ConfigurationDesk

Demo Projects

For ConfigurationDesk 6.7

Release 2021-A – May 2021

How to Contact dSPACE

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

How to Contact dSPACE Support

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

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

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

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

Software Updates and Patches

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

Important Notice

Contents

About This Document

Contents

The dSPACE software provides multiple demo projects for ConfigurationDesk. This document provides descriptions of the demo projects or refers you to detailed descriptions in other documents.

Symbols

dSPACE user documentation uses the following symbols:

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

Naming conventions

dSPACE user documentation uses the following naming conventions:

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

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

Special folders

Some software products use the following special folders:

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

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

or

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

Documents folder A standard folder for user-specific documents.

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

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

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

Accessing dSPACE Help and PDF Files

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

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

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

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

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

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

Overview and Access

Where to go from here

Information in this section

Overview of ConfigurationDesk Demo Projects

Available demo projects

The following demo projects and applications are available for you to open after you first started ConfigurationDesk.

> **Tip**
>
> The hardware topologies in most demo projects represent only one of the dSPACE hardware systems that provide the required channel types.

Tutorial demo projects The tutorial demo projects are based on different ConfigurationDesk tutorials that guide you through the basic steps of different use scenarios. You should only use them while working through the related tutorial.

Project and Application	Description	Function Blocks	Specific Hardware Used	Refer to...
BusManagerTutorial: *Several tutorial applications*	Introduces you to the basic steps of working with the Bus Manager. This includes: ▪ Configuring CAN and LIN communication for	▪ Bus Configuration ▪ CAN ▪ LIN	*SCALEXIO:* ▪ DS2672 Bus Module	Bus Manager Tutorial 🕮

Project and Application	Description	Function Blocks	Specific Hardware Used	Refer to...
	simulation, inspection, and manipulation purposes. ▪ Working with or without behavior models. ▪ Building real-time applications and generating bus simulation containers. ▪ Experimenting with ControlDesk.			
CfgMABXIIITutorial: *Several tutorial applications*	This tutorial shows you the basic configuration steps if you want to use MicroAutoBox III hardware in ConfigurationDesk.	▪ Voltage In ▪ Voltage Out	*MicroAutoBox III*	ConfigurationDesk Tutorial MicroAutoBox III 🕮
CfgStartingWithExternal Devices: *Several tutorial applications*	The contents of the `CfgStartingWithExternalDevices` demo project demonstrate the typical workflow for implementing real-time applications when you start out with an ECU. The project starts with creating an external device interface and ends with building a real-time application.	▪ Multi Bit In ▪ Multi Bit Out ▪ PWM/PFM In ▪ Power Switch	*SCALEXIO:* ▪ DS2680 I/O Unit	ConfigurationDesk Tutorial Starting with External Devices 🕮
CfgStartingWithSimulink Tutorial: *Several tutorial applications*	This tutorial shows you the basic configuration steps in ConfigurationDesk when you start out with a Simulink behavior model.	▪ Voltage In ▪ Voltage Out ▪ Multi Bit In ▪ Multi Bit Out	*SCALEXIO LabBox:* ▪ DS6201 Digital I/O Board ▪ DS6101 Multi I/O Board	ConfigurationDesk Tutorial Starting with Simulink 🕮

Example demo projects Example demo projects include an implementation of a ConfigurationDesk application using specific I/O functionality or hardware resources. Some example demo projects can be used as a basis for your own implementation.

Project and Application	Description	Function Blocks	Specific Hardware Used	Refer to...
CfgBasicIODemo: DemoDigitalFunctions	The **DemoDigitalFunctions** application serves to demonstrate a simple signal chain with digital I/O functionality.	▪ Multi Bit In ▪ Multi Bit Out ▪ PWM/PFM Out ▪ PWM/PFM In ▪ Digital Pulse Capture	*SCALEXIO:* ▪ DS2680 I/O Unit ▪ DS2621 Signal Generation Board ▪ DS2601 Signal Measurement Board	Using the DemoDigitalFunctions Application on page 25
CfgBasicIODemo: DemoMixedBasicIO	The **DemoMixedBasicIO** application serves to demonstrate a simple signal	▪ Voltage In ▪ Current In ▪ Triggered Current In	*SCALEXIO:* ▪ DS2680 I/O Unit	Using the DemoMixedBasicIO Application on page 29

Project and Application	Description	Function Blocks	Specific Hardware Used	Refer to...
	chain with basic I/O functionality.	▪ Voltage Out ▪ Current Sink ▪ Multi Bit In ▪ Multi Bit Out ▪ Resistance Out ▪ Potentiometer Out ▪ PWM/PFM Out		
CfgBasicIODemo: DemoSpringMassDamper	The DemoSpringMassDamper application illustrates the simulation of a damped spring-mass system.	▪ Voltage In ▪ Voltage Out	*MicroAutoBox III:* ▪ DS1513 Multi-I/O Board	Using the DemoSpringMassDamper Application on page 32
CfgCANMMDemo: CANMMAppl	The CANMMAppl application demonstrates simple CAN communication between two CAN controllers using the RTI CAN MultiMessage Blockset.	▪ CAN	*SCALEXIO:* ▪ DS2672 Bus Module	Using the CANMMAppl Application on page 16
CfgEthernetDemo: EthernetAppl	Two examples of Ethernet custom function blocks (Ethernet Send and Ethernet Receive) are included in the Ethernet demo in your dSPACE installation. They give you basic information on exchanging data with a second Ethernet port in your SCALEXIO or MicroAutoBox III system.	▪ Ethernet Setup *Custom function blocks:* ▪ Ethernet Receive ▪ Ethernet Send	*SCALEXIO:* ▪ SCALEXIO Processing Unit – Ethernet Adapter	Example: Implementing a Custom Function Block for Ethernet Communication (ConfigurationDesk Custom I/O Function Implementation Guide 📖)
CfgFlexRayConfigDemo: FlexRayConfigAppl	The FlexRayConfigAppl application serves to demonstrate simple FlexRay communication.	▪ FlexRay	*SCALEXIO:* ▪ DS2672 Bus Module	For the current dSPACE Release there is no specific documentation of the demo project. For more information on implementing FlexRay communication in ConfigurationDesk, refer to Basics on Implementing FlexRay Communication (ConfigurationDesk Real-Time Implementation Guide 📖).
CfgFPGAuartDemo: FPGAuartDemo	The demo project `CfgFPGAuartDemo` is an example of an FPGA application for implementing a configurable UART bus communication. It is not	*Custom function blocks:* ▪ DemoFPGAuart ▪ DemoFPGAuart _RS232_UART_1 – UART_4	*SCALEXIO:* ▪ DS2655 FPGA Base Board	Building the Signal Chain for UART Communication Using an FPGA Board (ConfigurationDesk

Project and Application	Description	Function Blocks	Specific Hardware Used	Refer to...
	necessary to have knowledge about FPGA programming to use the example or reuse the example in a ConfigurationDesk project.	▪ DemoFPGAuart_RS485_UART_5 – UART_8 ▪ DemoFPGAuart_Setup		UART Implementation 📖)
CfgLINMMDemo: LINMMAppl	The **LINMMAppl** application serves to demonstrate simple LIN communication between two LIN controllers using the RTI LIN MultiMessage Blockset.	▪ LIN	*SCALEXIO:* ▪ DS2672 Bus Module	Using the LINMMAppl Application on page 21
CfgUARTDemo: UARTAppl	The **UARTAppl** application is a simple example of serial communication, prepared for your first experience with custom function blocks. It shows ▪ Basics of custom function blocks ▪ Initializing a UART driver ▪ Using configuration properties ▪ Start/stop functions ▪ Sending and receiving data of a fixed length SCALEXIO processing hardware components provide a UART channel that you can use to implement UART serial communication. You can modify the **UARTAppl** application in the **CfgUARTDemo** project for simple UART serial communication to work with the UART channels provided by the processing hardware.	*Custom function block:* ▪ UART	*SCALEXIO:* ▪ DS2672 Bus Module or ▪ SCALEXIO Processing Unit or ▪ DS6001 Processor Board	*DS2672 Bus Module:* Example of Simple UART Serial Communication Using a DS2672 Bus Module (ConfigurationDesk UART Implementation 📖) *Processing hardware:* Example of UART Serial Communication Using Onboard UART of SCALEXIO Processing Hardware (ConfigurationDesk UART Implementation 📖)

CfgUARTDemo: UARTRS232_MABXIIIAppl	The **UARTRS232_MABXIIIAppl** application contains a custom function block that is implemented for a DS1511 or DS1513 Multi-I/O Board of a MicroAutoBox III.	*Custom function block:*
		▪ DS151x UART RS232 Demo
			▪ *MicroAutoBox III:*
			  ▪ DS1511 Multi-I/O Board
			  or
			  ▪ DS1513 Multi-I/O Board
				Example of UART Serial Communication Using a DS1511 or DS1513 Multi-I/O Board (MicroAutoBox III) (ConfigurationDesk UART Implementation 📖)

CfgUARTRS232FlowControlDemo: UARTRS232FlowControlAppl	The **UARTRS232FlowControlAppl** application is a complex example of serial communication, prepared for your advanced experience with	*Custom function block:* ▪ UART RS232 FlowControl	*SCALEXIO:* ▪ DS2671 Bus Board	Example of Complex UART Serial Communication (RS232 FlowControl) Using a DS2671 Bus Board

Project and Application	Description	Function Blocks	Specific Hardware Used	Refer to...
	custom function blocks. It shows ▪ Sending and receiving data of arbitrary length ▪ Usage of several hardware resources ▪ Generating events			(ConfigurationDesk UART Implementation 📖)
CfgUARTRS232FlowControlDemo: DS6321_UARTAppl	The **DS6321_UARTAppl** application contains a custom function block that is implemented for a DS6321 UART Board of a SCALEXIO LabBox. This board offers different transceiver types which can be used for serial communication (K-Line, RS232, RS422, RS485). The demo application is configured for the K-Line transceiver.	*Custom function block:* ▪ DS6321 UART Demo	*SCALEXIO LabBox:* ▪ DS6321 UART Board	Example of UART Serial Communication Using a DS6321 UART Board (SCALEXIO LabBox) (ConfigurationDesk UART Implementation 📖)
CfgUARTRS232FlowControlDemo: DS1521_MABXIII_UARTAppl	The **DS1521_MABXIII_UARTAppl** application contains a custom function block that is implemented for a DS1521 Bus Board of a MicroAutoBox III. This board offers different transceiver types which can be used for serial communication (RS232, RS422, RS485). The demo application is configured for the RS232 transceiver.	*Custom function block:* ▪ DS1521 UART Demo	*MicroAutoBox III:* ▪ DS1521 Bus Board	Example of UART Serial Communication Using a DS1521 Bus Board (MicroAutoBox III) (ConfigurationDesk UART Implementation 📖)
CfgUserStorageDemo: UserStorageApplication	The **UserStorageApplicaton** application provides custom function block types that let you use a SCALEXIO SSD.	*Custom function blocks:* ▪ User Storage Read ▪ User Storage Write	*SCALEXIO:* ▪ SCALEXIO Processing Unit with SCALEXIO SSD	Using the UserStorageApplication Application on page 37
EDrivesControlDemo: PMSM_Demo	The **PMSM_Demo** application shows an example of using the DS6121 Multi-I/O Board for controlling a permanent magnet synchronous motor (PMSM) with field-oriented control. In this demo, sinusoidal phase currents are generated to control the motor. This is called sine commutation. Field-oriented control means, that the coordinate system of the control loop is rotated with the	▪ Voltage In ▪ Multi-Channel PWM Out ▪ Digital Incremental Encoder In ▪ Hall Encoder In	*SCALEXIO LabBox:* ▪ DS6121 Multi-I/O Board	Introduction to the EDrivesControlDemo Project on page 41

Project and Application	Description	Function Blocks	Specific Hardware Used	Refer to...
	rotor position. This converts the sinusoidal setpoints for the currents into constant setpoints that are easier to control.			
EDrivesControlDemo: BLDC_Demo	The **BLDC_Demo** application is an example of using the DS6121 Multi-I/O Board for controlling a brushless DC motor in block commutation mode.	▪ Voltage In ▪ Block-Commutated PWM Out ▪ Digital Incremental Encoder In ▪ Hall Encoder In	*SCALEXIO LabBox:* ▪ DS6121 Multi-I/O Board	Introduction to the EDrivesControlDemo Project on page 41
EngineConfiguration: EngineExample	The **EngineExample** application servers to demonstrate the simulation of a 6-cylinder four-stroke piston engine.	▪ Voltage Out ▪ Resistance Out ▪ PWM/PFM In ▪ Angular Clock Setup ▪ Engine Simulation Setup ▪ Injection/Ignition Voltage In ▪ Injection/Ignition Current In ▪ Crank/Cam Voltage Out ▪ Crank/Cam Digital Out ▪ Knock Signal Out ▪ Lambda NCCR	*SCALEXIO:* ▪ SCALEXIO Processing Unit – Angle Unit Set ▪ DS2680 I/O Unit	Using the EngineExample Application on page 51
EngineControlDemo: EngineControl	The **EngineControl** application serves to demonstrate the control of a 6-cylinder four-stroke piston engine.	▪ Engine Control Setup ▪ Crank In ▪ Cam In ▪ Injection Out ▪ Ignition Out ▪ Knock In	*MicroAutoBox III:* ▪ DS1554 Engine Control I/O Module	Using the EngineControl Application on page 55
FunctionalSafetyDemo: FuSaDemoApp	The **FuSaDemoApp** application serves to demonstrate I/O functionality for functional safety in combination with the MicroAutoBox III.	▪ FuSa Setup ▪ FuSa Response Trigger ▪ FuSa Challenge-Response Monitoring	*MicroAutoBox III:* ▪ DS1403 Processor Board (FuSa Unit)	Using the FuSaDemoApp Application on page 65
MPTurnlampDemo: turnlamp	The **turnlamp** application is a multimodel, multi-processing-unit application that demonstrates the simulation of a turn-signal circuit that can be activated with a turn-signal	–	*SCALEXIO:* ▪ Two SCALEXIO Processing Units	Using the turnlamp Application on page 61

Project and Application	Description	Function Blocks	Specific Hardware Used	Refer to...
	lever and warning lights that can be switched on and off.			
WheelspeedOutDemo: WheelspeedDemoAppl	The **WheelspeedDemoAppl** application serves to demonstrate a signal chain with a **Wheelspeed Out** function block that simulates the signals provided by an active wheel speed sensor.	▪ Wheelspeed Out	*SCALEXIO:* ▪ DS2680 I/O Unit	Using the WheelspeedDemoAppl Application on page 71

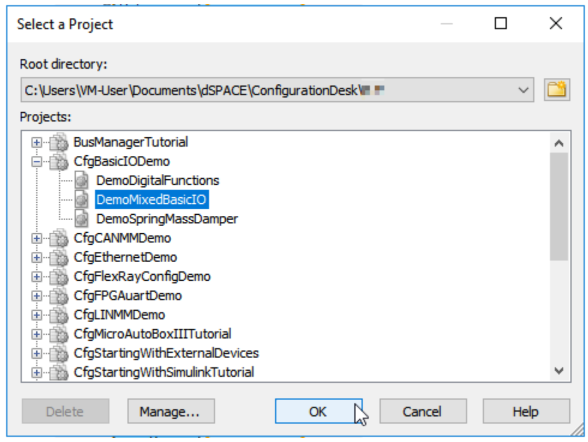
Python-based demo projects Python-based demo projects are *not* available in ConfigurationDesk right away. You have to execute a Python script first to work with them.

