

Model and Sensor Interface Blockset

Manual

For Model and Sensor Interface Blockset 1.1

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.

© 2020 - 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
Basics and Instructions	11
Basics of the Model and Sensor Interface Blockset.....	12
Basics of the Blockset and Connected Systems.....	12
Features of the Model and Sensor Interface Blockset.....	14
Blockset Supported Platforms.....	16
Overview of the Blocks of the Model and Sensor Interface Blockset.....	16
Migration from Previous Releases of the Model and Sensor Interface Blockset.....	19
Simulation Data Mathematical Principles.....	22
Coordinate System Used in MotionDesk.....	22
Vertex in a 3-D Space.....	23
Homogeneous Transformation.....	23
Euler Angles.....	25
Cardan Roll, Pitch, and Yaw Angles.....	25
3-D Objects in a 3-D Space.....	27
Working with the Model and Sensor Interface Blockset.....	28
Adapting Simulink Models.....	28
Workflow to Adapt a Simulation Model with the Model and Sensor Interface Blockset.....	29
How to Access the Model and Sensor Interface Blockset.....	32
How to Prepare the ASM Model with the Animation Interface Subsystem.....	36
How to Connect a Simulink Model to a Sensor Simulation System or Application.....	39
Connecting Simulation Platforms Using Ethernet Functions.....	42
How to Collect Status Information using the System Status block.....	43
How to Configure Sensor Failures for a Connected Sensor.....	44
How to Configure Sensor Feedback from the Environment Sensor Interface Unit.....	46
Configuring the Simulation Data Objects.....	48
Working with the Simulation Data Objects Block.....	49
How to Configure the Simulation Data Objects in a Simulation Model.....	51
How to Configure Multiple Transformation Objects.....	53

How to Import and Export Simulation Data Objects.....	56
Simulation Data Objects Block Automation API.....	59
Configuring Network Settings for Blockset Ethernet Connections.....	64
Network Basics for the Model and Sensor Interface Blockset.....	64
How to Configure the Windows Network Settings for a Multi-PC Solution.....	66
How to Configure the Network Settings in MotionDesk.....	67
Connecting Simulation Platforms to Sensor Simulation Systems.....	68
How to Connect a SCALEXIO Platform to a Single MotionDesk Observer.....	69
How to Connect a SCALEXIO Platform to Multiple MotionDesk Observers.....	73
How to Connect a VEOS Platform to a Single MotionDesk Observer.....	78
How to Connect a Simulation Platform to a SensorSim Application.....	81
Reference Information	89
Model and Sensor Interface Blockset.....	90
General Information on the Model and Sensor Interface Blockset.....	90
Model and Sensor Interface Blockset Library Overview.....	91
Connection Settings Block.....	93
Block Description (dsmsi_connection_settings).....	93
Block Parameters Page (dsmsi_connection_settings).....	98
Simulation Data Objects Block.....	100
Block Description (dsmsi_simulation_data_objects).....	100
Block Parameters Page (dsmsi_simulation_data_objects).....	104
System Status Block.....	115
Block Description (dsmsi_system_status).....	116
Block Parameters Page (dsmsi_system_status).....	118
Sensor Failure Block.....	120
Block Description (dsmsi_sensor_failure).....	120
Block Parameters Page (dsmsi_sensor_failure).....	122
Sensor Feedback Block.....	124
Block Description (dsmsi_sensor_feedback).....	125
Block Parameters Page (dsmsi_sensor_feedback).....	126
Model and Sensor Interface Blockset Demo Library.....	128
Demo Library Overview (dsmsi_demolib).....	128
Coordinate System Demo for SCALEXIO.....	129
Automotive Demo for SCALEXIO.....	132

Limitations 137

 Limitations When Using the Model and Sensor Interface Blockset..... 137

Index 141

About This Document

Contents

This document introduces you to the Model and Sensor Interface Blockset for the calculation and transmission of simulation data in a simulation to MotionDesk and to Sensor Simulation connected systems and applications. For example, to create realistic sensor raw data to be inserted into the sensor hardware or to the control unit.

The Model and Sensor Interface Blockset (MSI Blockset) can be used to simulate with sensors in a software-in-the-loop (SIL) or in a hardware-in-the-loop (HIL) test environment.

The documentation provides the information required for adapting the simulation models and connecting the systems using ConfigurationDesk Ethernet functions or blocks from the VEOS Ethernet Blockset Solution.

Target group

Engineers who want to stream simulation data to the MotionDesk PC and Sensor Simulation systems and applications must read this document.

Required knowledge



Basic knowledge in handling the dSPACE software and MotionDesk for scene creation and animation is assumed.




Knowledge of hardware-in-the-loop (HIL) real-time simulation, offline simulation using VEOS, and implementing models in MATLAB/Simulink is advised.

Familiarity with Sensor Simulation and the types of sensors used in ADAS/AD systems is also advised.

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.

Symbol	Description
 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

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.

Basics and Instructions

Where to go from here

Information in this section

Basics of the Model and Sensor Interface Blockset.....	12
Provides basic information on the Model and Sensor Interface Blockset for the calculation and transmission of the simulation, status, feedback, and failure data to and from Sensor Simulation systems and applications.	
Simulation Data Mathematical Principles.....	22
To prepare simulation models, you must know the mathematical principles for the movement and transformation of objects used by the simulation data for visualization in MotionDesk and for Sensor Simulation.	
Working with the Model and Sensor Interface Blockset.....	28
You can learn to use the Model and Sensor Interface Blockset to calculate and stream the data for sensor simulation.	

Basics of the Model and Sensor Interface Blockset

Introduction	Provides basic information on the Model and Sensor Interface Blockset for the calculation and transmission of the simulation, status, feedback, and failure data to and from Sensor Simulation systems and applications.
---------------------	--

Where to go from here

Information in this section

Basics of the Blockset and Connected Systems.....	12
dSPACE provides the Model and Sensor Interface Blockset to build Simulink models to calculate and handle the transmission of simulation, status, feedback, and failure data between the simulation and Sensor Simulation connected systems and applications.	
Features of the Model and Sensor Interface Blockset.....	14
You learn the basic features of the Model and Sensor Interface Blockset and connections to Sensor Simulation systems and applications.	
Blockset Supported Platforms.....	16
The Model and Sensor Interface Blockset is supported on specific dSPACE simulation platforms.	
Overview of the Blocks of the Model and Sensor Interface Blockset.....	16
You can use the blocks in the Model and Sensor Interface Blockset to specify the simulation data objects to transmit, the systems and applications to which to connect, and the feedback information you want to receive.	
Migration from Previous Releases of the Model and Sensor Interface Blockset.....	19
If you worked with Sensor Simulation using the Model and Sensor Interface Solution or a previous releases of the Model and Sensor Interface Blockset, you must migrate the simulation models to use the blocks of the latest release.	

Basics of the Blockset and Connected Systems

Introduction	dSPACE provides the Model and Sensor Interface Blockset to build Simulink models to calculate and handle the transmission of the simulation, status, feedback, and failure data between the simulation and Sensor Simulation connected systems and applications.
---------------------	--

Connected Systems	You must use the Model and Sensor Interface Blockset for Sensor Simulation.
--------------------------	---

The blockset calculates and handles the simulation, status, feedback, and data for sensor failures between the simulation and the connected systems and applications. For example, you can connect, MotionDesk, the SensorSim application, and the Environment Sensor Interface Unit.

MotionDesk visualizes the simulation and the SensorSim application creates the sensor raw data.

The Environment Sensor Interface Unit separates the sensor raw data for each of the sensors from the SensorSim application and inserts it into the relevant sensor hardware.

You can also send data to simulate sensor failures to each of the sensors connected to the Environment Sensor Interface Unit. The blockset can also receive feedback information from the Environment Sensor Interface Unit and status information from the other connected systems or applications.

New features

- TX rate can be controlled through an inport signal in the Connection Settings block.
- Multiple inlined objects for a single transformation object can be specified in the Simulation Data Objects block.
- Sensor sphere support using a position simulation data object type.
- Improvements to the automation API to support multiple objects.
- Block priorities are implemented that define the execution order.
- Improved handling of connected signals and error detection.
- Improved user interface control and error logging.
- Improved demos.

The simulation object data includes the following object types:

- **Transformation:** To transmit the translation, rotation, and scaling vectors of 3-D objects on a 3-D plane. For example, a car chassis and its four tires.
- **Signal:** To transmit instrument signals, for example, speed, acceleration, and engine revolutions.
- **Position:** To display the contact points of the sensors in the environment, for example, the 3-D point cloud sensor points.
- **Position and Scale:** To display the position and scale of a 3-D object, for example, the 3-D arrows that represent the active forces arrows at a point on the 3-D Object.

You must also connect Ethernet functions for the communication with the connected systems and applications.

In this document, Ethernet functions refer to the following, depending on the simulation platform used:

- **ConfigurationDesk:** Ethernet Setup and TCP functions for a SCALEXIO platform.
- **VEOS Ethernet Blockset Solution:** Ethernet TCP/IP Server and Client blocks for VEOS.

You can learn to use the Model and Sensor Interface Blockset to adapt Simulink models to calculate and stream simulation data in [Adapting Simulink Models](#) on page 28.

Related topics

Basics

[Working with the Model and Sensor Interface Blockset.....](#) 28

Features of the Model and Sensor Interface Blockset

Introduction

You learn the basic features of the Model and Sensor Interface Blockset and connections to Sensor Simulation systems and applications.

Overview

The Model and Sensor Interface Blockset provides you with the blocks required to connect and handle the transmission of the simulation, status, feedback, and failure data to and from MotionDesk, the SensorSim application and to the Environment Sensor Interface Unit.

Blockset features

The blockset has the following features:

- **Communicates on TCP/IP networks:** It has no hardware dependencies but connects using Ethernet functions. For more information, refer to [Basics of the Blockset and Connected Systems](#) on page 12..
- **Sends simulation data:** Sends simulation and signal data to MotionDesk, SensorSim applications, and the Environment Sensor Interface Unit.
- **Supports the creation of multiple objects for a specific transformation object,** for example, for multiple fellows in a simulation.
- **Supports kinematic chains:** Kinematic chains of objects can be defined in a parent-child relationship within one Simulation Data Objects block. Kinematic chains can also be defined by connecting together Simulation Data Objects blocks, whereby the output of one is the input to another.
- **Sends failure data:** Sends failure data to the Environment Sensor Interface Unit (ESI Unit). For example, you can implement a pixel error with a specific pixel position (x, y) and color value, such as 0 for black.
- **Reads sensor feedback data:** Reads feedback data back from the Environment Sensor Interface Unit (ESI Unit) on a connected sensor, for example, the exposure time of a camera sensor.
- **Reads status information:** Reads status information from MotionDesk, SensorSim applications, and the Environment Sensor Interface Unit (ESI Unit), for example, frame information and latency.

Blockset data transmission and directions

The connected systems and applications support the following data and directions through the Model and Sensor Interface Blockset.

MotionDesk The blockset transmits simulation data to be visualized in one or more MotionDesk PC instances as observers.

System status information can also be retrieved from MotionDesk.

Data	Direction
Simulation data	Transmits: Tx
System status	Receives: Rx

SensorSim application The blockset transmits simulation data to the SensorSim application, which calculates the sensor data. Several SensorSim application instances can be connected to one application.

System status information can also be retrieved from the SensorSim application.

Data	Direction
Simulation data	Transmits: Tx
System status	Receives: Rx

Environment Sensor Interface Unit (ESI Unit) The blockset streams sensor data and information from the simulation model to the Environment Sensor Interface Unit (ESI Unit), which splits and synchronizes the sensor data to be delivered to each connected sensor hardware in the architecture.

An Environment Sensor Interface Unit plug-on device (POD) that connects via an HDMI connection can also be connected to deliver data to camera sensors behind the sensor front end, for example, to the camera image processor.

The interface unit can also receive feedback and status information from the connected sensors, for example, camera and lidar sensors and provide the information to the blockset.

Sensor failures can also be enabled and disabled for sensors connected to the Environment Sensor Interface Unit.

Data	Direction
Sensor failures	Transmits: Tx
Sensor feedback	Receives: Rx
System status	Receives: Rx

Related topics**Basics**

Adapting Simulink Models.....	28
Basics of the Blockset and Connected Systems.....	12

Blockset Supported Platforms

Introduction

The Model and Sensor Interface Blockset is supported on specific dSPACE simulation platforms.

Supported platforms

The Model and Sensor Interface Blockset supports the following platforms that support Ethernet access:

- Simulink

Note

The blockset supports Normal and Accelerator Simulink modes

- SCALEXIO Systems
- VEOS and VEOS on Linux

Note

The Model and Sensor Interface Blockset is not supported on RTI platforms, for example, DS1006, DS1007, MicroLabBox, and MicroAutoBox II. For more information on the limitations, refer to [Limitations When Using the Model and Sensor Interface Blockset](#) on page 137

Related topics

Basics

[Introduction to Sensor Simulation \(Sensor Simulation Overview !\[\]\(e3f255517d37bb309a3a931ec4849e6a_img.jpg\)](#))

HowTos

[Connecting Simulation Platforms Using Ethernet Functions..... 42](#)

References

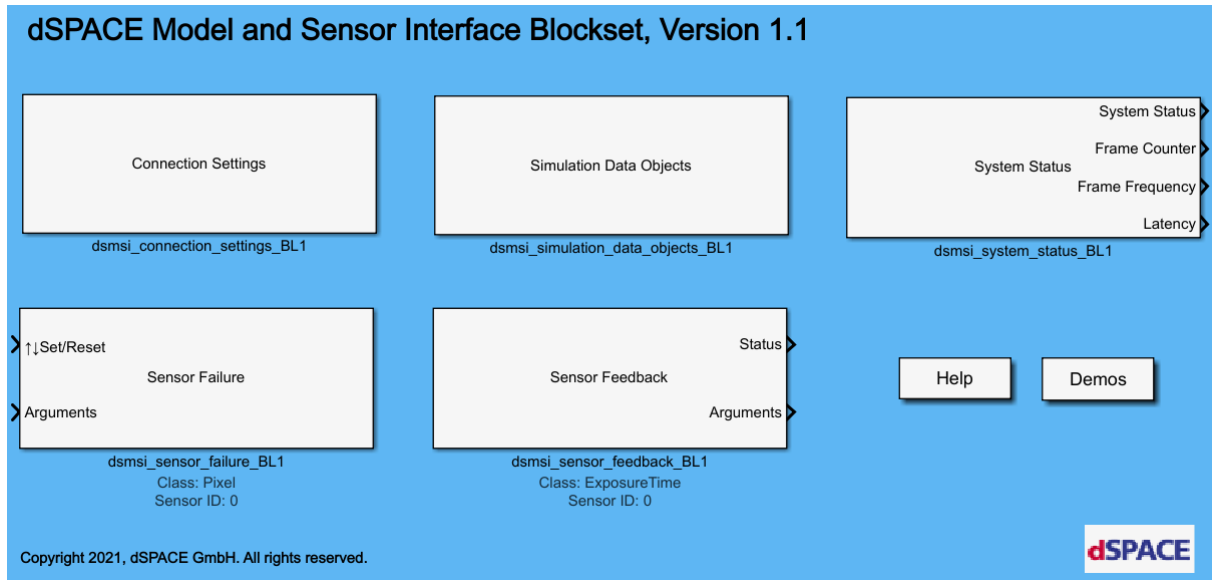
[Model and Sensor Interface Blockset..... 90](#)

Overview of the Blocks of the Model and Sensor Interface Blockset

Introduction

You can use the blocks in the Model and Sensor Interface Blockset to specify the simulation data objects to transmit, the systems and applications to which to connect, and the feedback information you want to receive.

Illustration



Simulink blocks

The following blocks are available in the Model and Sensor Interface Blockset. The blocks functions are described as follows:

Connection Settings block (dsmsi_connection_settings) The Connection Settings block handles the connection between the simulation model and the Sensor Simulation systems and applications. You add a Connection Settings block for each system, for example, MotionDesk, a SensorSim application, and an Environment Sensor Interface Unit (ESI Unit).

The simulation communicates through the Model and Sensor Interface Blockset and Ethernet functions with Sensor Simulation systems and applications. The data prepared by the block is transmitted by the Ethernet functions in data streams of sequenced data elements. For more information on the Ethernet functions, refer to [Basics of the Blockset and Connected Systems](#) on page 12. For information on the block, refer to [Connection Settings Block](#) on page 93.

Simulation Data Objects block (dsmsi_simulation_data_objects) The Simulation Data Objects stores the calculated data for the objects defined in the simulation, for example, the motion transformation data for the 3-D objects and sensor points, or signal, and instrument data. The Connection Settings block prepares the data generated by the Simulation Data Objects for transmission by means of the Ethernet functions to the connected systems and applications.

For each of the moving objects in the simulation, simulation data objects can be defined in kinematic chains in single object blocks using a parent-child relationship. You can also specify multiple objects for each transformation object in the Simulation Data Objects blocks, for example, for multiple fellows in a simulation.

Note

MotionDesk does not support kinematic chains deeper than parent and child relations.

You can learn how to define the Simulation Data Objects block in [Simulation Data Objects Block](#) on page 100.

System Status block (dsmsi_system_status) The System Status block connects to a Connection Settings block to output status information provided by the connected system or application.

You define the connection between the sensor data inputs and outputs on the connected system or application, and a single element in one of the block output vectors in a channel table.

The following information can be provided by the status block:

- **System Status:** Outputs the status of the connected system.
- **Frame Counter:** Outputs the number of received or transmitted frames per channel.
- **Frame Frequency:** Outputs the frame frequency of the incoming data per channel.
- **Latency:** Outputs the calculated time taken in seconds to transmit the data from the blockset and receive feedback data from the connected system or application.

You can learn how the System Status block provides status and statistics about the connected applications in [System Status Block](#) on page 115.

Sensor Failure block (dsmsi_sensor_failure) The Sensor Failure block is used to enable and disable sensor failures and specify the arguments for the failures on specific sensors connected to the Environment Sensor Interface. For example, you can specify a pixel error with a specific pixel position (x, y) and color value, such as 0 for black.

The block supports static and dynamic failure arguments. The sensor failure arguments, which can change dynamically while the simulation is running can be provided through the Arguments input. The preset type and dimensions remain static.

You can learn how to use the Sensor Failure block in [Sensor Failure Block](#) on page 120.

Sensor Feedback block (dsmsi_sensor_feedback) Sensor feedback data is sent from all the connected sensors through the Environment Sensor Interface Unit to the connected systems and applications.

The Sensor Feedback is configured with a Sensor ID. The block filters the data for this specific sensor and forwards the feedback data via the Arguments output to the model for further processing.

For example, a control unit changes the exposure time of the sensor. The Environment Sensor Interface Unit detects this feedback parameter and forwards the data to all connected systems.

You can learn how to use the Sensor Feedback block in [Sensor Feedback Block](#) on page 124.

dSPACE MSI Demo Library Coordinate System Demo for SCALEXIO

This demo model shows the influence of the transformation order on the position of the corresponding 3-D movable objects in the Model and Sensor Interface Blockset.

A Simulation Data Objects block is included in the demo in which the transformation order is modeled.

The demo includes a preconfigured Connection Settings for block MotionDesk connected to TCP model port blocks for the SCALEXIO platform.

For more information, refer to [Model and Sensor Interface Blockset Demo Library](#) on page 128.

Related topics

References

Connection Settings Block.....	93
Sensor Failure Block.....	120
Sensor Feedback Block.....	124
Simulation Data Objects Block.....	100
System Status Block.....	115

Migration from Previous Releases of the Model and Sensor Interface Blockset

Introduction

If you worked with Sensor Simulation using the Model and Sensor Interface Solution or a previous release of the Model and Sensor Interface Blockset, you must migrate the simulation models to use the blocks of the latest release.

Migrating from previous releases

When you open a model that was created with previous releases of the Model and Sensor Interface Blockset as of Version 1.0 in Release 20-B, the model is migrated automatically to use the blocks of the latest release.

Messages are displayed in the MATLAB Command Window and in the migration log to confirm that the migration was successful or if there are failures.

Tx rate The Tx rate: n value of the Connection Settings block is migrated to the Tx rate parameter.

Tx configuration - Use input port: If migrating from a previous release without the function to provide the transmission rate through an input port, Use input port is not selected after migration of the Connection Settings block. To specify that the transmission rate is provided as a signal from the input port while the simulation is running, you can select this parameter in the block dialog

after migration. For more information, refer to [How to Connect a Simulink Model to a Sensor Simulation System or Application](#) on page 39.

Migration log If the migration completes successfully, a message is displayed in the MATLAB Command Window. The migration log is created in the same folder as the simulation model with the following naming convention: `dsmsi_<model_name>_migration_log.txt`.

The migration log contains the version information before and after the migration of each block of the Model and Sensor Interface Blockset used in the model.

The migration file also contains the confirmation of the migration of the Tx rate parameter.

Migrating from the Model and Sensor Interface Solution

When you open a model that contains blocks of the **Model and Sensor Interface Solution** prior to Release 2020-B, a warning is displayed in the MATLAB Command Window indicating that you must migrate the model to use the blocks of the installed product version of the **Model and Sensor Interface Blockset**. The message contains information and a link to start the migration of the model.

In the MATLAB Command Window, messages on the start and completion of the migration are displayed. Other important information is also shown, for example, the migration of the Tx-time parameter of the **Global** block to the Tx rate parameter of the **Connection Settings** block.

Note

You must save the model after the migration is completed.

Detailed information is provided in the migration log file which has the following naming convention: `dsmsi_<model name>_migration_log.txt`. The log file is stored in the same folder as the simulation model.

Global blocks The Global blocks from the **Model and Sensor Interface Solution** are migrated to the **Connection Settings** blocks of the **Model and Sensor Interface Blockset**.

The following fields in the Global block are migrated as follows:

Global block field	Connection Settings block field
Application	Connect to
Reuse	Send objects
Tx-time	Tx rate: The value is not migrated. Tx rate: is 1 by default. You must specify the Tx rate: value after migration.

Note

You do not specify primary and secondary relationships for the Connection Settings blocks. For secondary Global blocks, the name of the source Global block is not shown after the migration.

If the Global block was a primary Global block, Send objects is selected in the migrated Connection Settings block. If it was a secondary Global block and Reuse was not selected, the Send objects option is not selected after the migration.

Object blocks The Object blocks of the Model and Sensor Interface Blockset Solution, which was available prior to the release of the Model and Sensor Interface Blockset, are migrated to Simulation Data Objects blocks of the Model and Sensor Interface Blockset.

The object types are migrated as follows:

Solution object type	Model and Sensor Interface Blockset object type
Object	Transformation
Point	Position
3-D Arrow	Position and scale

Simulation Data Mathematical Principles

Introduction

To prepare simulation models, you must know the mathematical principles for the movement and transformation of objects used by the simulation data for visualization in MotionDesk and for Sensor Simulation.

Where to go from here

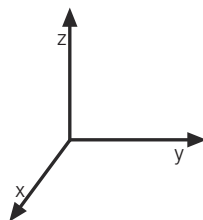
Information in this section

Coordinate System Used in MotionDesk.....	22
Provides information on the coordinate system that is used by MotionDesk.	
Vertex in a 3-D Space.....	23
A position of a point (vertex) can be described using a translation vector.	
Homogeneous Transformation.....	23
Adding a fourth dimension to the 3-D vector simplifies the calculation of 3-D operations, such as rotation, translation, and scaling.	
Euler Angles.....	25
The Euler angles are the three angles of rotation of an object around a fixed axis.	
Cardan Roll, Pitch, and Yaw Angles.....	25
The cardan angles of roll, pitch, and yaw angles are three angles that describe the orientation of a body in a three-dimensional space.	
3-D Objects in a 3-D Space.....	27
In contrast to a vertex, a 3-D object has an expansion.	

Coordinate System Used in MotionDesk

Introduction

MotionDesk works with a right-hand Cartesian coordinate system. The following illustration shows the coordinate system as used in MotionDesk:



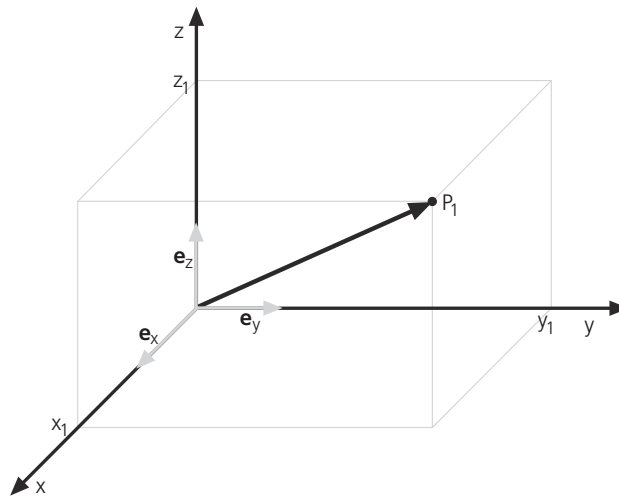
Vertex in a 3-D Space

Introduction

A position of a point (vertex) can be described using a translation vector.

Translation vector

To describe the position of a point (vertex) in a right-hand Cartesian coordinate system, three mutually orthogonal unit vectors (\vec{e}_x , \vec{e}_y , \vec{e}_z) are used, refer to the following illustration.



A point in space is referenced by a vector:

$$\mathbf{P} = \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} = x_1 \vec{e}_x + y_1 \vec{e}_y + z_1 \vec{e}_z$$

The vector describes the distance between the origin and a point in the coordinate system. In MotionDesk, this vector is called the translation vector.

Homogeneous Transformation

Introduction

Adding a fourth dimension to the 3-D vector simplifies the calculation of 3-D operations, such as rotation, translation, and scaling. The transformations can therefore be calculated by multiplication with a 4 x 4 matrix.

Translation

The following matrix moves an object in the x-, y-, z- directions:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Scaling

The following matrix scales an object in the x-, y-, z- directions:

$$\mathbf{S} = \begin{pmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation

In a three-dimensional coordinate system, three different rotations are possible. The different rotation matrices are shown below.

x-axis The following matrix rotates an object around its x-axis:

$$\mathbf{R}(x, \alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

y-axis The following matrix rotates an object around its y-axis:

$$\mathbf{R}(y, \alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

z-axis The following matrix rotates an object around its z-axis:

$$\mathbf{R}(z, \alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The equations above are for rotation around only one axis. If an object is rotated around two or three axes, the rotations must be done one after the other. The most common descriptions for rotation based on three independent variables are Euler angles (mechanics, gyroscopics) and roll, pitch and yaw angles (nautics and aeronautics).

Related topics

Basics

Cardan Roll, Pitch, and Yaw Angles.....	25
Euler Angles.....	25

Euler Angles

Introduction

The Euler angles are the three angles of rotation of an object around a fixed axis.

Euler angles

Using the Euler angles, a 3-D object is rotated around its axes in the following manner:

1. Rotation around the z-axis at angle ψ
2. Rotation around the y-axis at angle θ
3. Rotation around the actual z-axis at angle ϕ

The rotations are expressed in the following equation:

$$R_{\text{Euler}} = R_{\text{Euler}}(\phi, \theta, \psi) = R_{(z, \phi)} R_{(y, \theta)} R_{(z, \psi)}$$

The rotation matrix can be calculated by the following equation.

$$R_{\text{Euler}} =$$

$$\begin{pmatrix} \cos\phi \cos\theta \cos\psi - \sin\phi \sin\psi & -\cos\phi \cos\theta \sin\psi - \sin\phi \cos\psi & \cos\phi \sin\psi \\ \sin\phi \cos\theta \cos\psi + \cos\phi \sin\psi & -\sin\phi \cos\theta \sin\psi + \cos\phi \cos\psi & \sin\phi \sin\psi \\ -\sin\theta \cos\psi & \sin\theta \sin\psi & \cos\theta \end{pmatrix}$$

Related topics

Basics

Cardan Roll, Pitch, and Yaw Angles.....	25
Coordinate System Used in MotionDesk.....	22

Cardan Roll, Pitch, and Yaw Angles

Introduction

The Cardan angles of roll, pitch, and yaw angles are three angles that describe the orientation of a body in a three-dimensional space.

