

dSPACE Release

New Features and Migration

Release 2018-B – November 2018

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/support> 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.

© 2000 - 2018 by:
dSPACE GmbH
Rathenaustraße 26
33102 Paderborn
Germany

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

ConfigurationDesk, ControlDesk, MicroAutoBox, MicroLabBox, SCALEXIO, 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	13
Conventions Used in dSPACE User Documentation.....	13
Accessing dSPACE Help and PDF Files.....	14
Overview of dSPACE Release 2018-B	17
General Enhancements and Changes.....	17
Discontinuations.....	18
Product Version Overview.....	19
New Product Key Features.....	22
Aspects of Migrating from Previous Releases	31
Migrating to dSPACE Release 2018-B.....	31
Migrating Python Scripts from Python 2.7 to Python 3.6	33
Basic Information on Python 3.6.....	34
Main Changes in Python 3.6.....	34
Main Changes in Handling Python 3.6 with dSPACE Software.....	35
General Information on Using Python Installations.....	36
Location of Python 3.6 DLL Files.....	37
General Information on Migrating.....	38
Migration via the 2to3 Tool.....	38
Technical Changes.....	39
Examples of Migration Issues.....	42
Related Developer Documents.....	44
Product-Specific Information on Migrating.....	45
AutomationDesk.....	45
Bus Manager (Stand-Alone).....	46
ConfigurationDesk - Implementation Version.....	46
ControlDesk.....	47
ModelDesk.....	47
Platform API Package.....	47
Real-Time Testing.....	48
SYNECT.....	48
SystemDesk.....	50

Automotive Simulation Models (ASM)	51
All ASM Products.....	52
Changes in All ASM Demo Models.....	52
ASM Base InCylinder.....	53
Migrating to ASM Base InCylinder Blockset 2.4.....	53
ASM Brake Hydraulics.....	54
New Features of ASM Brake Hydraulics Blockset 2.1.....	54
Changes in the ASM Brake Hydraulics Demo Model.....	54
ASM Diesel Engine.....	55
New Features of ASM Diesel Engine Blockset 2.6.2.....	55
Migrating to ASM Diesel Engine Blockset 2.6.2.....	55
ASM Diesel Exhaust.....	57
Migrating to ASM Diesel Exhaust Blockset 2.1.7.....	57
ASM Diesel InCylinder.....	58
Changes in the ASM Diesel InCylinder Demo Model.....	58
Migrating to ASM Diesel InCylinder Blockset 2.6.....	58
ASM Drivetrain Basic.....	59
New Features of ASM Drivetrain Basic Blockset 5.2.....	59
Migrating to ASM Drivetrain Basic Blockset 5.2.....	59
ASM Electric Components.....	60
New Features of ASM Electric Components Blockset 3.7.....	60
Migrating to ASM Electric Components Blockset 3.7.....	60
ASM Environment.....	63
New Features of ASM Environment Blockset 4.9.....	63
Migrating to ASM Environment Blockset 4.9.....	64
ASM Gasoline Engine Basic.....	65
Migrating to ASM Gasoline Engine Basic Blockset 2.2.2.....	65
ASM Gasoline Engine.....	66
Migrating to ASM Gasoline Engine Blockset 4.0.2.....	66
ASM Gasoline InCylinder.....	67
New Features of ASM Gasoline InCylinder Blockset 2.6.....	67
Changes in the ASM Gasoline InCylinder Demo Model.....	67
Migrating to ASM Gasoline InCylinder Blockset 2.6.....	67
ASM KnC.....	69
New Features of ASM KnC 8.0.0.....	69
Migrating to ASM KnC 8.0.0.....	69

ASM Traffic.....	70
New Features of ASM Traffic Blockset 3.9.....	70
Changes in the ASM Traffic Demo Model.....	71
ASM Truck.....	72
Changes in the ASM Truck Demo Model.....	72
Migrating to ASM Truck Blockset 3.1.....	72
ASM Turbocharger.....	74
Migrating to ASM Turbocharger Blockset 3.2.3.....	74
ASM Utils.....	75
New Features of ASM Utils 4.1.....	75
ASM Vehicle Dynamics.....	76
New Features of ASM Vehicle Dynamics Blockset 4.1.....	76
Migrating to ASM Vehicle Dynamics Blockset 4.1.....	76
AutomationDesk	79
New Features of AutomationDesk 6.0.....	79
Migrating to AutomationDesk 6.0.....	82
Bus Manager (Stand-Alone)	85
Features of the Bus Manager (Stand-Alone) 6.2.....	85
Migrating to Bus Manager (Stand-Alone) 6.2.....	88
ConfigurationDesk	89
ConfigurationDesk - Implementation Version.....	90
New Features of ConfigurationDesk 6.2 (Implementation Version).....	90
Migrating to ConfigurationDesk 6.2.....	95
ControlDesk	97
New Features of ControlDesk 6.4.....	98
New Features of Platform Management and Platforms/Devices (ControlDesk 6.4).....	98
New Instrument Features (ControlDesk 6.4).....	100
New Calibration and Data Set Management Features (ControlDesk 6.4).....	102
New User Interface Handling Features (ControlDesk 6.4).....	102
New Automation Features (ControlDesk 6.4).....	103
New Bus Navigator Features (ControlDesk 6.4).....	104

New Electrical Error Simulation Features (ControlDesk 6.4).....	105
New Signal Editor Features (ControlDesk 6.4).....	105
Migrating to ControlDesk 6.4.....	108
Discontinuations in ControlDesk.....	108
Migrating to ControlDesk 6.4.....	109
DCI Configuration Tool	115
New Features of the DCI Configuration Tool 3.10.....	115
dSPACE FlexRay Configuration Package	117
New Features of dSPACE FlexRay Configuration Package 4.2.....	117
dSPACE Python Extensions	119
New Features of dSPACE Python Extensions 3.0.....	119
dSPACE XIL API .NET	121
New Features of dSPACE XIL API .NET 2018-B.....	121
Migrating to dSPACE XIL API .NET 2018-B.....	123
ECU Interface Manager	127
New Features of ECU Interface Manager 2.4.....	127
Compatibility of ECU Interface Manager 2.4.....	128
Migrating to ECU Interface Manager 2.4.....	128
Firmware Manager	131
New Features of Firmware Manager 2.6.....	131
Model Compare	133
New Features of Model Compare 2.9.....	133
Migration to Model Compare 2.9.....	135
ModelDesk	137
New Features of ModelDesk 5.0.....	137
Migration to ModelDesk 5.0.....	139

Model Interface Package for Simulink	141
New Features of the Model Interface Package for Simulink 4.0.....	141
Migration Aspects of the Model Interface Package for Simulink.....	145
MotionDesk	149
New Features of MotionDesk 4.3.....	149
Migrating to MotionDesk 4.3.....	150
Real-Time Testing	151
New Features of Real-Time Testing 4.0.....	151
Migrating to Real-Time Testing 4.0.....	152
RTI/RTI-MP and RTLib	153
New Features of RTI/RTI-MP and RTLib.....	153
Migration Aspects of RTI/RTI-MP and RTLib.....	154
RTI Bypass Blockset	155
Migrating to RTI Bypass Blockset 3.11.....	155
RTI CAN MultiMessage Blockset	157
New Features of the RTI CAN MultiMessage Blockset 5.1.....	157
Migrating to RTI CAN MultiMessage Blockset 5.1.....	159
Distribution of the Bus Manager to RTI CAN MultiMessage Blockset Customers.....	159
RTI FPGA Programming Blockset	163
New Features of the RTI FPGA Programming Blockset 3.6.....	163
Migrating to RTI FPGA Programming Blockset 3.6.....	165
RTI LIN MultiMessage Blockset	167
New Features of the RTI LIN MultiMessage Blockset 3.1.....	167
Migrating to RTI LIN MultiMessage Blockset 3.1.....	168
Distribution of the Bus Manager to RTI LIN MultiMessage Blockset Customers.....	168

SCALEXIO Firmware	171
New Features of the SCALEXIO Firmware 4.3.....	171
 Sensor Simulation	 175
New Product: Sensor Simulation.....	175
 SYNECT	 179
New Features of SYNECT 2.6.....	180
New General Features of SYNECT.....	180
New License Type.....	182
New Features of Test Management.....	184
New Features of Model Management and Signal & Parameter Management.....	187
New Features of Workflow Management.....	190
Migrating to SYNECT 2.6.....	191
Migrating Databases.....	191
Migrating Client API Scripts.....	192
Migrating Server Scripts of SYNECT 2.5.....	193
Migration Scenarios Related to Data Model Changes.....	195
Data Model Changes From SYNECT 2.5 to SYNECT 2.6.....	195
 SystemDesk	 199
New Features of SystemDesk 5.2.....	200
New General Features.....	200
Configuring ECUs.....	200
Managing V-ECUs.....	201
Creating Adaptive Applications.....	203
Migrating to SystemDesk 5.2.....	205
Migrating to SystemDesk 5.2.....	205
 TargetLink	 207
New Features of TargetLink 4.4 and TargetLink Data Dictionary 4.4.....	208
Tool Chain Support.....	208
Executing TargetLink Code on dSPACE Real-Time Hardware.....	208
Modeling in Simulink or Stateflow.....	209
Support of Arrays of Struct.....	209

Bus Support of the Constant Block.....	209
Other MATLAB/Simulink/Stateflow Features.....	210
TargetLink Module for MATLAB® Code.....	212
Code Generation Core Functionality.....	212
New Value Copy Access Functions.....	212
AUTOSAR.....	213
Supported AUTOSAR Releases.....	213
Support of Provide-Require Ports for Sender-Receiver Communication.....	214
Support of Data Store Blocks for Sender-Receiver Communication.....	214
Support of Explicit NvData Communication.....	215
Support of Rte_IWrite and Rte_IWriteRef in Sender-Receiver and NvData Communication.....	215
Exporting Splittable Elements.....	215
Target Simulation (PIL).....	216
Support for Virtual Evaluation Boards.....	216
Changes in the Target Simulation Modules.....	216
Data Dictionary and Data Management.....	217
Support of Show Masked Content and Show Library Content in the Property Manager.....	217
New Import and Export of View Sets in the Property Manager.....	218
Support for User Mode per Workspace in the Data Dictionary Manager.....	218
Code Generator Options.....	218
New Code Generator Options.....	218
API Functions and Hook Scripts.....	219
New API Functions.....	219
New Hook Scripts.....	220
Improved Hook Scripts.....	220
Other.....	220
General Enhancements and Changes.....	220
TargetLink Demos.....	221
Synchronizing Simulink and TargetLink.....	222
Migrating to TargetLink 4.4 and TargetLink Data Dictionary 4.4.....	223
General Migration Information.....	223
Upgrading Models, Libraries, and Data Dictionaries.....	223
Basics on Migrating Between TargetLink Versions.....	223
How to Upgrade a Data Dictionary with Included DD Files.....	226

How to Manually Upgrade Libraries and Models via the API.....	228
Migrating Data Dictionaries to CodeDecorationSets.....	229
Migrating from TargetLink 4.3 to 4.4.....	232
Code Generator Options.....	232
Migration Aspects Regarding Code Generator Options.....	232
API Functions and Hook Scripts.....	235
Changes in API Functions.....	235
Changes in Hook Scripts.....	235
Messages.....	235
Message changes.....	235
Other Migration Aspects.....	236
Various Migration Aspects.....	236
Code Changes.....	236
Access Functions.....	237
AUTOSAR.....	239
Block-Specific Code Changes.....	240
Efficiency.....	244
Inheritance of Struct Data Types.....	245
Mixed Operations (Floating-Point and Fixed-Point Types).....	247
Stateflow.....	248
Other.....	249
Discontinuations.....	252
Discontinued TargetLink Features	253
Obsolete API Functions.....	253
Obsolete Limitations.....	253
Changes in Future TargetLink Versions.....	255
Features to Be Discontinued.....	255
API Functions to Be Discontinued.....	256
Deprecated Code Generator Options.....	257
VEOS	259
New Features of VEOS 4.3.....	259
Compatibility of VEOS 4.3.....	261
Migrating to VEOS 4.3.....	263
Discontinuations in VEOS.....	264

Compatibility Information	265
Supported MATLAB Releases.....	265
Operating System.....	266
Run-Time Compatibility of dSPACE Software.....	268
Limitations for Using Windows Features.....	269
 Index	 273

About This Document

Contents

This document informs you about the new features of all the dSPACE software products in Release 2018-B. It also gives you an overview of software products with no or minor changes. There are instructions on migrating from earlier dSPACE releases, especially from earlier product versions, if required.

Where to go from here






Information in this section




Conventions Used in dSPACE User Documentation	13
Accessing dSPACE Help and PDF Files	14

Conventions Used in dSPACE User Documentation

Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
 DANGER	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
 WARNING	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
 CAUTION	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
 NOTICE	Indicates a hazard that, if not avoided, could result in property damage.
 Note	Indicates important information that you should take into account to avoid malfunctions.

Symbol	Description
	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\`

or

`%PROGRAMDATA%\dSPACE\`

Documents folder A standard folder for user-specific documents.

`%USERPROFILE%\My Documents\dSPACE\`

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\`

Accessing dSPACE Help and PDF Files

Introduction

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

Online help

There are various ways to open dSPACE Help.

Note

Not all the ways to open dSPACE Help are available for all dSPACE software products.


Opening from Windows You can open dSPACE Help on its home page:

- Via Windows Start Menu

Opening from dSPACE software with menu bar You can open dSPACE Help on a product's start page:

- Via the menu bar in a dSPACE product

Opening from dSPACE software with ribbons If you use dSPACE software with ribbons, you can open dSPACE Help:


- Via the Start page in dSPACE software
- Via the Backstage view in dSPACE software (leftmost ribbon tab)
- Via the  button

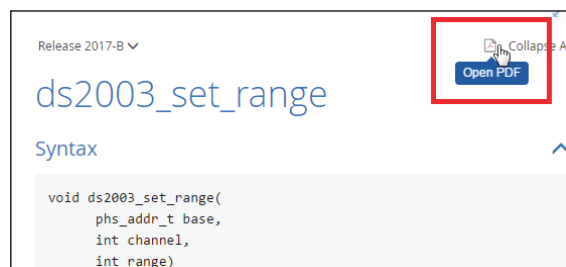
Opening context-sensitive help dSPACE Help provides context-sensitive help. You can open help on the active context in dSPACE software:

- Via F1
- Via the Help button


PDF files

You can open the PDF files as follows:

Opening from a topic in dSPACE Help You can access the PDF file with the current topic via the  button at the topic's top right. The following illustration shows an example:



The PDF document opens on its first page.

Opening from dSPACE software with ribbons If your dSPACE software has a user interface with ribbons, you can open a folder that contains the user documentation in PDF format via the  button in the Backstage view (leftmost ribbon tab).

Overview of dSPACE Release 2018-B

Introduction Gives you an overview of the new key features in Release 2018-B and information about unchanged products.

Where to go from here

Information in this section

General Enhancements and Changes	17
Discontinuations	18
Product Version Overview	19
New Product Key Features	22

General Enhancements and Changes

Introduction The following new features and changes concern several dSPACE products.

Python distribution The support of Python 2.7 is discontinued with dSPACE Release 2018-B and is replaced by Python 3.6. For more information, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33.

RCP and HIL software: C/C++ compilers for building MATLAB MEX files RCP and HIL software (such as RTI CAN MultiMessage Blockset, RTI LIN MultiMessage Blockset, or Automotive Simulation Models) now supports the following C/C++ compilers for building MATLAB MEX files:

- MinGW (GNU Compiler Collection (GCC 4.9.2)): In combination with MATLAB R2017a.

- MinGW (GNU Compiler Collection (GCC 5.3.0)): In combination with MATLAB R2017b and R2018a.
- MinGW (GNU Compiler Collection (GCC 6.3.0)): MATLAB R2018b.
- Microsoft Visual Studio 2015 Professional.

Note

The Microsoft Windows SDK 7.1 C/C++ compiler is no longer supported.

Enhanced API of dSPACE Installation Manager

As of dSPACE Release 2018-B, the API of dSPACE Installation Manager provides access to the following features:

- Collecting information about the dSPACE software environment on your PC and the problem you may have and saves it to a ZIP file. This you can send to the dSPACE Support for diagnostic purposes.

Printed user documentation

The printed user documentation is not delivered automatically. You can decide which of the available printed documents you would like to have. To order printed documentation, refer to <http://www.dspace.com/go/requestreleasematerial>.

Note

If you do not order printed documentation, use dSPACE Help or PDF files to learn about new features, enhancements, and the safety precautions regarding your products.

Discontinuations

Introduction

The following discontinuations for software and hardware are relevant to the current Release or are planned for future Releases.

For further end-of-life announcements, refer to <http://www.dspace.com/go/discontinuation>.

Discontinuation of dSPACE hardware

DS1103 PPC Controller Board This product was discontinued in December 2016. The software support of the DS1103 will be discontinued as of Release 2019-A.

For new projects, we recommend that you use the successor, dSPACE MicroLabBox.

MicroAutoBox I/O boards The following MicroAutoBox II variants were discontinued in December 2015:

- MicroAutoBox II 1401/1501
- MicroAutoBox II 1401/1504
- MicroAutoBox II 1401/1505/1507

The software support of these MicroAutoBox variants will be discontinued as of Release 2019-A.

For new projects, we recommend that you use the successor variants of MicroAutoBox II with the I/O boards DS1511, DS1513 and DS1514. The MicroAutoBox II variant 1401/1507 will still remain available.

Software support discontinuation

Python 2.7 The support of Python 2.7 is discontinued with dSPACE Release 2018-B. Python 3.6 is now supported.

For information on changes and migration aspects of Python scripts in dSPACE products, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33.

Product Version Overview

Product versions

The following table is an extract from product version histories showing the product versions of the current Release and of the three preceding Releases. If a product has new features, there is a link to the brief description in this document.

Product	dSPACE Release			
	2017-A	2017-B	2018-A	2018-B
AutomationDesk	5.4	5.5	5.6	6.0 Refer to AutomationDesk on page 79.
Automotive Simulation Models	8.4	9.0	9.1	9.2 Refer to Automotive Simulation Models (ASM) on page 51.
Bus Manager (stand-alone)	5.7	6.0	6.1	6.2 Refer to Bus Manager (Stand-Alone) on page 85.
ConfigurationDesk	5.7	6.0	6.1	6.2 Refer to ConfigurationDesk on page 89.
Container Manager	4.5	5.0	5.0	5.1
ControlDesk	6.1	6.2	6.3	6.4 Refer to ControlDesk on page 97.
DCI Configuration Tool	3.7.1	3.8	3.9	3.10 Refer to DCI Configuration Tool on page 115.
dSPACE CAN API Package	3.0.1	3.0.2	3.0.3	4.0.1

Product	dSPACE Release			
	2017-A	2017-B	2018-A	2018-B
dSPACE ECU Flash Programming Tool	2.3.2	2.4	2.5	2.5
dSPACE FlexRay Configuration Package	3.9	4.0	4.1	4.2 Refer to dSPACE FlexRay Configuration Package on page 117.
dSPACE Python Extensions	2.3	2.4	2.5	3.0 Refer to dSPACE Python Extensions on page 119.
dSPACE XIL API .NET	2017-A	2017-B	2018-A	2018-B Refer to dSPACE XIL API .NET on page 121.
ECU Interface Manager	2.1	2.2	2.3	2.4 Refer to ECU Interface Manager on page 127.
Firmware Manager	2.3	2.4	2.5	2.6 Refer to Firmware Manager on page 131.
Model Compare	2.7	2.8	2.8	2.9 Refer to Model Compare on page 133.
ModelDesk	4.5	4.6	4.7	5.0 Refer to ModelDesk on page 137.
Model Interface Package for Simulink	3.4	3.5	3.6	4.0 Refer to Model Interface Package for Simulink on page 141.
MotionDesk	4.0	4.1	4.2	4.3 Refer to MotionDesk on page 149.
MotionDesk Blockset	2.5.1	2.5.2	2.5.3	2.5.4 Refer to MotionDesk on page 149.
Real-Time Testing	3.2	3.3	3.4	4.0 Refer to Real-Time Testing on page 151.
RTI ¹⁾	7.8	7.9	7.10	7.11 Refer to RTI/RTI-MP and RTLib on page 153.
RTI-MP ²⁾	7.8	7.9	7.10	7.11 Refer to RTI/RTI-MP and RTLib on page 153.
RTI Bypass Blockset	3.8	3.9	3.10	3.11 Refer to RTI Bypass Blockset on page 155.
RTI CAN Blockset	3.4.4	3.4.5	3.4.6	3.4.7
RTI CAN MultiMessage Blockset	4.5	4.6	5.0	5.1 Refer to RTI CAN MultiMessage Blockset on page 157.
RTI Electric Motor Control Blockset	1.4	1.4.1	1.4.1	1.4.1
RTI Ethernet Blockset	1.2.2	1.2.3	1.2.3	1.2.3
RTI Ethernet (UDP) Blockset	1.4.2	1.4.3	1.4.3	1.4.3

Product	dSPACE Release			
	2017-A	2017-B	2018-A	2018-B
RTI FPGA Programming Blockset	3.3	3.4	3.5	3.6 Refer to RTI FPGA Programming Blockset on page 163.
RTI LIN MultiMessage Blockset	2.8	2.9	3.0	3.1 Refer to RTI LIN MultiMessage Blockset on page 167.
RTI RapidPro Control Unit Blockset	2.2.2	2.2.3	2.2.3	2.2.3
RTI USB Flight Recorder Blockset	1.2.1	1.2.2	1.2.2	1.2.2
RTI Watchdog Blockset	2.1	2.1.1	2.1.1	2.1.1
Sensor Simulation	-	-	-	1.0 Refer to New Product: Sensor Simulation on page 175.
SCALEXIO firmware	4.0	4.1	4.2	4.3 Refer to SCALEXIO Firmware on page 171.
SYNECT	2.3	2.4	2.5	2.6 Refer to SYNECT on page 179.
SystemDesk	4.8	5.0	5.1	5.2 Refer to SystemDesk on page 199.
TargetLink	4.2	4.3	4.3	4.4 Refer to TargetLink on page 207.
Variable Editor ³⁾	2.3	2.4	2.4	2.4
VEOS	4.0	4.1	4.2	4.3 Refer to VEOS on page 259.

¹⁾ Including the standard I/O blocksets.

²⁾ Including the RTI Gigalink Blockset.

³⁾ The Variable Editor is no longer part of the dSPACE Release DVD. It is available at <https://www.dspace.com/go/requestreleasedownload>.

If you have not performed regular updates, refer to the *New Features and Migration* documents for the dSPACE Releases listed above for information about the new features and necessary migration steps.

New Product Key Features

Introduction

This is an overview of each product's new key features. For more information, refer to the product-specific sections.

Where to go from here

Information in this topic

AutomationDesk	22
Bus Manager (stand-alone)	23
ConfigurationDesk (Implementation Version)	23
ControlDesk	23
dSPACE FlexRay Configuration Package	25
dSPACE XIL API	25
ECU Interface Manager	25
Firmware Manager	26
Model Compare	26
Model Interface Package for Simulink	26
ModelDesk	26
MotionDesk	26
Python Extensions	26
Real-Time Testing	27
RTI, RTI-MP, and RTLib	27
RTI CAN MultiMessage Blockset	27
RTI FPGA Programming Blockset	27
RTI LIN MultiMessage Blockset	27
SCALEXIO firmware	28
Sensor Simulation	28
SYNECT	28
SystemDesk	28
TargetLink	28
VEOS	29

AutomationDesk

The new key features of AutomationDesk are:

- Python 3.6 support
For information on the differences between Python 2.7, which was previously supported, and Python 3.6, and for general migration information, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33.
- Support of ASAM XIL API 2.1.0, which results in enhancements to the XIL API and XIL API Convenience libraries, such as using real-time scripts and capturing events.
- Python interpreter as a separate pane.
- Configurable namespace for Exec and ExecFile blocks.
- Enhancements to the Signal Editor, such as parameterizing values via symbolic names.
- Enhancements to the COM API to get access to a value list of a data object. The Platform Management automation is now included in AutomationDesk's COM API.

For more information on the new features, refer to [New Features of AutomationDesk 6.0](#) on page 79.

Bus Manager (stand-alone)

The new key features of the Bus Manager (stand-alone) are:

- Generating bus simulation containers without a mapped behavior model
- Enhanced port interface of bus simulation containers
- Support of secure onboard communication (already as of Bus Manager 6.1p1)
- Support of user code (already as of Bus Manager 6.1p1)
- New and enhanced bus configuration features (already as of Bus Manager 6.1p1)
- Adding ISignal IPDUs and ISignals to communication matrices (already as of Bus Manager 6.1p1)

For more information, refer to [Features of the Bus Manager \(Stand-Alone\) 6.2](#) on page 85.

ConfigurationDesk (Implementation Version)

The new key features of ConfigurationDesk are:

- Support of new Ethernet boards.
- Support of the new Virtual Ethernet Setup function block type.
- Support of SIC files generated with TargetLink
- Various enhancements of the Bus Manager for configuring bus communication for simulation, inspection, and manipulation purposes.

For more information, refer to [ConfigurationDesk - Implementation Version](#) on page 90.

ControlDesk

The new key features of ControlDesk 6.4 are:

General enhancements

- Python 3.6 support

For information on the differences between the previously supported Python 2.7 and Python 3.6 and for general migration information, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33.

For information on ControlDesk-specific migration aspects, refer to [Migrating to ControlDesk 6.4](#) on page 109.

Platform/device enhancements

- New GNSS device providing positioning data from a GNSS receiver
- SCALEXIO: Support of new Ethernet boards
- SCALEXIO: Support of LabBox (8-slot) and AutoBox (8-slot)
- SCALEXIO: Loading an application to the flash memory of SCALEXIO Real-Time PCs
- SCALEXIO: Displaying inter-FPGA connections
- VEOS: Support of VEOS Ethernet channels for bus monitoring and logging
- VEOS: Specifying the synchronization behavior for simulation time options

- XIL API MAPort: Support of XIL API version 2.1.0
- Bus monitoring devices: Support for AUTOSAR 4.3.1 and FIBEX 4.1.2
This feature was already provided by ControlDesk 6.3p1 as part of the dSPACE Release 2018-A AUTOSAR 4.3 Bus Feature Package.

For more information on the new features, refer to [New Features of Platform Management and Platforms/Devices \(ControlDesk 6.4\)](#) on page 98.

Instrument enhancements

- XY Plotter: Visualization of measurement arrays
- Time Plotter, Index Plotter: Undoing/redoing zoom and move actions

For more information on the new features, refer to [New Instrument Features \(ControlDesk 6.4\)](#) on page 100.

User interface handling enhancements

- Displaying license information

For more information on the new feature, refer to [New User Interface Handling Features \(ControlDesk 6.4\)](#) on page 102.

Calibration and data set management enhancements

- Support of memory segments and page switching for V-ECUs generated as of VEOS 4.3
- Creating a new application image based on a data set for SCALEXIO systems

For more information on the new features, refer to [New Calibration and Data Set Management Features \(ControlDesk 6.4\)](#) on page 102.

Bus Navigator enhancements

- Support of VEOS Ethernet channels for bus monitoring and logging
- Support for partly enabling bus configurations
This feature was already provided by ControlDesk 6.3p1 as part of the dSPACE Release 2018-A AUTOSAR 4.3 Bus Feature Package.
- Support for PDU user code
This feature was already provided by ControlDesk 6.3p1 as part of the dSPACE Release 2018-A AUTOSAR 4.3 Bus Feature Package.
- Support for frame length manipulation
This feature was already provided by ControlDesk 6.3p1 as part of the dSPACE Release 2018-A AUTOSAR 4.3 Bus Feature Package.
- Support of secured IPDUs
This feature was already provided by ControlDesk 6.3p1 as part of the dSPACE Release 2018-A AUTOSAR 4.3 Bus Feature Package.

For more information on the new features, refer to [New Bus Navigator Features \(ControlDesk 6.4\)](#) on page 104.

Electrical error simulation (failure simulation) enhancements

- Support of XIL API version 2.1.0

For more information on the new features, refer to [New Electrical Error Simulation Features \(ControlDesk 6.4\)](#) on page 105.

Signal Editor enhancements

- Exporting signal description sets and signal generators to STZ files according to XIL API version 2.1.0
- Globally parameterizing constant values of a signal description set

For more information on the new features, refer to [New Signal Editor Features \(ControlDesk 6.4\)](#) on page 105.

Automation enhancements

- Selecting and focusing variables and variable groups in the Variable Browser
- Checking whether reloading the variable description is required

For more information on the new features, refer to [New Automation Features \(ControlDesk 6.4\)](#) on page 103.

dSPACE FlexRay Configuration Package

The new key features of the dSPACE FlexRay Configuration Package are (already available with FlexRay Configuration Package 4.1p1):

- Support of container and contained IPDUs
- Support of secure onboard communication
- Support of global time synchronization
- Support of AUTOSAR E2E protection profiles 05 and 06

The new key features of the dSPACE FlexRay Configuration Tool are (already available with FlexRay Configuration Tool 4.1p1):

- Support of AUTOSAR System Template 4.3.0 and 4.3.1
- Support of FIBEX 4.1.2

For more information on the new features, refer to [New Features of dSPACE FlexRay Configuration Package 4.2](#) on page 117.

dSPACE XIL API

dSPACE XIL API previously supported ASAM XIL API version 2.0.1. With dSPACE Release 2018-B, it supports ASAM XIL API version 2.1.0.

The new key feature of dSPACE XIL API is:

- XIL API MAPort supports some new features introduced with ASAM XIL API 2.1.0, such as the support of real-time scripts or the capturing of client events.

For more information on the new features, refer to [New Features of dSPACE XIL API .NET 2018-B](#) on page 121.

ECU Interface Manager

The new key features of the ECU Interface Manager are:

- Improved variable-based ECU-synchronous data access
- Support of static DAQ lists

For more information on the new features, refer to [New Features of ECU Interface Manager 2.4](#) on page 127.

Firmware Manager

The new key feature of the Firmware Manager is:

- Support of new SCALEXIO boards.

For more information on the new feature, refer to [New Features of Firmware Manager 2.6](#) on page 131.

Model Compare

The new key features of Model Compare are:

- New Comfort Copy commands to resolve conflicts automatically
- Marking remaining differences as approved
- Creating new Model Compare sessions straight from the Simulink/Stateflow Editor
- Manual alignment of corresponding model elements

For more information on the new features, refer to [New Features of Model Compare 2.9](#) on page 133.

Model Interface Package for Simulink

The new key feature of Model Interface Package for Simulink is:

- New command for the creation of model port block periphery.
-

ModelDesk

The new key features of ModelDesk are:

- New test environment that is integrated in ModelDesk: ModelDesk Testing
- Road creation:
 - Height profile of junctions
 - Improved import of complex commercial OpenDRIVE network files
- Parameterization:
 - Multi-instance feature for custom blocks
 - Removing unused parameter sets

For more information on the new features, refer to [New Features of ModelDesk 5.0](#) on page 137.

MotionDesk

The new key features of MotionDesk are:

- Control and configuration of sensor simulation. Refer to [New Product: Sensor Simulation](#) on page 175.
- Data acquisition from models using the Model and Sensor Interface blockset.

For more information on the new features, refer to [New Features of MotionDesk 4.3](#) on page 149.

Python Extensions

Python Extensions 3.0 supports Python 3.6. You have to migrate your custom scripts manually. For more information, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33.

Real-Time Testing

The new key features of Real-Time Testing are:

- Real-Time Testing supports Python 3.6 on the host PC. The real-time part remains on Python 2.7.11. Therefore, the RTT sequences must not be migrated.
- Real-Time Testing supports ASAM XIL API 2.1.
- Accessing variables of remote VPUs on a VEOS platform.
- The rttlib.dsethernetapilib module supports V-ECUs executed on a VEOS platform

For more information on the new features, refer to [New Features of Real-Time Testing 4.0](#) on page 151.

RTI, RTI-MP, and RTLib

The new key features of RTI, RTI-MP, and RTLib are:

- Support of MATLAB R2018b
- Hook function for RTI-MP to customize the model separation and the update of submodels.

For more information on the new feature, refer to [New Features of RTI/RTI-MP and RTLib](#) on page 153.

