ranges of the I/O channels can occur during electrical error simulation.

Switching FRU relays

To prevent damage to the relays, switching is normally done when no current load is present. However, a current load cannot be avoided during switching in the following cases:

- Switching multiple electrical errors
- Load or signal disconnection when switching short circuits between channels

If FRU relays are switched for electrical error simulation, note the following warnings.

NOTICE

Risk of increased wear and permanent damage to the relays of the integrated SCALEXIO failure simulation hardware

To a varying extent, depending on which loads are connected and which currents and voltages are switched, electric arcs and contact erosion can occur in the FRU relays involved. This will eventually destroy the FRU relays. The board or unit on which they are mounted will no longer be usable and will have to be repaired or replaced.

Operating the relays outside the permitted range (i.e., above the maximum switching capacity) can also destroy the relays and will probably damage the board or unit on which they are mounted.

Before using FRU relays for electrical error simulation, you must fulfill the following preconditions:

- To minimize the risk of damage in a multiple error scenario, operate the relays only within the permitted conditions and ranges (i.e., under the maximum switching capacity). For concrete values, refer to [FRU Relays Data Sheet \(SCALEXIO Hardware Installation and Configuration !\[\]\(00454fbbe8db418db0de5eebfa916a08_img.jpg\)](#)).
- If the connected loads are inductive and you want to simulate electrical errors in their wiring, you must protect the dSPACE hardware from induced high voltages. Refer to [Safety Precautions for Using Inductive Loads \(SCALEXIO Hardware Installation and Configuration !\[\]\(fd0f3d0c9a8d9b3ff3951bcf7c4bf0c0_img.jpg\)](#)).
- Consider the increased risk of material wear or damage before you set Activation by FRU relay to Allowed in ConfigurationDesk.

Note that defects caused by material wear, misuse or operation outside the permitted ranges are not covered by any warranty, and no liability is accepted by dSPACE for any direct or indirect damage arising from such defects.

NOTICE

Risk of damage to a connected load

When FRU relays are switched for electrical error simulation, load rejection can be delayed up to 30 ms due to the switching times of the relays involved. A connected load can be damaged during these 30 ms.

Workflow for Simulating Electrical Errors

Introduction

This topics describes the process of simulating electrical errors with SCALEXIO systems.

Workflow

NOTICE

Before performing electrical error simulation, note the warning given in [Safety Precautions for Simulating Electrical Errors with a SCALEXIO System](#) on page 133.

For electrical error simulation you have to perform three steps:

1. Specifying the allowed failure class
You can specify the electrical errors which are allowed for a pin in ConfigurationDesk. Refer to [Allowing Electrical Error Simulation for Pins of the External Devices](#) on page 137.
2. Building the real-time application
In the build process, all the data necessary for simulating electrical errors is created and stored in the RTA file. If you change the failure specification after building the real-time application, the build process must be repeated.
3. Switching the failures
The failures are manually switched using the Failure Simulation Package of ControlDesk or they are automatically switched using automated testing based on dSPACE XIL API .NET and its EESPort implementation. Only failures which are allowed in ConfigurationDesk can be set. Refer to [Executing Electrical Error Simulation](#) on page 141.

Allowing Electrical Error Simulation for Pins of the External Devices

Introduction

You must specify which failure class is allowed for the pins when configuring the SCALEXIO system in ConfigurationDesk.

Specifying Allowed Failure Classes for ECU Pins

Introduction

Before you can simulate electrical errors, you must specify which failure class can be switched for an ECU pin.

Electrical error simulation and parallel channels

If you use several channels for one signal (parallel channels), for example, for current enhancement, the configuration of allowed failure classes applies to all the parallel channels. When the electrical error is switched, all the channels are switched simultaneously.

Possible failure classes

You can specify the allowed failure classes for each pin:

Allowed Failure	Description
Open circuit	The connection between ECU and simulator can be opened. This simulates a broken wire.
Short to GND	The signal of the ECU can be connected to ground.
Short to VBAT	The signal of the ECU can be connected to the battery voltage.
Short to signal generation channel	The signal of the ECU can be connected to a signal generation channel.
Short to signal measurement channel	The signal of the ECU can be connected to a signal measurement channel.
Short to bus channel	The signal of the ECU can be connected to a bus channel.

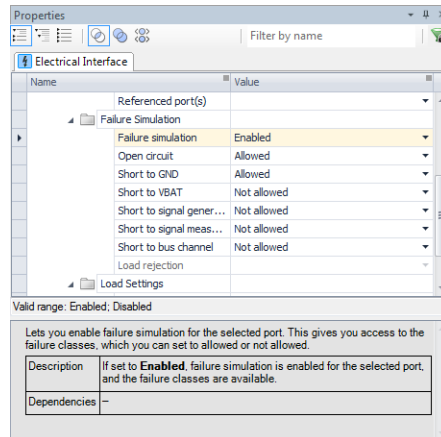
Note

There are some limitations for pulsed switching and load/signal disconnection. Refer to [Basics on Electrical Error Simulation with SCALEXIO Systems](#) on page 124.

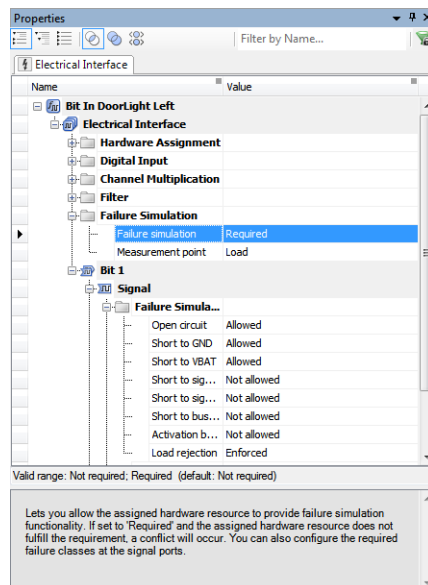
Specifying the allowed failure classes in ConfigurationDesk

The allowed failure classes are specified in ConfigurationDesk. The settings for electrical error simulation (failure simulation) can be specified as properties for the device ports of device blocks and signal ports of function blocks. If these settings of a device port and the mapped function ports differ, ConfigurationDesk displays a warning. You can only specify these settings for signals, reference lines cannot be used for electrical error simulation.

The following illustration shows the properties for the settings of electrical error simulation of a device block.



The following illustration shows the properties for the failure simulation settings of a function block.