Cardan roll, pitch, and yaw angles (z-y'-x'' convention)

Using the Cardan angles of roll, pitch and yaw, a 3-D object is rotated around its axes in the following manner if the z-y'-x'' convention is used:

1. Rotation around the z-axis at angle ϕ
2. Rotation around the y-axis at angle θ
3. Rotation around the actual x-axis at angle ψ

The rotations are expressed in the following equation:

$$R_{\text{RPY}} = R_{\text{RPY}}(\phi, \theta, \psi) = R_{(z, \phi)} R_{(y, \theta)} R_{(x, \psi)}$$

The rotation matrix can be calculated with the following equation.

$$R_{RPY} = \begin{pmatrix} \cos\phi \cos\theta & \cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi & \cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi \\ \sin\phi \cos\theta & \sin\phi \sin\theta \sin\psi + \cos\phi \cos\psi & \sin\phi \sin\theta \cos\psi - \cos\phi \sin\psi \\ -\sin\theta & \cos\theta \sin\psi & \cos\theta \cos\psi \end{pmatrix}$$

Cardan roll, pitch, and yaw angles (z-x'-y'' convention)

Using the Cardan angles of roll, pitch, and yaw, a 3-D object is rotated around its axes in the following manner if the z-x'-y'' convention is used:

1. Rotation around the z-axis at angle ϕ
2. Rotation around the x-axis at angle ψ
3. Rotation around the actual y-axis at angle θ

The rotations are expressed in the following equation:

$$R_{RPY} = R_{RPY}(\phi, \psi, \theta) = R_{(z, \phi)} R_{(x, \psi)} R_{(y, \theta)}$$

The rotation matrix can be calculated with the following equation.

$$R_{RYP} = \begin{pmatrix} \cos\phi \cos\psi - \sin\phi \sin\theta \sin\psi & -\sin\phi \cos\theta & \cos\phi \sin\psi + \sin\phi \sin\theta \cos\psi \\ \sin\phi \cos\psi + \cos\phi \sin\theta \sin\psi & \cos\phi \cos\theta & \sin\phi \sin\psi - \cos\phi \sin\theta \cos\psi \\ -\cos\theta \sin\psi & \sin\theta & \cos\theta \cos\psi \end{pmatrix}$$

Related topics

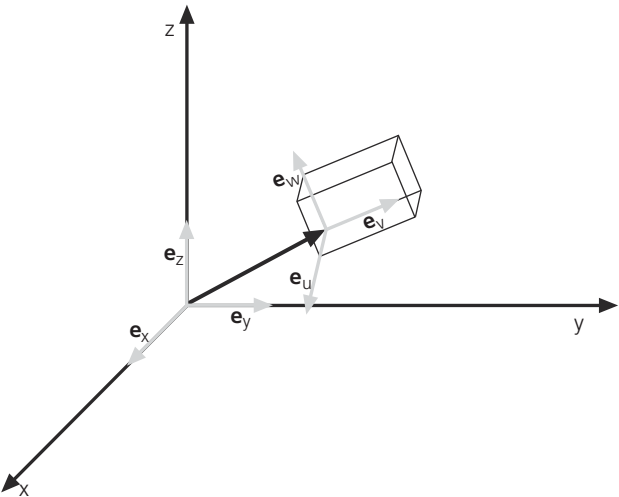
Basics

Coordinate System Used in MotionDesk.....	22
Euler Angles.....	25

3-D Objects in a 3-D Space

3-D objects

In contrast to a vertex, a 3-D object has an expansion, see the following illustration.



When defining a position for a 3-D object, you have to consider two different coordinate systems: the world and the local coordinate systems. Every 3-D object has its own local coordinate system.

In the illustration above, the world coordinate system is defined with the orthogonal unity vectors \vec{e}_x , \vec{e}_y , \vec{e}_z and the local coordinate system is defined with the orthogonal unity vectors \vec{e}_u , \vec{e}_v , \vec{e}_w .

In general, the orientations of a local coordinate system and the world coordinate system are different.

Related topics

Basics

Vertex in a 3-D Space.....	23
----------------------------	----

Working with the Model and Sensor Interface Blockset

Introduction

You can learn to use the Model and Sensor Interface Blockset to calculate and stream the data for sensor simulation.

Where to go from here

Information in this section

[Adapting Simulink Models..... 28](#)

You build a Simulink model with the blocks of the Model and Sensor Interface blocks for simulation with sensors.

[Configuring the Simulation Data Objects..... 48](#)

You add and configure Simulation Data Objects Blocks to the Simulink model to store and calculate the simulation data for the objects in the simulation.

[Configuring Network Settings for Blockset Ethernet Connections..... 64](#)

To work with the Model and Sensor Interface Blockset, you must configure your network settings to communicate via Ethernet.

[Connecting Simulation Platforms to Sensor Simulation Systems..... 68](#)

You connect a SCALEXIO or VEOS simulation platform to a Sensor Simulation system or application using the Model and Sensor Interface Blockset and Ethernet functions.

Adapting Simulink Models

Introduction

You build a Simulink model with the blocks of the Model and Sensor Interface blocks for simulation with sensors.

Where to go from here

Information in this section

[Workflow to Adapt a Simulation Model with the Model and Sensor Interface Blockset..... 29](#)

To build a Simulation model for sensor simulation with the blocks of the Model and Sensor Interface Blockset.

[How to Access the Model and Sensor Interface Blockset..... 32](#)

You can access the Model and Sensor Interface Blockset in MATLAB Simulink and use each of the blocks in the Simulink model for sensor simulation.

How to Prepare the ASM Model with the Animation Interface Subsystem.....	36
You add the Animation Interface subsystem to the ASM model.	
How to Connect a Simulink Model to a Sensor Simulation System or Application.....	39
You must add a Connection Settings block in the simulation model to connect it to a Sensor Simulation system or application.	
Connecting Simulation Platforms Using Ethernet Functions.....	42
You must connect the simulation platform in the Simulink model to the Model and Sensor Interface Blockset Connection Settings block using Ethernet functions.	
How to Collect Status Information using the System Status block.....	43
You can configure the System Status block to collect status and statistical information for the connected system or application.	
How to Configure Sensor Failures for a Connected Sensor.....	44
You can use the Sensor Failure block to specify and control sensor failures for a connected sensor.	
How to Configure Sensor Feedback from the Environment Sensor Interface Unit.....	46
You can use the Sensor Feedback block to configure the transmission of sensor feedback data from the Environment Sensor Interface Unit.	

Workflow to Adapt a Simulation Model with the Model and Sensor Interface Blockset

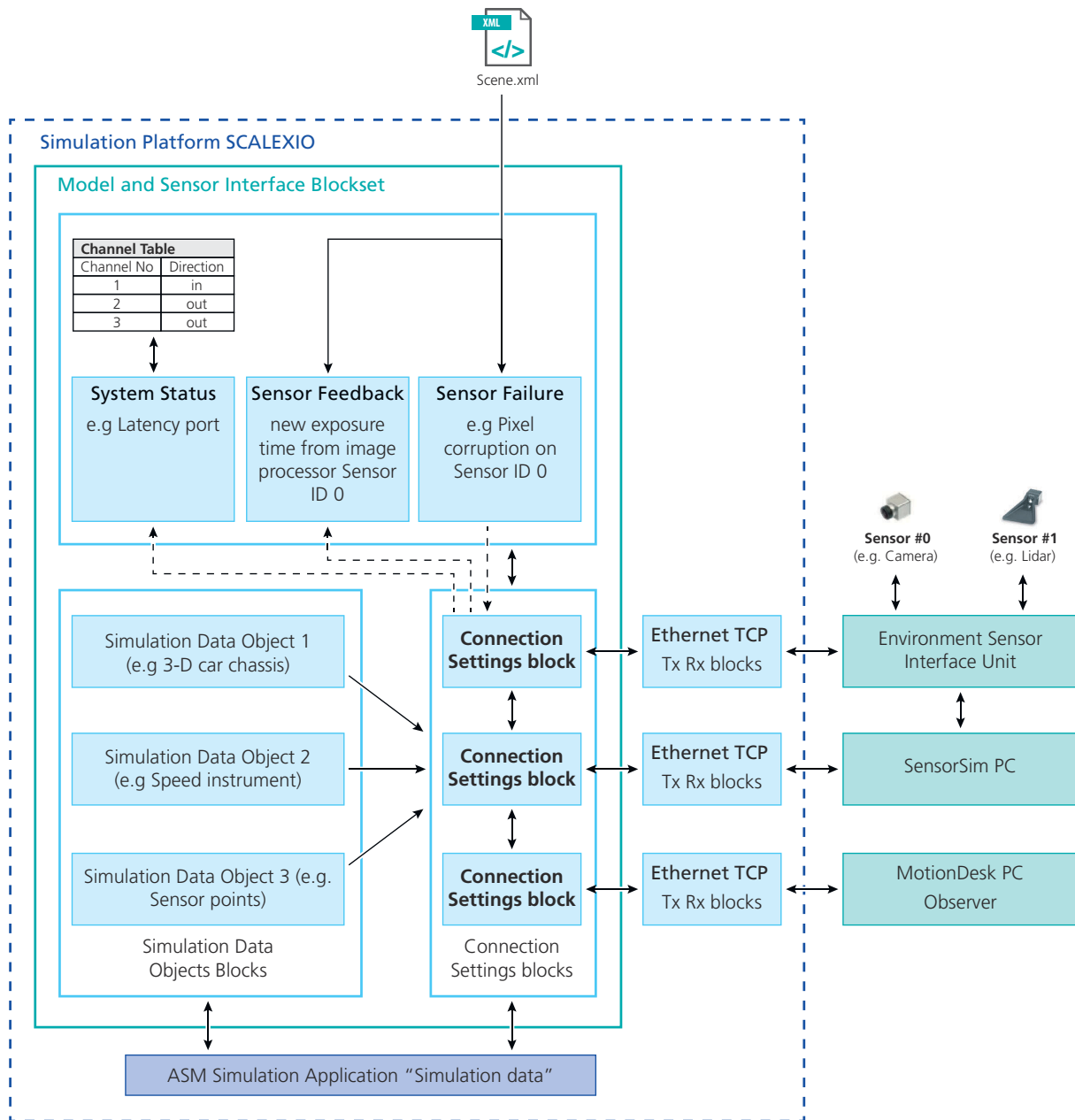
Introduction

To build a Simulation model for sensor simulation with the blocks of the Model and Sensor Interface Blockset.

Illustration

The illustration shows an example Simulation model using each block of the Model and Sensor Interface Blockset.

An example of the possible connections of the System Status, Sensor Feedback, and Sensor Failure blocks are shown connected to the Environment Sensor Interface Unit.



Overview

You can connect the simulation platform with MotionDesk, the SensorSim application, and the Environment Sensor Interface Unit for simulation with sensors.

This example workflow describes the steps required to build a Simulation model using each block of the Model and Sensor Interface Blockset.

Block connections

You must connect all blocks in the model. Unconnected blocks are not permitted.

- **Connection Settings:** The blocks must be connected to a Sensor Simulation system or application through Ethernet functions.
System Status, Sensor Feedback, and Sensor Failure blocks must be connected to only a single Connection Settings block.
- **Simulation Data Objects:** The blocks automatically connect to the Connection Settings blocks.
- **System Status, Sensor Feedback, and Sensor Failure:** The blocks that can be connected depend on the connected system or application:
 - **System Status:** Can be connected for any connected system or application.
 - **Sensor Feedback:** One or more can be used for a connection to an Environment Sensor Interface Unit.
 - **Sensor Failure:** One or more can be used for a connection to an Environment Sensor Interface Unit.

Note

The connections are created in the block parameters dialogs and cannot be created by connecting the blocks via signal wires in the simulation model.

For more information on the block descriptions and parameters, refer to [Model and Sensor Interface Blockset Library Overview](#) on page 91.

Workflow for building the simulation model

1. **Connect the simulation platform**
Connect the simulation platform to MotionDesk and the SensorSim application. The data is transferred via Ethernet communication. Ethernet functions must be connected.
For examples on connecting VEOS and SCALEXIO platforms, refer to [Connecting Simulation Platforms to Sensor Simulation Systems](#) on page 68.
2. **Adapt the simulation model**
 - **Simulation Data Objects**
In the simulation model, add Simulation Data Objects blocks and configure the objects with types, names, and other properties in the parameters dialog. Specify whether you require multiple objects for a specific transformation object.

The Simulation Data Objects blocks connect by default to the Connection Settings. The simulation data is transmitted if Send objects is selected.

- **System status information**

Connect a System Status block to the Connection Settings block and define a mapping table to map each in and out channel of the MotionDesk observer and the SensorSim application to one of the ports of the System Status block, for example, the Frame Counter or Latency port.

- **Sensor failure parameters**

Connect a Sensor Failure block to the Connection Settings block. Specify the Sensor ID on which to enable the sensor failure and add the failure class. The Sensor ID can be imported from the `scene.xml` file.

- **Sensor feedback parameters**

Connect a Sensor Feedback block to the Connection Settings block. Specify the Sensor ID on which to enable the sensor feedback and add the feedback parameter class. The Sensor ID can be imported from the `scene.xml` file.

Note

Camera, laser, and fish-eye sensors are supported if the sensor failure and feedback parameters are imported from a `scene.xml` from a MotionDesk project.

3. Build the real-time application in ConfigurationDesk for a SCALEXIO platform or the Simulink implementation container (SIC) for a software-in-the-loop (SIL) sensor simulation using VEOS.

Related topics

References

[Model and Sensor Interface Blockset Library Overview.....91](#)

How to Access the Model and Sensor Interface Blockset

Objective

You can access the Model and Sensor Interface Blockset in MATLAB Simulink and use each of the blocks in the Simulink model for sensor simulation.

Overview

To access the blocks of the Model and Sensor Interface Blockset and add them to your Simulink model, you can use the following methods:

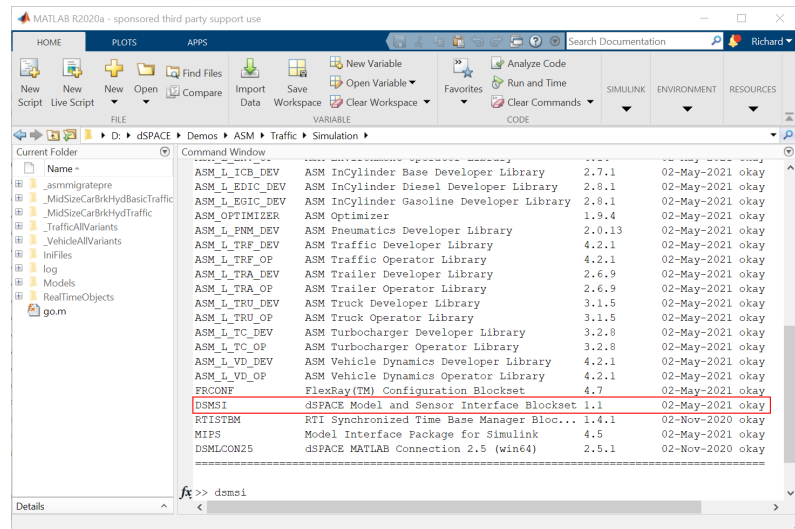
- [Method 1](#) on page 33.
- [Method 2](#) on page 34.

Method 1

To open the Model and Sensor Interface Blockset library

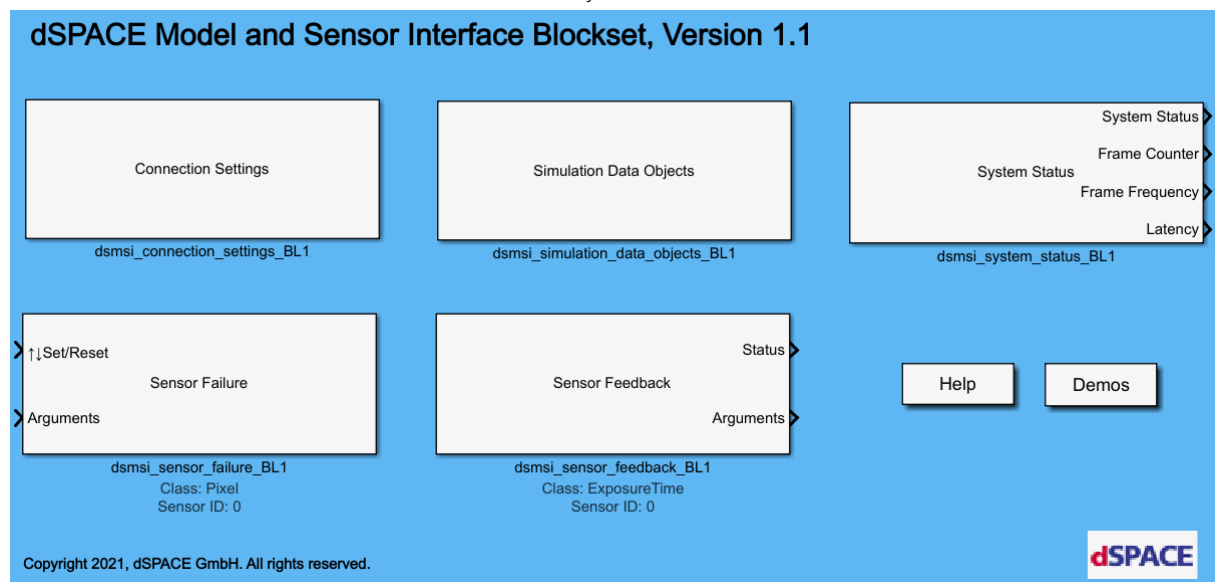
1 Start MATLAB.

The installed dSPACE Model and Sensor Interface Blockset is displayed in the Command Window.

2 In the MATLAB Command Window, type `dsmsi` at the Command Prompt.

3 Press Enter.

The blocks of the Model and Sensor Interface Blockset are displayed in the `dsmsilib` library.



Tip

You can also open the Model and Sensor Interface Blockset library from the installation folder in the File Explorer.

For example, in C:\Program Files\dSPACE RCPHIL {version}\MATLAB\DSMSI\DSMSI, double-click **dsmsilib.slx**.

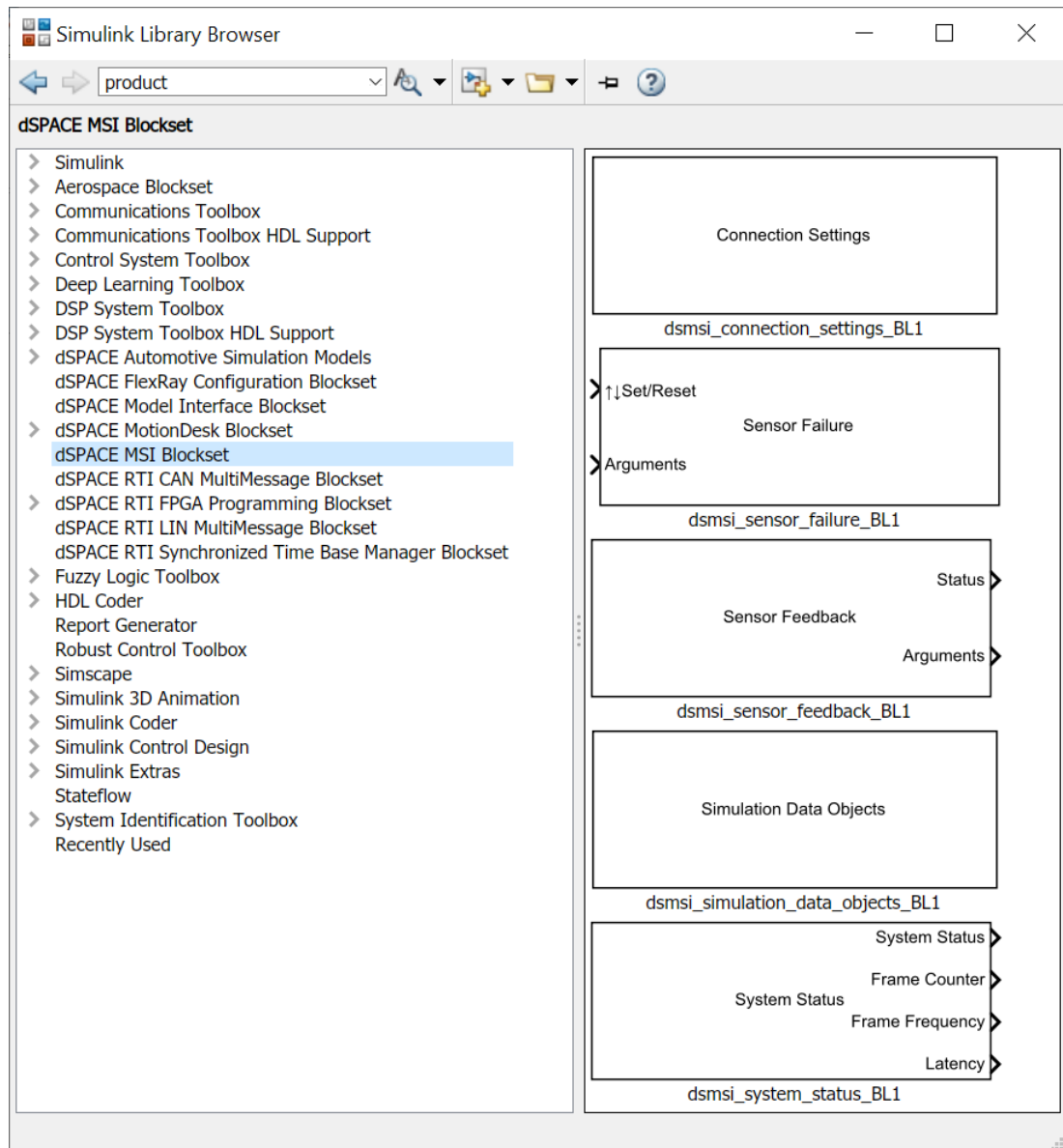
- 4 Drag the blocks you require and drop them into your Simulink model.

Method 2

To select the blocks in the Simulink library

- 1 In MATLAB, open the model in Simulink
- 2 Click Library Browser in the Simulation ribbon to open the Simulink Library Browser.

3 Click dSPACE DSMSI Blockset in the explorer tree.



4 Select a block and press **CTRL+I** to insert the block into the open model. You can also drag and drop the blocks into your model.

Result

You accessed the Model and Sensor Interface Blockset library and started adding the blocks into your ASM model.

Related topics

References

[Model and Sensor Interface Blockset Library Overview.....](#) 91

How to Prepare the ASM Model with the Animation Interface Subsystem

Objective

You add the Animation Interface subsystem to the ASM model.

Overview

To work with the ASM Traffic Model created in ModelDesk for a sensor simulation, you must add the Animation Interface subsystem to the model.

The MotionDesk Interface subsystem can be removed or co-exist with the Animation Interface subsystem in the model.

Note

The MotionDesk Blockset used in the ASM Model created by ModelDesk does not support Sensor Simulation.

Tip

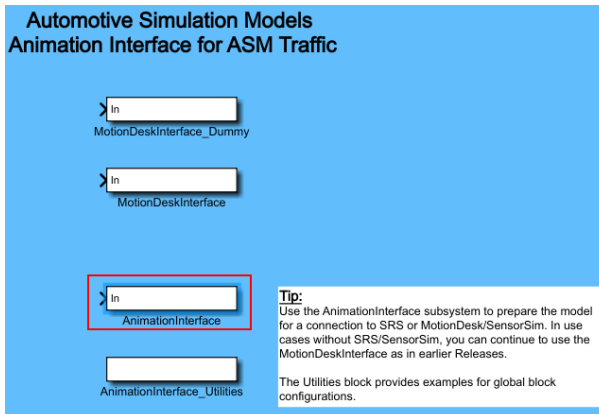
Configuration examples are also provided in the Animation Interface for ASM Traffic in the AnimationInterface_Uilities block.
For example, a configuration is prepared for a SCALEXIO platform with a Connection Settings block of the Model and Sensor Interface Blockset.
How to build the model manually is detailed in the following topics.

Method

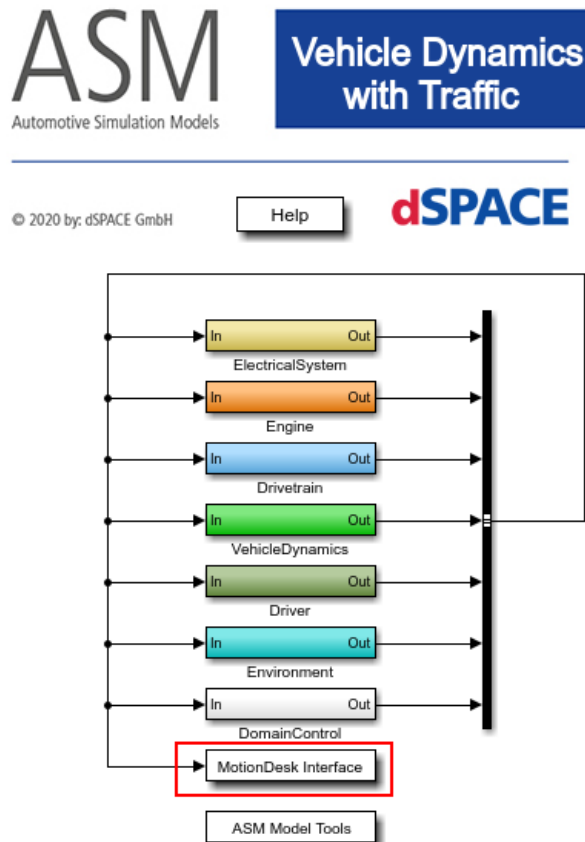
To add the Animation Interface block

- 1 To start working with the model, in MATLAB, navigate to the ...\\Simulation\\Models folder of the ModelDesk created ASM project and run the `go.m` script.
The ASM Model that was created by ModelDesk opens, for example, `ASM_Traffic.slx`.
- 2 In the same folder, double-click `ASM_AnimationInterface.slx`.
The ASM Animation Interface for ASM Traffic opens in Simulink.

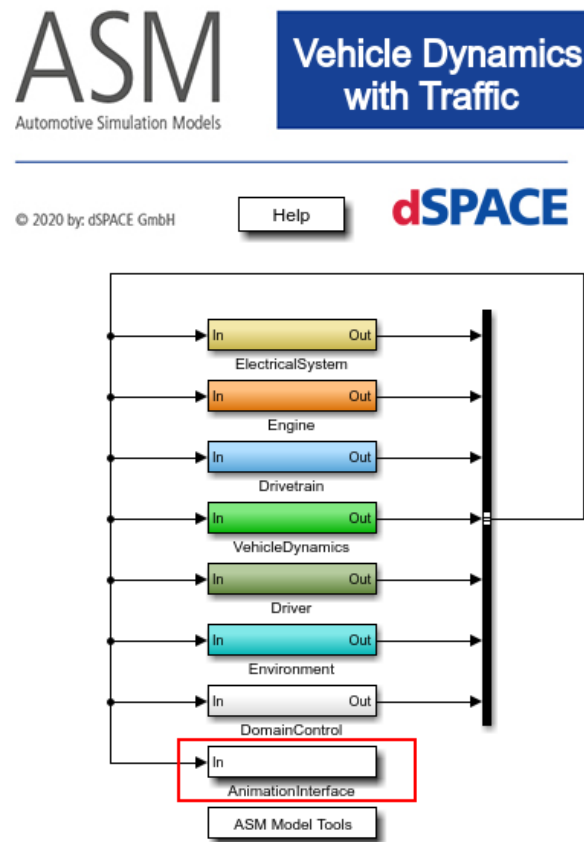
- 3 Copy the AnimationInterface subsystem.



- 4 Return to the ASM_Traffic Model and delete the MotionDeskInterface subsystem.



- 5 Paste the AnimationInterface subsystem into the ASM Traffic Model and connect as follows:



- 6 Save the ASM Traffic Model.

Result

You prepared the model for sensor simulation, by adding the ASM `AnimationInterface` subsystem in the ASM `Traffic Model`.

Related topics

HowTos

[Step 2: How to Prepare the ASM Traffic Model in Simulink \(Sensor Simulation Tutorial 📖\)](#)
[Step 2: How to Prepare the ASM Traffic Model in Simulink \(Sensor Simulation Tutorial 📖\)](#)

References

[Overview of the Vehicle Dynamics with Traffic Demo \(ASM Traffic Reference 📖\)](#)

How to Connect a Simulink Model to a Sensor Simulation System or Application

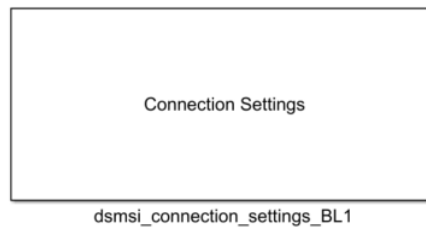
Objective You must add a Connection Settings block in the simulation model to connect it to a Sensor Simulation system or application, for example, MotionDesk, a SensorSim application, or an Environment Sensor Interface Unit.

Basics The Connection Settings block is connected to the Simulation Data Objects blocks that store and calculate the simulation data, for example, the transformation, signal, and sensor position data.

The Connection Settings block that prepares the data for transmission, is described in [Block Description \(dsmsi_connection_settings\)](#) on page 93.

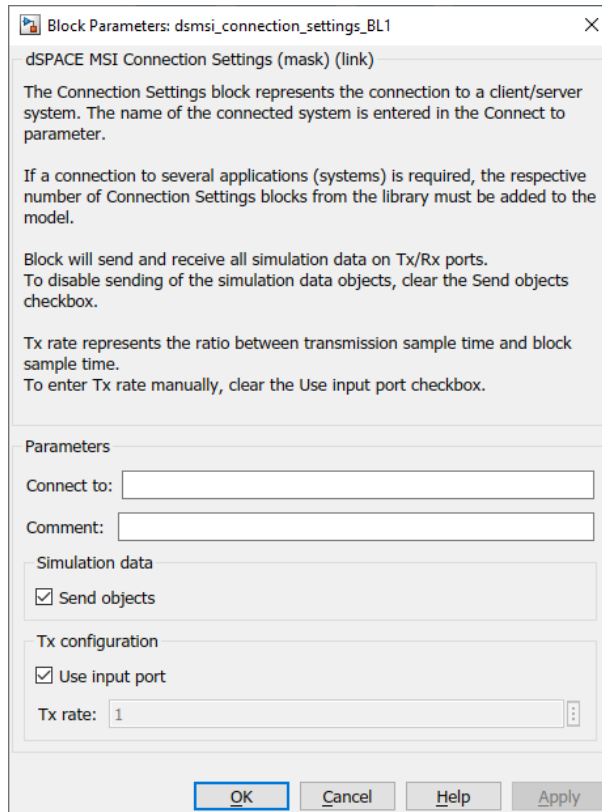
Method **To connect the Simulink model**

- 1 Drag and drop a Connection Settings block from the Model and Sensor Interface Blockset to your Simulink model.



The default name of the block is dsmsi_connection_settings_BL[x]. For more information on the object block names, refer to [Limitations When Using the Model and Sensor Interface Blockset](#) on page 137.

- 2 Double-click to open the Connection Settings block parameters dialog.



- 3 In the Connect to: field, type the name of the system to which this block connects, for example, MotionDesk. This must be provided to display the name, description, and ports in the block mask after you close the parameters dialog.
- 4 Add a description in the Comment field to be displayed below the block.
- 5 To transmit the simulation data calculated by the Simulation Data Objects blocks to the connected system, ensure that Simulation data - Send objects is selected.

