

AutomationDesk

Simulating Electrical Errors

For AutomationDesk 6.5

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.

© 2017 - 2021 by:
dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Germany

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

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

Contents

About This Document	5
New Features	9
New Features For Electrical Error Simulation.....	9
Basics and Instructions	11
Basics on Simulating Electrical Errors via the XIL API Convenience Library.....	11
How to Build a Basic Sequence for Simulating Electrical Errors.....	16
How to Simulate a Manually Triggered Error.....	17
How to Simulate a Software-Triggered Error.....	21
How to Simulate Errors from an Error Configuration File.....	25
Reference Information	29
Automation Blocks.....	30
XIL API Convenience (Electrical Error Simulation).....	30
Main Elements.....	30
DeactivateError.....	31
InitEESPort.....	32
ReleaseEESPort.....	34
Manually Triggered Errors.....	35
PrepareMultiPinError.....	35
PrepareSinglePinError.....	37
Trigger.....	39
Software Triggered Errors.....	40
PrepareMultiPinErrorWithCondition.....	41
PrepareMultiPinErrorWithDuration.....	44
PrepareSinglePinErrorWithCondition.....	46
PrepareSinglePinErrorWithDuration.....	49
WaitForTrigger.....	51
XIL API (Electrical Error Simulation).....	53
EESPort (Data Object).....	54
EESPortFactory (Data Object).....	55

ErrorConfiguration (Data Object).....	56
EESConfigurationReader (Data Object).....	56
EESConfigurationWriter (Data Object).....	57
ErrorSet (Data Object).....	58
BaseErrorBuilder (Data Object).....	59
ErrorFactory (Data Object).....	60
BaseError (Data Object).....	60
SpecificErrorFactory (Data Object).....	62
SpecificErrorFactory2 (Data Object).....	63
Commands And Dialogs.....	64
XIL API.....	64
Edit (SinglePinError).....	64
Edit (MultiPinError).....	66
 Automation.....	 69
Basics on Automating Electrical Error Simulation.....	69
 Limitations.....	 71
Limitations When Using the XIL API Convenience Library.....	71
 Glossary.....	 73
 Index.....	 89

About This Document

Content This document introduces you to AutomationDesk's basic practices.

Required knowledge Working with AutomationDesk requires:

- Basic knowledge in handling the PC and the Microsoft Windows operating system.
- Basic knowledge in developing applications or tests.
- Basic knowledge in handling the external device, which you control remotely via AutomationDesk.

dSPACE provides trainings for AutomationDesk. For more information, refer to <https://www.dspace.com/go/trainings>.

Legal information

Note


Legal Information on ASAM binaries and ASAM documentation
dSPACE software also installs components that are licensed and released by ASAM e.V. (Association for Standardisation of Automation and Measuring Systems).





dSPACE hereby confirms that dSPACE is a member of ASAM and as such entitled to use these licenses and to install the ASAM binaries and the ASAM documentation together with the dSPACE software.

You are not authorized to pass the ASAM binaries and the ASAM documentation to third parties without permission. For more information, visit <http://www.asam.net/license.html>.

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.

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

Naming conventions

dSPACE user documentation uses the following naming conventions:

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

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

Special folders

Some software products use the following special folders:

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

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

or

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

Documents folder A standard folder for user-specific documents.

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

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

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

Accessing dSPACE Help and PDF Files


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

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

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

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

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

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

New Features

Introduction

Information on enhancements and new features of AutomationDesk.

New Features For Electrical Error Simulation

New features and migration

For information on new features of the current version of AutomationDesk, refer to [New Features of AutomationDesk 6.5 \(New Features and Migration !\[\]\(e3f8612927870f2e0f9f5989e6dd3064_img.jpg\)](#)).

Basics and Instructions

Where to go from here	Information in this section
	Basics on Simulating Electrical Errors via the XIL API Convenience Library..... 11 General information on simulating electrical errors via the XIL API Convenience library.
	How to Build a Basic Sequence for Simulating Electrical Errors..... 16 Instructions on building a sequence that performs the generic actions required for simulating electrical errors.
	How to Simulate a Manually Triggered Error..... 17 Instructions on simulating an electrical error that is triggered from your sequence.
	How to Simulate a Software-Triggered Error..... 21 Instructions on simulating an electrical error that is triggered by a condition watcher.
	How to Simulate Errors from an Error Configuration File..... 25 Instructions on simulating errors from an error configuration file.

Basics on Simulating Electrical Errors via the XIL API Convenience Library

Introduction	General information on simulating electrical errors via the XIL API Convenience library.
---------------------	--

Warning

Performing electrical error simulation might be dangerous. You have to note the following safety precautions.

WARNING

Risk of unexpected high currents and voltages due to electrical error simulation

During electrical error simulation, high currents and voltages might be present on board channels and/or connector pins, which are not expected. This can result in death, personal injury, fire, and/or damage to the simulator and connected external devices.

- Set up a test zone to avoid contact to electrical components of the simulator and to moveable physical components controlled by the simulator.


Note

Especially signal measurement channels (such as channels connected in parallel or interconnected reference lines) can carry high currents.

Basic concept of the XIL API Convenience library

The XIL API Convenience library is a [custom library](#). It is write-protected to prevent modifications to its [blocks](#).

The library provides the following [folders](#) at its top level:

- The Model Access Port folder contains automation blocks to access a registered simulation platform. For more information on these blocks, refer to [Accessing Simulation Platforms via the XIL API Convenience Library](#) (AutomationDesk [Accessing Simulation Platforms](#) )
- The Electrical Error Simulation Port folder contains automation blocks for accessing connected electrical error simulation hardware. The usage of these blocks is described in the following.

The XIL API Convenience library is based on the ASAM XIL API standard.

For more information on the ASAM XIL API standard, refer to:

- [ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf](#)
- [ASAM_AE_XIL_Generic-Simulator-Interface_BS-2-4_CSharp-API-Technology-Reference-Mapping-Rules_V2-1-0.pdf](#)

Note

Legal Information on ASAM binaries and ASAM documentation
dSPACE software also installs components that are licensed and released by ASAM e.V. (Association for Standardisation of Automation and Measuring Systems).

dSPACE hereby confirms that dSPACE is a member of ASAM and as such entitled to use these licenses and to install the ASAM binaries and the ASAM documentation together with the dSPACE software.

You are not authorized to pass the ASAM binaries and the ASAM documentation to third parties without permission. For more information, visit <http://www.asam.net/license.html>.

All blocks of the XIL API Convenience library are implemented as [templates](#) of Exec blocks that use data object types and methods of the XIL API library.

The provided blocks facilitate the migration from dSPACE-specific failure simulation via the ControlDesk Access library to an ASAM-compliant simulation of electrical errors.

Basics on electrical error simulation

Electrical error simulation (EES) deals with typical wiring errors like loose contacts, broken cables and short circuits to neighboring pins, ground (chassis) or battery voltage. Supported by software, electrical error simulation is performed by the EES hardware of an HIL simulator. For information on which EES hardware is available for a type of simulator, refer to [Basics on Failure Simulation \(dSPACE XIL API Implementation Guide\)](#) and [Hardware for Failure Simulation \(dSPACE XIL API Implementation Guide\)](#).

Signal file Which errors can be simulated depends on the EES hardware that is used and is specified via failure classes in signal files. Prototypes of signal files are provided with the dSPACE XIL API installation. For more information, refer to [Defining Failure Classes with Signal Files \(dSPACE XIL API Implementation Guide\)](#) and [Failure Classes \(dSPACE XIL API Reference\)](#).

Basic concept of electrical error simulation via the ASAM XIL API

An [ASAM AE XIL API](#) EESPort provides access to the EES hardware and lets you simulate the electrical errors. It can be configured by the same files that you use when you access the XIL API via a script or graphically via [ControlDesk](#):

Port configuration file A port configuration file (PORTCONFIG) provides the hardware-dependent information for electrical error simulation in XML format. It contains:

- The configuration of the driver to access the EES hardware
- A potential map that relates the potentials used in short-circuit errors to an unique identifier
- Optionally, a signal mapping that relates the ECU pins that you are using in your HIL system to abstract names

For information on creating a port configuration file, refer to [Creating dSPACE EESPort Configuration Files \(dSPACE XIL API Implementation Guide\)](#).

When working with the XIL API Convenience library, an EESPort [data object](#) is initialized with the contents of a port configuration file. The EESPort data object is provided by the XIL API [library](#).

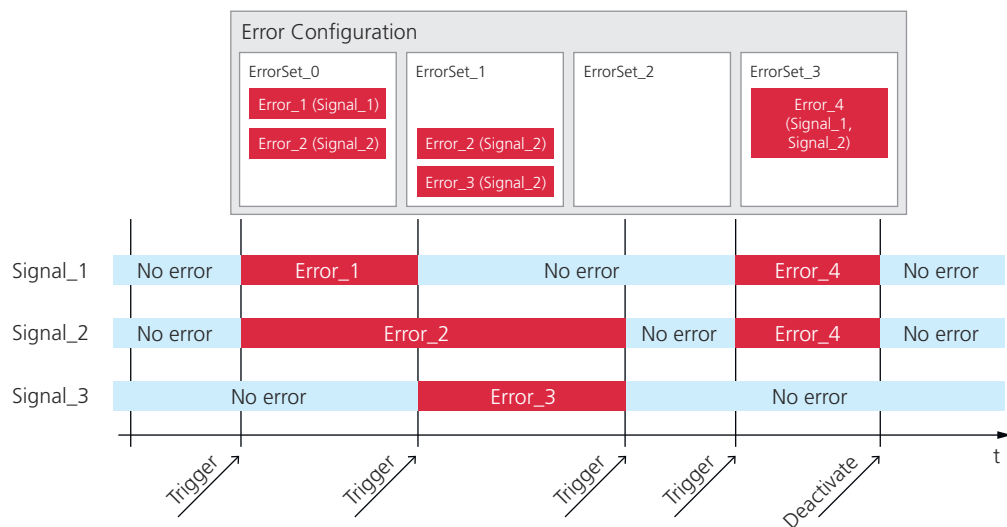
Error configuration file An error configuration file contains the specification of the simulated electrical errors in XML format.

An error configuration can consist of one or more error sets.

An error set specifies the simulated state of the wiring as a list of electrical errors that occur to the signals at the same time. An empty error set specifies a state with no errors. You can trigger an EESPort to change to the state that is specified in the next error set in the error configuration.

At the end of the electrical error simulation, the EES hardware must be set to the state with no errors by deactivating the EES port.

The following illustration shows how an example error configuration that contains four error sets is performed by the EES hardware.



If errors of the same kind are specified for the same signal in two consecutive error sets, like in *Error2* and *Error3* of the example, the simulated error is continued without any restart effects.

You can specify an error by its error category, its error type and the affected signals:

- **Category**
The error category specifies the kind of disturbance, such as short circuit to ground.
- **Error type**
The error type specifies whether the behavior of the error changes, such as a loose contact that is repeated for a specified duration, at a specified frequency and duty cycle.
- **Affected signals**
For each affected signal you can specify that a load rejection is performed when the error occurred.

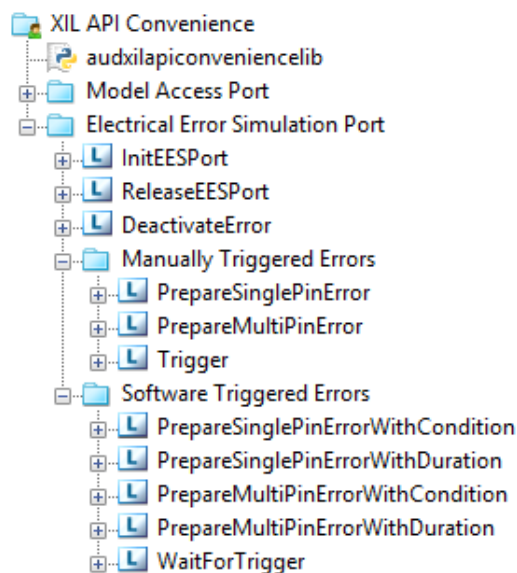
For information, refer to [Creating Error Configurations \(dSPACE XIL API Implementation Guide\)](#).

When working with the XIL API Convenience library, an `ErrorConfiguration` data object can be initialized with the contents of an error configuration file. The `ErrorConfiguration` data object is provided by the XIL API library.

Tip

In ControlDesk you can create an XIL API EESPort graphically, including its port configuration and related error configurations. For more information, refer to [Configuring Electrical Error Simulation \(ControlDesk Electrical Error Simulation via XIL API EESPort\)](#).

Overview of the XIL API Convenience library for electrical error simulation



The blocks of the XIL API Convenience library facilitate the following tasks:

- Managing the [Electrical error simulation port \(EESPort\)](#) data objects to provide access to the EES hardware
- Preparing a single error set that is triggered manually, i.e., the EES hardware starts to simulate the specified electrical error on one or multiple pins when a triggering block is executed
- Activating and deactivating the simulation of an electrical error as configured in an error set
- Preparing single error sets that are software-triggered, i.e., the EES hardware starts to simulate the specified electrical error on one or multiple pins when simulator variables fulfill a specified condition or a specified amount of time has elapsed
- Waiting with the execution of subsequent blocks until a trigger occurs

Demo projects

For the `ElectricalErrorSimulationPort` demo project on simulating electrical errors via the XIL API Convenience library, refer to `<DocumentsFolder>/XIL API Convenience`.


Related topics

References


[XIL API Convenience \(Model Access\) \(AutomationDesk Accessing Simulation Platforms !\[\]\(96cc62f861fdd6e50510c0224a756dff_img.jpg\)](#))

How to Build a Basic Sequence for Simulating Electrical Errors

Objective

You can build a basic [sequence](#)  that performs the generic steps to simulate electrical errors via the connected electrical error simulation (EES) hardware.

Preconditions

The name and path of the hardware-dependent [port configuration file](#)  (PORTCONFIG) for the connected EES hardware is required as input data.

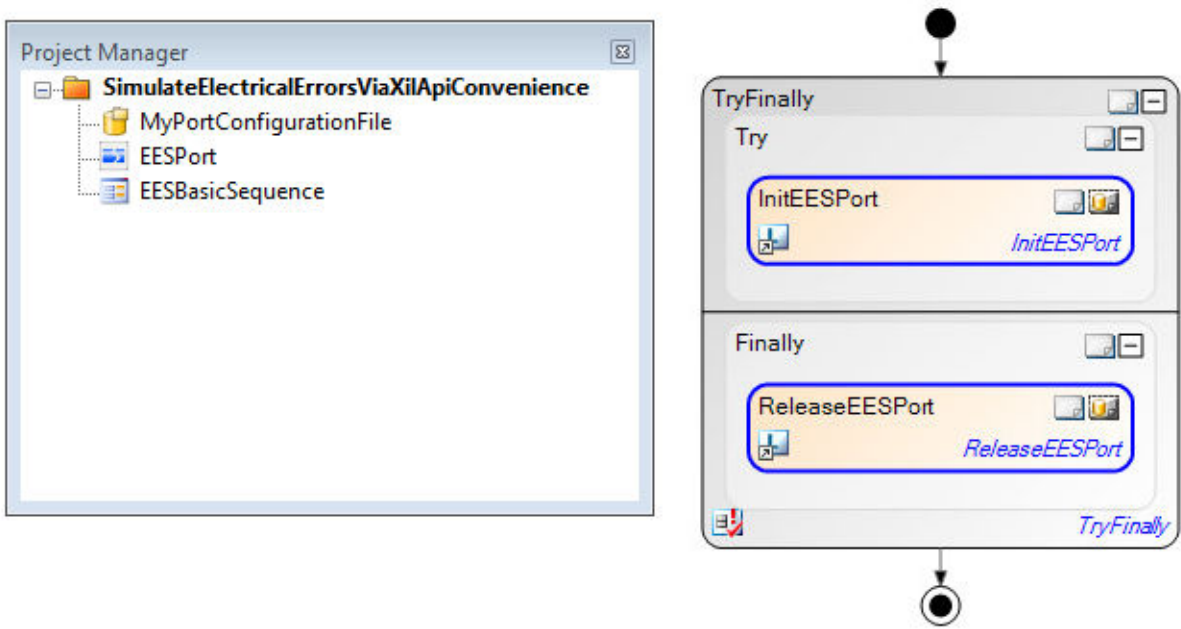
Method

To build a basic sequence for simulating electrical errors

- 1 Add a File data object to your project.
In the example below, this file is renamed to `MyPortConfigurationFile`.
- 2 In the File data object, specify the name and path of the port configuration file (PORTCONFIG) that contains the configuration to access your connected EES hardware.
- 3 Drag an `EESPort` data object from the XIL API library's `EESPort` folder to your project. This instantiates the port configuration to access the EES hardware.
- 4 Drag a `TryFinally` block from the Main Library to your sequence. This ensures that the blocks specified in the `Finally` path are executed after the `Try` path execution, even if an exception occurred.
- 5 Drag an `InitEESPort` block from the XIL API Convenience library's `Electrical Error Simulation Port` folder to the `Try` path. This initializes the access to the EES hardware that performs the electrical error simulation.
By default, the block's `EESPort` data object is set as a reference to the related project-specific data object.
- 6 Set the block's `PortConfigurationFile` data object as a reference to the related project-specific File data object.
- 7 Drag a `ReleaseEESPort` block to the `Finally` path. This releases no longer needed resources and sets the EES hardware to an initial state.