Tip

You can specify the settings for electrical error simulation (failure simulation) for an external device block (ECU, load) and transfer the settings to the mapped function block. Refer to [Transfer Settings – Allowed Failure Classes \(ConfigurationDesk User Interface Reference !\[\]\(c507f772dba2b921f86777f01218e570_img.jpg\)](#)).

For details on specifying the allowed failure classes, refer to [Specifying Failure Simulation in ConfigurationDesk \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(4729e517bc6a7cd81c8025b9646574fb_img.jpg\)](#)).

Short circuits between pins of different channel categories
Note

You cannot activate short circuits between pins of different channel categories in ControlDesk, if the allowed failure classes are specified incompletely in ConfigurationDesk.

To activate short circuits between pins of different channel categories (i.e., signal measurement channels, signal generation channels, and bus channels), the desired signals have to allow each other their corresponding failure class in ConfigurationDesk.

The following example shows the required setting in ConfigurationDesk for a pin of a signal measurement channel that is allowed for a short circuit to a signal generation channel.

Failure Simulation	
Open circuit	Not allowed
Short to GND	Not allowed
Short to VBAT	Not allowed
Short to signal generation cha...	Allowed
Short to signal measurement c...	Not allowed
Short to bus channel	Not allowed
Activation by FRU relay	Not allowed
Load rejection	Enforced

For the corresponding signal generation channel, you have to allow a short circuit to a signal measurement channel.

Failure Simulation	
Open circuit	Not allowed
Short to GND	Not allowed
Short to VBAT	Not allowed
Short to signal generation cha...	Not allowed
Short to signal measurement c...	Allowed
Short to bus channel	Not allowed
Activation by FRU relay	Not allowed
Load rejection	Enforced

Activation by FRU relay

By default, the electrical errors are actually switched by the central FIU and the FRU relays are only switched without load current. Because this leads to some

technical limitations, you can also allow the use of FRU relays with load current. This allows to extend the following operations:

- Switching multiple failures
- Disconnecting loads or signals when they cannot be disconnected by semiconductor switches

To allow the use of FRU relays for a channel, you must enable it in ConfigurationDesk by setting *Activation by FRU relay* to *Allowed*.

NOTICE

Using the FRU relays for failure simulation stresses the relays. Note the warnings given in [Safety Precautions for Simulating Electrical Errors with a SCALEXIO System](#) on page 133.

Executing Electrical Error Simulation

Introduction

When you have specified the electrical errors that are allowed and the application is downloaded to the SCALEXIO system, you can simulate the electrical errors.

Executing Electrical Error Simulation

Introduction


You can manually perform electrical error simulation or use an automation interface.

ASAM AE XIL API

In the context of dSPACE products, electrical error simulation is based on the ASAM AE XIL API 2.1.0 standard.


Electrical error simulation

Since dSPACE Release 2015-B, ControlDesk provides a graphical user interface (GUI) for electrical error simulation that is delivered with the Failure Simulation Package. The GUI bases on dSPACE XIL API .NET and its EESPort implementation. This GUI for electrical error simulation is targeted only at the initial operation of the failure simulation hardware and to perform first manual tests.

For details on working with the GUI, refer to [ControlDesk Electrical Error Simulation via XIL API EESPort](#) .

EESPort application

For extensive tests, it is recommended to use automated testing based on dSPACE XIL API .NET and its EESPort implementation, which is also delivered with the Failure Simulation Package.

For further information on electrical error simulation via automated testing, refer to [Implementing an EESPort Client Application \(dSPACE XIL API Implementation Guide\)](#) .

Troubleshooting

Introduction If you encounter a problem with the SCALEXIO system, refer to the information given in this section.

Where to go from here

Information in this section

[No Battery Voltage After the Overcurrent Protection is Activated..... 143](#)

The battery simulation power supply unit cannot be activated after an overcurrent protection is activated.

[Offset in Time of Signals of SCALEXIO and PHS-Bus-Based Systems..... 144](#)

In ControlDesk, signals in a SCALEXIO system and PHS-bus-based system are not synchronous, although they were triggered at the same time.

No Battery Voltage After the Overcurrent Protection is Activated

Problem

The battery simulation power supply unit cannot be activated after an overcurrent protection is activated. This happens when the following steps were performed:

- The real-time application was started and the overcurrent protection is enabled. This means the Overcurrent protection property of the Power Supply Control function block is Shutdown, refer to [Common Function Block Properties \(Power Supply Control\) \(ConfigurationDesk Function Block Properties !\[\]\(1f101ad452ef9a3f01bb1e89af34fc34_img.jpg\)](#)).
- A current in the power supply exceeds the current limit which activates the overcurrent protection.
- The real-time application is stopped.

- The overcurrent protection is deactivated. This means the Overcurrent protection property of the Power Supply Control function block is Saturation, refer to [Common Function Block Properties \(Power Supply Control\) \(ConfigurationDesk Function Block Properties !\[\]\(cd3e54d951a9fb854f48e4697cf550f9_img.jpg\)](#)).
- The real-time application is restarted but the battery simulation power supply unit cannot be activated. Its display shows "OFF".

Solution

There are several ways of solving the problem:

- Reload the real-time application. This reinitializes the battery simulation power supply unit.
- Before you deactivate the overcurrent protection and restart the real-time application, reactivate the overcurrent protection. This can be done in run time using the Release Shutdown function port of Power Supply Control function block, refer to [Voltage Out Function Port Properties \(Power Supply Control\) \(ConfigurationDesk Function Block Properties !\[\]\(48a7667d09d5a06397e047ee4537bb6f_img.jpg\)](#)).
- Reset the overcurrent protection automatically during real-time application stop. This can be done by configuring the Power Supply Control function block as follows:
 1. In the Model Interface, set Stopped status output to Use configured stop values.
 2. For the Release Shutdown function port, set Stop value to Active.
 3. Use the Output Enable run-time parameter to activate or deactivate the output of the battery simulation power supply unit.

For details on the Power Supply Control function block, refer to [Power Supply Control \(ConfigurationDesk Function Block Properties !\[\]\(96cc62f861fdd6e50510c0224a756dff_img.jpg\)](#)).

Offset in Time of Signals of SCALEXIO and PHS-Bus-Based Systems

Problem

In ControlDesk, signals in a SCALEXIO system and PHS-bus-based system are not synchronous, although they were triggered at the same time.

Reason

The host PC and the SCALEXIO system communicate via an Ethernet connection. In an Ethernet network, one or more Ethernet switches may be between them. As the transmission time from and to a SCALEXIO system differs, it is not possible to consider this time.