Note

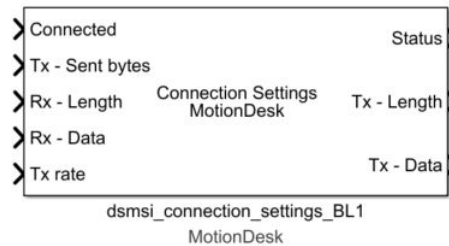
Each Connection Settings block receives all simulation data calculated by the Simulation Data Objects blocks and transmits it to the connected system if the Send objects checkbox is selected. Clear the checkbox to prevent simulation data from being sent to the connected system.

- 6 By default, Use input port is selected to indicate that the transmission rate is provided as a signal from the input port while running the simulation.
- 7 Clear Tx configuration - Use input port to enable the Tx configuration - Tx rate field to set the transmission rate manually as a numerical value.

The transmission rate is a fixed rate at which the simulation data is transmitted to the Connection Settings outputs to be forwarded to the connected system. The value in the Tx rate field represents the ratio between the transmission sample time and the block sample time.

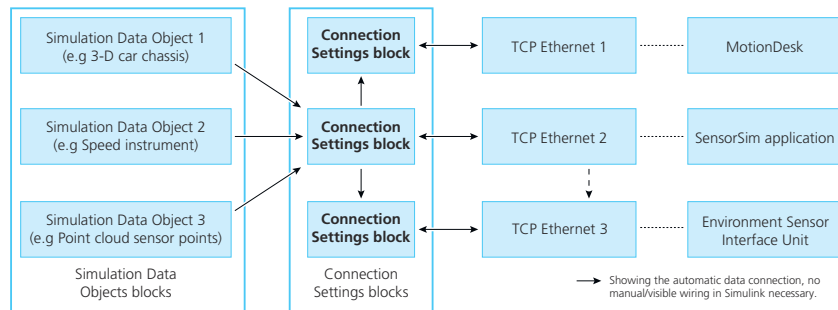
- 8 Click OK to save and close the block parameters dialog.

The block is displayed with the name of the connected system, the description, and the input and output ports of the block. The Tx rate input is displayed if Use input port is selected. If not, the manually set Tx rate is displayed on the block mask.



- 9 For each additional connected system or application, you must add a new Connection Settings block and specify the parameters.

The diagram displays an example of a Simulink model with three connected systems or applications.



Result

You added a Connection Settings block in the simulation model to connect it to a Sensor Simulation system or application.

To connect the simulation platform to a Sensor Simulation system or application, refer to [Connecting Simulation Platforms to Sensor Simulation Systems](#) on page 68.

Related topics

Basics

[Connecting Simulation Platforms to Sensor Simulation Systems..... 68](#)

Workflow to Adapt a Simulation Model with the Model and Sensor Interface Blockset.....	29
--	----

References

Block Description (dsmsi_connection_settings).....	93
Block Parameters Page (dsmsi_connection_settings).....	98

Connecting Simulation Platforms Using Ethernet Functions

Objective You must connect the simulation platform in the Simulink model to the Model and Sensor Interface Blockset Connection Settings block using Ethernet functions.

Basics You can connect SCALEXIO or VEOS simulation platforms to Sensor Simulation connected systems and applications using the Model and Sensor Interface Blockset.

The Model and Sensor Interface Blockset Connection Settings block has no I/O connection ports for data transmission.

Therefore you must configure Ethernet functions. For more information on the Ethernet functions for each platform, refer to [Basics of the Blockset and Connected Systems](#) on page 12

Supported Ethernet functions The Ethernet functions required for each of the supported simulation platforms are as follows:

Platform	Ethernet Functions
SCALEXIO (and DS6001 Processor board)	ConfigurationDesk Ethernet functions Ethernet Setup and TCP functions
VEOS and VEOS64 (VEOS on Linux64)	VEOS Ethernet Blockset Solution Ethernet TCP/IP Server and Client blocks

For a software-in-the-loop (SIL) sensor simulation using VEOS, you can build the model in Simulink with Ethernet server and client blocks from the VEOS Ethernet Blockset Solution.

For a HIL real-time sensor simulation using a SCALEXIO platform, you must add the Ethernet functions in ConfigurationDesk and propagate them to the Simulink model before you build the real-time application.

For examples on connecting simulation platforms to Sensor Simulation systems and applications using Ethernet functions, refer to [Connecting Simulation Platforms to Sensor Simulation Systems](#) on page 68.

Related topics

Basics

Basics of the Blockset and Connected Systems.....	12
Configuring Network Settings for Blockset Ethernet Connections.....	64
Connecting Simulation Platforms to Sensor Simulation Systems.....	68
Workflow to Adapt a Simulation Model with the Model and Sensor Interface Blockset.....	29

How to Collect Status Information using the System Status block

Objective

You can configure the System Status block to collect status and statistical information for the connected system or application.

Basics

The System Status (dsmsi_system_status) block must be connected to a specific Connection Settings block that defines the connection to MotionDesk, the SensorSim application, or the Environment Sensor Interface Unit.

The System Status has output ports to output the status and the statistical data from the connected system or application.

The System Status block can be defined to collect the below information types:

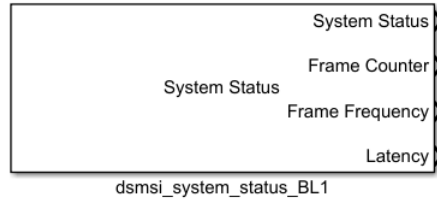
- **System Status:** Outputs the status of the connected system.
- **Frame Counter:** Outputs the number of received and transmitted frames per channel.
- **Frame Frequency:** Outputs the frequency of the incoming data per channel.
- **Latency:** Outputs the calculated time taken to transmit the data from the blockset and receive feedback data.

You can learn about the System Status block in the [Block Description \(dsmsi_system_status\)](#). An example of the configuration of a System Status block is included.

Method

To collect status information using the System Status block

- 1 In MATLAB Simulink, add a System Status block from the Model and Sensor Interface Blockset.



The default name of the block is `dsmsi_system_status_BL[x]`. For more information on the object block names, refer to [Limitations When Using the Model and Sensor Interface Blockset](#) on page 137.

- 2 Double-click to open the System Status block parameters dialog.
- 3 In the Connection Settings block parameter, select the relevant Connection Settings block for the system or application that you want to receive status information from.
- 4 In Channel table parameter, define a channel table to map each input and output channel on the connected system or application to a single element of the outputs of the System Status block, for example frame counter, frame frequency, and latency.

To learn about the System Status block parameters and ports, refer to the [Block Parameters Page \(dsmsi_system_status\)](#) on page 118.

Result

You configured the System Status block to collect status and statistical information for a connected system or application.

Related topics

Basics

[Workflow to Adapt a Simulation Model with the Model and Sensor Interface Blockset](#)..... 29

References

[Block Description \(dsmsi_system_status\)](#)..... 116
[Block Parameters Page \(dsmsi_system_status\)](#)..... 118

How to Configure Sensor Failures for a Connected Sensor

Objective

You can use the Sensor Failure block to specify and control sensor failures for a connected sensor.

Basics

The Sensor Failure (dsmsi_sensor_failure) block must be connected to a specific Connection Settings block that defines the connection to a Sensor Simulation system or application and the connected sensors.

The two inports on the Sensor Failure block are **Set/Reset** and **Arguments**.

Sensor failures are triggered with a signal change on the **Set/Reset** inport. If the signal changes from 0 to any other value, the failure is set (enabled). If the signal changes from any value to 0, the failure is reset (disabled).

The arguments for the sensor failure are provided through **Arguments** inport. You can also specify the arguments for the specific sensor failure class manually in the **Arguments** field on the block parameters dialog. You must first disable the arguments inport.

The dSPACE applications can simulate sensor failures. Two examples of sensor failures that can be received and sent by the Sensor Failure block to the connected system are as follows:

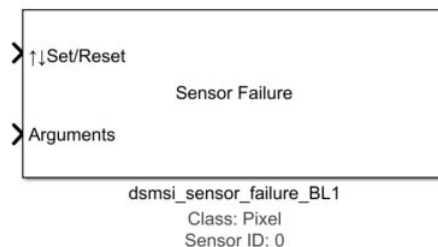
1. **Camera sensor pixel failure** - A deformed pixel can be simulated, for example, a black pixel at a specific position (x,y) on the camera sensor. This might be specified as [42, 24, 255]. The arguments indicate the x-, y-coordinates and the RGB pixel color.
2. **Image signal processor electrical failure** - An electrical failure can occur in the link between the camera image sensor and the image signal processing unit.

You can learn about the Sensor Failure block in the Block Description (dsmsi_sensor_failure).

Method

To configure sensor failures for a connected device

- 1 In MATLAB Simulink, add a Sensor Failure block from the Model and Sensor Interface Blockset.



The default name of the block is dsmsi_sensor_failure_BL[x]. For more information on the object block names, refer to [Limitations When Using the Model and Sensor Interface Blockset](#) on page 137.

- 2 Double-click to open the Sensor Failure block parameters dialog.
- 3 In Connection Settings block parameter, select the Connection Settings block that is connected to the Environment Sensor Interface Unit.
- 4 Specify the Sensor ID for the running sensor that is connected to the Environment Sensor Interface Unit.

The Sensor ID identifies the sensor data stream for the sensor to send a failure to. The ID can also be taken from the Scene.xml.

- 5 Specify a Failure class that is supported by the connected system and sensor.
- 6 Select Arguments - Use input port for failure arguments to be provided from the inport.
- 7 If you clear Use input port, you can specify the failure Arguments in the field and select the data type.

Note

The arguments that a connected system expects from the blockset must be defined by the connected system. They are not defined in the blockset.

To learn about the Sensor Failure block parameters and ports, refer to the [Block Parameters Page \(dsmsi_sensor_failure\)](#) on page 122.

Result

You have enabled sensor failures for the connected sensor.

Related topics

Basics

[Workflow to Adapt a Simulation Model with the Model and Sensor Interface Blockset](#)..... 29

References

[Block Description \(dsmsi_sensor_failure\)](#)..... 120
[Block Parameters Page \(dsmsi_sensor_failure\)](#)..... 122

How to Configure Sensor Feedback from the Environment Sensor Interface Unit

Objective

You can use the Sensor Feedback block to configure the transmission of sensor feedback data from the Environment Sensor Interface Unit.

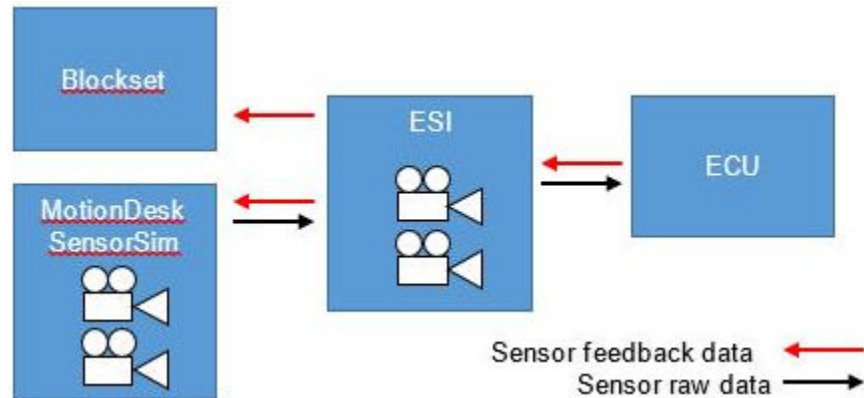
Basics

The Sensor Feedback (dsmsi_sensor_feedback) block must be connected to a specific Connection Settings block that defines the connection to the Environment Sensor Interface Unit and the connected sensor hardware devices.

The Environment Sensor Interface Unit sends the feedback data for all of the connected sensors to the connected systems and applications, for example, MotionDesk, the SensorSim application, and the blockset. The Sensor Feedback

is configured with a Sensor ID, feedback class and argument length. The block filters the data for this specific sensor and forwards the feedback data via the block Arguments outputport to the model for further processing. For example, the output can be connected for display or to another block in the model.

For example, a control unit changes the exposure time of the sensor. The Environment Sensor Interface Unit detects this feedback parameter and forwards the data to all connected systems. An illustration of an example data flow is as follows:



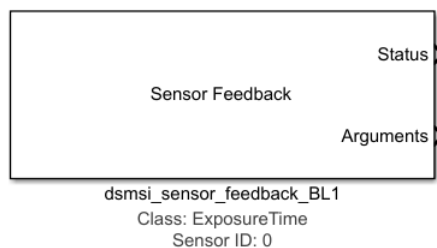
The Sensor Feedback has outputs for the sensor feedback status and for the sensor feedback arguments for the connected sensor feedback data. Sensor feedback is defined by the system that collects the feedback data. The system defines the feedback parameter class and the format of the arguments vector.

You can learn about the Sensor Feedback block in the [Block Description \(dsmsi_sensor_feedback\)](#) on page 125.

Method

How to configure a Sensor Feedback block

- 1 In MATLAB Simulink, add a Sensor Feedback block from the Model and Sensor Interface Blockset.



The default name of the block is dsmsi_sensor_feedback_BL[x]. For more information on the object block names, refer to [Limitations When Using the Model and Sensor Interface Blockset](#) on page 137.

- 2 Double-click to open the Sensor Feedback block parameters dialog.
- 3 In Connection Settings block parameter, select the Connection Settings block that is connected to the Environment Sensor Interface Unit.

- 4 Specify the Sensor ID for the running sensor that is connected to the Environment Sensor Interface Unit.
The Sensor ID identifies the sensor data stream for the sensor feedback. The Sensor ID can also be taken from the Scene.xml.
- 5 Specify the Feedback parameter class supported by the Environment Sensor Interface Unit.
- 6 Specify the length of the feedback data on the Arguments output.
To learn about the Sensor Feedback block parameters and ports, refer to [Block Parameters Page \(dsmsi_sensor_feedback\)](#) on page 126.

Result You have enabled sensor feedback for a connected sensor connected to the Environment Sensor Interface Unit.

Related topics

Basics

[Workflow to Adapt a Simulation Model with the Model and Sensor Interface Blockset](#)..... 29

References

[Block Description \(dsmsi_sensor_feedback\)](#)..... 125
[Block Parameters Page \(dsmsi_sensor_feedback\)](#)..... 126

Configuring the Simulation Data Objects

Introduction You add and configure Simulation Data Objects Blocks to the Simulink model to store and calculate the simulation data for the objects in the simulation.

Where to go from here

Information in this section

[Working with the Simulation Data Objects Block](#)..... 49

You can configure the Simulation Data Objects block for the transformation, signal, position, and position and scale object types used in the simulation. You can also create kinematic chains and specify multiple objects for a single transformation object.

[How to Configure the Simulation Data Objects in a Simulation Model](#)..... 51

You configure the Simulation Data Objects block that stores and calculates the simulation data for the objects in the simulation to Sensor Simulation systems and applications.

How to Configure Multiple Transformation Objects..... 53

You can specify multiple individual objects for each transformation object in the Simulation Data Objects block, for example, for multiple fellows in a simulation.

How to Import and Export Simulation Data Objects..... 56

You can import the configuration for a Simulation Data Objects block that you previously exported from another block.

Simulation Data Objects Block Automation API..... 59

You can automate a number of operations with the Simulation Data Objects block automation API provided with the blockset.

Working with the Simulation Data Objects Block

Introduction

You can configure the Simulation Data Objects block for the transformation, signal, position, and position and scale object types used in the simulation. You can also create kinematic chains and specify multiple objects for a single transformation object.

Basics

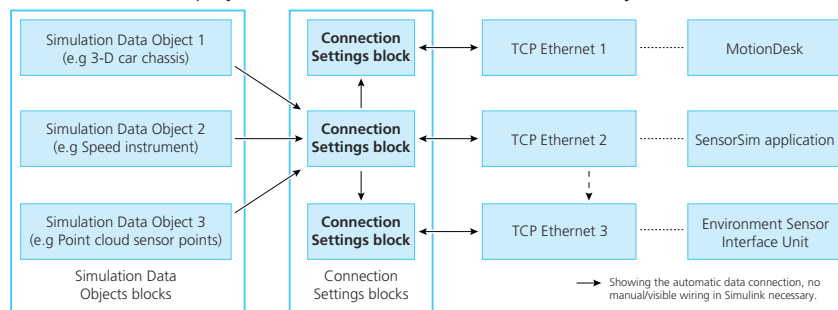
The Simulation Data Objects (`dsmsi_simulation_data_objects`) block calculates the simulation data.

The Connection Settings block then prepares the data for transmission to Sensor Simulation systems and applications by means of the Ethernet functions.

The Simulation Data Objects blocks calculate the transformation matrices of the objects, for example, their translation, rotation, and scaling, in addition to the sensor point and signal data.

In the block parameters, you can define the objects, types, and input and output parameters. For example, you can specify where the start transformation matrix is taken from. This is the reference with the start values of the transformation. You can also specify multiple individual objects for each transformation object.

The illustration displays three connected Simulation Data Objects blocks.



Note

Each Connection Settings block receives all simulation data calculated by the Simulation Data Objects blocks and transmits it to the connected system if the Send objects checkbox is selected. Clear the checkbox to prevent simulation data from being sent to the connected system.

For more information on the Simulation Data Objects block, refer to [Block Description \(dsmsi_simulation_data_objects\)](#) on page 100.

Kinematic chains

The Simulation Data Objects block supports the calculation of kinematic chains for transformation objects.

You can add multiple objects within one Simulation Data Objects block and create parent-child relationships, for example, a vehicle and its four tires. The child objects can also have different object types, depending on the parent object type. The child objects are indented in the block mask. Refer to [Type](#) in the [Block Parameters Page \(dsmsi_simulation_data_objects\)](#) on page 104.

You can also create kinematic chains by connecting Simulation Data Objects blocks, where the output of one block connects to the input of another.

The Transform matrix output of a transformation object connects to the input of one of the following:

- The Transform matrix input for a transformation object of another Simulation Data Objects block.
- The Transform In input of another Simulation Data Objects block.

Therefore, the calculated transformation matrix of the first object provides the starting point for the object of the second block.

The outputs and inputs of the transformation objects support multiple individual objects if this is specified for a transformation object. The 4x4[n] or 3x4[n] transformation matrix is used. Signals connected to a transformation object are transmitted as vectors. The length is also based on the number of individual objects specified.

Multiple transformation objects

You can also specify multiple individual objects to be included for each transformation object in the Simulation Data Objects block, for example, for multiple fellows in a simulation. Therefore, it is not necessary to add individual objects in the Simulation Data Objects blocks for each of the fellows. As a result, changes to the number of fellows in the simulation can be made without changing the model.

Related topics

References

Block Description (dsmsi_simulation_data_objects)	100
Block Parameters Page (dsmsi_simulation_data_objects)	104

How to Configure the Simulation Data Objects in a Simulation Model

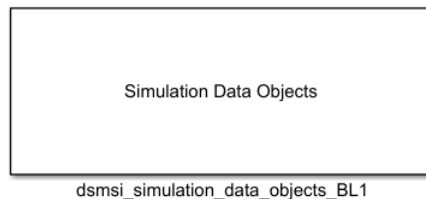
Objective

You configure the Simulation Data Objects block that stores and calculates the simulation data for the objects in the simulation to Sensor Simulation systems and applications.

Method

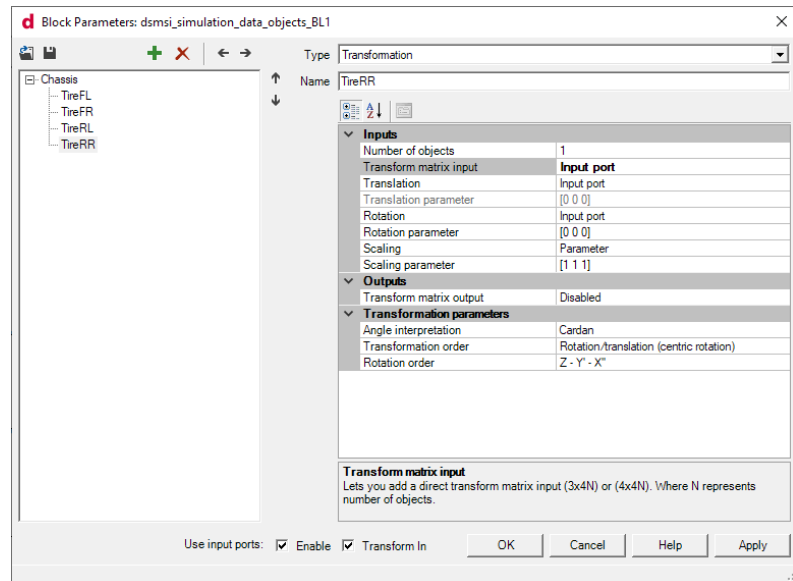
To configure the Simulation Data Objects in a Simulink model

- 1 Drag a Simulation Data Objects block from the Model and Sensor Interface Blockset into your Simulink model.



The default name of the block is `dsmsi_simulation_data_objects_BL[x]`. For more information on the object block names, refer to [Limitations When Using the Model and Sensor Interface Blockset](#) on page 137.


- 2 Double-click to open the Simulation Data Objects block parameters dialog.



Note

You cannot open the Simulation Data Objects dialog if the simulation using the model is running.

- 3 Select Enable input so the Simulation Data Objects block creates the Enable input in the block mask.

- 4 Select **Transform In** so the transformation matrix from this import is used as the starting transformation matrix for all of the objects in the block.
- 5 Click the add icon  to add a simulation data object to the object block tree.

Tip

You can also import simulation data object configuration to a new **Simulation Data Objects** block. For more information, refer to [How to Import and Export Simulation Data Objects](#) on page 56.

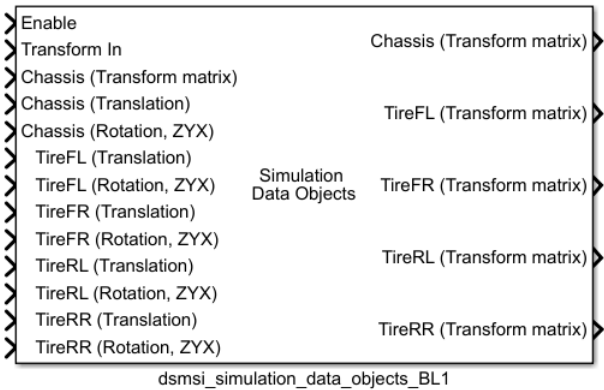
- 6 Click the object in the block tree and select a data object Type from the following list.
 - Transformation
 - Signal
 - Position
 - Position and Scale
- 7 Specify a name in the Name field for the object type, for example, Chassis for a vehicle transformation object.
- 8 Specify the parameters for each object type, for example, for the Transformation type, you can specify the input, output, and transformation parameters.
To learn about the Simulation Data Objects block types and parameters, refer to the [Block Parameters Page \(dsmsi_simulation_data_objects\)](#) on page 104.
- 9 Use the arrow buttons to change the order of the object and to create a parent-child relationship in a kinematic chain, for example, for the tires of a chassis.
For more information on creating kinematic chains and multiple objects for a transformation object, refer to [Working with the Simulation Data Objects Block](#) on page 49.

Note

If the number of objects is more than 1 for a transformation object, only the Signal object type can be added as children to the transformation object.

- 10 Click **Apply** to save your configuration or **OK** to save and close the dialog.

The block is displayed with the input and output ports of the block, for example, as follows:



Result You configured a Simulation Data Objects block. You added objects, specified their object types, and configured their parameters. You also created a parent-child object relationship in a kinematic chain.

Related topics

Basics	
Workflow to Adapt a Simulation Model with the Model and Sensor Interface Blockset.....	29
Working with the Simulation Data Objects Block.....	49
References	
Block Description (dsmsi_simulation_data_objects).....	100
Block Parameters Page (dsmsi_simulation_data_objects).....	104

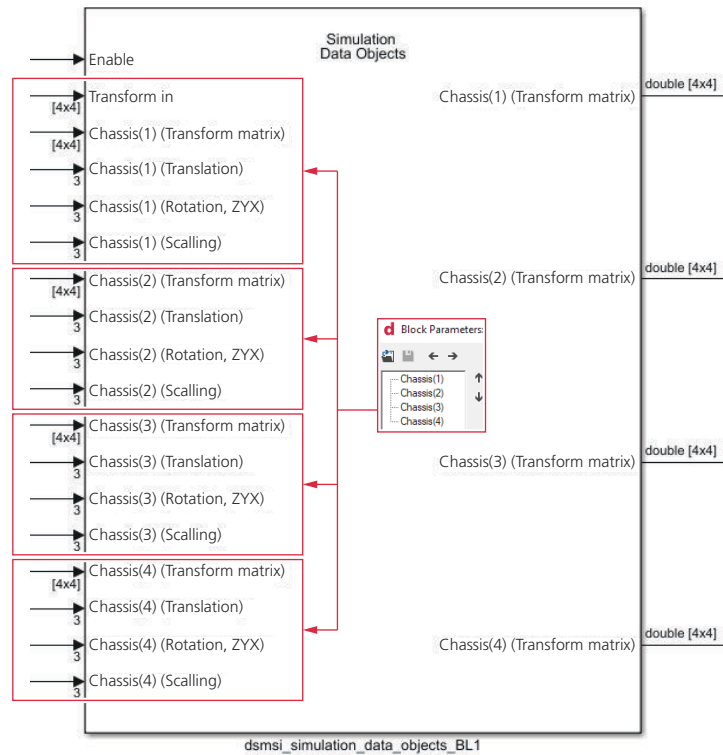
How to Configure Multiple Transformation Objects

Objective You can specify multiple individual objects for each transformation object in the Simulation Data Objects block, for example, for multiple fellows in a simulation.

Overview It is not necessary to add multiple identical transformation objects in the Simulation Data Objects dialog or to create multiple Simulation Data Objects blocks for each fellow. To create multiple instances of a transformation object, for example, for multiple fellows in a simulation, you can specify the number of objects in the Simulation Data Objects block dialog. The number of objects is shown in parenthesis in the block mask.

As a result, the number of fellows in the simulation can be changed without changing the model.

This is the equivalent of creating four identical objects in the dialog, as shown in the following block example. The four chassis objects, 01 to 04, are shown in colored boxes.



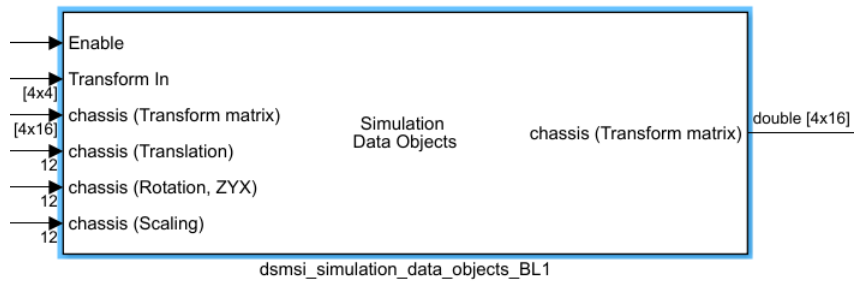
When you add a Signal object as a child of a transformation object

When you add a Signal object as a child of a root transformation object, for example, for vehicle speed, acceleration, engine revolutions, and the engaged gear, these objects inherit the number of objects from the parent transformation object. Therefore, the data is transmitted for each signal multiplied by the number of objects specified for the transformation object.

Note

Only the Signal type can be used in kinematic chains as children for a root transformation object if the number of objects specified is greater than 1. For more information on creating kinematic chains and multiple objects for a transformation object, refer to [Working with the Simulation Data Objects Block](#) on page 49.

You can specify the number objects for a transformation object in the dialog (for an animated graphic, refer to dSPACE Help).



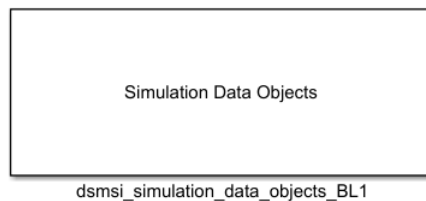
Tip


You can also automate the process of setting the number of objects for a transformation object. For more information refer to [Simulation Data Objects Block Automation API](#) on page 59.

Method

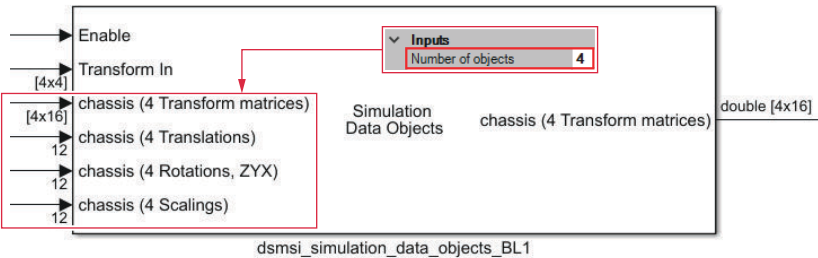
To configure the multiple transformation objects

- 1 Drag a Simulation Data Objects block from the Model and Sensor Interface Blockset into your Simulink model.



- 2 Double-click on a Simulation Data Objects block to load the block parameters dialog.
- 3 Click the add icon  to add a simulation data object to the object block tree.
- 4 Select the Transformation type in the properties for the new object.
- 5 Type a name for the object, for example, **Chassis**.
- 6 Select the input and output ports for the object, for example, the Transform matrix, Translation, Rotation, and Scaling input ports and the Transform matrix output port.
- 7 In the Number of objects input parameter, specify multiple objects, for example, **4**.
- 8 Click OK in the block parameters dialog.

In the following block example, the **Chassis** transformation object ports are duplicated to create four objects for this transformation object that represents four fellows in the simulation. A single transformation object is shown and the number of objects (**4**) is displayed in the parenthesis for each chassis input and output.