Project and Application	Description	Function Blocks	Specific Hardware Used	Refer to...
BusManagerDemo: BusManagerDemoApplication *Only available after executing the `BusManagerDemo.py` Python script*	The Bus Manager demo illustrates the basic steps of using a Python automation script to configure bus communication with the Bus Manager and build a real-time application. The demo is based on a simple example of door and window mechanisms of a car.	▪ Bus Configuration ▪ CAN ▪ LIN	*SCALEXIO:* ▪ DS2671 Bus Board ▪ DS2672 Bus Module	Introduction to the Bus Manager Demo (ConfigurationDesk Bus Manager Implementation Guide 📖)

Accessing ConfigurationDesk Demo Projects and Applications

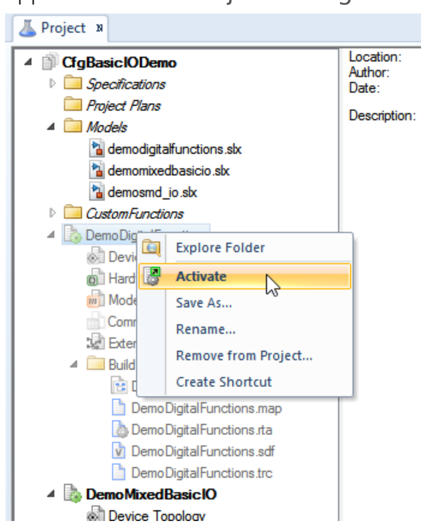
Opening demo projects

You can open demo projects like every ConfigurationDesk project, for example, via **File – Open – Open Project and Application**. ConfigurationDesk opens the **Select a Project** dialog where you can select a demo project from the list of ConfigurationDesk projects in the default root directory. You can also select a specific application to activate after the project is opened. If you do not select an application, the first application in the project will automatically be activated.



Activating applications

To activate a different application in an open demo project, right-click the application in the Project Manager and select Activate from the context menu.



File locations

The files of the demo projects are automatically unpacked to the *Documents folder* when ConfigurationDesk is first started.

They are also backed up in ZIP archives, which are located in the <RCP and HIL installation folder>\Demos\ConfigurationDesk folder after you install the dSPACE software.

Bus Communication Demos

Where to go from here

Information in this section

CfgCANMMDemo Project: Implementing CAN Communication

Using the CANMMAppl Application

Use scenario

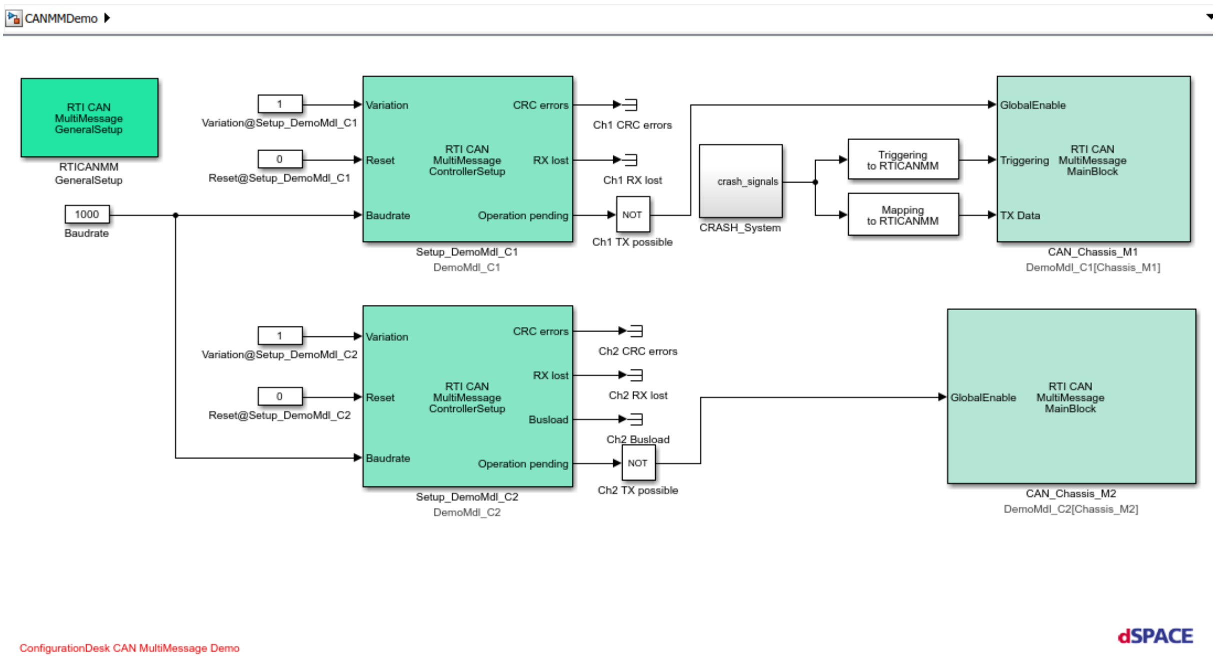
The CANMMAppl application demonstrates simple CAN communication between two CAN controllers using the RTI CAN MultiMessage Blockset.

Tip

- The demo is an example of implementing CAN communication in ConfigurationDesk using the RTI CAN MultiMessage Blockset. For more general information and instructions, refer to Basics on Implementing CAN Communication (ConfigurationDesk Real-Time Implementation Guide 📖).
- For more information on the RTI CAN MultiMessage Blockset, refer to Overview of the RTI CAN MultiMessage Blockset (RTI CAN MultiMessage Blockset Reference 📖).
- Alternatively, you can use the Bus Manager in ConfigurationDesk to configure and implement CAN communication in ConfigurationDesk. Refer to ConfigurationDesk Bus Manager Implementation Guide 📖.

Features in focus

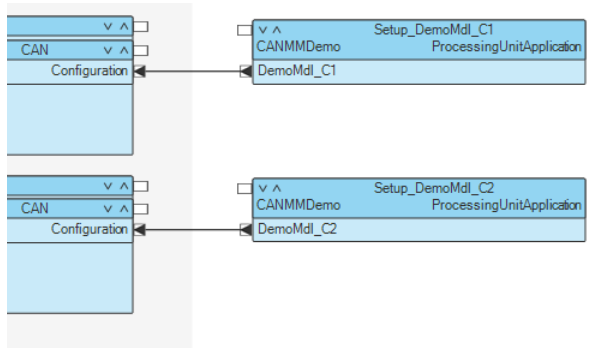
Simulink model with RTICANMM blocks The connected CANMMDemo Simulink model uses blocks from the RTI CAN MultiMessage Blockset to model CAN communication with two CAN controllers.



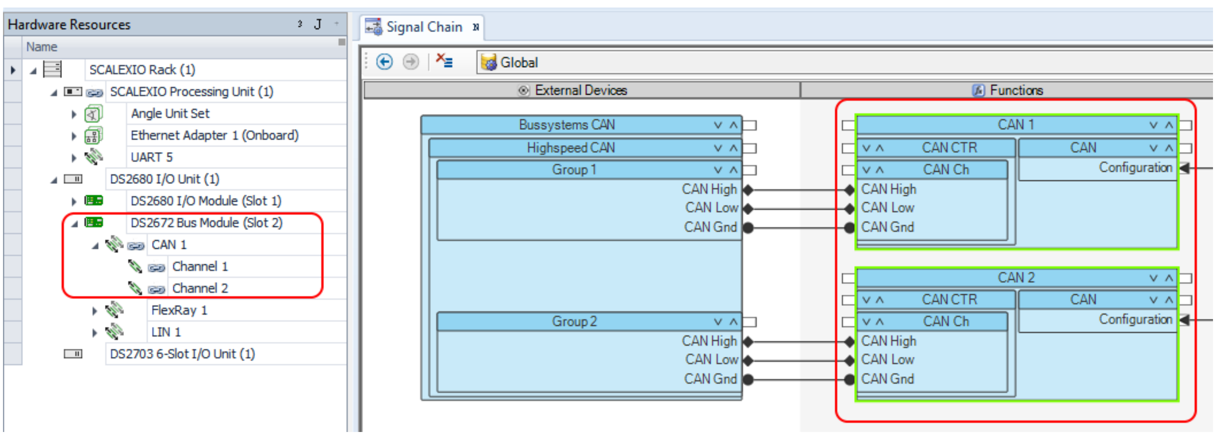
CAN controllers are hardware components that are located on real-time hardware. With a PHS bus system, the RTICANMM ControllerSetup blocks access the CAN controllers directly. But with a SCALEXIO system, access is realized in ConfigurationDesk via CAN function blocks, i.e., the RTICANMM ControllerSetup blocks cannot directly access the controllers. They serve as an interface between the real-time hardware (which is accessed via the CAN function blocks) and the CAN communication that is modeled in the RTICANMM MainBlocks.

Both controllers send and receive messages. CAN_Chassis_M1 is configured to simulate ENGINE_CONTROL ECU and CAN_Chassis_M2 simulates the rest of the ECUs (restbus simulation).

Configuration Port blocks In ConfigurationDesk, each RTICANMM ControllerSetup block is represented by a Configuration Port block in the signal chain. Configuration Port blocks are a specific type of model port block that can be created only by analyzing a Simulink model.

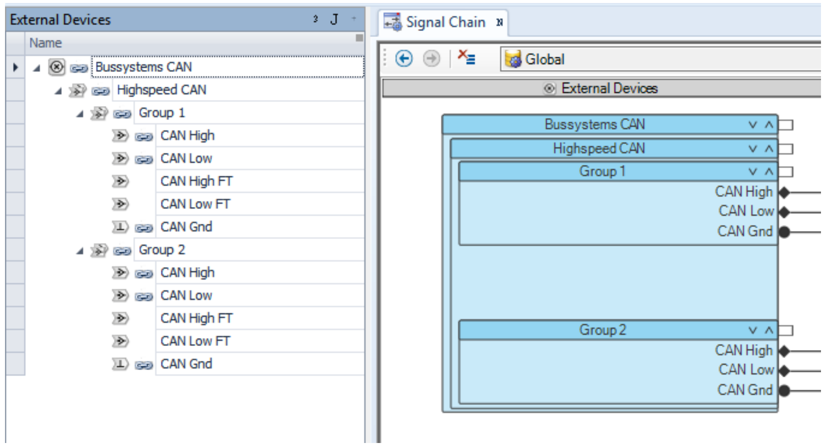


Function blocks and hardware resources The Configuration Port blocks are mapped to CAN function blocks to which CAN channels of the hardware topology are assigned.

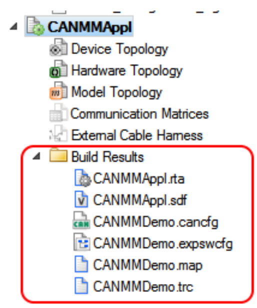


Some function block settings, such as the baud rate, are derived from the RTICANMM ControllerSetup blocks. Other hardware-related settings, such as the transceiver settings, are accessible via properties of the function blocks.

External device interface The external device interface contains a representation of a CAN bus system.



Build results Build results are already available in the Project Manager.



Actions and adjustments

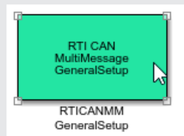
If you have a dSPACE hardware system that provides the required CAN 1 channel type, you can download the real-time application to it. For this purpose, the connection state of the ConfigurationDesk application must be Matching platform connected. The easiest way to achieve this is to register your hardware platform and replace the hardware topology by scanning the registered platform. For more information and instructions, refer to Managing Real-Time Hardware (ConfigurationDesk Real-Time Implementation Guide 📖).

If your dSPACE hardware system is different from the example system in the hardware topology, you have to adjust settings such as the hardware resource assignment. Then, you have to build a new real-time application. Refer to Building Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 📖).

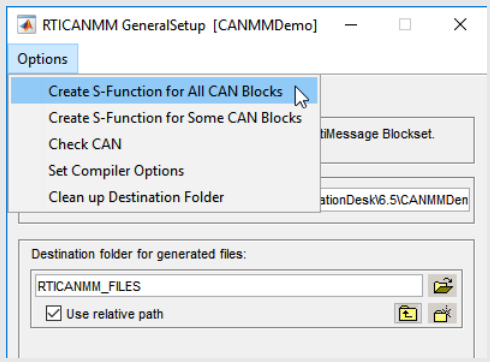
Note

Before you start the build process, you have to create S-functions for the RTI CAN MultiMessage blocks.

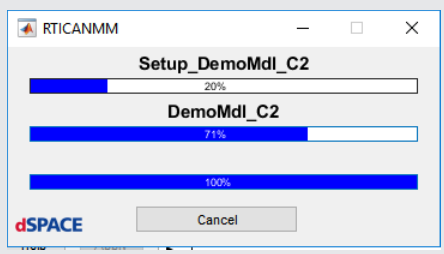
1. Double-click the RTICANMM GeneralSetup block to open its configuration dialog.



2. In the dialog, click Options – Create S-Functions for All CAN Blocks.



The S-functions are created. This might take a while.



3. Click OK to close the dialog.

For information and instructions on downloading the real-time application to a matching platform, refer to Downloading and Executing Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 🕮).

CfgLINMMDemo Project: Implementing LIN Communication

Using the LINMMAppl Application

Use scenario

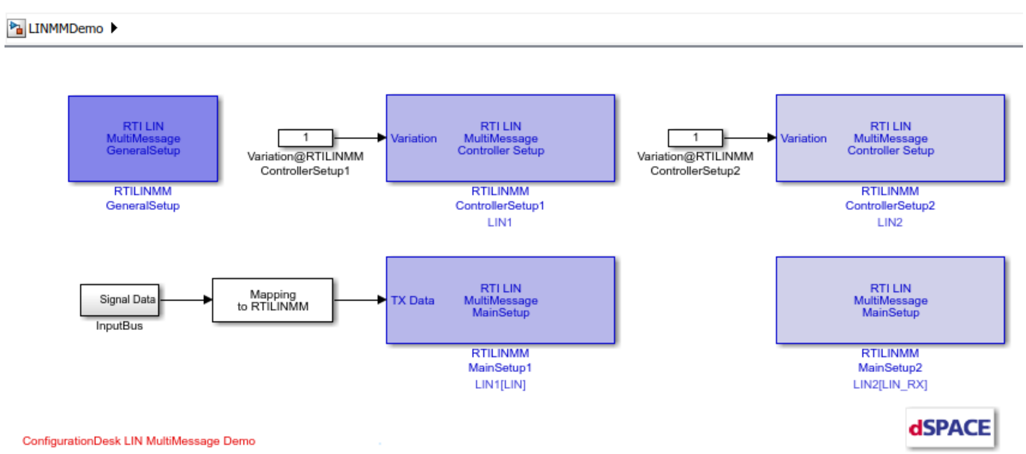
The LINMMAppl application serves to demonstrate simple LIN communication between two LIN controllers using the RTI LIN MultiMessage Blockset.

Tip

- The demo is an example of implementing LIN communication in ConfigurationDesk using the RTI LIN MultiMessage Blockset. For more general information and instructions, refer to Basics on Implementing LIN Communication (ConfigurationDesk Real-Time Implementation Guide 📖).
- For more information on the RTI LIN MultiMessage Blockset, refer to RTI LIN MultiMessage Blockset Reference 📖.
- Alternatively, you can use the Bus Manager in ConfigurationDesk to configure and implement LIN communication in ConfigurationDesk. Refer to ConfigurationDesk Bus Manager Implementation Guide 📖.

Features in focus

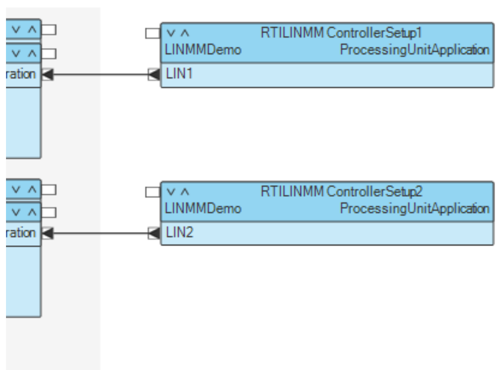
Simulink model with RTILINMM blocks The connected LINMMDemo Simulink model uses blocks from the RTI LIN MultiMessage Blockset to model LIN communication with two LIN controllers.