By default, the block's EESPort data object is set as a reference to the related project-specific data object.

Result You created a basic sequence for simulating electrical errors via the connected EES hardware as configured in the specified port configuration file.



Related topics

References

InitEESPort.....	32
ReleaseEESPort.....	34
TryFinally (AutomationDesk Basic Practices 📖)	

How to Simulate a Manually Triggered Error

Objective You can prepare an [error set](#) from scratch for an electrical error that is activated when a Trigger block in your [Sequence](#) is executed.

Basics The XIL API Convenience library provides [blocks](#) that let you simulate an electrical error from scratch. This lets you control the order of the simulated electrical errors from your automation sequence.

The simulation of the electrical error starts with the execution of the **Trigger** block and lasts until the **DeactivateError** block is executed or another error is triggered.

According to the number of affected signals, you can use a **PrepareSinglePinError** or a **PrepareMultiPinError** block to specify which error is to be simulated.

Preconditions

- You have built a basic sequence for simulating electrical errors. For instructions, refer to [How to Build a Basic Sequence for Simulating Electrical Errors](#) on page 16.
- The following information is required as input data:
 - The category of the error to be simulated
 - The name of the signal to be disturbed
 - The load type for each signal to be disturbed, which specifies whether a load rejection is performed when the error occurred

Which signals and which error categories are available depends on the EES hardware that is used. For more information, refer to [Basics on Failure Simulation \(dSPACE XIL API Implementation Guide !\[\]\(339a16584d5da0f0a3ca4e9ec17bf6a1_img.jpg\)](#)).

Restrictions

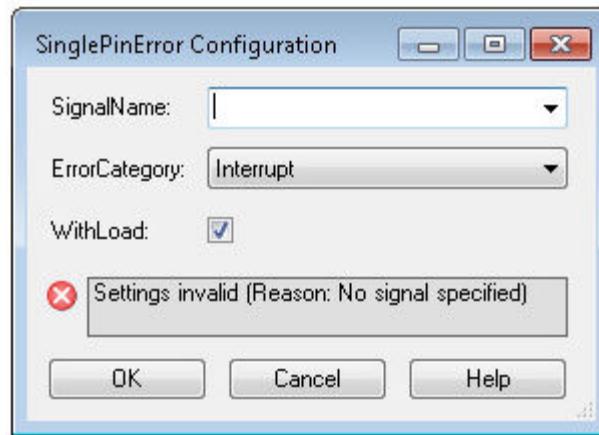
- You can simulate only electrical errors of the **Simple** error type .
- You cannot prepare more than one error at a time.

Method


To simulate a manually triggered electrical error

- 1** Drag a **PrepareSinglePinError** block from the XIL API Convenience library's **Electrical Error Simulation Port – Manually Triggered Errors** folder to the Try path. This enhances the **EESPort** data object with an error configuration that contains an error set with one error of the specified category.
By default, the block's **EESPort** data object is set as a reference to the related project-specific data object.
- 2** In the Try path, select the **InitEESPort** block and press **F5**.
The **InitEESPort** block is executed. It initializes the project-specific **EESPort** data object. This makes the **SinglePinError Configuration** dialog available for the parameterization of the **PrepareSinglePinError** block.
- 3** In the **PrepareSinglePinError** block, double-click one of the data objects **ErrorCategory**, **SignalName**, or **LoadType**.

The SinglePinError Configuration dialog opens.



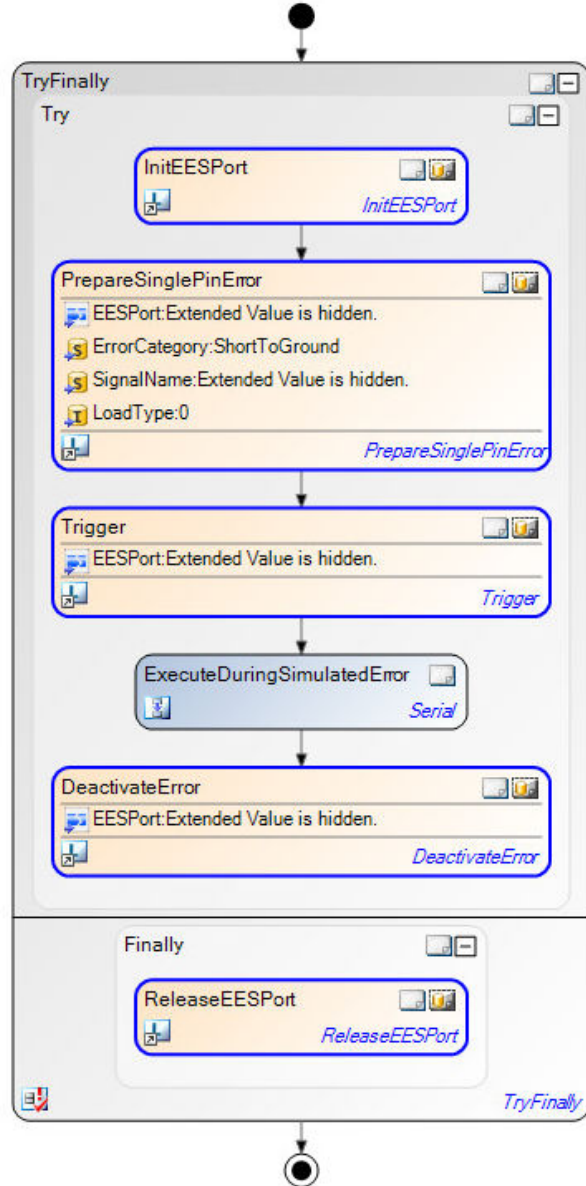
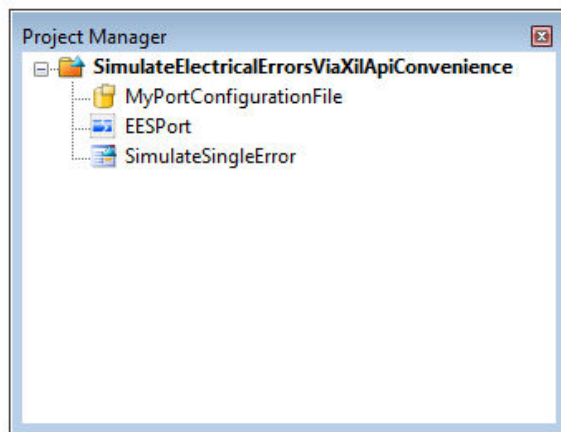
This dialog lets you specify the error category, the signal name, and the load type. It displays whether the parameter combination is valid for the specified EES port.

- 4 In **SignalName**, select the name of the signal to be disturbed by the EES simulation.
- 5 In **ErrorCategory**, select the category of the error to be simulated.
- 6 Clear the **WithLoad** checkbox if you want to perform a load rejection when the error occurs. Select this checkbox if you want the simulator's channels with load to stay connected.
If the combination of error category, signal name, and load type is valid, this is indicated by the  status icon.
- 7 Press **OK** to apply the changes to the **PrepareSinglePinError** block.
- 8 Drag a **Trigger** block to the **Try** path. This starts the error simulation on the EES hardware.
By default, the block's **EESPort** data object is set as a reference to the related project-specific data object.
- 9 Add the blocks to be executed during the simulated error below the **Trigger** block.
- 10 Drag a **DeactivateError** block to the **Try** path. This stops the electrical error simulation on the EES hardware.
By default, the block's **EESPort** data object is set as a reference to the related project-specific data object.

Result

You created a sequence to simulate an error of the specified category that affects the specified signal.

In the example, you disturb Signal 1 with a **ShortToGround** error. When the error occurs, the load will not be disconnected from the simulator platform.



Tip

- You can simulate electrical errors that affect more than one pins by replacing the PrepareSinglePinError block by a PrepareMultiPinError block.
- If you cannot initialize the EESPort data object during implementation, you can specify each of the ErrorCategory, SignalName, and LoadType data objects by using the Value Editor.

Related topics

References

DeactivateError.....	31
Edit (MultiPinError).....	66
Edit (SinglePinError).....	64
PrepareMultiPinError.....	35
PrepareSinglePinError.....	37
Trigger.....	39

How to Simulate a Software-Triggered Error

Objective

You can prepare an [error set](#) from scratch for electrical errors that are activated when a condition is fulfilled or a specified amount of time has passed.

Basics

The XIL API Convenience library provides blocks that let you prepare an electrical error set which is activated later, triggered by a condition or a duration watcher. This lets you control the start of an electrical error simulation via the values of simulator variables.

According to the number of affected signals, you can use a `PrepareSinglePinErrorWithCondition` or a `PrepareMultiPinErrorWithCondition` block to specify which error is to be simulated.

Within the timeout that you specify in the `PrepareSinglePinErrorWithCondition` block or the `PrepareMultiplePinErrorWithCondition` block, the simulation of the electrical error starts when the condition is fulfilled. After this timeout elapses an event is generated.

The error simulation lasts until the `DeactivateError` block is executed or another error set is triggered.

To wait for a specified amount of time for the condition to be fulfilled, you can use a `WaitForTrigger` block. When its timeout is reached, an exception occurs.

The `ElectricalErrorSimulationPort` demo project shows how to react to the different timeouts. It is located in `<DocumentsFolder>\XIL API Convenience`.

Preconditions

- You built a basic sequence for simulating electrical errors. For instructions, refer to [How to Build a Basic Sequence for Simulating Electrical Errors](#) on page 16.
- You downloaded and started your simulation application on your platform. For instructions, refer to [How to Download and Start a Simulation Application \(AutomationDesk Accessing Simulation Platforms\)](#).

- You configured the access to the simulator variables in a variable pool, via a Mapping data object, for example. For instructions, refer to [How to Make Simulator Variables Available \(AutomationDesk Accessing Simulation Platforms\)](#).

- The following information is required as input data:

- The category of the errors to be simulated
- The names of the signals to be disturbed
- The condition that triggers the simulation of the electrical error
- The load type for each signal to be disturbed, which specifies whether a load rejection is performed when the error occurred

Which signals and error categories are available depends on the EES hardware that is used. For more information, refer to [Basics on Failure Simulation \(dSPACE XIL API Implementation Guide\)](#).

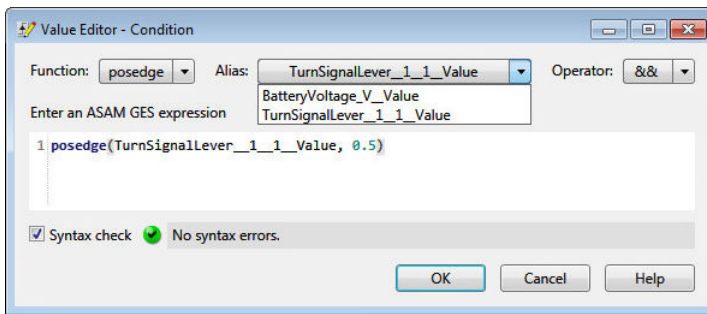
Restrictions

- You can simulate only electrical errors of the Simple error type.
- You cannot prepare more than one error at a time.

Method

To simulate a software-triggered multi-pin error

- 1 Drag a PrepareMultiPinErrorWithCondition block from the XIL API Convenience library's Electrical Error Simulation Port – Software Triggered Errors folder to the Try path. This enhances the EESPort data object with an error configuration that contains an error set with one error of the specified category.
By default, the block's EESPort data object is set as a reference to the related project-specific data object.
By default, the block's Timeout data object is set to -1, which disables the block's timeout.
- 2 In the block's ErrorCategory data object, select the category of the error to be simulated. This error category is then applied simultaneously to all signals that you specify in the SignalNames data object.
- 3 In the block's SignalNames String data object, specify a list of the names of the signals to be disturbed by the EES simulation.
- 4 In the block's LoadTypes List data object, specify the load types for the disturbed signals. Each load type corresponds to the signal at the same position in the SignalNames List data object. Specify 1 if you want to perform a load rejection when the error occurs. If you want the simulator platform to stay connected, specify 0.
- 5 Set the block's Defines data object as a reference to the project-specific Mapping data object. This lets you use all alias names that are contained in the variable pool when you specify the block's Condition.
- 6 In the Data Object Editor, double-click Condition to open the Expression Editor and enter the start condition for simulating the error in the ASAM General Expression Syntax.



When the specified condition is fulfilled, the simulation of the electrical error starts.

- 7 Drag a **WaitForTrigger** block to the Try path.
By default, the block's **EESPort** data object is set as a reference to the related project-specific data object.
- 8 In the block's **Timeout** data object, specify the maximum time to wait for the trigger to occur in milliseconds.
- 9 Add the blocks to be executed during the simulated error below the **WaitForTrigger** block.
- 10 Drag a **DeactivateError** block to the Try path. This stops the electrical error simulation on the EES hardware.
By default, the block's **EESPort** data object is set as a reference to the related project-specific data object.