Result You specified 4 objects for a **Chassis** transformation object in the Simulation Data Objects block, for example, for multiple fellows in a simulation.

Related topics

Basics	
Simulation Data Objects Block Automation API.....	59
Working with the Simulation Data Objects Block.....	49
References	
Block Description (dsmsi_simulation_data_objects).....	100
Block Parameters Page (dsmsi_simulation_data_objects).....	104

How to Import and Export Simulation Data Objects

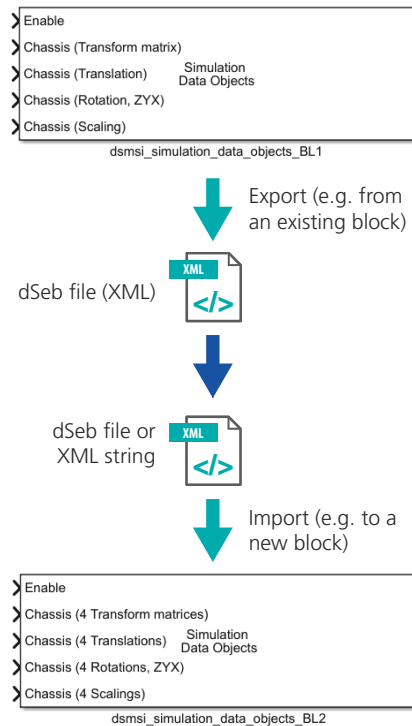
Objective You can import the configuration for a Simulation Data Objects block that you previously exported from another block.

Overview You can import simulation data objects in XML format to a new Simulation Data Objects block that you added to the model from the Simulink library. For example, you can import data that you exported from another Simulation Data Objects block.

The simulation data objects that you export for a block using the Simulation Data Objects dialog can be stored in a dSeb (dSPACE ESI blockset) file or in an XML file. You can edit the XML in a standard XML editor or the MATLAB internal XML parser and serializer. You can also import an XML string containing the simulation data objects.

If the file cannot be imported or exported, for example, if the target folder does not exist or the target file is read-only, a warning is displayed.

The following diagram shows an example of the import and export process. The simulation object data is exported from an existing block to a **dSeb** file. The xml is also updated to set the number of objects to **4** for the transformation object.



Note

It is recommended that you import to an empty **Simulation Data Objects** block. You import simulation data objects that you previously exported to a **dSeb** file from another block.


Editing the XML to reconfigure the properties of an existing pre-configured block is not recommended. Refer to [Limitations When Using the Model and Sensor Interface Blockset](#) on page 137.

Tip

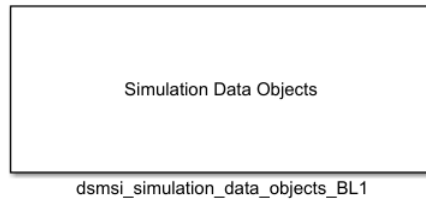
You can also automate the import and export process. For more information refer to [Simulation Data Objects Block Automation API](#) on page 59.


Method

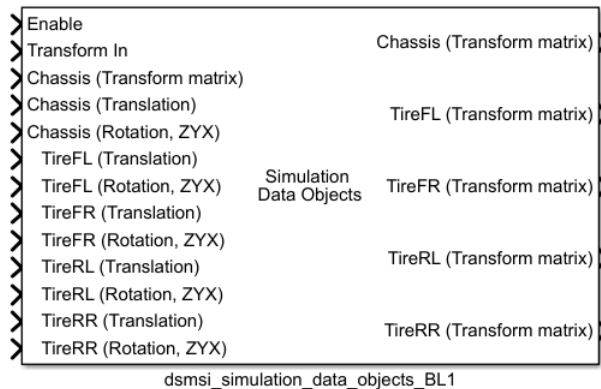
To import and export simulation data objects

- 1 In Simulink, double-click an existing **Simulation Data Objects** block that you previously configured in the block parameters dialog. The **Simulation Data Objects** block parameters dialog opens.
- 2 To export the simulation data objects block configuration to a **dSeb** file, click the export icon .

- 3 Search for the folder where you want to save the data objects block configuration to, type the file name, and click Save.
- 4 Click OK to close the Simulation Data Objects block parameters dialog.
- 5 Drag a Simulation Data Objects block from the Model and Sensor Interface Blockset into your Simulink model.



- 6 Double-click to open the Simulation Data Objects block parameters dialog.
- 7 To import the exported simulation data objects block configuration from the dSeb file, click  and search for the configuration file you previously exported.
- 8 Click Open.
The imported simulation data objects block configuration is shown in the block parameters dialog.
- 9 Click OK to close the Simulation Data Objects block parameters dialog.
The imported simulation data objects block configuration is shown in the block mask with the configured input and output ports, as shown in the following example.



Result

You exported a simulation data objects block configuration to a dSeb file and imported it into a new Simulation Data Objects block.

Related topics

Basics

[Simulation Data Objects Block Automation API](#)..... 59

Working with the Simulation Data Objects Block.....	49
---	----

References

Block Description (dsmsi_simulation_data_objects).....	100
Block Parameters Page (dsmsi_simulation_data_objects).....	104

Simulation Data Objects Block Automation API

Introduction

You can automate a number of operations with the Simulation Data Objects block automation API provided with the blockset.

Overview

The Model and Sensor Interface Blockset provides the `dsmsi_block_config_api` to automate a number of operations for the Simulation Data Objects. For example, you can import and export the simulation data objects of a block. You can also change the number of individual objects for a transformation object. The API is available through the MATLAB interface and implemented as a MATLAB function. A description with notes on how to use the API can also be shown in the MATLAB Command Window.

dsmsi_block_config_api arguments The `dsmsi_block_config_api` has four arguments that must be specified in the following order:

```
dsmsi_block_config_api ('<block_name>', '<operation_name>',  
'<option_1>', '<option_2>')
```

The operations are described in the following sections.

Importing and exporting simulation data objects

You can automate the import of simulation data objects in XML format to a new Simulation Data Objects block that you added to the model from the Simulink library using the API. For example, you can import data that you exported from another Simulation Data Objects block.

You specify the arguments for the API as follows:

```
dsmsi_block_config_api ('<block_name>', '<operation_name>',  
'<option_1>')
```

Note

An error message is shown if you import simulation data objects while the simulation is running.

Note

It is recommended to use the automation API to import to an empty Simulation Data Objects block an existing simulation data objects configuration that you previously exported. It is not recommended to edit the XML to reconfigure the properties of an existing preconfigured block. Refer to [Limitations When Using the Model and Sensor Interface Blockset](#) on page 137.

Block name You specify the Model name and the Simulation Data Objects block name, for example:

- 'ASM_model1/SimulationDataObjects_BL1'
- 'ASM_model1/Subsystem1/SimulationDataObjects_BL1'

Operation name You specify the operation command for the argument. The operations are as follows:

- **Import:** To import simulation data objects from a dSeb file.
- **Export:** To export simulation data objects to a dSeb file.
- **ImportXml:** To import simulation data objects directly from an XML string.

Tip

You can also import and export to another file type, for example, an XML file.

Option 1 You specify the argument depending on the operation as follows:

- **Import or Export:** The full path to the dSeb file.
- **ImportXml:** The XML string that contains the simulation data objects and their parameter configuration.

This second optional argument Option 2 is not required for the import and export operations.

Examples

- **Import**

```
dsmsi_block_config_api('ASM_model1/SimulationDataObjects_BL1', 'Import', 'D:\SDO_Chassis1.dSeb')
```

- **Export**

```
dsmsi_block_config_api('ASM_model1/SimulationDataObjects_BL1', 'Export', 'D:\SDO_Chassis1.dSeb')
```

- **ImportXml**

```
dsmsi_block_config_api('ASM_model1/SimulationDataObjects_BL1', 'ImportXml', Parameters - See the following example')
```

```

<?xml version="1.0" encoding="utf-8"?>
<Parameters xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance">
  <Version>1</Version>
  <TransformInput>true</TransformInput>
  <EnableInput>true</EnableInput>
  <Objects>
    <Object>
      <Oid>34</Oid>
      <Name>Chassis</Name>
      <TypeClass>
        <Name>Transformation</Name>
        <Class>1</Class>
        <TransformationProperty>
          <TransformMatrixOut>true</TransformMatrixOut>
        </TransformationProperty>
      </TypeClass>
      <Children>
        <Object>
          <Oid>35</Oid>
          <Name>TireFL</Name>
          <TypeClass>
            <Name>Transformation</Name>
            <Class>1</Class>
            <TransformationProperty>
              <TransformMatrixOut>true</TransformMatrixOut>
            </TransformationProperty>
          </TypeClass>
          <Children />
        </Object>
        <Object>
          <Oid>36</Oid>
          <Name>TireFR</Name>
          <TypeClass>
            <Name>Transformation</Name>
            <Class>1</Class>
            <TransformationProperty>
              <TransformMatrixOut>true</TransformMatrixOut>
            </TransformationProperty>
          </TypeClass>
          <Children />
        </Object>
        <Object>
          <Oid>7</Oid>
          <Name>TireRL</Name>
          <TypeClass>
            <Name>Transformation</Name>
            <Class>1</Class>
            <TransformationProperty>
              <TransformMatrixOut>true</TransformMatrixOut>
            </TransformationProperty>
          </TypeClass>
          <Children />
        </Object>
      </Children>
    </Object>
  </Objects>
</Parameters>

```

```
<Id>8</Id>
<Name>TireRR</Name>
<TypeClass>
  <Name>Transformation</Name>
  <Class>1</Class>
  <TransformationProperty>
    <TransformMatrixOut>true</TransformMatrixOut>
  </TransformationProperty>
</TypeClass>
<Children />
</Object>
</Children>
</Object>
</Objects>
</Parameters>
```

Multiple transformation objects

You can automate changes to the number of objects for a transformation object.

You specify the arguments for the API as follows;

```
dsmsi_block_config_api ('<block_name>', '<operation_name>',
  '<option_1>', '<option_2>')
```

For example, in the API function, you specify the name of the transformation object and the number of individual objects required.

Note

The Simulation Data Objects dialog must not be open and the simulation must not be running when using the function to change the number of objects. The function also checks whether the specified transformation object exists.

Block name You specify the Model name and the Simulation Data Objects block name, for example:

- 'ASM_model1/SimulationDataObjects_BL1'
- 'ASM_model1/Subsystem1/SimulationDataObjects_BL1'

Operation name To specify the number of individual objects required for a transformation object, you specify 'dimensions' as the operation name.

Option 1 You specify the name of the top-level Transformation object in the Simulation Data Objects block, for example, 'chassis'.

Option 2 You specify the number of individual objects required for the Transformation object, for example, '5'.

Example dsmsi_block_config_api('ASM_model1/SimulationDataObjects_BL1', 'Dimensions', 'chassis', '5')

Automation user files

XML Schema When you import or export the simulation data objects, the XML in the dSeb file or the XML string is validated using the Parameters.xsd XML schema. If the XML is not valid, an error is displayed during the import or export.

The Parameters.xsd is in the \MATLAB\DSMSI\Res folder in the Model and Sensor Interface Blockset installation folder.

Automation example An example MATLAB automation file dsmsi_automation_example is provided in the \MATLAB\DSMSI\Res folder in the Model and Sensor Interface Blockset installation folder.

The example shows how to export and import the Simulation Data Objects in XML format.

```
% Import Java XPath
import javax.xml.xpath.*

% Export configuration into tmp.xml
dsmsi_block_config_api(gcb, 'Export', 'tmp.xml');

% Import XML-file
dsmsi_block_config_api(gcb, 'Import', 'tmp.xml');

% Delete XML-file
delete('tmp.xml');
```

Automation API help

Help is provided for the automation API.

To load the help text, type `help dsmsi_block_config_api` in the MATLAB Command Window.

Related topics**Basics**

[Workflow to Adapt a Simulation Model with the Model and Sensor Interface Blockset.....](#) 29

References

[Block Parameters Page \(dsmsi_simulation_data_objects\).....](#) 104
[Limitations When Using the Model and Sensor Interface Blockset.....](#) 137

Configuring Network Settings for Blockset Ethernet Connections

Introduction To work with the Model and Sensor Interface Blockset, you must configure your network settings to communicate via Ethernet.

Where to go from here

Information in this section

[Network Basics for the Model and Sensor Interface Blockset..... 64](#)
Provides basic network settings information to work with the Model and Sensor Interface Blockset.

[How to Configure the Windows Network Settings for a Multi-PC Solution..... 66](#)
To connect multiple MotionDesk PCs to the simulation platform, you must configure the Windows settings for the network adapters.

[How to Configure the Network Settings in MotionDesk..... 67](#)
After configuring the network settings in Windows, you must configure the network settings in MotionDesk.

Network Basics for the Model and Sensor Interface Blockset

Introduction Provides basic network settings information to work with the Model and Sensor Interface Blockset.

TCP/IP protocols

The communication between PCs in a network is standardized by protocols. The Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) regulate the transport of data packages in a network in conjunction with the lower-level Internet Protocol (IP).

TCP transmits data as a data stream with no defined length. This is a sequence of transmitted data elements over a time period.

In contrast, UDP transmits data in datagrams and does not guarantee that each data package arrives or the time they take to arrive. However, UDP is faster than TCP and leads to shorter latency times.

Note

The Model and Sensor Interface Blockset uses TCP for data stream transmission. UDP is not supported.

IP address

Every PC in a network, or more precisely, every network adapter or host must have a unique address. TCP uses 32-bit values for the host address, known as the IP address.

The IP address is typically written in dotted-decimal format in 4 fields. Each part contains a number from 0 to 255 (one byte), and a typical network address is written as in the following example: 192.168.0.15.

The first one, two, or three fields of the address identify the network (subnet address), the rest represents the host address of the PC or network adapter itself. The fields used for the subnet address can be identified in the network mask. The network mask resembles a host address. Each part of the subnet address is given the number 255, each part of the host address the number 0.

If the IP address of your host is 192.168.0.15 and the mask is 255.255.255.0, then the subnet address is 192.168.0, and the host address is 15.

Private address space

In principle, you can use all possible numbers for IP addresses in a local network, as long as each host is given its own unique address.

However, this could cause problems if the local network is connected to the Internet in the future.

For this reason, it is advisable to use special addresses that are free for local networks and will not disturb any Internet connections. The following list shows the private address spaces:

- 10.0.0.0 to 10.255.255.255
- 172.16.0.0 to 172.16.255.255
- 192.168.0.0 to 192.168.255.255

Some numbers in the host part of the address are reserved for special purposes:

- If all host parts of the address are 0, this represents the entire network.
- If the host address is 255, you have the broadcast address, that is, all the hosts in the network receive the sent message. For example, 192.168.255.255 (network mask: 255.255.0.0) defines the broadcast address of subnet 192.168.
- Also reserved are the addresses 0.0.0.0 and 127.0.0.0 (network mask: 255.0.0.0). They define the so-called standard loop and loopback addresses.

TCP ports

The IP address specifies a specific host. Together with a port number, it defines a socket, for example, 192.168.0.1:2017. This allows for multiple processes to use TCP services on the same host for long-term conversations.

TCP provides port numbers from 1 to 65535. The numbers from 1025 to 65535 are not reserved and are free to use for your own network.

MotionDesk uses three TCP ports:

- Port 1 is used to send simulation data to the MotionDesk PCs
- Port 2 is used to send configuration requests from the MotionDesk PCs to the simulator, for example, to ask for the names of the 3-D objects.
- Port 3 is used to send configuration responses from the simulator to the MotionDesk PCs, for example, to send the names of the 3-D objects.

Firewall settings

To transfer data via Ethernet, the firewalls of the PCs must be configured to allow the communication for dSPACE software. As in a Simulink simulation, data is transferred via Ethernet, therefore the firewalls must allow the communication also for MATLAB.

Windows firewalls During the installation of dSPACE software, Windows firewalls are automatically configured to allow communication between the platform management instances for synchronization. You do not have to configure Windows firewalls manually. The first time you start a product involved in platform management synchronization, the firewall asks you to allow access. Confirm the product as trusted software.

Other firewalls If the host PC has a firewall different from the Windows firewall, configure that firewall manually to allow communication of dSPACE products and MATLAB via Ethernet.

How to Configure the Windows Network Settings for a Multi-PC Solution

Objective

To connect multiple MotionDesk PCs to the simulation platform, you must configure the Windows settings for the network adapters.

Method

To configure the Windows network settings for a multi-PC solution

- 1 In the Windows network settings dialog, configure the following parameters for the network adapter that is used for visualization in the MotionDesk PC:
 - Specify the IP address for the network adapter.
To build a local network, choose an address from the private address space (10.0.0.1 to 10.255.255.254, or 172.16.0.1 to 172.16.255.254, or 192.168.0.1 to 192.168.255.254).
 - Define the network mask (subnet mask), for example, **255.255.255.0**.
You do not have to define values for Default Gateway, DNS, WINS address, or Routing to build up a local network.
For more information, refer to your network adapter documentation.

Related topics**HowTos**

[How to Configure the Network Settings in MotionDesk..... 67](#)

How to Configure the Network Settings in MotionDesk

Objective

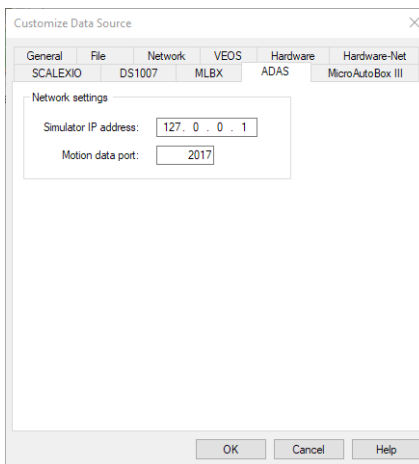
After configuring the network settings in Windows, you must configure the network settings in MotionDesk.

Preconditions

You configured the network settings in Windows and in the simulation model.

Method**To configure the network settings in MotionDesk**

- 1** In MotionDesk on the Home ribbon, select ADAS: TCP: Connection in the Platform - Sources list.
- 2** Click **Customize**.
The **Customize Data Source** dialog opens.
- 3** In **ADAS - Network Settings** set the values according to the simulation model.
 - Simulator IP address
 - Motion data port

**Result**

You configured the network settings in MotionDesk for an ADAS - TCP connection. You can visualize the model running in a simulation.

Related topics

HowTos

[How to Configure the Windows Network Settings for a Multi-PC Solution..... 66](#)

Connecting Simulation Platforms to Sensor Simulation Systems

Introduction

You connect a SCALEXIO or VEOS simulation platform to a Sensor Simulation system or application using the **Model and Sensor Interface Blockset** and Ethernet functions.

Where to go from here

Information in this section

[How to Connect a SCALEXIO Platform to a Single MotionDesk Observer..... 69](#)

You can connect a SCALEXIO platform and the simulation application to a single MotionDesk observer using the **Model and Sensor Interface Blockset** and **ConfigurationDesk Ethernet** functions.

[How to Connect a SCALEXIO Platform to Multiple MotionDesk Observers..... 73](#)

You can connect a SCALEXIO platform and the simulation application to a multiple MotionDesk observers using the **Model and Sensor Interface Blockset** and **ConfigurationDesk Ethernet** functions.

[How to Connect a VEOS Platform to a Single MotionDesk Observer..... 78](#)

You can connect a VEOS platform for software-in-the-loop (SIL) testing to a single MotionDesk observer using the **Model and Sensor Interface Blockset** and **VEOS Ethernet Blockset**.

[How to Connect a Simulation Platform to a SensorSim Application..... 81](#)

You can connect a simulation platform to a MotionDesk observer and to a SensorSim application for sensor simulations.

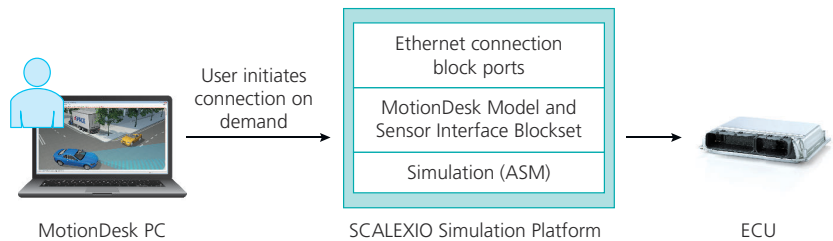
How to Connect a SCALEXIO Platform to a Single MotionDesk Observer

Objective

You can connect a SCALEXIO platform and the simulation application to a single MotionDesk observer using the Model and Sensor Interface Blockset and ConfigurationDesk Ethernet functions.

Illustration

The illustration shows an overview of the connection of a single MotionDesk observer to the SCALEXIO simulation platform. This is a client - server connection.



Connection Overview

An overview of the connection is as follows:

- You use the single MotionDesk PC client to visualize the simulation and to initiate the TCP/IP connection to the SCALEXIO platform.

The TCP function is the server as seen by the simulation platform. MotionDesk is the client.

- The SCALEXIO platform runs the real-time application that provides the simulation data to MotionDesk through the Model and Sensor Interface Blockset.
- In ConfigurationDesk, you import the model, add the Ethernet functions, specify the hardware, and extend the signal chain to create the model port blocks.

You then propagate the blocks to the ASM Simulink model and connect them with Connection Settings blocks that you add from the Model and Sensor Interface Blockset. You can then build the real-time application .

To learn to work with sensor simulation using a SCALEXIO platform to connect to MotionDesk and the SensorSim application , refer to [Lesson 3: Sensor Simulation on a SCALEXIO Platform \(Sensor Simulation Tutorial !\[\]\(0fb13ad0bfa3d86868cdd3883e5665b3_img.jpg\)](#)).

Note

Modeling in Simulink

You can also configure the model in Simulink. You connect the Connection Settings blocks with the model port blocks and build the SIC using the Simulink Coder.

You import the SIC to ConfigurationDesk, add the Ethernet Setup and TCP functions and configure them with the model port blocks in the signal chain. Lastly, you build the real-time application.

Prerequisites

To work with an ASM model for sensor simulation, you must add the **Animation Interface** block in the ASM model. Refer to [How to Prepare the ASM Model with the Animation Interface Subsystem](#) on page 36.

Note

The MotionDesk Blockset used in the ASM Model created by ModelDesk does not support Sensor Simulation.

Method

To connect a SCALEXIO platform to a single MotionDesk observer

- 1 In a new ConfigurationDesk project and application, import the ASM model that you created in ModelDesk for a SCALEXIO platform. The model is stored in the Simulation folder of the ModelDesk project.
- 2 Import or add a new hardware topology for the SCALEXIO simulation platform.
- 3 In the Function browser, add the following functions to the signal chain and display them in the Signal Chain browser:
 - Ethernet Setup
 - TCP
- 4 Right-click the Ethernet Setup function and select **Hardware Assignment - Assign Channel Set** on the context menu. Select the Ethernet adapter to use for the real-time application.
- 5 Select the function blocks and specify the following **General** and **Model Interface** properties for the ports:
 - Ethernet Setup

Property	Example value
IP address	192.168.1.1 You must configure the local IP address and subnet mask

- TCP

Property	Example value
Name	TCP (1) MotionDesk (Client)
Run as server/client	Server <i>The TCP function is the server as seen by the simulation platform. MotionDesk is the client.</i>
Local port	2017
TCP on Ethernet	192.168.1.1 <i>Select the Ethernet setup function.</i>

- TCP Transmit function

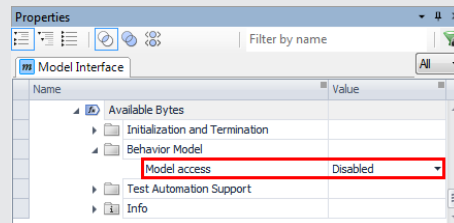
Property	Value
Maximum vector size	65507
Variable vector size	Defined by model
Message send mode	No delay

- TCP Receive function

Property	Value
Maximum vector size	65507
Variable vector size	Enabled
Message send mode	All available

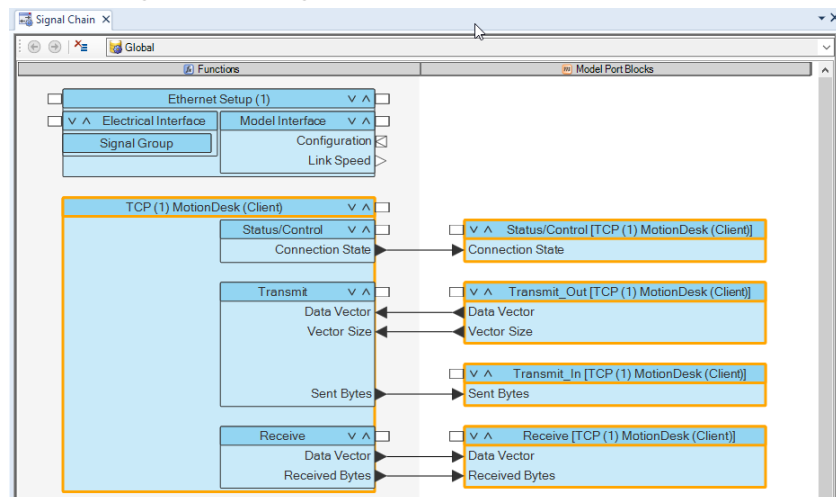
Tip

In Model Interface - Properties, disable Behavior Model - Model Access for the ports that are not required for the connection.



- 6 Select the TCP (1) function and extend the signal chain to create model port blocks for the TCP functions.

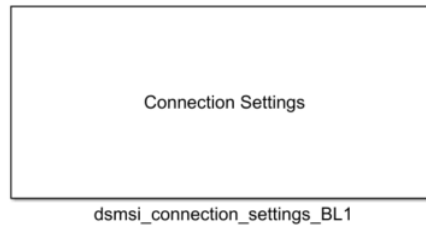
The ConfigurationDesk Functions and Model Port Blocks look like this in the ConfigurationDesk Signal Chain view:



- 7 Propagate the Model Port Blocks to the open ASM model in Simulink.

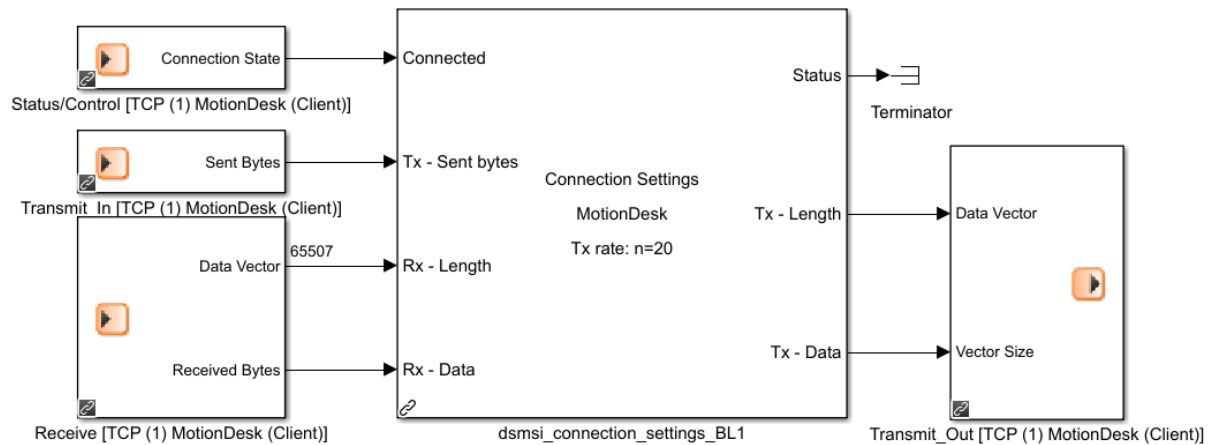
- 8 In the Simulink library, select dSPACE MSI Blockset.

Select the Connection Settings block and press **Ctrl+I** to insert the block into the ASM model.



- 9 Double-click the Connection Settings block and in the parameters dialog, specify the name, for example, **MotionDesk**. You can also specify the Tx rate on how often MotionDesk receives the simulation data.

- 10 In Simulink in the ASM model, connect the Ethernet model port blocks to the Connection Settings block of the Model and Sensor Interface Blockset as shown in the illustration:



- 11 In the Connection State parameters dialog, change the signal properties data Type to Boolean.

- 12 Build the model in ConfigurationDesk to create the real-time application.

Result

You connected a SCALEXIO platform and the simulation application to a single MotionDesk observer using the Model and Sensor Interface Blockset and ConfigurationDesk Ethernet functions.

Related topics**Basics**

[Lesson 3: Sensor Simulation on a SCALEXIO Platform \(Sensor Simulation Tutorial !\[\]\(666e09182d4cd268646ea700ea60dcdf_img.jpg\)\)](#)

HowTos

[Connecting Simulation Platforms Using Ethernet Functions..... 42](#)
[How to Connect a SCALEXIO Platform to Multiple MotionDesk Observers..... 73](#)
[How to Prepare the ASM Model with the Animation Interface Subsystem..... 36](#)

References

[Block Description \(dsmsi_connection_settings\)..... 93](#)

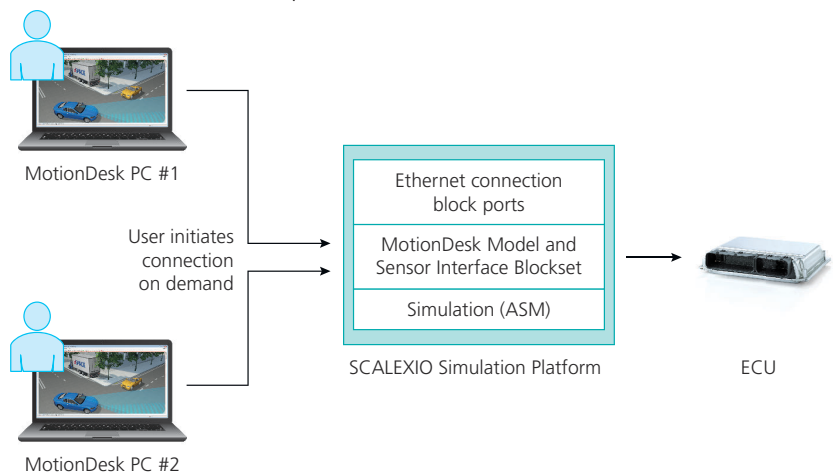
How to Connect a SCALEXIO Platform to Multiple MotionDesk Observers

Objective

You can connect a SCALEXIO platform and the simulation application to a multiple MotionDesk observers using the Model and Sensor Interface Blockset and ConfigurationDesk Ethernet functions.