RTI CAN MultiMessage Blockset

The new key features of the RTI CAN MultiMessage Blockset are (already available with RTI CAN MultiMessage Blockset 5.0p1):

- Support of AUTOSAR System Template 4.3.1
- Support of FIBEX 4.1.2
- Support of container and contained IPDUs
- Support of secure onboard communication
- Support of global time synchronization

For more information on the new features, refer to [New Features of the RTI CAN MultiMessage Blockset 5.1](#) on page 157.

RTI FPGA Programming Blockset

The new key features of the RTI FPGA Programming Blockset 3.6 are:

- Extended Xilinx® software support.
- Enhancements to the FPGA framework for MicroAutoBox.
- New FPGA framework for MicroLabBox
- Enhancements to the FPGA frameworks for the DS2655 FPGA Base Board.

For more information on the new features, refer to [New Features of the RTI FPGA Programming Blockset 3.6](#) on page 163.

RTI LIN MultiMessage Blockset

The new key features of the RTI LIN MultiMessage Blockset are (already available with RTI LIN MultiMessage Blockset 3.0p1):

- Support of AUTOSAR System Template 4.3.1
- Support of FIBEX 4.1.2

For more information on the new features, refer to [New Features of the RTI LIN MultiMessage Blockset 3.1](#) on page 167.

SCALEXIO firmware

The new key feature of the SCALEXIO firmware is:

- Support of the DS6333-CS Automotive Ethernet Board
- Support of the DS6333-PE Automotive Ethernet Board
- Support of the DS6334-PE Ethernet Board
- Support of the DS6335-CS Ethernet Board
- Support of the SCALEXIO LabBox (8-slot)
- The firmware of the DS6001 Processor Board has the following new features:
 - Gigalink connection
 - Wake-up on I/O
 - Data set storage

For more information on the new features, refer to [New Features of the SCALEXIO Firmware 4.3](#) on page 171.

Sensor Simulation

Sensor Simulation is a new product that allows you to validate camera and laser sensors.

For more information on the new product, refer to [New Product: Sensor Simulation](#) on page 175.

SYNECT

The new key features of SYNECT 2.6 are:

- New license type.
- Simplified execution of test cases.
- Improved diagrams for connecting models.

For more information on the new features, refer to [New Features of SYNECT 2.6](#) on page 180.

SystemDesk

The new key features of SystemDesk 5.2 are:

- Support for the automatic configuration and code generation of the dSPACE LIN driver MCAL module.
- Creating adaptive applications for V-ECUs according to the AUTOSAR Adaptive Platform.

For more information on the new features, refer to [New Features of SystemDesk 5.2](#) on page 200.

TargetLink

The new key features of TargetLink 4.4 are:

- TargetLink Module for MATLAB® Code:
 - Generate code from MATLAB Function blocks in Simulink® models

- Generate code from MATLAB® functions in Stateflow charts
- Full specification of fixed-point and other code properties
- AUTOSAR support to create standard-compliant AUTOSAR Software Components from models containing MATLAB code components
- Call legacy functions from MATLAB® code
- Tool chain support:
 - Simulink implementation container (SIC) export, including data protection for multirate systems (production code is executable on dSPACE real-time hardware)
 - Synchronization of libraries with TargetLink data
 - Parallel build for SIL simulations
- Modeling in Simulink or Stateflow:
 - DD-based code generation for arrays of struct and access to array elements using access functions and custom code
 - Bus capable Constant block
 - Fast Restart mode for iterative model simulations
- AUTOSAR release and communications standardization:
 - Supported AUTOSAR releases
 - Support of Provide-Require Ports for Sender-Receiver communication
 - Support of Explicit NvData communication
 - Support of Rte_IWrite and Rte_IWriteRef in Sender-Receiver and NvData communication
- Target simulation (PIL simulation):
 - Support for virtual evaluation boards (Instruction Set Simulators (ISS))
- Data Dictionary and Property Manager improvements:
 - Support of Show masked content and Show library content in the Property Manager
 - New import and export of view sets in the Property Manager
 - Support for User Mode per workspace in the Data Dictionary Manager
- Other:
 - TargetLink demos for the new TargetLink features.

For more information on all the new features, refer to [New Features of TargetLink 4.4](#) and [TargetLink Data Dictionary 4.4](#) on page 208.

For more information on the TargetLink migration aspects (TargetLink, TargetLink AUTOSAR Module, TargetLink Data Dictionary), refer to [Migrating to TargetLink 4.4](#) and [TargetLink Data Dictionary 4.4](#) on page 223.

VEOS

The new key features of VEOS are:

- Support of memory segments and page switching
- Simulation in soft real time
- Measuring and calibrating variables of 64-bit applications
- Building 64-bit VPUs from CTLGZ files

- Support of VEOS Ethernet channels for bus monitoring and logging
- Support of the LIN communication driver module for LIN masters

For more information on the new features, refer to [New Features of VEOS 4.3](#) on page 259.

Aspects of Migrating from Previous Releases

Introduction

After you install products of the current dSPACE Release, some additional steps might be necessary. The migration steps required when you update from the last dSPACE Release are described in the product-specific migration topics in this document. If you update from an earlier dSPACE Release, refer to the related *New Features and Migration* document.

Migrating to dSPACE Release 2018-B

Introduction

After you install Release 2018-B, some additional steps might be necessary.

Migrating from dSPACE Release 2018-A

Product-specific migration steps Product-specific migration steps are usually performed automatically. For exceptions, refer to the product-specific migration descriptions.

Migrating from dSPACE Release 2017-B or earlier

To migrate from dSPACE Release 2017-B or earlier to Release 2018-B, you also have to perform the migration steps of the intervening dSPACE Releases. All of the required migration steps can be performed with Release 2018-B installed.

For more information on the required migration steps, refer to the *New Features and Migration* documents of the intervening dSPACE Releases.

Previous release documents

The PDF files of previous Releases are called **NewFeaturesAndMigrationxx.pdf**, where xx stands for the Release number.

You can find the *New Features and Migration* files for previous Releases in the following locations:

- In the installation folder of the current dSPACE Help. Refer to `C:\Program Files\Common Files\dSPACE\Help 2018-B\Print\PreviousReleases`.

- On the dSPACE DVDs. Refer to `\Doc\PreviousReleases`.
- At www.dspace.com/go/migration for download. Here, you can also find *New Features and Migration* documents for very early Releases.

Migrating Python Scripts from Python 2.7 to Python 3.6

Where to go from here

Information in this section

Basic Information on Python 3.6	34
General Information on Migrating	38
Product-Specific Information on Migrating	45

Basic Information on Python 3.6

Introduction

For all dSPACE software products supplied with dSPACE Release 2018-B that use Python, dSPACE provides migrated Python scripts for all demos and extensions installed with dSPACE software. The dSPACE software products distributed with dSPACE Release 2018-B therefore do not require any further migration steps. For detailed information and migration instructions for custom scripts, refer to [Product-Specific Information on Migrating](#) on page 45.

For migrating from an earlier Python version than 2.7 to Python 3.6, refer to Releases earlier than Release 2013-B.

You find this information also on the dSPACE website, refer to <http://www.dspace.com/go/Python36Migration>.

Where to go from here

Information in this section

Main Changes in Python 3.6	34
Main Changes in Handling Python 3.6 with dSPACE Software	35
General Information on Using Python Installations	36
Location of Python 3.6 DLL Files	37

Main Changes in Python 3.6

Main reason for using Python 3.6 instead of Python 2.7

Due to the end of life of Python 2.7, updates will be provided until end of April 2020. To ensure a stable API and optimum support for our customers, we decided to switch to Python 3.6 with dSPACE Release 2018-B. Python 3.x introduced considerable changes, which lead to an extended migration effort. With the broad adoption of Python 3.6, all Python extension libraries required by dSPACE software are available, which reduces the migration effort.

Related documents available on the Python website

For a short overview of the main changes in Python 3.6, refer to [Technical Changes](#) on page 39. For information on all the changes in the Python language and environment from Python 2.7 to Python 3.6, refer to Python Software Foundation at www.python.org.

The Python Software Foundation provides the following documents:

- What's New from Python 2.7 to 3.0:
<https://docs.python.org/3.6/whatsnew/3.0.html>

- What's New from Python 3.0 to 3.1:
<https://docs.python.org/3.6/whatsnew/3.1.html>
- What's New from Python 3.1 to 3.2:
<https://docs.python.org/3.6/whatsnew/3.2.html>
- What's New from Python 3.3 to 3.3:
<https://docs.python.org/3.6/whatsnew/3.3.html>
- What's New from Python 3.3 to 3.4:
<https://docs.python.org/3.6/whatsnew/3.4.html>
- What's New from Python 3.4 to 3.5:
<https://docs.python.org/3.6/whatsnew/3.5.html>
- What's New from Python 3.5 to 3.6:
<https://docs.python.org/3.6/whatsnew/3.6.html>

Main Changes in Handling Python 3.6 with dSPACE Software

Introduction

With Python 3.6, some aspects of handling Python with dSPACE software have changed.

Libraries

The libraries and components used with Python 3.6 and distributed on dSPACE DVDs have changed as shown in the following table.

Package	RLS 2018-A	RLS 2018-B
comtypes	1.1.3	1.1.4
Core	2.7.14	3.6.5
cycler	0.10.0	0.10.0
kiwisolver	-	1.0.1
matplotlib	1.5.3	2.2.2
numpy	1.13.3	1.14.3
pillow	4.3.0	5.1.0
pip	9.0.1	9.0.3
py2exe	0.6.9	.1)
pyparsing	2.2.0	2.2.0
Python-dateutil	2.6.1	2.7.2
pythonnet	2.3.0	2.3.1
pytz	2017.3	2018.4
pywin32	221.10	223.10

Package	RLS 2018-A	RLS 2018-B
six	1.11.0	1.11.0
wxPython	3.0.2.0	4.0.2

¹⁾ Because py2exe is not available in a stable version, it is not included in the dSPACE distribution. You can download alternative packages from the Internet, such as cx_Freeze or PyInstaller.

General Information on Using Python Installations

Introduction

If you work with dSPACE software from dSPACE Release 2018-A or earlier, which supports Python 2.7, and dSPACE software from dSPACE Release 2018-B, which supports Python 3.6, both Python versions are installed on the PC and can be used in parallel.

Limitations when using Python 2.7 and Python 3.6 in parallel

You can use both Python versions in parallel. However, you must observe the following limitations:

- You can set the file associations for PY and PYW files to only one Python version. This is usually the latest Python version you installed.
- Environment variables are used by both Python versions. You must set their values, for example, for PYTHONHOME, to the Python installation you want to work with. For an overview of environment variables set by Python, refer to: <https://docs.python.org/3.6/using/windows.html>.

Note

To avoid unintentional effects, which can be difficult to identify, you are recommended to not use environment variables. For the same reason, you should not extend the system path by a Python installation path.

Using dSPACE test automation with both Python versions in parallel

If a test automation script uses dSPACE Python Modules and you do not want to migrate the script, you have to work with both Python versions in parallel. The dSPACE Python Modules for Python 2.7 are available up to and including Release 2018-A.

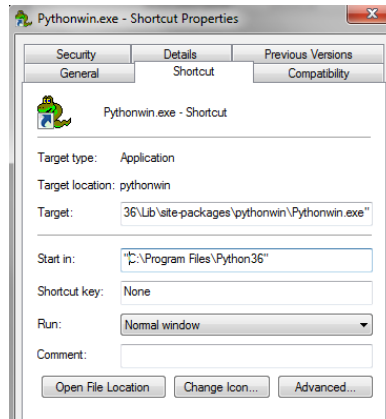
Note

It is recommended to use only dSPACE software from the same Release when using Python scripts. Using both Python versions in the same application context, such as automating the access to an older version of ControlDesk via AutomationDesk, might cause a conflict.

Location of Python 3.6 DLL Files

Introduction

To support multiple environments, DLL files are no longer installed in the system folder in Python 3.6. To locate Python 3.6 DLL files, you must set the system path to the Python installation folder, for example, by entering `Set PATH=C:\Program Files\Python36;%PATH%` or by specifying the start folder in a shortcut.



General Information on Migrating

Introduction The following introduces you to the major script modifications you need to perform.

Where to go from here

Information in this section

Migration via the 2to3 Tool	38
Technical Changes	39
Examples of Migration Issues	42
Related Developer Documents	44

Migration via the 2to3 Tool

Introduction With Python 3.x, substantial changes were introduced, but most of them can be migrated automatically. For this purpose, the Python distribution contains the **2to3** migration tool, which migrates Python 2 script code to Python 3 script code. The following introduces you to this migration tool and provides code switches for separating Python 2 from Python 3 code.

Required migration steps

The required migration steps depend on your particular requirements:

- Code must run only in Python 2.x: No action required.
- Code must run only in Python 3.x: Run the **2to3** migration tool. Then, check the syntax and manually change the non-migrated code to Python 3.x.
- Code must run in Python 2.x and 3.x: The code has to be compatible with Python 2.x. Write code in Python 2.x syntax and, before using Python 3.x, run the **2to3** migration tool. You do not have to change the code manually. However, if there are incompatible changes, you must surround these code fragments with a conditional expression.

After you performed the required steps, you execute the corresponding test code and check semantic changes, e.g., the return type is now byte instead of string.

Using the 2to3 migration tool

The **2to3** migration tool is based on the **lib2to3** library. It lets you incorporate the changes that were caused by the version update via *fixers*. You can call the tool via the command line.

```
"c:\Program Files\Python36\python.exe" "c:\Program Files\Python36\Tools\scripts\2to3.py" MyScript.py
```

After you executed the migration tool, the script changes are displayed in the command window. To change the script in the same directory, use the command line option `-w` (overwrite file) and `--no-diffs` (do not output differences).

```
"c:\Program Files\Python36\python.exe" "c:\Program Files\Python36\Tools\scripts\2to3.py" -w --no-diffs MyScript.py
```

For more information, refer to <https://docs.python.org/3.6/library/2to3.html>.

Incompatible code fragments

Incompatible code fragments are not converted automatically, because they are not identified by the `2to3` migration tool. To ensure that all code runs in Python 2.x and Python 3.x, you must surround these code fragments with a conditional expression. Refer to the following example:

```
if sys.version_info[0] >= 3:
    ...
else:
    ...
```

Technical Changes

Introduction

Migrating from Python 2.7 to Python 3.6 has caused substantial changes in the Python API, some of which are described in the following. Some of the changes are migrated automatically via the `2to3` migration tool, others must be migrated manually.

For known issues that you must migrate manually, refer to [Examples of Migration Issues](#) on page 42. If a specific migration problem is not described and you need further information, refer to [Related Developer Documents](#) on page 44.

Language syntax

The following table shows some of the differences identified between the language syntax of Python 2.7 and Python 3.6.

Change	Python 2.7	Python 3.6 Equivalent
<code>print</code> is a function	<code>print "The answer is", 2*2</code>	<code>print ("The answer is", 2*2)</code>
<code>exec</code> is a function	<code>exec stmt</code>	<code>exec(stmt)</code>
Text vs. data instead of Unicode vs. 8-bit	<code>u"..."</code> <code>"..."</code> <code>ur"\u20ac"</code>	<code>"..."</code> <code>b"..."</code> <code>"\u20ac"</code>
Views and iterators instead of lists	<code>k = d.keys()</code> <code>k.sort()</code> <code>map(...)</code> <code>zip(...)</code>	<code>k = d.keys()</code> <code>k = sorted(d)</code> <code>list(map(...))</code> <code>list(zip(...))</code>
Ordering comparisons	<code>1 < ''</code>	<code>TypeError</code> is raised
<code>int</code> is now <code>long</code>	<code>long(x)</code>	<code>int(x)</code>

Change	Python 2.7	Python 3.6 Equivalent
Division always returns float	<code>float(1)/2</code> <code>1/2</code>	<code>1/2</code> <code>1//2</code>
Non-ASCII letters as identifiers	<code>oe=10</code>	<code>ö=10</code>
New string format	<code>'Sir %s of %s' % ('Gallahad', 'Camelot')</code>	<code>'Sir {0} of {1}'.format('Gallahad', 'Camelot')</code>
Formatted string literals	-	<code>name = "Fred"</code> <code>f"He said his name is {name}."</code>
Underscores in numeric literals	-	<code>>>>1_000_000_000_000_000</code> <code>1000000000000000</code> <code>>>>0x_FF_FF_FF_FF</code> <code>4294967295</code>
Removed backticks	<code>`a`</code>	<code>repr(a)</code>
Removed string/long qualifiers	<code>u"..."</code> <code>2321838947817482382L</code>	<code>"..."</code> <code>2321838947817482382</code>
Removed generic imports in functions	<code>from module import *</code>	<code>- # define explicit import</code>
Removed exec file	<code>execfile(fn)</code>	<code>exec(open(fn).read())</code>
Removed reload	<code>reload()</code>	<code>importlib.reload()</code>
Removed has_key	<code>dict.has_key(x)</code>	<code>x in dict</code>
Removed <>	<code>A <> B</code>	<code>A != B</code>
Removed classic classes	<code>class Test(object):</code>	<code>class Test:</code>

Libraries

The following table shows the differences identified between the libraries of Python 2.7 and Python 3.6.

Change	Python 2.7	Python 3.6 Equivalent
Library name changes	<code>_winreg</code> <code>ConfigParser</code> <code>copy_reg</code> <code>Queue</code> <code>SocketServer</code> <code>markupbase</code> <code>repr</code> <code>test.test_support</code> <code>__builtin__</code>	<code>Winreg</code> <code>configparser</code> <code>copyreg</code> <code>queue</code> <code>socketserver</code> <code>_markupbase</code> <code>reprlib</code> <code>test.support</code> <code>builtins</code>
Raising exceptions	<code>raise Exception, args</code>	<code>raise Exception(args)</code>
Retrieving exception variables	<code>except SomeException, variable</code>	<code>except SomeException as variable</code>

Change	Python 2.7	Python 3.6 Equivalent
Renamed function members	<pre>func_closure func_code func_defaults func_dict func_doc func_globals func_name</pre>	<pre>__closure__ __code__ __defaults__ __dict__ __doc__ __globals__ __name__</pre>
New ordered dict	-	<code>collections.OrderedDict</code>
New argparse module	<code>import optparse</code>	<code>import argparse</code>
New literal_eval in ast	<code>eval("{'req': 3, 'func': 'pow', 'args': (2, 0.5)}")</code>	<code>ast.literal_eval("{'req': 3, 'func': 'pow', 'args': (2, 0.5)}")</code>
New escape in HTML	-	<pre>>>> html.escape('x > 2 && x < 7') 'x &gt; 2 &amp;&amp; x &lt; 7'</pre>
Changed names in unittest	<pre>assert_() assertEquals() assertNotEquals() assertAlmostEquals() assertNotAlmostEquals()</pre>	<pre>assertTrue() assertEqual() assertNotEqual() assertAlmostEqual() assertNotAlmostEqual()</pre>
Base for all IO/OS exceptions	-	Use <code>OSError</code>
New enum module	-	<pre>import enum enum.Enum enum.Flags</pre>
New os.scandir instead of os.walk	-	<pre>for entry in os.scandir(path): if not entry.name.startswith('.')\ and entry.is_file(): print(entry.name)</pre>
New constants inf, nan	<pre>inf = float('inf') nan = float('nan')</pre>	<pre>math.inf math.nan</pre>

Environment

The following introduces you to some of the differences between Python 2.7 and Python 3.6 with regard to the execution environment and the use of the file system.

- Source files are UTF-8 files.
- Byte-compiled files change name and location. The PYC files are located in a `__pycache__` folder, and the Python version is appended to the name of the byte code file, e.g., the file name of `mymodule.py` results in `mymodule.cpython-32.pyc`.
- Deprecation of PYO files. The new `opt-` tag in name replaces the PYO files.
- Support for unique PYD names for DLL extensions in the format `<name><_d>.cp<major><minor>-<platform>.pyd`, e.g., `win32com.cp36-win_amd64.pyd`.

- The compiler version has changed to VS2015. You must use this compiler version to build DLL extensions.
- Support for long paths.
- PY files are associated with a Python launcher.
- The new command line option `-I` for Python.exe excludes paths and environment variables.

Examples of Migration Issues

Introduction

The following introduces you to some migration issues that occur while migrating dSPACE products and libraries to Python 3.x.

Int and long are one type

In Python 2.x, integer type checks compared `int` and `long` types. Because `int` and `long` are one type in Python 3.x, you have to perform the following change to ensure code compatibility.

```
import sys
if sys.version_info.major < 3:
    integerTypes = (int, long)
else:
    integerTypes = (int)

if isinstance(XXX, integerTypes):
...

```

Old-style strings are byte sequences

Strings are unicode in Python 3.x, and file operations return bytes instead of strings. To ensure compatibility, you must convert from string to byte in Python 3.x. For compatible code, you must define a `b` function.

```
import sys
if sys.version_info < (3,):
    def b(x):
        return x
else:
    def b(x):
        return x.encode(sys.getdefaultencoding())

```

Type modules deprecated

In earlier Python versions, the `types` module was used. This option is no longer available in Python 3.x. Use the `type` class and `isinstance` function instead, as shown in the following example.

```
# Python < 2.x
if type(XXX) == types.int

# Python >= 2.x
if isinstance(XXX, int):

```

Sorted functions signature

The `sorted` function signature has changed. The explicit compare parameter has been removed. Change the call providing the sorting key.

```
# Python < 2.x
# sorted(iterable[, cmp[, key[, reverse]]])
sorted (myList, lambda x,y : cmp(x,y), lambda x: myDict["Key"])

# Python >= 2.x
# sorted(iterable[, key][, reverse])
sorted (myList, lambda x: myDict ["Key"])
```

String module functions removed

The compatibility functions in the `string` module have been removed. You now have to use the methods on the `string` object. Change the signature as shown in the following example.

```
# Python <= 2.x
validParamNamesStr=string.join(names + options,",")

# Python > 2.x
validParamNamesStr= ",".join(names + options)
```

None is not a type

`None` is now an object and no longer a type. You must therefore replace `type` checking with `None`, as shown in the following example.

```
# Python <= 2.x
if not isinstance(Variable, (int, type(None))):

# Python > 2.x
if not (Variable is None or isinstance(Variable, int)):
```

Subprocess changed return type

Like the file I/O methods, the methods of the subprocess module return bytes instead of strings. To enable comparison, you must convert one comparand to an equal type, i.e., either from string to bytes or bytes to string.

2to3 relative import conversion

By mistake, relative imports used with an alias are converted by the `2to3` migration tool. You must correct the import manually, as shown in the following example.

```
import Module.SubModule as SubModule
# converted to
from . import Module.SubModule as SubModule

# correct
from . Module import SubModule
```

Improved escape sequence check

Invalid escape sequences were ignored in Python 2.x. In Python 3.x, they are identified as syntax errors. You must change the escape sequence as shown in the following example.

```
# Incorrect escape sequence
"Menu\New File"
# correct escape sequence
r"Menu\New File"

"Menu\\New File"
```

Change `execfile` handling

The `execfile` Python API function was removed in Python 3.x. There are multiple ways to replace the corresponding code, either by executing the string or executing a code object. The execution of the code object retains the connection to the corresponding file and hence allows debuggers to set breakpoints in the code. The execution of the pure string is completely dynamic and has no connection to the source file.

```
# Python <= 2.x
execfile(filename)

# Python > 2.x
exec(open(filename).read())
exec(compile(open(filename).read(), filename, 'exec'))
```

Related Developer Documents

Introduction

The migration from Python 2.x to Python 3.x was performed by several developers throughout the last years. Typical migration problems were documented in different blogs and websites, and even further migration tools were introduced to enhance the features of the `2to3` migration tool.

The following list contains some document locations related to the migration of Python 2.x to Python 3.x:

- <https://portingguide.readthedocs.io/en/latest/>
- <http://python3porting.com/>
- http://python-notes.curious efficiency.org/en/latest/python3/questions_and_answers.html
- <https://docs.python.org/3.6/howto/pyporting.html>

Product-Specific Information on Migrating

Products with editable Python content

Python scripts are used both explicitly and implicitly in dSPACE products. With the change to Python 3.6, all scripts have to be migrated to run in the new environment. The following introduces you to the migration steps in dSPACE products that are affected by the Python migration.

All other dSPACE products are not affected by the Python migration.

Where to go from here

Information in this section

AutomationDesk	45
Bus Manager (Stand-Alone)	46
ConfigurationDesk - Implementation Version	46
ControlDesk	47
ModelDesk	47
Platform API Package	47
Real-Time Testing	48
SYNECT	48
SystemDesk	50

AutomationDesk

Basics on migrating Python in AutomationDesk

If you load an older AutomationDesk project or custom library with the current AutomationDesk version, the update mechanism of AutomationDesk automatically migrates the following Python contents to Python 3.6. The update mechanism internally uses the `2to3` migration tool.

- Python files contained as Python modules or packages in custom libraries.
- Python files contained in the project's attachment folder.
- Python code of Exec blocks.
- Python code of Eval blocks.
- Python terms of Condition data objects.
- Python code for data objects used by the Customize Edit Dialog feature.

One aspect of the migration is the conversion of `String` data objects. A `String` data object is migrated from `bytes` to `str` with UTF-8 encoding. If this migration fails, AutomationDesk attempts conversion with `mbs` encoding. If this

also fails, the UTF-8 encoding is used and the characters that were not migrated are replaced by the character `U+FFFD`.

This migration aspect is also relevant to byte strings used in value lists, which are converted to `text`. The byte strings in lists, dictionaries, tuples, and variants are converted to `text(str)`.

A backup file (.bak) is automatically created for an external Python file before the migration starts. Python code in blocks and data objects that cannot be migrated remains unchanged and must be migrated manually.

Artifacts not included in migration

The following Python scripts are not migrated automatically:

- Custom extension scripts
Custom extension scripts that are not included in the attachment file or have not been added as visible modules in the Library Browser are not migrated automatically.
- Custom user function scripts

Bus Manager (Stand-Alone)

Basics on migrating Python

Demo scripts provided by dSPACE All Python demo scripts that are provided for Bus Manager (stand-alone) as part of dSPACE's product installation can be used in Python 3.6.

Artifacts not included in migration

Custom user function scripts are not migrated automatically.

ConfigurationDesk - Implementation Version

Basics on migrating Python

Demo scripts provided by dSPACE All Python demo scripts that are provided for ConfigurationDesk - Implementation Version as part of dSPACE's product installation can be used in Python 3.6.

Artifacts not included in migration

Custom user function scripts are not migrated automatically.

ControlDesk

Basics on migrating Python

For information on ControlDesk-specific migration steps, refer to [Migrating to ControlDesk 6.4](#) on page 109.

ModelDesk

Basics on migrating Python

Migrating projects When you open a ModelDesk project, the project and experiment scripts are automatically migrated with the `2to3` migration tool.

Artifacts not included in migration

Custom user function scripts are not migrated automatically.

Platform API Package

Basics on migrating Python

The following products of the dSPACE Platform API Package are relevant to the Python migration:

- Test Automation Python Modules
- dSPACE XIL API Implementation (MAPort)
- dSPACE Platform Management API

Demo scripts provided by dSPACE All Python demo scripts that are provided as part of dSPACE's product installation can be used in Python 3.6.

Modules provided by dSPACE The `matlablib2` and `rs232lib2` modules provided by the Platform API Package are migrated and work only in Python 3.6.

Artifacts not included in migration

Custom user function scripts are not migrated automatically in the Platform API Package.

Note

You must migrate all custom scripts to Python 3.6 manually. The Platform API Package does not provide any features to automatically migrate custom scripts from Python 2.7 to Python 3.6.

Real-Time Testing

Basics on migrating Python

Demo scripts The Python code of the demo scripts work in Python 3.6. Because the Python interpreter on the real-time platforms continues to use Python 2.7, the RTT sequences must not be changed. However, you can use the syntax of Python 3.6.

Artifacts not included in migration

The following custom scripts are not migrated automatically:

- Custom scripts for the host PC
- Custom sequences for the real-time platform
- Custom user function scripts

SYNECT

Basics on migrating Python

Python Interpreters In SYNECT, Python is used on the client and on the server side. SYNECT client has an integrated Python interpreter and also uses another Python interpreter to execute extension scripts. These interpreters change from Python 2.7 to 3.6.

SYNECT server has an integrated IronPython interpreter. This interpreter is available in Python 2.7.

Note

To ensure that Python scripts can be run in SYNECT, all scripts executed by SYNECT client's internal interpreters must be migrated from Python 2.7 to 3.6 with the `2to3` migration tool. The server scripts have to remain in Python 2.7.

Scripts provided by dSPACE All Python scripts that are provided by dSPACE as part of the product installation, such as demos, add-ons, Python import/export plug-ins, are compatible to Python 3.6.

Artifacts not included in migration

You must migrate custom scripts to Python 3.6 manually. SYNECT does not provide any features to automatically migrate custom scripts from Python 2.7 to Python 3.6.

Migration of custom scripts

You must manually migrate to Python 3.6 to ensure that custom Python scripts are executable in SYNECT client's internal Python interpreter.

The following types of custom scripts have to be migrated:

- Stand-alone scripts
- Extension scripts
- Scripts that are part of add-ons
- Scripts that are part of script sequencers

Migration of custom Python plug-ins

SYNECT provides various import and export functionalities, such as the Test Management functionality, which let you use Python plug-in scripts to parse input files and generate export files.

Note

You must migrate the plug-in scripts to Python 3.6 manually, to ensure that they can be executed in SYNECT.

Migration of custom scripts used to preprocess and postprocess data

Some of SYNECT's import and export functionalities let you use scripts to preprocess and postprocess data. You must reference these scripts in ECXML files to execute them on the SYNECT client before or after an import or export operation.

Note

You must migrate the custom scripts used for preprocessing and postprocessing data to Python 3.6 manually to ensure that they can be executed in SYNECT.

Migration of custom Python workflow

SYNECT's Workflow Management supports Python steps. Python steps either contain embedded Python scripts, i.e., the Python code is stored in the SYNECT database, or reference a Python file in the file system. During the execution of a workflow, Python scripts of this type are executed in the SYNECT client.

Note

You must migrate the scripts that are used for Workflow Management to Python 3.6 manually to ensure that they can be executed in SYNECT. If versions of SYNECT with different Python versions access the same script, either on the file system or in the database, you must change these scripts in a way to ensure that they are compatible with all Python versions used.

Migration of server scripts

Server scripts must not be migrated. The IronPython interpreter in SYNECT server continues to use Python 2.7.

SystemDesk

Basics on migrating Python

Validation rule scripts Validation rule scripts must not be migrated. The validation rules are executed via the IronPython interpreter in SystemDesk. Because IronPython is independent of CPython, the language version continues to be Python 2.7.

Automotive Simulation Models (ASM)

Where to go from here

Information in this section

All ASM Products	52
ASM Base InCylinder	53
ASM Brake Hydraulics	54
ASM Diesel Engine	55
ASM Diesel Exhaust	57
ASM Diesel InCylinder	58
ASM Drivetrain Basic	59
ASM Electric Components	60
ASM Environment	63
ASM Gasoline Engine Basic	65
ASM Gasoline Engine	66
ASM Gasoline InCylinder	67
ASM KnC	69
ASM Traffic	70
ASM Truck	72
ASM Turbocharger	74
ASM Utils	75
ASM Vehicle Dynamics	76

All ASM Products

Changes in All ASM Demo Models

Engine start button

The ASM demo models are now equipped with an engine start button instead of a classic ignition key.

The model simulates the functionality of engine start and stop with a state machine by using a simple assumption for the key authorization process, modeled as an external switch.

If the button is pressed and held while the start conditions are being fulfilled, the engine starts.

These conditions are:

- Manual transmission: Clutch pedal is pressed.
- Automatic transmission: Brake pedal is pressed and selector lever is in neutral or in park position.

A running engine is switched off when the button is pressed and held.

The model also lets you stimulate the control signals and bypass the button state machine.

Migrated demo models do not automatically have the new feature. To benefit from this feature, you must manually modify the model. In this case, use the ASM demo models as a template.

Multi-instance support for custom library blocks

As of dSPACE Release 2018-B, blocks of a ModelDesk custom library support the multi-instance feature.

To use this feature, the custom library and the model that includes blocks of this library must be migrated.

For more information on migrating, refer to [How to Migrate Blocks of Custom Component Libraries](#) ([📖 ASM User Guide](#)).

Note

It is recommended to migrate the custom library and the model. Without migration, the blocks of this library are only supported without multi-instance features.