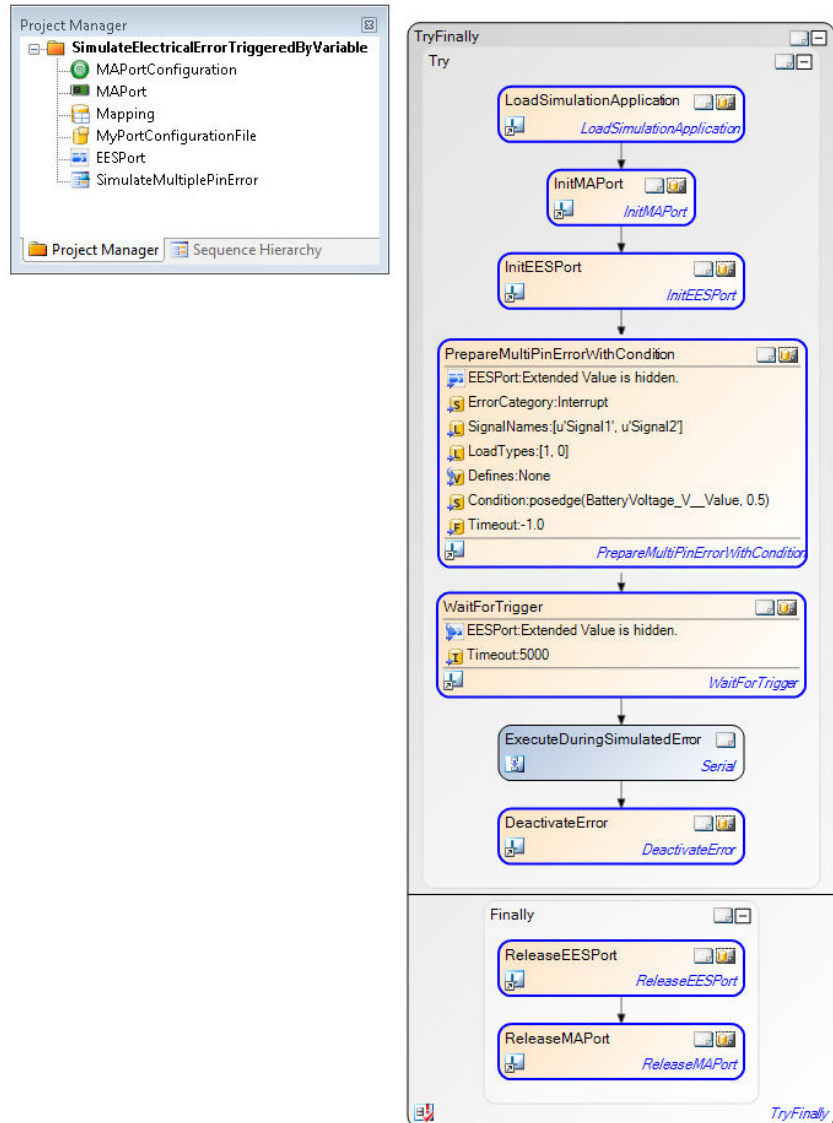
Result

You created a sequence for simulating a software-triggered multi-pin error. An error of the specified category that affects all configured signals simultaneously is simulated when the condition is fulfilled.

In the example, the EES signals **Signal11** and **Signal12** are interrupted when the simulator variable for the signal lever position in the AutomationDesk turn signal demo changes from 0 to 1.

When the error occurs, the load on **Signal11** is disconnected from the simulator platform.

If the condition does not become true within 5000 milliseconds, an exception occurs, i.e., the execution terminates.



Tip

- You can simulate electrical errors that are triggered when a specified amount of time has passed by replacing the PrepareMultiplePinErrorWithCondition block with a PrepareSinglePinErrorWithDuration or a PrepareMultiPinErrorWithDuration block.
- According to the number of affected signals, you can use an Edit (SinglePinError) command or an Edit (MultiPinError) command to specify the error category, the signal names and the load types of the errors to be simulated in a separate dialog.

Related topics

References

Edit (MultiPinError).....	66
Edit (SinglePinError).....	64
PrepareMultiPinErrorWithCondition.....	41
PrepareMultiPinErrorWithDuration.....	44
PrepareSinglePinErrorWithCondition.....	46
PrepareSinglePinErrorWithDuration.....	49
WaitForTrigger.....	51

How to Simulate Errors from an Error Configuration File

Objective

You can read an error configuration from an XML file to an `ErrorConfiguration` data object and process the contained error sets.

Basics

You can reuse the electrical error configuration that you specified graphically with `ControlDesk` or scripted via the `EESPort` implementation of dSPACE XIL.NET. In both cases, the error configuration must be stored in an XML file. For more information on creating such a file, refer to [How to Create and Configure an Electrical Error \(ControlDesk Electrical Error Simulation via XIL API EESPort\)](#) and to [Basic Information on Configuring Errors \(dSPACE XIL API Implementation Guide\)](#).

AutomationDesk's XIL API library provides the data objects needed to read the error configuration from the XML file and to configure the `EESPort` accordingly.

Via a `Trigger` block, you can activate the error sets consecutively in the same order as in the XML file. Starting with the first, each error set is active until the next `Trigger` block is executed. The final `DeactivateError` block makes sure that the last error is deactivated.

Preconditions

- You have built a basic sequence for simulating electrical errors. For instructions, refer to [How to Build a Basic Sequence for Simulating Electrical Errors](#) on page 16.
- The name and path of the [error configuration file](#) is required as input data.

Method

To simulate errors from an error configuration file

- 1 Add a File data object to your project.
Rename the data object `MyErrorConfigurationFile`. In the Python script below, this name is used as the file to read the error sets from.

2 In the **MyErrorConfigurationFile** File data object, specify the name and path of your error configuration file.

3 Drag the following data objects from the XIL API library to your project:

- From the top level:

- **TestbenchFactory**
- **Testbench**

From the EESPort folder:

- **EESPortFactory**
- **ErrorConfiguration**
- **ErrorConfigurationReader**

These data objects are used to provide the methods to read the error configuration file.

4 Add an Int data object to your project.

Rename data object **MyNumberOfErrorSets**. In the Python script below, this name is used as the integer to store the number of read error sets.

5 Add an Exec block with the following Python script to the Try path in the base sequence.

Note that the code contains a literal that specifies the dSPACE XIL API version that might be adapted.

```
# Initialize testbench
_AD_.TestbenchFactory.InitTestbenchFactory()
_AD_.Testbench = _AD_.TestbenchFactory.CreateVendorSpecificTestbench("dSPACE GmbH", "XIL API", "2016-A")
# Create error configuration
_AD_.ErrorConfiguration = _AD_.Testbench.EESPortFactory.CreateErrorConfiguration("MyErrorConfName")
# Create error configuration reader
_AD_.EESConfigurationReader = \
    _AD_.ErrorConfiguration.CreateEESConfigurationFileReader(_AD_.MyErrorConfigurationFile.GetAbsolutePath())
# Read error configuration file
_AD_.ErrorConfiguration.Load(_AD_.EESConfigurationReader)
# Get number of contained error sets
_AD_.MyNumberOfErrorSets = _AD_.ErrorConfiguration.Count
# Set the loaded error configuration to EESPort
_AD_.EESPort.SetErrorConfiguration(_AD_.ErrorConfiguration)
# Download the EES configuration
_AD_.EESPort.Download()
# Start the electrical error simulation
_AD_.EESPort.Activate()
```

This script reads the contents of the error configuration file to the instantiated **ErrorConfiguration** data object and stores the number of contained error sets.

6 Add a For block to the Try path to process the error sets one by one.

7 Set the block's Stop data object as a reference to the **MyNumberOfErrorSets** data object that stores the number of error sets.

8 Drag a Trigger block from the XIL API Convenience library's **Electrical Error Simulation Port** folder to the body of the For block.

By default, the block's **EESPort** data object is set as a reference to the related project-specific data object.

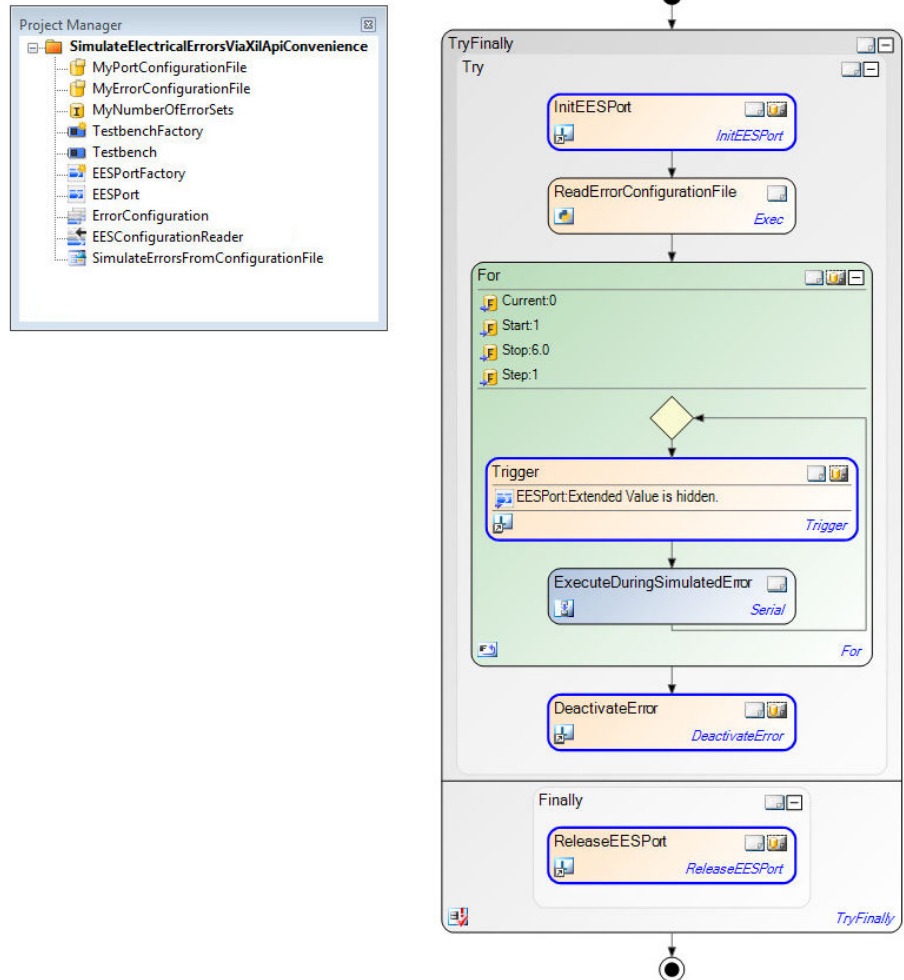
9 Add the blocks to be executed during the simulated error to the body of the For block.

10 Drag a DeactivateError block below the For block. This stops the electrical error simulation on the EES hardware.

By default, the block's EESPort data object is set as a reference to the related project-specific data object.

Result



You created a sequence that reads the specified error configuration file and simulates the contained error sets one by one.



In the example, the contents of the error configuration file is read to the ErrorConfiguration data object. In the For block, one after the other of the contained error sets are activated, i.e., the specified electrical errors are simulated by the EES hardware. The last error is terminated when the DeactivateError block is executed.

Related topics

References

DeactivateError.....	31
EESConfigurationReader (Data Object).....	56
EESPortFactory (Data Object).....	55
ErrorConfiguration (Data Object).....	56
For (AutomationDesk Basic Practices )	
Testbench (Data Object) (AutomationDesk Accessing Simulation Platforms )	
Trigger.....	39

Reference Information

Where to go from here

Information in this section

Automation Blocks.....	30
Commands And Dialogs.....	64

Automation Blocks

Where to go from here	Information in this section
	XIL API Convenience (Electrical Error Simulation)..... 30
	Description of the specific blocks and data objects for electrical error simulation using the XIL API Convenience library.
	XIL API (Electrical Error Simulation)..... 53
	Description of the specific blocks and data objects for electrical error simulation using the XIL library.

XIL API Convenience (Electrical Error Simulation)

Introduction	AutomationDesk provides specific blocks and data objects for electrical error simulation using the XIL API Convenience library.
---------------------	---

Where to go from here	Information in this section
	Main Elements..... 30
	Manually Triggered Errors..... 35
	Software Triggered Errors..... 40

Main Elements

Introduction	The main elements of the library are directly available in the Electrical Error Simulation Port folder.
---------------------	---

Where to go from here	Information in this section
	DeactivateError..... 31
	To stop the simulation of errors configured by the referenced EESPort data object.

InitEESPort.....	32
To initialize the electrical error simulation port data object (EESPort).	
ReleaseEESPort.....	34
To release an EESPort instance.	

DeactivateError

Graphical representation



Purpose

To stop the simulation of errors configured by the referenced EESPort data object.

Description

The DeactivateError automation block is used to deactivate a running error configuration on the connected EES hardware. The downloaded error configuration is deleted from the hardware. You have to use this block before you can prepare another error via the PrepareSinglePinError or PrepareMultiPinError block.

Data objects

This automation block provides the following data objects:

Name	In/Out	Type	Default Value	Description
EESPort	In	EESPort (Data Object)	None	Specifies the EESPort data object that you created using the InitEESPort block.

Access via Exec block

The Python module `audxilapiconveniencelib` provides the `DeactivateError` method, which uses the same arguments in the same order, with the same semantics as the DeactivateError block.

Example

```
import audxilapiconveniencelib
audxilapiconveniencelib.DeactivateError(_AD_.EESPort)
```

For examples of how to use the EESPort blocks in the XIL API Convenience library and how to access the basic methods of the XIL API standard in Exec blocks, refer to the ElectricalErrorSimulationPort demo project.

Related topics**Basics**

[Basics of the XIL API Library Elements \(AutomationDesk Accessing Simulation Platforms !\[\]\(3d8c13c92b853674f749aac6fa869926_img.jpg\)\)](#)

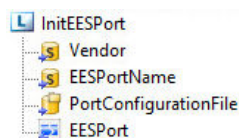
HowTos

[How to Simulate a Manually Triggered Error..... 17](#)
[How to Simulate Errors from an Error Configuration File..... 25](#)

References

[EESPort \(Data Object\)..... 54](#)
[InitEESPort..... 32](#)
[Trigger..... 39](#)

InitEESPort

Graphical representation**Purpose**

To initialize the electrical error simulation port data object (EESPort).

Description

The InitEESPort automation block is used to create and configure an EESPort instance.

The electrical error simulation port (EESPort) provides access to a failure insertion unit (FIU) for simulating failures in the ECU wiring. You can access the simulation platform with the failure insertion units only if the simulation application is already running. The EESPort implementation does not contain functions for registering the platform and downloading the simulation application to it. After registering the platform, you can use the LoadSimulationApplication block to download and start the simulation application.

An initialized EESPort data object gets the basic information on the FIU hardware to be used in XML format via the port configuration file. The port configuration file must contain the data for the interface used for the FIU hardware connection. Depending on the selected driver type, you must specify additional data. The configuration file also contains the path to the signal list of your simulator. The list provides information on the available signals and potentials. If your platform provides several potentials, you must specify how to map them in the configuration file.

<InstallationPath_dSPACE_XILAPI.NET>\Demos\EESPort\CommonData\PortConfigurations contains some examples of EESPort configuration files. These files are also used in the XIL API Convenience demo project for the EESPort.

For detailed information on the dSPACE EESPort configuration file, refer to [dSPACE EESPort Configuration File \(dSPACE XIL API Reference \[1\]\)](#).

Data objects

This automation block provides the following data objects:

Name	In/Out	Type	Default Value	Description
Vendor	In	String	"dSPACE"	Specifies the vendor and version of the used XIL API implementation. With dSPACE specified as the vendor, the current version of the dSPACE XIL API implementation is used. To specify another version or another vendor, use the following format: <div><VendorName>_<ProductName>_<ProductVersion></div> Example: <div>dSPACE GmbH_XIL_API_2019-A</div> It is recommended to use a globally managed Vendor data object as a reference providing the vendor name. For more information, refer to Vendor (Data Object) (AutomationDesk Accessing Simulation Platforms [1]) .
EESPortName	In	String	"DefaultName"	Specifies a custom name for the instantiated EESPort.
PortConfigurationFile	In	File	None	Specifies the EESPort configuration file that provides information on the FIU hardware.
EESPort	Out	EESPort	None	Returns the initialized EESPort data object.

Access via Exec block

The Python module `audxilapiconveniencelib` provides the `InitEESPort` method, which uses the same arguments in the same order, with the same semantics as the `InitEESPort` block. The return value corresponds to the output data object.

Example

```
import audxilapiconveniencelib
_AD_EESPort = audxilapiconveniencelib.InitEESPort(
    "dSPACE",
    "MyEESPortName",
    _AD.PortConfigFile)
```

For examples of how to use the EESPort blocks in the XIL API Convenience library and how to access the basic methods of the XIL API standard in Exec blocks, refer to the `ElectricalErrorSimulationPort` demo project.