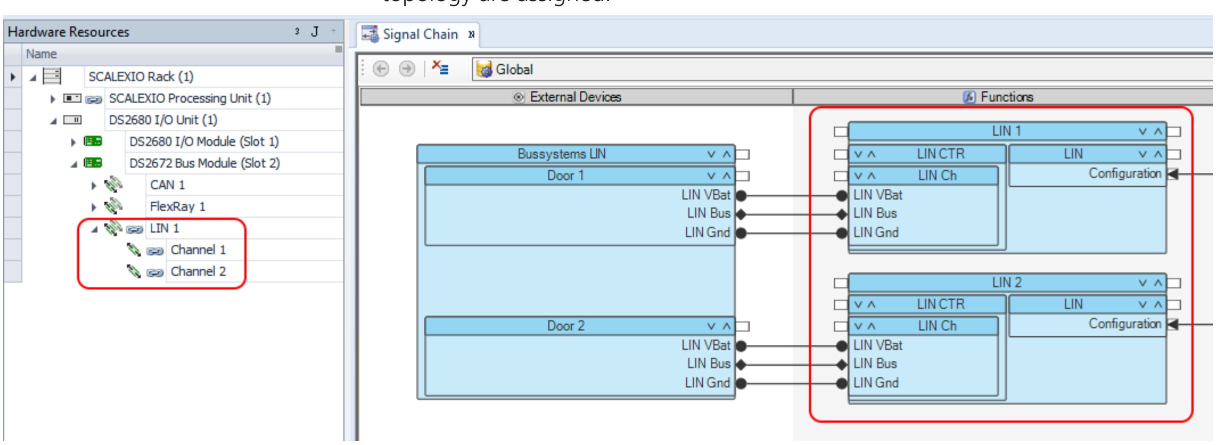
LIN controllers are hardware components that are located on real-time hardware. With a PHS bus system, the RTILINMM ControllerSetup blocks access the LIN controllers directly. But with a SCALEXIO system, access is realized in ConfigurationDesk via LIN function blocks, i.e., the RTILINMM ControllerSetup blocks cannot directly access the controllers. They serve as an interface between the real-time hardware (which is accessed via the LIN function blocks) and the LIN communication that is modeled in the RTILINMM MainSetup blocks.

The communication is based on a master-slave configuration where the Node5 node is the master node. The database delivered with the demo model consists of five nodes (Node1-Node5). The RTILINMM MainSetup1 block simulates the LIN master, i.e., Node5, and the RTILINMM MainSetup2 block simulates the LIN slaves (Node1 ... Node4).

Configuration Port blocks In ConfigurationDesk, each RTILINMM ControllerSetup block is represented by a Configuration Port block in the signal chain. Configuration Port blocks are a specific type of model port block that can be created only by analyzing a Simulink model.

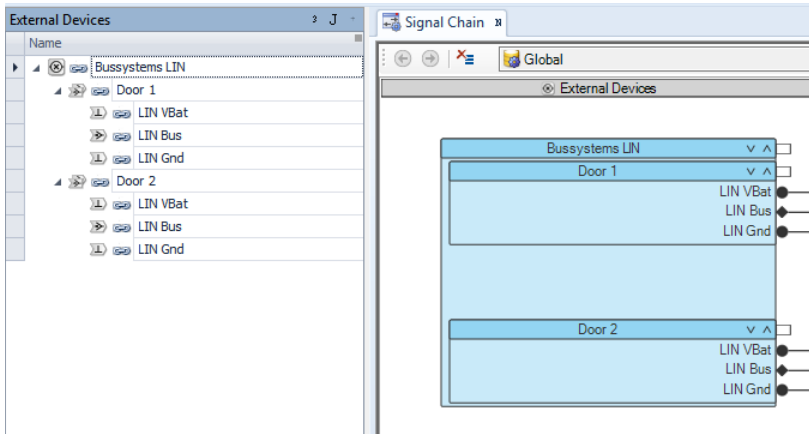


Function blocks and hardware resources The Configuration Port blocks are mapped to LIN function blocks to which LIN channels of the hardware topology are assigned.

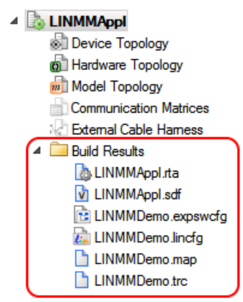


Most function block settings, such as the baud rate, are derived from the RTILINMM ControllerSetup blocks.

External device interface The external device interface contains a representation of a LIN bus system.



Build results Build results are already available in the Project Manager.



Actions and adjustments

If you have a dSPACE hardware system that provides the required LIN 1 channel type, you can download the real-time application to it. For this purpose, the connection state of the ConfigurationDesk application must be Matching platform connected. The easiest way to achieve this is to register your hardware platform and replace the hardware topology by scanning the registered platform. For more information and instructions, refer to Managing Real-Time Hardware (ConfigurationDesk Real-Time Implementation Guide 🕮).

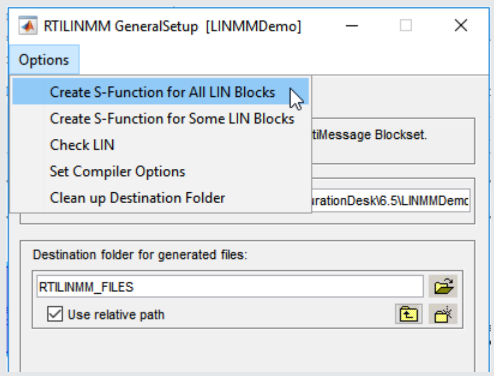
If your dSPACE hardware system is different from the example system in the hardware topology, you have to adjust settings such as the hardware resource assignment. Then, you have to build a new real-time application. Refer to Building Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 🕮).

Note

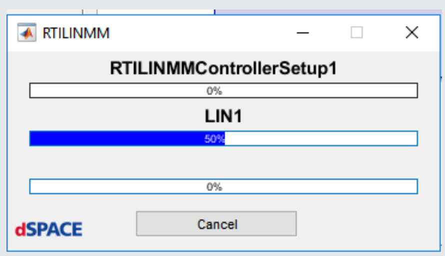
Before you start the build process, you have to create S-functions for the RTI LIN MultiMessage blocks.

1. Double-click the RTILINMM GeneralSetup block to open its configuration dialog.

2. In the dialog, click Options – Create S-Functions for All LIN Blocks.



The S-functions are created. This might take a while.



3. Click OK to close the dialog.

For information and instructions on downloading the real-time application to a matching platform, refer to Downloading and Executing Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 📖).

CfgBasicIODemo Project: Demonstrating Basic I/O Functionalities

Introduction

The **CfgBasicIODemoProject** contains ConfigurationDesk applications that serve to demonstrate basic I/O functionality by using basic function block types from different areas of the function library.

Where to go from here

Information in this section

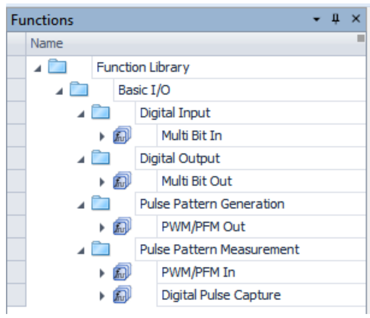
Using the DemoDigitalFunctions Application

Use scenario

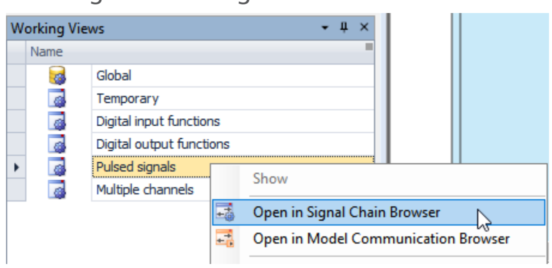
The **DemoDigitalFunctions** application serves to demonstrate a simple signal chain with digital I/O functionality.

Features in focus

Digital I/O functionality The signal chain of the DemoDigitalFunctions application contains several function blocks that measure or create digital signals.

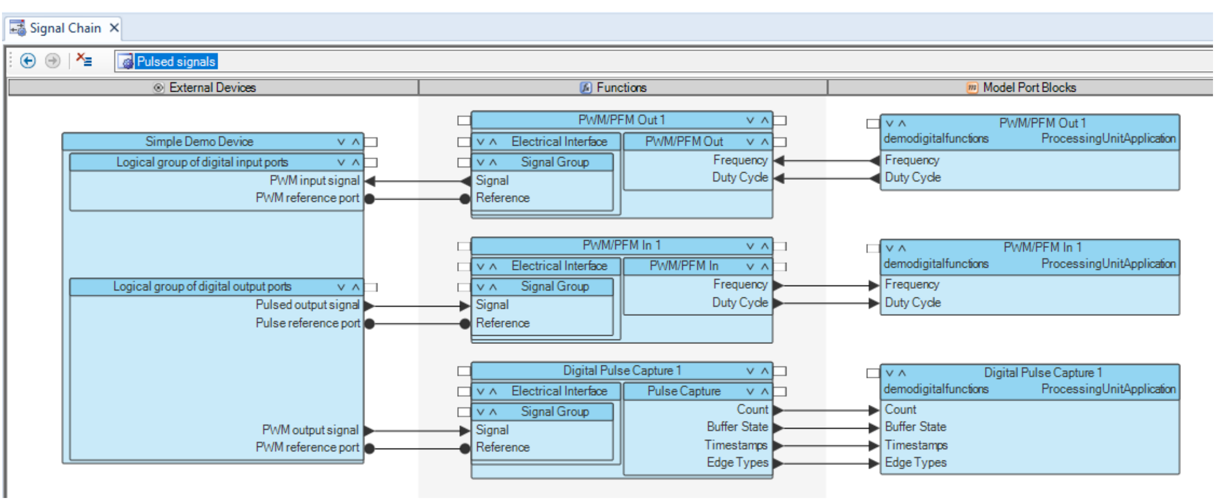


Working views The demo application offers different working views that focus on different areas of the signal chain. You can open them from the Working View Manager.



For more information on working views, refer to Handling the Signal Chain in Working Views (ConfigurationDesk Real-Time Implementation Guide 📖).

Pulse signals The Pulsed signals working view contains the function blocks that generate or measure pulse signals.



- The PWM/PFM Out function block type can be used to generate one-phase pulse-width-modulated signals.

For more information, refer to PWM/PFM Out (ConfigurationDesk I/O Function Implementation Guide 📖).

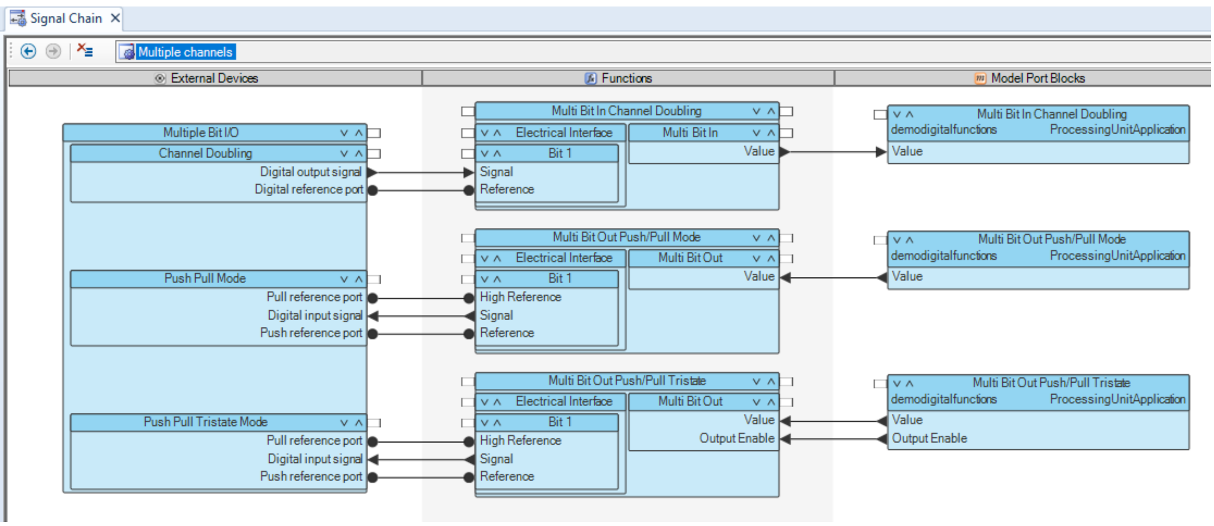
- With the **PWM/PFM In** function block, you can measure one-phase pulse-width-modulated signal patterns.

  For more information, refer to PWM/PFM In (ConfigurationDesk I/O Function Implementation Guide 📖).

- The **Digital Pulse Capture** function block converts signals coming from an external device (e.g., ECU) into corresponding time stamps, angle positions, and edge polarities.

  For more information, refer to Digital Pulse Capture (ConfigurationDesk I/O Function Implementation Guide 📖).

Channel multiplication The **Multiple channels** working view contains the function blocks that are configured to use channel multiplication.



With the channel multiplication feature, you can enhance the max. current or max. voltage of a single hardware channel. For more information, refer to Specifying Current and Voltage Values for Channel Multiplication (ConfigurationDesk I/O Function Implementation Guide 📖).

The following function block types are used to illustrate channel multiplication:

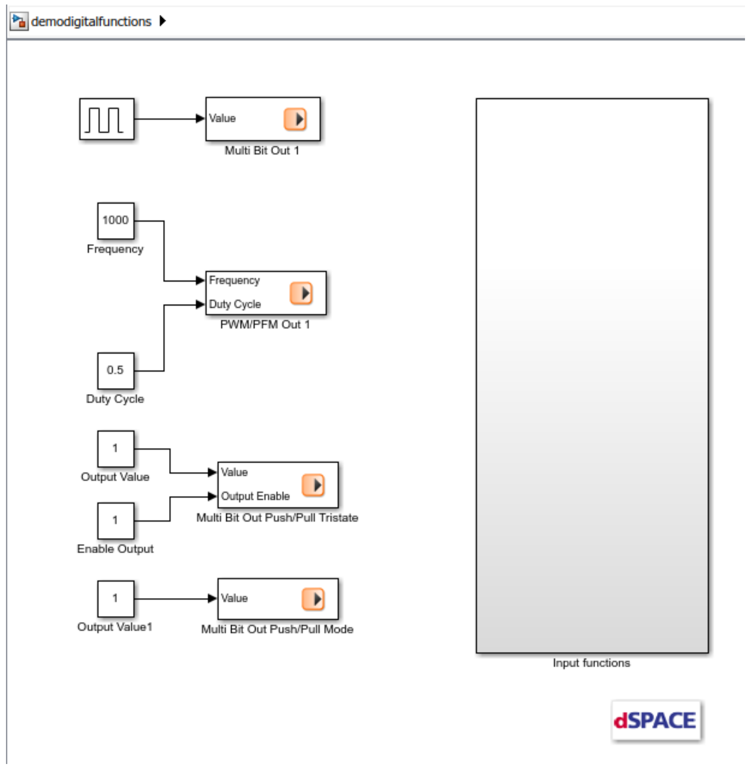
- The **Multi Bit In** function block type lets you measure digital signals coming from an external device.

  For more information, refer to Multi Bit In (ConfigurationDesk I/O Function Implementation Guide 📖).

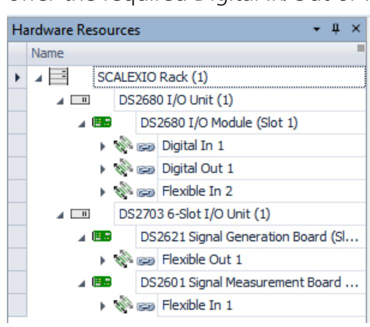
- The **Multi Bit Out** function block type lets you stimulate digital inputs of an external device.

  For more information, refer to Multi Bit Out (ConfigurationDesk I/O Function Implementation Guide 📖).

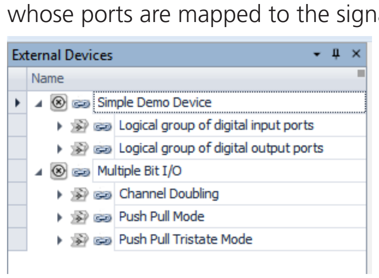
Simulink model The connected **demodigitalfunctions** Simulink model contains simple model port blocks to generate or receive the digital signals. The model has no logical internal structure.



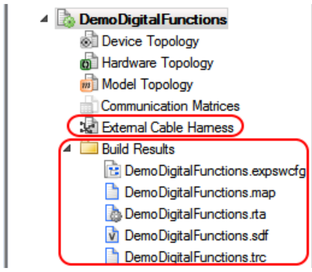
Assigned hardware resources The assigned SCALEXIO hardware resources offer the required Digital In/Out or Flexible In/Out channel types.