Solution

Use a peer-to-peer connection between host PC and the SCALEXIO system. Refer to [Connecting the SCALEXIO System to the Host PC \(SCALEXIO Hardware Installation and Configuration !\[\]\(4688aadfd656ded00cd6bdfae55089a9_img.jpg\)](#)).

Glossary

Introduction	The glossary briefly explains the most important expressions and naming conventions used in the SCALEXIO documentation.
--------------	---

Where to go from here

Information in this section

A.....	146
B.....	147
C.....	148
D.....	149
E.....	151
F.....	152
G.....	153
H.....	154
I.....	154
L.....	155
M.....	155
N.....	156
P.....	156
R.....	158
S.....	159
T.....	160
U.....	161

V.....	161
W.....	162

A

Analog In 1 channel type The *Analog In 1* channel type is used to measure an analog voltage signal in the range 0 V ... +60 V.

Analog In 16 channel type The *Analog In 16* channel type is used to measure an analog voltage signal in the range -10 V ... +10 V or -60 V ... +60 V, e.g., for phase current measurement.

Analog In 2 channel type The *Analog In 2* channel type is used to measure analog voltage signals of sensors that require a fast response between an input and output channel.

Analog In 4 channel type The *Analog In 4* channel type is used to measure an analog voltage signal in the range 0 V ... +60 V.

Analog In 5 channel type The *Analog In 5* channel type is used to measure analog voltage signals of sensors that require a fast response between input and output channels. You can measure voltages in the range of ± 2 V ... ± 10 V.

Analog In 6 channel type The *Analog In 6* channel type is used to measure an analog voltage signal in the range -10 V ... +10 V.

Analog input A collection of functions for measuring voltages or currents with the SCALEXIO system.

Analog Out 1 channel type The *Analog Out 1* channel type is used to output voltages for sensor simulation.

Analog Out 2 channel type The *Analog Out 2* channel type is used to output voltages or currents to simulate sensors that require a fast response between input and output channels.

Analog Out 3 channel type The *Analog Out 3* channel type is used to output AC voltages (transformer-coupled) for sensor simulation.

Analog Out 4 channel type The *Analog Out 4* channel type is used to output voltages or currents for sensor simulation.

Analog Out 6 channel type The *Analog Out 6* channel type is used to output voltages for sensor simulation.

Analog Out 7 channel type The *Analog Out 7* channel type is used to output voltages or currents to simulate sensors that require a fast response between input and output channels. The channel is galvanically isolated from system ground.

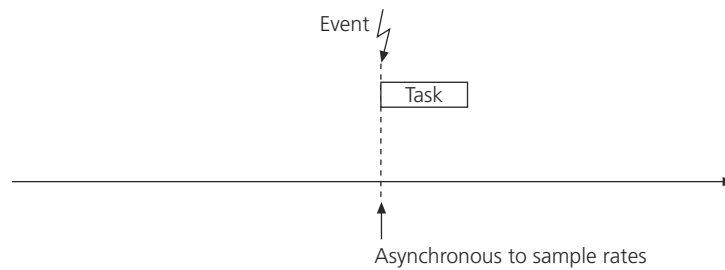
Analog Out 8 channel type The *Analog Out 8* channel type is used to output AC voltages (transformer-coupled) for sensor simulation.

Analog Out 9 channel type The *Analog Out 9* channel type is used to output voltages or currents for sensor simulation.

Analog output A collection of functions for generating a voltage or a current sink with the SCALEXIO system.

Asynchronous tasks Asynchronous tasks are triggered by asynchronous events, e.g., the rising edge of an input signal. The events occur asynchronously to the simulation time (real time).

Some function block types in ConfigurationDesk can generate I/O events. You can assign these events to tasks to trigger the tasks asynchronously.



Note

FlexRay and LIN communication is based on the interaction between several asynchronous tasks.

Automotive Simulation Model (ASM) A Simulink model that is intended for simulation of an automotive engine (gasoline and Diesel) and vehicle dynamics. All the Simulink blocks in the model are visible, so it is possible to add or replace components with custom models to adapt the properties of modeled components to individual requirements.

B

Backplane A printed circuit board that connects all the boards inside a SCALEXIO slot unit or SCALEXIO LabBox.

Battery simulation controller The battery simulation controller sets the voltage and current values of the battery simulation power supply unit.

Battery simulation power supply unit The battery simulation power supply unit is used to simulate the vehicle battery. The battery simulation power supply unit is controlled via software by a battery simulation controller.

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). The behavior model is modeled in Simulink. A behavior model and an I/O model form a real-time model.

Bus 1 channel type The *Bus 1* channel type allows communication via different bus or serial interface types.

Bus board A board that is part of a SCALEXIO system. It provides a bus interface or serial interface to the SCALEXIO system. The bus or serial interface type is specified in ConfigurationDesk by assigning a function to one or more channels.

C

CAN 1 channel type The *CAN 1* channel type allows communication via a CAN bus type according to ISO11898-2 or ISO 11898-3.

CAN 2 channel type The *CAN 2* channel type allows CAN/CAN FD communication via a CAN bus according to ISO 11898, ISO 11898-2:2016, ISO 11898-3.

Central FIU The major component of the SCALEXIO failure simulation hardware. Usually, the central FIU activates a single electrical error which is preconfigured by a channel's failure routing unit (FRU). For electrical error simulation on a bus channel and for simulating multiple errors, the central FIU does not activate an error but only preconfigures it.

Channel multiplication Some SCALEXIO I/O hardware – mainly HighFlex I/O board and MultiCompact I/O units – supports channel multiplication, which lets you connect channels in parallel or series if the signals you want to test exceed the physical ranges of one channel.

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.

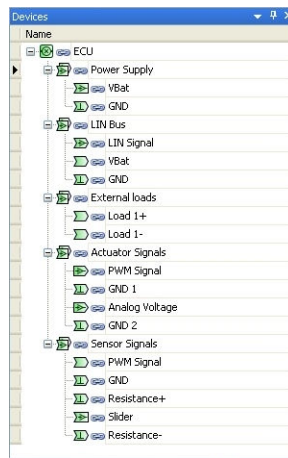
Channel type A category of channels on a SCALEXIO I/O unit or I/O board that provide exactly the same characteristics. For an overview of the channel types, refer to [Overview of SCALEXIO Channel Types \(SCALEXIO Hardware Installation and Configuration !\[\]\(dd161862f9164df98f62b726e9846241_img.jpg\)](#)).

CompactPCI® Serial CompactPCI® Serial is a standard for modular computer systems that uses (among others) the communication protocol PCI Express (PCIe) for data exchange. This standard is partially realized for SCALEXIO LabBox to support CompactPCI Serial boards.