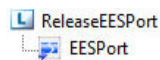
Related topics**HowTos**

[How to Build a Basic Sequence for Simulating Electrical Errors.....](#) 16

References

[dSPACE EESPort Configuration File \(dSPACE XIL API Reference !\[\]\(339a16584d5da0f0a3ca4e9ec17bf6a1_img.jpg\)\)](#)
[EESPort \(Data Object\).....](#) 54
[LoadSimulationApplication \(AutomationDesk Accessing Simulation Platforms !\[\]\(e06a1d39938b2f5d7a2c3618fea4f77f_img.jpg\)\)](#)

ReleaseEESPort

Graphical representation**Purpose**

To release an EESPort instance.

Description

The ReleaseEESPort automation block is used to clear the instantiated EESPort.

Data objects

This automation block provides the following data object:

Name	In/Out	Type	Default Value	Description
EESPort	In	EESPort (Data Object)	None	Specifies the EESPort data object that you created using the InitEESPort block.

Access via Exec block

The Python module `audxilapiconveniencelib` provides the `ReleaseEESPort` method with the same semantics as the ReleaseEESPort block.

Example

```
import audxilapiconveniencelib
audxilapiconveniencelib.ReleaseEESPort(_AD_.EESPortObject)
```

For examples of how to use the EESPort blocks in the XIL API Convenience library and how to access the basic methods of the XIL API standard in Exec blocks, refer to the ElectricalErrorSimulationPort demo project.

Related topics**Basics**

[Basics of the XIL API Library Elements \(AutomationDesk Accessing Simulation Platforms !\[\]\(4729e517bc6a7cd81c8025b9646574fb_img.jpg\)\)](#)

HowTos

[How to Build a Basic Sequence for Simulating Electrical Errors..... 16](#)

References

[EESPort \(Data Object\)..... 54](#)

Manually Triggered Errors

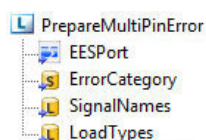
Introduction

You can use the blocks of the **Manually Triggered Errors** library folder to create electrical errors and trigger their simulation immediately.

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

PrepareMultiPinError.....	35
To create an error set that affects multiple pins.	
PrepareSinglePinError.....	37
To create an error set that affects a single pin.	
Trigger.....	39
To activate an error.	

PrepareMultiPinError

Graphical representation**Purpose**

To create an error set that affects multiple pins.

Description

The PrepareMultiPinError automation block is used to create an error set that affects multiple pins of the EES hardware. If you activate a multi-pin error, the specified signals are switched simultaneously.

According to the XIL API standard, the block automatically creates the required error configuration, including the error set that is configured as specified by the block's parameters. You cannot prepare more than one error configuration for the same EESPort at the same time.

If you parameterize the block with an EESPort data object that you initialized before, for example, by using an InitEESPort block, you can edit the block's error set via the **MultiPinError Configuration** dialog. This lets you specify the error set by selection instead of specifying each of its data objects manually. For more information, refer to **Edit (MultiPinError)**.

You can activate the error set that you created by using a Trigger block.

If the XIL API library is in offline mode, this block is ignored.

Data objects

This automation block provides the following data objects:

Name	In/Out	Type	Default Value	Description
EESPort	In	EESPort (Data Object)	None	Specifies the EESPort data object that you created using the InitEESPort block.
ErrorCategory	In	String	"Interrupt"	Specifies the category of the error. You can select: <ul style="list-style-type: none"> ▪ Interrupt ▪ ShortToGround ▪ ShortToUbat ▪ Pin2Pin For detailed information on the error categories, refer to BaseError (Data Object) on page 60.
SignalNames	In	List	[]	Specifies the names of the signals to be affected by the error. The signal names must exist in the signal list of your simulation platform.
LoadTypes	In	List	[]	Specifies the load types to be applied for the specified signals. <ul style="list-style-type: none"> ▪ 0: The load will not be disconnected from the simulator platform when a signal is disturbed. ▪ 1: The load will be disconnected from the simulator platform when a signal is disturbed. Each value in this load type list relates to the signal with the same position in the signal list.

Access via Exec block

The Python module `audxilapiconvenience.lib` provides the `PrepareMultiPinError` method, which uses the same arguments in the same

order, with the same semantics as the PrepareMultiPinError block. For manually triggered errors, the WatcherObject parameter is not used.


Example

```
import audxilapiconveniencelib
audxilapiconveniencelib.PrepareMultiPinError(
    _AD_.EESPort,
    "Interrupt",
    ["Signal_a", "Signal_b"],
    [0, 1])
```

For examples of how to use the EESPort blocks in the XIL API Convenience library and how to access the basic methods of the XIL API standard in Exec blocks, refer to the ElectricalErrorSimulationPort demo project.

Related topics

Basics

Basics of the XIL API Library Elements (AutomationDesk Accessing Simulation Platforms )

HowTos

How to Simulate a Manually Triggered Error..... 17

References

Edit (MultiPinError)..... 66

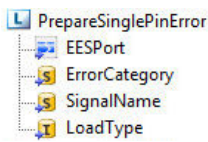
InitEESPort..... 32

PrepareSinglePinError..... 37

Trigger..... 39

PrepareSinglePinError

Graphical representation



Purpose	To create an error set that affects a single pin.
Description	The PrepareSinglePinError automation block is used to create an error set that affects a single pin of the EES hardware.

According to the XIL API standard, the block automatically creates the required error configuration, including the error set that is configured as specified by the block's parameters. You cannot prepare more than one error configuration for the same EESPort at the same time.

If you parameterize the block with an EESPort data object that you initialized before, for example, by using an InitEESPort block, you can edit the block's error set via the **SinglePinError Configuration** dialog. This lets you specify the error set by selection instead of specifying each of the data objects manually. For more information, refer to **Edit (SinglePinError)**.

You can activate the error set that you created by using a **Trigger** block.

If the XIL API library is in offline mode, this block is ignored.

Data objects

This automation block provides the following data objects:

Name	In/Out	Type	Default Value	Description
EESPort	In	EESPort (Data Object)	None	Specifies the EESPort data object that you created using the InitEESPort block.
ErrorCategory	In	String	"Interrupt"	Specifies the category of the error. You can select: <ul style="list-style-type: none"> ▪ Interrupt ▪ ShortToGround ▪ ShortToUbat For detailed information on the error categories, refer to BaseError (Data Object) on page 60.
SignalName	In	String	" "	Specifies the name of the signal to be affected by the error. The signal name must exist in the signal list of your simulation platform.
LoadType	In	Int	0	Specifies the load type to be applied for the specified signal. <ul style="list-style-type: none"> ▪ 0: The load will not be disconnected from the simulator platform when a signal is disturbed. ▪ 1: The load will be disconnected from the simulator platform when a signal is disturbed.

Access via Exec block

The Python module `audxilapiconvenience.lib` provides the `PrepareSinglePinError` method, which uses the same arguments in the same order, with the same semantics as the `PrepareSinglePinError` block. For manually triggered errors, the `WatcherObject` parameter is not used.

Example

```
import auxilapiconveniencelib
auxilapiconveniencelib.PrepareSinglePinError(
    _AD_.EESPort,
    "Interrupt",
    "Signal_a",
    1)
```

For examples of how to use the EESPort blocks in the XIL API Convenience library and how to access the basic methods of the XIL API standard in Exec blocks, refer to the `ElectricalErrorSimulationPort` demo project.

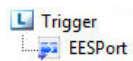
Related topics**HowTos**

[How to Simulate a Manually Triggered Error.....](#) 17

References

[Edit \(SinglePinError\).....](#) 64

Trigger

Graphical representation**Purpose**

To activate an error.

Description

The Trigger automation block is used to activate the error that was specified for the referenced EESPort data object via the `PrepareSinglePinError` or `PrepareMultiPinError` block. According to the XIL API standard, the trigger activates an error set that is located in an error configuration. Each error set within the current error configuration must be started by a separate trigger. The errors within each error set are then activated without further triggers.

If the XIL API library is in offline mode, this block is ignored.

Data objects

This automation block provides the following data object:

Name	In/Out	Type	Default Value	Description
EESPort	In	EESPort (Data Object)	None	Specifies the EESPort data object that you created using the <code>InitEESPort</code> block.

Access via Exec block

The Python module `audxilapiconveniencelib` provides the `Trigger` method with the same semantics as the `Trigger` block.

Example

```
import audxilapiconveniencelib
audxilapiconveniencelib.Trigger(_AD_.EESPort)
```

You can also use the Exec block to trigger not only the current error configured by a `PrepareSinglePinError` or `PrepareMultiPinError` block, but a series of errors or error sets.

For examples of how to use the EESPort blocks in the XIL API Convenience library and how to access the basic methods of the XIL API standard in Exec blocks, refer to the `ElectricalErrorSimulationPort` demo project.

Related topics**Basics**

[Basics of the XIL API Library Elements \(AutomationDesk Accessing Simulation Platforms !\[\]\(f95dab70c751fda7d824b8b03650f7aa_img.jpg\)](#))

HowTos

[How to Simulate a Manually Triggered Error..... 17](#)
[How to Simulate Errors from an Error Configuration File..... 25](#)

Software Triggered Errors

Introduction

You can use the blocks of the `Software Triggered Errors` library folder to create electrical errors that are triggered by a duration watcher or a condition watcher.

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

[PrepareMultiPinErrorWithCondition..... 41](#)
 To create an error set that affects multiple pins and that is triggered by a condition.

[PrepareMultiPinErrorWithDuration..... 44](#)
 To create an error set that affects multiple pins and that is triggered after a duration.

[PrepareSinglePinErrorWithCondition..... 46](#)
 To create an error set that affects a single pin and that is triggered by a condition.

PrepareSinglePinErrorWithDuration.....49

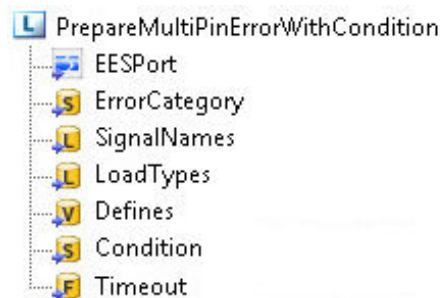
To create an error set that affects a single pin and that is triggered after a duration.

WaitForTrigger.....51

To wait until the next error set is triggered for the specified EES port.

PrepareMultiPinErrorWithCondition

Graphical representation



Purpose

To create an error set that affects multiple pins and that is triggered by a condition.

Description

The PrepareMultiPinErrorWithCondition automation block is used to create an error set that affects multiple pins of the EES hardware, i.e., the specified signals are switched simultaneously. This error set is triggered by a condition watcher, which means that the error set is activated when the specified condition is fulfilled.

According to the XIL API standard, the block automatically creates the required error configuration, including the error set that is configured as specified by the block's parameters. You cannot prepare more than one error configuration for the same EESPort at the same time.

If you parameterize the block with an EESPort data object that you initialized before, for example, by using an InitEESPort block, you can edit the block's error set via the **MultiPinError Configuration** dialog. This lets you specify the error set by selection instead of specifying each of its data objects manually. For more information, refer to **Edit (MultiPinError)**.

You can specify a timeout for the triggering, which generates an event, if the trigger does not occur within the specified time. For this, no changes are made to the EES hardware.

A disabled timeout corresponds to an infinite timeout.

To wait for the triggering of the error set that was created, you can use the WaitForTrigger automation block.

If the XIL API library is in offline mode, this block is ignored.

Data objects

This automation block provides the following data objects:

Name	In/Out	Type	Default Value	Description
EESPort	In	EESPort (Data Object)	None	Specifies the EESPort data object that you created using the InitEESPort block.
ErrorCategory	In	String	"Interrupt"	Specifies the category of the error. You can select: <ul style="list-style-type: none"> Interrupt ShortToGround ShortToUbat Pin2Pin For detailed information on the error categories, refer to BaseError (Data Object) on page 60.
SignalNames	In	List	[]	Specifies the names of the signals to be affected by the error. The signal names must exist in the signal list of your simulation platform.
LoadTypes	In	List	[]	Specifies the load types to be applied for the specified signals. <ul style="list-style-type: none"> 0: The load will not be disconnected from the simulator platform when a signal is disturbed. 1: The load will be disconnected from the simulator platform when a signal is disturbed. Each value in this load type list relates to the signal with the same position in the signal list.
Defines	In	Variant	None	Lets you specify the set of variable aliases that can be used in the block's Condition data object by referencing a project-specific Mapping data object, a data container, or a Dictionary data object. <ul style="list-style-type: none"> Using a Mapping data object: You have to add each variable to the Mapping data object via the Mapping Editor. Using a DataContainer: For each variable, you have to add a String data object to the container, where the name of the String data object is the VariableName, and the value of the String data object is the ModelPath. Using a Dictionary: For each variable, you have to add a key-value pair as <VariableName>:<ModelPath>.
Condition	In	String	" "	Lets you specify the trigger condition for simulating the electrical error in ASAM General Expression Syntax. You can use the variables from the block's Defines data object.

Name	In/ Out	Type	Default Value	Description
Timeout	In	Float	-1.0	<p>Example: <code>posedge(BatteryVoltage, 12.0)</code></p> <p>If you use the Expression Editor to specify the string, the condition can be syntactically checked while you are editing it. For more information, refer to <i>Appendix A. Syntax of Watcher Conditions</i> in ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf</p> <p>Lets you specify the maximum time to wait for the EESPort trigger to occur in seconds.</p> <p>To disable the timeout, set the value of this data object to -1.</p>

Access via Exec block

The Python module `audxilapiconveniencelib` provides the `PrepareMultiPinErrorWithCondition` method, which uses the same arguments in the same order, with the same semantics as the `PrepareMultiPinErrorWithCondition` block.

Example

```
import audxilapiconveniencelib
audxilapiconveniencelib.PrepareMultiPinErrorWithCondition(
    _AD_.EESPort,
    "Interrupt",
    ["Signal_a", "Signal_b"],
    [0, 1],
    _AD_.Mapping,
    "TurnSignalLever==1",
    -1)
```

For examples of how to use the EESPort blocks in the XIL API Convenience library and how to access the basic methods of the XIL API standard in Exec blocks, refer to the `ElectricalErrorSimulationPort` demo project.

Related topics

Basics

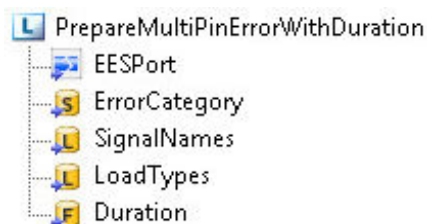
[Basics on Simulating Electrical Errors via the XIL API Convenience Library..... 11](#)

References

[Edit \(MultiPinError\)..... 66](#)
[PrepareMultiPinErrorWithDuration..... 44](#)
[PrepareSinglePinErrorWithCondition..... 46](#)
[PrepareSinglePinErrorWithDuration..... 49](#)
[WaitForTrigger..... 51](#)

PrepareMultiPinErrorWithDuration

Graphical representation



Purpose

To create an error set that affects multiple pins and that is triggered after a duration.

Description

The PrepareMultiPinErrorWithDuration automation block is used to create an error set that affects multiple pins of the EES hardware, i.e., the specified signals are switched simultaneously. This error is triggered by a duration watcher. This means that the specified error is activated when the duration has been exceeded.

According to the XIL API standard, the block automatically creates the required error configuration, including the error set that is configured as specified by the block's parameters. You cannot prepare more than one error configuration for the same EESPort at the same time.

If you parameterize the block with an EESPort data object that you initialized before, for example, by using an InitEESPort block, you can edit the block's error set via the **MultiPinError Configuration** dialog. This lets you specify the error set by selection instead of specifying each of its data objects manually. For more information, refer to **Edit (MultiPinError)**.