External device interface The external device interface contains two devices whose ports are mapped to the signal ports of the function blocks.



Build results and wiring information Build results and the calculated wiring information for an external cable harness are already available in the Project Manager.



Actions and adjustments

You are not required to perform specific actions because the demo illustrates a finished ConfigurationDesk application.

However, you are free to adjust the configuration and, for example, start a new build process.

You can also use the demo as a basis for a real use scenario that requires digital I/O functionality.

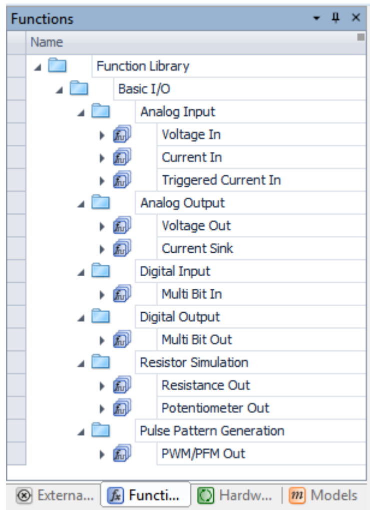
Using the DemoMixedBasicIO Application

Use scenario

The DemoMixedBasicIO application serves to demonstrate a simple signal chain with basic I/O functionality.

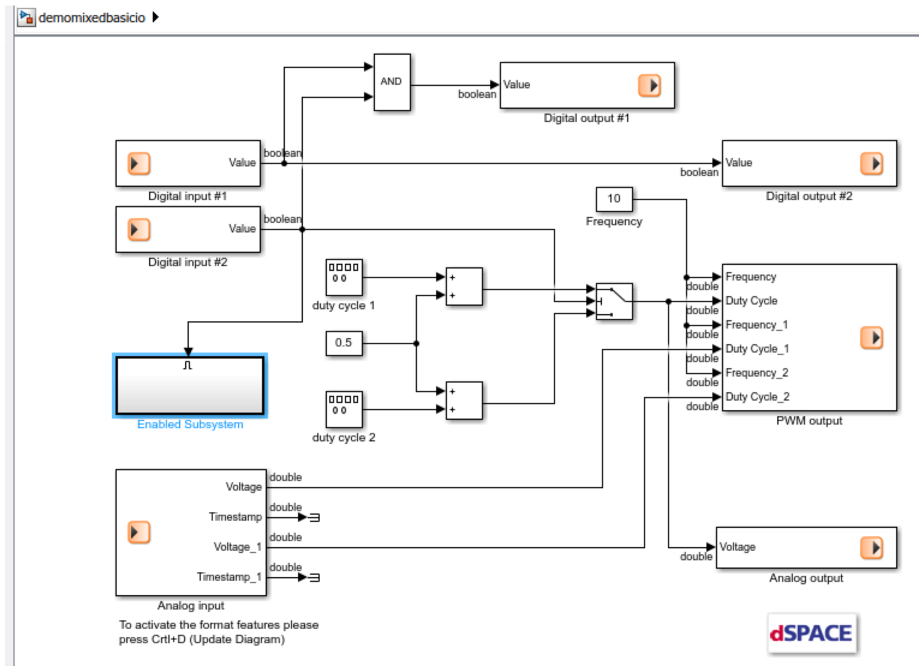
Features in focus

Basic I/O functionality The signal chain of the DemoMixedBasicIO application contains several function blocks of the Basic I/O category in the function library.

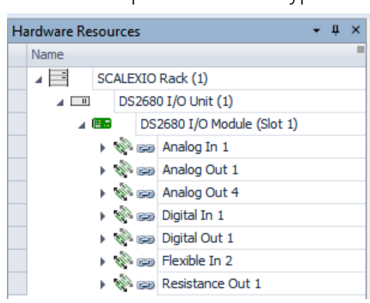


For a short description of each function block type and links to detailed documentation for each type, refer to the Basic I/O category here: Overview on the Available Function Block Types (ConfigurationDesk I/O Function Implementation Guide 📖).

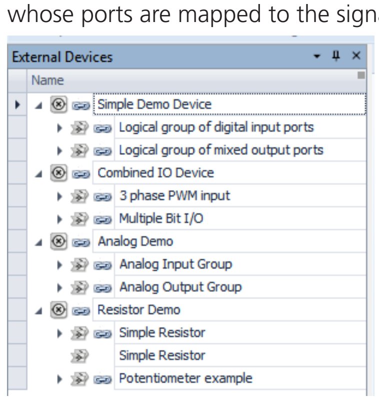
Simulink model The connected demomixedbasicio Simulink model contains simple model port blocks to generate or receive the required signals. The model has no logical internal structure.



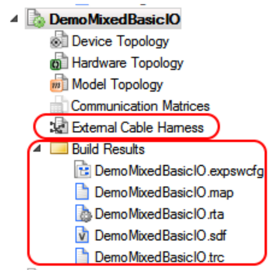
Assigned hardware resources The assigned SCALEXIO hardware resources offer the required channel types.



External device interface The external device interface contains four devices whose ports are mapped to the signal ports of the function blocks.



Build results and wiring information Build results and the calculated wiring information for an external cable harness are already available in the Project Manager.



Actions and adjustments

You are not required to perform specific actions because the demo illustrates a finished ConfigurationDesk application.

However, you are free to adjust the configuration and, for example, start a new build process.

You can also use the demo as a basis for a real use scenario that requires some of the demonstrated I/O functionality.

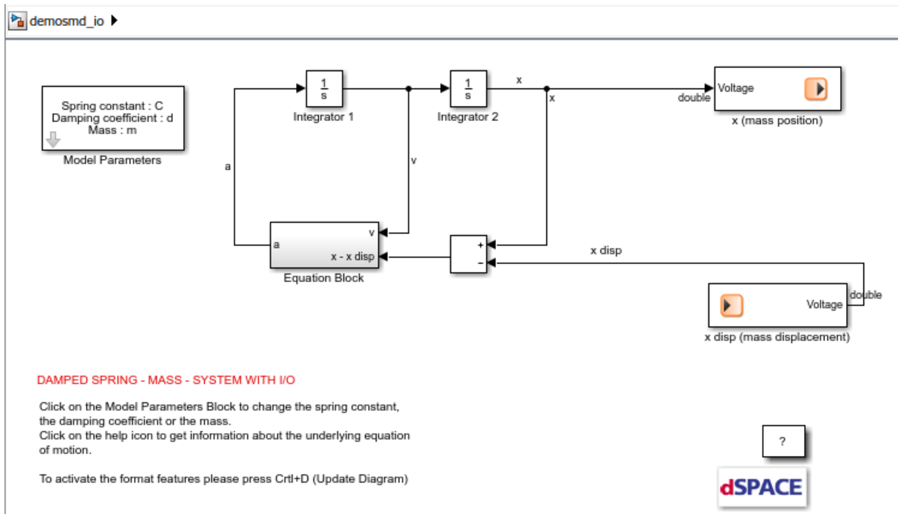
Using the DemoSpringMassDamper Application

Use scenario

The DemoSpringMassDamper application illustrates the simulation of a damped spring-mass system. The anchor point where the spring is connected will be deflected by an external force. The resulting signal describes the deflection of the mass at the other side of the spring.

Features in focus

Modeling a damped spring-mass system The demosmd_io Simulink model connected to the ConfigruationDesk application simulates a damped spring-mass system.



The model sends and receives voltage signals to ConfigurationDesk via model port blocks.

The equation of motion for the center of the mass driven by an external force is:

$a = -(d/m) \cdot v - (C/m) \cdot (x - x\,disp)$

Variable	Description	Unit
a	Acceleration of the mass	m/s^2
d	Damping coefficient	kg/s
m	Mass	kg
v	Velocity of the mass	m/s
c	Spring constant	kg/s^2
x	Position of the mass	m
$x\,disp$	Excitation	m
f	Frequency	Hz
t	Time	s
u	Amplitude of excitation	m

The output of the Integrator 1 block (v) is multiplied with d/m in the Equation Block. The output x of the Integrator 2 block can be measured on DAC channel 1. It is added to $x\,disp$, and the sum is multiplied with C/m in the Equation Block. The output of the ADC is the exciting displacement of the mass: $x\,disp$.

In the case of a sine-like excitation, the driving force is:

$F = x\,disp \cdot Cx\,disp = u \cdot cos(2\pi \cdot f \cdot t)$

The solution of the equation of motion is:

$x(t) = xo \cdot cos(2\pi \cdot f \cdot t - \alpha)$

A harmonic oscillation with the original frequency, an amplitude xo, and a phase α, both depending on the model parameters.

In the case of a square-like excitation, the driving force results from a single displacement of u, twice in the period $T = 1/f$:

$F = u \cdot C$

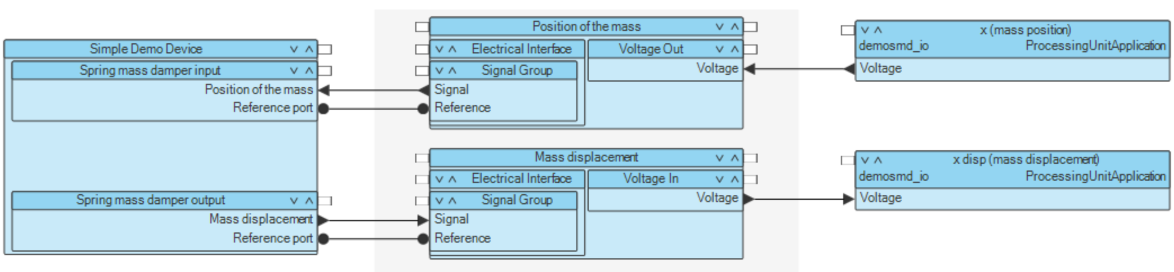
The solution of the equation of motion is:

$x(t) = xo \cdot exp[-(d/2m) \cdot t] \cdot cos(\omega \cdot t)$

An exponentially decreasing sine wave with the natural frequency:

$\omega = \sqrt{(C/m) - (d^2/4m^2)}$

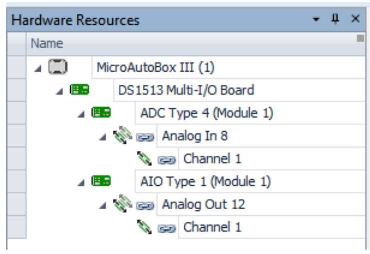
Signal chain in ConfigurationDesk In ConfigurationDesk, the x (mass position) signal is received from the Simulink model via a model port block that is connected to a Voltage Out function block. The signal ports of the function block are connected to a Simple Demo Device that represents an ECU. A Voltage In function block is connected to an outport of the device and to a model port block that returns the x disp (mass displacement) signal to the Simulink model.



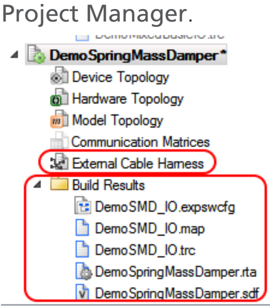
For more information on the used function block types, refer to:

- Voltage Out (ConfigurationDesk I/O Function Implementation Guide 📖)
- Voltage In (ConfigurationDesk I/O Function Implementation Guide 📖)

Hardware resources The DS1513 Multi-I/O Board of the MicroAutoBox III is used as an example of dSPACE hardware that provides the required Analog Out and Analog In channels.



Build results and wiring information Build results and the calculated wiring information for an external cable harness are already available in the Project Manager.



Actions and adjustments

- You can change the spring constant, the damping coefficient, and the mass in the Model Parameters block of the Simulink model.
- If you have a dSPACE hardware system that provides the required Analog Out and Analog In channel types, you can download the real-time application to it. For this purpose, the connection state of the ConfigurationDesk application must be Matching platform connected. For more information and instructions, refer to Managing Real-Time Hardware (ConfigurationDesk Real-Time Implementation Guide 📖).

  For information and instructions on downloading the real-time application to a matching platform, refer to Downloading and Executing Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 📖).

  Note

  If your dSPACE hardware system is different from the example MicroAutoBox III system in the hardware topology, you have to replace the hardware topology and adjust settings such as the hardware resource assignment. Then, you have to build a new real-time application. Refer to Building Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 📖).

- You can connect external devices to dSPACE hardware using the wiring information from the calculated external cable harness. Refer to Calculating an External Cable Harness (ConfigurationDesk Real-Time Implementation Guide 📖).

Note

- Make sure that the device pins and the I/O connector pins of the dSPACE hardware are connected according to the calculated wiring information. You can export the wiring information to an XML or Microsoft Excel™ (XLSX) file.
- If you apply changes to the configuration that affect the wiring information, such as changing the hardware resource assignment, you must recalculate the external cable harness.

- If you downloaded the real-time application to a matching platform, you can start experimenting in ControlDesk. Your ConfigurationDesk installation contains a ControlDesk demo project that is based on this ConfigurationDesk demo project.

  The **CDNG_BasicIODemo** is located next to the **CfgBasicIODemo** in the `<Documents folder>\BasicIODemo\` folder. It is also available in a ZIP archive, which is located in the `<RCP and HIL installation folder>\Demos\ConfigurationDesk\BasicIODemo` folder. The files in the `Variable Descriptions` folder are the build results of the ConfigurationDesk project.

  Refer to:

  - For SCALEXIO: Experimenting with a SCALEXIO System (SCALEXIO – Hardware and Software Overview 🕮).
  - For MicroAutoBox III: Software for Experimenting with the MicroAutoBox III (MicroAutoBox III - Hardware and Software Overview 🕮)

Note

If you run the demo without external wiring, it can be used only with test automation (TA) access in ControlDesk. In the layout, set **Voltage/TA_Switchvalue** to **Substitute value**. When you adjust **Voltage/TA_Replacevalue**, the change of mass position will be simulated and displayed in the plotter and the red **Voltage/MDL_Signal** will respond to the changes in a damped curve.

CfgUserStorageDemo Project: Using a SCALEXIO SSD

Using the UserStorageApplication Application

Use scenario

The UserStorageApplicaton application provides custom function block types that let you use a SCALEXIO SSD.

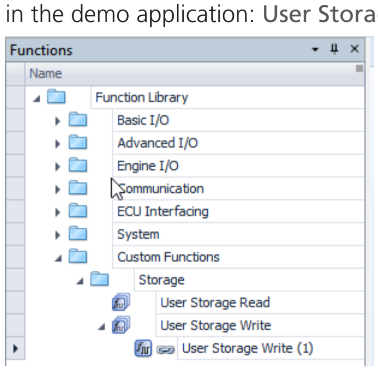
Features in focus

SCALEXIO SSD The SCALEXIO SSD is an optional feature of the SCALEXIO Processing Unit. The individual user data of the SSD can be accessed via FTP (ftp://<SCALEXIO_IP_address>/userstorage1).

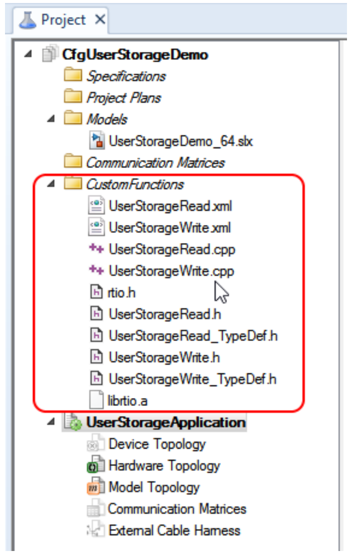
The SCALEXIO SSD is not available in the hardware topology of the demo application because it has to be added by scanning a registered platform that contains the SCALEXIO SSD.

For more information on the hardware specifics, refer to Features of the SCALEXIO Processing Unit (SCALEXIO Hardware Installation and Configuration 📖).

Custom function block types Two custom function block types are provided in the demo application: User Storage Write and User Storage Read.



The custom function files are located in the project folder and are also available in the Project Manager.



Function Block Type	Description	Specific Properties
User Storage Write	Bundles data from the connected Simulink model in blocks and writes them to the SCALEXIO SSD.	▪ Filename: The name of the file on the SCALEXIO SSD to which you write. ▪ Split files on EOF: Due to limitations of some file systems regarding maximum file size (FAT32: 4GB), this parameter lets you split the target file into multiple parts when the maximum file size is reached.