ConfigurationDesk application An application that contains all the files for implementing a real-time application on a SCALEXIO system. It is managed with the Project and Application Manager in ConfigurationDesk.

Device topology A structure that describes the topology of the external devices, for example, ECU and external loads, which are connected to the SCALEXIO system in ConfigurationDesk. A device topology is structured in port groups and ports. It represents all the connectors and pins which connect the external devices to the channels of the SCALEXIO system. You can specify the electrical characteristics of the ports to simplify the assignment of the pins to the signals of the SCALEXIO system.

The device topology is displayed in the **External Device Browser**. You can drag the ports of the device topology to the graphical window of a working view and connect them to ports of function blocks.



Digital In 1 channel type The *Digital In 1* channel type is used to measure a digital voltage signal in the range 0 V ... +60 V with a threshold voltage of 0 V ... +23.8 V.

Digital In 2 channel type The *Digital In 2* channel type is used to measure a digital voltage signal in the range 0 V ... +60 V with a threshold voltage of +1 V ... +23.8 V.

Digital In 3 channel type The *Digital In 3* channel type is used to measure a digital voltage signal in the range 0 V ... +60 V with a threshold voltage of 0 V ... +24 V.

Digital In/Out 1 channel type The *Digital In/Out 1* channel type has two functions: measuring a digital voltage signal in the range 0 V ... +60 V with a threshold voltage of +1 V ... +23.8 V, or simulating a digital output stage of 0 V or in the range +5 V ... +60 V. In output mode, the channel is configurable as a low-side or high-side switch, or for push-pull mode. Each channel has its own pin for connecting an individual high reference voltage.

Digital In/Out 3 channel type The *Digital In/Out 3* channel type has two functions: measuring a digital voltage signal in the range 0 V ... +60 V with a threshold voltage of +1 V ... +23 V, or simulating a digital output stage of 0 V or

in the range +3.3 V ... +60 V. In output mode, the channel is configurable as a low-side or high-side switch, or for push-pull mode.

Digital In/Out 5 channel type The *Digital In/Out 5* channel type lets you measure a digital signal in the range 0 V ... +30 V with a threshold voltage of 0 V ... +12 V or simulate a digital output stage of 0 V, +3.3 V, or +5 V. In output mode, the channel is configurable as a low-side or high-side switch, or for push-pull mode.

Digital In/Out 9 channel type The *Digital In/Out 9* channel type lets you measure a digital signal in the range 0 V ... +60 V with a configurable threshold voltage of 0 V ... +12 V or generate a digital output signal of 0 V, +3.3 V, or +5 V. In output mode, the channel is configurable as a low-side or high-side switch, or for push-pull tri-state mode.

Digital input A collection of functions for reading digital bits and words with the SCALEXIO system.

Digital Out 1 channel type The *Digital Out 1* channel type is used to simulate a digital output stage of 0 V or in the range +5 V ... +60 V. The channel is configurable as a low-side or high-side switch, or for push-pull mode. All the channels of the Digital Out 1 channel type have a common pin for connecting the external reference voltage.

Digital Out 2 channel type The *Digital Out 2* channel type is used to simulate a digital output stage of 0 V or in the range +5 V ... +60 V. The channel is configurable as a low-side or high-side switch, or for push-pull mode. Each channel has its own pin to connect an individual high reference voltage.

Digital Out 3 channel type The *Digital Out 3* channel type is used to simulate a digital output stage of 0 V (low-side level) or in the range +5 V ... +60 V (high-side level). The channel is configurable as a low-side or high-side switch, or for push-pull mode. All the channels of the Digital Out 3 channel type have common pins for connecting two external high reference voltages.

Digital Out 8 channel type The *Digital Out 8* channel type lets you generate a digital output signal of 0 V, +3.3 V, or +5 V, for example, for PWM signal generation for inverter control. The channel can be configured as a low-side or high-side switch, or for push-pull tri-state mode.

Digital output A collection of functions for setting digital bits and words with the SCALEXIO system.

Dynamic host configuration protocol (DHCP) A protocol that is used by networked devices (clients) to obtain the parameters necessary for operation in an Internet Protocol (IP) network. A SCALEXIO system can be configured as a DHCP client to get its IP address from a DHCP server, for example.

Earth ground A ground potential that is the protective earth (PE) connected to parts of the enclosure of the SCALEXIO system via the mains.

Electrical error simulation To test ECU software under error conditions, electrical error simulation is used. Electrical error simulation deals with typical wiring errors like loose contacts, broken cables, short-circuits to neighboring pins, to ground (chassis) or to battery voltage. Supported by software, the electrical error simulation is performed by the failure simulation hardware of a SCALEXIO system.

Electrical Error Simulation port (EESPort) An *Electrical Error Simulation port* (EESPort) provides access to a failure simulation hardware for simulating electrical errors in an ECU wiring according to the ASAM AE XIL API standard. The configuration of the EESPort is described by a hardware-dependent *port configuration* and one or more *error configurations*.

Electrical interface unit A unit that provides the interface of the function block to the external devices and to the real-time hardware (via hardware resource assignment). Each electrical interface unit of a function block needs usually a different channel set to be assigned to. It also provides properties to configure the characteristics of the hardware.

Electronic fuse A fuse that protects a channel of a SCALEXIO board against overcurrent. Its trigger value can be set by software or is specified by hardware (it cannot be set by the user).

Ethernet Adapter 1 channel type The *Ethernet Adapter 1* channel type lets you connect one external device to the real-time application via Ethernet.

Ethernet Adapter 2 channel type The *Ethernet Adapter 2* channel type lets you connect one or more external devices to the real-time application via an integrated Ethernet switch.

Ethernet Switch 1 channel type The *Ethernet Switch 1* channel type lets you manage an integrated Ethernet switch.

Events The execution of tasks is triggered by events. The following event types are available:

- Timer events are periodic events with a sample rate and an optional offset.
- I/O events are asynchronous events triggered by I/O functions.
- Software events are parts of predefined tasks provided by the behavior model. They are available in ConfigurationDesk after model analysis.

External device A device that is connected to a SCALEXIO system externally, for example, an electronic control unit (ECU) to be tested, or external loads for hardware-in-the-loop simulation, or actuators and sensors for rapid control prototyping.

External load A load that is connected to a SCALEXIO system externally. The SCALEXIO system connects the external load to a signal measurement channel. The external load is connected to a specific load connector. Because it is outside

the enclosure of the SCALEXIO system, there are no restrictions for its physical dimensions. You can use a real load, for example, actuators, as an external load.