To wait for the triggering of the error set that was created, you can use the WaitForTrigger automation block.

If the XIL API library is in offline mode, this block is ignored.

Data objects

This automation block provides the following data objects:

Name	In/Out	Type	Default Value	Description
EESPort	In	EESPort (Data Object)	None	Specifies the EESPort data object that you created using the InitEESPort block.
ErrorCategory	In	String	"Interrupt"	Specifies the category of the error. You can select: <ul style="list-style-type: none"> ▪ Interrupt ▪ ShortToGround ▪ ShortToUbat ▪ Pin2Pin

Name	In/ Out	Type	Default Value	Description
SignalNames	In	List	[]	For detailed information on the error categories, refer to BaseError (Data Object) on page 60. Specifies the names of the signals to be affected by the error. The signal names must exist in the signal list of your simulation platform.
LoadTypes	In	List	[]	Specifies the load types to be applied for the specified signals. <ul style="list-style-type: none"> 0: The load will not be disconnected from the simulator platform when a signal is disturbed. 1: The load will be disconnected from the simulator platform when a signal is disturbed. Each value in this load type list relates to the signal with the same position in the signal list.
Duration	In	Float	0.0	Specifies the duration in seconds.

Access via Exec block

The Python module `audxilapiconveniencelib` provides the `PrepareMultiPinErrorWithDuration` method, which uses the same arguments in the same order, with the same semantics as the `PrepareMultiPinErrorWithDuration` block.

Example

```
import audxilapiconveniencelib
audxilapiconveniencelib.PrepareMultiPinErrorWithDuration(
    _AD_.EESPort,
    "Interrupt",
    ["Signal_a", "Signal_b"],
    [0, 1],
    5.2)
```

For examples of how to use the EESPort blocks in the XIL API Convenience library and how to access the basic methods of the XIL API standard in Exec blocks, refer to the `ElectricalErrorSimulationPort` demo project.

Related topics

Basics

[Basics on Simulating Electrical Errors via the XIL API Convenience Library](#)..... 11

References

[Edit \(MultiPinError\)](#)..... 66
[PrepareMultiPinErrorWithCondition](#)..... 41
[PrepareSinglePinErrorWithCondition](#)..... 46
[PrepareSinglePinErrorWithDuration](#)..... 49
[WaitForTrigger](#)..... 51

PrepareSinglePinErrorWithCondition

Graphical representation



Purpose

To create an error set that affects a single pin and that is triggered by a condition.

Description

The PrepareSinglePinErrorWithCondition automation block is used to create an error set that affects a single pin of the EES hardware. This error is triggered by a condition watcher, which means that the error set is activated when the specified condition is fulfilled.

According to the XIL API standard, the block automatically creates the required error configuration, including the error set that is configured as specified by the block's parameters. You cannot prepare more than one error configuration for the same EESPort at the same time.

If you parameterize the block with an EESPort data object that you initialized before, for example, by using an InitEESPort block, you can edit the block's error set via the **SinglePinError Configuration** dialog. This lets you specify the error set by selection instead of specifying each of the data objects manually. For more information, refer to **Edit (SinglePinError)**.

You can specify a timeout for the triggering, which generates an event, if the trigger does not occur within the specified time. For this, no changes are made to the EES hardware.

A disabled timeout corresponds to an infinite timeout.

To wait for the triggering of the error set that was created, you can use the WaitForTrigger automation block.

If the XIL API library is in offline mode, this block is ignored.

Data objects

This automation block provides the following data objects:

Name	In/Out	Type	Default Value	Description
EESPort	In	EESPort (Data Object)	None	Specifies the EESPort data object that you created using the InitEESPort block.
ErrorCategory	In	String	"Interrupt"	<p>Specifies the category of the error.</p> <p>You can select:</p> <ul style="list-style-type: none"> ▪ Interrupt ▪ ShortToGround ▪ ShortToUbat <p>For detailed information on the error categories, refer to BaseError (Data Object) on page 60.</p>
SignalName	In	String	" "	Specifies the name of the signal to be affected by the error. The signal name must exist in the signal list of your simulation platform.
LoadType	In	Int	0	<p>Specifies the load type to be applied for the specified signal.</p> <ul style="list-style-type: none"> ▪ 0: The load will not be disconnected from the simulator platform when a signal is disturbed. ▪ 1: The load will be disconnected from the simulator platform when a signal is disturbed.
Defines	In	Variant	None	<p>Lets you specify the set of variable aliases that can be used in the block's Condition data object by referencing a project-specific Mapping data object, a data container, or a Dictionary data object.</p> <ul style="list-style-type: none"> ▪ Using a Mapping data object: You have to add each variable to the Mapping data object via the Mapping Editor. ▪ Using a DataContainer: For each variable, you have to add a String data object to the container, where the name of the String data object is the VariableName, and the value of the String data object is the ModelPath. ▪ Using a Dictionary: For each variable, you have to add a key-value pair as <VariableName>:<ModelPath>.
Condition	In	String	" "	<p>Lets you specify the trigger condition for simulating the electrical error in ASAM General Expression Syntax. You can use the variables from the block's Defines data object.</p> <p>Example: <code>posedge(BatteryVoltage, 12.0)</code></p> <p>If you use the Expression Editor to specify the string, the condition can be syntactically checked while you are editing it.</p> <p>For more information, refer to <i>Appendix A. Syntax of Watcher Conditions</i> in ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf</p>

Name	In/ Out	Type	Default Value	Description
Timeout	In	Float	-1.0	Lets you specify the maximum time to wait for the EESPort trigger to occur in seconds. To disable the timeout, set the value of this data object to -1.

Access via Exec block

The Python module `audxilapiconveniencelib` provides the `PrepareSinglePinErrorWithCondition` method, which uses the same arguments in the same order, with the same semantics as the `PrepareSinglePinErrorWithCondition` block.

Example

```
import audxilapiconveniencelib
audxilapiconveniencelib.PrepareSinglePinErrorWithCondition(
    _AD_.EESPort,
    "Interrupt",
    "Signal_a",
    1,
    _AD_.Mapping,
    "TurnSignalLever==1",
    -1)
```

For examples of how to use the EESPort blocks in the XIL API Convenience library and how to access the basic methods of the XIL API standard in Exec blocks, refer to the `ElectricalErrorSimulationPort` demo project.

Related topics**Basics**

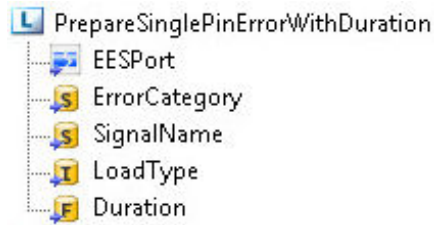
[Basics on Simulating Electrical Errors via the XIL API Convenience Library..... 11](#)

References

[BaseError \(Data Object\)..... 60](#)
[Edit \(SinglePinError\)..... 64](#)
[PrepareMultiPinErrorWithCondition..... 41](#)
[PrepareMultiPinErrorWithDuration..... 44](#)
[PrepareSinglePinErrorWithDuration..... 49](#)
[WaitForTrigger..... 51](#)

PrepareSinglePinErrorWithDuration

Graphical representation



Purpose

To create an error set that affects a single pin and that is triggered after a duration.

Description

The PrepareSinglePinErrorWithDuration automation block is used to create an error set that affects a single pin of the EES hardware. This error is triggered by a duration watcher. This means that the specified error is activated when the duration has been exceeded.

According to the XIL API standard, the block automatically creates the required error configuration, including the error set that is configured as specified by the block's parameters. You cannot prepare more than one error configuration for the same EESPort at the same time.

If you parameterize the block with an EESPort data object that you initialized before, for example, by using an InitEESPort block, you can edit the block's error set via the **SinglePinError Configuration** dialog. This lets you specify the error set by selection instead of specifying each of the data objects manually. For more information, refer to **Edit (SinglePinError)**.

To wait for the triggering of the error set that was created, you can use the WaitForTrigger automation block.

If the XIL API library is in offline mode, this block is ignored.

Data objects

This automation block provides the following data objects:

Name	In/Out	Type	Default Value	Description
EESPort	In	EESPort (Data Object)	None	Specifies the EESPort data object that you created by using the InitEESPort block.
ErrorCategory	In	String	"Interrupt"	Specifies the category of the error. You can select: <ul style="list-style-type: none"> Interrupt ShortToGround ShortToUbat

Name	In/ Out	Type	Default Value	Description
SignalName	In	String	" "	For detailed information on the error categories, refer to BaseError (Data Object) on page 60. Specifies the name of the signal to be affected by the error. The signal name must exist in the signal list of your simulation platform.
LoadType	In	Int	0	Specifies the load type to be applied for the specified signal. <ul style="list-style-type: none"> 0: The load will not be disconnected from the simulator platform when a signal is disturbed. 1: The load will be disconnected from the simulator platform when a signal is disturbed.
Duration	In	Float	0.0	Specifies the duration in seconds.

Access via Exec block

The Python module `audxilapiconveniencelib` provides the `PrepareSinglePinErrorWithDuration` method, which uses the same arguments in the same order, with the same semantics as the `PrepareSinglePinErrorWithDuration` block.

Example

```
import audxilapiconveniencelib
audxilapiconveniencelib.PrepareSinglePinErrorWithDuration(
    _AD_.EESPort,
    "Interrupt",
    "Signal_a",
    1,
    5.2)
```

For examples of how to use the EESPort blocks in the XIL API Convenience library and how to access the basic methods of the XIL API standard in Exec blocks, refer to the `ElectricalErrorSimulationPort` demo project.

Related topics**Basics**

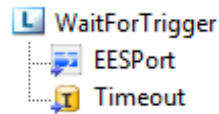
[Basics on Simulating Electrical Errors via the XIL API Convenience Library](#)..... 11

References

[PrepareMultiPinErrorWithCondition](#)..... 41
[PrepareMultiPinErrorWithDuration](#)..... 44
[PrepareSinglePinErrorWithCondition](#)..... 46
[WaitForTrigger](#)..... 51

WaitForTrigger

Graphical representation



Purpose

To wait until the next error set is triggered for the specified EES port.

Description

This block is used to wait for electrical errors that you prepared to be triggered by a condition watcher or a duration watcher, e.g., via one of the following blocks:

- PrepareSinglePinErrorWithCondition
- PrepareSinglePinErrorWithDuration
- PrepareMultiPinErrorWithCondition
- PrepareMultiPinErrorWithDuration

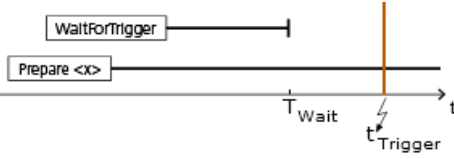
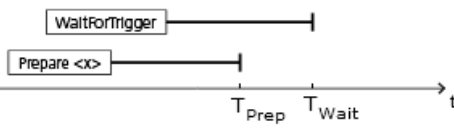
Note

You must specify a timeout, which defines the maximum time to wait for the trigger that is prepared for the specified EES port.

- If there is a duration watcher prepared for the EES port, the WaitForTrigger block's timeout must be greater than the duration of the watcher.
- If there is a condition watcher prepared for the EES port, the WaitForTrigger block's timeout must be greater than the timeout of the watcher. This does not apply, when the condition watcher's timeout is disabled, i.e., set to infinity.

The block's timeout interacts with the timeouts of the Prepare<x> blocks and the occurrence of the trigger as shown in the following table:

Timeouts and Trigger Occurrence on the Timescale	Time Interval ^{1), 2), 3)}	Reaction
	$T_{\text{Prep}} = \text{Infinity}$ $t_{\text{Trigger}} < T_{\text{Wait}}$	At t_{Trigger} the prepared error set is simulated on the EES hardware. AutomationDesk proceeds with the block after the WaitForTrigger block. This is the preferred behavior for most use cases.
	$t_{\text{Trigger}} < T_{\text{Prep}} < T_{\text{Wait}}$	

Timeouts and Trigger Occurrence on the Timescale	Time Interval ^{1), 2), 3)}	Reaction
	$T_{Prep} = \text{Infinity}$ $T_{Wait} < t_{Trigger}$	<p>At T_{Wait} an exception is generated. If AutomationDesk is executing a TryExcept block, the Exception path is executed. Otherwise the execution stops. This is the same behavior, as if the trigger never occurs.</p>
	$T_{Prep} < T_{Wait}$ The trigger does not occur.	<p>At T_{Prep} an event is generated that you can use to start an event handling routine⁴⁾. The electrical error simulation hardware is left unchanged.</p> <p>At T_{Wait} an exception is generated. If AutomationDesk is executing a TryExcept block, the Exception path is executed. Otherwise the execution stops.</p>

¹⁾ T_{Prep} : Timeout of the Prepare<x> block

²⁾ T_{Wait} : Timeout of the WaitForTrigger block

³⁾ $t_{Trigger}$: Time, when the watcher triggers, because the specified condition is fulfilled, or the specified duration has been exceeded.

⁴⁾ For further information, refer to the SimpleErrorConfiguration_Demo sequence in the ElectricalErrorSimulationPort demo project.

If the XIL API library is in offline mode, this block is ignored.

Data objects

This automation block provides the following data object:

Name	In/Out	Type	Default Value	Description
EESPort	In	EESPort (Data Object)	None	Specifies the EESPort data object that you created using the InitEESPort block.
Timeout	In	Int	0	Lets you specify the maximum time to wait for the EESPort trigger to occur in milliseconds.

Access via Exec block

The Python module `audxilapiconveniencelib` provides the `WaitForTrigger` method with the same semantics as the WaitForTrigger block.

Example

```
import audxilapiconveniencelib
audxilapiconveniencelib.WaitForTrigger(
    _AD_.EESPort,
    5000)
```

For examples of how to use the EESPort blocks in the XIL API Convenience library and how to access the basic methods of the XIL API standard in Exec blocks, refer to the ElectricalErrorSimulationPort demo project.

Related topics**Basics**

[Basics on Simulating Electrical Errors via the XIL API Convenience Library.....](#) 11

References

[PrepareMultiPinErrorWithCondition.....](#) 41
[PrepareMultiPinErrorWithDuration.....](#) 44
[PrepareSinglePinErrorWithCondition.....](#) 46
[PrepareSinglePinErrorWithDuration.....](#) 49

XIL API (Electrical Error Simulation)

Introduction

AutomationDesk provides specific blocks and data objects for electrical error simulation using the XIL API library.

Using XIL API Convenience library features in Python scripts

You can use functions and other definitions of the XIL API Convenience library in Python scripts after you imported the `audxilapiconvenience.lib` module to the current namespace.

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

[EESPort \(Data Object\).....](#) 54
 To provide an EESPort object created by an EESPortFactory instance.

[EESPortFactory \(Data Object\).....](#) 55
 To instantiate a factory object to create an EESPort for electrical error simulation.

[ErrorConfiguration \(Data Object\).....](#) 56
 To provide access to an error configuration.

[EESConfigurationReader \(Data Object\).....](#) 56
 To provide a file reader object for an error configuration file.

[EESConfigurationWriter \(Data Object\).....](#) 57
 To provide a file writer object for an error configuration file.

[ErrorSet \(Data Object\).....](#) 58
 To provide access to an error set.

[BaseErrorBuilder \(Data Object\).....](#) 59
 To provide a BaseError object.

ErrorFactory (Data Object).....	60
To instantiate a factory object to create errors.	
BaseError (Data Object).....	60
To provide access to an error.	
SpecificErrorFactory (Data Object).....	62
To instantiate a factory object to create errors assigned to the SpecificErrorFactory.	
SpecificErrorFactory2 (Data Object).....	63
To instantiate a factory object to create errors assigned to the SpecificErrorFactory2.	