User Storage Read	Reads data from the SCALEXIO SSD and makes it available in the application.	▪ Filename: The name of the file on the SCALEXIO SSD from which you read.
		▪ Wrap behavior: Defines what happens when the end of the file is reached.
		  ▪ Wrap: The read process starts again at the beginning of the file.
		  ▪ No wrap: No data is read after the end of the file has been reached.
		  ▪ Initial values: The initial values of the custom function block are provided after the end of the file has been reached.

Note

The signal chain of the application contains only a User Storage Write function block for the following reasons:

- You cannot write to and read from the same file simultaneously.
- Before you can read from the SCALEXIO SSD, data must have been written to it.

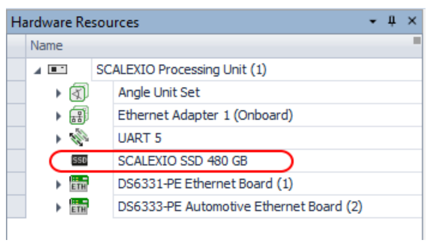
Tip

- The `rtio.h` custom function file contains developer comments that provide additional details.
- For more information and instructions on creating and using custom function block types in ConfigurationDesk, refer to ConfigurationDesk Custom I/O Function Implementation Guide.

Simulink Model The connected UserStorageDemo_64 Simulink model contains simple write and read example subsystems that write/read the current date to/from the ConfigurationDesk application via model port blocks.

Actions and adjustments

- If you have a SCALEXIO system with a SCALEXIO SSD, you can register and scan the platform to create a matching hardware topology that contains the SCALEXIO SSD. Refer to Managing Real-Time Hardware (ConfigurationDesk Real-Time Implementation Guide).



  You can then build, download, and execute the real-time application.

  Refer to:
  - Building Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide)
  - Downloading and Executing Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide).
- To work with the data written to the SCALEXIO SSD with a real-time application using the **User Storage Write** custom function block type, you can create, build, download, and execute an additional real-time application using the **User Storage Read** custom function block type. You can do so in the same ConfigurationDesk project or create a new one.
- You can use the demo as a basis for a more complex use scenario that writes/reads different data.

EDrivesControlDemo Project: Example of Electric Motor Control Using the DS6121 Multi-I/O Board

Where to go from here

Information in this section

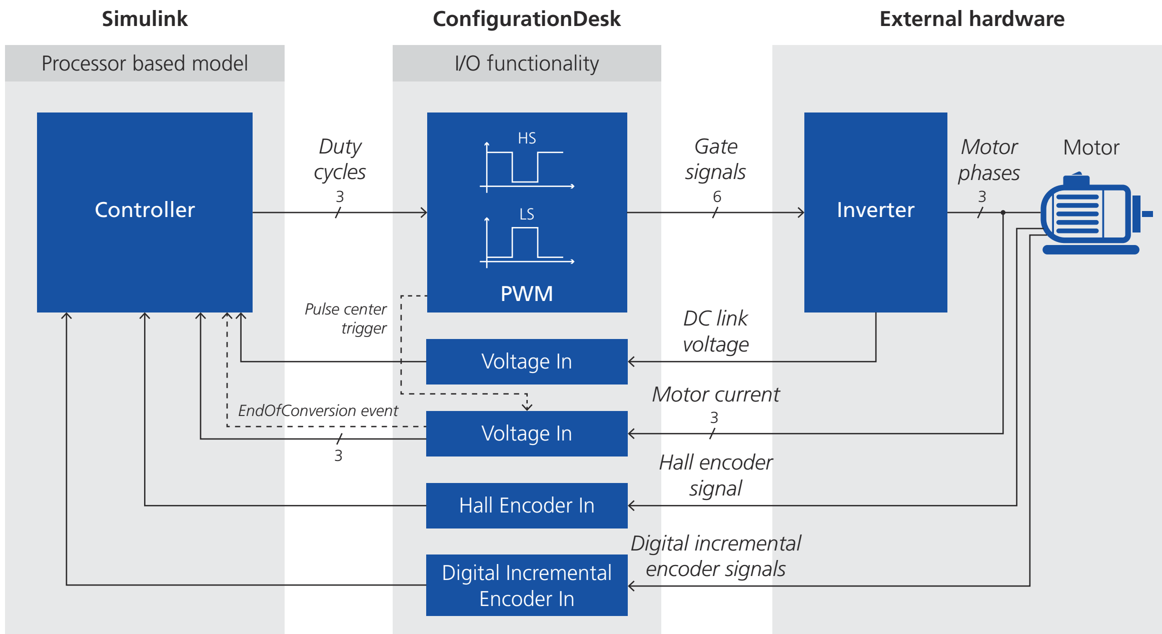
Introduction to the EDrivesControlDemo Project

Use scenario

ConfigurationDesk provides the EDrivesControlDemo project as an example for electric motor control in combination with the DS6121 Multi-I/O Board. You can use the demo project as a template for your own electric motor control projects.

Demo project overview

In the demo applications, the controller is implemented in a Simulink model. It controls the generation of the three-phase PWM signal, which is fed into the PWM channels of the DS6121 Multi-I/O Board. Refer to the following illustration:



I/O functionality in ConfigurationDesk The function blocks in ConfigurationDesk have the following purpose:

- Generating three-phase PWM signals including the inverted phases to control an electric motor

  The PWM signals are output by the DS6121 Multi-I/O Board.

  The PWM signals can be generated by a Multi-Channel PWM Out function block (PMSM_Demo) or a Block-Commutated PWM Out function block (BLDC_Demo). For more information on the function blocks, refer to the following topics:

  - Multi-Channel PWM Out (ConfigurationDesk I/O Function Implementation Guide 🕮)
  - Block-Commutated PWM Out (ConfigurationDesk I/O Function Implementation Guide 🕮)

- Calculation of the rotor position and the velocity of the motor

  The calculation of the position and the velocity of the motor is done via a Hall Encoder In function block and a Digital Incremental Encoder In function block. For more information on the function blocks, refer to the following topics:

  - Hall Encoder In (ConfigurationDesk I/O Function Implementation Guide 🕮)
  - Digital Incremental Encoder In (ConfigurationDesk I/O Function Implementation Guide 🕮)

- A/D conversion of the phase currents and the DC link voltage

  The A/D conversion of the phase currents and the DC link voltage is done via Voltage In function blocks.

For more information on the function block, refer to Voltage In (ConfigurationDesk I/O Function Implementation Guide 📖).

Note

As an alternative, the **Voltage Signal Capture** function block can be used for this purpose. For more information, refer to Voltage Signal Capture (ConfigurationDesk I/O Function Implementation Guide 📖).

- Triggering the execution of the control algorithm

  All A/D conversions for current measurement are triggered synchronously at the pulse center of the generated PWM signal. After all A/D conversions have finished, the asynchronous **Interrupt** task is triggered. This task then triggers the execution of the control algorithm in Simulink.

  Only one of the three **Voltage In** function blocks for A/D conversion of the phase currents generates the **EndOfConversion** event. It is ensured that all A/D conversions are completed when the event is sent.

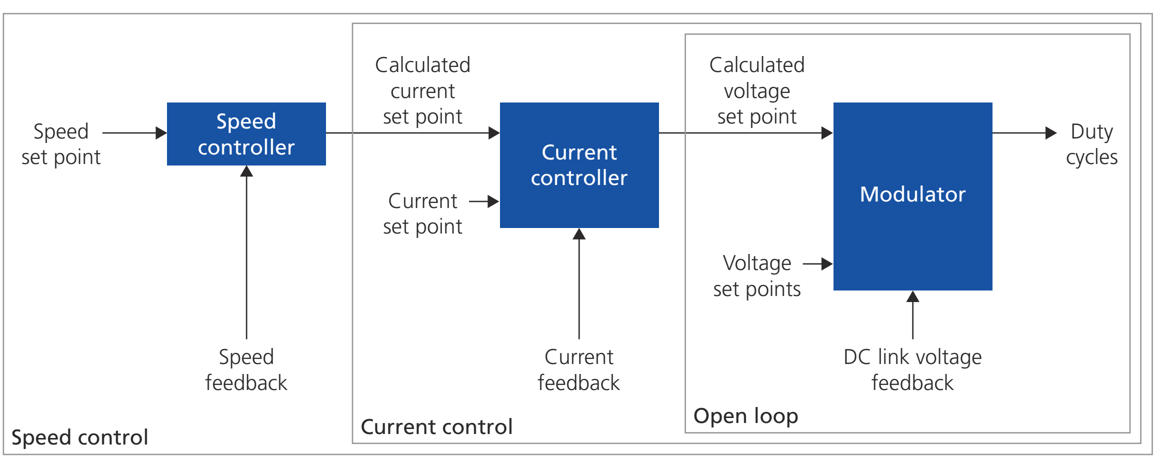
Note

The measured analog values are adapted in the **Input** subsystem of the controller model.

Control algorithm in Simulink The controller reads the position and speed values, the phase currents and the DC link voltage. It calculates the values that are required for the modulation of the PWM signal, such as duty cycle or frequency, to the function block that generates the PWM signals.

Cascaded controller structure

The control algorithms in both demos have a cascaded control architecture with three levels. Refer to the following illustration:



Modulator The modulator resides at the innermost level of the controller model and converts voltage setpoints to duty cycles. The output of the modulator is sent to the relevant **Data Outport** block (**PMSM_Demo**: **Three_Phase_PWM**, **BLDC_Demo**: **Block-Commutated_PWM**). The voltage setpoints can either be user-defined voltage setpoints, or they can be calculated voltage setpoints received by the current controller.

Current controller The controller at the next level up is the current controller. It contains a PI controller that uses the difference between the current setpoint and the measured current values to calculate a correction value that is then applied to the modulator. The current setpoint can either be user-defined or it can be a calculated current setpoint received from the speed controller.

Speed controller The speed controller is located at the top level of the controller model. It contains a PI controller that uses the difference between the speed setpoint and the measured speed values to calculate a correction value that is then applied to the current controller.

Demo parameters

The controller in the Simulink model contains the Demo Parameters subsystem. This subsystem contains all the relevant parameters that are loaded from the MATLAB workspace. You can change the demo parameters in real time in a ControlDesk application. The parameters are grouped as follows:

- Setpoints group:

  The setpoints are user-defined values for the different controller levels, i.e., the voltage setpoint and the current setpoint. The Setpoints group also contains a parameter that is used to specify the control mode such as speed control or open loop control.
- Controller Parameters group:

  This group contains the controller gains and filter time constants for the speed controller and the current controller in the demo applications.
- Setpoint Limits group:

  This group contains constants that are used to limit the user-defined input setpoints, or to saturate the output of the PI controllers.

Initializing the controller The default values of the demo parameters for initializing the controller are specified via MATLAB script files. In the script files you can also configure the motor settings and the controller settings. There is a specific MATLAB script available for each demo application:

- `PMSM_Demo_ini.m`
- `BLDC_Demo_ini.m`

If you start a ConfigurationDesk build process including a model build process, the script files are executed automatically. You can specify your own script files via the `PreLoadFcn` model callback in the Model Explorer in Simulink.

Demo applications

The demo project contains the following ConfigurationDesk applications:

- PMSM_demo for controlling a permanent magnet synchronous motor (PMSM) with field-oriented control. Refer to Using the PMSM_Demo Application on page 45.
- BLDC_demo for controlling a brushless DC motor (BLDC) in block commutation mode. Refer to Using the BLDC_Demo Application on page 47.

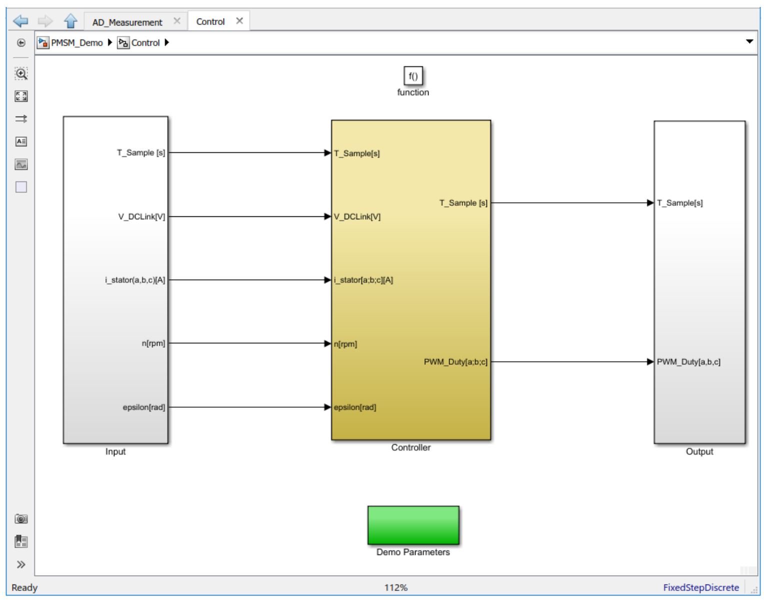
Using the PMSM_Demo Application

Introduction

The PMSM_Demo application shows an example of using the DS6121 Multi-I/O Board for controlling a permanent magnet synchronous motor (PMSM) with field-oriented control. In this demo, sinusoidal phase currents are generated to control the motor. This is called sine commutation. Field-oriented control means, that the coordinate system of the control loop is rotated with the rotor position. This converts the sinusoidal setpoints for the currents into constant setpoints that are easier to control.

Controller structure

The controller in Simulink is divided into different subsystems as shown in the following illustration:



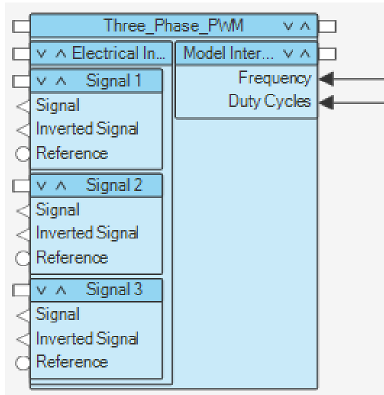
Input subsystem The Input subsystem is used to configure encoders and read encoder output signals as well as phase current and DC link voltage.

Controller subsystem The Controller subsystem contains the control algorithm.

Output subsystem The Output subsystem contains the output signals that are used by the Multi-Channel PWM Out function block in ConfigurationDesk to generate sine commutated PWM signals.

Generation of sinusoidal commutated PWM signals

The sinusoidal-commutated PWM signal for controlling the electric motor is generated by the Multi-Channel PWM Out function block in ConfigurationDesk.



The sinusoidal-commutated PWM signal is modulated according to its input signals, i.e., the duty cycle and the frequency. The frequency has a constant value in this demo application.

Initializing the controller

You must specify default values for initializing the controller in the `PMSM_Demo_ini.m` script file:

PWM period You can specify the PWM period via the `T_PWM` parameter.

Machine parameters The script file lets you specify the following machine parameters:

- Stator inductance and stator resistance
- The flux induced by the magnet
- Number of pole pairs

  You must specify the `PolePairs` parameter according to the number of pole pairs of the motor and the Hall encoder. This value must also be specified for the Number of Pole Pairs property of the Hall Encoder In function block in ConfigurationDesk.

  For more information, refer to Introduction to Hall Encoders (ConfigurationDesk I/O Function Implementation Guide 📖).
- The DC link voltage
- Maximum values for currents and the mechanical speed

Controller settings The script file lets you specify the following parameters:

- Default controller settings, for example, proportional or integral gain
- Setpoints for the currents and the mechanical velocity
- The source of the motor position (Hall encoder or incremental encoder)

- Parameters for the open loop control mode

  For the open loop control mode, you must specify the voltage values (`PMSM_Ctrl_v_d_Set` and `PMSM_Ctrl_v_q_Set`), and the `PMSM_Ctrl_fRS_Set` parameter, which indicates the rotation frequency of the motor.
- Field weakening

  You can specify the field weakening current via the `Const_Id_Fieldweakening` parameter. The `SW_Fieldweakening` parameter lets you switch the field weakening on or off.