Illustration

The illustration shows an overview of the connection of multiple MotionDesk observers to the SCALEXIO platform. This is a client-server connection.

**Connection Overview**

An overview of the connection is as follows:

- You use each MotionDesk PC client to visualize the simulation and to initiate the TCP/IP connection to the SCALEXIO platform.

The TCP function is the server as seen by the simulation platform. MotionDesk is the client.

- The SCALEXIO platform runs the real-time application that provides the simulation data to MotionDesk through the **Model and Sensor Interface Blockset**.
- In ConfigurationDesk, you import the model, add the Ethernet functions, specify the hardware, and extend the signal chain to create the model port blocks.

You then propagate the blocks to the ASM Simulink model and connect them with **Connection Settings** blocks that you add from the **Model and Sensor Interface Blockset**. You can then build the real-time application .

To learn to work with sensor simulation using a SCALEXIO platform to connect to MotionDesk and the SensorSim application , refer to [Lesson 3: Sensor Simulation on a SCALEXIO Platform \(Sensor Simulation Tutorial !\[\]\(0f848bbd71cef6b345273b16f905912a_img.jpg\)](#)).

Note

Modeling in Simulink

You can also configure the model in Simulink. You connect the **Connection Settings** blocks with the model port blocks and build the SIC using the Simulink Coder.

You import the SIC to ConfigurationDesk, add the Ethernet Setup and TCP functions and configure them with the model port blocks in the signal chain. Lastly, you build the real-time application.

Prerequisites

You must first connect a SCALEXIO platform to a single MotionDesk observer. Refer to [How to Connect a SCALEXIO Platform to a Single MotionDesk Observer](#)

To connect an additional MotionDesk observer, you must add an additional Ethernet function and connect this with an additional **Connection Settings** block in the Simulink model.

Method

To connect a SCALEXIO platform to multiple MotionDesk observers

- 1** Open the ConfigurationDesk project you used to add the first MotionDesk application connection.
In this project, you imported the ASM model, assigned the hardware, and added the Ethernet Setup function and the first TCP (server) function for MotionDesk.
- 2** In the Function browser, add an additional TCP function to the signal chain and display it in the Signal Chain browser.
- 3** Select the TCP function and specify the following **General** and **Model Interface** properties for the ports:

- TCP

Property	Example value
Name	TCP (2) MotionDesk (Client)
Run as server/client	Server <i>The TCP function is the server as seen by the simulation platform. MotionDesk is the client.</i>
Local port	2018 <i>The port numbers must be incremented for each MotionDesk observer, for example, the second Connection Settings block must be connected to port 2018.</i>
TCP on Ethernet	192.168.1.1 <i>Select the Ethernet setup function.</i>

- TCP Transmit function

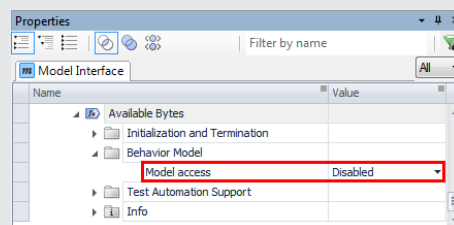
Property	Value
Maximum vector size	65507
Variable vector size	Defined by model
Message send mode	No delay

- TCP Receive function

Property	Value
Maximum vector size	65507
Variable vector size	Enabled
Message send mode	All available

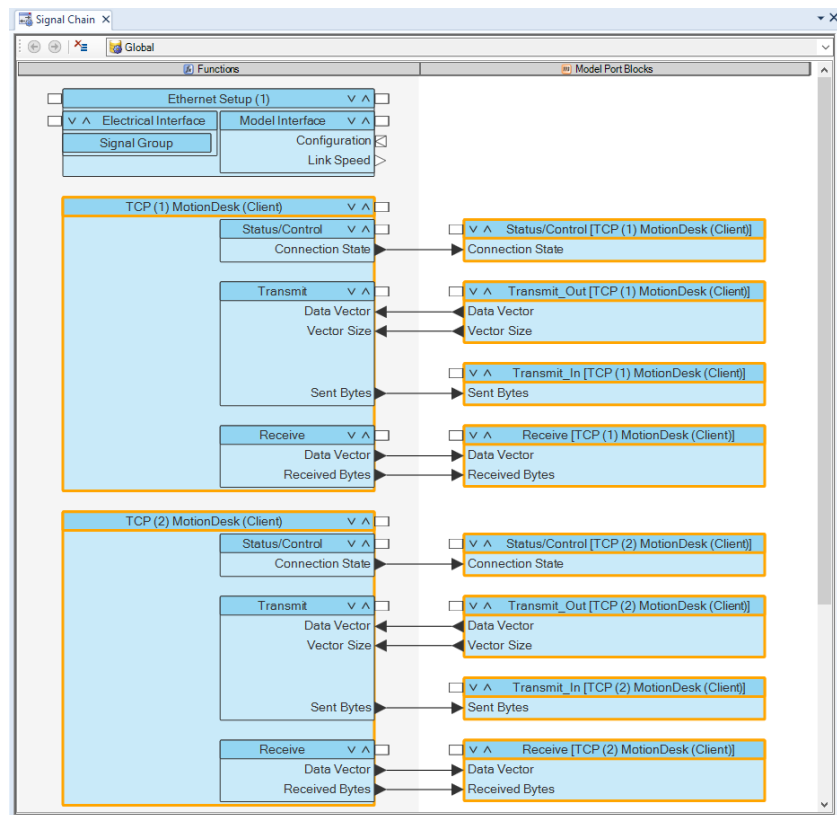
Tip

In Model Interface - Properties, disable Behavior Model - Model Access for the ports that are not needed for the connection.

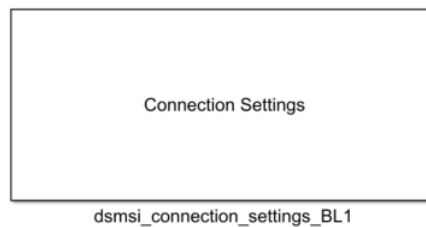


- 4 Select the TCP (2) function and extend the signal chain to create model port blocks for the TCP functions.

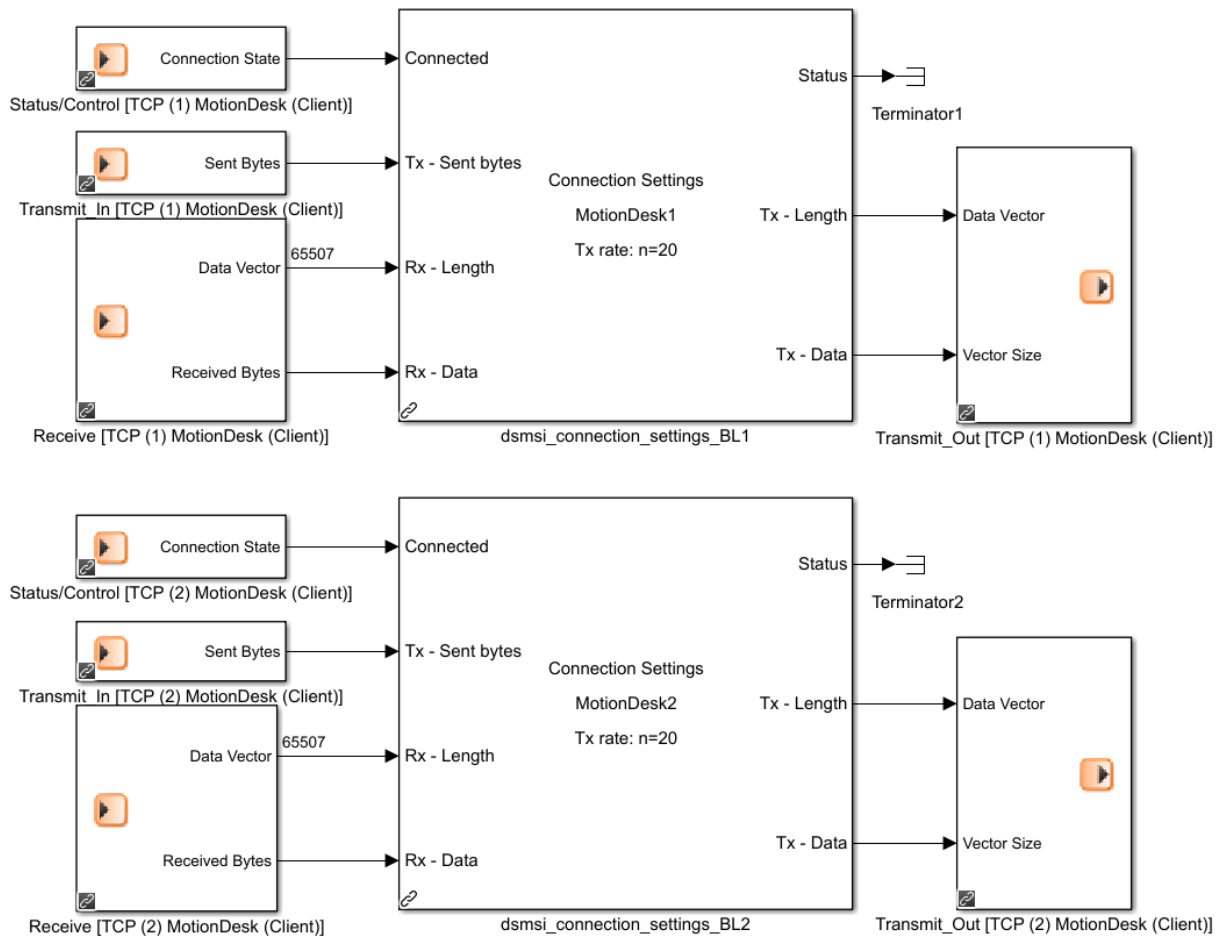
The ConfigurationDesk Functions and Model Port Blocks look like this in the ConfigurationDesk Signal Chain view:



- 5 Select the second MotionDesk (Client) TCP function and propagate the Model Port Blocks to the open ASM model in Simulink.
- 6 In the Simulink library, select dSPACE MSI Blockset. Select the Connection Settings block and press **Ctrl I** to insert the block into the ASM model.



- 7 Double-click the Connection Settings block and in the parameters dialog, specify the name, for example, **MotionDesk2**. You can also specify the Tx rate on how often MotionDesk gets the motion data.
- 8 In Simulink in the ASM model, connect the Ethernet model port blocks to the Connection Settings block of the Model and Sensor Interface Blockset as shown in the illustration:



9 In the Connection State parameters dialog, change the signal properties data Type to **Boolean**.

10 Build the model in ConfigurationDesk to create the real-time application.

Result

You connected a SCALEXIO platform and the simulation application to multiple MotionDesk observers using the Model and Sensor Interface Blockset and ConfigurationDesk Ethernet functions.

Related topics

Basics

[Lesson 3: Sensor Simulation on a SCALEXIO Platform \(Sensor Simulation Tutorial\)](#)

HowTos

[Connecting Simulation Platforms Using Ethernet Functions..... 42](#)
[How to Connect a SCALEXIO Platform to a Single MotionDesk Observer..... 69](#)

How to Prepare the ASM Model with the Animation Interface Subsystem.....	36
--	----

References

Block Description (dsmsi_connection_settings).....	93
--	----

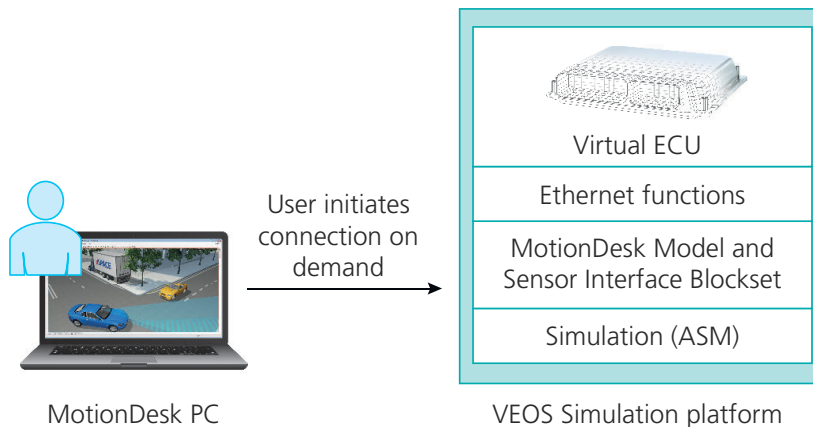
How to Connect a VEOS Platform to a Single MotionDesk Observer

Objective

You can connect a VEOS platform for software-in-the-loop (SIL) testing to a single MotionDesk observer using the Model and Sensor Interface Blockset and VEOS Ethernet Blockset.

Illustration

The illustration shows an overview of the connection of a single MotionDesk observer to a VEOS platform. This is a client-server connection.



Connection Overview

To connect a MotionDesk observer to a VEOS platform, you must create an Ethernet connection for VEOS and connect it with a **Model and Sensor Interface Blockset - Connection Settings** block.

An overview of the connection is as follows:

- You use the single MotionDesk PC client to visualize the simulation and to initiate the TCP/IP connection to the VEOS platform.
The TCP block is the server as seen by the simulation platform. MotionDesk is the client.
- The VEOS platform runs the simulation that provides the simulation data to MotionDesk through the **Model and Sensor Interface Blockset**.

- In Simulink in the ASM model, you add the **Connection Settings** blocks from the **Model and Sensor Interface Blockset** and the **Ethernet TCP/IP Server** blocks from the **VEOS Ethernet Blockset Solution** using the Simulink Library browser.

You then connect the blocks with the **Connection Settings** blocks before building the Simulink implementation container (SIC) file in the Simulink coder.

You can then create a new offline simulation application in VEOS and import the SIC file.

To learn to work with sensor simulation using a VEOS platform to connect to MotionDesk and the SensorSim application, refer to [Lesson 2: Sensor Simulation Using a VEOS Platform \(Sensor Simulation Tutorial !\[\]\(9dfdaff1d86ba3c1f8353b4d1b61b8c5_img.jpg\)](#)). This also includes steps to adapt the model and add a connection to a SensorSim application that is used to create the sensor data for over-the-air sensor simulations and to connected sensor hardware.

Prerequisites

To work with an ASM model for sensor simulation, you must add the **Animation Interface** block in the ASM model. Refer to [How to Prepare the ASM Model with the Animation Interface Subsystem](#) on page 36.

Note

The MotionDesk Blockset used in the ASM Model created by ModelDesk does not support Sensor Simulation.

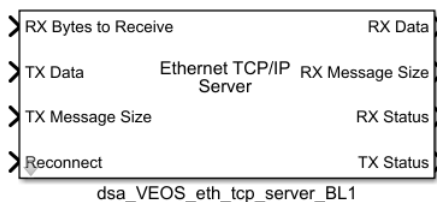
Method

To connect a VEOS platform to a single MotionDesk observer

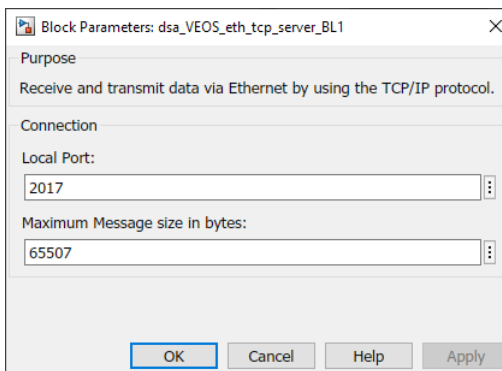
- 1 In MATLAB, open the ASM model that you created in ModelDesk for a VEOS platform. The model is stored in the **Simulation** folder of the ModelDesk project.
Run the `go.m` MATLAB file to open the model.
- 2 Double-click the **Animation Interface** block to open the block.
- 3 In the Simulink library, select the **dSPACE [SOL] VEOS Ethernet Blockset**. Select the **Ethernet TCP/IP Server** (`dsa_VEOS_eth_tcp_server_BL1`) block and press **Ctrl I** to insert the block into the ASM model.

Note

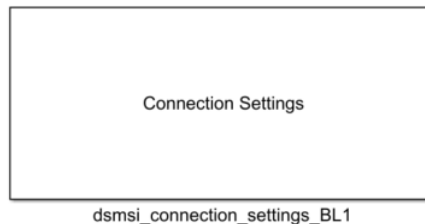
The TCP block is the server as seen by the simulation platform. MotionDesk is the client.



- 4 Double-click the Ethernet TCP/IP Server block and set the block parameters, for example, as shown.



- 5 In the Simulink library, select dSPACE MSI Blockset. Select the Connection Settings block and press **Ctrl I** to insert the block into the ASM model.



- 6 Double-click the Connection Settings block and in the parameters dialog, specify the name, for example, **MotionDesk**. You can also specify the Tx rate on how often MotionDesk gets the simulation data.
- 7 In Simulink, connect the Ethernet TCP/IP Server block ports to the Connection Settings block of the Model and Sensor Interface Blockset as shown in the illustration:

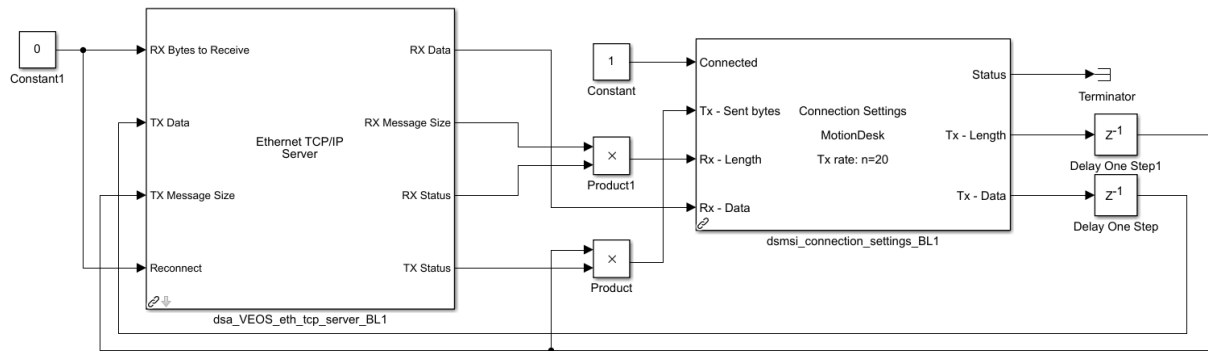
CAUTION

You must also add the constant, product, delay, and terminator functions from the Simulink library.

The following component settings must also be configured:

- **Constant:** The Constant component linked to the Connected port of the Connection Settings block must be set to **[1]** to initialize the connection to the system or application.
- **Output data type:** The Product component linked to the Rx - Length port of the Connection Settings block must be set to Inherit: Inherit via back propagation in the block parameters.

If the Connected inport of the Connection Settings block receives a **[0]**, the connection to connected system is disconnected and the initialization of the connection expires. The initialization process repeats with the next connection.



- 8 Open the Simulink Model Settings dialog for the model. In Code Generation, select the dsrt.tlc dSPACE Run-Time Target.
- 9 On the C-Code page, click the build icon  to build the model or click CTRL+B to create the SIC file.

Result

You connected a VEOS platform and the simulation application to a single MotionDesk observer using the Model and Sensor Interface Blockset and VEOS Ethernet Blockset Solution functions.

Related topics

Basics

[Lesson 2: Sensor Simulation Using a VEOS Platform \(Sensor Simulation Tutorial !\[\]\(003082e50e3009141f59bd5df831749f_img.jpg\)\)](#)

HowTos

[Connecting Simulation Platforms Using Ethernet Functions..... 42](#)

[How to Prepare the ASM Model with the Animation Interface Subsystem..... 36](#)

References

[Block Description \(dmsi_connection_settings\)..... 93](#)

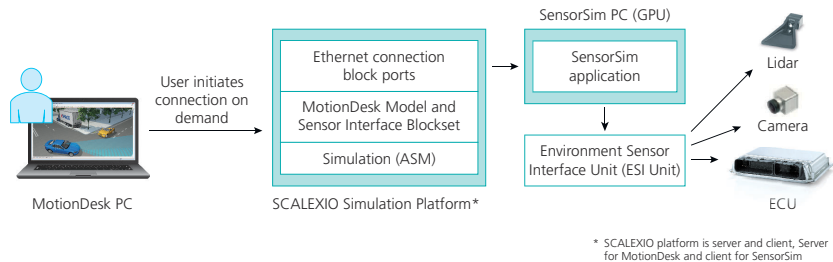
How to Connect a Simulation Platform to a SensorSim Application

Objective

You can connect a simulation platform to a MotionDesk observer and to a SensorSim application for sensor simulations.

Illustration

The illustration shows an overview of the connection of a simulation platform to a MotionDesk observer and to a SensorSim application on a SensorSim PC.



This diagram also shows the Environment Sensor Interface Unit that splits and synchronizes the data from the SensorSim application for each sensor and delivers it to the relevant sensor hardware. The model can also be configured to receive sensor status and feedback information on the connected sensors from the Environment Sensor Interface Unit.

Note

The connection of the Environment Sensor Interface Unit is not described in the steps in this document as it is custom specific.

Connection Overview

An overview of connection is as follows:

- You use the single MotionDesk PC client to visualize the simulation and to initiate the TCP/IP connection to the simulation platform.
The TCP function is the server as seen by the simulation platform. MotionDesk is the client.
- The SCALEXIO platform runs the real-time application that provides the simulation data to MotionDesk through the **Model and Sensor Interface Blockset**.
- The simulation platform also initiates the TCP/IP connection to the SensorSim application that generates raw sensor data through the **Model and Sensor Interface Blockset**.
The simulation platform becomes a TCP client and the SensorSim application is the server.
- In ConfigurationDesk, you import the model, add the Ethernet functions, specify the hardware, and extend the signal chain to create the model port blocks.
You then propagate the blocks to the ASM Simulink model and connect them with **Connection Settings** blocks that you add from the **Model and Sensor Interface Blockset**. You can then build the real-time application.

To learn to work with sensor simulation using a SCALEXIO platform to connect to MotionDesk and the SensorSim application, refer to [Lesson 3: Sensor Simulation on a SCALEXIO Platform \(Sensor Simulation Tutorial\)](#).

Prerequisites

You must first connect the SCALEXIO platform to MotionDesk. Refer to [How to Connect a SCALEXIO Platform to a Single MotionDesk Observer](#) on page 69.

The steps describe how to add a connection to a SensorSim application.

Method**To connect a simulation platform to MotionDesk and to a SensorSim application**

- 1 Open the ConfigurationDesk project and application that you used in [How to Connect a SCALEXIO Platform to a Single MotionDesk Observer](#) on page 69. This is stored in the Simulation folder of the ModelDesk project.
- 2 In the Function browser, add an additional TCP function to the signal chain and display it in the Signal Chain browser:
An Ethernet Setup and a TCP function running as a server for the MotionDesk client are already configured in this project.
- 3 Select the TCP function and specify the following General and Model Interface properties for the ports:
 - TCP

Property	Example value
Name	TCP (2) SensorSim (Server)
Run as server/client	Client <i>The TCP function is the client as seen by the simulation platform. The SensorSim application is the server.</i>
Dynamic destination	Disabled
Destination IP address	192.168.1.2. The IP address of the SensorSim PC, for example, 192.168.1.2.
Destination port	2017 The port number of the SensorSim PC, for example, 2017.
Local port configuration	Auto
TCP on Ethernet	192.168.1.1 Select the Ethernet setup function for the simulation platform.

- TCP Status/Control function

Port property	Value
Connection Request - Initialization and Termination - Initial value	Connect This initializes the connection.

▪ TCP Transmit function

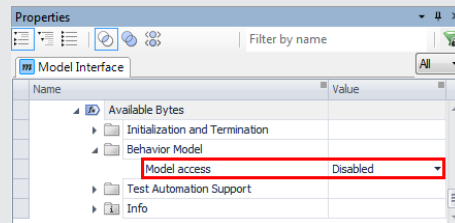
Property	Value
Maximum vector size	65507
Variable vector size	Defined by model
Message send mode	No delay

▪ TCP Receive function

Property	Value
Maximum vector size	65507
Variable vector size	Enabled
Message send mode	All available

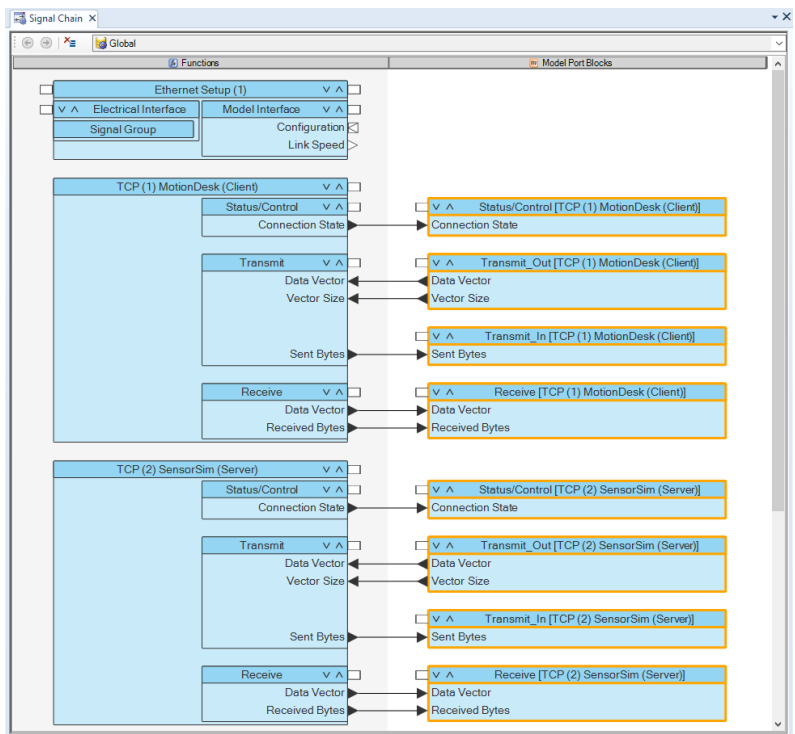
Tip

In Model Interface - Properties, disable Behavior Model - Model Access for the ports that are not required for the connection.

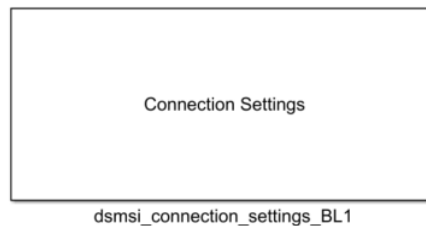


- 4 Select the TCP SensorSim (Server) function and extend the signal chain to create model port blocks for the TCP functions.

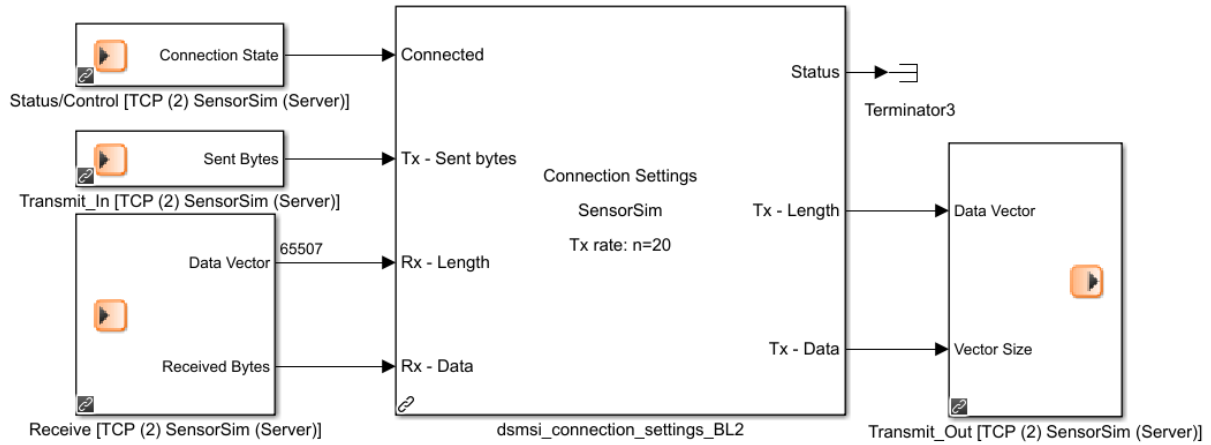
The ConfigurationDesk Functions and Model Port Blocks look like this in the ConfigurationDesk Signal Chain view:



- 5 Select the SensorSim (Server) TCP function and propagate the Model Port Blocks to the open ASM model in Simulink.
- 6 In the Simulink library, select dSPACE MSI Blockset.
Select the Connection Settings block and press **Ctrl I** to insert the block into the ASM model.