ASM Base InCylinder

Migrating to ASM Base InCylinder Blockset 2.4

WALL_HEAT block	The name of the Ignition[0]1] inport has been corrected.
INTAKE_VALVE block	The background color of the look-up table for the valve lift has been changed to ASM standard green.
EXHAUST_VALVE block	The background color of the look-up table for the valve lift has been changed to ASM standard green.
APU_EVENT block	The issue that injection times have to be nonzero for at least one cycle before later injections can be correctly passed through has been solved.
Resettable delay	<p>The Unit Delay Resettable block from the slupdate library has been replaced with the Delay block from the Discrete Simulink library. The blocks behave in exactly the same way. This has been done to avoid warnings. The Unit Delay Resettable block will be discontinued by MATLAB in the future.</p> <p>The following ASM blocks are affected:</p> <ul style="list-style-type: none"> ▪ DIRECTINJECTOR_TIMING ▪ EGR_RATE_CONTROL ▪ ENGINE_OPERATION ▪ TRIGGER_INJ_UPDATE

Related topics

Basics

[Migrating ASM Models \(📖 ASM User Guide\)](#)

ASM Brake Hydraulics

Where to go from here

Information in this section

New Features of ASM Brake Hydraulics Blockset 2.1	54
Changes in the ASM Brake Hydraulics Demo Model	54

New Features of ASM Brake Hydraulics Blockset 2.1

MASTER_BRAKE_CYLINDER block

Two new inports have been added to the MASTER_BRAKE_CYLINDER block:

- p_Brake_Desired
- Enable_p_Brake_Desired

In addition, a new brake-by-wire brake cylinder mode has been added to the block to set an externally requested brake pressure.

Changes in the ASM Brake Hydraulics Demo Model

BRAKE_HYDRAULICS block

To adapt the structure of the brake hydraulics model to a battery electric and hybrid vehicle brake system (brake-by-wire), the master brake cylinder has been moved outside the hydraulic demo model.

The PRESSURE_INTERVENTION_SOFT_ECU subsystem has been deleted. Its functionality has been integrated in the MASTER_BRAKE_CYLINDER block.

The general functionality of the demo model has not been changed.

BRAKE_HYDRAULICS_BASIC block

To adapt the structure of the brake hydraulics model to a battery electric and hybrid vehicle brake system (brake-by-wire), the master brake cylinder has been moved outside the hydraulic demo model.

The PRESSURE_INTERVENTION_SOFT_ECU subsystem has been deleted. Its functionality has been integrated in the MASTER_BRAKE_CYLINDER block.

The general functionality of the demo model has not been changed.

ASM Diesel Engine

Where to go from here

Information in this section

New Features of ASM Diesel Engine Blockset 2.6.2	55
Migrating to ASM Diesel Engine Blockset 2.6.2	55

New Features of ASM Diesel Engine Blockset 2.6.2

Reset

All blocks that contain integrators and memory blocks are reset at the same time when the reset is active. Even if the simulation contains a NaN or Inf value, the simulation is restored when you click the Reset button in ControlDesk.

Migrating to ASM Diesel Engine Blockset 2.6.2

COMBUSTION_TORQUE_CI block

The continuous integrator of the PT1 block in the Engine Out Temperature block inside the COMBUSTION_TORQUE_CI block has been replaced by a discrete integrator.

Resettable delay

The unit delay and memory blocks are replaced with a Delay block with reset input. The following blocks are affected:

- CYLINDER_INLET
- DIRECTINJECTOR
- PUMP_TORQUE

The Unit Delay Resettable block from the slupdate library has been replaced with the Delay block from the Discrete Simulink library. The blocks behave in exactly the same way. This has been done to avoid warnings. The Unit Delay Resettable block will be discontinued by MATLAB in the future.

The following blocks are affected:

- DIRECTINJECTOR_TIMING
- EGR_RATE_CONTROL
- ENGINE_OPERATION
- HPP_CRANKBASED
- IGNITION_SET
- INJECTOR_MODE

- PORTINJECTOR_TIMING
- RAIL_CONTROL
- RAIL_CONTROL_CRANKBASED
- TRIGGER_INJ_UPDATE

Related topics

Basics

[Migrating ASM Models](#) (📖 ASM User Guide)

ASM Diesel Exhaust

Migrating to ASM Diesel Exhaust Blockset 2.1.7

DIESEL_OXIDATION_ **CATALYST block**

There is a new inport in the DIESEL_OXIDATION_CATALYST/Pressure_Drop/PT1_p_In_DOC block: p_Ambient[Pa]. It is used to initialize the integrator of the block.

Until now, the integrator was initialized with a constant block. During migration, a constant block with the original value is added to the new inport to obtain the same functionality as before.

Related topics

Basics

[Migrating ASM Models \(📖 ASM User Guide\)](#)

ASM Diesel InCylinder

Where to go from here

Information in this section

Changes in the ASM Diesel InCylinder Demo Model	58
Migrating to ASM Diesel InCylinder Blockset 2.6	58

Changes in the ASM Diesel InCylinder Demo Model

Reset

All blocks that contain integrators and memory blocks are reset at the same time when the reset is active. Even if the simulation contains a NaN or Inf value, the simulation is restored when you click the Reset button in ControlDesk.

Migrating to ASM Diesel InCylinder Blockset 2.6

Resettable delay

The Unit Delay Resettable block from the slupdate library has been replaced with the Delay block from the Discrete Simulink library. The blocks behave in exactly the same way. This has been done to avoid warnings. The Unit Delay Resettable block will be discontinued by MATLAB in the future.

The following ASM blocks are affected:

- DIRECTINJECTOR_TIMING
- ENGINE_OPERATION
- EGR_RATE_CONTROL
- RAIL_CONTROL
- TRIGGER_INJ_UPDATE

Related topics

Basics

[Migrating ASM Models \(📖 ASM User Guide\)](#)

ASM Drivetrain Basic

Where to go from here

Information in this section

New Features of ASM Drivetrain Basic Blockset 5.2	59
Migrating to ASM Drivetrain Basic Blockset 5.2	59

New Features of ASM Drivetrain Basic Blockset 5.2

START_BUTTON block

A new START_BUTTON block has been added to the SoftECU subsystem to simulate an engine start button.

Migrating to ASM Drivetrain Basic Blockset 5.2

LONGITUDINAL_CONTROL block

There have been bug fixes related to inaccurate tracking of low velocities and to not responding to the preview reference speed during vehicle start.

TEST_CYCLE block

The new Map_Sw_Engine parameter has been added to simulate the engine start button for test cycles.

MANEUVER_CONTROL block

The ASMSignalBus output has been added.

Related topics

Basics

[Migrating ASM Models \(📖 ASM User Guide\)](#)

ASM Electric Components

Where to go from here

Information in this section

New Features of ASM Electric Components Blockset 3.7	60
Migrating to ASM Electric Components Blockset 3.7	60

New Features of ASM Electric Components Blockset 3.7

TRQ_REQUEST_ COORDINATION_BEV block

Two new parameters for defining the nonlinear torque request depending on the accelerator pedal position have been added to the block:

- Map_Factor_Trq_Max_EM_Front
- Map_Factor_Trq_Max_EM_Rear

In addition, the Map_Pos_AccPedal_Start parameter has been added to the block to set an additional start accelerator pedal position depending on the vehicle velocity.

Migrating to ASM Electric Components Blockset 3.7

BATTERY_MULTICELL block

The calculation of the thermal radiation has been corrected to avoid malfunction if *Number of cells in y direction* is set to 1.

The calculation of the thermal loss reaction power has been corrected to avoid a long-term battery temperature drift.

The exchange of the P_Joule signal with the P_Cooling signal in the ASMSignalBus has been corrected.

BATTERY block

The calculation of the thermal loss reaction power has been corrected to avoid a long-term battery temperature drift.

BATTERY_THERMAL block

The calculation of the thermal loss reaction power has been corrected to avoid a long-term battery temperature drift.

AIR_CHANNEL block	The new SIM.StepSize parameter has been added for the sample time of the discrete integrator instead of a fixed numerical value.
TRQ_REQUEST_ COORDINATION_BEV block	<p>The new v_Vehicle[km h] inport has been added and is connected to a constant block with the value zero during migration.</p> <p>This inport must be connected to the vehicle velocity signal if the new map parameter Map_Pos_AccPedal_Start for setting an additional start accelerator pedal position depending on the vehicle velocity is used. Otherwise, a permanent zero value for the vehicle velocity can lead to errors in the simulation results.</p>
ENGINE_POSITION block	The new Reset[0 1] inport has been added to reset the output value.
SQUIRREL_CAGE_ ASYNCHRONOUS_ MACHINE_D_Q block	The name of the ASMSignalBus has been changed from SquirrelCage_AsynchronousMachine_d_q to SCIM_d_q.
PERMAMENT_MAGNET_ SYNCHRONOUS_MACHINE_D _Q block	The name of the ASMSignalBus has been changed from PermanentMagnet_SynchronousMachine_d_q to PMSM_d_q.
PMSM_D_Q_NONLINEAR block	The name of the ASMSignalBus has been changed from PermanentMagnet_SynchronousMachine_d_q to PMSM_d_q_Nonlinear.
SCIM_CONTROLLER_BASIC block	The tag visibility of the Goto blocks has been changed from <i>global</i> to <i>scoped</i> .
SCIM_CONTROLLER block	The tag visibility of the Goto blocks has been changed from <i>global</i> to <i>scoped</i> .
PMSM_CONTROLLER_BASIC block	The tag visibility of the Goto blocks has been changed from <i>global</i> to <i>scoped</i> .
PMSM_CONTROLLER block	The tag visibility of the Goto blocks has been changed from <i>global</i> to <i>scoped</i> .
PMSM_CONTROLLER_THREE_ LEVEL block	The tag visibility of the Goto blocks has been changed from <i>global</i> to <i>scoped</i> .

HALF_BRIDGE_INVERTER block	The current states, the freewheeling state, and the voltage drop at negative current have been updated.
THREE_PHASE_INVERTER block	The current states, the freewheeling state, and the voltage drop at negative current have been updated.
TRHEE_PHASE_DCM_INVERTER block	The current states, the freewheeling state, and the state of the half bridge <i>state_Halfbridge</i> when the duty cycles are nonzero have been updated.
THREE_PHASE_RECTIFIER block	The du_dt block has been replaced by the Discrete_Differentiator block in the Current_and_Voltage_Calculation subsystem.
ALTERNATOR block	The exciter voltage and the armature current have been limited. The RL_System_Select block has been updated.
ALTERNATOR_5_0 block	The block has been moved to the former version blocks to obtain the old functionality.
BRUSHLESS_DC_MACHINE_ALPHA_BETA block	The Const_Psi parameter for flux induced by permanent magnets has been changed to the Const_Back_EMF parameter for the back EMF constant.
STARTER block	The RL_System_Select block has been updated.

Related topics

Basics

[Migrating ASM Models \(📖 ASM User Guide\)](#)

ASM Environment

Where to go from here

Information in this section

New Features of ASM Environment Blockset 4.9	63
Migrating to ASM Environment Blockset 4.9	64

New Features of ASM Environment Blockset 4.9

V_ROAD_REF block

The driver behavior when the vehicle slips off the road has been improved.

Until now, when the vehicle was no longer on the road, the driver kept trying to find the road with the same maneuver speed. In most cases, this speed was too high and led to an instable steering behavior, which caused the driver to oscillate about the road reference line.

The new model assumes the maneuver speed to be valid only on the road. Once the vehicle leaves the road, a relatively small speed is assumed.

ROAD block

The ROAD block now supports junctions that feature height gradients. Until now, the junction plate had to be level.

DRIVER_CURVATURE block

The new DRIVER_CURVATURE block provides the curvature for the reference velocity calculation.

The block supports two different modes:

1. **ROAD:** The curvature is calculated on the basis of the geometric form factor of the road network.
2. **DRIVER:** The curvature is calculated on the basis of the desired trajectory of the ASM vehicle.

GPS_POSITION block

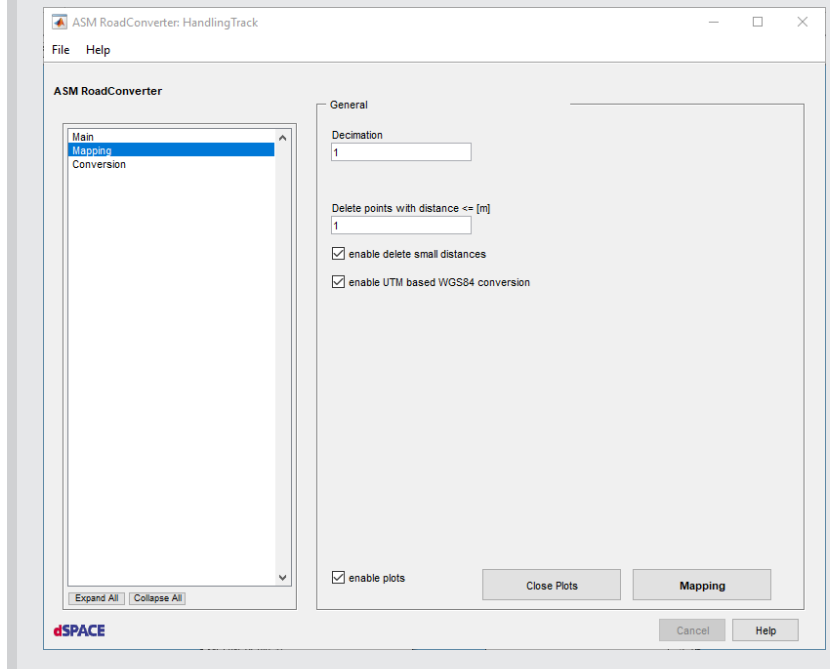
The new GPS_POSITION block converts the x/y-position information to GPS coordinates depending on the specified conversion method (spherical or based on the WGS 84 reference system). Additional position offset information provided by the ROAD block via user signals is also processed.

The GPS_POSITION block is included in all ASM demo models linked to the ASM Environment Library.

Note

When working with a model from a Release earlier than Release 2018-B and ASM Road Converter from Release 2018-B, do not select *enable UTM based WGS84 conversion*.

For models older than Release 2018-B, only the spherical conversion method is available.



Migrating to ASM Environment Blockset 4.9

V_ROAD_REF block

Two new inports have been added to the block. During migration, these ports are connected to dummy values to maintain the previous behavior.

Related topics

Basics

[Migrating ASM Models \(📖 ASM User Guide\)](#)

ASM Gasoline Engine Basic

Migrating to ASM Gasoline Engine Basic Blockset 2.2.2

Resettable delay

The Unit Delay Resettable block from the slupdate library has been replaced with the Delay block from the Discrete Simulink library. The blocks behave in exactly the same way. This has been done to avoid warnings. The Unit Delay Resettable block will be discontinued by MATLAB in the future.

The following blocks are affected:

- ENGINE_OPERATION
- IGNITION_SET
- PORTINJECTOR_TIMING
- TRIGGER_INJ_UPDATE

Related topics

Basics

[Migrating ASM Models](#) (📖 ASM User Guide)

ASM Gasoline Engine

Migrating to ASM Gasoline Engine Blockset 4.0.2

CATALYST block

A new inport has been added to the block: `p_Ambient[Pa]`. It is used to initialize the integrator of the `CATALYST/Pressure_Drop/PT1` block. Until now, the integrator was initialized with a constant block. During migration, a constant block with the original value is added to the new `p_Ambient[Pa]` inport to obtain the same functionality as before.

Until now, the `PT1` block in `Mapbased_T_In_Cat` had incorrect connections. Therefore, the `Mapbased_T_In_Cat` block could not be used in the simulation. This has been corrected.

LAMBDA_CONTROL block

Division by zero is now avoided if `lambda` is set to zero by the simulation.

Resettable delay

The `Unit Delay Resettable` block from the `slupdate` library has been replaced with the `Delay` block from the `Discrete Simulink` library. The blocks behave in exactly the same way. This has been done to avoid warnings. The `Unit Delay Resettable` block will be discontinued by MATLAB in the future.

The following blocks are affected:

- `CYLINDER_INLET`
- `DIRECTINJECTOR`
- `DIRECTINJECTOR_TIMING`
- `EGR_RATE_CONTROL`
- `ENGINE_OPERATION`
- `HPP_CRANKBASED`
- `IGNITION_SET`
- `INJECTOR_MODE`
- `PORTINJECTOR_TIMING`
- `PUMP_TORQUE`
- `RAIL_CONTROL`
- `RAIL_CONTROL_CRANKBASED`
- `TRIGGER_INJ_UPDATE`

Related topics

Basics

[Migrating ASM Models \(📖 ASM User Guide\)](#)

ASM Gasoline InCylinder

Where to go from here

Information in this section

New Features of ASM Gasoline InCylinder Blockset 2.6	67
Changes in the ASM Gasoline InCylinder Demo Model	67
Migrating to ASM Gasoline InCylinder Blockset 2.6	67

New Features of ASM Gasoline InCylinder Blockset 2.6

CATALYST block

A new CATALYST block is added to correctly simulate lambda response considering oxygen storage.

Changes in the ASM Gasoline InCylinder Demo Model

Reset

All blocks that contain integrators and memory blocks are reset at the same time when the reset is active. Even if the simulation contains a NaN or Inf value, the simulation is restored when you click the Reset button in ControlDesk.

Migrating to ASM Gasoline InCylinder Blockset 2.6

Catalyst_7_0 block

During migration, the former Catalyst block is moved to a former version.

LAMBDA_CONTROL block

Division by zero is now avoided if lambda is set to zero by the simulation.

Resettable delay

The Unit Delay Resettable block from the slupdate library has been replaced with the Delay block from the Discrete Simulink library. The blocks behave in exactly the same way. This has been done to avoid warnings. The Unit Delay Resettable block will be discontinued by MATLAB in the future.

The following ASM blocks are affected:

- DIRECTINJECTOR_TIMING
- EGR_RATE_CONTROL
- ENGINE_OPERATION
- IGNITION_SET
- INJECTOR_MODE
- PORTINJECTOR_TIMING
- RAIL_CONTROL
- TRIGGER_INJ_UPDATE

Related topics

Basics

[Migrating ASM Models](#) (📖 ASM User Guide)

ASM KnC

Where to go from here

Information in this section

New Features of ASM KnC 8.0.0	69
Migrating to ASM KnC 8.0.0	69

New Features of ASM KnC 8.0.0

Installation

ASM KnC is now integrated in the RCP and HIL setup. After RCP and HIL installation and successful decryption, you can start ASM KnC in the MATLAB Command Window via the `asm_knc` command.

User interface

The user interface of ASM KnC has been improved.

Functionality

There are only minor changes to the functionality of ASM KnC.
Like all dSPACE products, ASM KnC writes log messages in the dSPACE log file.
Missing properties in the property grid have been added.

User documentation

User documentation is now available for ASM KnC.

Migrating to ASM KnC 8.0.0

Automatic migration

ASM KnC automatically migrates old version project files to the new version 8.0.0.

ASM Traffic

Where to go from here

Information in this section

New Features of ASM Traffic Blockset 3.9	70
Changes in the ASM Traffic Demo Model	71

New Features of ASM Traffic Blockset 3.9

COLLISION_SENSOR block	The COLLISION_SENSOR block has been added to the Sensors subsystem. The sensor detects collisions between the ASM vehicle and surrounding objects and fellows based on their contour lines.
FELLOW_VELOCITIES block	The FELLOW_VELOCITIES block has been added to the ASM Traffic library. It calculates the linear and angular velocities of the fellows. The linear velocities are calculated in local fellow coordinates.
LINE_SENSOR block	The LINE_SENSOR block outputs the simulation ID of the detected shapes. The simulation ID can be set in ModelDesk and is also output by the Object_Sensor_2D block.
OBJECT_SENSOR_3D_GEOMETRY_PARAMETERS block	<p>The Sw_Occlusion_Mode parameter has been extended by a new occlusion mode: Occlusion Ratio 3-D.</p> <p>A new parameter has been added to the block: Threshold_Occlusion_Ratio. If the calculated occlusion ratio of the object sensor 3-D exceeds the defined threshold, the object is not detected.</p>
OBJECT_SENSOR_3D_PARAMETERS block	<p>The Sw_Occlusion_Mode parameter has been extended by a new occlusion mode: Occlusion Ratio 3-D.</p> <p>A new parameter has been added to the block: Threshold_Occlusion_Ratio. If the calculated occlusion ratio of the object sensor 3-D exceeds the defined threshold, the object is not detected.</p>
OBJECT_SENSOR_3D_CALCULATION block	The OBJECT_SENSOR_3D_CALCULATION block has been extended by a new occlusion mode: Occlusion Ratio 3-D:

In this mode, all detected objects are projected to a projection plane. For each object, the occlusion ratio is calculated in the following way: *hidden projection area/total projection area*

It is calculated, whether the nearest point of an object is occluded by another object.

Changes in the ASM Traffic Demo Model

Support of DS1006 Processor Board and DS1202 Base Board

The number of features which are part of the ASM Traffic demo model continually grows. This is the reason why the current ASM Traffic configuration triggers task overruns on the following hardware:

- The DS1006 Processor Board
- MicroLabBox with DS1202 Base Board

For a workaround, refer to [Troubleshooting](#) ( [ASM Traffic Guide](#)).

FellowMovement subsystem

A new FELLOW_VELOCITIES block has been added to the FellowMovement subsystem. It calculates fellow velocities.

Sensors subsystem

A new COLLISION_SENSOR block has been added to the Sensors subsystem.

ASM Truck

Where to go from here

Information in this section

Changes in the ASM Truck Demo Model	72
Migrating to ASM Truck Blockset 3.1	72

Changes in the ASM Truck Demo Model

Cabin mass

Additional cabin mass has been included in the model.

Suspension kinematics

Suspension kinematics of the rigid truck has been made the default suspension kinematics.

Migrating to ASM Truck Blockset 3.1

SUSCOMP_RIGID_SYM_REAR_2ND

A new inport has been added to the block: Trq_y_Tire.

SUSCOMP_RIGID_SYM_REAR_3RD

A new inport has been added to the block: Trq_y_Tire.

SUSPENSION_COMPLIANCE_REAR_2ND

A new inport has been added to the block: Trq_y_Tire.

SUSPENSION_COMPLIANCE_REAR_3RD

A new inport has been added to the block: Trq_y_Tire.

SUSPENSION_COMPLIANCE_OPP_REAR_2ND

A new inport has been added to the block: Trq_y_Tire.

SUSPENSION_COMPLIANCE_
OPP_REAR_3RD A new inport has been added to the block: Trq_y_Tire.

CABIN_MASS Internal calculations have been simplified.

PIVOT_POS_CABIN A signal has been added to the ASMSignalBus.

VEHICLE_FLEXIBLE_MASS_
AND_ADD_LOADS Three inports have been added to the block.

- M_Body
- PosVec_CoG_Body[x;y;z]
- Inertia_Body[3x3]

Related topics

Basics

[Migrating ASM Models \(📖 ASM User Guide\)](#)

ASM Turbocharger

Migrating to ASM Turbocharger Blockset 3.2.3

Resettable delay

The Unit Delay Resettable block from the slupdate library has been replaced with the Delay block from the Discrete Simulink library. The blocks behave in exactly the same way. This has been done to avoid warnings. The Unit Delay Resettable block will be discontinued by MATLAB in the future.

The following blocks are affected:

- SHAFT_TC
- SHAFT_TC_HP
- TURBINE
- TURBINE_HP
- TURBO_BASIC_2STAGE
- TURBO_CONTROL_MODE

Related topics

Basics

[Migrating ASM Models \(📖 ASM User Guide\)](#)


ASM Utils

New Features of ASM Utils 4.1

Multi-instance support for custom component library blocks

An ASM block of an ASM library can be used multiple times in a model. With ModelDesk's multi-instance feature, each of these blocks can be accessed.

Now, the multi-instance feature also supports blocks from a ModelDesk Custom Component Library.

For more information, refer to [How to Migrate Blocks of Custom Component Libraries](#) ( [ASM User Guide](#))

ASM Vehicle Dynamics

Where to go from here

Information in this section

New Features of ASM Vehicle Dynamics Blockset 4.1	76
Migrating to ASM Vehicle Dynamics Blockset 4.1	76

New Features of ASM Vehicle Dynamics Blockset 4.1

START_BUTTON block	A new START_BUTTON block has been added to the SoftECU subsystem to simulate an engine start button.
SUSCOMP_2D_FRONT	The torque about the y-axis has been added to the compliance calculation.
SUSCOMP_2D_REAR	The torque about the y-axis has been added to the compliance calculation.
DRUM_BRAKE_2D	A new drum brake block modeled as 2-D look-up table has been added.

Migrating to ASM Vehicle Dynamics Blockset 4.1

VEHICLE_MOVEMENT_INFO_CAR	The RotationMatrix_V_to_E[3x3] signal has been added to the signal bus.
SUSCOMP_2D_FRONT	A new inport has been added to the block: Trq_y_Tire.
SUSCOMP_2D_REAR	A new inport has been added to the block: Trq_y_Tire.
SUSCOMP_RIGID_SYM_FRONT	A new inport has been added to the block: Trq_y_Tire.
SUSCOMP_RIGID_SYM_REAR	A new inport has been added to the block: Trq_y_Tire.

SUSPENSION_COMPLIANCE_FRONT

A new inport has been added to the block: Trq_y_Tire.

SUSPENSION_COMPLIANCE_REAR

A new inport has been added to the block: Trq_y_Tire.

SUSPENSION_COMPLIANCE_OPP_FRONT

A new inport has been added to the block: Trq_y_Tire.

SUSPENSION_COMPLIANCE_OPP_REAR

A new inport has been added to the block: Trq_y_Tire.

TIRE_MF

The internal damping at low speed for slip x and for slip y has been reduced.

VEHICLE_MASS_AND_ADDITIONAL_LOADS

Three inports have been added to the block.

- M_Body
- PosVec_CoG_Body[x;y;z]
- Inertia_Body[3x3]

Related topics**Basics**

[Migrating ASM Models \(📖 ASM User Guide\)](#)

AutomationDesk

Where to go from here

Information in this section

New Features of AutomationDesk 6.0	79
Migrating to AutomationDesk 6.0	82

New Features of AutomationDesk 6.0

Where to go from here

Information in this topic

General enhancements	79
Python 3.6 support	79
XIL API 2.1 support	80
64-bit VEOS platform	80
Usability improvements	80
Enhancements to the Signal Editor	80
Enhancements to Python usage in AutomationDesk	80
Enhancements to the libraries	81
XIL API library	81
Enhancements to the COM API	81

General enhancements

The following general enhancements in dSPACE Release 2018-B also affect AutomationDesk.

Python 3.6 support Python scripts that are contained in your AutomationDesk projects are automatically migrated when you open them. For the required manual migration steps and general information on the migration from Python 2.7 to Python 3.6, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33. For enhancements of the Python support in

AutomationDesk, refer to [Enhancements to Python usage in AutomationDesk](#) on page 80.

XIL API 2.1 support The dSPACE XIL API .NET implementation now supports XIL API 2.1.0. AutomationDesk's XIL API library, XIL API Convenience library, the Signal Editor, and the Mapping Editor are based on this implementation, so AutomationDesk also provides XIL API-based enhancements. Refer to [XIL API library](#) on page 81 and [Enhancements to the Signal Editor](#) on page 80.

However, AutomationDesk still supports XIL API 2.0 servers.

The Signal Editor and the Mapping Editor can load files in the ASAM XIL API 2.0.1 formats. After loading, the data is automatically migrated to the ASAM XIL API 2.1.0 formats and is saved in the new formats. To store the data in the older format, you have to select the version in the related Save As dialog.

64-bit VEOS platform AutomationDesk supports the 64-bit VEOS platform.

Usability improvements The following usability improvements in AutomationDesk are available:

- The icons of import and export functions for projects and libraries in AutomationDesk have been changed to achieve a consistent icon representation of these functions in all dSPACE <x>Desk products.
- A new Licenses dialog to display the licenses available for AutomationDesk, and which of them are currently in use.
- If you have specified a value list for a data object, you can now access it via an Exec block and AutomationDesk's COM API. Refer to [Enhancements to the COM API](#) on page 81.
- Reports generated for TestBuilder test cases and test steps now contain hyperlinks to the test step details.

Enhancements to the Signal Editor

The Signal Editor provides the following new features:

- Signal description sets can be parameterized by means of symbolic names.
- The positive and negative timestamps of the start time of a signal value segment is now set according to ASAM XIL API 2.1.0. The time stamps of a signal value segment are automatically synchronized to t=0.0.
- With the **changedpos** and **changedneg** methods, you can detect signal changes in the positive and negative direction. These methods can be configured at the related Watcher object via the Expression Editor.

Enhancements to Python usage in AutomationDesk

The following enhancements for working with Python are available:

- Python interpreter

The Python interpreter is now available as a separate Interpreter pane to facilitate the development of Python scripts. The `_AD_` namespace to be used for code execution in the interpreter is specified by the selected project or library element.
- Namespace options

The variables specified in the script in the Exec or ExecFile block can be explicitly configured as local variables or as global variables. Formerly, variables


that you defined in an Exec or ExecFile block were implicitly configured as global variables. By setting the local namespace for a block, the variable is only valid for this block. In the AutomationDesk Options dialog, you can configure to use the local or the global namespace as the default namespace for all Exec and ExecFile blocks used in all projects.

Enhancements to the libraries

The following library was enhanced:

XIL API library The XIL API library provides the following new data objects introduced with ASAM XIL API 2.1.0:

- **DurationFactory, Duration**
These data objects are used to provide a duration as a string or constant value. The data objects are available in the new Duration subfolder in XIL API/Common.
- **Script, TargetScriptFactory, TargetScriptFileReader**
These data objects are used to instantiate a real-time script to monitor or stimulate variables synchronously to the real-time application running on the target system. The data objects are available in the new Script subfolder in XIL API/Common.
- **ErrorInfo**
This data object is used to provide information on an error. The data object is available in the MetaInfo subfolder in XIL API/Common.
- **CaptureEvent**
This data object is used to get exact information on start and stop timestamps of a capture result. The data object is available in the CaptureResult subfolder in XIL API/Common.
- **ScriptParameterInfo**
This data object is used to get descriptive information on script parameters. The data object is available in the Signal subfolder in XIL API/Common.
- With the **DownloadParameterSets** method of an MAPort data object, you can load a CDFX file, which you created with ControlDesk's Parameter Editor, to set the model variables.
- With the **TriggerClientEvent** method of a Capture data object, you can write the client events to the MDF file.

For more information, refer to [XIL API \(Model Access\)](#) ( [AutomationDesk Accessing Simulation Platforms](#)).

Enhancements to the COM API

The AutomationDesk COM API provides the following enhancements:

- You can get the ValueList of a data object via the **ValueList** property. The **ValueList** property is available for the following data objects in the Main Library:
 - String
 - Int

- Float
- List
- Dict
- Tuple
- The Platform Management is now included in the COM API. For more information on accessing a platform via the API, refer to [dSPACE Platform Management API Reference](#).

For more information, refer to [AutomationDesk Automation](#).

Migrating to AutomationDesk 6.0

General migration aspects

If you open an AutomationDesk project with a later AutomationDesk version, the software automatically detects whether migration is required. Click OK in the message dialog to start the migration. If you also want to continue working with the old project, you must not overwrite it with the migrated project, because the versions are not downward compatible. Save the migrated project to another path or name.

Note

Before you open an older project with the new AutomationDesk version, make sure that the following preconditions are fulfilled:

- You must create backups of the project and of the linked custom libraries.
- AutomationDesk must be running properly. The Log Viewer must not display any error messages.
- The built-in libraries, required custom libraries, and other packages must be loaded properly.
- To import an older project to a new AutomationDesk version, the exported project or custom library must be available in ZIP format. The automatic migration does not support the XML format.

If you use a version control system, there are some preconditions for successful migration. Refer to [How to Migrate Projects Under Version Control](#) ([AutomationDesk Basic Practices](#)).

For more information, refer to [Migrating AutomationDesk](#) ([AutomationDesk Introduction And Overview](#)).

Note

For information on the Python 3.6 migration, refer to [AutomationDesk](#) on page 45.

Libraries

XIL API library With ASAM XIL API 2.0.1, the required states for calling a method were specified, but not implemented. With ASAM XIL API 2.1.0, calling a method in an incompatible state leads to an exception. For example, if you use the `StartSimulation` method of the `MAPort` data object in the `eSIMULATION_RUNNING` state, an exception occurs.