Specifying the type of control Via the `PMSM_Ctrl_Ctrl_mode_SW` parameter in the script file or in ControlDesk, you can specify the type of control. Refer to the following table:

PMSM_Ctrl_Ctrl_mode_SW Parameter Value	Description
1	Open loop
2	Current control
3	Speed control

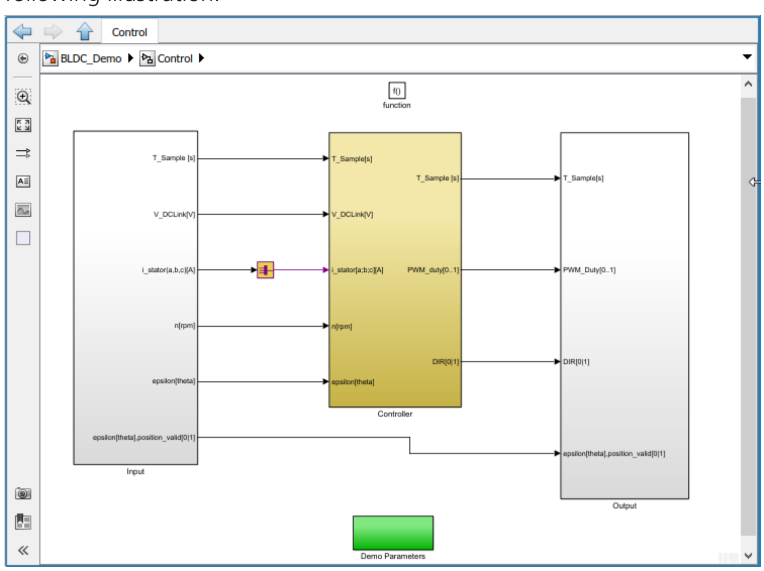
Using the BLDC_Demo Application

Introduction

The BLDC_Demo application is an example of using the DS6121 Multi-I/O Board for controlling a brushless DC motor in block commutation mode.

Controller structure

The controller in Simulink is divided into different subsystems as shown in the following illustration:



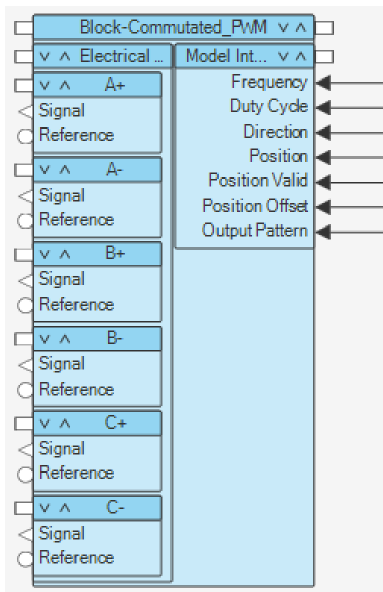
Input subsystem The Input subsystem is used to configure encoders and read encoder output signals as well as phase currents and DC link voltage.

Controller subsystem The Controller subsystem contains the control algorithm.

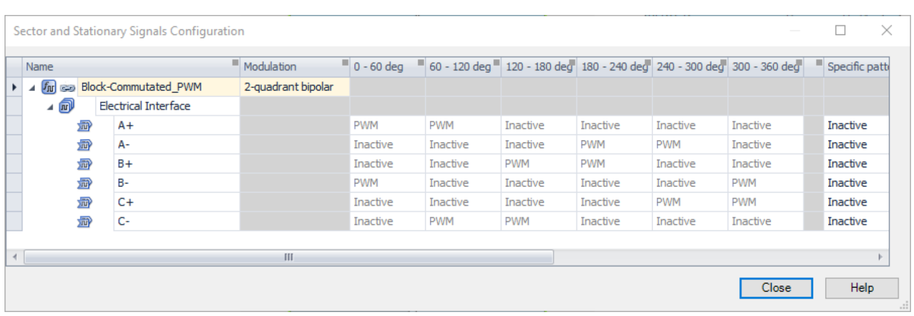
Output subsystem The Output subsystem contains the output signals that are used by the Block-Commutated PWM Out function block in ConfigurationDesk to generate block-commutated PWM signals.

Generation of the block-commutated PWM signal

The block-commutated PWM signal for controlling the electric motor is generated by the Block-Commutated PWM Out function block in ConfigurationDesk.



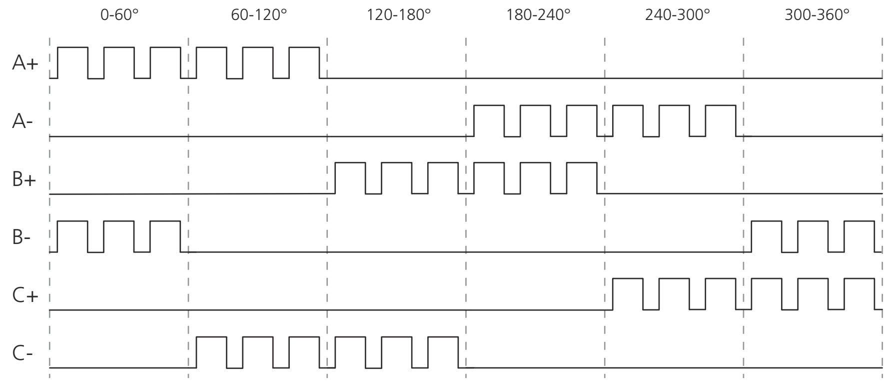
The block-commutated PWM signal is modulated according to the input signals and the configuration of the Block-Commutated PWM Out function block. In the BLDC_Demo application, the function block generates motor-position-dependent sector signals according to the specified modulation type. The Block-Commutated PWM Out function block provides different predefined modulation types. In this demo application, the 2-quadrant bipolar modulation type is preconfigured. The related signal settings are displayed in the Sector and Stationary Signals dialog:


Sector and Stationary Signals Configuration

Name	Modulation	0 - 60 deg	60 - 120 deg	120 - 180 deg	180 - 240 deg	240 - 300 deg	300 - 360 deg	Specific patt
Block-Commutated_PWM	2-quadrant bipolar							
Electrical Interface								
A+		PWM	PWM	Inactive	Inactive	Inactive	Inactive	Inactive
A-		Inactive	Inactive	Inactive	PWM	PWM	Inactive	Inactive
B+		Inactive	Inactive	PWM	PWM	Inactive	Inactive	Inactive
B-		PWM	Inactive	Inactive	Inactive	Inactive	PWM	Inactive
C+		Inactive	Inactive	Inactive	Inactive	PWM	PWM	Inactive
C-		Inactive	PWM	PWM	Inactive	Inactive	Inactive	Inactive

Close Help

The following illustration shows the corresponding signal pattern:



For more information on sector and stationary signals, refer to Configuring Sector and Stationary Signals (Block-Commutated PWM Out) (ConfigurationDesk I/O Function Implementation Guide 📖).

Initializing the controller

You must specify default values for initializing the controller in the `BLDC_Demo_ini.m` script file:

PWM period You can specify the PWM period via the `T_PWM` parameter.

Machine parameters The script file lets you specify the following machine parameters:

- Stator inductance and stator resistance
- The flux induced by the magnet
- Number of pole pairs

  You must specify the `PolePairs` parameter according to the number of pole pairs of the motor and the Hall encoder. This value must also be specified for the **Number of Pole Pairs** property of the **Hall Encoder In** function block in ConfigurationDesk. For more information, refer to Introduction to Hall Encoders (ConfigurationDesk I/O Function Implementation Guide 📖).

  Note

  The **Pole pair factor** property of the **Block-Commutated PWM Out** function block must be 1, because the conversion between electrical and mechanical angle is performed in the model. For more information, refer to Block-Commutated PWM Out (ConfigurationDesk I/O Function Implementation Guide 📖).

  For more information, refer to Introduction to Hall Encoders (ConfigurationDesk I/O Function Implementation Guide 📖).
- The DC link voltage
- Maximum values for current and mechanical speed

Controller settings The script file lets you specify the following parameters:

- Default controller settings, for example, proportional or integral gain
- The speed setpoint and the current setpoint
- The source of the motor position (Hall encoder or incremental encoder)

The following parameters must be specified according to the Block-Commutated PWM Out function block configuration in ConfigurationDesk:

- Indicator vector for determining the direction of the phase currents

  The vector is specified via the `BLDC_Ctrl_is_dir` parameter. In this vector, the sign of the three phase-currents is provided for the forward rotation for each sector. The vector is used for the correct current measurement.
- Sector offset

  The `BLDC_Ctrl_Sector_Offset` parameter lets you specify a value for the offset angle to be added to the start angle of the first sector. This value must also be specified in the Sector offset property of the Block-Commutated PWM Out function block.

  The angle of the motor sectors is determined by the number of sectors, which is fixed to six in the Block-Commutated PWM Out function block. The fixed angle and the `BLDC_Ctrl_Sector_Offset` parameter result in fixed values for the `BLDC_Ctrl_Sector_<n>_End` parameters that must not be changed.

Specifying the type of control Via the `BLDC_Ctrl_Ctrl_mode_SW` parameter in the script file or in ControlDesk, you can specify the type of control. Refer to the following table:

BLDC_Ctrl_Ctrl_mode_SW Parameter Value	Description
1	Open loop
2	Current control
3	Speed control

EngineConfiguration Project: Simulating an Engine

Using the EngineExample Application

Use scenario

The EngineExample application servers to demonstrate the simulation of a 6-cylinder four-stroke piston engine.

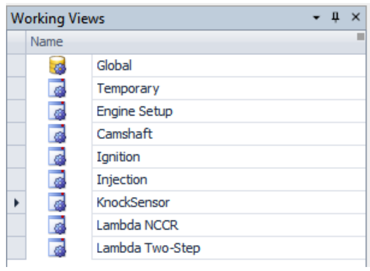
Features in focus

I/O functionality for engine simulation ConfigurationDesk I/O functionality is used for the following:

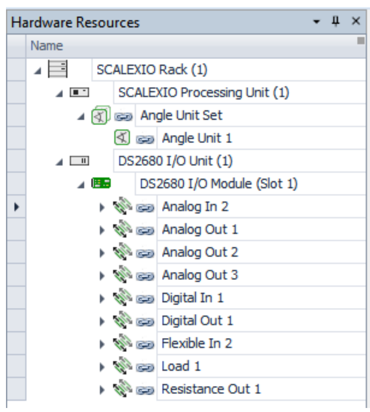
- Defining a virtual 6-cylinder engine
- Generating crankshaft and camshaft signals
- Measuring injection and ignition signals
- Generating knock signals
- Generating lambda signals

For more information on the basics of engine simulation and the involved ConfigurationDesk function block types, refer to Engine Simulation (ConfigurationDesk I/O Function Implementation Guide 📖).

Working views for different functionalities To focus on different areas of the engine simulation, the demo application offers different working views for you to open in the Signal Chain Browser.



Assigned hardware resources The assigned SCALEXIO hardware resources offer the required channel types and angle unit.

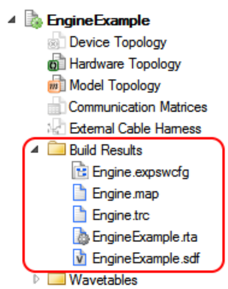


Tip

On the context menu of selected channels/angle units, you can use the Show – Select Assigned Function command to select and show the function blocks they are assigned to. In turn, you can use the Show – Select Assigned Hardware Channel command from the context menu of selected function blocks to select and show the channels/angle units assigned to them.

Simulink model The connected Engine Simulink model contains only the model port blocks for the model interface. It has no logical internal structure.

Build results Build results are already available in the Project Manager.



Actions and adjustments

You are not required to perform specific actions because the demo illustrates a finished ConfigurationDesk application.

However, you are free to adjust the configuration and, for example, start a new build process.

You can also use the demo as a basis for a real use scenario that requires engine simulation functionality. The model port blocks from the Simulink model can be used in a different model by using the Paste and Keep IDs command.

EngineControlDemo Project: Controlling an Engine

Using the EngineControl Application

Use scenario

The EngineControl application serves to demonstrate the control of a 6-cylinder four-stroke piston engine.

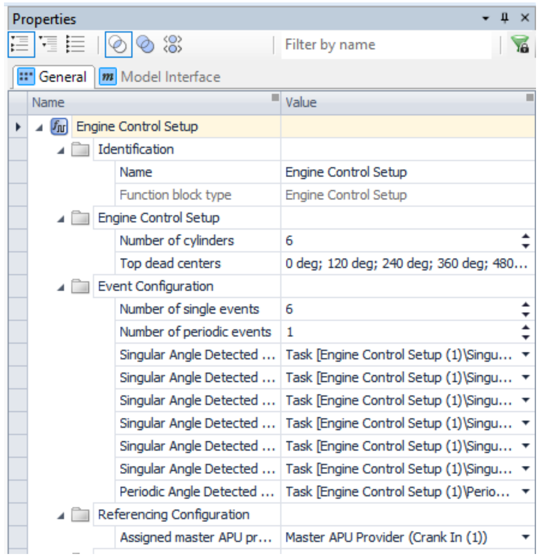
Summary of the main characteristics:

- Evaluating the crankshaft signal: Use of a crankshaft wheel with a tooth width of 6° and a gap width of 12°.
- Evaluating a single camshaft signal: Camshaft pulse from 120° to 140°.
- The length of an engine cycle is 720° (4-stroke engine).
- Generating 5 ignition and 5 injection pulses for each of the 6 cylinders.
- Measuring knock signals for each cylinder.

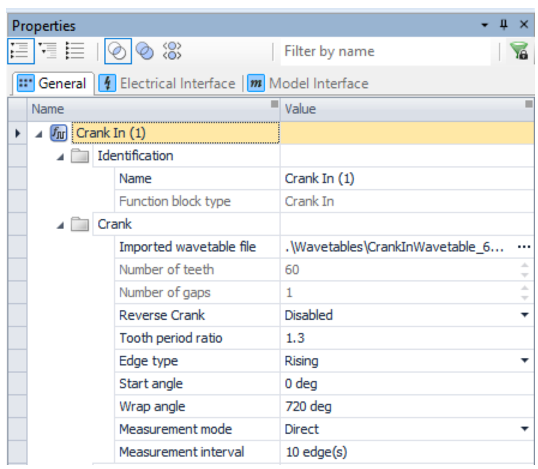
Features in focus

Signal chain for engine control functionality Together, the ConfigurationDesk engine control I/O functionality and the connected EngineControl Simulink model provide the signal chain for engine control:

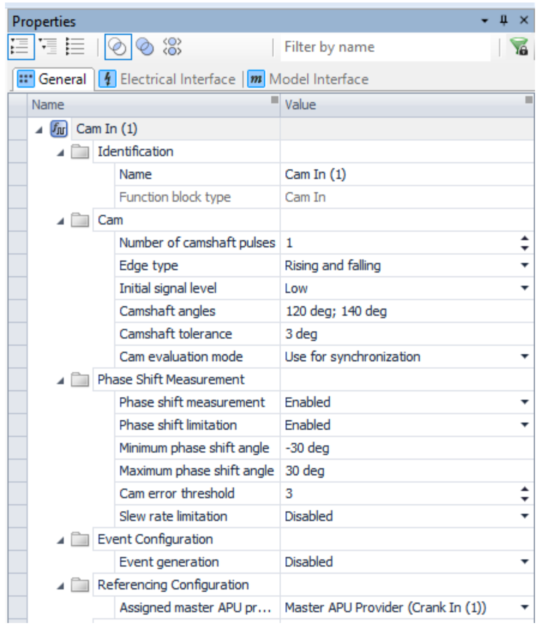
- The Engine Control Setup function block specifies basic characteristics of the engine, such as the number of cylinders. The function block works as a provider: Other function blocks can use it to obtain information on the characteristics of the specified engine. It provides, for example, the pulses for the Injection Out and Ignition Out function blocks.



- The Crank In function block measures the crankshaft of the engine and calculates the current position, speed and rotational direction. The specification of the crankshaft wheel (number of teeth and gaps, etc.) is based on wavetable files. The function block also serves as a master APU provider for the Engine Control Setup and the Cam In function blocks.



- The **Cam In** function block measures the phase shift angle between the camshaft and the coupled crankshaft.

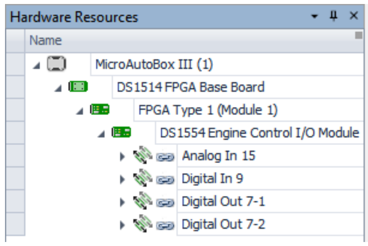


- Based on the top dead center angles configured in the **Engine Control Setup** block, the **Injection Out** and **Ignition Out** function blocks generate injection and ignition pulses for each cylinder of the engine.