F

Failrail segment switch A component used to connect/disconnect a greater part of the SCALEXIO system (such as a slot unit, an I/O unit, or a rack) to the common failrails of the system. The selective disconnection of greater sections from the failrails reduces noise signals due to unavoidable, system inherent parasitic effects (e.g. parasitic capacitances of semiconductor switches) to a minimum.

Failure An electrical error that is simulated at one or more pins of the external device (i.e., ECU to be tested) during electrical error simulation. For example, you can simulate an open circuit.

Failure class A class that is an identifier for an electrical failure type. It describes the failure which can be simulated at a pin.

Failure insertion unit (FIU) A hardware component for simulating electrical errors in the wiring of an external device. In SCALEXIO systems, the failure insertion unit consists of a central FIU, failure routing units (FRU) on the SCALEXIO boards, and the failrails on the backplane which connect FRUs to the central FIU.

Failure routing unit (FRU) A component of the SCALEXIO failure simulation hardware. Each signal channel and each bus channel provides one or more failure routing units (FRUs) to connect the channel to the failrails.

Failure simulation Simulating electrical errors in the wiring of the external device, for example, an ECU. You can use electrical error simulation to test the reaction to faulty signals. You can simulate broken wires, or short circuits to different potentials (battery voltage, ground, or another signal).

Failure Simulation Module An optional software module of ControlDesk. You can use it to set and control all failure types in a graphical environment.

Flexible In 1 channel type The *Flexible In 1* channel type is used to measure analog and digital signals which are generated by an ECU. If required, analog voltage and digital current measurement or digital voltage and analog current measurement can be performed at the same time.

Flexible In 2 channel type The *Flexible In 2* channel type is used for digital voltage measurement, digital current measurement, or analog current measurement.

Flexible In 3 channel type The *Flexible In 3* channel type is used for analog and digital voltage measurement.

Flexible In/Out 1 channel Type The *Flexible In/Out 1* channel type supports digital signal measurement/generation and analog signal measurement for position sensors.

Flexible Out 1 channel type The *Flexible Out 1* channel type is used to generate analog and digital input signals for an ECU.

FlexRay 1 channel type The *FlexRay 1* channel type allows communication via a FlexRay bus type.

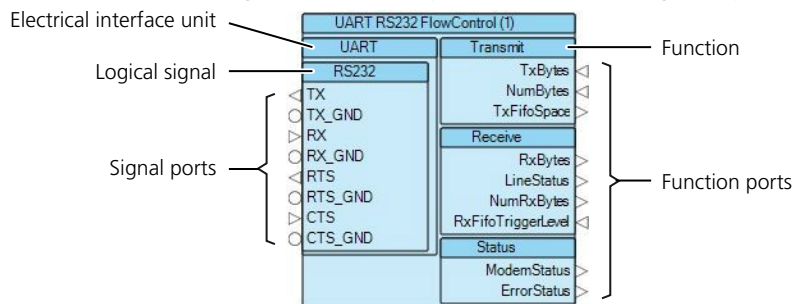
FlexRay 2 channel type The *FlexRay 2* channel type allows communication via a FlexRay bus type.

FPGA application An application that is implemented for a field programmable gate array (FPGA). Using FPGA applications you can realize very fast closed feedback loops with low turnaround times because the simulation model is executed directly on the FPGA.

Function block A block in the graphical window of ConfigurationDesk that represents a function which is part of the I/O model. The functions are executed in the local processor of the I/O boards or units of a SCALEXIO system. There are different kinds of functions:

- Functions for generating signals
- Functions for measuring signals coming from the external devices
- Functions for connecting the external devices to simulated buses
- Functions for controlling the simulated battery voltage and power switches

A function block has signal and function ports, see the following example.



Function port A port of the function block that provides the interface to the behavior model.

Gigalink Gigalink is a dSPACE-specific communication bus that is used in PHS-bus-based real-time systems from dSPACE.

H

Hardware topology A structure that represents the real-time hardware of the connected SCALEXIO system in ConfigurationDesk. It contains the processing unit or processor board and the I/O boards and I/O units.

Hardware-in-the-loop (HIL) simulation A method of testing a controller with a model of the controlled system which is simulated in real time. When you have produced a new controller, you usually want to test it. For the final tests you usually connect the real controller to a model of the controlled system which, of course, must be simulated in real time. This way you can ensure that the controller does not contain any errors that could damage the real plant. This technique is called hardware-in-the-loop simulation (HIL).

HighFlex I/O board A versatile and finely scalable board to use its channels for different I/O functions. You can select and configure the I/O functions for the channels in ConfigurationDesk.

Host PC The host PC is a standard PC that the dSPACE test and experiment software is installed on. Via the host PC, you can configure the SCALEXIO hardware, download a real-time application to the system and control the simulation.

I

I/O connector A connector that is used to connect input and output signals to a dSPACE system.

I/O model A model that contains all the I/O functions of the simulation model. It is modeled in ConfigurationDesk. An I/O model and a behavior model form a real-time model.

Internal load A load that is mounted on a signal measurement board. It is connected to the ECU outputs directly on the board. Because the available space is restricted, its physical dimensions and power dissipation are limited.


IOCNET IOCNET (I/O carrier network) is a dSPACE-specific high-speed serial communication bus that connects all the real-time hardware in a SCALEXIO system. IOCNET can also be used to build a multiprocessor system that consists of multiple SCALEXIO processor hardware components.

L

LIN 1 channel type The *LIN 1* channel type allows communication via a LIN or K-Line bus type.

LIN 2 channel type The *LIN 2* channel type allows communication via a LIN bus.


Load 1 channel type The *Load 1* channel type has no I/O signal conditioning circuit and only acts as a carrier for small elements like resistors, capacitors or coils which can be plugged onto the DS2680-IL Load Board.

Local processor A processor that is part of I/O hardware. It computes the functions which are assigned to the channels via ConfigurationDesk. On signal measurement channels it controls the measurement of input signals. On signal generation channels it controls the generation of output signals. On bus boards it computes the reading and writing of bus signals. A local processor communicates with the main processor via the [IOCNET](#) .

Logical signal A signal that combines all the signal ports which belong together to provide the functionality of the signal.

Logical signal chain A representation of the entire chain of signals from the external device to the model function in the Simulink model in ConfigurationDesk. It includes the elements: external device ↔ function block ↔ model port ↔ model function.