EESPort (Data Object)

Graphical representation



EESPort

Purpose

To provide an EESPort object created by an EESPortFactory instance.

Description

The EESPort data object is used as a wrapper for the EESPort class or contains the EESPort instance which is created when using the EESPortFactory (refer to [EESPortFactory \(Data Object\)](#) on page 55) data object. The EESPort is the central point for downloading an error configuration and executing errors simulated on the HIL simulator.

An EESPort instance can only be used for one simulation application. If you want to load another simulation application, you must release the EESPort instance beforehand.

Access via Exec block

This data object can be accessed via script within an Exec block.

The name completion feature of the Python Editor provides the available properties and methods of the current object. For more information on the properties and methods, refer to [ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf](#).

Related topics**Basics**

[Accessing Simulation Platforms via XIL API \(AutomationDesk Accessing Simulation Platforms !\[\]\(feabb98897b440bc8695a03336a6e2df_img.jpg\)\)](#)

References

[EESPortFactory \(Data Object\)..... 55](#)

EESPortFactory (Data Object)

Graphical representation**Purpose**

To instantiate a factory object to create an EESPort for electrical error simulation.

Description

The EESPortFactory data object is used to create and configure an EESPort object.

Access via Exec block

This data object can be accessed via script within an Exec block.

The name completion feature of the Python Editor provides the available properties and methods of the current object. For more information on the properties and methods, refer to [ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf](#).

Related topics**Basics**

[Accessing Simulation Platforms via XIL API \(AutomationDesk Accessing Simulation Platforms !\[\]\(df47d6bec273bbb8b349135fff3a20f7_img.jpg\)\)](#)

HowTos

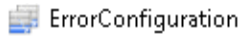
[How to Simulate Errors from an Error Configuration File..... 25](#)

References

[EESPort \(Data Object\)..... 54](#)
[ErrorConfiguration \(Data Object\)..... 56](#)

ErrorConfiguration (Data Object)

Graphical representation



Purpose

To provide access to an error configuration.

Description

The ErrorConfiguration data object is used to provide access to an error configuration. An error configuration consists of at least one error set. The properties and methods of the ErrorConfiguration data object lets you create error sets, save a created or modified error configuration and load an existing error configuration.

Access via Exec block

This data object can be accessed via script within an Exec block.

The name completion feature of the Python Editor provides the available properties and methods of the current object. For more information on the properties and methods, refer to [ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf](#).

Related topics

Basics

[Accessing Simulation Platforms via XIL API \(AutomationDesk Accessing Simulation Platforms !\[\]\(e9474ce1d70442456f8fe9c393ea149c_img.jpg\)\)](#)

HowTos

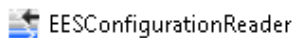
[How to Simulate Errors from an Error Configuration File..... 25](#)


References

[EESConfigurationReader \(Data Object\)..... 56](#)
[EESConfigurationWriter \(Data Object\)..... 57](#)
[EESPort \(Data Object\)..... 54](#)
[ErrorSet \(Data Object\)..... 58](#)

EESConfigurationReader (Data Object)

Graphical representation



Purpose	To provide a file reader object for an error configuration file.
Description	The EESConfigurationReader data object is used to provide a file reader object for loading an already existing error configuration file.
Access via Exec block	<p>This data object can be accessed via script within an Exec block.</p> <p>The name completion feature of the Python Editor provides the available properties and methods of the current object. For more information on the properties and methods, refer to ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf.</p>
Related topics	<p>Basics</p> <p>Accessing Simulation Platforms via XIL API (AutomationDesk Accessing Simulation Platforms )</p> <p>HowTos</p> <p>How to Simulate Errors from an Error Configuration File..... 25</p> <p>References</p> <p>EESConfigurationWriter (Data Object)..... 57 ErrorConfiguration (Data Object)..... 56</p>

EESConfigurationWriter (Data Object)

Graphical representation  EESConfigurationWriter

Purpose	To provide a file writer object for an error configuration file.
Description	The EESConfigurationWriter data object is used to provide a file writer object for saving the current error configuration to the specified error configuration file.
Access via Exec block	This data object can be accessed via script within an Exec block.

The name completion feature of the Python Editor provides the available properties and methods of the current object. For more information on the properties and methods, refer to [ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf](#).

Related topics

Basics

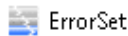
[Accessing Simulation Platforms via XIL API \(AutomationDesk Accessing Simulation Platforms !\[\]\(0f848bbd71cef6b345273b16f905912a_img.jpg\)\)](#)

References

EESConfigurationReader (Data Object).....	56
ErrorConfiguration (Data Object).....	56

ErrorSet (Data Object)

Graphical representation



ErrorSet

Purpose

To provide access to an error set.

Description

The ErrorSet data object is used to provide access to an error set. An error set contains all the errors that you want to execute subsequently when the error set is triggered. An error configuration consists of one or more error sets. The properties and methods of the ErrorSet data object lets you get information on the contained errors or add new errors to it.

Access via Exec block

This data object can be accessed via script within an Exec block.

The name completion feature of the Python Editor provides the available properties and methods of the current object. For more information on the properties and methods, refer to [ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf](#).

Related topics

Basics

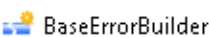
[Accessing Simulation Platforms via XIL API \(AutomationDesk Accessing Simulation Platforms !\[\]\(4729e517bc6a7cd81c8025b9646574fb_img.jpg\)\)](#)

References

[ErrorFactory \(Data Object\)..... 60](#)

BaseErrorBuilder (Data Object)

Graphical representation



Purpose

To provide a BaseError object.

Description

The BaseErrorBuilder data object is used to provide an error builder for errors of BaseError type, such as simple errors (default), dynamic errors or resistor errors.

Access via Exec block

This data object can be accessed via script within an Exec block.

The name completion feature of the Python Editor provides the available properties and methods of the current object. For more information on the properties and methods, refer to [ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf](#).

Related topics

Basics

[Accessing Simulation Platforms via XIL API \(AutomationDesk Accessing Simulation Platforms !\[\]\(84f47badaad7772cd95667a7c387a639_img.jpg\)\)](#)

References

[BaseError \(Data Object\)..... 60](#)
[SpecificErrorFactory \(Data Object\)..... 62](#)
[SpecificErrorFactory2 \(Data Object\)..... 63](#)

ErrorFactory (Data Object)

Graphical representation



ErrorFactory

Purpose

To instantiate a factory object to create errors.

Description

The ErrorFactory data object is used to create and configure errors. The errors that you can simulate depends on your EES hardware. For information on the supported error types, refer to the user documentation of your hardware.

For an overview of supported errors on dSPACE hardware, refer to [Failure Classes \(dSPACE XIL API Reference !\[\]\(aa53ad6fea213b8b2226d3077e30533a_img.jpg\)](#)).

For most of the errors, you can specify whether the electrical error is to be simulated with load or without load.

Access via Exec block

This data object can be accessed via script within an Exec block.

The name completion feature of the Python Editor provides the available properties and methods of the current object. For more information on the properties and methods, refer to [ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf](#).

Related topics

Basics

[Accessing Simulation Platforms via XIL API \(AutomationDesk Accessing Simulation Platforms !\[\]\(899d8b7697d64725bf017d3296cfcf1b_img.jpg\)](#))

References

BaseError (Data Object).....	60
BaseErrorBuilder (Data Object).....	59
SpecificErrorFactory (Data Object).....	62
SpecificErrorFactory2 (Data Object).....	63

BaseError (Data Object)

Graphical representation



BaseError

Purpose

To provide access to an error.

Description

The BaseError data object is used to get information on a specific error in the error set. An error is basically specified by its error category, error type and load.

The error category specifies the kind of disturbance, such as short circuit to ground.

Error Category	Enumeration	Value
ErrorPin2Pin	ePIN_2_PIN	0
InterruptError	eINTERRUPT	1
ErrorToGround	eSHORT_CIRCUIT_GROUND	2
ErrorToUbatt	eSHORT_CIRCUIT_UBATT	3
ErrorToPotential	eSHORT_CIRCUIT_POTENTIAL	4
ErrorMultiPin2Pin	eMULTI_PIN_2_PIN	5
ErrorMultiToGround	eMULTI_SHORT_CIRCUIT_GROUND	6
ErrorMultiToUbatt	eMULTI_SHORT_CIRCUIT_UBATT	7
ErrorMultiToPotential	eMULTI_SHORT_CIRCUIT_POTENTIAL	8
InterruptAtPositionError	eINTERRUPT_AT_POSITION	9
ErrorInterchangedPins	eINTERCHANGED	10

The error type specifies whether the behavior of the error changes, such as a loose contact that is repeated for a specified duration, at a specified frequency and duty cycle.

Error Type	Enumeration	Value
Base	eBASE	0
Simple	eSIMPLE	1
Dynamic	eDYNAMIC	2
LooseContact	eLOOSE_CONTACT	3
Resistor	eRESISTOR	4
ResistorDynamic	eRESISTOR_DYNAMIC	5
ResistorLooseContact	eRESISTOR_LOOSE_CONTACT	6

The load type specifies whether a load rejection is performed when the error occurred.

Load Type	Enumeration	Value
With load	eWITH_LOAD	0
Without load	eWITHOUT_LOAD	1

Access via Exec block

This data object can be accessed via script within an Exec block.

The name completion feature of the Python Editor provides the available properties and methods of the current object. For more information on the properties and methods, refer to [ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf](#).

Related topics**Basics**


[Accessing Simulation Platforms via XIL API \(AutomationDesk Accessing Simulation Platforms !\[\]\(eafc244b53721dd1ec133f0772f70fc7_img.jpg\)\)](#)

References

BaseErrorBuilder (Data Object).....	59
SpecificErrorFactory (Data Object).....	62
SpecificErrorFactory2 (Data Object).....	63

SpecificErrorFactory (Data Object)

Graphical representation

 **SpecificErrorFactory**

Purpose

To instantiate a factory object to create errors assigned to the SpecificErrorFactory.

Description

The SpecificErrorFactory data object is used to create most of the available errors with additional attributes, such as dynamic errors.

The following errors must be created by using a different error factory object:

- InterchangedPins errors are created by using the BaseErrorBuilder data object.
- MultiPin2Pin errors are created by using the SpecificErrorFactory2 data object.

Access via Exec block

This data object can be accessed via script within an Exec block.

The name completion feature of the Python Editor provides the available properties and methods of the current object. For more information on the properties and methods, refer to [ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf](#).

Related topics**Basics**

[Accessing Simulation Platforms via XIL API \(AutomationDesk Accessing Simulation Platforms !\[\]\(111c5272ee3f91361f0d2e3665dd6ad0_img.jpg\)\)](#)

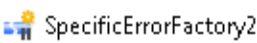
References

BaseError (Data Object).....	60
BaseErrorBuilder (Data Object).....	59

SpecificErrorFactory2 (Data Object).....	63
--	----

SpecificErrorFactory2 (Data Object)

Graphical representation



Purpose

To instantiate a factory object to create errors assigned to the SpecificErrorFactory2.

Description

The SpecificErrorFactory2 data object is used to create MultiPin2Pin errors with additional attributes, such as dynamic errors.

Access via Exec block

This data object can be accessed via script within an Exec block.

The name completion feature of the Python Editor provides the available properties and methods of the current object. For more information on the properties and methods, refer to [ASAM_AE_XIL_Generic-Simulator-Interface_BS-1-4-Programmers-Guide_V2-1-0.pdf](#).

Related topics

Basics

[Accessing Simulation Platforms via XIL API \(AutomationDesk Accessing Simulation Platforms !\[\]\(0d7ca0919e6c47bbd874bfa0189fe22e_img.jpg\)\)](#)

References

BaseError (Data Object).....	60
ErrorFactory (Data Object).....	60
SpecificErrorFactory (Data Object).....	62

Commands And Dialogs

XIL API

Introduction

AutomationDesk provides specific commands for electrical error simulation using the XIL API library.

Where to go from here

Information in this section

Edit (SinglePinError).....	64
To configure an error set that affects a single pin.	
Edit (MultiPinError).....	66
To configure an error set that affects multiple pins.	


Edit (SinglePinError)

Access

You can access this command via:

Ribbon	None
Context menu of	<p>The data objects described below that are provided by the following automation blocks:</p> <ul style="list-style-type: none"> ▪ PrepareSinglePinError ▪ PrepareSinglePinErrorWithCondition ▪ PrepareSinglePinErrorWithDuration <p>Data objects:</p> <ul style="list-style-type: none"> ▪ ErrorCategory ▪ SignalName ▪ LoadType
Shortcut key	None
Icon	None
Others	<p>Double-click the data objects described below that are provided by the following automation blocks:</p> <ul style="list-style-type: none"> ▪ PrepareSinglePinError ▪ PrepareSinglePinErrorWithCondition ▪ PrepareSinglePinErrorWithDuration <p>Data objects:</p> <ul style="list-style-type: none"> ▪ ErrorCategory



- SignalName
- LoadType

Purpose	To configure an error set that affects a single pin.
Result	<p>If the EES port that is specified in the block's EESPort data object is initialized, the SinglePinError Configuration dialog opens and lets you specify a single pin error set.</p> <p>If the EESPort data object is empty or not initialized, the Edit dialog for the related data object is opened.</p>
Description	<p>This command lets you specify an error set in the SinglePinError Configuration dialog.</p> <p>The PrepareSinglePinError<x> blocks provide the SignalName, the ErrorCategory, and the LoadType data object, to specify the error set to be prepared. It depends on the connected EES hardware, which value combinations for these data objects are valid. By using this command, the validity of the specified value combination is permanently checked.</p>
Dialog settings	<p>The SinglePinError Configuration dialog provides the following settings:</p> <p>SignalName Lets you specify the name of the signal to be disturbed. You can enter the signal name in the input field or select it from the list of provided signals.</p> <p>The list of signal names is derived from the PORTCONFIG file that was used for the initialization of the EES port. For dSPACE EES hardware, the list is based on a CSV file that is specified in the PORTCONFIG file. If the PORTCONFIG file contains a signal mapping, only the mapping name of a signal is displayed in the list. For more information, refer to Creating dSPACE EESPort Configuration Files (dSPACE XIL API Implementation Guide ).</p> <p>ErrorCategory Lets you select the category of the simulated error. The following categories are provided:</p> <ul style="list-style-type: none"> ▪ Interrupt ▪ ShortToGround ▪ ShortToUbat <p>For more information, refer to BaseError (Data Object) on page 60.</p> <p>WithLoad Lets you specify the load type to be applied for the specified signal.</p>

If the **WithLoad** checkbox is enabled, the load is not disconnected from the simulator platform when the error occurred, i.e., a load rejection is not performed.

If the checkbox is disabled, a load rejection is performed.

Validation check The validity of the currently specified error set is displayed in a status icon and a status message. The following status icons are provided:

Status Icon	Meaning
	The currently specified error set is valid.
	The currently specified error set in SignalName, ErrorCategory and LoadType is invalid. The OK button is still enabled, in case you want to specify an error set for a signal that is not defined yet.

Related topics

References

BaseError (Data Object).....	60
Edit (MultiPinError).....	66
PrepareSinglePinError.....	37
PrepareSinglePinErrorWithCondition.....	46
PrepareSinglePinErrorWithDuration.....	49

Edit (MultiPinError)

Access

You can access this command via:

Ribbon	None
Context menu of	The data objects described below that are provided by the following automation blocks: <ul style="list-style-type: none"> PrepareMultiPinError PrepareMultiPinErrorWithCondition PrepareMultiPinErrorWithDuration Data objects: <ul style="list-style-type: none"> ErrorCategory SignalNames LoadTypes
Shortcut key	None
Icon	None
Others	Double-click the data objects described below that are provided by the following automation blocks: <ul style="list-style-type: none"> PrepareMultiPinError PrepareMultiPinErrorWithCondition

- PrepareMultiPinErrorWithDuration

Data objects:

- ErrorCategory
- SignalNames
- LoadTypes

Purpose

To configure an error set that affects multiple pins.