- 7 Double-click the Connection Settings block and in the parameters dialog, specify the name, for example, **SensorSim**. You can also specify the Tx rate on how often the SensorSim application gets the simulation data to calculate the sensor data.
- 8 In Simulink in the ASM model, connect the Ethernet model port blocks to the Connection Settings block of the Model and Sensor Interface Blockset as shown in the illustration:



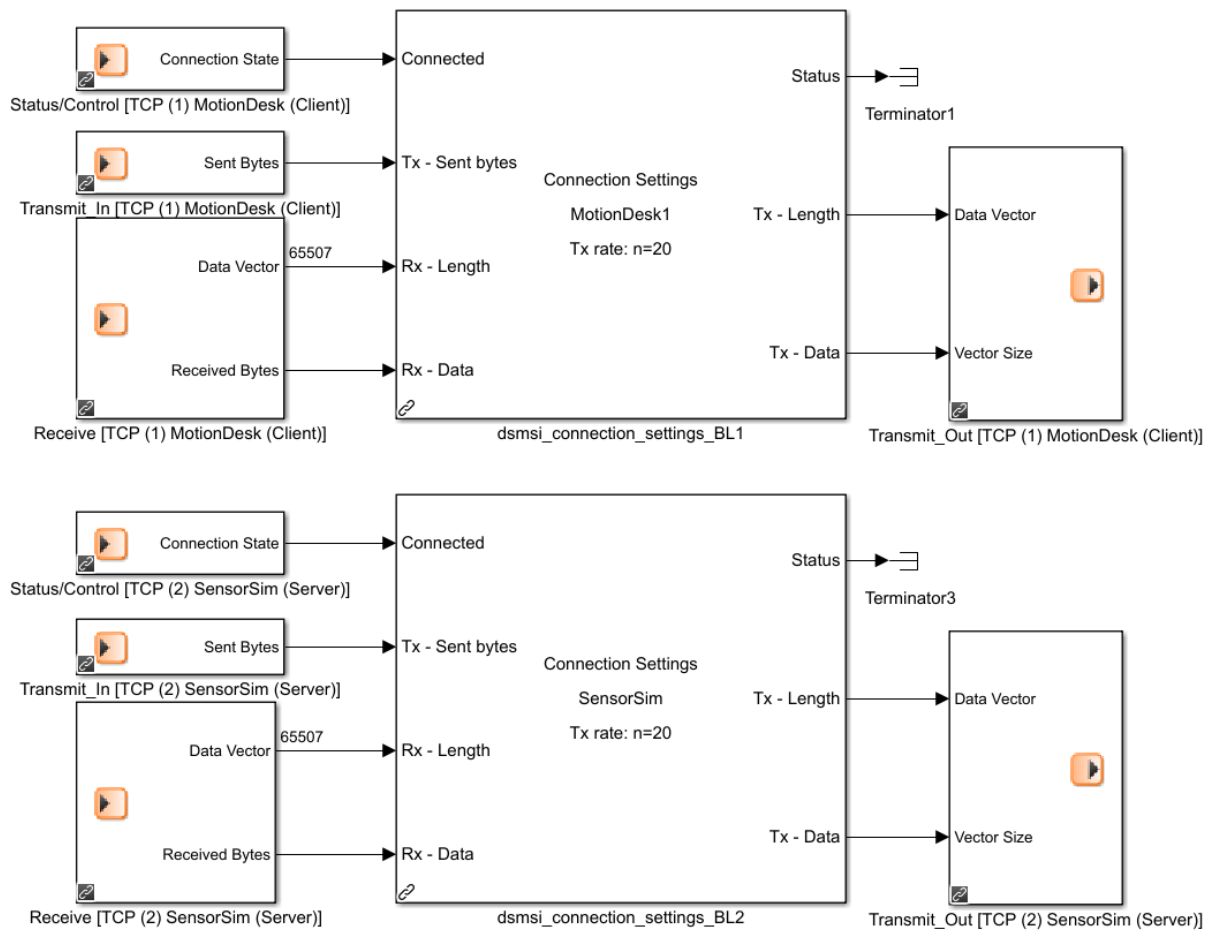
9 In the Connection State parameters dialog, change the signal properties data Type to Boolean.

10 Build the model in ConfigurationDesk to create the real-time application.

Result

You connected a SCALEXIO platform and the simulation application to a MotionDesk observer and to a SensorSim application on the SensorSim PC using the Model and Sensor Interface Blockset and ConfigurationDesk Ethernet functions.

The prepared model looks as follows:



Related topics

Basics

[Lesson 3: Sensor Simulation on a SCALEXIO Platform \(Sensor Simulation Tutorial !\[\]\(83f22ed94ec5517769dd76d702c6bfd8_img.jpg\)\)](#)

HowTos

[Connecting Simulation Platforms Using Ethernet Functions..... 42](#)
[How to Prepare the ASM Model with the Animation Interface Subsystem..... 36](#)

References

[Block Description \(dsmsi_connection_settings\)..... 93](#)

Reference Information

Model and Sensor Interface Blockset

Introduction	Provides reference information on the blocks of the Model and Sensor Interface Blockset.
---------------------	--

Where to go from here

Information in this section

[General Information on the Model and Sensor Interface Blockset..... 90](#)

Provides an overview of the Model and Sensor Interface Blockset used for Sensor Simulation.

[Connection Settings Block..... 93](#)

To connect a Sensor Simulation system or application to the Simulink model.

[Simulation Data Objects Block..... 100](#)

To create the Simulation Data Objects and calculate the transformation matrices for the objects used in the simulation.

[System Status Block..... 115](#)

To configure the System Status block that provides status and statistical data from the connected system, for example, MotionDesk or the SensorSim application.

[Sensor Failure Block..... 120](#)

To enable or disable sensor failures on the connected sensor device, for example, pixel errors, and specify the arguments for the failures on specific sensors connected via the Environment Sensor Interface Unit.

[Sensor Feedback Block..... 124](#)

To configure the Sensor Feedback block that provides feedback data for the simulated sensor from the Environment Sensor Interface Unit.

[Model and Sensor Interface Blockset Demo Library..... 128](#)

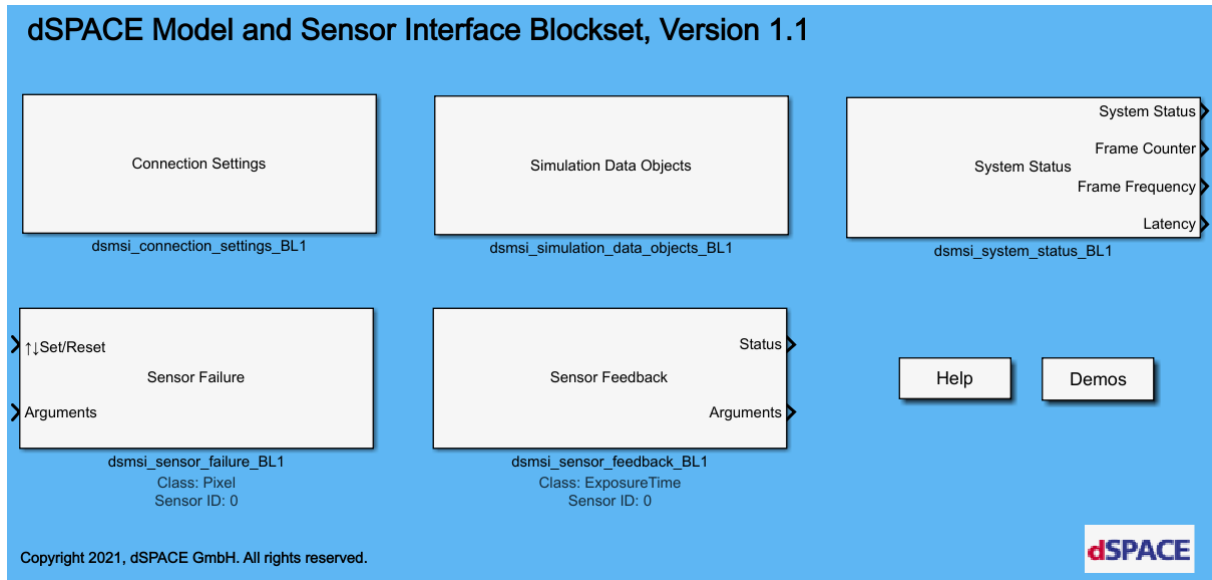
A library with demo models is provided.

General Information on the Model and Sensor Interface Blockset

Introduction	Provides an overview of the Model and Sensor Interface Blockset used for Sensor Simulation.
---------------------	---

Model and Sensor Interface Blockset Library Overview

Illustration



Library components

The following blocks are available in the Model and Sensor Interface Blockset:

- Connection Settings Block
- Simulation Data Objects Block
- System Status Block
- Sensor Failure Block
- Sensor Feedback Block

A demo library is also included. For example, the Coordinate System Demo for SCALEXIO, which includes a Connection Settings block example with Ethernet functions. For more information, refer to [Model and Sensor Interface Blockset Demo Library](#) on page 128.

Block priority

Default priorities are applied to the blocks of the Model and Sensor Interface Blockset. Simulink tries to adhere to these priorities to control the block execution order for the model.

⚠ WARNING

Simulink warning

A warning is displayed in Simulink if Simulink cannot adhere to the block priorities when generating the code, for example, if there is a conflict with the data dependencies.

The Connection Settings blocks are implemented with the lowest default priority relative to other blocks. Therefore the simulation data is transmitted to the Connection Settings block from the other blocks, before the Connection Settings block transmits the data to the connected system or application.

When you add a block from the Model and Sensor Interface Blockset library, the default block priority is used. The default block priorities are as follows:

Block Type	Priority ¹⁾
Simulation Data Objects	10001
Sensor Failure	10002
Sensor Feedback	10003
System Feedback	10004
Connection Settings	10009

¹⁾ The lower the number, the higher the priority.

If you copy a block from an existing model, the same block priority is used for the copied block.

You can change the block priorities in the Simulink Block Priorities dialog. Right-click the block and select Properties in the context menu to load the block properties dialog.

You can also set the priorities using the standard MATLAB function as follows:
`set_param(<block_name>, 'priority', '<value>')`

WARNING

Model and Sensor Interface Blockset warning

If you change the default Model and Sensor Interface Blockset block priorities you must ensure that the priorities of the Connection Settings blocks remain lower than the other blocks of the Model and Sensor Interface Blockset.

When you build, start, or update the model, the blockset checks the consistency of the block priorities. A warning is displayed if the block priorities in the model are not set according to the rules.

Related topics**Basics**

[Blockset Supported Platforms..... 16](#)

HowTos

[How to Access the Model and Sensor Interface Blockset..... 32](#)

References

[Connection Settings Block..... 93](#)
[Sensor Failure Block..... 120](#)
[Sensor Feedback Block..... 124](#)
[Simulation Data Objects Block..... 100](#)
[System Status Block..... 115](#)

Connection Settings Block

Purpose

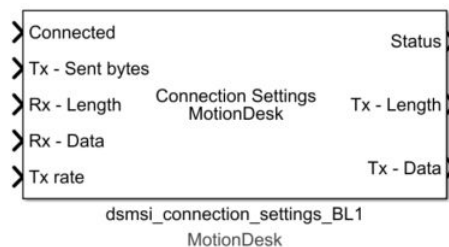
To connect a Sensor Simulation system or application to the Simulink model.

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

[Block Description \(dsmsi_connection_settings\)..... 93](#)
 To understand the Connection Settings block inports and outputs.

[Block Parameters Page \(dsmsi_connection_settings\)..... 98](#)
 To define the parameters of the Connection Settings block for each system or application.

Block Description (dsmsi_connection_settings)

Illustration

Purpose

To understand the Connection Settings block inports and outports.

Description

The Connection Settings (dsmsi_connection_settings) block handles the connection to the Sensor Simulation system or application, for example, MotionDesk, a SensorSim application, or an Environment Sensor Interface Unit (ESI Unit).

For each connected system or application, you must have a Connection Settings block, connected by Ethernet functions using TCP based communication. The Connection Settings blocks are also automatically connected to the Simulation Data Objects blocks that store the calculated simulation data.

You can also connect Sensor Failure, Sensor Feedback, and System Status blocks to the Connection Settings blocks, depending on the connected system or application.

I/O characteristics

Inports The following table shows the block inports:

Simulink Input	Simulink Data Type	Description
Connected	Boolean double single Int8 UInt8 Int16 UInt16 Int32 UInt32	Indicates if a TCP connection to the system or application is established: <ul style="list-style-type: none"> 1: Connected 0: Not connected
Tx – Sent bytes	double	Indicates how many bytes were already sent
Rx – Length	UInt32	Indicates how many bytes were received via TCP and are available on the Rx – Data input port.
Rx – Data	UInt8	Indicates the vector of bytes (uint8) with the received payload.
Tx rate	double single Int8 UInt8 Int16 UInt16 Int32 UInt32	The Tx rate inport receives a numerical value that specifies the ratio between the transmission sample time and the block sample time. If Use input port is cleared in the block parameters dialog, this port is not shown on the block mask. The transmission rate is then

Simulink Input	Simulink Data Type	Description
		<p>defined manually in the parameters dialog and shown in the block mask.</p> <div> <p>Note</p> <p>The transmission rate is the rate at which the simulation data is sent to the Connection Settings block outputs: Tx - Length and Tx - Data.</p> <p>The block sample time is sample time at which the block is sampled. The block inherits the sample time from the connected signals. All Model and Sensor Interface Blockset blocks must have unique sample time.</p> <p>For example, if the block sample time is 1 ms and the Tx rate value is 20, then the transmission sample time is 20 ms. Therefore, the transmission rate of 1/20 ms is equal to 50 Hz.</p> </div> <div> <p>Note</p> <p>If the Tx rate inport is not connected, a warning is displayed when building or running the model. The transmission rate is then defaulted to 1. This also applies if zero or a negative Tx rate value is provided as a signal through the inport during the simulation.</p> <p>The blockset also validates the Connected and Tx rate inport signal data type and dimensions.</p> </div>

Outputs The following table shows the block outputs:

Simulink Output	Simulink Data Type	Description
Status	double	<p>Outputs the status of the Model and Sensor Interface Blockset.</p> <p>Common blockset status and error codes</p> <ul style="list-style-type: none"> ▪ 0: Success ▪ 1: New data ▪ 2: Error - Out of memory ▪ 3: Error - Data output too short

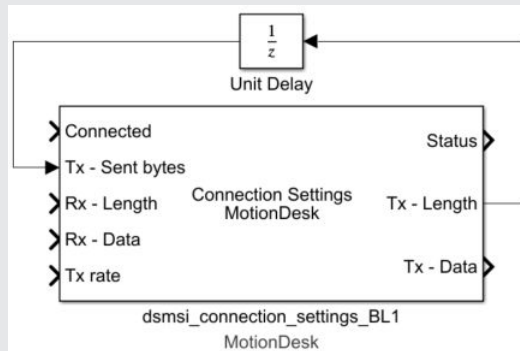
Simulink Output	Simulink Data Type	Description
Tx – Length	Uint32	<ul style="list-style-type: none"> ▪ 4: Error - Not all data was transmitted to destination ▪ Other: Unknown error <p>Indicates how many bytes are ready for transmission on the Tx – Data output port. It is refreshed at a fixed transmission rate defined in the block parameters dialog.</p> <div> <p>Note</p> <p>The transmission rate is the rate at which the simulation data is sent to the Connection Settings block outputs: Tx - Length and Tx - Data.</p> <p>The block sample time is sample time at which the block is sampled. The block inherits the sample time from the connected signals. All Model and Sensor Interface Blockset blocks must have unique sample time.</p> <p>For example, if the block sample time is 1 ms and the Tx rate value is 20, then the transmission sample time is 20 ms. Therefore, the transmission rate of 1/20 ms is equal to 50 Hz.</p> </div>
Tx – Data	Uint8	Indicates the vector of bytes with the payload for transmission that is sent at a fixed transmission rate defined in the block parameters dialog.

Tip

If your Ethernet TCP blockset does not have an output port that defines the number of transmitted bytes, you can connect the two ports through a Unit Delay block:

- Tx - Sent Bytes input port
- Tx - Length output port

You can see an example of the setup in a Simulink model:



In this connection, the Connection Settings block cannot detect if all data is successfully transmitted to its destination.

Dialog pages

The parameters can be specified on the following dialog pages:

- Block Parameters Page (dsmsi_connection_settings)

Related topics**Basics**

[Network Concept \(SCALEXIO Hardware Installation and Configuration !\[\]\(4b7a79268f6ba26c1471d4232fffa85a_img.jpg\)](#))

HowTos

[Connecting Simulation Platforms Using Ethernet Functions..... 42](#)
[How to Connect a Simulink Model to a Sensor Simulation System or Application..... 39](#)

References

[Sensor Failure Block..... 120](#)
[Sensor Feedback Block..... 124](#)
[Simulation Data Objects Block..... 100](#)
[System Status Block..... 115](#)

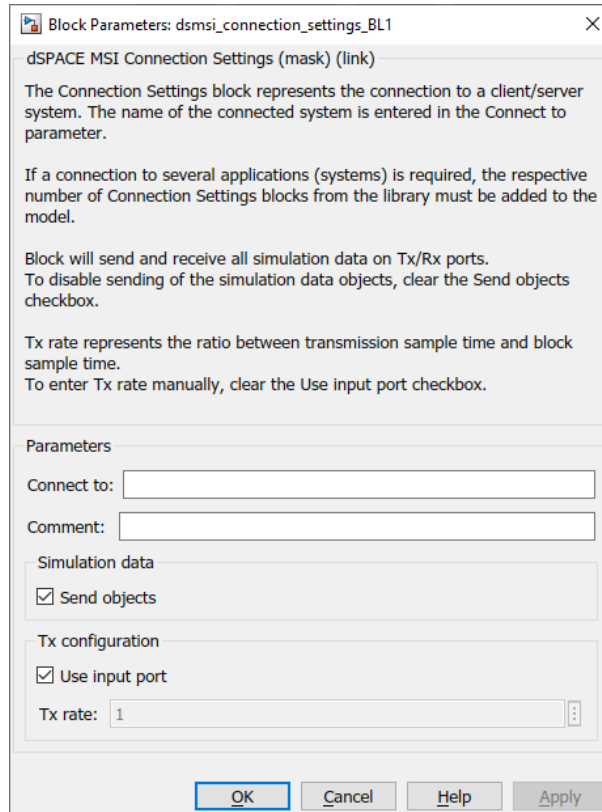
Block Parameters Page (dsmsi_connection_settings)

Purpose To define the parameters of the Connection Settings block for each connected system or application.

Dialog settings

Block parameters dialog To load the Block Parameters dialog, double-click the Connection Settings block in Simulink.

The Block Parameters: dsmsi_connection_settings dialog is displayed.



Application parameters The table specifies the application parameters for the Connection Settings block.

Parameter	Description
Connect to	<p>Type the name of the Sensor Simulation system or application to which this Connection Settings block connects through an Ethernet blockset. This is displayed on block mask, for example:</p> <ul style="list-style-type: none"> ▪ MotionDesk ▪ SensorSim application ▪ Environment Sensor Interface Unit <p>The name must be completed to display the name, description, and ports in the block after you close the parameters dialog.</p>

Parameter	Description
Comment	Add a description label the connection in Comment , for example, Observer or Camera front . This comment is displayed under the block name label in Simulink.
Send objects	Each Connection Settings block sends all simulation data calculated by the Simulation Data Objects blocks to the connected system if the Send objects checkbox is selected in the Connection Settings block parameters. Clear the checkbox to prevent sending simulation data to the connected system.
Use input port	If Use input port is selected, the transmission rate is determined as a signal from the input port while running the simulation. The Tx rate input is displayed in the block mask. Clear Use input port to specify the transmission rate manually in Tx rate . The port is not shown on the block mask.
Tx rate:	Type the Tx rate as a numerical value that specifies the transmission rate at which the simulation data is transmitted to the Connection Settings ports. The Tx rate is the ratio between the transmission sample time and the block sample time. This parameter is disabled if Use input port is selected. <div data-bbox="730 915 1398 1346"> <p>Note</p> <p>The transmission rate is the rate at which the simulation data is sent to the Connection Settings block outputs: Tx - Length and Tx - Data.</p> <p>The block sample time is sample time at which the block is sampled. The block inherits the sample time from the connected signals. All Model and Sensor Interface Blockset blocks must have unique sample time.</p> <p>For example, if the block sample time is 1 ms and the Tx rate value is 20, then the transmission sample time is 20 ms. Therefore, the transmission rate of 1/20 ms is equal to 50 Hz.</p> </div>

Related topics

References

Block Description (dsmsi_connection_settings)..... 93

Simulation Data Objects Block

Purpose To create the Simulation Data Objects and calculate the transformation matrices for the objects used in the simulation.

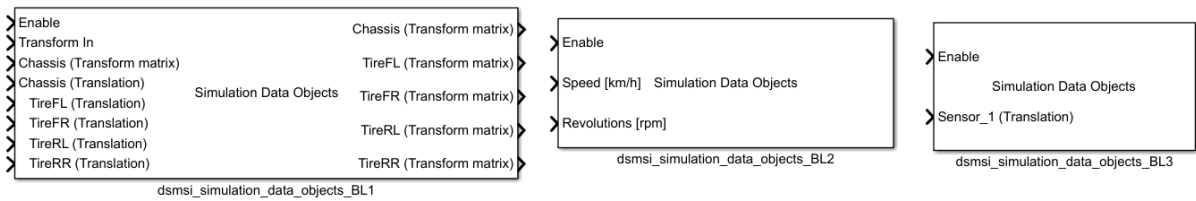
Where to go from here Information in this section

Block Description (dsmsi_simulation_data_objects)..... 100
To understand the inports and outports of the Simulation Data Objects block.

Block Parameters Page (dsmsi_simulation_data_objects)..... 104
To define the parameters of the Simulation Data Objects block for each object type.

Block Description (dsmsi_simulation_data_objects)

Illustration



Purpose To understand the inports and outports of the Simulation Data Objects block.

Description The Simulation Data Objects store the calculated data for the objects defined in the simulation, for example, the motion transformation data for the 3-D objects and sensor points or signal and instrument data. The Connection Settings block prepares the data generated by the Simulation Data Objects for transmission by the Ethernet functions to the connected systems and applications.

For each of the moving objects in the simulation, simulation data objects can be defined in kinematic chains in a single block using a parent-child relationship. You can also specify the number of objects that are included for each transformation object in the Simulation Data Objects block, for example, for multiple fellows in a simulation.

For more information, refer to [Block Description \(dsmsi_connection_settings\)](#) on page 93.

I/O characteristics

Object tree parameters The following parameters apply to the objects block and all of the objects in the object tree:

Simulink Input	Range	Simulink Data Type	Meaning
Enable		Boolean double single Int8 Uint8 Int16 Uint16 Int32 Uint32	Enable and disable the transmission of the data from the block. If you select Enable in the dialog, the Enable inport is created. The transmission of the simulation data from block then depends on the value at this inport: <ul style="list-style-type: none"> ▪ 0: The transmission of the simulation data is disabled. ▪ Any other value: The transmission of the simulation data is enabled. If you clear Enable in the dialog, the Enable inport is not created and the simulation object data is always sent.
Transform In	3×4 or 4×4 matrix	double	If Transform In is selected in the dialog, the reference starting transformation matrix for all of the movable objects of the block is provided via the inport. The port is displayed in the block mask. If the inport is not active, the unit [4×4] matrix is used as a reference starting transformation matrix.

Transformation object type The following table shows the possible block input ports for a 3-D object:

Simulink Input	Range	Simulink Data Type	Meaning
Transformation matrix	3×4[n] or 4×4[n] matrix	double	The transformation matrix is provided via the inport for this specific object. The input port receives a signal with the vector matrix dimensions of [4×4n] or [3×4n], where [n] is the number of objects specified. This is the base matrix for each of the multiple objects in the specific transformation object and the children.
Translation	1×3[n] vector	double	The input port is used to define the object's movement and initial offset rotation from its parent. It must be a

Simulink Input	Range	Simulink Data Type	Meaning
Rotation	$1 \times 3[n]$ vector	double	<p>vector with three elements x, y, and z. The values must be in meters.</p> <p>The input port receives a signal for the translation inputs of the specific object, where [n] is the number of objects specified, for example, Ax, Ay, Az, Bx, By, Bz, Cx, Cy, Cz.</p> <p>The input port is used to define the object's rotation and initial offset from its parent. It must be a vector with three elements x, y, and z. Their values must be in degrees. You can define the angles in Euler or Cardan angle format.</p> <p>The input port receives a signal for the rotation inputs of the specific object, where [n] is the number of objects specified.</p>
Scaling	$1 \times 3[n]$ vector	double	<p>The input port defines the scaling factors of the object and initial offset scaling from its parent. . It must be a vector with three elements x, y, and z. The elements are factors. No scaling is applied with values of [1, 1, 1].</p> <p>The input port receives a signal for the scaling inputs of the specific object, where [n] is the number of objects specified.</p>

The following table shows the block output port:

Simulink Output	Range	Simulink Data Type	Meaning
Transform matrix	$4 \times 4[n]$ matrix	double	<p>The calculated transformation matrix can be output and used as an input to other Simulation Data Objects blocks.</p> <p>The output port transmits a signal with the vector matrix dimensions of $[4 \times 4n]$, where [n] is the number of objects specified. It outputs a concatenated matrix that contains a number of matrices</p>

Simulink Output	Range	Simulink Data Type	Meaning
			corresponding to the number of objects specified.

Signal object type The following table shows the block input ports for a signal object type:

Simulink Input	Range	Simulink Data Type	Meaning
Object name	1x3 vector	Boolean double single Int8 Uint8 Int16 Uint16 Int32 Uint32	If the simulation data object of the type Signal is created, the input port represented with the object name is always created in the block. The input can be a scalar or vector using a specific data type, for example, uint8, int8, int32, double, single, Boolean or inherited via the Simulink back propagation function.

Position object type The following table shows the block input ports for a position object type:

Simulink Input	Range	Simulink Data Type	Meaning
Translation	1x3[n] vector or 3x[n] matrix	double	If a simulation data object with the type Position is created, the input port represented by the object name (Translation) is always created in the block. If the input is a vector 1*3n , 3 represents the three axis and n is a multiple, for example, x1, y1, z1, x2, y2, z2, xn, yn, zn. If the input is a matrix 3*n , 3 represents the three axis and n is a multiple greater than 1.

Position and scale object type The following table shows the block input ports for a position and scale object type:

Simulink Input	Range	Simulink Data Type	Meaning
Translation	1x3 vector	double	The translation input port is created in the block with the object name (Translation) if the input port is selected in the dialog.
Scaling	1x3 vector	double	The scaling input port is always created in the block with the object name (Component).

High precision feature

The accuracy of the T-matrices must be increased for large worlds. Therefore, the **Simulation Data Objects** block supports the option of switching on and off high-precision feature in the T-matrices.

This can be done using the following commands in the MATLAB Command Window:

Feature	Command
On	set_param (gcb, 'highPrecision', '1')
Off	set_param (gcb, 'highPrecision', '0')

Dialog pages

The dialog settings can be specified on the following dialog pages:

- **Block Parameters Page** (dsmsi_simulation_data_objects)

Related topics**HowTos**

[How to Configure the Simulation Data Objects in a Simulation Model..... 51](#)

References

[Connection Settings Block..... 93](#)
[Sensor Failure Block..... 120](#)
[Sensor Feedback Block..... 124](#)
[System Status Block..... 115](#)

Block Parameters Page (dsmsi_simulation_data_objects)

Purpose

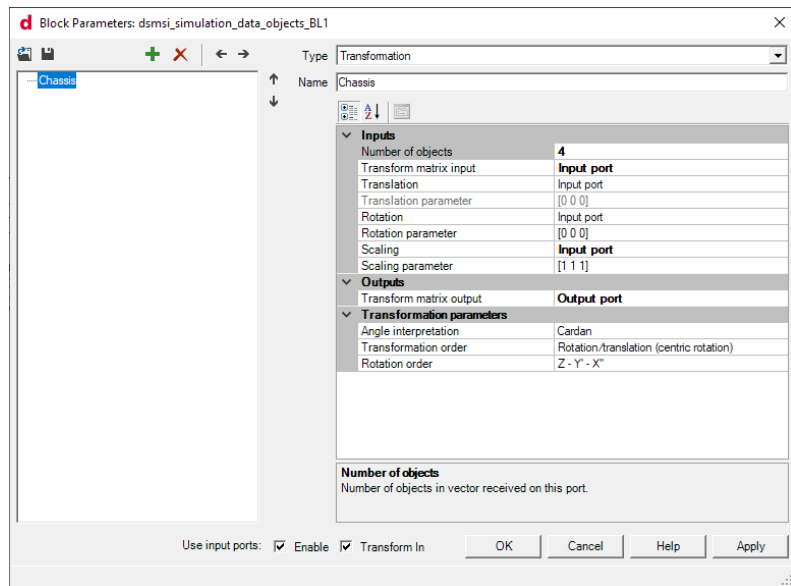
To define the parameters of the **Simulation Data Objects** block for each object type.

Dialog settings

Block parameters dialog To open the Block Parameters dialog, double-click the **Simulation Data Objects** block in Simulink.


The **Block Parameters: dsmsi_simulation_data_objects** dialog is displayed.


The dialog shows the object tree on the left and the property pane for each object on the right. A single transformation object **Chassis** is added in the dialog.




Note


To open the dialog, the simulation must be stopped.
The parameters dialog cannot be opened if the block name contains invalid characters. For more information, refer to [Limitations When Using the Model and Sensor Interface Blockset](#) on page 137.


Import  Import the simulation data objects configuration from an existing block that you previously exported to a dSeb (dSPACE ESI blockset file) or to an XML file. The dSeb file is in XML format.

Export  Export your simulation data objects configuration to a dSeb (dSPACE ESI blockset file) or from an XML file. You can then import this configuration file to a new Simulation Data Objects block. The dSeb file is in XML format.

Add  Add a new object to the simulation data objects tree. The object is added below the selected entry on the same level.

Delete  Delete an object from the Simulation Data Objects block tree.