You have to check the exception handling in the related `Exec` and `ExecFile` blocks.

Bus Manager (Stand-Alone)

Where to go from here

Information in this section

Features of the Bus Manager (Stand-Alone) 6.2	85
Migrating to Bus Manager (Stand-Alone) 6.2	88

Features of the Bus Manager (Stand-Alone) 6.2

Generating bus simulation containers without a mapped behavior model

The Bus Manager (stand-alone) now lets you generate bus simulation containers without a mapped behavior model. When you configure bus communication in bus configurations and you work without a behavior model, you can now manually assign each bus configuration to an application process. When you generate bus simulation containers, one BSC file can be generated for each application process to which at least one bus configuration is assigned, regardless of whether the bus configuration is mapped to a behavior model.

For more information, refer to [Generating Bus Simulation Containers \(📖 Bus Manager \(Stand-Alone\) Implementation Guide\)](#).

Enhanced port interface of bus simulation containers

The port interface of bus simulation containers now also includes function ports of bus configurations. Function ports of bus configurations are available at the port interface of BSC files if the following preconditions are fulfilled:

- The function ports are part of a bus configuration that is assigned to an application process for which a bus simulation container is generated.
- Model access is enabled for the function ports, but they are not mapped to model ports.

When you import a BSC file in VEOS Player, the function ports are available as signals. When you import a BSC file in ConfigurationDesk, model port blocks are

created, providing access to the ports of the bus simulation container's port interface.

For more information, refer to [Generating Bus Simulation Containers \(Bus Manager \(Stand-Alone\) Implementation Guide\)](#).

AUTOSAR 4.3.1 support

As of Bus Manager 6.1p1, the Bus Manager (stand-alone) supports AUTOSAR files based on AUTOSAR 4.3.1 as communication matrices.

Support of secure onboard communication (SecOC)

As of Bus Manager 6.1p1, the Bus Manager (stand-alone) supports secure onboard communication according to AUTOSAR.

If the communication matrix specifies secured IPDUs, you can add the secured IPDUs and the related authentic IPDUs to the ConfigurationDesk application and assign them to bus configurations. To implement secure onboard communication in an executable application, you must enable SecOC support for each affected bus configuration and provide the OEM-specific implementation for generating authentication information via user code. When you do this, the Bus Manager can generate the authentication information that is required to secure an authentic IPDU and transmit the authentic IPDU and its related secured IPDU on the bus.

For more information, refer to [Implementing Secure Onboard Communication in Executable Applications \(Bus Manager \(Stand-Alone\) Implementation Guide\)](#).

Support of user code

As of Bus Manager 6.1p1, the Bus Manager (stand-alone) supports user code.

User code lets you implement user-specific algorithms in executable applications. Via the user-specific algorithms, you can add functionality to the Bus Manager, for example, for generating authentication information in secure onboard communication scenarios or calculating checksum or counter signal values. User code must be written in C or C++ and can consist of a source file (C, CPP) and optional include files (H, HPP), such as header files.

For more information, refer to [Working with User Code \(Bus Manager \(Stand-Alone\) Implementation Guide\)](#).

New and enhanced bus configuration features

As of Bus Manager 6.1p1, the Bus Manager (stand-alone) provides the following new bus configuration features:

Bus Configuration Feature	Available for Bus Configuration Part	Purpose
PDU User Code	<ul style="list-style-type: none"> ▪ Simulated ECUs ▪ Inspection 	Lets you apply user code to PDUs. This lets you, for example, write checksum values that are calculated according to user-specific algorithms to ISignals of the related PDUs.
Frame Length	Manipulation	Lets you manipulate the length of CAN frames. You can manipulate the frame length temporarily or permanently, and change the specified settings during run time.

Additionally, the Bus Configuration Enable feature was enhanced and provides an RX enabled only state. This state lets you enable the related bus configuration only in part, i.e., the bus configuration only receives data via the bus but does not transmit data.

For more information, refer to [Working with Bus Configuration Features](#) ([📖 Bus Manager \(Stand-Alone\) Implementation Guide](#)).

Adding ISignal IPDUs and ISignals to communication matrices

As of Bus Manager 6.1p1, the Bus Manager (stand-alone) lets you add ISignal IPDUs and ISignals to communication matrices. You can add the elements only to CAN communication clusters, i.e.:

- Add ISignal IPDUs to CAN channels that are specified in the communication matrix.
- Add ISignals to CAN PDUs that are specified in the communication matrix.

This lets you, for example, configure ISignal IPDUs and ISignals that are not available in the communication matrix yet.

For more information, refer to [Modifying Communication Matrices](#) ([📖 Bus Manager \(Stand-Alone\) Implementation Guide](#)).

Enhanced bus configuration tables

As of Bus Manager 6.1p1, the Bus Manager (stand-alone) provides enhanced bus configuration tables:

- The Bus Simulation Features, Bus Inspection Features, and Bus Manipulation Features tables provide subviews that let you access only the available PDU-related bus configuration features or signal-related bus configuration features, respectively.
- The bus access requests of bus configurations have been removed from the Bus Configurations table. Instead, the bus access requests can now be accessed via the new Bus Access Requests table.

For more information, refer to [Bus Configuration Tables](#) ([📖 Bus Manager \(Stand-Alone\) Implementation Guide](#)).

New bus configuration feature conflicts

As of Bus Manager 6.1p1, the Bus Manager (stand-alone) provides conflicts that can support you in using bus configuration features. For example, conflicts occur if you configure multiple bus configuration features for a bus configuration element and the combination of these bus configuration features is not supported.

For more information, refer to [Handling Bus Manager-Related Conflicts](#) ([📖 Bus Manager \(Stand-Alone\) Implementation Guide](#)).


Migrating to Bus Manager (Stand-Alone) 6.2

Software support discontinuation

Python 2.7 The support of Python 2.7 is discontinued with dSPACE Release 2018-B. Python 3.6 is now supported.

For information on changes and migration aspects of Python scripts in dSPACE products, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33.

Changes to the tool automation interface

Changes have been made to the tool automation interface. Some of these changes affect the data model and can cause code from previous Releases to malfunction. For more information, refer to [Changes to the Automation Interface for Release 2018-B](#) ( [ConfigurationDesk Automating Tool Handling](#)).

ConfigurationDesk

Two variants for different use scenarios

ConfigurationDesk is provided in two variants that are useful for different use scenarios. You can use ConfigurationDesk - Implementation Version to implement real-time applications. You can use ConfigurationDesk - Configuration Version to configure dSPACE RapidPro hardware.

ConfigurationDesk - Implementation Version

Where to go from here

Information in this section

New Features of ConfigurationDesk 6.2 (Implementation Version)	90
Migrating to ConfigurationDesk 6.2	95

New Features of ConfigurationDesk 6.2 (Implementation Version)

Support of delayed tasks

ConfigurationDesk now supports tasks that are delayed in relation to a specific task. Refer to [Configuring Delayed Tasks](#) ([📖 ConfigurationDesk Real-Time Implementation Guide](#)).

Support of SIC files generated with TargetLink

ConfigurationDesk now lets you add SIC files to ConfigurationDesk applications that are generated with TargetLink 4.4 (dSPACE Release 2018-B). However, the following limitations apply:

- SIC files generated with TargetLink are not supported in application processes that have multiple model implementations assigned.
- You cannot use SIC files that are generated with TargetLink for BSC file generation.

Refer to [Basics on Simulink Implementation Containers](#) ([📖 ConfigurationDesk Real-Time Implementation Guide](#)).

Supported SIC file versions

ConfigurationDesk 6.2 supports SIC file versions as listed below:

SIC Files Created With ...	SIC Version
dSPACE Release 2018-B: <ul style="list-style-type: none"> ▪ Model Interface Package for Simulink 4.0 ▪ TargetLink 4.4 	1.5
dSPACE Release 2018-A: <ul style="list-style-type: none"> ▪ Model Interface Package for Simulink 3.6 	1.4
dSPACE Release 2017-B: <ul style="list-style-type: none"> ▪ Model Interface Package for Simulink 3.5 	1.3
dSPACE Release 2017-A: <ul style="list-style-type: none"> ▪ Model Interface Package for Simulink 3.4 	1.2.1

Supported BSC file versions

ConfigurationDesk 6.2 supports BSC files of version 1.5.

Supported V-ECU implementation container versions

ConfigurationDesk 6.2 supports V-ECU implementation container versions as listed below:

V-ECU Implementations Created With...	V-ECU Implementation Version
dSPACE Release 2018-B: <ul style="list-style-type: none"> SystemDesk 5.2 TargetLink 4.4 	2.8 ¹⁾
dSPACE Release 2018-A: <ul style="list-style-type: none"> SystemDesk 5.1 	2.7 ¹⁾
dSPACE Release 2017-B: <ul style="list-style-type: none"> SystemDesk 5.0 TargetLink 4.3 	2.6 ¹⁾
dSPACE Release 2017-A: <ul style="list-style-type: none"> SystemDesk 4.8 	2.5 ¹⁾

¹⁾ There is a migration issue for VEOS if the container file to be imported contains static libraries. For more information, refer to [Migrating from VEOS 4.1 to 4.2](#) ([📖 VEOS Manual](#)).

New function block types

Virtual Ethernet Setup With the Virtual Ethernet Setup function block type, you can configure a virtual Ethernet adapter to access an Ethernet network, including VLAN. The function block works as a provider: Other function blocks can use it to access the configured Ethernet adapter.

For more information, refer to [Virtual Ethernet Setup](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)).

Enhanced function block types

CAN and LIN The CAN and LIN function block types now let you wake up SCALEXIO systems that are based on a DS6001 Processor Board via wake-up frames.

Refer to [CAN](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)) and [LIN](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)).

SENT The SENT function block types now let you receive or transmit data values of multiple SENT messages with each model step. This also makes a more convenient use of serial data protocols possible.

Refer to [Configuring the Basic Functionality \(SENT In\)](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)) and [Configuring the Basic Functionality \(SENT Out\)](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)).

Improved angular processing units (APU) The angular processing units of SCALEXIO systems now support virtual engines with a velocity up to 1,200,000 %/s (200,000 rpm). Furthermore, function blocks can be set to an angle range that is different from that of the APU.

The following function block types now support the improved APU:

- The Angular Setup function block type, refer to [Angular Clock Setup](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)).

- The Current In function block type, refer to [Current In](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)).
- The Voltage In function block type, refer to [Voltage In](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)).
- The Digital Pulse Capture function block type, refer to [Digital Pulse Capture](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)).
- The Current Signal Capture function block type, refer to [Current Signal Capture](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)).
- The Voltage Signal Capture function block type, refer to [Voltage Signal Capture](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)).
- The Wavetable function block types, refer to [Wavetable Generation](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)).
- The Waveform function block types, refer to [Waveform Generation](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)).

The maximum speed that is supported by a function block depends on the assigned hardware resources.

Ethernet Setup The Ethernet Setup function block type now supports the following hardware resources:

- DS6333-CS Automotive Ethernet Board (Ethernet Adapter 2 channel type)
- DS6333-PE Automotive Ethernet Board (Ethernet Adapter 2 channel type)
- DS6334-PE Ethernet Board (Ethernet Adapter 1 channel type)
- DS6335-CS Ethernet Board (Ethernet Adapter 2 channel type)

For more information, refer to [Ethernet Setup](#) ([📖 ConfigurationDesk I/O Function Implementation Guide](#)).

New features of the Bus Manager

Generating bus simulation containers without a mapped behavior model

The Bus Manager now lets you generate bus simulation containers without a mapped behavior model. When you configure bus communication in bus configurations and you work without a behavior model, you can now manually assign each bus configuration to an application process, for example. When you generate bus simulation containers, one BSC file can be generated for each application process to which at least one bus configuration is assigned, regardless of whether the bus configuration is mapped to a behavior model.

For more information, refer to [Generating Bus Simulation Containers](#) ([📖 ConfigurationDesk Bus Manager Implementation Guide](#)).

Enhanced port interface of bus simulation containers The port interface of bus simulation containers now also includes function ports of bus configurations. Function ports of bus configurations are available at the port interface of BSC files if the following preconditions are fulfilled:

- The function ports are part of a bus configuration that is assigned to an application process for which a bus simulation container is generated.
- Model access is enabled for the function ports, but they are not mapped to model ports.

When you import a BSC file in VEOS Player, the function ports are available as signals. When you import a BSC file in ConfigurationDesk, model port blocks are

created, providing access to the ports of the bus simulation container's port interface.

For more information, refer to [Generating Bus Simulation Containers](#) ([📖 ConfigurationDesk Bus Manager Implementation Guide](#)).

AUTOSAR 4.3.1 support As of ConfigurationDesk 6.1p1, the Bus Manager supports AUTOSAR files based on AUTOSAR 4.3.1 as communication matrices.

Support of secure onboard communication (SecOC) As of ConfigurationDesk 6.1p1, the Bus Manager supports secure onboard communication according to AUTOSAR.

If the communication matrix specifies secured IPDUs, you can add the secured IPDUs and the related authentic IPDUs to the ConfigurationDesk application and assign them to bus configurations. To implement secure onboard communication in an executable application, you must enable SecOC support for each affected bus configuration and provide the OEM-specific implementation for generating authentication information via user code. When you do this, the Bus Manager can generate the authentication information that is required to secure an authentic IPDU and transmit the authentic IPDU and its related secured IPDU on the bus.

For more information, refer to [Implementing Secure Onboard Communication in Executable Applications](#) ([📖 ConfigurationDesk Bus Manager Implementation Guide](#)).

Support of user code As of ConfigurationDesk 6.1p1, the Bus Manager supports user code.

User code lets you implement user-specific algorithms in executable applications. Via the user-specific algorithms, you can add functionality to the Bus Manager, for example, for generating authentication information in secure onboard communication scenarios or calculating checksum or counter signal values. User code must be written in C or C++ and can consist of a source file (C, CPP) and optional include files (H, HPP), such as header files.

For more information, refer to [Working with User Code](#) ([📖 ConfigurationDesk Bus Manager Implementation Guide](#)).

New and enhanced bus configuration features As of ConfigurationDesk 6.1p1, the Bus Manager provides the following new bus configuration features:

Bus Configuration Feature	Available for Bus Configuration Part	Purpose
PDU User Code	<ul style="list-style-type: none"> ▪ Simulated ECUs ▪ Inspection 	Lets you apply user code to PDUs. This lets you, for example, write checksum values that are calculated according to user-specific algorithms to ISignals of the related PDUs.
Frame Length	Manipulation	Lets you manipulate the length of CAN frames. You can manipulate the frame length temporarily or permanently, and change the specified settings during run time.

Additionally, the Bus Configuration Enable feature was enhanced and provides an RX enabled only state. This state lets you enable the related bus configuration only in part, i.e., the bus configuration only receives data via the bus but does not transmit data.

For more information, refer to [Working with Bus Configuration Features](#) ([📖 ConfigurationDesk Bus Manager Implementation Guide](#)).

Adding ISignal IPDUs and ISignals to communication matrices As of ConfigurationDesk 6.1p1, the Bus Manager lets you add ISignal IPDUs and ISignals to communication matrices. You can add the elements only to CAN communication clusters, i.e.:

- Add ISignal IPDUs to CAN channels that are specified in the communication matrix.
- Add ISignals to CAN PDUs that are specified in the communication matrix.

This lets you, for example, configure ISignal IPDUs and ISignals that are not available in the communication matrix yet.

For more information, refer to [Modifying Communication Matrices](#) ([📖 ConfigurationDesk Bus Manager Implementation Guide](#)).

Enhanced bus configuration tables As of ConfigurationDesk 6.1p1, the Bus Manager provides enhanced bus configuration tables:

- The Bus Simulation Features, Bus Inspection Features, and Bus Manipulation Features tables provide subviews that let you access only the available PDU-related bus configuration features or signal-related bus configuration features, respectively.
- The bus access requests of bus configurations have been removed from the Bus Configurations table. Instead, the bus access requests can now be accessed via the new Bus Access Requests table.

For more information, refer to [Bus Configuration Tables](#) ([📖 ConfigurationDesk Bus Manager Implementation Guide](#)).

New bus configuration feature conflicts As of ConfigurationDesk 6.1p1, the Bus Manager provides conflicts that can support you in using bus configuration features. For example, conflicts occur if you configure multiple bus configuration features for a bus configuration element and the combination of these bus configuration features is not supported.

For more information, refer to [Handling Bus Manager-Related Conflicts](#) ([📖 Bus Manager \(Stand-Alone\) Implementation Guide](#)).


New features concerning hardware support

ConfigurationDesk supports the following new SCALEXIO hardware:

- **DS6333-CS Automotive Ethernet Board**
The board has a Compact PCI Serial interface. It can be installed in a PCI Express slot of SCALEXIO LabBox (version 2) in combination with the DS6001 Processor Board.
- **DS6333-PE Automotive Ethernet Board**
The board has a PCI Express interface. It can be installed in the Real-Time PC of a SCALEXIO Processing Unit.
- **DS6334-PE Ethernet Board**
The board has a PCI Express interface. It can be installed in the Real-Time PC of a SCALEXIO Processing Unit.

- DS6335-CS Ethernet Board
The board has a Compact PCI Serial interface. It can be installed in a PCI Express slot of SCALEXIO LabBox (version 2) in combination with the DS6001 Processor Board.
- SCALEXIO LabBox (8-slot)
This is a compact extension of the SCALEXIO family that offers space for up to 7 SCALEXIO I/O boards to cover all the essential I/O functions.

New features of the tool automation interface

ConfigurationDesk's automation interface supports additional features of ConfigurationDesk. For more information, refer to [Changes to the Automation Interface for Release 2018-B](#) ( [ConfigurationDesk Automating Tool Handling](#)).

Migrating to ConfigurationDesk 6.2

Discontinuation of Python 2.7

The support of Python 2.7 is discontinued with dSPACE Release 2018-B. Python 3.6 is now supported.

Note

Python scripts that have been added to a ConfigurationDesk project in a previous ConfigurationDesk version via **Insert Script** or **Import Script** are automatically converted to Python 3.6 when you open the project. The script migration cannot be reverted.

For information on changes and migration aspects of Python scripts in dSPACE products, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33.

Discontinuation of SCALEXIO Ethernet Solution

The SCALEXIO Ethernet Solution is discontinued as follows:

- The end-of-life date is January 31, 2021. You can still buy the product up to and including January 31, 2019.
- New Releases of the SCALEXIO Ethernet Solution will still be available for customers with a Software Maintenance Service contract until at least January 31, 2020.
- Customers with a Software Maintenance Service contract who work with dSPACE Release 2018-B will be automatically migrated to the new ConfigurationDesk UDP/TCP function blocks.

For new projects (using dSPACE Release 2018-A and later), we recommend that you use the new UDP/TCP function blocks that are natively integrated in ConfigurationDesk. They provide additional and new options such as IPv6, UDP Multicast support, and enhanced TCP status information.


Note: The dedicated license is required for using the new UDP/TCP function blocks in ConfigurationDesk.

FPGA custom function blocks with APU functionality

As of dSPACE Release 2018-B, the angle range handling of the angular processing unit (APU) has been changed. FPGA custom function blocks that use the APU in the 360° angle range are incompatible if they are built with the FPGA Programming Blockset 3.5 or earlier.

To resolve the incompatibility, use the FPGA model/code of the incompatible FPGA custom function block and build a new FPGA custom function block with the RTI FPGA Programming Blockset 3.6. The RTI FPGA Programming Blockset automatically migrates the framework of the FPGA model/code to the current version.

Changes regarding the Bus Manager

Changes to the tool automation interface Changes have been made to the tool automation interface. Some of these changes affect the data model regarding Bus Manager elements and can cause code from previous Releases to malfunction. For more information, refer to [Changes to the Automation Interface for Release 2018-B](#) ( [ConfigurationDesk Automating Tool Handling](#)).

ControlDesk

Where to go from here

Information in this section

New Features of ControlDesk 6.4	98
Migrating to ControlDesk 6.4	108

New Features of ControlDesk 6.4

Where to go from here

Information in this section



New Features of Platform Management and Platforms/Devices (ControlDesk 6.4)	98
Gives an overview of new features of platform management and platforms/devices for ControlDesk 6.4.	
New Instrument Features (ControlDesk 6.4)	100
Gives an overview of the new instrument features as of ControlDesk 6.4.	
New Calibration and Data Set Management Features (ControlDesk 6.4)	102
Gives an overview of the new data set management features as of ControlDesk 6.4.	
New User Interface Handling Features (ControlDesk 6.4)	102
Gives an overview of the new user interface handling features as of ControlDesk 6.4.	
New Automation Features (ControlDesk 6.4)	103
Gives an overview of the new automation interface features as of ControlDesk 6.4.	
New Bus Navigator Features (ControlDesk 6.4)	104
Gives an overview of the new Bus Navigator features as of ControlDesk 6.4.	
New Electrical Error Simulation Features (ControlDesk 6.4)	105
Gives an overview of the new electrical error simulation features as of ControlDesk 6.4.	
New Signal Editor Features (ControlDesk 6.4)	105
Gives an overview of the new Signal Editor features as of ControlDesk 6.4.	

New Features of Platform Management and Platforms/Devices (ControlDesk 6.4)

New GNSS device providing positioning data from a GNSS receiver

ControlDesk provides the new *GNSS* (*GPS*, *GLONASS*, *Galileo*, ...) device. The device provides positioning data from a GNSS receiver (e.g., a serial GPS mouse) in ControlDesk. It supports various global navigation satellite systems.

For more information, refer to:

- [How to Configure a GNSS Device](#) ( [ControlDesk Platform Management](#))
- [How to Visualize GNSS Positioning Data in a Map](#) ( [ControlDesk Instrument Handling](#))

SCALEXIO: Support of new Ethernet boards

ControlDesk now supports the following Ethernet boards:


- DS6333-CS Automotive Ethernet Board
- DS6333-PE Automotive Ethernet Board
- DS6334-PE Ethernet Board
- DS6335-CS Ethernet Board

SCALEXIO: Support of LabBox (8-slot) and AutoBox (8-slot)

ControlDesk now supports SCALEXIO LabBox (8-slot) and AutoBox (8-slot).

SCALEXIO: Loading an application to the flash memory of SCALEXIO Real-Time PCs

Loading an application to the flash memory now is possible also for SCALEXIO Real-Time PCs.

For instructions, refer to [How to Load an Application to the Flash Memory of dSPACE Real-Time Hardware](#) ( [ControlDesk Platform Management](#))

SCALEXIO: Displaying inter-FPGA connections

ControlDesk now displays the connections between DS2655/DS2656 FPGA Base Boards.


Support of VEOS Ethernet channels for bus monitoring and logging

Ethernet channels on VEOS are now supported as Ethernet interfaces for bus monitoring.

For more information, refer to [Basics on Monitoring, Logging, and Replaying Bus Communication](#) ( [ControlDesk Bus Navigator](#)).


VEOS: Specifying the synchronization behavior for simulation time options

ControlDesk now lets you specify how to synchronize the VEOS simulation time option values, such as the real-time acceleration factor when the VEOS platform goes online. You can either write the values from the experiment platform to the assigned platform, or read them from the assigned platform to the experiment platform.

Refer to [VEOS Simulation Time Options Properties](#) ( [ControlDesk Platform Management](#)).

XIL API MAPort: Support of XIL API version 2.1.0

The XIL API MAPort platform now also supports XIL API version 2.1.0. This includes support for the `SupportAvailableTasks` option, which lets you specify whether the XIL API MAPort implementation supports the `AvailableTasks` feature.

For more information, refer to [Basics on the XIL API MAPort Platform](#) ( [ControlDesk Platform Management](#)).

**Bus monitoring devices:
Support for AUTOSAR 4.3.1
and FIBEX 4.1.2**


ControlDesk now supports:

- AUTOSAR system description files of standard version 4.3.1
- FIBEX files of standard version 4.1.2

Enhanced AUTOSAR/FIBEX support is available for the following devices:

- CAN Bus Monitoring Device
- FlexRay Bus Monitoring Device
- LIN Bus Monitoring Device

This feature was already provided by ControlDesk 6.3p1 as part of the dSPACE Release 2018-A AUTOSAR 4.3 Bus Feature Package.

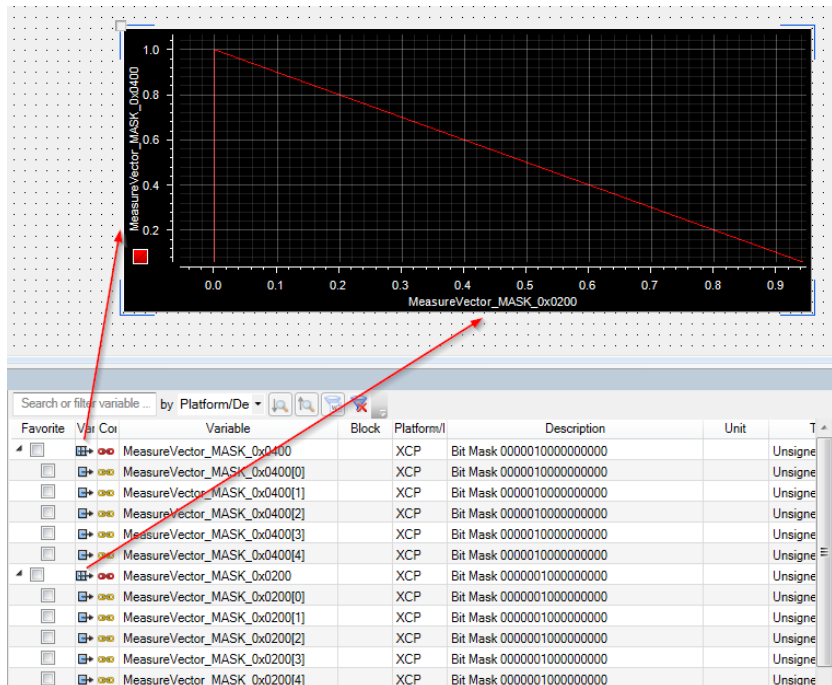
For more information, refer to [Variable Descriptions Supported by ControlDesk](#) ( [ControlDesk Variable Management](#)).

New Instrument Features (ControlDesk 6.4)

**XY Plotter: Visualization of
measurement arrays**

The XY Plotter in ControlDesk 6.4 now supports the visualization of one-dimensional measurement arrays. The measurement arrays connected to the x-axis and y-axis of the XY Plotter must have the same length.

The following illustration shows an example of two one-dimensional measurement arrays each with five array elements visualized in an XY Plotter.

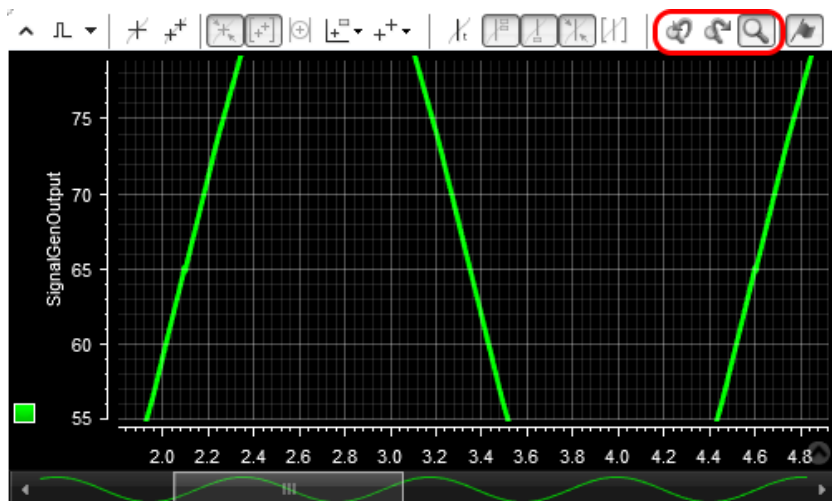


Refer to [Basics of Handling the XY Plotter](#) ([ControlDesk Instrument Handling](#)).



Time Plotter, Index Plotter: Undoing/redoing zoom and move actions

ControlDesk now saves zoom and move actions in the chart view of the Time Plotter and Index Plotter so that you can undo and redo them.

You can undo/redo zoom and move actions via the Cursor Toolbar of the plotter and via keyboard shortcuts. The following illustration shows the Cursor Toolbar of the Time Plotter as an example:




For more information, refer to:

- [Zooming and Moving the Chart \(Time Plotter\)](#) ( [ControlDesk Instrument Handling](#))
- [Zooming and Moving the Chart \(Index Plotter\)](#) ( [ControlDesk Instrument Handling](#))

New Calibration and Data Set Management Features (ControlDesk 6.4)

Creating a new application image based on a data set for SCALEXIO systems

For SCALEXIO systems, ControlDesk now lets you create a new application image based on a data set.



For instructions, refer to [How to Create an Application Image](#) ( [ControlDesk Calibration and Data Set Management](#)).

Support of memory segments and page switching for V-ECUs generated as of VEOS 4.3

The calibration memory segments of V-ECUs generated as of VEOS 4.3 (dSPACE Release 2018-B) are divided into two calibration pages. You can use ControlDesk to switch the active V-ECU calibration page during run time.

During the build process, VEOS also generates an ECU Image file that contains the initial parameter values of the V-ECU application. Adding the ECU Image file to the XCP on Ethernet device that represents the V-ECU in a ControlDesk experiment lets you perform offline calibration on the V-ECU with ControlDesk, for example.

Further reading

- For more information on V-ECU generation as of VEOS 4.3, refer to [New Features of VEOS 4.3](#) on page 259.
- For more information on handling calibration pages with ControlDesk, refer to [Basics on Memory Pages](#) ( [ControlDesk Calibration and Data Set Management](#)) and [How to Activate the Working or Reference Page](#) ( [ControlDesk Calibration and Data Set Management](#)).

New User Interface Handling Features (ControlDesk 6.4)

Displaying license information

ControlDesk now lets you display information on the licenses that are relevant to the current ControlDesk session. This includes information on whether a license is currently *accessible* or being *used*.

The following illustration shows an example:

License Name	Short Name	Status
▲ ControlDesk		
ControlDesk	CONTROLDESK	Used
ControlDesk - Operator Version	CONTROLDESK_OPERATOR	Accessible
ControlDesk Bus Navigator Module	CONTROLDESK_BNV	Accessible
ControlDesk ECU Diagnostics Module	CONTROLDESK_DIAG	Accessible
ControlDesk ECU Interface Module	CONTROLDESK_ECU	Accessible
ControlDesk Signal Editor Module	CONTROLDESK_SE	Accessible
▲ SCALEXIO		
SCALEXIO Failure Simulation (Unlimited Functions)	SCLX_FS_UNLTD	Accessible
SCALEXIO Failure Simulation for 100 Functions	SCLX_FS_100	Accessible
SCALEXIO Failure Simulation for 1000 Functions	SCLX_FS_1000	Accessible
SCALEXIO Failure Simulation for 200 Functions	SCLX_FS_200	Accessible
SCALEXIO Failure Simulation for 300 Functions	SCLX_FS_300	Accessible
SCALEXIO Failure Simulation for 500 Functions	SCLX_FS_500	Accessible

Refer to [Licenses](#) (📖 ControlDesk User Interface Handling).

New Automation Features (ControlDesk 6.4)

Selecting and focusing variables and variable groups in the Variable Browser

The `SelectItems` method of the new `VariableBrowserVariableView / IXaVariableBrowserVariableView <<Interface>>` lets you select a list of variables in the Variable Browser, and focus one of them.

The `SelectGroups` method of the new `VariableBrowserGroupView / IXaVariableBrowserGroupView <<Interface>>` lets you select a list of variable groups in the Variable Browser. Via the `SelectRoot` method, you can select the root node.

For more information, refer to:

- [VariableBrowserVariableView / IXaVariableBrowserVariableView <<Interface>>](#) (📖 ControlDesk Automation)
- [VariableBrowserGroupView / IXaVariableBrowserGroupView <<Interface>>](#) (📖 ControlDesk Automation)

Checking whether reloading the variable description is required

The `ActiveVariableDescription / IXaActiveVariableDescription <<Interface>>` provides the new `CheckSourceForChanges` method, which lets you check whether reloading the active variable description is required.

For more information, refer to [ActiveVariableDescription / IXaActiveVariableDescription <<Interface>>](#) (📖 ControlDesk Automation).

New Bus Navigator Features (ControlDesk 6.4)

Support of VEOS Ethernet channels for bus monitoring and logging


Ethernet channels on VEOS are now supported as Ethernet interfaces for bus monitoring.