Result

If the EES port that is specified in the block's **EESPort** data object is initialized, the **MultiPinError Configuration** dialog opens and lets you specify a multi pin error set.

If the **EESPort** data object is empty or not initialized, the **Edit** dialog for the related data object is opened.

Description

This command lets you specify the error set in the **MultiPinError Configuration** dialog.

The **PrepareMultiPinError<x>** blocks provide the **SignalNames**, the **ErrorCategory**, and the **LoadTypes** data objects, to specify the error set to be prepared. It depends on the connected EES hardware, which value combinations for these data objects are valid. By using this command, the validity of the specified value combination is permanently checked.



Dialog settings

The **MultiPinError Configuration** dialog provides the following settings:

Available SignalNames Lets you select the names of the signals to be disturbed from the list of provided signal names. Only signals that are not already selected are contained in the list.

The list of signal names is derived from the **PORTCONFIG** file that was used for the initialization of the EES port. For dSPACE EES hardware, the list is based on a CSV file that is specified in the **PORTCONFIG** file. If the **PORTCONFIG** file contains a signal mapping, only the mapping name of a signal is displayed in the list. For more information, refer to [Creating dSPACE EESPort Configuration Files \(dSPACE XIL API Implementation Guide\)](#).

Set SignalNames Displays a table with the list of signals to be disturbed.

Column	Meaning
Status Icon	<p>The following icons indicate the status of an entry in the list of signals to be disturbed:</p> <ul style="list-style-type: none"> ▪  The specified error set is valid. ▪  The specified error set is invalid.
SignalName	The name of a signal to be disturbed.

Column	Meaning
WithLoad	<p>A checkbox that lets you specify the load type to be applied for the specified signal.</p> <ul style="list-style-type: none"> ▪ If the checkbox is enabled, the load is not disconnected from the simulator platform when the error occurred, i.e., a load rejection is not performed. ▪ If the checkbox is disabled, a load rejection is performed.

> Moves the selected entries from the Available SignalNames list to the Set SignalNames list.

< Moves the selected entries from the Set SignalNames list to the Available SignalNames list.



Add Lets you expand the list of the signals to be disturbed by a signal name, that is not provided by the list of available signals. Specify the signal name in the edit field and press Add.

ErrorCategory Lets you select the category of the simulated error that is applied to the signals in the Set SignalNames list. The following categories are provided:

- Interrupt
- ShortToGround
- ShortToUbat
- Pin2Pin

For more information, refer to [BaseError \(Data Object\)](#) on page 60.

Validation check for the entire error set The validity of the currently specified error set is displayed in a status icon and a status message. The following status icons are provided:

Status Icon	Meaning
	The currently specified error set is valid.
	<p>The currently specified error set contains at least one invalid entry. To analyze the cause, find the invalid entry in the Set SignalNames list and inspect its tool tip.</p> <p>The OK button is still enabled, in case you want to specify an error set for a signal that is not defined yet.</p>

Related topics

References

Edit (SinglePinError)	64
PrepareMultiPinError	35
PrepareMultiPinErrorWithCondition	41
PrepareMultiPinErrorWithDuration	44

Automation

Basics on Automating Electrical Error Simulation





Introduction

AutomationDesk provides a [COM-based](#)  API to automate the handling of AutomationDesk.

Related information

The AutomationDesk COM API provides specific objects for configuring electrical error simulation using the XIL API library.

Testbench handling:

- [Mapping \(Object\)](#) (AutomationDesk Automation )
- [PortConfig](#) (AutomationDesk Automation )
- [Testbench](#) (AutomationDesk Automation )
- [TestbenchFactory](#) (AutomationDesk Automation )





EESPort handling:





- [EESPort](#) (AutomationDesk Automation )
- [EESPortFactory](#) (AutomationDesk Automation )

Error handling:


- [EESConfigurationReader](#) (AutomationDesk Automation )
- [EESConfigurationWriter](#) (AutomationDesk Automation )
- [ErrorConfiguration](#) (AutomationDesk Automation )
- [ErrorFactory](#) (AutomationDesk Automation )
- [ErrorSet](#) (AutomationDesk Automation )
- [SpecificErrorFactory](#) (AutomationDesk Automation )
- [SpecificError2Factory](#) (AutomationDesk Automation )

Common data objects handling:

- [Attributes](#) (AutomationDesk Automation )
- [DataType](#) (AutomationDesk Automation )
- [Symbol](#) (AutomationDesk Automation )
- [SymbolFactory](#) (AutomationDesk Automation )

- [TaskInfoFactory](#) ([AutomationDesk Automation](#) )
- [ValueInfo](#) ([AutomationDesk Automation](#) )
- [Watcher](#) ([AutomationDesk Automation](#) )
- [WatcherFactory](#) ([AutomationDesk Automation](#) )

When you work with the XIL API Convenience library, the same data objects are used.

For basic information and instructions, refer to [Basics on Automating AutomationDesk Handling](#) ([AutomationDesk Basic Practices](#) ).

Limitations

Limitations When Using the XIL API Convenience Library

Restriction on stimulating fixed-point variables

When you stimulate an application variable of a fixed-point data type via a SignalGenerator data object, the variable's scaling factor that is specified in the variable description file (TRC file) is ignored.

Reading, writing and capturing application variables of a fixed-point data type considers the scaling factor.

Glossary

Introduction	The glossary briefly explains the most important expressions and naming conventions used in the AutomationDesk documentation.
--------------	---

Where to go from here

Information in this section

Symbols.....	74
A.....	74
B.....	76
C.....	76
D.....	77
E.....	78
F.....	79
H.....	79
I.....	79
L.....	80
M.....	81
O.....	82
P.....	82
R.....	83
S.....	84
T.....	86
U.....	86
V.....	86

W.....	87
X.....	87

Symbols

@ADLX folder A file system folder with the name <CustomLibraryName>@adlx that stores information on the elements in the related [custom library](#), such as [template](#) files, attached Python modules, or packages.

In the [Library Browser](#), you always use this folder in combination with the related [library file \(ADLX\)](#).

@ADPX folder A file system folder with the name <ProjectName>@adpx that stores information on the elements in the related [project](#), such as [sequence](#) files, attached files, [results](#), and [reports](#).

In the [Project Manager](#), you always use this folder in combination with the related [project file \(ADPX\)](#).

@BLKX folder A file system folder with the name <ElementName>@blkx that stores information on the subelements in an exported [element](#), such as [sequence](#) files, attached files, Python modules, or packages.

You always use this folder in combination with the related [parent element file \(BLKX\)](#).

A

ADL file An AutomationDesk legacy library file that contains the specification of a [custom library](#) which was saved to the file system.

These files were created using AutomationDesk 6.1 or earlier. You can open and migrate them to [library XML files \(ADLX\)](#).

ADL.ZIP file An AutomationDesk legacy archive file that contains the specification of a [custom library](#) which was managed under version control.

These files were created using AutomationDesk 6.1 or earlier and must be migrated to let you continue to work under version control.

ADLX file An AutomationDesk library XML file that contains the specification of a saved or exported [custom library](#). The ADLX file is located in the same folder as the [@ADLX folder](#).

You can open, edit, save, import, and export ADLX files via the [Library Browser](#).

ADO file An AutomationDesk display options file that contains information on how a [project](#) or [library](#) is displayed when it is opened in the AutomationDesk user interface. This includes [bookmarks](#), [breakpoints](#), and the collapse state of folders and blocks.

These files are created when a project or library is saved or closed.

ADP file An AutomationDesk legacy project file that contains a [project's](#) specification, its [results](#), and its [reports](#).

These files were created using AutomationDesk 6.1 or earlier. You can open and migrate them to [project XML files \(ADPX\)](#).

ADP.ZIP file An AutomationDesk legacy archive file that contains the specification of a [project](#) which was managed under version control.

These files were created using AutomationDesk 6.1 or earlier and must be migrated to let you continue to work under version control.

ADPX file An AutomationDesk project XML file that contains the specification of a saved or exported AutomationDesk [project](#). The ADPX file is located in the same folder as the [@ADPX folder](#).

You can open, edit, save, import, and export ADPX files via the [Project Manager](#).

ALX file An AutomationDesk library legacy XML file that contains the specification of an exported [custom library](#).

These files were created using AutomationDesk 6.0 or earlier. You can import ALX files in the [Library Browser](#).

APX file An AutomationDesk project legacy XML file that contains the specification of an exported AutomationDesk [project](#).

These files were created using AutomationDesk 6.0 or earlier. You can import APX files in the [Project Manager](#).

ASAM AE XIL API An API standard for the communication between test automation tools, such as AutomationDesk, and test benches, such as dSPACE real-time hardware. The notation XIL indicates that the standard can be used for various *in-the-loop* systems, e.g., SIL, MIL, PIL, and HIL. The XIL API standard is defined by the Association for Standardisation of Automation and Measuring Systems (ASAM).

ASAM General Expression Syntax (ASAM GES) The syntax definition that is used in AutomationDesk to specify trigger conditions. It is part of the XIL API standard that is defined by the Association for Standardisation of Automation and Measuring Systems (ASAM).

Automation block A part of a [sequence](#) that implements an automation task, similar to a subroutine.

Templates for automation blocks are provided by AutomationDesk [libraries](#). Via the [Sequence Builder](#), you can arrange automation blocks to implement the control flow of your automation task.

AutomationDesk Options A dialog that lets you modify the appearance and behavior of some AutomationDesk [panes](#) and the layout of the generated [reports](#).

Automotive Simulation Model (ASM) The dSPACE product that provides open MATLAB®/Simulink® models that are relevant for the simulation of automotive engines (gasoline and diesel) and vehicle dynamics.

B

BLKX file An AutomationDesk element XML file that contains the specification of a saved or exported AutomationDesk [element](#).

You can import and export BLKX files in the [Project Manager](#), the [Sequence Hierarchy Browser](#), the [Sequence Builder](#), and the [Library Browser](#).

Block-specific data object A [data object](#) that resides in the interface of an [automation block](#). It can be used to parameterize the block or to return a resulting data object after block execution.

Most blocks provided by AutomationDesk provide a static interface. However, some blocks let you add data objects to their interfaces dynamically, for example, [Exec blocks](#).

Bookmark A label that you can attach to an [automation block](#) to use it later for quick navigation within the user interface.

Breakpoint A flag that you can set for a [sequence](#) or an [automation block](#) that pauses the execution in debug mode when the element with a set breakpoint is reached. You can manually control whether to resume the execution or to terminate it.

Built-in library The type of [library](#) that is included in AutomationDesk as a software component.

In contrast to [custom libraries](#), you cannot create your own built-in libraries and you cannot view the library's source code.

C

Capture A data object type of the [ASAM AE XIL API](#) that is used to parameterize the capturing of measurement data.

In addition to the [model access port \(MAPort\)](#) to be used and the [variables](#) to be captured, you can specify, a condition to start or to stop data capturing, for example.

CaptureResult A data object type of the [ASAM AE XIL API](#) that is used to handle the captured data. It contains the time stamps and the related measured values of the captured [variables](#).

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>

Component Object Model (COM) An interface in Microsoft Windows that allows software products of different providers to communicate and to control each other.

ControlDesk The dSPACE software product for managing, instrumenting and executing experiments for ECU development. ControlDesk also supports calibration, measurement and diagnostics access to ECUs via standardized protocols such as CCP, XCP, and ODX.

Control-flow-based testing A test strategy that is based on implementing an automation task by specifying its control flow in [sequences](#).

Custom library The type of [library](#) that you can create and include in AutomationDesk. The [elements](#) that you can add to a custom library are [templates](#) for [data objects](#), [automation blocks](#), and [sequences](#). You can use the library elements as templates by adding [library links](#) to projects or sequences.

Some predefined custom libraries are part of the AutomationDesk product. They are read-only by default.

D

Data object Objects that can store a value according to the data object's type. You can specify a data object *by value* via an editor that depends on the type or *by reference* via the [Data Object Editor](#).

Data objects can be instantiated specific to a [project](#), to a [sequence](#), or to an [automation block](#).

Templates for data objects of various types are provided via AutomationDesk [libraries](#) and can be created via the [Project Manager](#) or the [Sequence Builder](#), for example.

Data Object Editor A [pane](#) that lets you access the values and references of the data objects of the selected object.

Data Object Selector A dialog that lets you specify a [data object](#) by selecting one from the tree of available data objects.

DataContainer An element that lets you bundle [data objects](#) to structure them. DataContainers can be nested.

Debug mode A mode that lets you execute a [project](#) or a [sequence](#) successively and control the execution manually, for example, by using [breakpoints](#).

Documents folder A standard folder for user-specific documents.

%USERPROFILE%\Documents\dSPACE\[<ProductName>](#)
[<VersionNumber>](#)

dSPACE Help The component that contains all the relevant user documentation for dSPACE products. Via the F1 key or the Help button in the dSPACE software, you get context-sensitive help on the active context.

dSPACE Log A [pane](#) that displays the errors, warnings, information, and advice issued by all installed dSPACE products.

E

Edit dialog The dialog that lets you specify the value of a [data object](#). The default edit dialog depends on the data type of the data object, but you can also use a customized edit dialog.

Electrical error simulation (EES) The simulation of errors in the wiring, such as loose contacts, broken cables, or short-circuits. Electrical error simulation is performed by the EES hardware of an HIL simulator.

Electrical error simulation port (EESPort) A data object type of the [ASAM AE XIL API](#) that is used to provide access to the [electrical error simulation \(EES\)](#) hardware of an HIL simulator.

Element The representation of a resource of a [project](#) in the [Project Manager](#) or a [library](#) in the [Library Browser](#).

An element is displayed as an icon that reflects the element's type followed by the element's name.

Error configuration file A file in XML format that contains the specification of the simulated electrical errors as a series of states which are each specified via an [error set](#).

Error set A list of electrical errors that occur to the signals at the same time and that specifies the simulated state of the wiring. An empty error set specifies a state with no errors.

Exec block An [automation block](#) that is specified by the Python script to be executed.

You can edit the script via AutomationDesk's [Python Editor](#).

F

FDX file An AutomationDesk project folder legacy XML file that contains the specification of an exported AutomationDesk [project folder](#).

These files were created using AutomationDesk 6.0 or earlier. You can import FDX files in the [Project Manager](#).

H

Hyperlink A click-able reference. When you click the link, the target is opened in an appropriate component.

I

Input dialog A dialog window that demands a manual input.

Instance description The property of an instantiated [element](#) that contains a text which describes the element's purpose.

L

LabeledValue A type of data object for which you can define a dictionary of valid label-value pairs. LabeledValues can be set either by specifying a label or by specifying a value.

LFX file An AutomationDesk library folder legacy XML file that contains the specification of an exported [library folder](#).

These files were created using AutomationDesk 6.0 or earlier. You can import LFX files in the [Library Browser](#).

Library A container for [templates](#) that you can use to instantiate [data objects](#), [sequences](#), or [automation blocks](#) in your [projects](#).

Libraries are handled via the [Library Browser](#). Each library is organized as a tree and can be structured using [library folders](#).

There are [built-in libraries](#) and [custom libraries](#).

Library Browser A [pane](#) in AutomationDesk that provides access to the [elements](#) of the open [libraries](#).

Library folder An element that structures the contents of a [library](#) as a tree.

Library link A type of [element](#) that you can create in a [project](#) or [sequence](#). This type of element is linked to a [template](#) in a [library](#). Depending on the [link mode](#), the library link represents an instance of the linked library element or a reference to this library element.