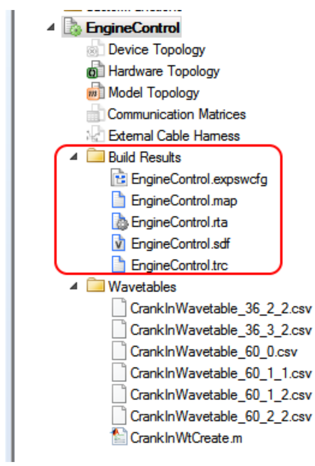
  For the runtime configuration of the ignition and injection pulses, a subsystem is assigned to each cylinder (Update_InjIgn_Cyl[1–6]). Each of these subsystems is executed once per engine cycle at a given angle before the top dead center (TDC) of the assigned cylinder.
- The **Knock In** function block measures and processes knock sensor signals. The result is provided to the behavior model and can be used, for example, to avoid/minimize pre-ignitions caused by improper ignition timing.
  - Within a subsystem executed before the TDC of each cylinder (Update_Knock_MW), start and end angles of the measurement performed for each cylinder are configured.
  - The read-out of the knock values is performed within a separate subsystem (Read_Knock_Data), which is executed immediately after the end of each cylinder-related measuring process.

For more information on ConfigurationDesk engine control I/O functionality, refer to Engine Control (ConfigurationDesk I/O Function Implementation Guide 📖).

Assigned hardware resources The assigned MicroAutoBox III hardware resources offer the required channel types.



Build results Build results are already available in the Project Manager.



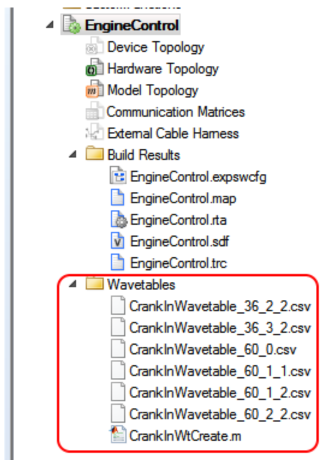
Actions and adjustments

You are not required to perform specific actions because the demo illustrates a finished ConfigurationDesk application.

However, you are free to adjust the configuration and, for example, start a new build process.

You can also use the demo as a basis for a real use scenario that requires engine control functionality.

Wavetable file generation In addition to the wavetable file used by the Crank In function block, the demo application provides a number of different wavetable file examples. It also provides a MATLAB script file (`CrankInWtCreate.m`) that lets you generate more wavetable files according to your requirements. Check the comments in the MATLAB script file for information.


EngineControl
Device Topology
Hardware Topology
Model Topology
Communication Matrices
External Cable Harness
Build Results
EngineControl.expswcfg
EngineControl.map
EngineControl.rta
EngineControl.sdf
EngineControl.trc
Wavetables
CrankInWavetable_36_2_2.csv
CrankInWavetable_36_3_2.csv
CrankInWavetable_60_0.csv
CrankInWavetable_60_1_1.csv
CrankInWavetable_60_1_2.csv
CrankInWavetable_60_2_2.csv
CrankInWtCreate.m

MPTurnlampDemo Project: Using a Multimodel, Multi-Processing-Unit Application

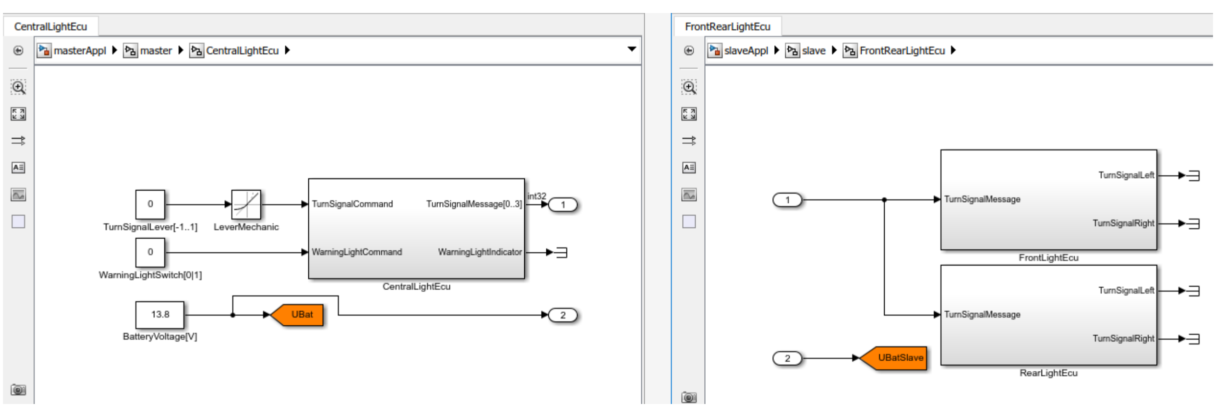
Using the turnlamp Application

Use scenario

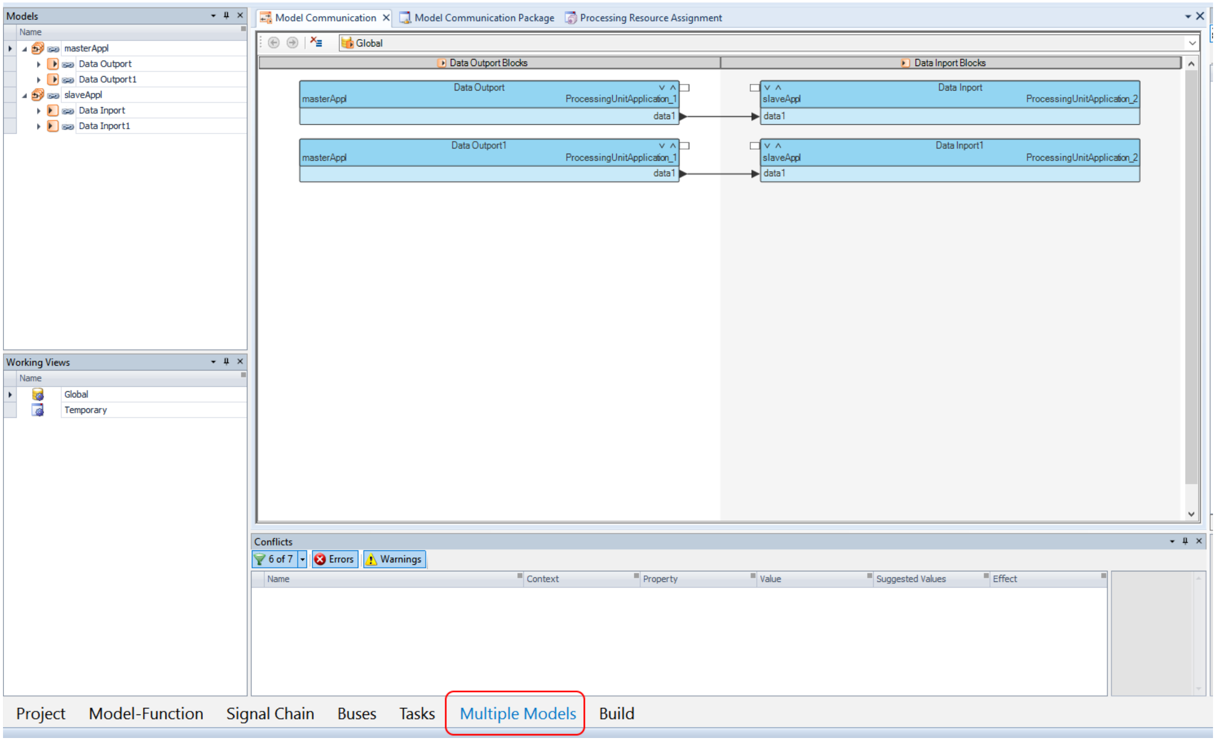
The turnlamp application is a multimodel, multi-processing-unit application that demonstrates the simulation of a turn-signal circuit that can be activated with a turn-signal lever and warning lights that can be switched on and off.

Features in focus

Model communication The application is connected to two Simulink models (masterAppl and slaveAppl) that simulate the behavior of three ECUs for controlling turn signals depending on a TurnSignalLever and a WarningLightSwitch.

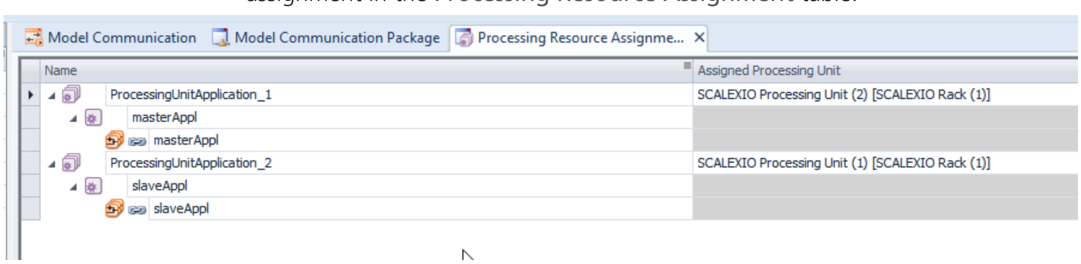


Model communication between the two models has been configured by mapping model port blocks from the ConfigurationDesk model interface of both models in ConfigurationDesk. The configuration of model communication can best be accessed in the Multiple Models view set.

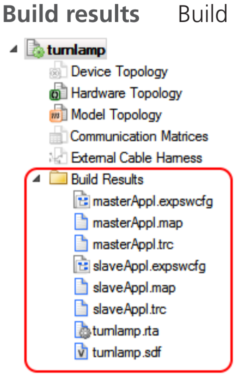


For more information on model communication, refer to Setting Up Model Communication (ConfigurationDesk Real-Time Implementation Guide 📖).

Multi-processing-unit application The application processes for the two models are assigned to different processing unit applications so that the real-time application can be executed on two processing units. You can access the assignment in the Processing Resource Assignment table.



Build results Build results are already available in the Project Manager.



Actions and adjustments

- If you have a dSPACE hardware system that provides multiple processing units, you can download the real-time application to it. For this purpose, the connection state of the ConfigurationDesk application must be **Matching platform connected**. For more information and instructions, refer to Managing Real-Time Hardware (ConfigurationDesk Real-Time Implementation Guide 📖).

  The processing units must be connected via IOCNET. Refer to Communication Network of a SCALEXIO System (SCALEXIO Hardware Installation and Configuration 📖).

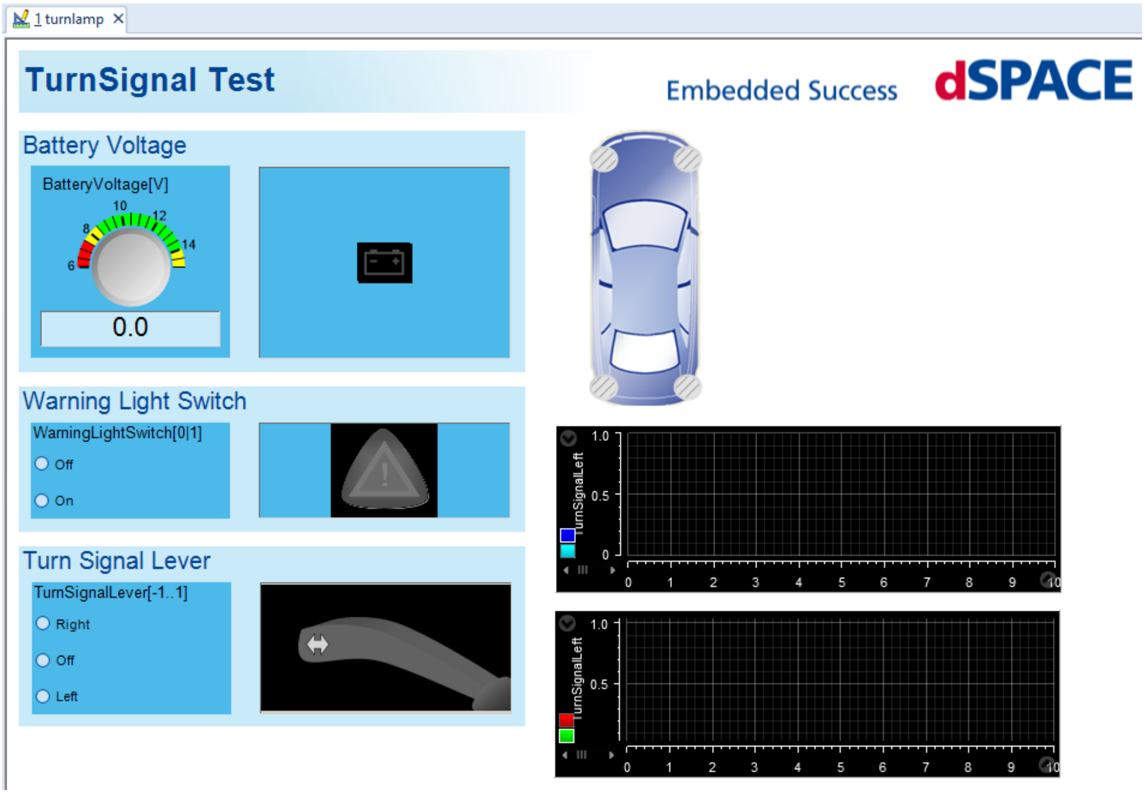
  The processing units must be assigned to the processing unit applications/application processes of the models. Refer to How to Assign Processing Units to Processing Unit Applications (ConfigurationDesk Real-Time Implementation Guide 📖).

  For information and instructions on downloading the real-time application to a matching platform, refer to Downloading and Executing Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 📖).
- If you downloaded the real-time application to a matching platform, you can start experimenting in ControlDesk. Your ConfigurationDesk installation contains a ControlDesk demo project that is based on this ConfigurationDesk demo project.

  The **CDNG_MPTurnlampDemo** is located in the same folder as the **MPTurnlampDemo** : `<Documents folder>\MPTurnlampDemo\`. It is also available in a ZIP archive, which is located in the `<RCP and HIL installation folder>\Demos\ConfigurationDesk\MPTurnlampDemo` folder. The files in the `Variable Descriptions` folder are the build results of the ConfigurationDesk project.

  Refer to Experimenting with a SCALEXIO System (SCALEXIO – Hardware and Software Overview 📖).

- You can control the simulation in the ControlDesk experiment.



- Use the Turn Signal Lever and the Warning Light Switch to control the turn signal and the warning lights.

  You can, for example, switch the turn-signal lever very quickly from left to right to provoke errors.
- Change the Battery Voltage to provoke malfunctions in certain edge cases.

  For example, lowering the voltage to <= 8.2 V causes the simulated receiver in the front and rear ECUs to become unstable, thus disturbing the timing of the generated turn signal.

FunctionalSafetyDemo Project: Example of Using MicroAutoBox III Functional Safety Features

Using the FuSaDemoApp Application

Use scenario

The FuSaDemoApp application serves to demonstrate I/O functionality for functional safety in combination with the MicroAutoBox III.

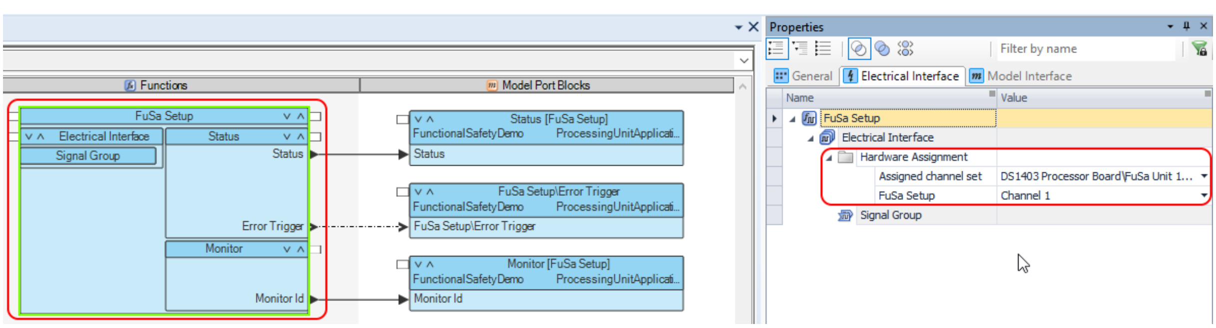
Features in focus

Functional safety with the MicroAutoBox III The MicroAutoBox III FuSa concept includes a set of functional safety (FuSa) functionalities. These let you implement elements of functional safety in a prototyping system.