For more information, refer to [Basics on Monitoring, Logging, and Replaying Bus Communication](#) ( [ControlDesk Bus Navigator](#)).

Support for partly enabling bus configurations

For bus communication modeled with the Bus Manager, ControlDesk now lets you enable a bus configuration only in part, i.e., the bus configuration only receives data via the bus but does not transmit data.

This feature was already provided by ControlDesk 6.3p1 as part of the dSPACE Release 2018-A AUTOSAR 4.3 Bus Feature Package.





For more information, refer to [Bus Instrument \(Global Layout Type\)](#) ( [ControlDesk Bus Navigator](#)).

Support for PDU user code

For bus communication modeled with the Bus Manager, ControlDesk now lets you display the value that the user code execution returns.

This feature was already provided by ControlDesk 6.3p1 as part of the dSPACE Release 2018-A AUTOSAR 4.3 Bus Feature Package.


For more information, refer to:

- [Bus Instrument \(RX Type for LIN\)](#) ( [ControlDesk Bus Navigator](#))
 - [Bus Instrument \(RX Type for CAN\)](#) ( [ControlDesk Bus Navigator](#))
 - [Bus Instrument \(Inspection Type for CAN\)](#) ( [ControlDesk Bus Navigator](#))
 - [Bus Instrument \(Inspection Type for LIN\)](#) ( [ControlDesk Bus Navigator](#))
-

Support for frame length manipulation

For bus communication modeled with the Bus Manager, ControlDesk now supports the manipulation of the length of a CAN frame. You can manipulate the frame length temporarily or permanently, and change the specified settings at run time.


This feature was already provided by ControlDesk 6.3p1 as part of the dSPACE Release 2018-A AUTOSAR 4.3 Bus Feature Package.

For more information, refer to [Bus Instrument \(Manipulation Type for CAN\)](#) ( [ControlDesk Bus Navigator](#)).

Support of secured IPDUs

For bus communication modeled with the Bus Manager, ControlDesk now supports secured IPDUs. A secured IPDU is an IPDU that secures the payload of another PDU (i.e., authentic IPDU) for secure onboard communication (SecOC).

This feature was already provided by ControlDesk 6.3p1 as part of the dSPACE Release 2018-A AUTOSAR 4.3 Bus Feature Package.

For more information, refer to [Bus Configuration Structure in the Bus Navigator Tree](#) ( [ControlDesk Bus Navigator](#)).

New Electrical Error Simulation Features (ControlDesk 6.4)

Support of XIL API version 2.1.0	Electrical error simulation is based on the ASAM AE XIL API standard version 2.1.0.
---	---

Related topics

Basics

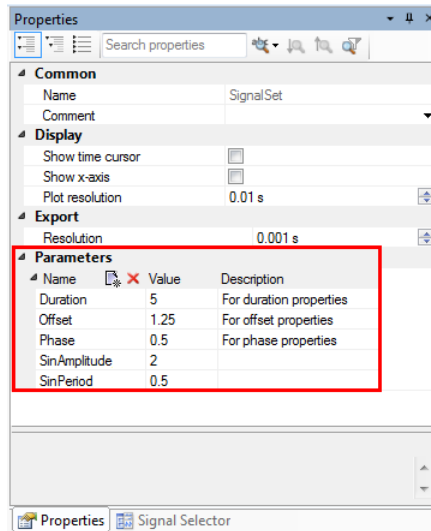
[Compatibility with XIL API](#) ( [ControlDesk Introduction and Overview](#))

New Signal Editor Features (ControlDesk 6.4)

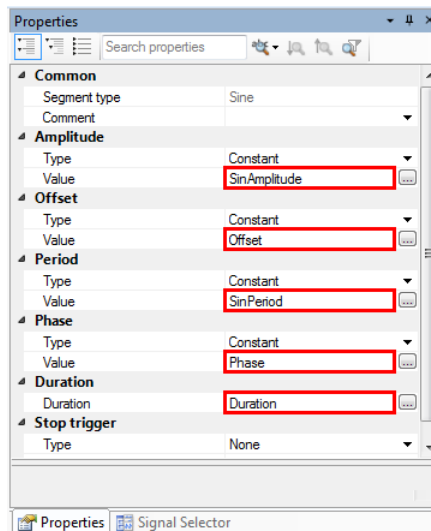
Exporting signal description sets and signal generators to STZ files according to XIL API version 2.1.0	ControlDesk's Signal Editor now exports signal description sets and signal generators to STZ files according to XIL API 2.1. Refer to Export/Export Generator ( ControlDesk Signal Editor).
--	---

Globally parameterizing constant values of a signal description set	According to XIL API 2.1, ControlDesk's Signal Editor now lets you centralize the specification of constant properties via <i>global parameters</i> for all segments of a signal description set. If you modify the value of a global parameter later, all constant properties that use this parameter are changed automatically. This means you do not have to edit them manually.
--	---

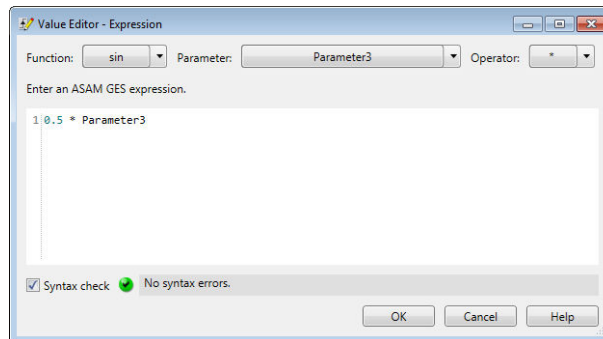
The following illustration shows the specification of global parameters as an example.



The following illustration shows the use of global parameters for constant values as an example.



You can use the Value Editor to edit expressions for constant values according to the ASAM General Expression Syntax (GES).



For more information, refer to [Specifying Constant Properties of Segments via Parameters](#) ( [ControlDesk Signal Editor](#)).

Migrating to ControlDesk 6.4

Where to go from here

Information in this section

Discontinuations in ControlDesk	108
Gives you an overview of the discontinuations in ControlDesk.	
Migrating to ControlDesk 6.4	109
To migrate from ControlDesk 6.3 to ControlDesk 6.4 and reuse existing experiments, you might have to carry out the following migration steps.	

Discontinuations in ControlDesk

Discontinuations in ControlDesk (dSPACE Release 2019-A and later)

Video support As of dSPACE Release 2019-A, ControlDesk no longer supports capturing and monitoring of Video data. The Video Capturing device and the Video Monitoring instrument will no longer be available.

XCP on FlexRay device As of dSPACE Release 2019-A, the XCP on FlexRay device is no longer available.

FlexRay Bus Monitoring device As of dSPACE Release 2019-A, the FlexRay Bus Monitoring device is no longer available.

Data Set Manager As of dSPACE Release 2019-A, the Data Set Manager, ControlDesk's software component to manage the data sets of an experiment, is no longer available.

As a consequence:

- Showing and listing the data sets of the active experiment and displaying their parameter values is no longer possible.
- Creating, comparing and merging data sets is no longer possible.
- Generating data set reports is no longer possible.

However, ControlDesk's support for data sets and calibration page handling will continue.

IDF file support ControlDesk's support for IDF measurement data files will be discontinued:

- As of dSPACE Release 2019-A, recording and exporting files in the IDF format will no longer be supported.
- As of dSPACE Release 2020-A, importing files in the IDF format will no longer be supported.

Use the ASAM MDF 4.1 file format (file name extension: MF4), which is ControlDesk's standard file format for measurement data files.

MDF file (format versions 2.0 and 3.0) import As of dSPACE Release 2019-A, ControlDesk no longer supports the import of MDF measurement data files (MDF file format versions 2.0 and 3.0).

To import measurement data, use the ASAM MDF 4.1 file format (file name extension: MF4), which is ControlDesk's standard file format for measurement data files.

FXF file support As of dSPACE Release 2019-B, ControlDesk no longer supports FXF files to exchange formulas of calculated variables.

However, ControlDesk's support for VXF files to exchange calculated variables and the assigned formulas will continue.

End of software support for discontinued dSPACE hardware For information on the end of software support for discontinued dSPACE hardware, refer to [Discontinuations](#) on page 18.

Migrating to ControlDesk 6.4

Introduction

To migrate from ControlDesk 6.3 to ControlDesk 6.4 and reuse existing experiments, you might have to carry out the following migration steps.

Note

To migrate to ControlDesk 6.4 from versions previous to 6.3, you might also have to perform the migration steps of the intervening ControlDesk versions.

For more information, refer to [Migrating from Prior Versions of ControlDesk](#) ([📖 ControlDesk Introduction and Overview](#)).

Where to go from here

Information in this topic

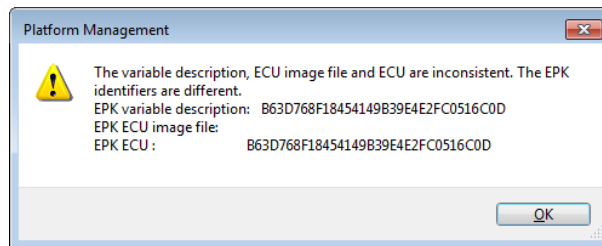
Updating V-ECUs generated with VEOS 4.2 or earlier	109
Double slashes in SDF/TRC variable paths	110
Change to the 'Collect variables from subgroups' option	111
Tool automation changes	111
Migrating from earlier ControlDesk versions	113

Updating V-ECUs generated with VEOS 4.2 or earlier

Support of memory segments and page switching for V-ECUs generated as of VEOS 4.3 The calibration memory segments of V-ECUs generated as of VEOS 4.3 (dSPACE Release 2018-B) are divided into two calibration pages. This lets you switch the active V-ECU calibration page during run time.

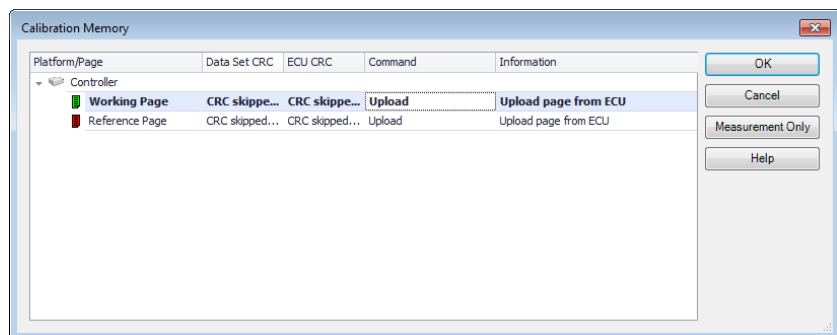
The calibration memory of V-ECUs generated with VEOS 4.2 and earlier was not divided into pages.

Migration when updating an offline simulation application with V-ECUs When you update an offline simulation application (OSA) file containing one or more V-ECUs generated with VEOS 4.2 or earlier with an OSA file generated with VEOS 4.3 or later and replace the related V-ECU A2L variable description files, ControlDesk opens a dialog similar to the following for each V-ECU when you start online calibration. ControlDesk opens this dialog due to the changed page concept of V-ECUs generated with VEOS 4.3 or later.



Click OK to close the dialog for each V-ECU.

ControlDesk then opens the Calibration Memory dialog for you to get identical parameter values in ControlDesk's mirrored memory and on the V-ECU.



Select Upload for both the working and reference page of each V-ECU.

Further reading

- For more information on V-ECU generation as of VEOS 4.3, refer to [New Features of VEOS 4.3](#) on page 259.
- For more information on handling calibration pages with ControlDesk, refer to [Basics on Memory Pages](#) ([ControlDesk Calibration and Data Set Management](#)) and [How to Activate the Working or Reference Page](#) ([ControlDesk Calibration and Data Set Management](#)).

Double slashes in SDF/TRC variable paths

The following migration issue applies to *projects initially created with ControlDesk 5.0 or earlier* regardless of whether they were migrated to later ControlDesk versions.

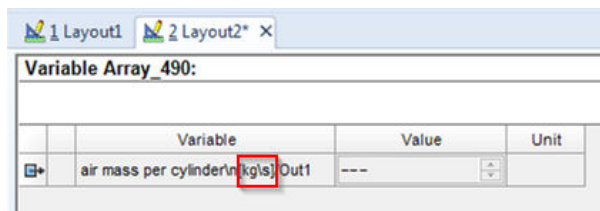
Up to and including ControlDesk 6.3 Up to and including ControlDesk 6.3, double slashes (//) in variable paths of SDF/TRC variable description files in *projects initially created with ControlDesk 5.0 or earlier* were converted to single backslashes (\) in ControlDesk.

As of ControlDesk 6.4 As of ControlDesk 6.4, double slashes are no longer converted, not even in *projects initially created with ControlDesk 5.0 or earlier*. As a result, when you reload or replace the SDF/TRC variable description in a *project initially created with ControlDesk 5.0 or earlier*, ControlDesk cannot reestablish connections to variables with double slashes in the variable path. In this case, you have to reestablish these connections manually. For instructions, refer to [Repairing Variable Connections](#) ([ControlDesk Instrument Handling](#)).

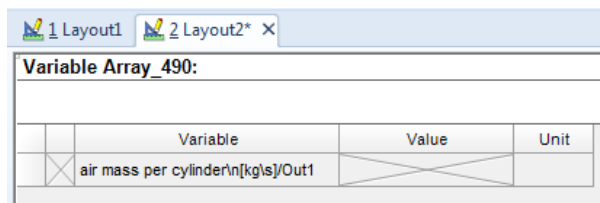
Example As an example, the following line from a TRC file shows a variable block group declaration with a double slash:

```
group "air mass per cylinder\n[kg//s]" -- block-group
```

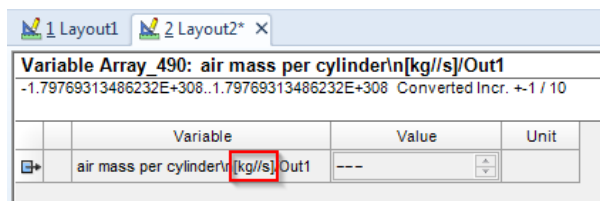
Up to and including ControlDesk 6.3, double slashes were converted to single backslashes as shown in the following illustration:



When you reload the variable description in ControlDesk 6.4 or later, ControlDesk cannot reestablish the connections of instruments to the related variables as shown in the following illustration.



The following illustration shows the variable connection after it was reestablished manually.



Change to the 'Collect variables from subgroups' option

In ControlDesk 6.4, the Collect variables from subgroups option was changed. The 'Filled' state is no longer available.

For more information, refer to [Variables Page](#) ([ControlDesk Variable Management](#)).

Tool automation changes

Migrating to Python 3.6 As of dSPACE Release 2018-B, dSPACE products no longer support Python 2.7. Instead, Python 3.6 is supported.

The following table explains the ControlDesk-specific migration aspects:

Script Type	Migration
Scripts added to a project, and event handler code in the 'Project' event context	<p>To automatically let project script code and event handler code in the 'Project' event context be adapted to Python 3.6, open the related project in ControlDesk 6.4 or later.¹⁾</p> <p>Note</p> <ul style="list-style-type: none"> ▪ Project script code and event handler code in the 'Project' event context of closed projects is not adapted to Python 3.6. ▪ Automatic script code adaptation to Python 3.6 does not guarantee script runnability. You have to check script runnability yourself.
Scripts added to an experiment, and event handler code in the 'Experiment' event context	<p>To automatically let experiment script code and event handler code in the 'Experiment' event context be adapted to Python 3.6:</p> <ol style="list-style-type: none"> 1. Open the related project in ControlDesk 6.4 or later.¹⁾ 2. Activate the related experiment. <p>Note</p> <ul style="list-style-type: none"> ▪ Experiment script code and event handler code in the 'Experiment' event context of experiments that have not been activated at least once is not adapted to Python 3.6. ▪ Automatic script code adaptation to Python 3.6 does not guarantee script runnability. You have to check script runnability yourself.
Scripts added to an instrument or layout	<p>To automatically let script code added to an instrument or layout be adapted to Python 3.6:</p> <ol style="list-style-type: none"> 1. Open the related project in ControlDesk 6.4 or later.¹⁾ 2. Activate the related experiment. 3. Open the related layout in ControlDesk 6.4 or later.²⁾ <p>Note</p> <ul style="list-style-type: none"> ▪ Instrument and layout script code of layouts that have not been opened at least once is not adapted to Python 3.6. ▪ Automatic script code adaptation to Python 3.6 does not guarantee script runnability. You have to check script runnability yourself.
Scripts added to instruments in an instrument library	<p>To automatically migrate scripts of instruments in an instrument library:</p> <ol style="list-style-type: none"> 1. If the instruments are contained in an instrument library (ILX) file, import the file to ControlDesk's Instrument Selector. 2. Place the instruments on a layout. 3. Add the migrated and tested instruments to the instrument library to update it. <p>Note</p> <p>Automatic script code adaptation to Python 3.6 does not guarantee script runnability. You have to check script runnability yourself.</p>

Script Type	Migration
Custom scripts: <ul style="list-style-type: none"> ▪ Extension scripts ▪ User function scripts ▪ Event handler code in the 'User' event context³⁾ ▪ Custom libraries 	To manually migrate custom scripts, refer to Migrating Python Scripts from Python 2.7 to Python 3.6 on page 33.

¹⁾ This involves automatic project migration.

²⁾ This involves automatic layout migration.

³⁾ Migration is necessary only if the event handler code was imported from ControlDesk 6.3 or earlier.

For more information on the differences between the two Python versions and for general migration information, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33.

Change to the IXaVariablesManagement interface In ControlDesk 6.4, the `VariablesManagement / IXaVariablesManagement <<Interface>>` changes due to the change to the Collect variables from subgroups option. As of ControlDesk 6.4, the `CollectVariablesEnabled` property of this interface returns a Boolean value. Up to and including ControlDesk 6.3, a `CollectVariablesFromSubgroupsBehavior` enumeration value was returned by the `CollectVariablesEnabled` property.

Refer to [VariablesManagement / IXaVariablesManagement <<Interface>>](#) ([📖 ControlDesk Automation](#)).

Change to the IXaCalibrationManagement interface In ControlDesk 6.4, the `StartOnlineCalibration` method of the `CalibrationManagement / IXaCalibrationManagement <<Interface>>` changes.

As of ControlDesk 6.4, the method returns the new enumeration value `NoChange` if online calibration was started before the method was called.

Refer to [CalibrationManagement / IXaCalibrationManagement <<Interface>>](#) ([📖 ControlDesk Automation](#)).

Change to the IXaMultiScalingEntry interface In ControlDesk 6.4, the `Remove` method of the `MultiScalingEntry / IXaMultiScalingEntry <<Interface>>` was removed.

Refer to [MultiScalingEntry / IXaMultiScalingEntry <<Interface>>](#) ([📖 ControlDesk Automation](#)).

Migrating from earlier ControlDesk versions

To migrate from earlier ControlDesk versions and reuse existing experiments, you might have to carry out additional migration steps. For more information on the migration steps, refer to [Migrating from Prior Versions of ControlDesk](#) ([📖 ControlDesk Introduction and Overview](#)).

Related topics

Basics

- [Basics on Migrating from Prior Versions of ControlDesk \(📖 ControlDesk Introduction and Overview\)](#)
- [Repairing Variable Connections \(📖 ControlDesk Instrument Handling\)](#)

References

- [Replace Variable Description \(📖 ControlDesk Variable Management\)](#)

DCI Configuration Tool

New Features of the DCI Configuration Tool 3.10

Firmware versions for DCI-GSI1 and DCI-GSI2 interfaces

The following firmware versions for the DCI-GSI1 and DCI-GSI2 interfaces are delivered with the DCI Configuration Tool 3.10:

- DCI-GSI1 firmware version 1.6.8
- DCI-GSI2 firmware version 1.4.12

Note

The firmware version delivered with the DCI Configuration Tool is not always the latest firmware version available. If you encounter any problems, contact dSPACE Support to check if a later firmware version is available.

dSPACE FlexRay Configuration Package

New Features of dSPACE FlexRay Configuration Package 4.2

FlexRay Configuration Package

Support of container and contained IPDUs The FlexRay Configuration Package supports *container IPDUs*. A container IPDU is an IPDU that contains one or several smaller IPDUs (i.e., *contained IPDUs*). The contained IPDUs can be ISignal IPDUs or multiplexed IPDUs, for example.

Refer to [Aspects of Miscellaneous Supported AUTOSAR Features](#) ( [FlexRay Configuration Tool Guide](#)).

This feature has already been available since FlexRay Configuration Package 4.1p1.

Support of secure onboard communication The FlexRay Configuration Package supports *secure onboard communication (SecOC)* according to AUTOSAR. Secure onboard communication provides authentication mechanisms to identify PDU data that was modified or transmitted without authorization, e.g., due to a replay attack. To implement secure onboard communication, you must enable SecOC support for the project and provide the OEM-specific implementation for generating authentication information via user code.

Refer to [Aspects of Miscellaneous Supported AUTOSAR Features](#) ( [FlexRay Configuration Tool Guide](#)).

This feature has already been available since FlexRay Configuration Package 4.1p1.

Support of global time synchronization The FlexRay Configuration Package supports *global time synchronization (GTS)* according to AUTOSAR. Global time synchronization means providing and distributing synchronized times in the vehicle across all ECUs.

Refer to [Aspects of Miscellaneous Supported AUTOSAR Features](#) ( [FlexRay Configuration Tool Guide](#)).

This feature has already been available since FlexRay Configuration Package 4.1p1.

The *RTI Synchronized Time Base Manager Blockset* implements global time synchronization (GTS) on dSPACE systems by using the dSPACE ECU time base

manager (DsEcuTbM) contained in the RTLib of SCALEXIO and other supported hardware. The blockset provides the following tasks, among others:

- Creating and configuring synchronized time base instances.
- Reading the synchronized time information from the synchronized time base instances, and providing the time and status of a time base instance.
- Simulating global time masters.

For more information on the RTI Synchronized Time Base Manager Blockset, refer to its documentation, which you can open by clicking the Help button in the block library.

The RTI Synchronized Time Base Manager Blockset can already be used with FlexRay Configuration Package 4.1p1.

Support of AUTOSAR E2E protection profiles 05 and 06 The FlexRay Configuration Package now also supports end-to-end communication protection (E2E protection) according to the AUTOSAR end-to-end protection profiles 05 and 06 as described in AUTOSAR 4.2.1.

Refer to [Aspects of Miscellaneous Supported AUTOSAR Features](#) ( [FlexRay Configuration Tool Guide](#)).

This feature has already been available since FlexRay Configuration Package 4.1p1.

FlexRay Configuration Tool

Support of AUTOSAR System Template 4.3.0 and 4.3.1 The FlexRay Configuration Tool now also supports the AUTOSAR System Template based on AUTOSAR Release 4.3.0 and 4.3.1 for describing FlexRay networks.

This feature has already been available since FlexRay Configuration Package 4.1p1.

Support of FIBEX 4.1.2 The FlexRay Configuration Tool now also supports FIBEX 4.1.2 files for describing FlexRay networks.

This feature has already been available since FlexRay Configuration Package 4.1p1.

dSPACE Python Extensions

New Features of dSPACE Python Extensions 3.0

New features

Python Extensions 3.0 has no new features, but the Test Automation Python modules have been migrated to Python 3.6.

Python 3.6 support

For information on the new features and general migration instructions, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33.

The `matlablib2` and `rs232lib2` modules provided by dSPACE Python Extensions only work in Python 3.6.

You must manually migrate all custom scripts to Python 3.6 according to the migration information. Refer to [General Information on Migrating](#) on page 38.

dSPACE XIL API .NET

Where to go from here

Information in this section

New Features of dSPACE XIL API .NET 2018-B	121
Migrating to dSPACE XIL API .NET 2018-B	123

New Features of dSPACE XIL API .NET 2018-B

Python 3.6 support

The dSPACE XIL API Python demos are migrated to Python 3.6.

You must manually migrate all custom scripts to Python 3.6 according to the migration information. Refer to [General Information on Migrating](#) on page 38.

XIL API 2.1.0 support

The dSPACE XIL API .NET implementation is now based on the ASAM XIL API 2.1.0 standard. This implies some new features, but also changes in the dSPACE implementation.

Enhanced MAPort functionality

The following features of ASAM XIL API 2.1.0 are supported by dSPACE XIL API:

- New `PauseSimulation` method to pause a simulation.
The method is only available for:
 - PHS bus systems based on DS1005 or DS1006
 - DS1103
 - DS1104
 - MicroAutoBox
 - VEOS
- New `Script` and `TargetScript` interfaces to directly load a Real-Time Testing sequence for stimulation to the real-time target.

- You can define parameters for a signal description set. With these parameters, you can specify the value of a signal segment according to the syntax of `ConstSymbol` expressions.
- Parameter values that are contained in parameter set files can be downloaded to the simulation tool or hardware via the `DownloadParameterSets` method. The dSPACE XIL API implementation only supports CDFX files that you created with ControlDesk's Parameter Editor.
- User-defined events of `eCLIENTEVENT` type can be generated by using the `TriggerClientEvent` method of a Capture object.
- Model paths can be checked via the `CheckVariableNames` method.
- With the new `AvailableTasks` property of the `VariableInfo` object, you can get the tasks in which the current variable can be measured.
- The `ConditionWatcher` element now provides the `Defines` property in an STI file.
- The General Expression Syntax has been enhanced by the new `changedpos` and `changedneg` methods to detect value changes as well as the `INF` constant to specify the limits of a data file segment.
- In XIL API 2.1.0, some methods and properties are mentioned as deprecated. You are recommended to replace them with newer methods and properties.
- With the `KeepCompleteCaptureResult` setting in the MAPort configuration, you can avoid storage problems in certain use cases when you use a `CaptureResultMemoryWriter` element.

There are new demo projects for the new features. Use the Copy XIL API .NET Demos shortcut in the Windows Start menu to access the demos.

For more information, refer to the ASAM XIL API documentation and to [Implementing an MAPort Application](#) ([📖 dSPACE XIL API Implementation Guide](#)).

Enhanced EESPort functionality

The ASAM interfaces now provide the `SignalInfo` and `PotentialInfo` properties. Casting the EESPort object to the dSPACE-specific EESPort is therefore no longer required.

For more information, refer to [Migrating to dSPACE XIL API .NET 2018-B](#) on page 123 and [Implementing an EESPort Client Application](#) ([📖 dSPACE XIL API Implementation Guide](#)).

Changes in the PlatformManagement API

Enhanced platform support The dSPACE Platform Management API and the XIL API (MAPort) now provide:

- Support of VEOS as a 64-bit platform.

For more information, refer to [Basics on the Platform Management API](#) ([📖 dSPACE Platform Management API Reference](#)) and [📖 dSPACE XIL API Implementation Guide](#).

Python 3.6 support The dSPACE Platform Management API now works with Python 3.6. There is no support of an automatic migration of your scripts. You

have to migrate custom scripts manually. For more information, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33.

Migrating to dSPACE XIL API .NET 2018-B

Migrating applications from dSPACE XIL API .NET 2018-A and earlier to dSPACE XIL API .NET 2018-B

With dSPACE Release 2018-B, XIL API support changed from ASAM XIL API 2.0.1 to ASAM XIL API 2.1.0. You have to observe the following migration instructions.

Migrating MAPort applications

There are some behavior changes and new features ASAM XIL API 2.1.0. Some of the new features are recommended to be used instead of the older deprecated features.

Applications based on XIL API 2.0.1 are compatible with applications based on XIL API 2.1.0.

Behavior changes You have to migrate test scripts due to the following behavior changes:

- States are observed
With XIL API 2.0.1, the required states for calling a method were specified, but not implemented. With XIL API 2.1.0, calling a method in an incompatible state leads to an exception. For example, calling `StartSimulation` in the `eSIMULATION_RUNNING` state leads to an exception.
- Defined overwrite mode
Writer objects, such as the `SignalGeneratorWriter`, now use the specified overwrite mode. Previously, an existing file was always overwritten.
- Synchronization of `SignalValueSegments`
`SignalValueSegments` that start with a positive or negative offset are now automatically synchronized to `t=0.0`.
This also affects `CreateSignalValue` and `CreateSignalGroupValue`.

Deprecated elements The following table shows you the deprecated elements and the elements you can use for replacement.

Affected Component	Deprecated Element	Replacement
Testbench.Common.Symbol		
SymbolFactory	Method: <code>CreateConstSymbolByValue</code>	Method: <code>CreateConstSymbolByExpression</code>
ConstSymbol	Property: <code>Value</code>	Property: <code>Expression</code>

Affected Component	Deprecated Element	Replacement
Testbench.Common.WatcherHandling		
WatcherFactory	Method: CreateDurationWatcher	Methods: CreateDurationWatcherByTimeSpan or CreateDurationWatcherByCycleNumber
DurationWatcher	Property: Duration	Property: Duration2 The property value cannot be changed. Create a new DurationWatcher for a different duration value.
Testbench.Common.Capturing		
Capture	Method: SetStartTriggerCondition Method: SetStopTriggerCondition Property: DurationUnit	Method: SetStartTrigger Method: SetStopTrigger Implicitly set by passing a TimeSpanDuration or CycleNumberDuration object from the Testbench.Common.Duration package to the methods of SetStartTrigger and SetStopTrigger .
Testbench.Common.Capturing.Enum		
DurationUnit	Enumeration: DurationUnit	Implicit representation by TimeSpanDuration and CycleNumberDuration in the Testbench.Common.Duration package.
Testbench.Common.SignalGenerator		
SignalGenerator	Property: State	Property: ScriptState inherited from the Script interface in the Testbench.Common package. SignalGenerator is derived from Script .
Testbench.Common.SignalGenerator.Enum		
SignalGeneratorState	Enumeration: SignalGeneratorState	Enumeration: ScriptState in the Testbench.Common.Script.Enum package.

Migrating EESPort applications

You have to migrate only the source code that is used for handling the custom properties, i.e., the **SignalInfo** property and the **PotentialInfo** property.

dSPACE EESPort 2018-A	dSPACE EESPort 2018-B
<pre> dsEESPort = IDSEESPort(eesport) ... # Iterate through all the signals in EESPort for signal in dsEESPort.Signals: dsConfiguration.Signals.Add(signal.Name, signal.ECUName, signal.PinName) ... # Iterate through all the potentials in EESPort for potential in dsEESPort.Potentials: dsConfiguration.Potentials.Add(potential.Name, PotentialType.Potential) ... </pre>	<pre> # Iterate through all the signals in EESPort for signal in eesPort.SignalInfos: dsConfiguration.Signals.Add(signal.Name, signal.ECUName, signal.PinName) ... # Iterate through all the potentials in EESPort for potential in eesPort.PotentialInfos: dsConfiguration.Potentials.Add(potential.Name, PotentialType.Potential) ... </pre>

ECU Interface Manager

Where to go from here

Information in this section

New Features of ECU Interface Manager 2.4	127
An overview of the new features of ECU Interface Manager 2.4.	
Compatibility of ECU Interface Manager 2.4	128
Provides information on the compatibility of ECU Interface Manager 2.4.	
Migrating to ECU Interface Manager 2.4	128
Information on how to migrate to ECU Interface Manager 2.4.	

New Features of ECU Interface Manager 2.4

Improved variable-based ECU-synchronous data access

In connection with the ECU-synchronous data access to ECU variables, the ECU Interface Manager 2.4 now supports 1,000 scalar variables or arrays with an overall amount of 3,000 array elements per data access.

Up to and including ECU Interface Manager 2.3, ECU-synchronous data access to ECU variables was limited to 255 variables per data access.

Refer to [Basics on Configuring Data Accesses](#) ( [ECU Interface Manager Manual](#)).

Support of static DAQ lists

The ECU Interface Manager now supports static DAQ lists.

Related topics

Basics

[Basics on Configuring Data Accesses](#) ( [ECU Interface Manager Manual](#))

Compatibility of ECU Interface Manager 2.4

Compatibility in general

dSPACE recommends using only software products from the same dSPACE Release. This ensures maximum run-time compatibility.