M

Main processor A processor that is part of the processing unit or the processor board. It computes the simulation model. It communicates with the host PC via Ethernet and the other boards of the SCALEXIO system via the [IOCNET](#) .

MAP file A file that maps symbolic names to physical addresses. It is generated by ConfigurationDesk during the build process. It must be located in the same folder as the TRC file.

Melting fuse A fuse that protects a wire connected to a channel of a SCALEXIO against overcurrent and therefore also a cable fire. The rated current of a melting fuse is not configurable. A melting fuse is not replaceable by the user. If it has blown, it must be replaced by dSPACE.

Message Viewer A window that shows messages generated during work with the dSPACE software to give information about errors and performed tasks. Located in the tool window.

Model topology A structure that represents the topology of a Simulink model in ConfigurationDesk. It is displayed in the model browser of

ConfigurationDesk. The ports of the model topology can be connected to the ports of function blocks to write the parameters to out functions or to read the signal values from in functions.

MultiCompact I/O unit A unit having a large number of I/O channels and a compact design. Each channel has a specific channel type which you can configure in ConfigurationDesk. The units are mounted in the SCALEXIO rack. Each MultiCompact I/O unit has an IOCNET router to connect it to the IOCNET.

Multi-pin error A feature of the SCALEXIO concept for electrical error simulation that lets you simulate a short circuit between three or more signal channels and/or bus channels. The channels can be located on the same or different boards or I/O units. You can simulate a short circuit between:

- Channels of the same signal category (e.g., four signal generation channels)
- Channels of different signal categories (e.g., three signal generation channels and two signal measurement channels)
- Signal channels and bus channels (e.g., two signal generation channels, one signal measurement channel, and one bus channel)

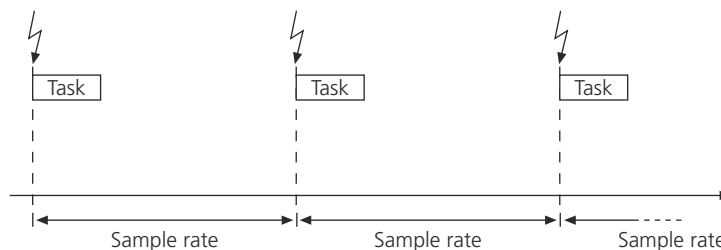
Multiple electrical errors A feature of the SCALEXIO concept for electrical error simulation that lets you switch electrical errors at the same time or in succession. For example, you can simulate an open circuit for one channel and a short circuit for another channel at the same time, without deactivating the first error.

N

Noise generator An electronic circuit which generates a signal with a random variation of voltage or current (noise). The noise can be added to analog signals.

P

Periodic tasks Periodic tasks are triggered by timer events according to a specific sample rate, e.g., 20 ms.



Periodic tasks are generally generated by creating a new task and creating a timer event for the new task.

PHS-bus-based system A modular dSPACE system consisting of a processor board such as a DS100x Processor Board and I/O boards.

Physical signal chain The physical connection of external devices (for example, ECU and loads in HIL simulations or sensors and actors in RCP simulations) to the SCALEXIO system. It includes the external cable harness and the pinouts of the connectors.

Platform An environment where a simulation model is computed either in non-real-time (Simulink or VEOS) or in real time (dSPACE real-time hardware). In a SCALEXIO system the platform is the SCALEXIO Processing Unit or DS6001 Processor Board.

Position sensor simulation A collection of functions for generating position sensor signals, for example, the signal of a digital incremental encoder.

Power Control 1 channel type The *Power Control 1* channel type is used to access the battery simulation controller which sets the battery simulation power supply unit.

Power control bus The power control bus lets you cascade SCALEXIO AutoBoxes/LabBoxes to power them up or down at the same time by pressing only one of their On/Off buttons. Further, power-up might be performed for all cascaded SCALEXIO AutoBoxes/LabBoxes after a dedicated wake-up message or frame is received at an installed bus board.

Power Switch 1 channel type The *Power Switch 1* channel type is used to switch the battery simulation supply voltage to connected external devices (e.g., the ECU) and to measure the current provided to the external devices. In contrast to the Power Switch 2 channel type, switching is performed by semiconductor switches.

Power Switch 2 channel type The *Power Switch 2* channel type is used to switch the battery simulation supply voltage to connected external devices (e.g., the ECU). In contrast to the Power Switch 1 channel type, there is no current measurement on the channel and switching is performed by relays. That is why current enhancement (channel multiplication) is not possible.

Pulse pattern generation A collection of functions for generating pulse patterns, for example, for pulse width modulation (PWM) or pulse frequency modulation (PFM) signals.

Pulse pattern measurement A collection of functions for measuring pulse width modulation or pulse frequency modulation signals. The functions provide the frequency and duty cycle of the signals or capture an event.

Python "Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid application development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.

Python supports modules and packages, which encourages program modularity and code reuse." (*"Python Reference Manual"*)

For more details on Python, refer to <http://www.python.org>.

R

Real load The load that is actually used in the scenario being simulated. You can use a real load if it is difficult to reproduce its electrical characteristics. Real loads can be connected to the SCALEXIO system externally.

Real-time application An application running in real time on a dSPACE platform. A real-time application can be built, for example, from a Simulink model containing RTI blocks. A real-time application for a SCALEXIO system is built from a real-time model in ConfigurationDesk.

Real-time application file A file that contains the executable object file for processor boards. The extension of the file is RTA, so it is also called the RTA file. After the build process, the RTA file can be downloaded to the SCALEXIO system.

Real-time model A model that is calculated in real-time. For SCALEXIO systems, it consists of a behavior model and an I/O model. The source code of the real-time application is automatically generated from both model parts.

Real-Time Test Manager A graphical user interface that you can use to manage [RTT sequences](#) on a dSPACE platform.

Real-time testing A special execution environment that makes it possible to execute tests synchronously with the real-time model. A Python interpreter on a dSPACE platform executes the tests ([RTT sequence](#)) which have access to model variables in every simulation step.

Resistance Out 1 channel type The *Resistance Out 1* channel type is used for resistance simulation.

Resistance Out 2 channel type The *Resistance Out 2* channel type is used for resistance simulation.

Resistor simulation A collection of functions for simulating ohmic resistance with the real-time system.

Resolver In 2 channel type The *Resolver In 2* channel type is used for single-coil excited resolvers.

Restbus simulation Simulation of all the received bus signals of a communication node. The SCALEXIO system simulates the communication to all the other communication nodes of the bus.

RTT sequence Scripts which are executed in real-time on a dSPACE platform for test purposes. The scripts are written in Python.