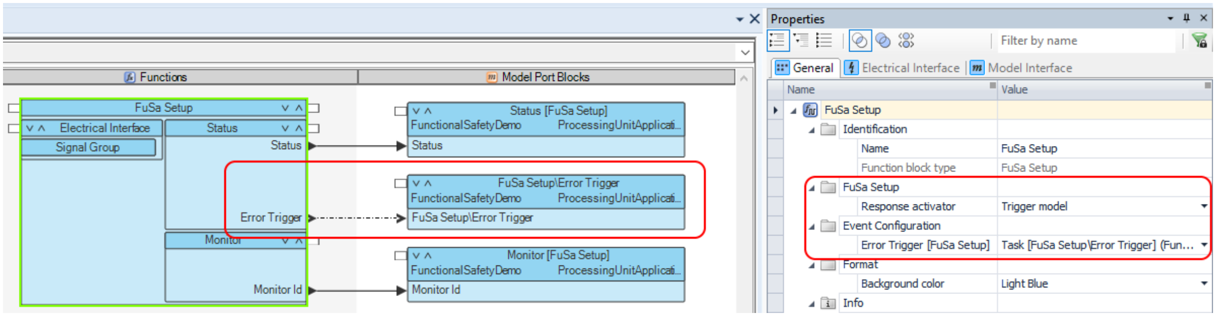
For more basic information, refer to Basics on Using FuSa with the MicroAutoBox III (ConfigurationDesk I/O Function Implementation Guide 🕮).

FuSa Setup The FuSa Setup function block provides the basic functionality for implementing functional safety in your system. The function block triggers basic error responses and lets you enable additional error responses.

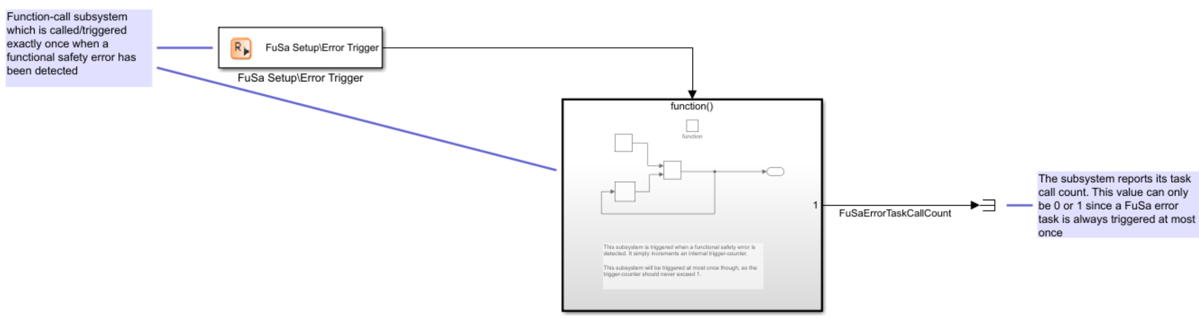
You require a FuSa Setup function block that is assigned to the FuSa unit of a MicroAutoBox III in a ConfigurationDesk application to activate and use the FuSa functionalities of the MicroAutoBox III.



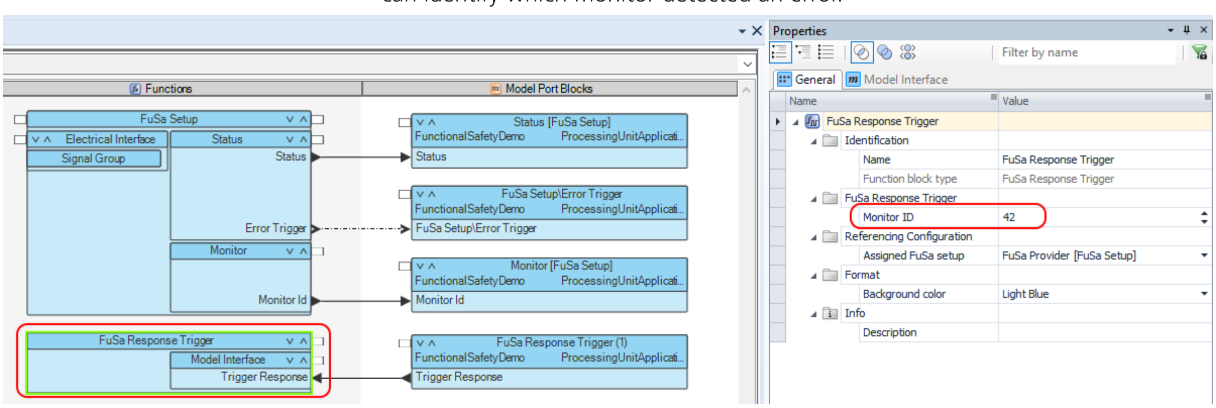
The FuSa Setup function block triggers error responses according to errors monitored by different FuSa function blocks that are assigned to it. As an additional error response, the FuSa Setup function block in the demo application is configured to trigger an I/O event in the behavior model.



In the connected FunctionalSafetyDemo Simulink model, the runnable function block that can be triggered by the I/O event is connected to a Function-Call Subsystem.

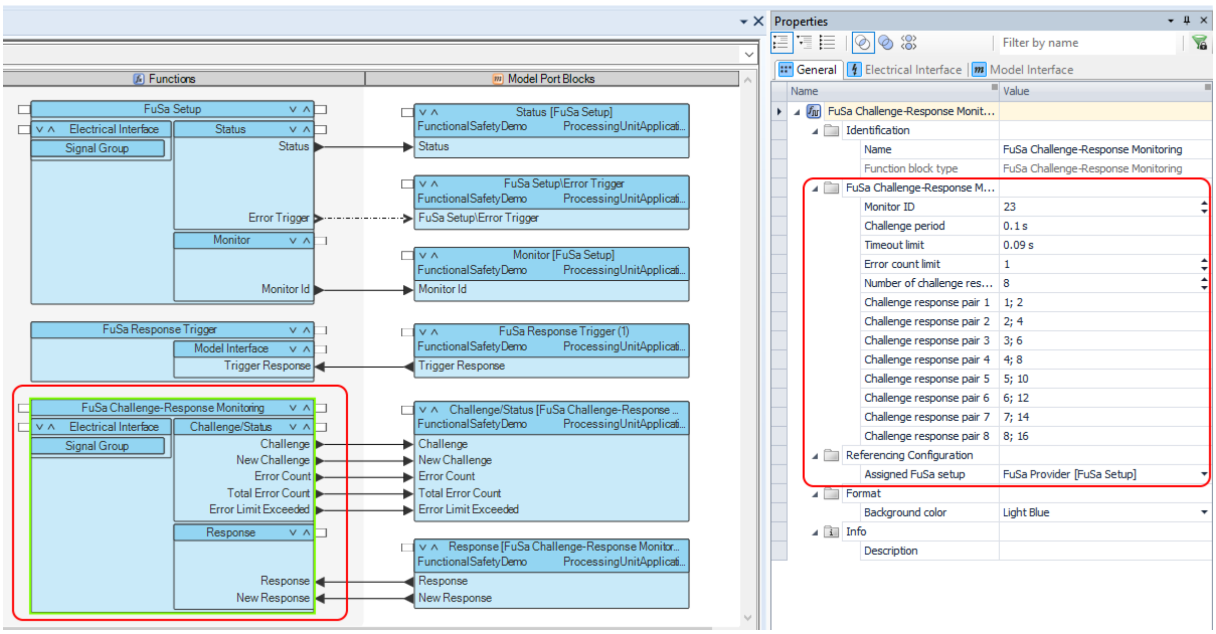


A FuSa Challenge-Response Monitoring and a FuSa Response Trigger function block are assigned to the FuSa Setup function block as monitors. Each monitor has a user-defined monitor ID so that the FuSa Setup function block can identify which monitor detected an error.



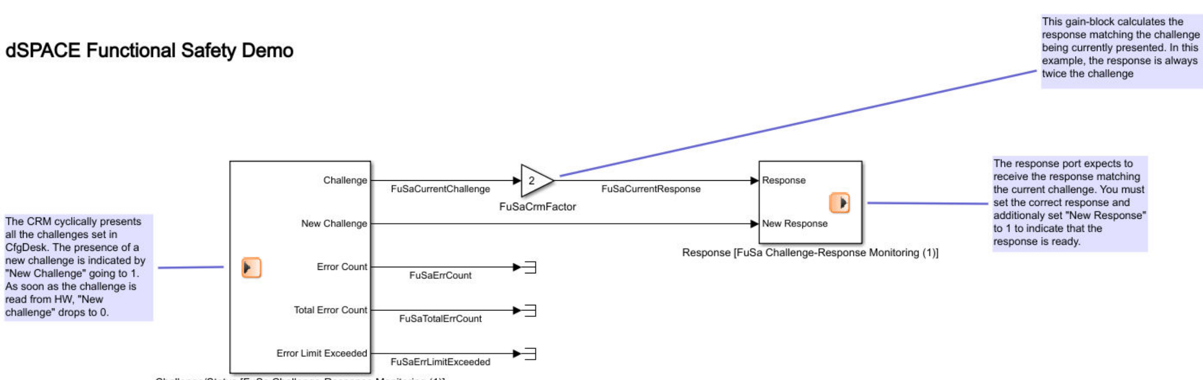
FuSa Challenge-Response Monitoring The FuSa Challenge-Response Monitoring function block works as a monitor based on the challenge and response principle. This means that value pairs consisting of challenge values and expected response values are configured. At run time, the challenge response monitor provides challenge values to the behavior model and expects the according response values within specified timing constraints.

In the demo application, eight challenge-response pairs are configured. New challenges are provided to the behavior model every 0.1 s (Challenge period) and the behavior model has to return the response value within 0.09 s (**Timeout limit**). Because the **Error count limit** is set to 1 the function block will trigger the FuSa response at the **FuSa Setup** function block as soon as *more* than one error has occurred. An error is typically either a wrong response or a response which has been sent too late. The errors need to be *consecutive*. If after a single error a correct response is given, the internal error counter is reset to 0.



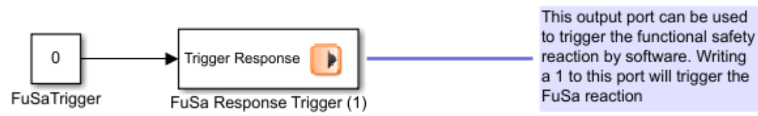
For more information on configuring FuSa challenge-response monitoring, refer to Configuring the Basic Functionality (FuSa Challenge-Response Monitoring) (ConfigurationDesk I/O Function Implementation Guide 🕮).

In the connected **FunctionalSafetyDemo** Simulink model, the current challenge value is doubled and then returned to match the expected response.



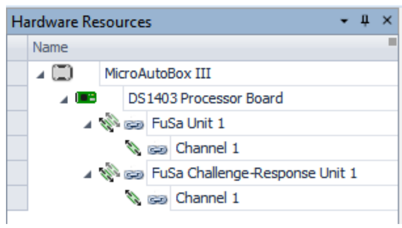
FuSa Response Trigger The **FuSa Response Trigger** function block works as a monitor that can be triggered directly from within the behavior model. In the demo application, no meaningful logic is connected to the function block in the Simulink model. The connected value is simply set to 0 so that no error response is triggered. You can however use ControlDesk to manipulate the

FuSaTrigger value from an external host PC to demonstrate that this manual trigger is working correctly.

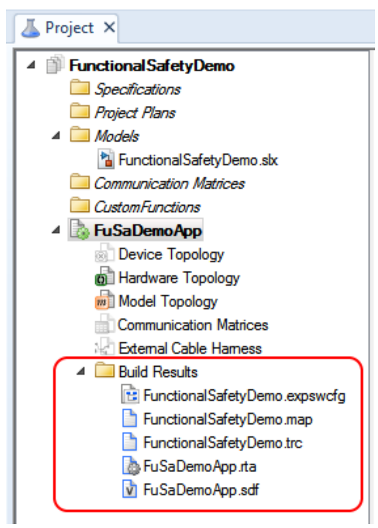


For more information on the FuSa Response Trigger function block, refer to FuSa Response Trigger (ConfigurationDesk I/O Function Implementation Guide 📖).

Assigned hardware resources The assigned MicroAutoBox III hardware resources offer the required FuSa Unit and FuSa Challenge-Response Unit channels.



Build results Build results are already available in the Project Manager.



Actions and adjustments

You are not required to perform specific actions because the demo illustrates a finished ConfigurationDesk application.

However, you are free to adjust the configuration and, for example, start a new build process.

You can also use the demo as a basis for a real use scenario that requires functional safety functionality.

If you have a MicroAutoBox III, you can download the real-time application to it. For this purpose, the connection state of your ConfigurationDesk application must be **Matching platform connected**. For more information and instructions, refer to Managing Real-Time Hardware (ConfigurationDesk Real-Time Implementation Guide 🕮).

For information and instructions on downloading the real-time application to a matching platform, refer to Downloading and Executing Real-Time Applications (ConfigurationDesk Real-Time Implementation Guide 🕮).

If you downloaded the real-time application to a matching platform, you can start experimenting in ControlDesk. Refer to Software for Experimenting with the MicroAutoBox III (MicroAutoBox III - Hardware and Software Overview 🕮).

WheelspeedOutDemo Project: Simulating an Active Wheel Speed Sensor

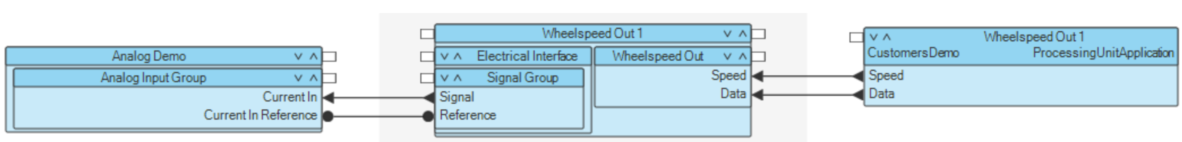
Using the WheelspeedDemoAppl Application

Use scenario

The WheelspeedDemoAppl application serves to demonstrate a signal chain with a Wheelspeed Out function block that simulates the signals provided by an active wheel speed sensor.

Features in focus

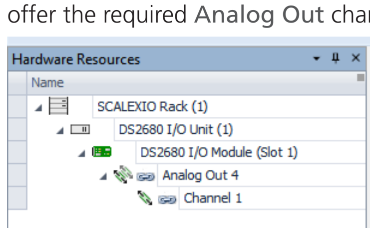
Wheelspeed Out I/O functionality The signal chain of the WheelspeedDemoAppl application contains a Wheelspeed Out function block.



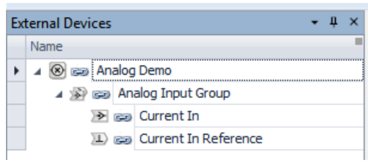
For information on the Wheelspeed Out function block, refer to Wheelspeed Out (ConfigurationDesk I/O Function Implementation Guide 📖).

Simulink model For information on the connected Simulink behavior model, refer to Demo Model of a Wheelspeed Out Interface (Model Interface Package for Simulink - Modeling Guide 📖).

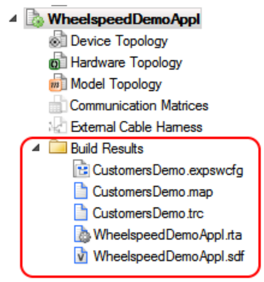
Assigned hardware resources The assigned SCALEXIO hardware resources offer the required Analog Out channel type.



External device interface The external device interface contains a device whose ports are mapped to the signal ports of the function blocks.



Build results Build results are already available in the Project Manager.



Actions and adjustments

You are not required to perform specific actions because the demo illustrates a finished ConfigurationDesk application.

However, you are free to adjust the configuration and, for example, start a new build process.

You can also use the demo as a basis for a real use scenario that requires active wheel speed sensor functionality.