Compatibility between EIC files and ConfigurationDesk

The following table shows the compatibility between EIC files and ConfigurationDesk:

	EIC Files Created With ...				
	ECU Interface Manager 2.0p1 ¹⁾	ECU Interface Manager 2.1 ²⁾	ECU Interface Manager 2.2 ³⁾	ECU Interface Manager 2.3 ⁴⁾	ECU Interface Manager 2.4 ⁵⁾
ConfigurationDesk 6.2 ⁵⁾	✓	✓	✓	✓	✓
ConfigurationDesk 6.1 ⁴⁾	✓	✓	✓	✓	—
ConfigurationDesk 6.0 ³⁾	✓	✓	✓	—	—
ConfigurationDesk 5.7 ²⁾	✓	✓	—	—	—
ConfigurationDesk 5.6 SP1 ¹⁾	✓	✓	—	—	—

¹⁾ dSPACE Release 2016-B

²⁾ dSPACE Release 2017-A

³⁾ dSPACE Release 2017-B

⁴⁾ dSPACE Release 2018-A

⁵⁾ dSPACE Release 2018-B

Migrating to ECU Interface Manager 2.4

Automatic migration of projects

You can reuse projects in ECU Interface Manager 2.4 if the projects were last saved with ECU Interface Manager 2.0 p1 or later.

When you open the projects in ECU Interface Manager 2.4, they are migrated automatically.

Note

In ECU Interface Manager 2.4, you cannot reuse projects that were last saved with ECU Interface Manager 2.0 or earlier.

Additional migration steps in some cases

To migrate to ECU Interface Manager 2.4 from versions earlier than ECU Interface Manager 2.2, you might also have to perform the migration steps of the intervening ECU Interface Manager versions.

Firmware Manager

New Features of Firmware Manager 2.6

Enhanced platform support

The Firmware Manager supports updating the firmware of the following SCALEXIO hardware:

- LabBox (8-slot)

Model Compare

Where to go from here

Information in this section

New Features of Model Compare 2.9	133
Migration to Model Compare 2.9	135




New Features of Model Compare 2.9

New Comfort Copy commands to resolve conflicts automatically

A three-way analysis of the reference, the comparison and the common ancestor models not only detects the differences between the comparison and the reference model. If model elements or properties in the common ancestor model are modified differently in the reference model and in the comparison model, they differ in all three models. For these conflicts, Model Compare now provides enhanced Comfort Copy commands that let you resolve the conflicts automatically.

For this conflict resolution, you can now decide whether the comparison model or the reference model is treated as the master for model merging. If you specify the source model to be the master, the conflicts are resolved by copying the elements to the target model. If you specify the target model to be the master, the conflicts remain unchanged and their review state is set to **Approved**, which is marked in the Model Navigator.

Related documentation

- [Basics on Merging Models After a Three-Way Analysis](#) ( Model Compare Guide)
- [How to Merge a Node's Model Elements at Once](#) ( Model Compare Guide)
- [Comfort Copy to Right with Comparison as Master](#) ( Model Compare Reference)

- [Comfort Copy to Right with Reference as Master](#) (📖 Model Compare Reference)
- [Comfort Copy to Left with Comparison as Master](#) (📖 Model Compare Reference)
- [Comfort Copy to Left with Reference as Master](#) (📖 Model Compare Reference)

Mark remaining differences as approved

If you accept differing elements or conflicts, you can now mark the related element pair as approved by setting its review state to **Approved**. This can be undone by resetting the review state.

Each change of the review state is documented by an entry in the elements' comments, including the user name and a time stamp.

In the **Model Navigator**, approved elements are marked with a check mark (✓). In reports, they are marked as **Approved**.

You can filter the elements displayed in the **Model Navigator** to hide all approved elements or to show only approved elements.

Related documentation

- [How to Mark an Element Pair as Approved](#) (📖 Model Compare Guide)
- [Set Review State to Approved](#) (📖 Model Compare Reference)
- [Clear Review State](#) (📖 Model Compare Reference)

Create a new comparison session straight from the Simulink/Stateflow Editor

You can start a new Model Compare session from the MATLAB Simulink Editor and the MATLAB Stateflow Editor via an additional Model Compare menu. In the opened **New Session** dialog, the reference model is preset to the current model and the MATLAB version is adjusted. In the input fields, browsing for model files and initialization files starts with the current model folder.

Related documentation

- [How to Create Comparison Sessions](#) (📖 Model Compare Guide)
- [New Session](#) (📖 Model Compare Reference)

Manual alignment

In the **Model Navigator**, you can now manually align two nodes that correspond to each other in the reference and in the comparison model, but whose relationship cannot be detected by an algorithm. Alignments are saved with the Model Compare session, so they persist even when models are dumped again and sessions are opened again.

You can manually align two blocks or subsystems, two annotations, or two Stateflow objects of the same type.

Related documentation

- [How to Align Two Nodes Manually](#) (📖 Model Compare Guide)
- [Align Node With](#) (📖 Model Compare Reference)
- [Remove Alignment](#) (📖 Model Compare Reference)

Further improvements

Furthermore, Model Compare 2.9 provides the following improvements:

- Differing properties are now represented in the same color in reports and in the Property Inspector.
- The line handling after merging is improved.
- The performance of highlighting elements in large models is improved.

Migration to Model Compare 2.9

No adaptation necessary

You can migrate from Model Compare 2.8 to Model Compare 2.9 without adaptations.

ModelDesk

Where to go from here

Information in this section

New Features of ModelDesk 5.0	137
Migration to ModelDesk 5.0	139

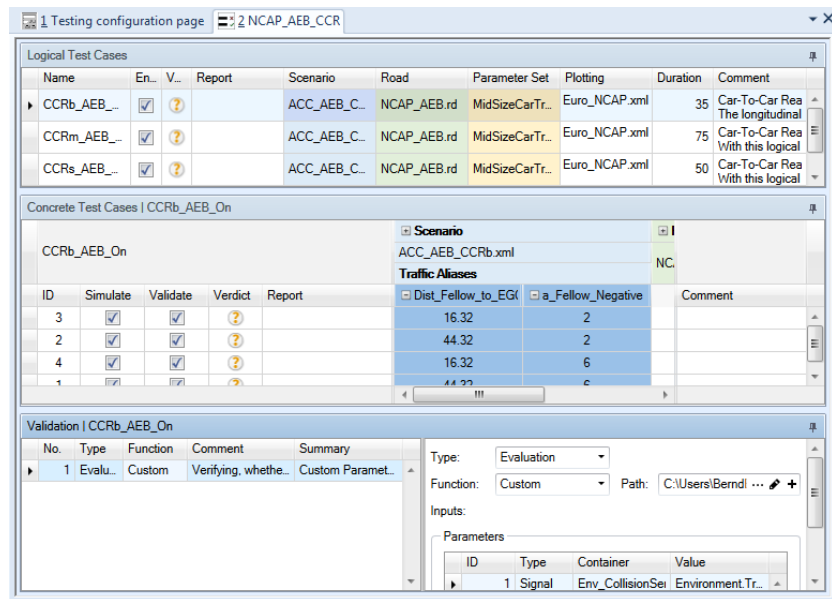
New Features of ModelDesk 5.0

ModelDesk Testing

ModelDesk Testing is a test environment integrated in ModelDesk.

It provides the following features:

- Testing based on ASM.
- Supporting all simulation platforms (Simulink, VEOS, and real-time platforms).
- Graphical user interface to specify the test cases.
- Starting the simulation of all the specified test cases.
- Validating and reporting the results of the test cases.



Project management

Shortcuts to MotionDesk and ControlDesk You can start MotionDesk and ControlDesk via shortcuts in ModelDesk. The projects that are related to the ModelDesk project can be specified so that they are automatically loaded.

Tool automation You can specify the model configuration via tool automation.

Road creation

Junction ModelDesk can adapt the height profile of the junction to the connected road elements. You can influence the junction's height profile by specifying the position, height, and slope of the reference point.

OpenDRIVE import and export When OpenDRIVE files are imported or exported, the height profile of junctions is considered.

Improved support of complex commercial OpenDRIVE network files
When roads are exported in the OpenDRIVE format, the dimensions of the traffic objects are considered.

Tool automation You can create a new road network file via tool automation.

Scenario creation

Alias of route index You can specify an alias variable for the route index.

Tool automation You can create a new scenario file via tool automation.

Parameterization

Custom blocks The multi-instance feature also works for custom blocks. You can use multiple custom blocks of the same type in one model.

Removing unused parameter sets You can remove unused parameter sets from the Pool section of the ModelDesk project.

Related topics**Basics**

[Features of ModelDesk Testing](#) (📖 ModelDesk Testing)

Migration to ModelDesk 5.0

Maneuver Editor

As of ModelDesk 4.7, the Maneuver Editor is obsolete. You can specify maneuvers using the Scenario Editor. Maneuvers specified with the Maneuver Editor are automatically migrated to scenarios for the Scenario Editor. However, scripts that use the tool automation of the Maneuver Editor cannot be migrated. If you want to use such scripts, you must activate the Maneuver Editor by using the Maneuver Compatibility command.

Note

When you enable maneuver compatibility, manual modifications of the simulation model might be required to enable proper usage of the maneuver definition of the Maneuver Editor.

Tool automation for plotting

As of ModelDesk 4.4, ModelDesk has new plotters, and the tool automation for plotting has been changed. To reuse scripts for plotting, you must adapt scripts written for ModelDesk 4.3 and earlier.

Triggering of plots

As of ModelDesk 4.6, plotting is triggered by the simulation model. Previously, ModelDesk triggered plotting. The plots are usually identical but can differ in some cases.

Tip

To compare measurements, it is useful to use the XY Plotter and use the maneuver time as a signal for the x-axis.

Related topics**References**

[Maneuver Compatibility](#) (📖 ModelDesk Scenario Creation)

Model Interface Package for Simulink

Where to go from here

Information in this section

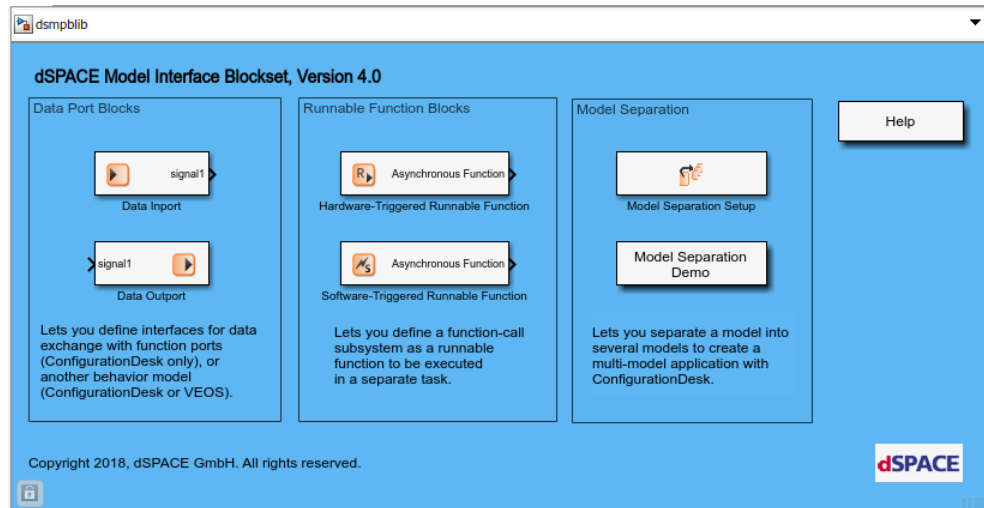
New Features of the Model Interface Package for Simulink 4.0 141

Migration Aspects of the Model Interface Package for Simulink 145

New Features of the Model Interface Package for Simulink 4.0

Redesigned block library

For Model Interface Package for Simulink 4.0, the Model Port Block Library and the Model Separation Block Library were merged to the new Model Interface Blockset. The Model Interface Blockset contains all the model port blocks that you need to model the interface of behavior models for use in ConfigurationDesk and VEOS Player. It also provides redesigned runnable function blocks for exporting function-call subsystems as runnable functions, and access to the Model Separation Setup. Refer to the following illustration:



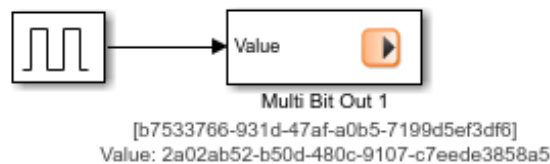
Accessing the Model Separation Setup In addition to the Model Separation Setup block in the Model Interface Blockset, you can access the Model Separation Setup via the new Open Model Separation Setup in the Model Port Blocks menu.

Handling model port block and signal IDs

With Model Interface Package 4.0, the handling of model port block and signal IDs has been improved. For creating behavior model variants with an identical model interface, you can now use the standard Copy command together with the new Paste and Keep IDs command for Data Inport and Data Outputport blocks. The Paste and Keep IDs command lets you create model port blocks with identical IDs. This is useful for the following use cases:

- Creating variants of behavior models.
- Creating variants of behavior model parts.

Viewing and changing block IDs For a better overview, the model port block and signal IDs are displayed in the related Data Inport and Data Outputport block dialogs. In addition, the block dialogs provide the Display signal and block IDs checkbox. If you select this checkbox, the IDs of the related model port block and its signals are displayed in the behavior model.



Using custom libraries You can now copy model port blocks, including their IDs, from a behavior model to a custom library and vice versa.

For more information, refer to [Basics on Model Port Block IDs and Signal IDs](#) ([Model Interface Package for Simulink - Modeling Guide](#)).

Redesign of Runnable Function blocks

For a better overview, the Model Interface Package for Simulink now provides two blocks for exporting function-call subsystems as runnable functions:

- **Hardware-Triggered Runnable Function block:**

The Hardware-Triggered Runnable Function block lets you export a function-call subsystem as a runnable function for use in ConfigurationDesk. You can use this runnable function for modeling asynchronous tasks in ConfigurationDesk.



- **Software-Triggered Runnable Function block:**






The Software-Triggered Runnable Function block exports a function-call subsystem as predefined task with an assigned runnable function for use in ConfigurationDesk or VEOS Player.

For more information, refer to [Basics on Runnable Functions](#) ([Model Interface Package for Simulink - Modeling Guide](#)).

New features of the data port blocks

The dialogs of the Data Inport and Data Outport blocks now provide additional features.

Creating bus signals You can now create bus signals directly via the block dialogs of Data Inport blocks and Data Outport blocks. New buttons on the Signal Configuration page (, , ) let you create structures for bus signals. You can create untyped buses by defining signals that are bus elements. Alternatively, you can use a Simulink.Bus object (defined in the base workspace or the model's Data Dictionary) to define a virtual or nonvirtual bus. Refer to [Basics of Data Port Blocks with Structured Data Ports](#) ([Model Interface Package for Simulink - Modeling Guide](#)).

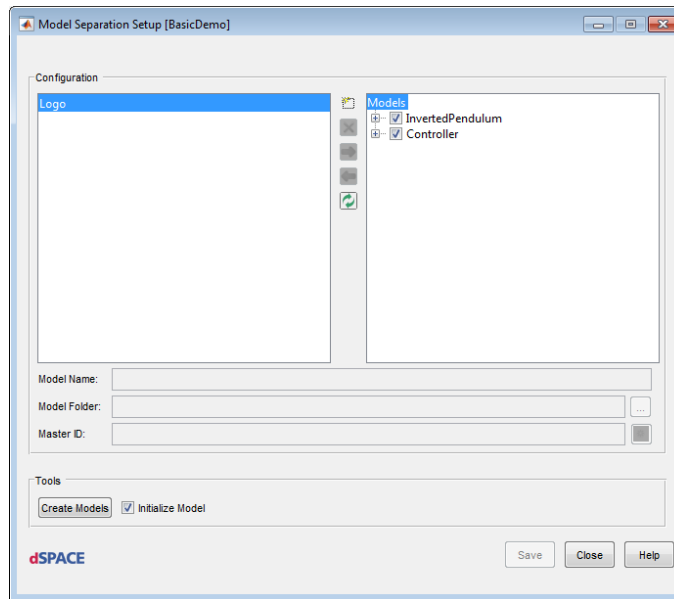
Signal and block IDs You can have model port blocks display their block and signal IDs in the behavior model. Additionally, you can assign new IDs with an API, and the model port block dialogs. Refer to [Basics on Model Port Block IDs and Signal IDs](#) ([Model Interface Package for Simulink - Modeling Guide](#)).

Variable referencing for signal properties The signal widths, initial values and sample times are now MATLAB expressions that can reference variables in the base workspace, the mask workspace of the block's parent system, the model workspace, or the model's Data Dictionary.

Model port block API The new `dsmptb_set`, `dsmptb_get`, `dsmptb_addsignal` and `dsmptb_rmssignal` commands let you read and write data from/to model port blocks. Refer to [General Information on the Model Interface Blockset](#) ([Model Interface Package for Simulink - Modeling Guide](#)).

Redesigned Model Separation Setup

The Model Separation Setup has changed for a better overview:



The Model Separation Setup provides the following new features:

Master ID The Master ID provided by the Model Separation Setup displays an ID from which block and signal IDs are derived during model separation.

Elements supported for model separation You can now separate subsystems with enabled or triggered ports as well as Stateflow charts.

Evaluation of signal lines Model separation considers signal lines in the overall model. For example, if two subsystems are fed by the same input signal and copied into one separated model, model separation inserts one Data Inport block and connects it to both subsystems.

Specifying signals via Simulink objects In the overall model, you can use Simulink.NumericType, Simulink.Bus, and Simulink.AliasType objects to specify signals. However, limitations concerning data types and signal dimensions still apply. Refer to [Limitations of Model Separation](#) ([Model Interface Package for Simulink - Modeling Guide](#)).

Checking the model for code generation compliance Before separating the overall model, model separation checks whether the specified subsystems and their signals comply with the requirements for model separation. For each block and signal that does not comply, model separation produces a message with a link to the block or signal source. This helps you modify the model separation configuration.

For more information, refer to [Separating Models from an Overall Model to Build Multimodel Applications](#) ([Model Interface Package for Simulink - Modeling Guide](#)).

Checking the model for DSRT code generation	The new Check Model for DSRT Code Generation command lets you check if the behavior model fulfills the requirements and conditions for DSRT code generation.
Display of messages	When you work with the Model Interface Package for Simulink, messages inform you about possible problems and upcoming actions. The messages are displayed in the MATLAB Command Window or in dialogs. They provide links or buttons for you to open the related topic in the new Model Interface Package for Simulink Message Reference, which contains more information. Additional links enable you to highlight the block that caused the message. Refer to Model Interface Package For Simulink Message Reference .
New API commands and new API command reference	<p>The Model Interface Package for Simulink provides new and modified API commands. Some of the former API commands can thus be omitted.</p> <p>The new Model Interface Package for Simulink API Reference provides comprehensive information on the API commands provided by the Model Interface Package for Simulink.</p>
Compatibility with MATLAB	The Hardware-Triggered Runnable Function block and the Software-Triggered Runnable Function block now support the Simulink Function-Call Split block.

Migration Aspects of the Model Interface Package for Simulink

Block and signal IDs	<p>The Model Interface Package for Simulink now always assigns block and signal IDs to all model port blocks. When you open Simulink models from previous Model Interface Package for Simulink versions, messages might be displayed in the MATLAB Command Window stating that IDs were assigned to a particular block.</p> <p>Model port blocks in libraries As opposed to previous Model Interface Package for Simulink versions, model port blocks in libraries are also assigned IDs. When you open a library, messages might be displayed in the MATLAB Command Window stating that IDs were assigned to a particular block. In this case, you have to save the library to make sure that these modifications become effective. If you fail to do so, you are asked whether to save the library.</p>
-----------------------------	--

Tip

To migrate models and libraries, you can also use the `dsmpb_migrate()` API command. Refer to [dsmpb_migrate](#) ([Model Interface Package for Simulink API Reference](#)).

Fixing broken library links

After you opened a model, Model Separation Setup blocks might be displayed as broken library links. To fix this, open the Model Separation Setup via the Model Port Blocks - Open Model Separation Setup menu command, or run the `dsmpb_migrate()` API command for the model.

Unresolved model port blocks in ConfigurationDesk

For model port blocks generated during model separation, block and signal IDs assigned to the block no longer depend on port and signal names. If you re-create the separated models, blocks and signals therefore have different IDs. If you added the models to a ConfigurationDesk project and re-analyze the newly generated models, model port blocks in ConfigurationDesk might be unresolved. You have to replace the unresolved model port blocks in the signal chain and restore their mapping to function blocks or other model port blocks. You can delete the unresolved model port blocks in ConfigurationDesk.

Error and message handling

The Model Interface Package for Simulink provides joint error and message handling. Model Interface Package for Simulink automation tools return a `dsmpb.MsgContainer` object that contains messages that inform you whether errors or warnings have occurred. For more information, refer to the [Model Interface Package for Simulink API Reference](#) to get information about input and output parameters of Model Interface Package for Simulink tools. To get information about interfaces of the `dsmpb.MsgContainer` class, type the following command in the MATLAB Command Window:

```
>> doc dsmpb.MsgContainer
```

Replaced API commands

The following API commands are obsolete and replaced by new API commands:

Old API Command	New API Command
<code>dscms_modelsave(mode)</code>	<code>dsmpb_pref()</code>
<code>dsrt_customization()</code>	<code>dsmpb_pref()</code>
<code>dscms_slrootportsconfig()</code>	<code>dsmpb_configurerootslports()</code>
<code>dsmpb_createversedmodelportblock()</code>	<code>dsmpb_createinversedmodelportblock()</code>
<code>dsmpb_copywithidentity()</code>	<code>dsmpb_copyblockwithids()</code>

The Model Interface Package for Simulink 4.0 still supports the obsolete API commands (except `dsmpb_copywithidentity()`), but support will be discontinued in future versions.

Note

The signatures of the new API commands are not identical with the signatures of the obsolete API commands.

Obsolete MATLAB classes

The following MATLAB classes are no longer supported:

- DsDataInport
- DsDataOutport
- DsRunnableFunction
- DsModelPortBlock

Instead, use the following API commands to read and write model port block data:

- `dsmptb_get()`
- `dsmptb_set()`
- `dsmptb_rmssignal()`
- `dsmptb_addsignal()`

Modified API command signatures

The signatures of the following API commands have changed:

- `dsmptb_generatemodelportblock()`
- `dsmptb_migrate()`
- `dsmptb_md1cleanup()`
- `dsmptb_createmodelportblockperiphery()`

For more information, refer to [Model Interface Package for Simulink API Reference](#).

No model port block update with unsupported bus signals

If you connect a Simulink bus signal that contains unsupported signals (for example, variable size signals or complex signals) to a structured data port or use a bus in the context of the automation API commands and update the structured data port from its input signals, the update is aborted with an error message.

Unsupported new features of MATLAB R2018b

The following new features introduced with MATLAB R2018b are not supported by the Model Interface Package for Simulink:

- With MATLAB R2018b, the String data type, which was introduced with MATLAB R2018a, can now be an element of a non-virtual bus. Because the String data type is currently not supported for TRC file generation, non-virtual buses that contains strings will also be omitted from the TRC file.
- As of MATLAB R2018b, you can specify whether the layout of matrices with two or more dimensions must be *column major* or *row major*. You can specify this for each model. If you select the *column major* option, the first index indicates the column. If you select the *row major* option, the first index indicates the row.

In previous Simulink versions, all matrices were generated in *column major* format.

The *row major* format is not supported by the Model Interface Package for Simulink.

MotionDesk

Where to go from here

Information in this section

New Features of MotionDesk 4.3	149
Migrating to MotionDesk 4.3	150

New Features of MotionDesk 4.3

Sensor Simulation

MotionDesk supports Sensor Simulation. It is used for the following tasks in Sensor Simulation:

- Creating several camera sensors and laser sensors.
- Specifying the properties of the sensors
- Managing the pool of SensorSim applications
- Configuring and managing the SensorSim applications
- Supporting shadows for camera sensors
- Providing an API for postprocessing of images for camera sensors
- Downloading the scene to the SensorSim applications

For more information, refer to [New Product: Sensor Simulation](#) on page 175.

Data acquisition

The new Model and Sensor Interface blockset has been developed to get motion data for animating a scene. MotionDesk can receive, save, and use the motion data from this blockset. You must use this blockset when working with sensor simulation.

Performance optimization

To increase the frame rate, hardware instancing of commonly used static objects, such as houses and trees, is implemented.

Video generation

You can integrate a logo in the video.

Related topics**Basics**

[Working with the Model and Sensor Interface Blockset \(MotionDesk Calculating and Streaming Motion Data\)](#)

Migrating to MotionDesk 4.3

Using endless ground plate and horizon

In MotionDesk 4.0 and earlier, the virtual world of a scene was built using ground plate and dome 3-D objects. If you want to use the endless ground plate and sky, these 3-D objects are obsolete. When you use an old scene, delete these objects before activating the endless ground and sky.

Using advanced lighting mode

In advanced lighting mode, the static objects used for domes are not suitable for building the virtual world. Use the endless horizon of the environment instead.

Migrating 3-D custom objects

If you want to use 3-D custom objects in the VRML2 format that you used in MotionDesk 2.2.1 or earlier, you have to convert the VRML2 files to COLLADA format files. You can convert the files at any time using the 3-D Library Manager.

Migrating from MotionDesk version 2.2.1 and earlier

The current MotionDesk version cannot read old MotionDesk experiments in the MDX file format (used in MotionDesk 2.1.6 and earlier) or scenes stored in the ESD format (used in MotionDesk 2.2.1 and earlier). It is therefore not possible to migrate from MotionDesk projects and experiments of these versions.

If you want to use older projects and experiments, you can migrate them by using MotionDesk 3.0 up to MotionDesk 3.6 and then open them in the current MotionDesk version.

Real-Time Testing

Where to go from here

Information in this section

New Features of Real-Time Testing 4.0	151
Migrating to Real-Time Testing 4.0	152

New Features of Real-Time Testing 4.0

New Python version

Real-Time Testing supports Python 3.6 on the host PC. The real-time part continues to use Python 2.7.11. Therefore, the RTT sequences must not be migrated.

ASAM XIL API

Real-Time Testing supports ASAM XIL API 2.1.

To support this version, Real-Time Testing provides the following new features:

- The watcherlib module provides new functions to detect a positive or negative value change.
- For data replay of a MAT file, you can specify a start value and duration.
- The RTT sequences can return an output parameter to the host PC.

Ethernet access of V-ECUs

The rtt.dsethernetapilib module supports Ethernet access on V-ECUs executed on a VEOS platform.

Remote VPUs

You can access variables of remote VPUs on a VEOS platform.

Related topics

References

[rttlib.datastream Module](#) (📖 Real-Time Testing Library Reference)
[rttlib.dsethernetapilib Module](#) (📖 Real-Time Testing Library Reference)
[rttlib.variable Module](#) (📖 Real-Time Testing Library Reference)
[rttlib.watcherlib Module](#) (📖 Real-Time Testing Library Reference)

Migrating to Real-Time Testing 4.0

Incompatible BCG files

The BCG files generated with Real-Time Testing 2.6 or earlier cannot be used for Real-Time Testing 4.0. You must create the BCG file of the Real-Time Testing sequence again. Refer to [Creating and Starting RTT Sequences in Python Scripts](#) (📖 Real-Time Testing Guide).

Static code analysis

As of Real-Time Testing 4.0, the static code analysis is not longer included.

Variable access and rounding behavior

If the Simulink variable type is integer and the value written to the variable is of floating point type, the floating point value is rounded to fit the Simulink integer variable.

RTI/RTI-MP and RTLib

Where to go from here

Information in this section

New Features of RTI/RTI-MP and RTLib	153
Migration Aspects of RTI/RTI-MP and RTLib	154

New Features of RTI/RTI-MP and RTLib

New features in RTI/RTI-MP

RTI-MP now provides a hook function to customize the model separation and the update of submodels. To use the hook function, a specific M file must be available when you apply settings in the **Multiprocessor Setup** dialog or during the build process. The **Templates** sublibrary of the RTI-MP blockset contains a template for this M file. The sublibrary also contains an example file that shows you how to specify a *PostSeparation* step, which sets the Simulink block priority of an IPC block.

For more information on the `rti1xxx_template_rtimphook.m` file, refer to [User-Supplied Files](#) ([RTI and RTI-MP Implementation Reference](#)).

Unsupported new features of MATLAB R2018b

The following new features introduced with MATLAB R2018b are not supported by RTI/RTI-MP:

- With MATLAB R2018b, the String data type, which was introduced with MATLAB R2018a, can now be an element of a non-virtual bus. Because the String data type is currently not supported for TRC file generation, non-virtual buses that contains strings will also be omitted from the TRC file.
- As of MATLAB R2018b, you can specify whether the layout of matrices with two or more dimensions must be *column major* or *row major*. You can specify this for each model. If you select the *column major* option, the first index

indicates the column. If you select the *row major* option, the first index indicates the row.

In previous Simulink versions, all matrices were generated in *column major* format.

The *row major* format is not supported by RTI and RTI-MP.

- If you use the obsolete `rti_build` command with MATLAB R2018b, generating code for protected models is not supported, Use the `rti_build2` command instead.

Migration Aspects of RTI/RTI-MP and RTLib

Modified features in later MATLAB versions

Switching to a later MATLAB version If you install a new MATLAB version, some settings are adopted from previously installed MATLAB versions. To prevent unexpected behavior of your Simulink models when switching to a later MATLAB version or dSPACE Release, always reset the MATLAB and Simulink preferences to their defaults before you start using them.

RTI Bypass Blockset

Migrating to RTI Bypass Blockset 3.11

Working with models from earlier RTI Bypass Blockset versions 3.x and 2.x

The current Release contains RTI Bypass Blockset 3.11, which is compatible with earlier blockset versions 3.x and 2.x. However, there are some points to note:

- *Working with models from RTI Bypass Blockset 2.5 or earlier*

Data management was changed in comparison to the prior RTI Bypass Blockset versions. If you have a Simulink model built with RTI Bypass Blockset 2.5 or earlier and you open it with RTI Bypass Blockset 3.11, the old Data Dictionary file (with the file name extension .dd) is replaced by a new Data Dictionary file (.vdb) using the information stored in the Setup block. This happens as soon as you open and close the Setup block dialog by clicking OK, or you open the Read, Write, Upload, or Download block dialog and click Fill Variable Selector on the Variables page.

If you have a model that was saved with RTI Bypass Blockset 3.11 and want to use it with RTI Bypass Blockset 2.5 or earlier, the model's Data Dictionary file required for blockset version 2.5 or earlier (file name extension .dd) is created. This happens when you update the A2L files in the Setup block, or you open the Read, Write, Upload, or Download block and click Fill Variable Selector on the Variables page. The Data Dictionary file created under RTI Bypass Blockset 3.11 (.vdb) remains on the disk.

To enable the RTI Bypass Blockset to recreate the Data Dictionary, the database files specified in the Setup block must be accessible at the specified location and must be unchanged.

- *Working with models from RTI Bypass Blockset 2.6 up to and including RTI Bypass Blockset 3.10*

If a Simulink model was built with RTI Bypass Blockset 2.6 up to RTI Bypass Blockset 3.10, and you open it with RTI Bypass Blockset 3.11, the old Data Dictionary file is replaced by a new Data Dictionary file. However, the new Data Dictionary file cannot be used in earlier RTI Bypass Blockset versions. If you want to reuse the model with RTI Bypass Blockset 2.6 up to RTI Bypass Blockset 3.10, you have to create a suitable database in the earlier RTI Bypass Blockset version by reimporting the database files (A2L files) specified in the Setup block.

RTI CAN MultiMessage Blockset

Where to go from here

Information in this section

New Features of the RTI CAN MultiMessage Blockset 5.1	157
Migrating to RTI CAN MultiMessage Blockset 5.1	159
Distribution of the Bus Manager to RTI CAN MultiMessage Blockset Customers	159

New Features of the RTI CAN MultiMessage Blockset 5.1

Support of AUTOSAR System Template 4.3.1

The RTI CAN MultiMessage Blockset supports the AUTOSAR System Template based on AUTOSAR Release 4.3.1 for describing CAN networks.

Refer to [General Settings Page \(RTICANMM MainBlock\)](#) ([📖 RTI CAN MultiMessage Blockset Reference](#)).

This feature has already been available since RTI CAN MultiMessage Blockset 5.0p1.

Support of FIBEX 4.1.2

The RTI CAN MultiMessage Blockset now also supports FIBEX 4.1.2 files as database files.

Refer to [General Settings Page \(RTICANMM MainBlock\)](#) ([📖 RTI CAN MultiMessage Blockset Reference](#)).

This feature has already been available since RTI CAN MultiMessage Blockset 5.0p1.