Runnable functions Runnable functions are functions that are called by a task to compute results. A runnable function can be executed in a periodic or asynchronous task.

S

SCALEXIO processing hardware Each SCALEXIO system must have at least one processing hardware component that executes the real-time application and provides an interface to the IOCNET for communication with I/O units, I/O boards, or other processing hardware. The processing hardware communicates with the host PC via Ethernet. In a SCALEXIO system, two types of processing hardware can be used, a DS6001 Processor Board or a SCALEXIO Processing Unit. For system configuration and support, each SCALEXIO processing hardware component has a web interface you can open in any Internet browser.

SCALEXIO Processing Unit A SCALEXIO Processing Unit is based on a SCALEXIO Real-Time PC, a real-time operating system, and a DS2502 IOCNET Link Board for communication with other SCALEXIO components. The SCALEXIO Processing Unit comes in a 19" unit as a rack-mount version with front brackets to be inserted into a SCALEXIO rack or as a desktop version with feet to be used on a desktop.

SCALEXIO processor board A board that computes the real-time application. It has an operating system that controls all calculations. The board has an integrated interface to the [IOCNET](#) for communication to the I/O units and I/O boards.

SCALEXIO rack A SCALEXIO rack contains all the hardware components of a SCALEXIO system or parts of it. Normally, a SCALEXIO rack contains a SCALEXIO Processing Unit as SCALEXIO processing hardware. In addition, the rack provides sufficient space for MultiCompact units, HighFlex I/O boards, and SCALEXIO I/O boards. To insert I/O boards into the rack, slot units in two sizes are available (6 slots and 20 slots). To simulate a car battery, a SCALEXIO rack can have a battery simulation power supply unit that can be controlled by the SCALEXIO system.

SCALEXIO Real-Time PC The SCALEXIO Real-Time PC provides the calculation power to execute the real-time application. The SCALEXIO Real-Time PC consists of a high-standard industry ATX motherboard and a multicore main processor that is qualified by dSPACE for use with SCALEXIO. Its operating system is a Linux-based real-time operating system. One of the available processor cores is reserved for service jobs, the others are used for computing real-time models.

SCALEXIO system A system that is used in real-time simulation for developing and testing electronic control units (ECUs).

SENT SENT (Single Edge Nibble Transmission) is a protocol used between sensors and ECUs. It is defined in the SAE J2716 standard defined by the Society of Automotive Engineers (SAE). It is used to transmit data of high-resolution (10

bits or more) sensors as an alternative to an analog interface. The sensor signal is transmitted as a series of pulses with data measured as falling to falling edge times.

Signal generation board A board for a SCALEXIO system that generates signals which are available as outputs of the SCALEXIO system and are connected to an external device, for example, an ECU. The signal to be simulated is specified in ConfigurationDesk by assigning a function block to one or more channels of the signal generation board.

Signal ground The reference potential of a SCALEXIO system for connecting external devices. It is used as the ECU ground potential (KL 31). If the SCALEXIO system has a battery simulation power supply unit, signal ground is connected to the negative output terminal of the battery simulation power supply unit.

Signal measurement board A board for a SCALEXIO system that measures signals which are generated by an external device and are connected to inputs of the SCALEXIO system. The values to be measured are specified in ConfigurationDesk by assigning a function block to one or more channels of the signal measurement board. Loads can be connected to a channel (see [External load](#) and [Internal load](#)).

Signal port A port of the function block that provides the interface to external devices via device blocks. They represent the electrical connection points of a function block.

Slot unit A unit used to install HighFlex I/O boards in a SCALEXIO rack. Slot units are mounted in the SCALEXIO rack. Each slot unit has an IOCNET router to connect it to the IOCNET.

Substitute load A load connected to an output of an external device in a SCALEXIO system. It replaces the real load during simulation. A substitute load can be installed on a signal measurement board inside the SCALEXIO system if it is small enough (see [Internal load](#)).

System description file A file that specifies the real-time application and its parameters and measurement variables. The variable description is stored as a system description (SDF) file. System description files are created automatically by ConfigurationDesk during the build process.

System ground A ground potential of the SCALEXIO system. It is the negative output terminal of the internal power supply (+24 V) of the SCALEXIO system.

T

Tasks Tasks are pieces of code whose execution is controlled by a real-time operating system (RTOS). Only one task can run at a time. However, multiple tasks can run concurrently, taking turns to use the resources of the processing unit or processor board. Each task is assigned a priority according to its relative

importance. The RTOS suspends a low-priority task so that a high-priority task is given a turn (preemptive multitasking). When the high-priority task has been executed, the suspended low-priority task resumes execution. In ConfigurationDesk, a task executes one or more runnable functions.

Timing sequence A specification of the time duration between state changes during electrical error simulation. Timing sequences are necessary if you want to use pulsed switching, for example, to simulate loose contacts or switch bouncing.

Trigger In 1 channel type The *Trigger In 1* channel type is a specific channel that is used as a trigger input with a threshold voltage in the range of 0 V ... +24 V.

Trigger In 2 channel type The *Trigger In 2* channel type is a specific channel that is used as a trigger input with a threshold voltage in the range of 0 V ... +24 V.

Tunable parameter A parameter whose values can be changed by the experiment software when the real-time application is running. There are two types of tunable parameters:

- *Stop-run-tunable parameters* change their values when the state of the real-time applications changes from stop to run.
- *Run-time-tunable parameters* change their values when the background task of the real-time application is executed.

U

UART 1 channel type The *UART 1* channel type provides a multiprotocol transceiver for RS232, RS422, and RS485 and a transceiver for K-Line to allow communication with one of these interface or bus types.

UART5 channel type The *UART5* channel type provides a serial interface (UART) to communicate with standard RS232 devices with 2 ... 115,200 bit/s and a clock frequency of 1,843,200.0 Hz.

UART6 channel type The *UART 6* channel type provides a serial interface (UART) to communicate with a standard RS232 device with 14 ... 459,559 bit/s and a clock frequency of 14,705,882.35294 Hz.

V

Variable description file An ASCII file that is generated for dSPACE real-time hardware. The extension of the file is TRC, so it is also called the TRC file. The TRC file provides information on the variables of a real-time application or a

Simulink model that is required for connecting variables to instruments in a layout of the experiment software, accessing variables in the test software, etc. It can be generated automatically by ConfigurationDesk or written manually.

W

Wavetable generation A collection of functions for generating an analog or digital signal from specified wavetables.

Appendix

Abbreviations

List of abbreviations

The following abbreviations are used in this document.