Up and down arrows  You can use the up and down arrows to move the objects up and down within the same level in the simulation data objects tree.

Left and right arrows  You can use the left and right arrows to move the simulation data objects up and down in the hierarchical structure in the object tree. This creates a parent child relationship between objects and sub-

objects, for example, a vehicle and the tires. In the Simulink model, the children of a parent object are shown as indented in the block mask.

Type You can select the simulation data object type in the list:

- **Transformation:** To calculate the transformation matrices from the translation, rotation and scaling vectors for 3-D objects on a 3-D plane. For example, a car chassis and its four tires.
- **Signal:** To handle signals for any scalar value, for example, speed, acceleration, engine revolutions, and vehicle lights.
- **Position:** To display the contact points of the sensors in the environment, for example, the 3-D point cloud sensor points.
- **Position and Scale:** To display the position and scale of a 3-D object, for example, the 3-D vectors that represent the active forces arrows at a point on the 3-D Object.

Note

If you change the object type and select OK or Apply, the connected lines are removed from the object imports in the block mask.

For child objects, the allowed type depends on the parent object type. The following combinations are possible:

Parent object	Allowed child objects
Transformation	All types are permitted
Transformation [n]	Signal [n]
Signal	Signal
Position	Position Signal
Position and scale	Position and scale Signal

Note

If the number of transformation objects [n] is more than 1, only Signal object types can be added as children to the transformation object. The signal child object transmits the same number of objects that is specified for the parent transformation object.

Name You can enter a name for the simulation data object. Spaces are permitted.

For information on the support characters in a block name, refer to [Limitations When Using the Model and Sensor Interface Blockset](#) on page 137

The names of the root level objects must be unique in the model. Child object names inside a parent object must also be unique.

Note

You can use the **ManeuverState** name to generate the simulation start time of an ASM maneuver. You can use this in the MotionDesk Multi-Track feature.

Use input ports: The following checkboxes are used to control the input data:

Enable The **Simulation Data Objects** block calculates and stores the data that is prepared by the Connection Settings blocks for transmission via Ethernet functions. You can enable and disable the transmission of the data from the block.

If you select **Enable**, the **Enable** inport is created in the block mask. The transmission of the simulation data from the block then depends on the value provided at this inport:

- **0:** The transmission of the simulation data is disabled.
- **Any other value:** The transmission of the simulation data is enabled.

If you clear **Enable**, the **Enable** inport is not created in the block mask and the simulation object data is always sent.

Transform In You can provide a reference position from which each movable object of the **Simulation Data Objects** block is calculated.

If **Transform In** is selected, the reference starting transformation matrix for all the movable objects in the object block is provided via the inport.

If **Transform In** is cleared, the **Transform In** port is not created and instead the unit matrix is used as a starting transformation matrix. The objects are calculated based on the reference point, for example, where x , y , z are equal to 0.

Transformation object parameters

Inputs

Parameter	Simulink Data Type	Description
Number of objects	int	Specify the number of objects that are included for each root transformation object. This can be used, for example, for multiple fellows in a simulation without changing the model. It is therefore not necessary to add multiple identical

Parameter	Simulink Data Type	Description
Transformation matrix input	string	<p>transformation objects in the Simulation Data Objects dialog.</p> <p>You can specify from 1 to 99 objects. If you input a value outside this range, it defaults to the nearest value, 1 or 99.</p> <p>If a child object with the Signal type is added, it inherits the number of objects from the root transformation object, for example, the speed of each vehicle.</p> <p>For more information, refer to How to Configure the Simulation Data Objects in a Simulation Model on page 51.</p> <p>Select from the following:</p> <ul style="list-style-type: none"> ▪ Input port: The transformation matrix defines the relative position of the object in respect to its parent, which is provided by via the inport. The transformation matrix inport is created in the block mask. The input port receives a signal, which is multiplied by the number of objects as the reference position for each of the multiple objects. ▪ Disabled: The transformation matrix inport is removed from the block. The object's transformation matrix is calculated by multiplying the translation, rotation, and scaling matrix. This defines the position in respect to its parent.
Translation	string	<p>Select from the following:</p> <ul style="list-style-type: none"> ▪ Input port: the input port with the object name (Translation) is created in the block mask. The signal must be connected to the inport and provide translation vectors. The translation parameter field is disabled. The vector signal can be changed during simulation runtime. Where the number of objects is [n] and the translation, rotation, and scaling inputs are enabled, the corresponding input port must have the dimension [1x3n]. ▪ Parameter: Enable the Translation parameter field to specify a translation vector. The input port is not shown on the block mask.

Parameter	Simulink Data Type	Description
Translation parameter	1x3 vector of doubles	<p>Specify the translation vector for the position of the object.</p> <p>The translation vector must have 3 elements specified along the x, y, z axes in meters. The elements cannot be changed during the simulation. The default value is [0 0 0].</p> <p>If you specify multiple objects, the x, y, z-axis values are automatically applied to each object. For example, if the number of objects is 4 and the translation parameter is [3 -2.4 0.7], the internally generated vector is as follows:</p> <pre>[3 -2.4 0.7 3 -2.4 0.7 3 -2.4 0.7 3 -2.4 0.7]</pre>
Rotation	string	<p>Select from the following:</p> <ul style="list-style-type: none"> ▪ Input port: the input port with the object name and the angle interpretation is created in the block mask. For example, chassis (Rotation, ZYX). The signal must be connected to the inport and provide rotation vectors. <p>The rotation parameter field is disabled. The vector can be changed during simulation runtime.</p> <p>The input port receives a signal, which is multiplied by the number of objects as the reference rotation for each of the multiple objects.</p> <ul style="list-style-type: none"> ▪ Parameter: Enable the Rotation parameter field to specify a translation vector. The input port is not shown on the block mask.
Rotation parameter	1x3 vector of doubles	<p>Specify the rotation vector for the orientation of the object.</p> <p>The rotation vector must have 3 elements specified along the x, y, z-axes in [degrees]. The elements cannot be changed during the simulation. The default value is [0 0 0].</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>The Angle interpretation and Rotation order transformation parameters must be considered when specifying the rotation parameters.</p> </div> <p>If you specify multiple objects, the x, y, z-axis values are automatically applied to each object.</p>

Parameter	Simulink Data Type	Description
Scaling	string	<p>Select from the following:</p> <ul style="list-style-type: none"> ▪ Input port: the input port with the object name (Scaling) is created in the block mask. The signal must be connected to the inport and provide scaling vectors. The scaling parameter field is disabled. The vector can be changed during simulation runtime. The input port receives a signal, which is multiplied by the number of objects as the reference scaling for each of the multiple objects. ▪ Parameter: Enable the Scaling parameter field to specify a scaling vector. The input port is not shown on the block mask.
Scaling parameter	1x3 vector of doubles	<p>Specify the scaling vector for the size of the object.</p> <p>The scaling vector must have 3 elements specified as [multipliers] for each of the respective dimensions, along the x, y, z-axes. The elements cannot be changed during the simulation. The default value is [1 1 1], where no scaling is applied.</p> <p>If you specify multiple objects, the x, y, z-axis values are automatically applied to each object.</p>

Note

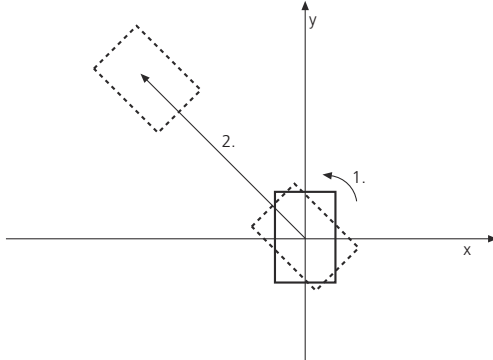
If you add invalid characters in the translation, rotation or scaling parameters, an error is displayed when you click OK or Apply.

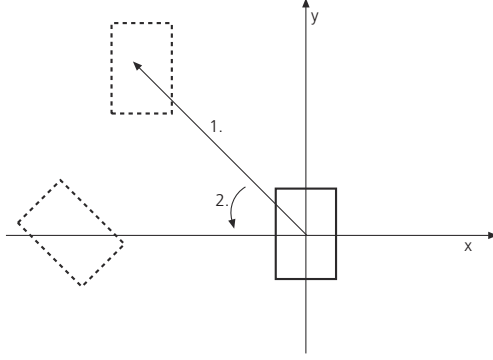
Outputs

Parameter	Simulink Data Type	Description
Transformation matrix out	string	<p>Select from the following:</p> <ul style="list-style-type: none"> ▪ Output port: The calculated transformation matrix is provided to other simulation data objects as an input in the form of a 4x4 matrix.

Parameter	Simulink Data Type	Description
		<p>The output port transmits a signal, which is multiplied by the number of objects as the reference position for each of the multiple objects.</p> <ul style="list-style-type: none"> ▪ Disabled: The transformation matrix output port is disabled and the calculated transformation matrix is not available to other Simulation Data Objects blocks. The input port is not shown on the block mask.

Transformation parameters

Parameter	Simulink Data Type	Description
Angle interpretation	string	<p>Specify the rotation angle interpretation as Cardan angles with roll, pitch, and yaw rotational movements or as Euler.</p> <p>If Euler is selected, the Rotation order is disabled as an edit field. For Euler angles, the rotation order is always Z - Y' - Z''.</p> <p>For more information, refer to Euler Angles on page 25.</p>
Transformation order	string	<p>Specify the sequence for the transformation as follows:</p> <ul style="list-style-type: none"> ▪ Rotation/translation (centric rotation): The object is first rotated and then it is moved to the new position as shown in the following diagram.  <ul style="list-style-type: none"> ▪ Translation/rotation (off centric rotation): The object is first moved to the new position and

Parameter	Simulink Data Type	Description
Rotation order	string	<p>is then rotated as shown in the following diagram.</p>  <p>Specify the sequence order of rotations per axes for the Cardan angle interpretation. You can select Z - Y' - X'' (default) or Z - X' - Y''.</p> <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Note</p> <p>Euler angles use the same axis for both the first and third rotations. The Euler angle interpretation the rotation order is always Z - Y' - Z''. Therefore, this parameter is disabled if the angle interpretation is Euler.</p> </div> <p>Z - Y' - X'': The 3-D object is rotated around its axes in the following manner:</p> <ol style="list-style-type: none"> 1. Yaw rotation around the z-axis with the Phi Φ angle. 2. Pitch rotation around the y-axis with the Theta θ angle. 3. Roll rotation around the actual x-axis with the Psi ψ angle. <p>The Pitch and Roll steps are performed around newly formed coordinates after the previous rotation completes.</p> <p>Z - X' - Y'': The 3-D object is rotated around its axes in the following manner:</p> <ol style="list-style-type: none"> 1. Yaw rotation around the z-axis with the Phi Φ angle. 2. Roll rotation around the x-axis with the Psi ψ angle. 3. Pitch rotation around the actual y-axis with the Theta θ angle. <p>For more information, refer to Cardan Roll, Pitch, and Yaw Angles on page 25.</p>

Signal object parameters

Inputs

Parameter	Simulink Data Type	Description
Number of objects	int	<p>If the signal object, for example, for the speed of a vehicle is a child object of a root transformation object, the number of objects is inherited from the parent transformation object. The field cannot be edited.</p> <p>If the signal object is not a child of a root transformation object, the value is always 1.</p> <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> <p>Note</p> <p>You can specify multiple objects only for a root transformation object.</p> </div>
Signal input	string	An input port with the object name [unit] is created in the block mask. This cannot be edited. The signal value is always provided by the input port.

Signal parameters

Parameter	Simulink Data Type	Description
Description	string	Specify a description for the signal. The field supports only printable ASCII characters (ASCII value 32-126) except " and \.
Unit	string	<p>Specify the unit for the signal, for example, km/h or Rpm. The unit is displayed in the MotionDesk instrument for the connected signal. The unit is also shown in the block mask.</p> <p>The field supports only printable ASCII characters (ASCII value 32-126) except " and \.</p>
Type	string	Specify the data type for the signal from the following types: single, double, Inherited (determined by Simulink built-in methods), Boolean, and Int8, 16, or 32 / UInt8, 16, or 32.
Vector length	int	<p>Specify the vector length of the signal.</p> <ul style="list-style-type: none"> ▪ 1: Scalar ▪ >1: Vector ▪ -1: Inherited <p>If the vector length -1 is specified for a signal, the signal length is determined by Simulink built-in methods.</p>

Position object parameters

Inputs

Parameter	Simulink Data Type	Description
Translation	string	An input port with the object name (Translation) is created in the block mask. The field is always set to Input port, and the signal must be connected to the inport. The translation vector is always provided by an inport, and the field cannot be edited.
Input port dimensions	string	Lets you specify the dimensions used for the input port: <ul style="list-style-type: none"> ▪ 1x3[n] vector ▪ 3x[n] matrix

Position and scale object parameters

Inputs

Parameter	Simulink Data Type	Description
Translation	string	Select from the following: <ul style="list-style-type: none"> ▪ Input port: The input port with the object name (Translation) is created in the block mask. The signal must be connected to the inport. The translation parameter field is disabled. The vector can be changed at simulation run time. ▪ Parameter: Enable the Translation parameter field to specify a translation vector. The Translation input port is hidden in the block mask.
Translation parameter	1x3 vector of doubles	Specify the translation vector for the position of the object. The translation vector must have three elements, along the x-, y-, z-axes. The elements are in [meters] and cannot be changed during the simulation. The default value is [0 0 0].
Scaling	string	An input port with the object name (Scale) is created in the block mask. The scaling vector must have three elements, along the x-, y-, z-axes. It is always provided from the input port of the simulation data object and cannot be changed during the simulation. The field is disabled.

Note

If you add invalid characters or a format that is not 1x3 vector of doubles in the translation parameter, an error is displayed when you click OK or Apply.

Related topics

Basics

Cardan Roll, Pitch, and Yaw Angles.....	25
Euler Angles.....	25
Simulation Data Mathematical Principles.....	22

References

Block Description (dsmsi_simulation_data_objects).....	100
--	-----

System Status Block

Purpose

To configure the System Status block that provides status and statistical data from the connected system, for example, MotionDesk or the SensorSim application.

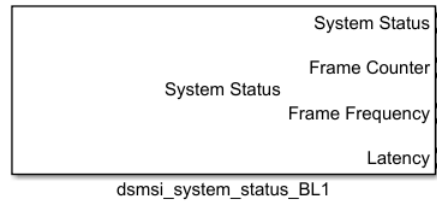
Where to go from here

Information in this section

Block Description (dsmsi_system_status).....	116
To understand the inports and outports of the System Status block.	
Block Parameters Page (dsmsi_system_status).....	118
To define the parameters of the System Status block for each connected system or application.	

Block Description (dsmsi_system_status)

Illustration



Purpose

To understand the inports and outports of the System Status block.

Description

The System Status block connects to a Connection Settings block to output status information provided by the connected system or application.

It defines the connection between the sensor data inputs and outputs and one element of the block output vectors.

- **System Status:** Outputs the status of the connected system.
- **Frame Counter:** Outputs the number of received or transmitted frames per channel.
- **Frame Frequency:** Outputs the frame frequency of the incoming data per channel.
- **Latency:** Outputs the calculated time taken in seconds to transmit the data from the blockset and receive feedback data from the connected system or application.

I/O characteristics

The following table shows the block outputs:

Simulink Input	Simulink Data Type	Description
Status	double	Outputs the received status of the connected system. The received status value is controlled by the connected system or application.
Frame Counter	double	Outputs the number of transmitted or received frames per channel.
Frame Frequency	double	Outputs the received frequency of the incoming data per channel.
Latency	double	Outputs the latency in [s] of the channel. The latency is the calculated time taken to transmit the data from the blockset and receive feedback data.

Dialog pages

The dialog settings can be specified on the following dialog pages:

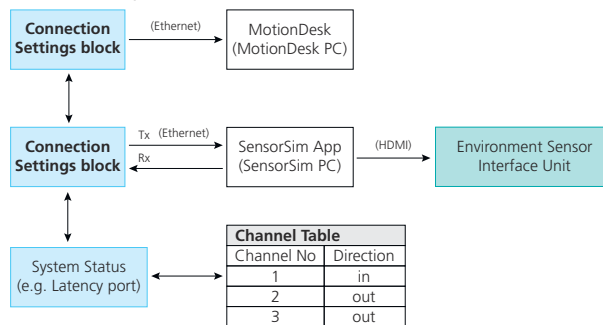
- **Block Parameters Page (dsmsi_system_status)** on page 118

Example

An example of a System Status block configuration is as follows:

In this example, the System Status block is connected to the Connection Settings block that connects to a SensorSim application to provide the statistical data required.

MotionDesk and the SensorSim application have one input for the simulation data from the simulation model. The SensorSim application has an output to an Environment Sensor Interface Unit. The statistical data is received by the same Connection Settings block from the Environment Sensor Interface Unit and the connected system.

**Note**

The channel numbers are defined by the connected system.

The System Status blockset Channel table matrix can be configured as follows:

Index	Channel no.	Direction
0	0	in
1	2	out
2	3	out

The three outputs, Frame Counter, Frame Frequency, and Latency provide vectors that have as many elements as defined in the mapped Channel Table.

Index	Channel	Direction	Description
0	1	in	Index 0 of the Frame Counter, Frame Frequency, and Latency output vectors is mapped to the first MotionDesk input for the simulation model simulation data.
1	1	out	Index 1 of the Frame Counter, Frame Frequency, and Latency output vectors is

Index	Channel	Direction	Description
2	3	out	<p>mapped to the first MotionDesk output for the MotionDesk PC monitor.</p> <p>Index 2 of the Frame Counter, Frame Frequency, and Latency output vectors is mapped to the second MotionDesk output for the Environment Sensor Interface Unit.</p>

Related topics

HowTos

[How to Collect Status Information using the System Status block..... 43](#)

References

[Connection Settings Block..... 93](#)
[Sensor Failure Block..... 120](#)
[Sensor Feedback Block..... 124](#)
[Simulation Data Objects Block..... 100](#)

Block Parameters Page (dsmsi_system_status)

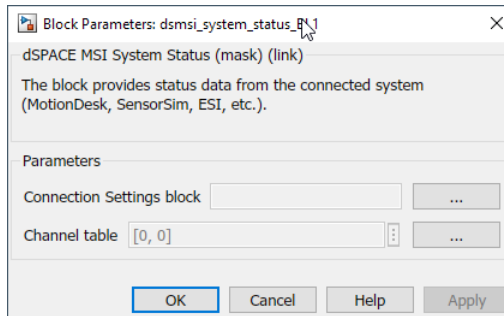
Purpose

To define the parameters of the System Status block for each connected system or application.

Dialog settings

Block parameters dialog To load the Block Parameters dialog, double-click the System Status block in Simulink.

The Block Parameters: dsmsi_system_status dialog is displayed.



Application parameters The table details the application parameters for the System Status block.

Parameter	Description
Connection Settings block	<p>Select the Connection Settings block to which the System Status block connects.</p> <p>You can use the browse function to select the Connection Settings block from your Simulink model.</p> <p>If there is no Connection Settings block in the model, a warning is displayed.</p>
Channel table	<p>Specify the mapping configurations of the MotionDesk, SensorSim application and the Environment Sensor Interface Unit on the System Status block.</p> <p>You must define a mapping table to map each input and output channel on the connected system application to a single element of the outputs of the System Status block, for example, Frame Counter, Frame Frequency, and Latency.</p> <p>You can use the browse function to load the Channel Table Parameter dialog. Each Channel number and the Direction can be set in this dialog.</p> <p>The channel numbers are defined by the connected system. These numbers must be used in the channel table.</p>

Related topics

References

Block Description (dsmsi_system_status)..... 116

Sensor Failure Block

Purpose To enable or disable sensor failures on the connected sensor device, for example, pixel errors, and specify the arguments for the failures on specific sensors connected via the Environment Sensor Interface Unit.

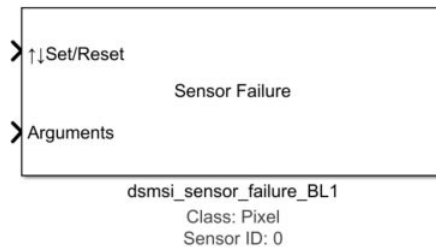
Where to go from here

Information in this section

Block Description (dsmsi_sensor_failure).....	120
To understand the inports and outports of the Sensor Failure block.	
Block Parameters Page (dsmsi_sensor_failure).....	122
To define the parameters of the Sensor Failure block for each connected sensor.	

Block Description (dsmsi_sensor_failure)

Illustration



Purpose To understand the inports and outports of the Sensor Failure block.

Description

The Sensor Failure block is used to enable and disable sensor failures and specify the arguments for the failures on specific connected sensors.

The block can be connected only to the Connection Settings block that is connected to the Environment Sensor Interface Unit.

The block supports static and dynamic failure arguments that change during runtime. The sensor failure arguments, which can change dynamically during the simulation can be provided through the Arguments inport.

I/O characteristics

The following table shows the block inports:

Simulink Input	Simulink Data Type	Description
Set/Reset	Boolean double single Int8 UInt8 Int16 UInt16 Int32 UInt32	Sensor failures for the connected sensor are true or false. <ul style="list-style-type: none"> ▪ 0: False ▪ 1: True The arrows in the block mask indicate that the inport is edge triggered. It reacts to a change of state and not the state itself. After detecting the trigger, the sensor failure is enabled or disabled for one simulation time step.
Arguments	Boolean double single Int8 UInt8 Int16 UInt16 Int32 UInt32	<p>The inport accepts and provides the sensor failure arguments that a connected system expects from the blockset. The argument values can change dynamically during the simulation. You cannot change type and dimension of the arguments during the simulation.</p> <p>If static arguments are specified manually in the block parameters dialog, the Arguments inport is disabled.</p> <p>For example, a pixel sensor failure might be specified as [42, 24, 255]. The arguments indicate the x-, y-coordinates and the RGB pixel color. The data type and its length are automatically determined.</p>

Note

When starting the simulation or building the model, the blockset validates the inport signal data type and signal dimensions. If a data type or dimension is not allowed, a warning is displayed. For example, Int64 and UInt64 data types are not supported.

Dialog pages

The dialog settings can be specified on the following dialog pages:

- [Block Parameters Page \(dsmsi_sensor_failure\)](#) on page 122

Related topics

HowTos

How to Configure Sensor Failures for a Connected Sensor.....	44
--	----

References

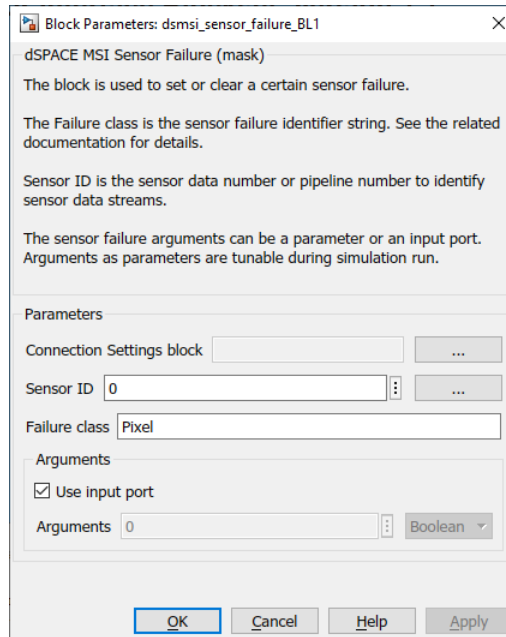
Connection Settings Block.....	93
Sensor Feedback Block.....	124
Simulation Data Objects Block.....	100
System Status Block.....	115

Block Parameters Page (dsmsi_sensor_failure)

Purpose	To define the parameters of the Sensor Failure block for each connected sensor.
---------	---

Dialog settings	Block parameters dialog To load the Block Parameters dialog, double-click the Sensor Failure block in Simulink.
-----------------	--

The Block Parameters: dsmsi_sensor_failure dialog is displayed.



Application parameters The table details the application parameters for the Sensor Failure block.

Parameter	Description
Connection Settings block	Select the Connection Settings block that the Sensor Failure block connects to. The sensor failure communicates via the Connection Settings block to the destination sensor device. You can use the browse function to select the Connection Settings block from your Simulink project. If there is no Connection Settings block in the model, a warning is displayed.
Sensor ID	Specify the Sensor ID. The number range starts at 0. You can type the Sensor ID or use the browse function to select the Sensor ID from the <code>scene.xml</code> file. The Sensor ID identifies the sensor data stream for the connected sensor to send a failure to.
Failure class	Specify the failure class, for example, <code>Pixel</code> for a camera pixel failure. The failure class identifies the type of failure and must be supported by the Environment Sensor Interface Unit and the connected sensor device, for example, a camera sensor. <div style="background-color: #f0f0f0; padding: 10px; margin-top: 10px;"> Note The failure class parameter is not supported by MotionDesk or the SensorSim application. </div>
Arguments - Use input port	Select for the failure arguments, which can change dynamically during the simulation to be taken from the Arguments inport.

Parameter	Description
Arguments	<p>Clear Use input port to enable the Arguments property to specify the failure arguments manually.</p> <p>If Use input port is cleared, you can specify the argument values for the sensor failure class.</p> <p>The arguments that the connected system expects from the blockset must be defined in the connected system. They are not defined in the blockset.</p>
Argument data type	<p>Select the data type from the list. The data types for manually added arguments are as follows:</p> <ul style="list-style-type: none"> ▪ Double ▪ Float ▪ UInt32 ▪ Int32 ▪ Boolean

Related topics

References

[Block Description \(dsmsi_sensor_failure\)..... 120](#)

Sensor Feedback Block

Purpose

To configure the Sensor Feedback block that provides feedback data for the simulated sensor from the Environment Sensor Interface Unit.

Where to go from here

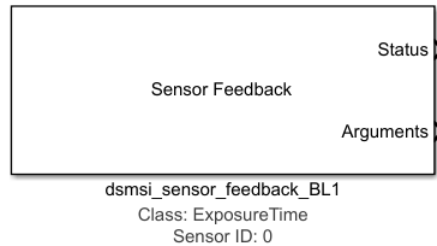
Information in this section

[Block Description \(dsmsi_sensor_feedback\)..... 125](#)
 To understand the inports and outports of the Sensor Feedback block.

[Block Parameters Page \(dsmsi_sensor_feedback\)..... 126](#)
 To define the parameters of the Sensor Feedback block for each connected sensor.

Block Description (dsmsi_sensor_feedback)

Illustration



Purpose

To understand the inports and outports of the Sensor Feedback block.

Description

Sensor feedback data is sent from the connected sensors through the Environment Sensor Interface Unit. The **Sensor Feedback** block receives the feedback data, filters it for a specific sensor and forwards it to the model for further processing.

The **Sensor Feedback** has outports for the sensor feedback status and for the sensor feedback arguments for the connected sensor feedback data. Sensor feedback is defined by the system that collects the feedback data. The system defines the feedback parameter class and the format of the arguments vector.

I/O characteristics

The following table shows the block outputs:

Simulink Input	Simulink Data Type	Description
Status	double	Common blockset status and error codes <ul style="list-style-type: none"> 0: Success 1: New data 2: Error - Out of memory 3: Error - Data output too short 4: Error - Not all data was transmitted to destination Other: Unknown error
Arguments	double	Specify the sensor feedback arguments. You can set the length of the vector in the dialog. Outputs the sensor feedback data vector of the length specified by Length parameter in the block dialog.

Dialog pages

The dialog settings can be specified on the following dialog pages:

- [Block Parameters Page \(dsmsi_sensor_feedback\)](#) on page 126

Related topics

HowTos

How to Configure Sensor Feedback from the Environment Sensor Interface Unit.....	46
--	----

References

Connection Settings Block.....	93
Sensor Failure Block.....	120
Simulation Data Objects Block.....	100
System Status Block.....	115

Block Parameters Page (dsmsi_sensor_feedback)

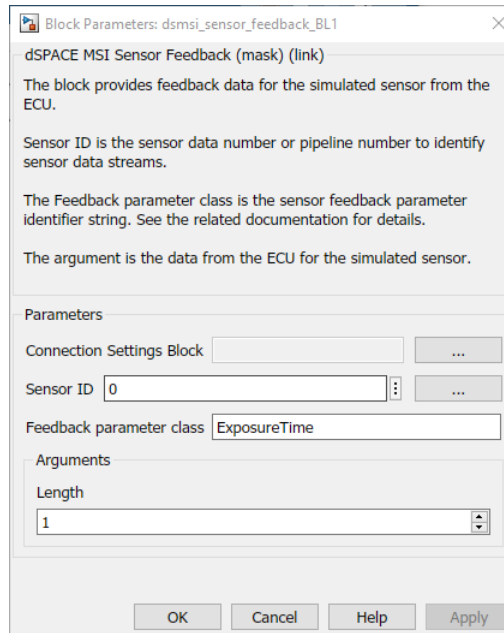
Purpose

To define the parameters of the Sensor Feedback block for each connected sensor.

Dialog settings

Block parameters dialog To load the Block Parameters dialog, double-click the Sensor Feedback block in Simulink.

The Block Parameters: dsmsi_sensor_feedback dialog is displayed.



Application parameters The table details the application parameters for the Sensor Feedback block.

Parameter	Description
Connection Settings block	<p>Select the Connection Settings block that the Sensor Feedback block connects to. The sensor feedback communicates via the Connection Settings block to the destination sensor device.</p> <p>You can use the browse function to select the Connection Settings block from your Simulink project.</p> <p>If there is no Connection Settings block in the model, a warning is displayed.</p>
Sensor ID	<p>Specify the Sensor ID. The number range starts at 0. You can type the Sensor ID or use the browse function to select the Sensor ID from the <code>scene.xml</code> file.</p> <p>The Sensor ID identifies the sensor data stream for the connected sensor for feedback.</p>
Feedback parameter class	<p>Specify the feedback parameter class. This identifies the feedback parameters between multiple sensor feedback parameters sent to the ECU.</p> <p>The value entered in the block must be supported by the Environment Sensor Interface Unit. For the SensorSim application, only the <code>ExposureTime</code> parameter class for a camera sensor is supported.</p>

Related topics**References**

[Block Description \(dsmsi_sensor_failure\).....](#) 120

Model and Sensor Interface Blockset Demo Library

Purpose

A library with demo models is provided.