Support of container and contained IPDUs

The RTI CAN MultiMessage Blockset supports *container IPDUs*. A container IPDU is an IPDU that contains one or more smaller *contained IPDUs*. A container IPDU can be used to transmit multiple contained IPDUs in one container IPDU in a single CAN FD message on the CAN bus.

Refer to [Aspects of Miscellaneous Supported AUTOSAR Features](#) ( [RTI CAN MultiMessage Blockset Reference](#)).

This feature has already been available since RTI CAN MultiMessage Blockset 5.0p1.

Support of secure onboard communication

The RTI CAN MultiMessage Blockset supports *secure onboard communication (SecOC)* according to AUTOSAR 4.3.0. Secure onboard communication provides authentication mechanisms to identify PDU data that was modified or transmitted without authorization, e.g., due to a replay attack. To implement secure onboard communication, you must enable the support of secure onboard communication and provide the OEM-specific implementation for generating authentication information via user code.

Refer to [Aspects of Miscellaneous Supported AUTOSAR Features](#) ( [RTI CAN MultiMessage Blockset Reference](#)).

This feature has already been available since RTI CAN MultiMessage Blockset 5.0p1.

Support of global time synchronization

The RTI CAN MultiMessage Blockset supports *global time synchronization (GTS)* according to AUTOSAR 4.3.0. Global time synchronization means providing and distributing synchronized times in the vehicle across all ECUs.

Refer to [Aspects of Miscellaneous Supported AUTOSAR Features](#) ( [RTI CAN MultiMessage Blockset Reference](#)).

This feature has already been available since RTI CAN MultiMessage Blockset 5.0p1.

The *RTI Synchronized Time Base Manager Blockset* implements global time synchronization (GTS) on dSPACE systems by using the dSPACE ECU time base manager (DsEcuTbM) contained in the RTLib of SCALEXIO and other supported hardware. The blockset provides the following tasks, among others:

- Creating and configuring synchronized time base instances.
- Reading the synchronized time information from the synchronized time base instances, and providing the time and status of a time base instance.
- Simulating global time masters.

For more information on the RTI Synchronized Time Base Manager Blockset, refer to its documentation, which you can open by clicking the Help button in the block library.

The RTI Synchronized Time Base Manager Blockset can already be used with RTI CAN MultiMessage Blockset 5.0p1.

Migrating to RTI CAN MultiMessage Blockset 5.1

Working with models from earlier RTI CAN MultiMessage Blockset versions

To reuse a model created with an earlier RTI CAN MultiMessage Blockset version, you must update the S-functions for all the RTICANMM blocks and save the model before modifying the CAN configuration.

To create new S-functions for all the RTICANMM blocks in a model in one step, you can perform one of the following actions after opening the model:

- In the MATLAB Command Window, enter `rtimmsu_update('System', bdroot)`.

For more information on the command and its options, enter `help rtimmsu_update` in the MATLAB Command Window.

- Select the Create S-Function for all CAN Blocks command from the Options menu of the RTICANMM GeneralSetup block.

For more information, refer to [Limitations with RTICANMM](#) ([RTI CAN MultiMessage Blockset Reference](#)).

Compiler messages when using code generated by an RTI CAN MultiMessage Blockset version < 4.0

If you use code that was generated by an RTI CAN MultiMessage Blockset version < 4.0, several compiler warning messages that contain the phrase `<<argument of type "can_tp1_canChannel *" is incompatible with parameter of type "DsTCanCh">>` will be displayed during the build process of a simulation model. This is due to a modified data type. These warnings can be ignored and disappear after you use the current blockset version to generate the RTICANMM code again.

Using existing checksum algorithms

Checksum algorithms that were originally developed for an application and contain CAN messages cannot be reused for applications that contain CAN FD messages, because CAN FD includes new message types and longer data fields. Existing checksum algorithms can still be used for applications that contain only classic CAN messages. For CAN FD applications, you must adapt the checksum algorithms.

Distribution of the Bus Manager to RTI CAN MultiMessage Blockset Customers

Introduction

With dSPACE Release 2018-B, the Bus Manager is automatically distributed to certain customers of the RTI CAN MultiMessage Blockset with a valid Software Maintenance Service (SMS) contract. For these customers, the SMS contract is automatically migrated so that it covers both products, the RTI CAN MultiMessage Blockset and the Bus Manager.

The Bus Manager is dSPACE software for configuring bus communication and implementing it in real-time applications for dSPACE SCALEXIO systems, or for generating bus simulation containers. Bus simulation containers can be used on various dSPACE simulation platforms. The Bus Manager is available as a component in ConfigurationDesk and as a stand-alone version.

Note

The automatic migration of the SMS contract is done only with Release 2018-B, not with any following Release.

Preconditions

The Bus Manager is automatically distributed to customers that fulfill all the following preconditions:

- They use the RTI CAN MultiMessage Blockset.
- They have a valid Software Maintenance Service (SMS) contract for the RTI CAN MultiMessage Blockset.
- They use a SCALEXIO system.

Tip

If you fulfill all of the preconditions but the Bus Manager is not distributed to you, contact dSPACE.



Distribution routine

dSPACE sends you an e-mail with a license overview sheet based on your SMS contract. This sheet lists not only the products for which you purchased an SMS contract, but also the following:

- Bus Manager Base (BUS_MANAGER_BASE)
- Bus Manager Plus (BUS_MANAGER_PLUS)
- ConfigurationDesk CAN module (CFD_I_CAN)

Steps required for using the Bus Manager

To use the Bus Manager, including the user documentation and demo files, you must do the following:

1. Activate the newly distributed licenses.
Refer to [Workflows for Updating to dSPACE Release 2018-B \(with SMS Contract\)](#) ( [Working with CodeMeter Licensing Technology](#)).
2. Decrypt the related encrypted archives.
Refer to [Decrypting Encrypted Archives of dSPACE Software Installations](#) ( [Managing dSPACE Software Installations](#)).

Depending on the installed product sets, the following Bus Manager versions can be available:

Bus Manager in ConfigurationDesk	Bus Manager (Stand-Alone)
<ul style="list-style-type: none"> ▪ Available for the ConfigurationDesk - Implementation Version product set ▪ Refer to 📖 ConfigurationDesk Bus Manager Implementation Guide 	<ul style="list-style-type: none"> ▪ Available for the Bus Support product set ▪ Refer to 📖 Bus Manager (Stand-Alone) Implementation Guide

Using the Bus Manager and the RTI CAN MultiMessage Blockset

You can use both products, the Bus Manager and the RTI CAN MultiMessage Blockset, at the same time. For example, for the following tasks:

- Setting up new projects with the Bus Manager while using the RTI CAN MultiMessage Blockset for existing projects.
- Implementing new functionalities in existing projects with the Bus Manager while existing functionalities were implemented with the RTI CAN MultiMessage Blockset.

Effects on the SMS contract

When you activate the newly distributed licenses, your SMS contract of the RTI CAN MultiMessage Blockset is migrated to an SMS contract for the Bus Manager. However, this contract covers the licenses of both products. This has the following effects:

- If you update the Bus Manager on the basis of the SMS contract, the RTI CAN MultiMessage Blockset is updated automatically.
- If you cancel the SMS contract for the Bus Manager, the contract is canceled for the RTI CAN MultiMessage Blockset as well, i.e., you lose the maintenance service for both products.

RTI FPGA Programming Blockset

Where to go from here

Information in this section

New Features of the RTI FPGA Programming Blockset 3.6	163
Migrating to RTI FPGA Programming Blockset 3.6	165

New Features of the RTI FPGA Programming Blockset 3.6

Extended Xilinx® support

The RTI FPGA Programming Blockset now supports the following products and versions of the Xilinx design tools:

Xilinx Design Tools Version	MATLAB Version ¹⁾	Operating System
Vivado 2018.2	<ul style="list-style-type: none"> ▪ MATLAB R2017a ▪ MATLAB R2017b ▪ MATLAB R2018a 	All PC operating systems that are supported by RCP and HIL software of dSPACE Release 2018-B. Refer to Operating System on page 266.

¹⁾ The Processor Interface sublibrary of the RTI FPGA Programming Blockset also supports MATLAB R2018b.

Enhancements to the MicroAutoBox frameworks

The FPGA1401Tp1 frameworks for MicroAutoBox now provide the following enhancement.

Tracing of FPGA signals You can now make FPGA signals of MicroAutoBox traceable. This lets you access FPGA signals with the experiment software, for example, ControlDesk. Refer to [Tracing FPGA Signals](#) ([RTI FPGA Programming Blockset Guide](#)).

Tuning of FPGA constants You can add tunable FPGA constants to an FPGA application and adjust them with the experiment software.

For more information, refer to [Adjusting Values of FPGA Constants](#) ([RTI FPGA Programming Blockset Guide](#)).

Support of FPGA test access and scaling For FPGA applications, you can enable FPGA test access and scaling of I/O signals by means of your experiment software.

FPGA test access lets you use intervention points for the experiment software. The intervention points let you set values for the I/O interface and for data values that are exchanged with the processor interface.

Scaling I/O signals includes scaling, saturating, and inverting of I/O signals.

For more information, refer to [Enabling the FPGA Test Access and Scaling](#) ([RTI FPGA Programming Blockset Guide](#)).

64-bit data format The MicroAutoBox frameworks now provide 64-bit registers and buffers to exchange data values between the FPGA and the processor application.

New MicroLabBox framework

An I/O channel of MicroLabBox is used either by the standard I/O features of MicroLabBox or by a custom FPGA application. The standard I/O features of MicroLabBox let you access the I/O interface with the RTI Blockset of MicroLabBox and the RTI Electric Motor Control Blockset. These features are automatically implemented on the FPGA if the real-time application that you load to the hardware does not contain a custom FPGA application. Custom FPGA applications that are built with the DS1202 FPGA I/O Type 1 framework do not provide the standard I/O features for remaining I/O channels. Therefore, you cannot access the remaining I/O channels with the RTI1202 and the EMC blocksets.

The new DS1202 FPGA I/O Type 1 (Flexible I/O) framework for MicroLabBox provides the following features:

- During the build process of the custom FPGA application, the standard I/O features for remaining I/O channels are added. The standard I/O features let you use the remaining I/O channels with the RTI1202 and the EMC blocksets.
- The framework provides the same blocks as the DS1202 FPGA I/O Type 1 framework.
- The framework is compatible with the DS1202 FPGA I/O Type 1 framework. You can switch between the DS1202 FPGA I/O Type 1 (Flexible I/O) and the DS1202 FPGA I/O Type 1 frameworks.

Enhancements to the frameworks of the DS2655 FPGA Base Board

FPGA applications that are modeled with function blocks for the DS2655 FPGA Base Board and for the I/O modules provide the following enhancements.

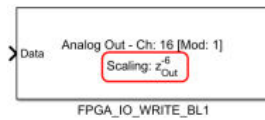
Improved angular processing units (APU) The angular processing units of SCALEXIO systems now support virtual engines with a velocity up to 1,200,000 %/s (200,000 rpm). Furthermore, I/O functions can be set to an angle range that is different to the APU.

The APU Master and APU Slave I/O functions now support the improved APU.

General enhancements

Improved access to reports The Parameters page of the FPGA_SETUP_BL block lets you display the last timing and resource utilization reports. You can select the report via FPGA model action.

Display of the latency FPGA scaling of analog I/O signals might cause additional latency. If the latency can be calculated during modeling, analog I/O functions display the total latency.



Related topics

Basics

Migrating to RTI FPGA Programming Blockset 3.6	165
--	-----

Migrating to RTI FPGA Programming Blockset 3.6

Introduction

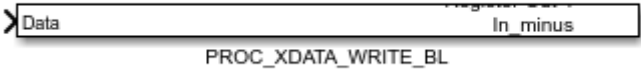
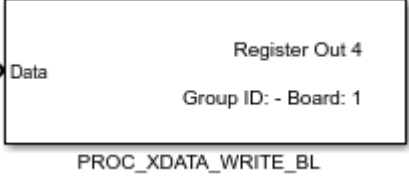
There are various ways to migrate an existing model, depending on the blockset version used.

Migrating from RTI FPGA Programming Blockset 1.1 and higher to 3.6

If you implemented your FPGA application with RTI FPGA Programming Blockset Version 1.1 and later and want to use it with RTI FPGA Programming Blockset 3.6, the framework automatically updates itself to the current framework version.

The update handles all the subsystems in the model/subsystem. The parameters of the blocks stay the same after updating to the current framework version.

Appearance of migrated processor interfaces with 'Goto' and 'From' blocks With RTI FPGA Programming Blockset 3.1 ... 3.3, you modeled the processor interface of a SCALEXIO system with Simulink Goto and From blocks. If you migrate a model with Goto and From blocks, the update process migrates these blocks to the processor interface blocks of the Processor Interface sublibrary. The migration process does not change the size of the origin blocks to keep the block arrangement of the model. Therefore, the appearance of the migrated blocks is different to the default appearance of processor interface blocks. The following illustrations give you an example.

Appearance After Migration	Default Appearance
	

ConfigurationDesk custom functions incompatible with dSPACE Release 2018-B

FPGA custom function block types that are not built with the RTI FPGA Programming Blockset 3.6 might be incompatible with the current ConfigurationDesk version.

FPGA Programming Blockset 3.5 or earlier As of dSPACE Release 2018-B, the angle range handling of the angular processing unit (APU) has been changed. FPGA custom function blocks that use the APU in the 360° angle range are incompatible if they are built with the FPGA Programming Blockset 3.5 or earlier.

To resolve the incompatibility, use the FPGA model/code of the incompatible FPGA custom function block and build a new FPGA custom function block with the RTI FPGA Programming Blockset 3.6. The RTI FPGA Programming Blockset automatically migrates the framework of the FPGA model/code to the current version.

RTI FPGA Programming Blockset 2.5 An FPGA custom function block generated with RTI FPGA Programming Blockset 2.5 from dSPACE Release 2013-A and the real-time applications containing the FPGA custom function block are incompatible with the current dSPACE Release. To produce a usable custom function, you have to rebuild the FPGA model by using the current RTI FPGA Blockset.

Using different dSPACE hardware

Using an FPGA model on different dSPACE hardware requires some model modifications. Refer to [Migrating to Another dSPACE Hardware Platform](#) ([RTI FPGA Programming Blockset Guide](#)).

RTI LIN MultiMessage Blockset

Where to go from here

Information in this section

New Features of the RTI LIN MultiMessage Blockset 3.1	167
Migrating to RTI LIN MultiMessage Blockset 3.1	168
Distribution of the Bus Manager to RTI LIN MultiMessage Blockset Customers	168

New Features of the RTI LIN MultiMessage Blockset 3.1

Support of AUTOSAR System Template 4.3.1

The RTI LIN MultiMessage Blockset supports the AUTOSAR System Template based on AUTOSAR Release 4.3.1 for describing LIN networks.

Refer to [General Settings Page \(RTILINMM MainSetup\)](#) ([📖 RTI LIN MultiMessage Blockset Reference](#)).

This feature has already been available since RTI LIN MultiMessage Blockset 3.0p1.

Support of FIBEX 4.1.2

The RTI LIN MultiMessage Blockset now also supports FIBEX 4.1.2 files as database files.

Refer to [General Settings Page \(RTILINMM MainSetup\)](#) ([📖 RTI LIN MultiMessage Blockset Reference](#)).

This feature has already been available since RTI LIN MultiMessage Blockset 3.0p1.

Migrating to RTI LIN MultiMessage Blockset 3.1

Working with models from earlier RTI LIN MultiMessage Blockset versions

To reuse a model created with an earlier RTI LIN MultiMessage Blockset version, you must update the S-functions for all the RTILINMM blocks and save the model before modifying the LIN configuration.

To create new S-functions for all the RTILINMM blocks in a model in one step, you can perform one of the following actions after opening the model:

- In the MATLAB Command Window, enter `rtimmsu_update('System', bdroot)`.

For more information on the command and its options, enter `help rtimmsu_update` in the MATLAB Command Window.

- Select the Create S-Function for all LIN Blocks command from the Options menu of the RTILINMM GeneralSetup block.

For more information, refer to [Limitations of RTI LIN MultiMessage Blockset](#) ( [RTI LIN MultiMessage Blockset Reference](#)).

Distribution of the Bus Manager to RTI LIN MultiMessage Blockset Customers

Introduction

With dSPACE Release 2018-B, the Bus Manager is automatically distributed to certain customers of the RTI LIN MultiMessage Blockset with a valid Software Maintenance Service (SMS) contract. For these customers, the SMS contract is automatically migrated so that it covers both products, the RTI LIN MultiMessage Blockset and the Bus Manager.

The Bus Manager is dSPACE software for configuring bus communication and implementing it in real-time applications for dSPACE SCALEXIO systems, or for generating bus simulation containers. Bus simulation containers can be used on various dSPACE simulation platforms. The Bus Manager is available as a component in ConfigurationDesk and as a stand-alone version.

Note

The automatic migration of the SMS contract is done only with Release 2018-B, not with any following Release.

Preconditions

The Bus Manager is automatically distributed to customers that fulfill all the following preconditions:

- They use the RTI LIN MultiMessage Blockset.
- They have a valid Software Maintenance Service (SMS) contract for the RTI LIN MultiMessage Blockset.
- They use a SCALEXIO system.

Tip

If you fulfill all of the preconditions but the Bus Manager is not distributed to you, contact dSPACE.

Distribution routine

dSPACE sends you an e-mail with a license overview sheet based on your SMS contract. This sheet lists not only the products for which you purchased an SMS contract, but also the following:

- Bus Manager Base (BUS_MANAGER_BASE)
- Bus Manager Plus (BUS_MANAGER_PLUS)
- ConfigurationDesk LIN module (CFD_I_LIN)

Steps required for using the Bus Manager

To use the Bus Manager, including the user documentation and demo files, you must do the following:

1. Activate the newly distributed licenses.
Refer to [Workflows for Updating to dSPACE Release 2018-B \(with SMS Contract\)](#) ([Working with CodeMeter Licensing Technology](#)).
2. Decrypt the related encrypted archives.
Refer to [Decrypting Encrypted Archives of dSPACE Software Installations](#) ([Managing dSPACE Software Installations](#)).

Depending on the installed product sets, the following Bus Manager versions can be available:

Bus Manager in ConfigurationDesk	Bus Manager (Stand-Alone)
<ul style="list-style-type: none"> ▪ Available for the ConfigurationDesk - Implementation Version product set ▪ Refer to ConfigurationDesk Bus Manager Implementation Guide 	<ul style="list-style-type: none"> ▪ Available for the Bus Support product set ▪ Refer to Bus Manager (Stand-Alone) Implementation Guide

Using the Bus Manager and the RTI LIN MultiMessage Blockset

You can use both products, the Bus Manager and the RTI LIN MultiMessage Blockset, at the same time. For example, for the following tasks:

- Setting up new projects with the Bus Manager while using the RTI LIN MultiMessage Blockset for existing projects.
- Implementing new functionalities in existing projects with the Bus Manager while existing functionalities were implemented with the RTI LIN MultiMessage Blockset.

Effects on the SMS contract

When you activate the newly distributed licenses, your SMS contract of the RTI LIN MultiMessage Blockset is migrated to an SMS contract for the Bus Manager.

However, this contract covers the licenses of both products. This has the following effects:

- If you update the Bus Manager on the basis of the SMS contract, the RTI LIN MultiMessage Blockset is updated automatically.
- If you cancel the SMS contract for the Bus Manager, the contract is canceled for the RTI LIN MultiMessage Blockset as well, i.e., you lose the maintenance service for both products.

SCALEXIO Firmware

New Features of the SCALEXIO Firmware 4.3

New supported hardware

The SCALEXIO firmware supports the following new hardware:

- **DS6333-CS/DS6333-PE Automotive Ethernet Board**

The DS6333-CS board has a Compact PCI Serial interface. It can be installed in a PCI Express slot of the LabBox (version 2) in combination with the DS6001 Processor Board.

The DS6333-PE board has a PCI Express interface. It can be installed in the Real-Time PC of a SCALEXIO Processing Unit.

The boards are available in several variants supporting Automotive Ethernet (transfer rates of 100 and 1000 M Bit/s) and standard Ethernet (transfer rates of 10, 100 and 1000 MBit/s):

- 4 × automotive Ethernet, 1 × standard Ethernet
- 5 × Standard Ethernet
- 2 × automotive Ethernet, 3 × standard Ethernet

- **DS6334-PE Ethernet Board**

The board has a PCI Express interface. It can be installed in the Real-Time PC of a SCALEXIO Processing Unit.

The board has 4 Intel I210 Ethernet Controller. The RJ45 port supports a transfer rate of 10, 100, and 1000 Mbit/s.

The Ethernet Configuration Package does not support the DS6334-PE Ethernet Board.

- **DS6335-CS Ethernet Board**

The board has a Compact PCI Serial interface. It can be installed in a PCI Express slot of the LabBox (version 2) in combination with the DS6001 Processor Board.

The board is available in several variants that support automotive Ethernet (transfer rates of 100 and 1000 MBit/s) and standard Ethernet (transfer rates of 10, 100 and 1000 MBit/s):

- 4 × automotive Ethernet, 1 × standard Ethernet
- 5 × Standard Ethernet
- 2 × automotive Ethernet, 3 × standard Ethernet

The Ethernet Configuration Package does not support the DS6335-CS Ethernet Board.

- SCALEXIO LabBox (8-slot)

This is a compact extension of the SCALEXIO family that offers space for up to 7 SCALEXIO I/O boards to cover all the essential I/O functions. SCALEXIO LabBox (8-slot) is designed for standard SCALEXIO I/O boards and dSPACE-qualified CompactPCI® Serial boards (PCI Serial boards can only be used in combination with a DS6001). The LabBox has one system slot to insert a DS6001 Processor Board as processing hardware or an IOCNET router to add SCALEXIO I/O boards to an existing SCALEXIO system. If a DS6001 Processor Board is in the system slot, the LabBox can be used as a compact, stand-alone SCALEXIO system, or as a root or node in a distributed SCALEXIO system.

DS6001 Processor Board

The firmware of the DS6001 Processor Board has new features:

Gigalink connection You can connect a DS6001 Processor Board to a PHS-bus-based system (i.e., a modular system with a DS1005, DS1006, or DS1007) via Gigalink to exchange data.

Wake-up on I/O You can use the DS6001 Processor Board for a wake-up scenario if a suitable CAN or LIN board is connected.

The SCALEXIO system that is based on a DS6001 Processor Board can be woken up by a CAN or LIN wake-up. If you enable power wake-up for the CAN or LIN function block, the SCALEXIO system can change its operation mode from standby mode to normal operation mode when a wake-up is sent via the CAN or LIN bus.

Data set storage You can store data sets to the board's nonvolatile memory. Even if the DS6001 is shut down, the data set still exists. The real-time application can use this data in the following execution.


Support of SCALEXIO Hypervisor Extension

As of Release 2018-B the SCALEXIO Hypervisor Extension is part of the regular dSPACE Release. The hypervisor extends the SCALEXIO firmware and the files required for installing and working with the hypervisor are delivered by dSPACE on a specific CompactFlash Card.


With the SCALEXIO Hypervisor Extension you can integrate Linux software components, for example, modeling and simulation software, into SCALEXIO real-time applications. The Hypervisor Extension also supports the parallel and synchronized execution of SCALEXIO real-time applications and Linux real-time applications as well as non-real-time applications.

Note

If you already use the SCALEXIO Hypervisor Extension in your projects with dSPACE Release 2018-A or earlier, you cannot use the provided and installed files of these Releases with dSPACE Release 2018-B. If a migration to dSPACE Release 2018-B is required, contact dSPACE Support (www.dspace.com/go/supportrequest).

For features, basics, and instructions on using the SCALEXIO Hypervisor Extension, refer to  [SCALEXIO Hypervisor Extension](#).

Related topics**Basics**

[Configuring the Basic Functionality \(CAN\)](#) ( ConfigurationDesk I/O Function Implementation Guide)

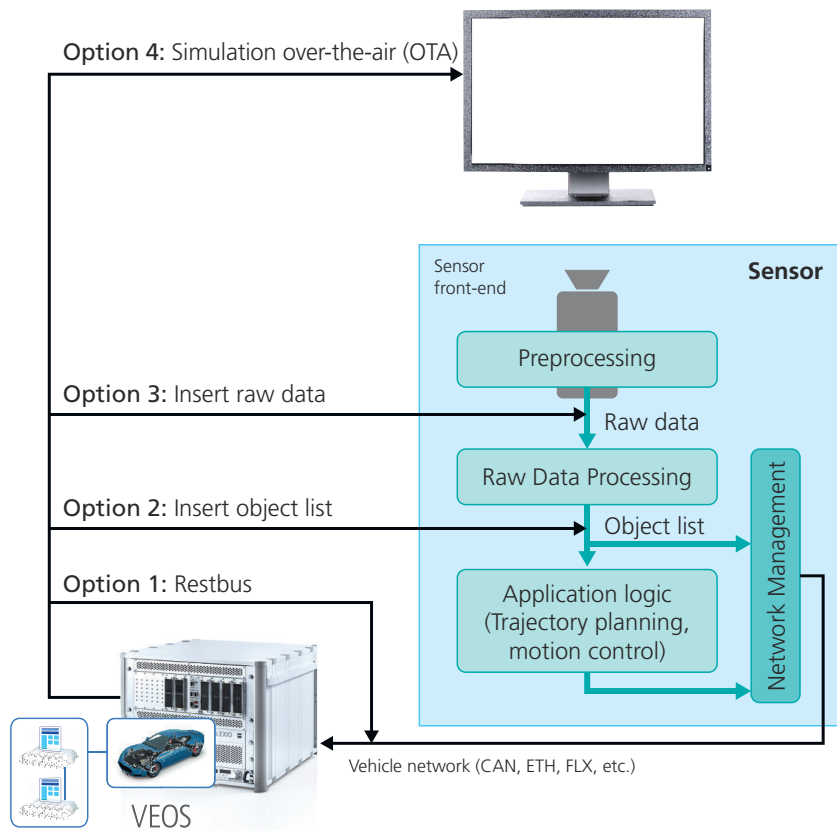
[Configuring the Basic Functionality \(LIN\)](#) ( ConfigurationDesk I/O Function Implementation Guide)

Sensor Simulation

New Product: Sensor Simulation

Introduction

In particular for autonomous driving, a vehicle must be equipped with sensors that provide information of the environment of the vehicle. For testing purposes, the sensors must be integrated into the simulation. The following illustration shows the options to do this.



Option 1 and 2 In option 1 and 2, the objects recognized by the sensor ECU are inserted in the simulation using a restbus simulation or an object list. This method is independent of the technology as it implemented in software. This can be done using ASM.

Option 4 In option 4 the real sensor ECU is tested on a test bench that must provide the image of the environment. For example, you can place a camera sensor in front of a display that shows the appropriate image. This can be done using MotionDesk.

Option 3a and 3b In option 3a and 3b, raw data is fed directly into the sensor ECU. This is more related to the measurement principle of the sensor but requires an appropriate interface. The sensor front end is bypassed so no large test bench is necessary. This can be done using Sensor Simulation.

Features of Sensor Simulation

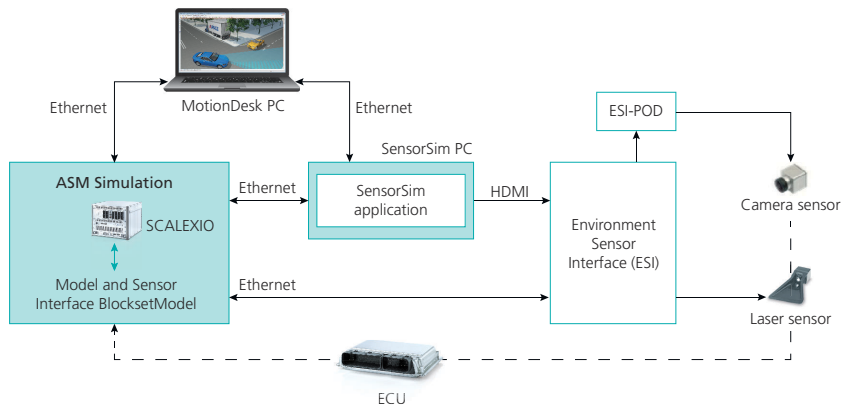
Sensor Simulation provides the following features:

- Supporting camera sensors if a suitable interface is available
- Supporting laser sensors if a suitable interface is available
- Usable in hardware-in-the-loop simulation
- Usable in software-in-the-loop simulation

- Controlled by MotionDesk (for more information, refer to [New Features of MotionDesk 4.3](#) on page 149)

Architecture for HIL

You can perform sensor simulation in a hardware-in-the-loop (HIL) simulation. The following illustration shows a typical architecture for sensor simulation.



Simulator The simulator calculates the environment model for the sensors. In the environment model the **Model and Sensor Interface** blockset must be used. The blockset calculates the motion data and sends it to the **SensorSim** application. It can also receive data coming from the sensors.

The **Model and Sensor Interface** blockset can be used in real-time simulation (as displayed in the example above) and offline simulation.

SensorSim application The **SensorSim** application is an important part of sensor simulation. It calculates the environment data for the sensors based on the scene and the motion data of the simulation. The data for the sensors is output to the **Environment Sensor InterfaceUnit** via the HDMI interface.




Usually, the **SensorSim** application is installed on a Windows PC with a high-quality graphics card. A high computation power is required because the PC calculates the data for all the sensors simultaneously to the simulation.

ESI and ESI pods The **Environment Sensor InterfaceUnit** and **ESI pods** (plug-on device) are the hardware components that make the raw data available for the sensors. The **ESI Unit** is connected to the PC with the **SensorSim** application via the HDMI interface. It gets the image containing all the raw data for the sensors. The **ESI Unit** separates the raw data for each sensor and sends it to the **ESI pods**. The **ESI pods** adapt the raw data coming from the **ESI Unit** to the specific sensor. The adaption requires engineering by dSPACE.

MotionDesk **MotionDesk** is used to specify the properties of the sensors and to control the sensor simulation. It downloads the scene and all the settings to the **SensorSim** application and the **Environment Sensor Interface Unit**. For more information, refer to [New Features of MotionDesk 4.3](#) on page 149.

Related topics

Basics

New Features of MotionDesk 4.3	149
 MotionDesk Sensor Simulation Control	
 Sensor Simulation Hardware and Software Overview	
 Sensor Simulation Manual	

SYNECT

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

New Features of SYNECT 2.6	180
Migrating to SYNECT 2.6	191

New Features of SYNECT 2.6

Where to go from here

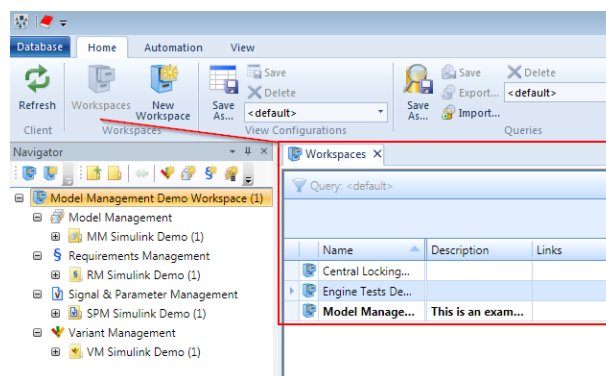
Information in this section

New General Features of SYNECT	180
Provides an overview of SYNECT's new general features.	
New License Type	182
The licensing of SYNECT was changed.	
New Features of Test Management	184
The following improvements have been made for test management.	
New Features of Model Management and Signal & Parameter Management	187
The following improvements have been made for model, signal, and parameter management.	
New Features of Workflow Management	190
The following improvements have been made for workflow management.	

New General Features of SYNECT

Accessing workspaces

SYNECT now provides the Workspaces data grid. This lets you use the features of data grids in SYNECT, such as sorting, grouping, and filtering items for workspaces.



Query improvements

The following improvements were made for queries:

- You can export queries that you stored in the SYNECT database to SQXML files. Items, attributes, conditions, and the name of the query are serialized to the exported file. This lets you exchange queries between databases. You can

import queries from SQXML files. However, you can only import queries to a matching data model. If a query with the same name exists, you can overwrite it.