Abbreviation	Description
ACC	Adaptive cruise control
ADC	Analog/digital converter
API	Application programming interface
ASM	Automotive simulation model
ASR	Anti-slip regulation (traction control system)
BEV	Battery electric vehicle
BLDC	Brushless DC machine
BoB	Breakout box
CAN	Controller area network
CARB	California Air Resources Board
CRC	Cyclic redundancy check
CSMA/CR	Carrier sense multiple access with collision resolution
DAC	Digital/analog converter
DSP	Digital signal processor
DHCP	Dynamic host configuration protocol
ECU	Electronic control unit
EML	Error management logic
ESP	Electronic stability program (electronic stability control)
FIU	Failure Insertion Unit
FMI	Functional mock-up interface
FRU	Failure routing unit
GUI	Graphical user interface
HIL	Hardware-in-the-loop

Abbreviation	Description
I/O	Input/output
ISR	Interrupt service routine
LIN	Local interconnect network
MP	Multiprocessor
OBD	On-board diagnostics
PFM	Pulse frequency modulation
PMSM	Permanent magnet synchronous machine
PWM	Pulse width modulation
RCP	Rapid control prototyping
RTA file	Real-time application file
RTD	Resistance temperature detector
RTI	Real-time interface
RTP	Real-time processor
RTR	Remote transmission request
RTT	Real-time testing
SCIM	Squirrel cage asynchronous machine
SDF file	System description file
SENT	Single edge nibble transmission
TRC file	Variable description file
UART	Universal asynchronous receiver transmitter
VTG	Variable turbine geometry

A

abbreviations 163
 ASM 31
 ASMs
 basics 95
 modeling 97
 AutomationDesk 34

B

battery simulation 85
 battery voltage
 controlling 87
 behavior model 45
 Bus Navigator 109

C

CAN
 workflow for implementing a CAN bus connection 71
 CAN bus 71
 restbus simulation 72
 car battery 85
 central FIU 130
 Common Program Data folder 8
 communication
 hardware 70
 communication network 41
 CompactPCI Serial board 22
 compatibility check 104
 configuration
 workflow 52
 configuration process
 overview 52
 ConfigurationDesk 27
 ConfigurationDesk application 56
 connecting loads 64
 ControlDesk 33
 working 108
 controller area network 71
 controlling
 battery voltage 87
 current consumption
 measuring 88

D

Documents folder 8
 DS2502 IOCNET Link Board 17
 DS2601 Signal Measurement Board 21
 DS2621 Signal Generation Board 21
 DS2642 FIU & Power Switch Board 21
 DS2671 Bus Board 21
 DS2672 Bus Module 22
 DS2680 I/O Unit 22
 DS2680-IL Load Board 22
 DS2690 Digital I/O Board 22
 DS6001 Processor Board 17

E

ECU interface 47
 electrical connection 37
 electrical error simulation
 hardware 128
 multi-pin errors 128
 workflow 135
 electrical error simulation basics
 for SCALEXIO systems 124
 Electrical error simulation concept
 SCALEXIO system 128
 electrical errors 125
 environmental conditions 37
 executing
 electrical error simulation 141
 experimenting 107
 workflow 107
 external device 47
 external load 47, 65

F

failrail segment switch 131
 failrails 131
 failure class
 specifying 137
 failure routing unit 131
 failure simulation
 pulsed switching 126
 FIU concept
 SCALEXIO system 128
 FlexRay
 workflow for implementing a FlexRay bus connection 74
 FlexRay bus 74
 restbus simulation 74
 FlexRay Configuration Blockset 30
 function block 48
 function block types
 for generating signals 67
 for measuring signals 63

H

hardware-in-the-loop simulation 10
 high rails
 switching 88
 HighFlex I/O boards 16

I

I/O model 45
 installation 37
 internal load 64
 IOCNET
 system communication 41

L

LIN
 workflow for implementing a LIN bus connection 77

LIN bus 77
 restbus simulation 78
 load rejection 65
 loads
 connecting 64
 local interconnect network 77
 Local Program Data folder 8
 logical signal chain 47

M

MATLAB/Simulink 28
 measuring current consumption 88
 model port block 45, 50
 Model Port Block Library 28
 Model Separation Block Library 29
 ModelDesk 31
 modeling
 with ASMs 97
 MotionDesk 33
 MultiCompact I/O hardware 16
 MultiCompact I/O units 16
 multi-pin errors 128

O

overview
 configuration process 52
 overvoltage
 simulating 87

P

physical signal chain 46
 projects in ConfigurationDesk 55
 pulsed switching 126

R

rapid control prototyping 11
 real-time clock
 SCALEXIO 106
 Real-Time Testing 35
 restbus simulation
 CAN bus 72
 FlexRay bus 74
 LIN bus 78
 road models 99
 RTI CAN MultiMessage Blockset 29
 RTI FPGA Programming Blockset 31
 RTI LIN MultiMessage Blockset 29
 RTT sequences 118
 implementing 119
 managing 119

S

safety precautions
 transporting 38
 SCALEXIO AutoBox 14
 SCALEXIO hardware
 overview 13
 SCALEXIO I/O boards 16

- SCALEXIO LabBox 14
- SCALEXIO processing hardware 17
- SCALEXIO Processing Unit 17
- SCALEXIO rack 15
 - transporting 38
- SCALEXIO Real-Time PC 17
- SCALEXIO system
 - communication network 41
 - transporting 38
- SCALEXIO system time 106
- SCALEXIO_RTLib license 106
- scenarios 100
- signal chain
 - physical and logical 46
- Signal Editor 109
- signal generation
 - function block types 67
 - hardware 66
- signal measurement
 - function block types 63
 - hardware 62
- simulating
 - overvoltage 87
 - undervoltage 87
 - voltage fluctuation 87
- Simulink blocksets 28
- software tool
 - ASM 31
 - AutomationDesk 34
 - ConfigurationDesk 27
 - ControlDesk 33
 - MATLAB/Simulink 28
 - ModelDesk 31
 - MotionDesk 33
 - overview 24
 - Real-Time Testing 35
 - Simulink blocksets 28
- system clock
 - SCALEXIO 106
- system time
 - SCALEXIO 106

- configuration 52
- electrical error simulation 135
- experimenting 107
- implementing CAN communication 72
- implementing FlexRay communication 75
- implementing LIN communication 78
- serial interface 80

T

- tool chain 24
- transportation 37
- transporting
 - SCALEXIO rack 38
- troubleshooting 143

U

- undervoltage
 - simulating 87

V

- voltage fluctuation
 - simulating 87

W

- workflow