Library links let you reuse a library element at multiple positions in one or multiple projects.

Link mode The way in which an instantiated object in your [project](#) can be connected to its related [template](#) in the [library](#).

The link mode determines the synchronization behavior after you modified an object's template.

The following link modes are available:

- *Dynamically linked* - A modification of the template takes immediate effect.
- *Statically linked* - A modification of the template takes effect after you manually synchronized it.

If you break the link between an instantiated object and its template, the object becomes independent from the template and cannot be linked again.

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>
```


M

Mapping (For [XIL API Testbench](#) only) An object type of the [ASAM AE XIL API](#) for a data object that contains a mapping of variable aliases to their model paths in the related [simulation application](#).

Only one Mapping data object is supported per [project](#) and it always resides at the top level of the [object hierarchy](#).

Mapping Editor (For [XIL API Framework](#) only) A component that lets you configure an XIL API Framework. This includes, for example, the mapping of aliases to model paths, which you can use in your test cases and which are required to access [variables](#) via a model access port.

The configuration is saved to a framework configuration file (XML) as well as related port configuration and mapping files.

Mapping Viewer A [pane](#) that displays the contents of the used variable mapping.

If you are working with an [XIL API Framework](#), the mapping relates to the framework configuration, which you can edit via the [Mapping Editor](#).

If you are working with the [XIL API Testbench](#), the mapping relates to the project's [Mapping](#) data object, which you can edit in the Mapping Viewer.

MAT file A file that contains measurement data in a format that allows data exchange with MATLAB.

MDF file A file that contains measurement data in a format that complies with the ASAM Common MDF standard. For version 4.1 of this standard, the file name extension is MF4.

Message dialog A dialog that demands manual confirmation for a message, error, warning, or information.

Message Viewer A [pane](#) that displays the history of all error and warning messages that occur while you are working with AutomationDesk.

MF4 file Refer to [MDF file](#).

Model access port (MAPort) A data object type of the [ASAM AE XIL API](#) that is used to provide access to the [variables](#) of a running [simulation application](#).

ModelDesk The dSPACE software product for parameterizing [ASM models](#) via graphical representations of the modeled components and controlling the related real-time simulation, offline simulation, or MATLAB®/Simulink® simulation.

MotionDesk The dSPACE software product that lets you visualize the movement of 3-D objects controlled by a running simulation application.

O

Object hierarchy The hierarchy tree that is built by all objects that are instantiated in a specific [project](#).

Offline simulation application (OSA) A simulation application that can be executed without real-time hardware on a host PC with [VEOS](#). The OSA file that implements the simulation application can be built from a Simulink model by the VEOS Player.

Operation mode A feature that is provided by some [libraries](#) and lets you decide whether to work online with the related device or to work with previously recorded data.

Operation signal The signal type of [signals](#) that are specified as an arithmetic operation (addition or multiplication) of two other signals.

Output Viewer A [pane](#) that displays all output messages generated by AutomationDesk.

P

PADL.ZIP file An AutomationDesk legacy element archive file that contains the specification of a [custom library](#), a [library folder](#), or a [template](#) which was managed under version control.

These files were created using AutomationDesk 6.1 or earlier and must be migrated to let you continue to work under version control.

PADP.ZIP file An AutomationDesk legacy element archive file that contains a [project](#), a [project folder](#), a [sequence](#), or a [result](#) which was managed under version control.

These files were created using AutomationDesk 6.1 or earlier and must be migrated to let you continue to work under version control.

Pane The section of a window that provides related controls. AutomationDesk panes are arranged in [view sets](#).

Parameter Any variable type that can be calibrated.

PCONFIG file An [ASAM AE XIL API](#) EES port configuration file that provides the hardware-dependent information for an [electrical error simulation \(EES\)](#) in XML format.

Platform A software component representing a simulator where a [simulation application](#) is computed in real-time (on dSPACE real-time hardware) or in non-real-time (on [VEOS](#)).

Platform Manager A software component that is commonly used by various dSPACE products to register and access [platforms](#) and to control the execution of [simulation applications](#) on the [platforms](#).

Project A container for all instantiated resources that implement a specific automation task.

Projects are handled via the [Project Manager](#). Each project is organized as a tree and can be structured using [project folders](#).

Project folder An element that structures the contents of a [project](#) as a tree.

Project Manager A [pane](#) in AutomationDesk that provides access to the [elements](#) of the open [projects](#).

Project-specific data object A [data object](#) that is created within a [Project](#) or a [Project folder](#) in the [Project Manager](#). It can be used to parameterize elements a lower level in the [object hierarchy](#).

Properties A [pane](#) that lets you access the properties of selected elements.

Python Editor A component that lets you edit the Python scripts for [Exec blocks](#), their [templates](#), and Python modules and packages that are integrated in AutomationDesk [libraries](#). Each of these elements can be opened in a separate Python Editor [pane](#).

R

Real-time application An application that can be executed in real time on dSPACE real-time hardware. A real-time application can be built from a Simulink model containing RTI blocks, for example.

Real-Time Testing (RTT) The dSPACE software product that provides components for creating and executing Python scripts which run on the real-time hardware in parallel to the [real-time application](#).

Record depth The attribute of an execution that specifies which project [elements](#) are to include in the execution's [result](#) depending on the element's [result levels](#).

The following record depths are provided:

- No result
- High elements only
- High and medium elements

Report A document in PDF or in HTML format that is generated from an execution's [result](#).

Result A set of data that results from the execution of a [project](#), a [project folder](#), or a [sequence](#).

From a result, you can generate a [report](#).

Result Browser A component that displays the [result](#) of the execution of a [project](#), a [project folder](#), or a [sequence](#) during the execution in form of a tree of the involved data objects and their values .

Each result that you open in the Result Browser is displayed in a separate [pane](#).

Result level The attribute of an element that specifies whether to include the element in an execution's [result](#), depending on the execution's [record depth](#).

AutomationDesk provides the None, Medium, and High result levels.

Result parameter The attributes that specify whether an [element](#) is included in an execution's [result](#). For this, AutomationDesk provides the [result level](#) and the [record depth](#) attributes.

Root element The top-level element of a tree data structure. A root element represents the entire element tree of a [project](#) in the [Project Manager](#) or a [library](#) in the [Library Browser](#), for example.

S

Segment signal The signal type of the signals that are specified as a sequence of [signal segments](#).

Sequence The implementation of an automation task as a control flow specified with [automation blocks](#).

Sequences are edited via the [Sequence Builder](#).

Sequence Builder A component that lets you graphically edit the control flow of a [sequence](#), sequence [template](#) or subsequence template. Each of these elements can be opened in a separate Sequence Builder [pane](#).

Sequence Hierarchy Browser A [pane](#) in AutomationDesk that provides access to the [elements](#) of the [sequence](#) that is currently displayed in the [Sequence Builder](#).

SequenceFrame A [template](#) that is provided by the Framework Builder [built-in library](#) and that lets you specify a predefined frame for implementing similar [sequences](#).

SFX file A sequence frame legacy XML file that contains the specification of an exported [SequenceFrame](#) object.

These files were created using AutomationDesk 6.0 or earlier. You can use the [Project Manager](#) to import SFX files for handling instantiated sequence frames and the [Library Browser](#) for handling sequence frame [templates](#).

Signal The specification or measurement of the change of a value over time.

Signals can be specified by their shape in a [signal description set](#) or as a [segment signal](#) or as an [operation signal](#).

Signal description set A container for a set of [signal](#) specifications that implement a specific [signal-based test](#).

Signal description sets are handled via the [Signal Editor](#) as a table of the contained signals.

Signal Editor A component that lets you graphically edit a [signal description set](#) as a table of its contained [signals](#).

Multiple signal description sets can be opened in separate Signal Editor [panes](#).

Signal file A file in CSV format that defines via failure classes which electrical errors can be simulated by the specific EES hardware.

Signal generator A software component, that can be configured and controlled via a [data object](#) in AutomationDesk. A signal generator can be downloaded to a [platform](#) and stimulate [variables](#) in a running [simulation application](#) in real-time.

Signal segment One member in the sequence of segments that builds a [segment signal](#). A segment is specified by its type and by its other properties. The segment type is specified at the segment's creation via the [Signal Selector](#). Its other properties can be specified via the [Signal Editor](#) or the [Properties panes](#).

Signal Selector The [pane](#) that provides elements to add [segment signals](#), [operation signals](#), and [segments](#) of various segment types to your [signal description set](#) by dragging them to the [Signal Editor](#).

Signal-based testing A test strategy that is based on implementing an automation task by using [templates](#) of the Signal-Based Testing [library](#) and specifying all involved [signals](#) in a [signal description set](#).

Simulation application The generic term for [offline simulation application \(OSA\)](#) and [real-time application](#).

SQX file An AutomationDesk sequence legacy XML file that contains the specification of an exported AutomationDesk [sequence](#).

These files were created using AutomationDesk 6.0 or earlier. You can use the [Project Manager](#) to import SQX files for handling instantiated sequences and the [Library Browser](#) for handling sequence [templates](#).

Stylesheet An XSL file that specifies the layout for the generation of a [report](#) from an execution's [result](#).

STZ file A ZIP file that contains the description of a [signal description set](#) in STI format. The STI format is defined by the [ASAM AE XIL API](#) standard. You can create and manage STZ files in AutomationDesk's [Signal Editor](#).

Subsequence An [automation block](#) that can contain other automation blocks to implement a part of a [sequence's](#) control flow, for example, a loop or a subroutine.

T

Task A thread that is executed on dSPACE real-time hardware.

The execution of tasks is triggered by timer events, I/O events, or software events.

TBX file A block template legacy XML file that contains the specification of an exported [library folder](#).

These files were created using AutomationDesk 6.0 or earlier. You can import TBX files in the [Library Browser](#).

Template The reusable pattern of a [data object](#), an [automation block](#), or a [sequence](#).

To make a template executable, you must instantiate it as an object in your [project](#).

Template description The property of a [template](#) that provides a text which describes the template's purpose.

TSX file A sequence frame legacy XML file that contains the specification of an exported TestSequence (Test Framework) object.

These files were created using AutomationDesk 6.0 or earlier. You can use the [Project Manager](#) to import TSX files for handling instantiated sequence frames and the [Library Browser](#) for handling TestSequence [templates](#).

U

User function The call of an external program that you can integrate in AutomationDesk's user interface.

V

Value Editor A component that opens a modal [Input dialog](#) to edit the selected [data object's](#) value.

The appearance of the dialog depends on the type of the selected data object.

Variable A parameter in the [simulation application](#) that can be read and written.

A parameter identified by its [variable path](#).

Variable description file The SDF file, the RTA file, or the [OSA](#) file that contains the specifications for an executable [simulation application](#).

Variable path The path to the [variable](#) in the hierarchy of the model from which the [simulation application](#) is built.

Variables pane A component that lets you edit the configuration of an [model access port \(MAPort\)](#). You can select the [platform](#) type to be accessed and specify the [variable description file](#) to be used. Then you can browse the tree of the provided model [variables](#). Each MAPort configuration can be opened in a separate Variables [pane](#).

Variant A type of [data object](#) that can reference other data objects of any type.

VEOS A dSPACE software product that can execute [offline simulation applications](#) on a HostPC independently of real time. No real-time hardware is required.

Verdict A type of [data object](#) that is used to qualify the current success status of a [sequence](#), [subsequence](#), or [automation block](#).

View set A configuration of the screen arrangement. You can create various view sets and switch between them. By default, AutomationDesk provides the preconfigured view sets Sequences, Signals, and Execution.

VirtualCOM An interface object for handling AutomationDesk's COM objects. VirtualCOM ensures a proper cleanup of deleted objects in AutomationDesk's namespace.

W

Working area The central area of AutomationDesk's user interface.

X

XIL API Framework An access layer that is defined in the [ASAM AE XIL API](#) standard.

It lets you centrally configure the access to the entire test infrastructure in XML files. This decouples test cases from the real and virtual test systems you use.

XIL API Testbench An access layer that is defined in the [ASAM AE XIL API](#) standard.

It lets you configure the access from a test to its environment, such as a simulator, by using ports. For example, the access to [variables](#) of a [simulation application](#) is configured by using a [model access port](#). This decouples test software from test hardware.

B

BaseError (Data Object)
 XIL API library 60
 BaseErrorBuilder (Data Object)
 XIL API library 59

C

Common Program Data folder 6, 77

D

DeactivateError
 XIL API Convenience library 31
 Documents folder 6, 78

E

edit
 multi pin error set 66
 single pin error set 64
 EESConfigurationReader (Data Object)
 XIL API library 56
 EESConfigurationWriter (Data Object)
 XIL API library 57
 EESPort (Data Object)
 XIL API library 54
 EESPortFactory (Data Object)
 XIL API library 55
 electrical error simulation
 basic concept 13
 Electrical error simulation
 XIL API 53
 XIL API Convenience 30
 ErrorConfiguration (Data Object)
 XIL API library 56
 ErrorFactory (Data Object)
 XIL API library 60
 ErrorSet (Data Object)
 XIL API library 58

I

InitEESPort
 XIL API Convenience library 32

L

limitations
 XIL API Convenience library 71
 Local Program Data folder 6, 80

M

multi pin error set
 edit 66
 MultiPinError Configuration dialog 66

P

PrepareMultiPinError
 XIL API Convenience library 35

PrepareMultiPinErrorWithCondition
 XIL API Convenience library 41
 PrepareMultiPinErrorWithDuration
 XIL API Convenience library 44
 PrepareSinglePinError
 XIL API Convenience library 37
 PrepareSinglePinErrorWithCondition
 XIL API Convenience library 46
 PrepareSinglePinErrorWithDuration
 XIL API Convenience library 49

R

ReleaseEESPort
 XIL API Convenience library 34

S

simulating a manually triggered electrical error 17
 simulating a multi-pin electrical error 21
 simulating a single-pin electrical error 17
 simulating a software-triggered electrical error 21
 simulating electrical errors
 XIL API Convenience library 11
 single pin error set
 edit 64
 SinglePinError Configuration dialog 64
 SpecificErrorFactory (Data Object)
 XIL API library 62
 SpecificErrorFactory2 (Data Object)
 XIL API library 63

T

Trigger
 XIL API Convenience library 39

W

WaitForTrigger
 XIL API Convenience library 51

X

XIL API
 electrical error simulation 53
 XIL API Convenience
 electrical error simulation 30
 MultiPinError Configuration dialog 66
 SinglePinError Configuration dialog 64
 XIL API Convenience library
 basic concept 12
 building a basic sequence for simulating electrical errors 16
 DeactivateError 31
 demo projects 16
 InitEESPort 32
 library overview for electrical error simulation 15
 PrepareMultiPinError 35
 PrepareMultiPinErrorWithCondition 41

PrepareMultiPinErrorWithDuration 44
 PrepareSinglePinError 37
 PrepareSinglePinErrorWithCondition 46
 PrepareSinglePinErrorWithDuration 49
 ReleaseEESPort 34
 simulating a multi-pin electrical error 21
 simulating a simple electrical error set 17, 25
 simulating a software-triggered electrical error 21
 simulating electrical errors 11
 Trigger 39
 WaitForTrigger 51
 XIL API library
 BaseError (Data Object) 60
 BaseErrorBuilder (Data Object) 59
 EESConfigurationReader (Data Object) 56
 EESConfigurationWriter (Data Object) 57
 EESPort (Data Object) 54
 EESPortFactory (Data Object) 55
 ErrorConfiguration (Data Object) 56
 ErrorFactory (Data Object) 60
 ErrorSet (Data Object) 58
 SpecificErrorFactory (Data Object) 62
 SpecificErrorFactory2 (Data Object) 63
 XIL API library commands 64