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

[Demo Library Overview \(dsmsi_demolib\).....](#) 128

The Model and Sensor Interface Blockset provides two demo blockset configurations in the demo library.

[Coordinate System Demo for SCALEXIO.....](#) 129

This demo model shows how the transformation order influences the position of the corresponding 3-D movable objects in the Model and Sensor Interface Blockset.

[Automotive Demo for SCALEXIO.....](#) 132

This demo model demonstrates the connections of a simple ASM model used for a vehicle dynamics simulation to blocks of the Model and Sensor Interface Blockset.

Demo Library Overview (dsmsi_demolib)

Purpose

The Model and Sensor Interface Blockset provides two demo blockset configurations in the demo library.

Access

To access the demo library, double-click the Demos button in the Model and Sensor Interface Blockset Library or type the `dsmsi_demolib` command in MATLAB Command Window.

Overview

The Model and Sensor Interface Blockset includes the following demos in the demo library

Demos

- **Coordinate System Demo for SCALEXIO:** Demonstrates how the transformation order influences the position of the 3-D movable objects.
- **Automotive Demo for SCALEXIO:** Demonstrates how a simple ASM model used for a vehicle dynamics simulation is connected to blocks of the Model and Sensor Interface Blockset. For example, the vehicle transformation outputs from the model can be connected to a Simulation Data Objects block.

Related topics

References

[Model and Sensor Interface Blockset Library Overview.....](#) 91

Coordinate System Demo for SCALEXIO

Purpose

This demo model shows the basic usage of a transformation object. It shows how the transformation order influences the position of the corresponding 3-D movable objects in the Model and Sensor Interface Blockset. You can connect inputs and change parameters to observe the outputs.

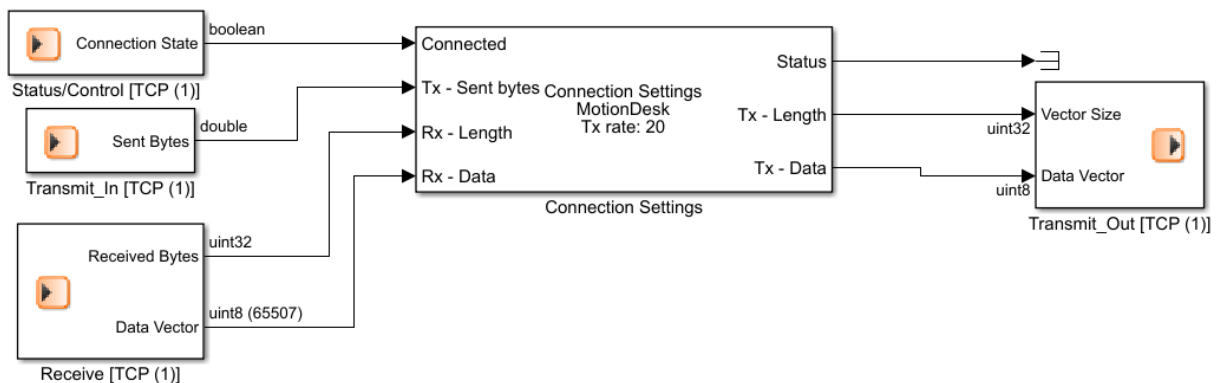
It also demonstrates the use of multiple objects in a single transformation object.

Connection Settings

The demo includes a pre-configured Connection Settings block for a MotionDesk observer (client). The block is connected to model port blocks for the SCALEXIO platform. This can be used to simulate with sensors in a hardware-in-the-loop architecture.

To run the simulation on SCALEXIO, you must configure the Ethernet communication using the Ethernet Setup and TCP function blocks in ConfigurationDesk. For more information, refer to [How to Connect a SCALEXIO Platform to a Single MotionDesk Observer](#) on page 69.

The Connection Settings configuration is shown as follows.

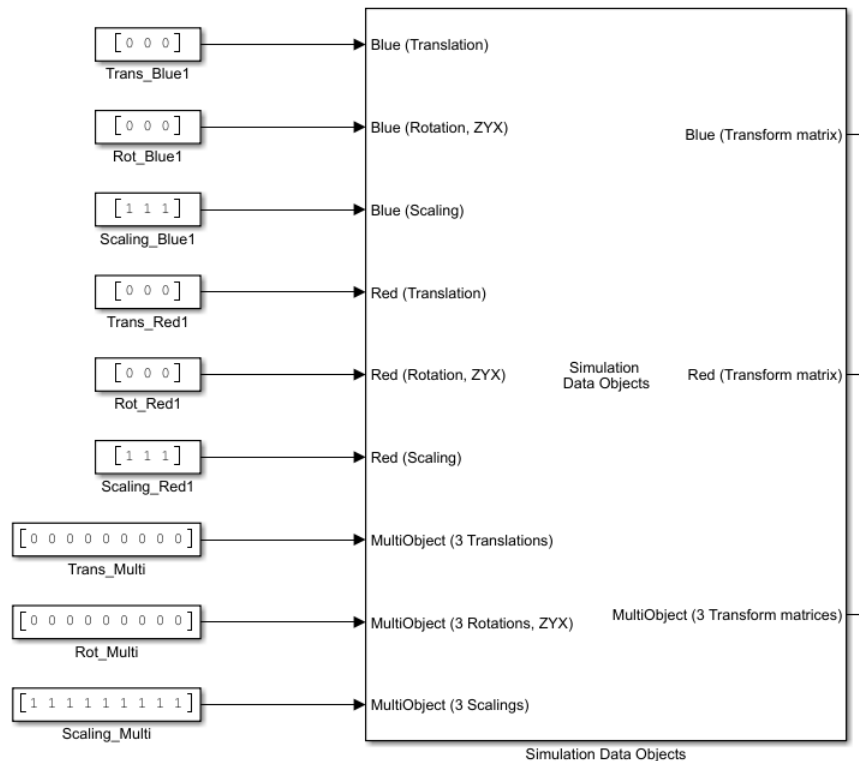


Simulation Data Objects block

The Simulation Data Objects block contains three objects: Blue, Red, and MultiObject.

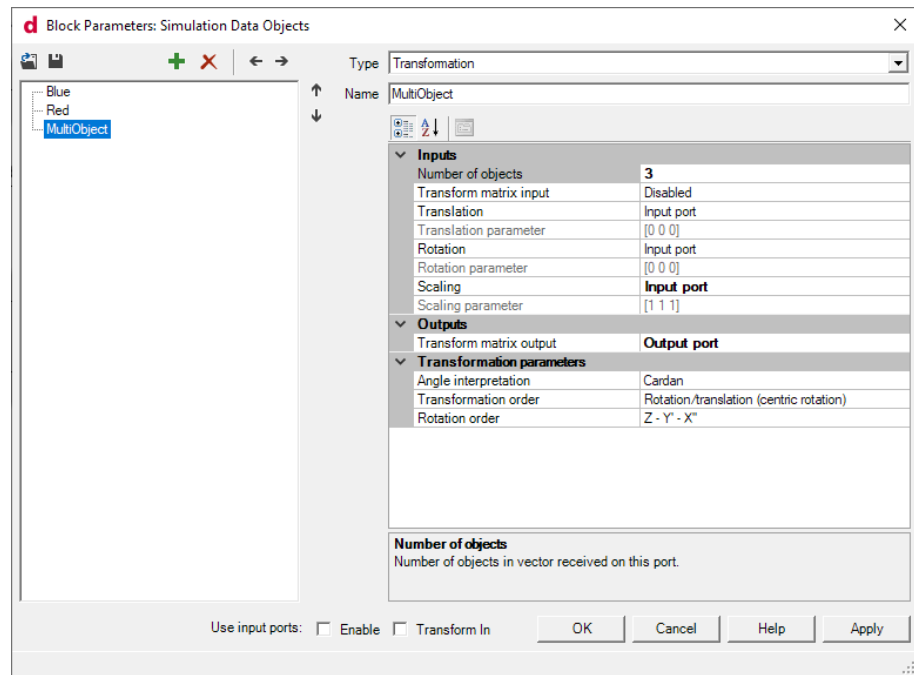
For both objects, the transformation, rotation, and scaling input ports and the output ports are enabled. The transformation angle interpretation uses the Cardan format. To view the behavior of the Euler angle interpretation, change the **Angle Interpretation** parameter in the block parameters dialog. The MultiObject object is a transformation object that contains three inlined objects.

The illustration shows the Simulation Data Objects for the Coordinate System Demo.

**Simulation Data Objects dialog**

You can double-click the Simulation Data Objects block to load the block parameters dialog.

The Simulation Data Objects block dialog is displayed with the MultiObject object in focus. Three objects are specified for the transformation object as follows:



The following table describes the objects in the Simulation Data Objects block that are configured in the dialog.

Name	Type	Description
Blue	Transformation	The blue coordinate system rotates around its local coordinates and moves to the position specified by the translation vector.
Red	Transformation	The red coordinate system rotates off-center in a radius defined by the translation vector
MultiObjects	Transformation	The transformation object has multiple objects, for example, for three fellows in a simulation.

You can change the Number of objects and observe the size and content of the input and output signals (vectors and matrices). For example:

- If the Number of objects is 1, a single object is connected to the ports. The input vectors are [1x3] and the output matrix is [4x4].
- If the Number of objects is 3, multiple objects are connected to ports. The input vectors are [1x9] and the output matrix is [4x12].

For more information on the Simulation Data Objects block and the ports, refer to [Block Description \(dsmsi_simulation_data_objects\)](#) on page 100 and to [How to Configure the Simulation Data Objects in a Simulation Model](#) on page 51.

Related topics

HowTos

[How to Configure the Simulation Data Objects in a Simulation Model](#)..... 51

How to Connect a SCALEXIO Platform to a Single MotionDesk Observer.....	69
---	----

References

Connection Settings Block.....	93
Sensor Failure Block.....	120
Sensor Feedback Block.....	124
Simulation Data Objects Block.....	100
System Status Block.....	115

Automotive Demo for SCALEXIO

Purpose

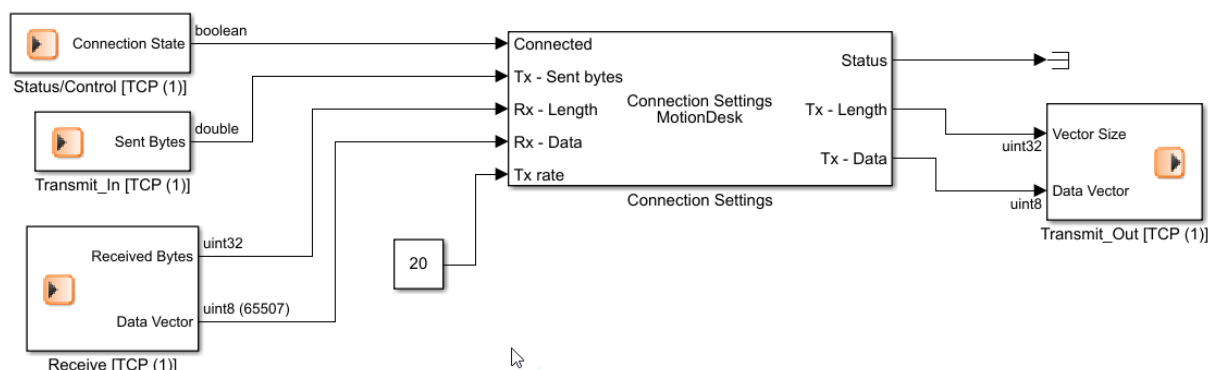
The automotive demo simulates simple car dynamics. The demo model demonstrates the connections of a simple ASM model used for a vehicle dynamics simulation to blocks of the Model and Sensor Interface Blockset.

Connection Settings

The demo includes a pre-configured Connection Settings block for a MotionDesk observer (client). The block is connected to model port blocks for the SCALEXIO platform. This can be used to simulate with sensors in a hardware-in-the-loop architecture.

To run the simulation on SCALEXIO, you must configure the Ethernet communication using the Ethernet Setup and TCP function blocks in ConfigurationDesk. For more information, refer to [How to Connect a SCALEXIO Platform to a Single MotionDesk Observer](#) on page 69.

The Connection Settings configuration is shown as follows.



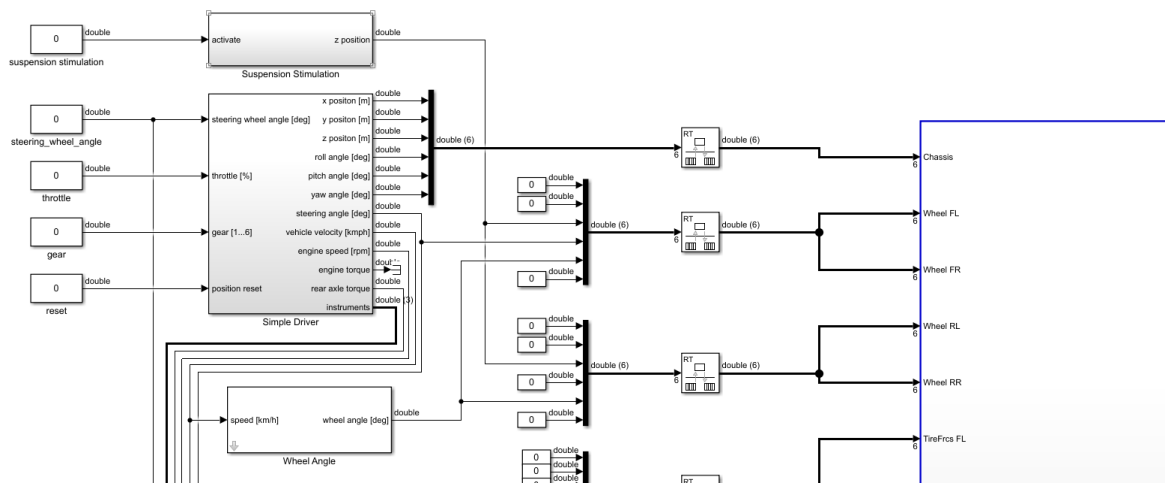
ASM model

The ASM model includes blocks for the vehicle control: for example, actuator inputs and the output ports for the position, rotation, and steering angles of the vehicle. Outputs for the signal instrument data and additional blocks for the

position and orientation of the wheels, tire forces, and suspension position are included.

The outputs of the ASM model blocks are connected by rate transition functions to the vehicle subsystem that includes the blocks from the Model and Sensor Interface Blockset.

A section of the ASM model for the vehicle connected to the subsystem that contains the connections to blocks of the Model and Sensor Interface Blockset is shown as follows:

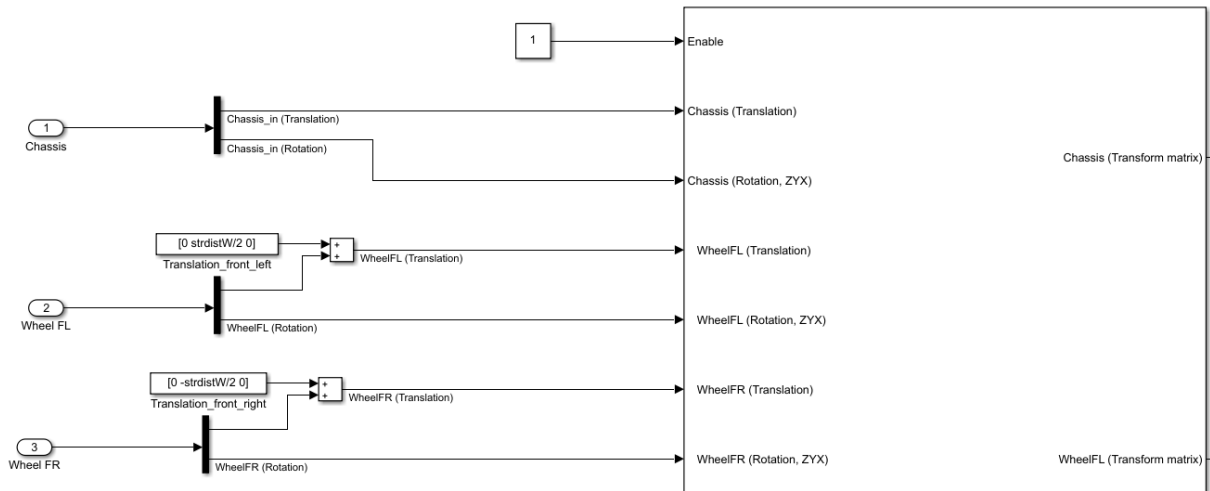


Simulation Data Objects block

Each of the vehicle outputs from the model are connected to the inports of a Simulation Data Objects block to calculate the data to be transmitted to the connected system. It receives and transforms the signals and sends them to the Connection Settings block.

The Simulation Data Objects block contains inports for the chassis transformation object and children objects in a kinematic chain for the wheels, tire forces, and steering wheel. For the chassis and wheel objects, the inports are enabled for the position and rotation data. The transformation angle interpretation uses the Cardan format. To view the behavior of the Euler angle interpretation, change the **Angle Interpretation** parameter in the block parameters dialog. Additional objects inports are available for signal instrument data.

The illustration shows part of the Simulation Data Objects block in the model subsystem for the Automotive Demo. The inports for the chassis transformation object and two of the wheels in the kinematic chain are as follows:



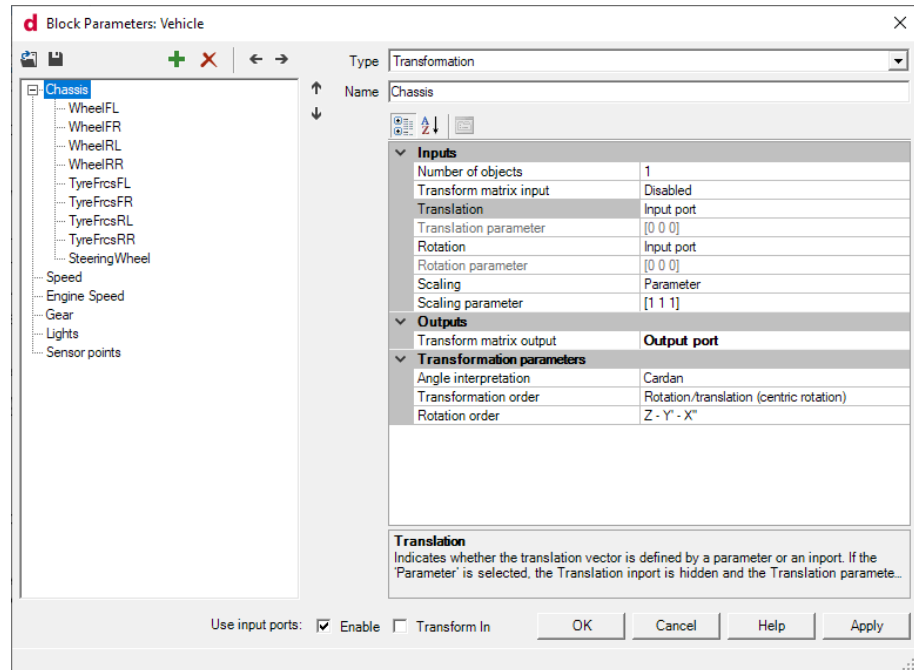
Simulation Data Objects dialog

You can double-click the Simulation Data Objects block to load the block parameters dialog.

The Simulation Data Objects block dialog is displayed with the chassis transformation object for the ego-vehicle and its wheels, tire forces, and steering wheel as child elements in a kinematic chain. The signal instrument objects for the vehicle are also shown in the dialog.

You can edit the input parameters, for example, the steering wheel angle, throttle position, and gear to drive a vehicle model. The output for the vehicle is the motion of a vehicle and wheels, in addition to some dynamic parameters, for example the transmission torques and the forces on the wheels.

The chassis transformation object is in focus as follows:



The following table describes the objects and the types in the Simulation Data Objects block that are configured in the dialog. Each of the four object types are used.

Name	Type	Description
Chassis	Transformation	The position and orientation of the vehicle are provided by the inport. The Cardan angle interpretation is selected and a parameter is specified for the scaling of the vehicle.
WheelIFL,FR,RL,RR	Transformation	A child object of the chassis in a kinematic chain for the position, orientation, scaling, and angle interpretation of each of the four wheels.
TyreFrcsFL,FR,RL,RR	Position and Scale	A child object of the chassis in a kinematic chain for the position and value of the forces exerted on each of the tires of the vehicle during the simulation, for example, on acceleration and cornering.
SteeringWheel	Transformation	A child object of the chassis in a kinematic chain for the position, orientation, scaling, and angle interpretation of the steering wheel.

Name	Type	Description
Speed	Signal	The speed of the vehicle in km/h is provided by the inport.
EngineSpeed	Signal	The engine speed or revolutions of the vehicle in rpm is provided by the inport.
Gear	Signal	The engaged gear number of the vehicle as a number is provided by the inport.
Lights	Signal	The on/off status of the vehicle lights is provided by the inport.
Sensor points	Position	The sensor points for a lidar sensor used to build a 3-D point cloud of the vehicle environment within the field of view of the sensor.

For more information on the **Simulation Data Objects** block and the ports, refer to [Block Description \(dsmsi_simulation_data_objects\)](#) on page 100 and to [How to Configure the Simulation Data Objects in a Simulation Model](#) on page 51.

Related topics

HowTos

How to Configure the Simulation Data Objects in a Simulation Model.....	51
How to Connect a SCALEXIO Platform to a Single MotionDesk Observer.....	69

References

Connection Settings Block.....	93
Sensor Failure Block.....	120
Sensor Feedback Block.....	124
Simulation Data Objects Block.....	100
System Status Block.....	115

Limitations

Limitations When Using the Model and Sensor Interface Blockset

Introduction	There are a number of limitations for implementing models with the Model and Sensor Interface Blockset.
Not supported on RTI platforms	<p>The Model and Sensor Interface Blockset is not supported on RTI platforms, for example, DS1006, DS1007, MicroLabBox, and MicroAutoBox II.</p> <p>An error message is displayed if you build a Simulink model with blocks from Model and Sensor Interface Blockset on an RTI target, for example, rti1006, rti1007, rti1401, rti1104, rti1202.</p>
dSPACE Host Interface communication	<p>Using the Model and Sensor Interface Blockset communication via the dSPACE Host Interface is not possible.</p> <p>You must use the relevant Ethernet functions for the supported VEOS and SCALEXIO simulation platforms. Refer to Basics of the Blockset and Connected Systems on page 12.</p>
Block names	<p>The following characters are not supported in the Model and Sensor Interface Blockset Simulink block names.</p> <p>/, \, :, *, ?, , and %.</p> <p>The Simulation Data Objects block name supports all printable ASCII characters except the following:</p> <p>/, \, :, <, >, ", and .</p>

Spaces are permitted between words and as the first character. However a space cannot be used as the only character in the name.

Note

The parameters dialog cannot be opened if the block contains invalid characters in its name.

Simulation Data Objects - Object name

In the Simulation Data Object Name field, you can insert all printable ASCII characters except the following:

/, \, #, ", and |.

Spaces are permitted between words but not at the beginning or end of the object name. Line breaks are not supported.

When importing simulation data objects or changing the number of objects for a transformation object using the automation API, object names with non-ASCII characters are also not supported.

Simulation Data Objects - Multiple transformation objects

If the number of transformation objects [n] is more than 1 for a transformation object in the Simulation Data Object, only objects of the Signal type can be added as children to the transformation object.

The signal child object transmits the same number of objects that is specified for the parent transformation object.

Automation API

The automation API does not support the importing of the data objects to add, delete, and rearrange the objects or to change the input and output properties of an existing Simulation Data Objects block that is already configured with objects.

Using blocks in Simulink subsystems

Limitations using blocks of the Model and Sensor Interface Blockset in the Simulink subsystems:

- Simulink Function subsystem
- Simulink subsystems with read/write permissions set to **NoReadOrWrite**

Using blocks in Simulink subsystems with asynchronous triggers

Limitations using blocks of the Model and Sensor Interface Blockset blocks in the Simulink subsystems with asynchronous triggers:

- If blocks are used within subsystems with asynchronous triggers, all Model and Sensor Interface Blockset blocks must be within the subsystems and all of the subsystems must be triggered with the same asynchronous trigger.
- If each subsystem has its own asynchronous trigger, it is not possible to use the blocks within subsystems.

- It is not possible in the same model to use stand-alone blocks outside a subsystem together with blocks within subsystems with asynchronous triggers.

Excluding blocks from simulation	Commenting out or through blocks in Simulink model to exclude them from the simulation is not supported.
Referenced models	Model and Sensor Interface Blockset blocks are not supported in referenced models.
Row-major code generation	The blockset does not support the row-major code generation feature introduced in MATLAB 2018b.
Restricted support of variant subsystem block	There must be no Model and Sensor Interface Blockset blocks in a variant subsystem or any of its subsystems, whether activated or deactivated.
Task configuration in ConfigurationDesk for SCALEXIO systems	<p>The task configuration in ConfigurationDesk is limited when the behavior model contains Model and Sensor Interface Blockset blocks.</p> <p>In the model, the following options for the jitter and latency optimization for a task within the same application process are not supported:</p> <ul style="list-style-type: none"> ▪ No jitter, low latency ▪ Low jitter, low latency
Parallel simulation	The Model and Sensor Interface Blockset does not support the parallel simulation of a model because MotionDesk requires a continuous data stream of simulation data.
Simulink accelerator mode	<p>To build MEX files in Simulink accelerator mode, a C/C++ compiler is required.</p> <p>When using the blockset with the blocks of the VEOS Ethernet Blockset Solution, you must use the Visual Studio Compiler when using Simulink accelerated mode. Other compilers are not supported.</p>
Block sample time	All Model and Sensor Interface Blockset blocks used in the model must run at same sampling rate.
Using blocks in a multiprocessor system	In each model, the blocks of the Model and Sensor Interface Blockset must run in the same process. The Connection Settings block creates a shared memory area that the connected blocks can use. This is available only in the same process.

The blockset can run in parallel in different models and processes at the same time within the framework of multiple instantiation.

Data assignment

The assignment of objects to simulation data is based on an index. If you record MDF files and change the order of IDs in your simulation model, the simulation data must be reassigned to the movable objects.

Related topics

Basics

[Limitations \(RTI and RTI-MP Implementation Guide !\[\]\(c694a3ff3b077d76910920a6a1593ab4_img.jpg\)](#))

Numerics

3-D objects 27

A

ASM model
 add the animation interface 36
 automation API 59
 import and export of simulation data objects 59
 set the number of objects 62
 user files 63
 automotive demo
 ASM model 132
 Connection Settings 132
 Simulation Data Objects block 133
 Simulation Data Objects dialog 134

B

block description
 dsmsi_connection_settings 93
 dsmsi_sensor_failure 120
 dsmsi_sensor_feedback 125
 dsmsi_simulation_data_objects 100
 dsmsi_system_status 116
 block parameters
 dsmsi_connection_settings 98
 dsmsi_sensor_failure 122
 dsmsi_sensor_feedback 126
 dsmsi_simulation_data_objects 104
 dsmsi_system_status 118

C

Common Program Data folder 8
 connecting simulation platforms
 SCALEXIO 69
 SCALEXIO to multiple observers 73
 SensorSim application 81
 VEOS 78
 Connection Settings block
 using Ethernet functions 42
 coordinate system 22
 coordinate system demo
 Connection Settings 129
 Simulation Data Objects block 130
 Simulation Data Objects dialog 130

D

demo library dsmsi_demolib
 automotive demo 132
 coordinate system demo 129
 overview 128
 demo library overview
 access 128
 Documents folder 8
 dsmsi_connection_settings block 93
 dsmsi_sensor_failure block 120
 dsmsi_sensor_feedback block 124

dsmsi_simulation_data_objects block 100
 dsmsi_system_status block 115

E

Euler angles 25

F

firewall settings 66

H

homogeneous transformation 23

I

IP address 65
 private address space 65

L

Local Program Data folder 8

M

Model and Sensor Interface blocks
 Connection Settings 93
 Sensor Failure 120
 Sensor Feedback 124
 Simulation Data Objects 100
 System Status 115
 Model and Sensor Interface blockset
 demo library 128
 Model and Sensor Interface Blockset
 accessing 32
 adapting simulation models workflow 29
 adapting Simulink models 28
 block overview 16
 block priority 91
 blockset features 14
 configuring simulation data objects 48
 connected systems basics 12
 connecting simulation platforms 68
 Ethernet functions 42
 library overview 91
 limitations 137
 migration from previous releases 19
 migration from the solution 19
 network basics 64
 network settings for windows 66
 network settings in MotionDesk 67
 supported platforms 16
 Model and Sensor Interface blockset demo
 library 128

P

pitch angle 25

R

roll angle 25

S

Sensor Failure block
 configure sensor failures 44
 Sensor Feedback block
 configure sensor feedback 46
 simulation data objects
 automation API 59
 high-precision feature 104
 simulation data objects block 49
 basics 49
 kinematic chains 50
 multiple transformation objects 50
 Simulation Data Objects block
 configure 51
 export 56
 import 56
 multiple transformation objects 53
 Simulation model
 workflow 29
 Simulink model
 connecting to a Sensor Simulation system or application 39
 supported platforms 16
 System Status block
 collect status information 43

T

TCP ports 65
 TCP/IP protocols 64
 translation vector 23

V

vertex in 3-D space 23

Y

yaw angle 25