- You can now add queries to add-ons. Queries are imported when you install the add-on. This lets you use and distribute queries that are designed for specific use cases.
- The server API now provides classes that let you work with all the queries stored in the database. The classes let you access the queries, execute them, and access found items. For reference information, refer to [dSPACE.Synect.Server.Extension.Queries](#) ([SYNECT Server API Reference](#)).
- You can now use attributes that are calculated from other attribute values, such as the Last Verdict and the Last Execution Date of test cases, in queries.

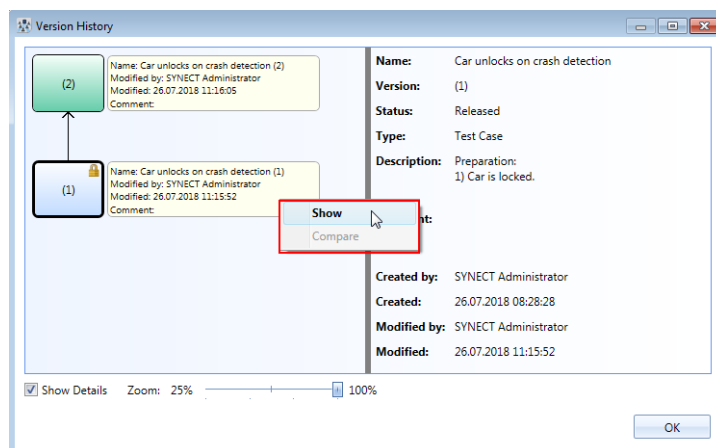
Customizing the SYNECT data model

The following improvements were made for the customization of the SYNECT data model:

- You can now add custom attributes to users and groups. This lets you add information, such as a room number or a department name, to the items.
- You can now reference SYNECT users from items, such as test cases and models, with custom links. This lets you specify which user is responsible for which SYNECT item.

General improvements

- You can now open an item version from the Version History dialog. The item is displayed in a data grid of the project or, if it is not part of the project, in the Global Search pane.



- You can now extend the context menu of the compare view to add commands of your own.

New License Type

Introduction

The new SYNECT user licenses replace the single user licenses on dongles.

SYNECT licenses are now checked by the SYNECT server when a client requests access to data or an operation of the SYNECT server.

This applies to operations such as the following:

- Opening a workspace or project.
- Reading and changing project data, such as test cases or models.

Note

Connecting with a SYNECT server is not protected by license. This allows you to log on a SYNECT server to assign SYNECT user licenses for working with a SYNECT server.

License types

There are the following types of licenses for SYNECT:

Floating network licenses Floating network licenses let you work on a SYNECT server regardless of your SYNECT user login or the host name of the PC you are using. Floating network licenses limit the number of SYNECT users that can work on a SYNECT server concurrently.

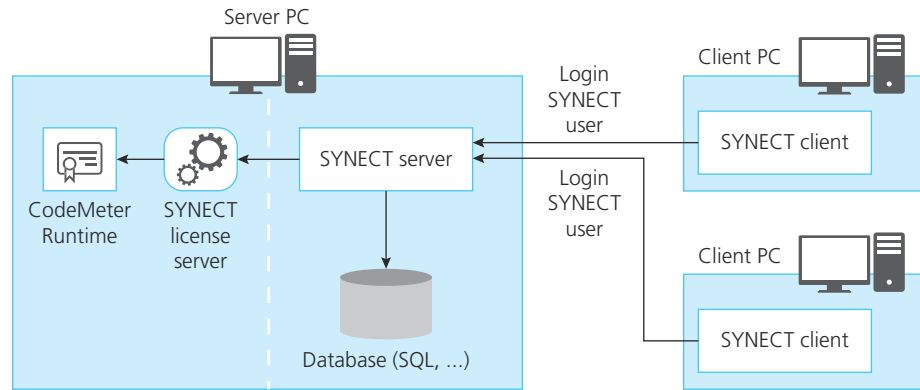
SYNECT user licenses SYNECT user licenses let you work on a SYNECT server with a specific SYNECT user or from a PC with a specific host name. You have to assign SYNECT user licenses to specific SYNECT users or specific PCs before working on the SYNECT server.

Note

The SYNECT user licenses replace the single user licenses on dongles. You have to assign the licenses to work with SYNECT 2.6.

Getting license information

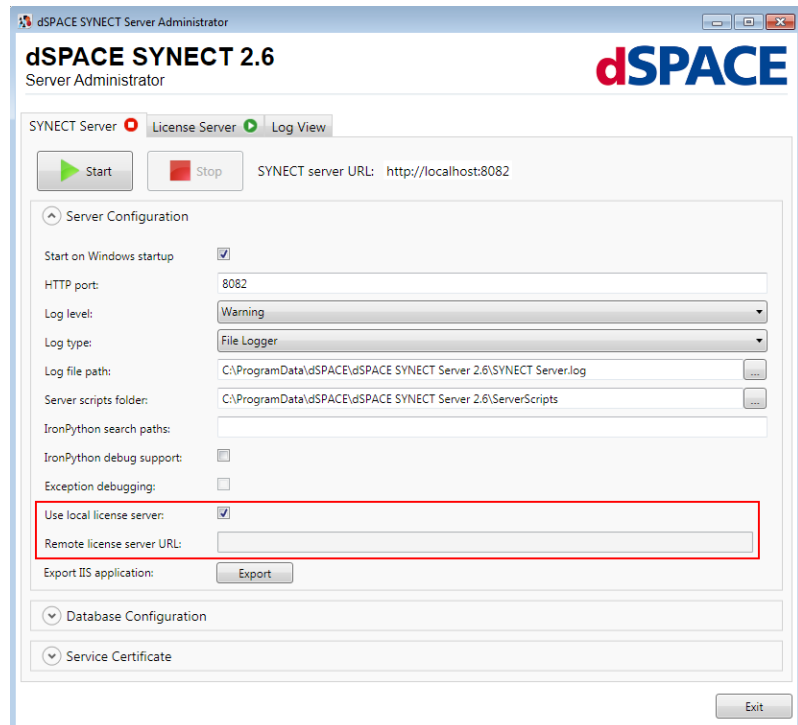
The following illustration shows how the SYNECT server gets license information on a SYNECT user that requests an operation.



You can configure the CodeMeter Runtime with the CodeMeter WebAdmin to let the SYNECT license server access the available SYNECT licenses.

Configuring the SYNECT license server

You can configure and start the SYNECT license server with the SYNECT Server Administrator. The tool also lets you select the SYNECT license server to use to get license information. This lets you work with the SYNECT license server the SYNECT server, and the database server in different topologies that support your SYNECT use case.



Assigning SYNECT user licenses

From the Database ribbon, select User Management – User Licenses. The User Licenses list displays all the SYNECT user licenses that are available.

The screenshot shows the SYNECT software interface with the 'Database' ribbon selected. The 'User Management' section is active, and the 'User Licenses' sub-section is selected. The 'Assign...' button is highlighted with a red box. Below the button is a table listing available licenses.

License ID	License	Product	Assigned to User	Assigned to PC	Last Assignment	Remaining Assignments
	SYNECT_VARIANT_MGMT	SYNECT Variant Management				4
	SYNECT_MODEL_MGMT	SYNECT Model Management				4
	SYNECT_BASE	SYNECT Base				4

To assign a SYNECT user license to a user or a specific PC click Assign. This opens the Assign User License dialog that lets you perform the assignment.

Typically, you need a SYNECT_BASE license to read data and the license of a SYNECT module to write data and perform operation on the SYNECT server.

Note

The permission for managing users and groups is required to assign SYNECT user licenses.

Further reading

Refer to [Basics on License Protection](#) (📖 [The SYNECT Server Guide](#)).

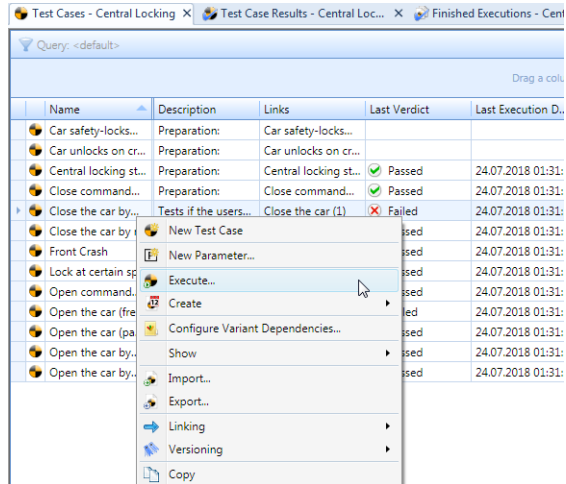
New Features of Test Management

Improvements for test case execution

Executing selected test cases You can now select test cases and execute them without planning the execution. This lets you execute selected test cases quickly when you design test cases.

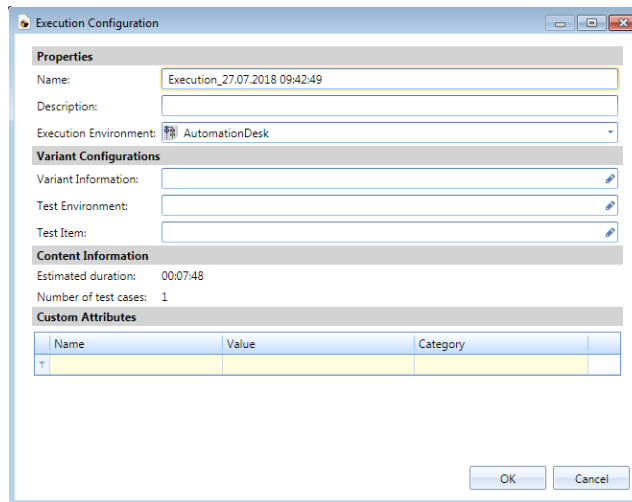
You have to perform the following steps:

1. Select Execute from the context menu of selected test cases that you want to execute.



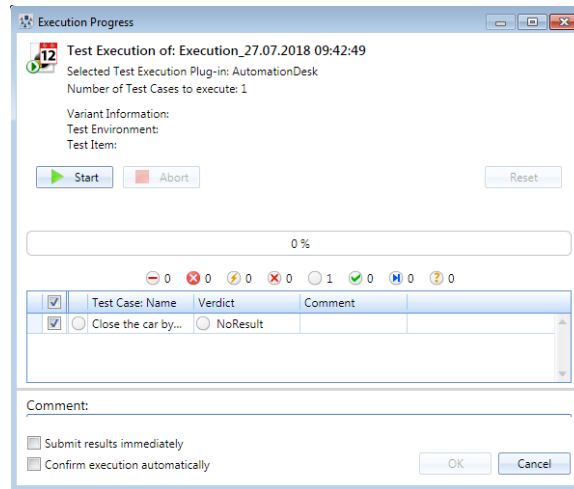
SYNECT opens a dialog.

2. Specify the related execution in the Execution Configuration dialog and click OK.



SYNECT opens a dialog.

3. Start the test case execution.



Creating executions for test case results You can now create executions for test case results. This lets you execute selected test cases again.

You have to perform the following steps:

1. Select Create Execution from the context menu of the test case results that you want to execute again.
SYNECT opens the Execution Configuration dialog.
2. Specify the execution attributes as required and click OK.
SYNECT closes the dialog and opens the Execution Progress dialog.
3. Start the test case execution.

Recurring execution of finished executions You can now create executions on the basis of finished executions. This lets you copy a finished execution to execute it again.

You have to perform the following steps:

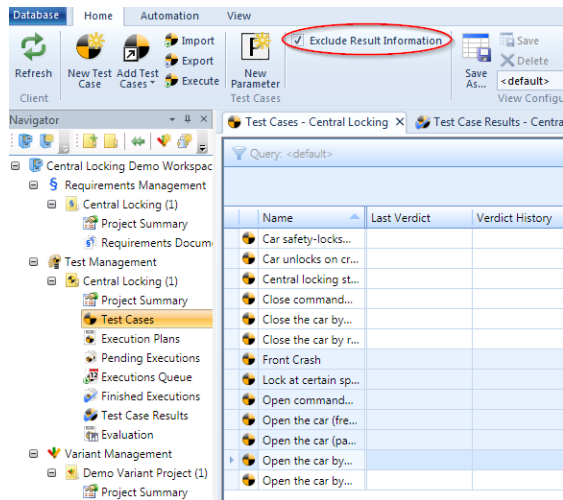
1. Select Create Execution from the context menu of a finished execution.
2. Configure the execution and click OK.
SYNECT creates a planned execution and displays it in the Pending Executions data grid. If the project contains an execution plan related to the execution SYNECT adds the execution to it.

Configuring results

You can specify whether recent test case executions are analyzed in the Test Cases data grid. This lets you open the grid faster if large data volumes have to be analyzed.

This applies to the following derived attributes of test cases:

- Last verdict
- Verdict history
- Last execution date
- # Test case results



New Features of Model Management and Signal & Parameter Management

Introduction

The following improvements have been made for model management and signal & parameter management.

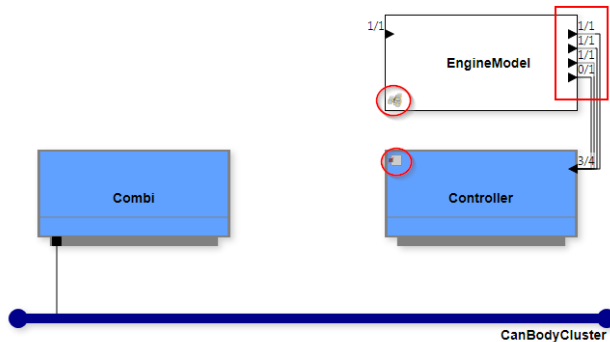
Improvements for diagrams

The handling of diagrams has been improved in the following points:

- Diagrams now provide the following options:
 - Show connection information lets you inspect the signal mapping of the models in a diagram. If you select the option, the number of signals of port

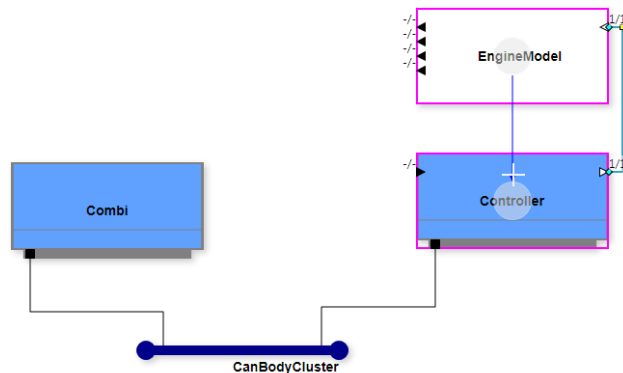
interfaces and of connected signals is displayed. Highlighted for the outputs of EngineModel in the following illustration.

- Show submodel information shows whether a model that is displayed in a diagram has submodels. Highlighted for EngineModel in the following illustration.
- Diagrams now show whether a model has ports or bus access requests that are not displayed in the diagram. Highlighted for Controller in the following illustration.



- If you installed the Auto Connect add-on, you can now connect submodels in a diagram.

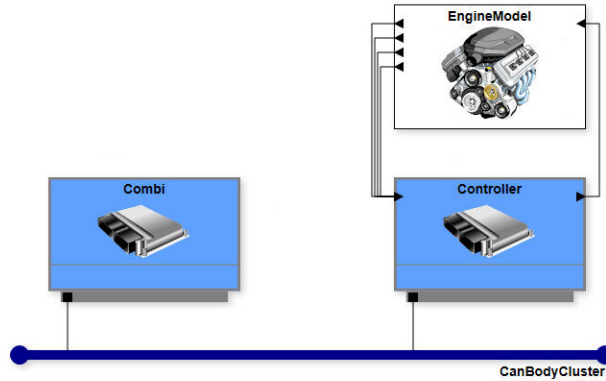
When you drag one model from its center to another model, SYNECT proposes a connection between the models. The Auto Connect add-on creates port connections and signal mappings.



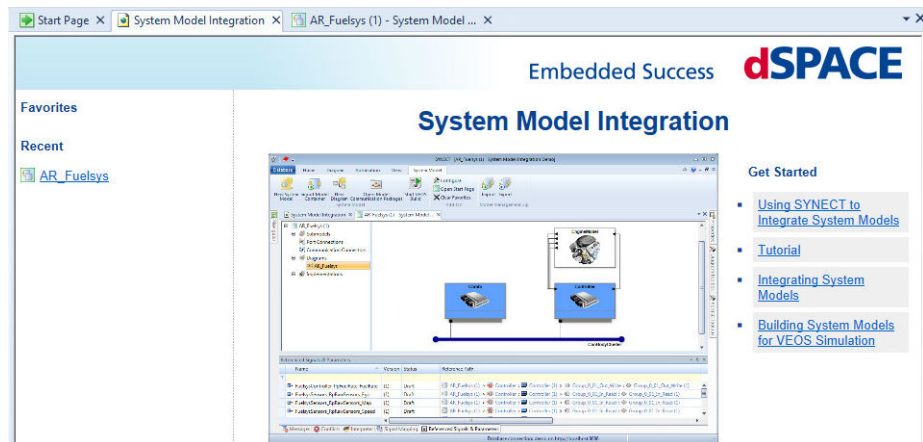
Improvements of the System Model Integration add-on

The following improvements were made:

- SYNECT provides a sample workspace for system model integration. The workspace contains the AR Fuelsys demo model that is used in the Tutorial of the user documentation as an example.

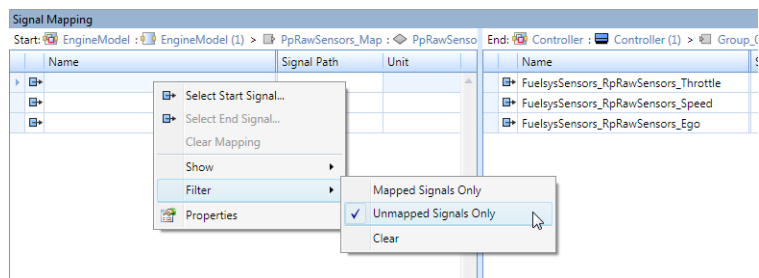


- The add-on now provides a Start page. The page lets you access favorite and recently used system models and the user documentation.



Mapping signals

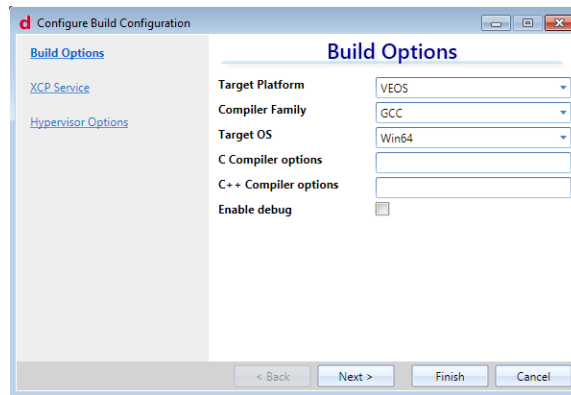
You can now filter the signals that are displayed in the Signal Mapping pane according to the mapping status as shown in the following illustration. This simplifies mapping signals.



Configuring the VEOS build

You can add a build configuration file (BCFG) to the working folder of a model implementation to specify a simulation target for the VEOS build of the related model container. This overrides a default simulation target.

SYNECT now provides commands for adding a BCFG file and for editing the file with a wizard to specify build options.



New Features of Workflow Management

Introduction

The following improvements have been made for workflow management.

- Embedded code in MATLAB and Python workflow steps is now highlighted.
- New parameter type: Search criteria
Lets you find and select parameters by defining a parameter pattern and settings for searching the parameter tree.
- New parameter type: File System Filter
Lets you find and select files by file properties, such as file name or file name extension.
- Workstation: New indicator for the status of jobs and the WFM Job starter. You can now pause an active WFM Job Starter.
- You can now rename instances of a workflow step in a workflow. This helps you with workflows that contain several instances of one workflow step.
- You can now provide custom icons for workflows and workflow steps.
- SYNECT now provides an add-on that supports GIT for workflow steps

Migrating to SYNECT 2.6

Single user licenses The single user licenses on dongles were replaced by SYNECT user licenses. You have to assign the licenses to SYNECT users or PCs to work with SYNECT 2.6. Refer to [New License Type](#) on page 182.

Where to go from here

Information in this section

Migrating Databases	191
To use the data from previous SYNECT versions with SYNECT 2.6, you have to migrate SYNECT's database.	
Migrating Client API Scripts	192
You have to migrate client scripts from SYNECT 2.5 to SYNECT 2.6.	
Migrating Server Scripts of SYNECT 2.5	193
You have to migrate server scripts from SYNECT 2.5 to SYNECT 2.6.	
Migration Scenarios Related to Data Model Changes	195
You have to migrate work products that use the SYNECT data model because of changes introduced with SYNECT 2.6.	
Data Model Changes From SYNECT 2.5 to SYNECT 2.6	195
Some parts of the SYNECT data model have been changed from SYNECT 2.5 to SYNECT 2.6.	

Migrating Databases


Introduction

To use the data from previous SYNECT versions with SYNECT 2.6, you have to migrate SYNECT's database.

To migrate databases for SYNECT Versions 2.0 - 2.5 to SYNECT 2.6, SYNECT 2.6 provides the Database Migrator.

Note

Contact dSPACE Support if you want to migrate SYNECT versions prior to SYNECT 2.0.

For basic information and instructions on migrating databases, refer to [Migrating Databases from Previous SYNECT Versions](#) ( [The SYNECT Server Guide](#)).

Migrating Client API Scripts

Python distribution

The support of Python 2.7 is discontinued with dSPACE Release 2018-B and is replaced by Python 3.6.

You have to migrate client API scripts accordingly. However, the server API IronPython 2.7 support remains unchanged.

For more information, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33.

Extending the context menu of items in the Compare pane

You can now extend the context menu of items in the Compare pane. The arguments that the selected elements provide are different from the arguments of selected elements in other context menus.

Therefore, you now first have to identify the context menu before you can work with the provided arguments.

Old listing The following listing shows an example where the Selection property of the arguments is accessed before the context menu is identified.

```
class ApplicationContextMenuOpening(object):
    """Handle SYNECT applications ContextMenuOpening events."""

    def OnInvoke(self, EventSource, Arguments):
        """Called when an extensible context menu is opened."""
        Arguments = Dispatch(Arguments)
        Menu = Dispatch(Arguments.Menu)
        Selection = Arguments.Selection

        if len(Selection) > 0:
            if Menu.Key == "RequirementsManagement.ProjectSummary":
                Menu.AddSeparator()
                Command = Menu.AddCommand()
                Command.Key = "RequirementsManagement.Project.DemoExtension"
                Command.Caption = "Demo"
            elif Menu.Key == "RequirementsManagement.Documents":
                Menu.AddSeparator()
                Command = Menu.AddCommand()
                Command.Key = "RequirementsManagement.Documents.DemoExtension"
                Command.Caption = "Demo"
```


New listing The following listing shows an example where the context menu is identified before the arguments are used.

```
class ApplicationContextMenuOpening(object):
    """Handle SYNECT applications ContextMenuOpening events."""

    def OnInvoke(self, EventSource, Arguments):
        """Called when an extensible context menu is opened."""
        Arguments = Dispatch(Arguments)
        Menu = Dispatch(Arguments.Menu)
        print(Menu.Key)

        if Menu.Key == "ProjectHandling.Workspaces":
            if len(Arguments.Selection) > 0:
                Menu.AddSeparator()
                Command = Menu.AddCommand()
                Command.Key = "ProjectHandling.Workspaces.DemoExtension"
                Command.Caption = "Demo"
        elif Menu.Key == "ItemCompare.Node":
            print(Arguments.LeftItemId);
            Menu.AddSeparator()
            Command = Menu.AddCommand()
            Command.Key = "Compare.DemoExtension.Node"
            Command.Caption = "Demo Compare Nodes"
        elif Menu.Key == "ItemCompare.Value":
            Menu.AddSeparator()
            Command = Menu.AddCommand()
            Command.Key = "Compare.DemoExtension.Value"
            Command.Caption = "Demo Compare Values"

        # Navigator
        elif Menu.Key == "Navigator.Workspace":
            if len(Arguments.Selection) > 0:
                Menu.AddSeparator()
                Command = Menu.AddCommand()
                Command.Key = "Navigator.Workspace.DemoExtension"
                Command.Caption = "Demo"
```

Compare pane You can now extend the following item views of the Compare pane:

Item View ¹⁾	Menu Key ²⁾
Compare View	
Compare (item tree view)	ItemCompare.Node
Compare (Properties list)	ItemCompare.Value

¹⁾ Specifies the item view (the location of the item's context menu you can extend).

²⁾ Specifies the unique key of the context menu.

Migrating Server Scripts of SYNECT 2.5

Using universal time

You have to use universal time (UTC) objects to program the SYNECT server. However, with SYNECT 2.6 using UTC objects is validated more strictly and exceptions occur on the SYNECT server if you use other objects.

The changes apply to the following properties and methods :

- Planned property of [Execution](#) ([SYNECT Server API Reference](#))
- CreateSchedule(date) method of [ExecutionPlan](#) ([SYNECT Server API Reference](#))
- Create(name, testCaseContexts, plannedDate) method of [TestManagementProject2Executions](#) ([SYNECT Server API Reference](#))

Using .NET universal time objects To avoid problems it is recommended to use UTC objects of the .NET framework.

The following listing shows an example:

```
from System import DateTime
...
execution.Planned = DateTime.Now.ToUniversalTime()
```

Creating test case results

You now have to create test case results in the context of an execution.

SYNECT manages test case results automatically if you plan the test case execution with SYNECT. However, if you do not use SYNECT's planning features and manage test case results via the server API, you have to migrate server scripts.

Note

This applies to server scripts for SYNECT 2.5.

As of May 2018, dSPACE has provided SYNECT 2.5 SP1, which introduced the API changes that are described here.

Outdated server script With SYNECT 2.5, you were able to create test case results in the context of test management projects.

```
tmp2TestCaseResults = myTestmanagementProject.TestCaseResults
tcr = tmp2TestCaseResults.Create("New Test Case Result")
```

Migrated server script With SYNECT 2.6, you have to create test case results in the context of executions.

```
exe2TestCaseResults = myExecution.TestCaseResults
tcr = exe2TestCaseResults.Create("New Test Case Result")
```

The following changes were made to the server API:

- The [TestCaseResults](#) property of the [TestManagementProject](#) class was changed. The property now returns a generic collection of test case results. Refer to [TestManagementProject](#) (refer to [TestManagementProject](#) ([SYNECT Server API Reference](#))).
- The [Execution2TestCaseResults](#) class was changed. It now provides methods to create and delete test case results. Refer to [Execution2TestCaseResults](#) (refer to [Execution2TestCaseResults](#) ([SYNECT Server API Reference](#))).
- The [TestManagementProject2TestCaseResults](#) class was removed from the server API.

Migration Scenarios Related to Data Model Changes

Introduction You have to migrate work products that use the SYNECT data model because of changes introduced with SYNECT 2.6.

Migrating queries You have to migrate queries that use changed parts of the SYNECT data model, such as item types or references that were deleted.

Data Model Changes From SYNECT 2.5 to SYNECT 2.6

Introduction Some parts of the SYNECT data model have been changed from SYNECT 2.5 to SYNECT 2.6.

Deleted item types Item types were not deleted.

Deleted attributes The following attributes were deleted:

Domain	Item Type	Attribute
Workflow Management	Selection Type (SYNECT Data Model Reference)	Selection Values

Deleted reference types The following reference types have been deleted:

Name	Source	Target
Execution/Results	Execution (SYNECT Data Model Reference)	Test Case Result (SYNECT Data Model Reference)
Test Case Results/Project	Test Management Project (SYNECT Data Model Reference)	Test Case Result (SYNECT Data Model Reference)
Favorite VCs/Project	Workflow Management Project (SYNECT Data Model Reference)	Favorite Variant Configuration
Favorite Workflows/Project	Workflow Management Project (SYNECT Data Model Reference)	Favorite Workflow

New item types

The following item types have been added to the SYNECT data model:

- [Appointment](#) ([SYNECT Data Model Reference](#))
- [Calendar](#) ([SYNECT Data Model Reference](#))
- [Selection Type Value](#) ([SYNECT Data Model Reference](#))
- [Workstation Status Item](#) ([SYNECT Data Model Reference](#))

New attributes

The following attributes have been added to the SYNECT data model:

Domain	Item Type	Attributes
Calendars	Appointment (SYNECT Data Model Reference)	<ul style="list-style-type: none"> ▪ Start ▪ End ▪ All Day
Model Management	Diagram Element (SYNECT Data Model Reference)	Z Order Index
Model Management	Diagram Image (SYNECT Data Model Reference)	SVG Image
Model Management	File (SYNECT Data Model Reference)	Generated Date
Model Management	Model Implementation (SYNECT Data Model Reference)	Source Format
Model Management	Model Signal Port Diagram Element (SYNECT Data Model Reference)	Show Port Names
Model Management	Submodel Diagram Element (SYNECT Data Model Reference)	Show Port Names
Test Management	Execution (SYNECT Data Model Reference)	Passed Percentage
Test Management	Execution Plan (SYNECT Data Model Reference)	Estimated Duration
Workflow Management	CM Configuration (SYNECT Data Model Reference)	<ul style="list-style-type: none"> ▪ Use Head Revision ▪ Branch
Workflow Management	Workflow (SYNECT Data Model Reference)	Icon
Workflow Management	Workflow Step Base (SYNECT Data Model Reference)	<ul style="list-style-type: none"> ▪ Requires Write Access ▪ Icon
Workflow Management	Workstation Status Item (SYNECT Data Model Reference)	<ul style="list-style-type: none"> ▪ Last Update Time ▪ Polling Interval ▪ Workstation Status

New reference types

The following reference types have been added to the SYNECT data model:

Name	Source	Target
Test Case Results/Execution	Execution (SYNECT Data Model Reference)	Test Case Result (SYNECT Data Model Reference)
Appointments/Calendar	Calendar (SYNECT Data Model Reference)	Appointment (SYNECT Data Model Reference)
Value/Selection Type	Selection Type (SYNECT Data Model Reference)	Selection Type Value (SYNECT Data Model Reference)
Value/Selection Type Of Default Value	Selection Type (SYNECT Data Model Reference)	Selection Type Value (SYNECT Data Model Reference)
Default Resource/Workflow Management Project Of Default Resource	Workflow Management Project (SYNECT Data Model Reference)	Parameter Context (SYNECT Data Model Reference)
Workstation Status Item/Workstation	Workstation (SYNECT Data Model Reference)	Workstation Status Item (SYNECT Data Model Reference)
Last Test Case Result/Test Case Contexts	Test Case Context (SYNECT Data Model Reference)	Test Case Result (SYNECT Data Model Reference)
Owner Variation Point/All Variants	Variant (SYNECT Data Model Reference)	Variation Point (SYNECT Data Model Reference)

SystemDesk

Where to go from here

Information in this section

New Features of SystemDesk 5.2	200
Migrating to SystemDesk 5.2	205

New Features of SystemDesk 5.2

Where to go from here

Information in this section

New General Features	200
Provides information on new general features.	
Configuring ECUs	200
Provides information on new features for configuring ECUs.	
Managing V-ECUs	201
Provides information on improvements that were made for managing V-ECUs.	
Creating Adaptive Applications	203
With this version, SystemDesk lets you create adaptive applications to execute Classic Platform software components on the Adaptive Platform.	

New General Features

AUTOSAR releases supported by SystemDesk 5.2

AUTOSAR release for modeling SystemDesk lets you model software and system architectures with a data model according to the AUTOSAR 4.3.1 Release. However, SystemDesk lets you exchange data according to other AUTOSAR releases as well.

Data exchange support SystemDesk supports AUTOSAR 4.3.1, 4.3.0, 4.2.2, 4.2.1, 4.1.3, 4.1.2, 4.1.1, 4.0.3, and 4.0.2 for data exchange.

Configuring ECUs

LIN support

dSPACE now provides a LIN driver module (Lin) of the microcontroller abstraction layer. SystemDesk provides support for automatic configuration and code generation. This lets you integrate third-party basic software that exchanges data via LIN buses in V-ECUs.

You can use the following commands for configuring ECU with LIN buses:

Module Definition	Command	Description
AUTOSAR/EcucDefs/Linlf (according to AUTOSAR 4.3.0 and 4.2.2)	Update Linlf Configuration	Lets you auto configure the AUTOSAR LIN interface module configurations according to AUTOSAR upstream mappings.
dSPACE_Sim/Lin	Update Lin Configuration	Lets you auto configure the dSPACE LIN driver MCAL module configuration according to AUTOSAR upstream mappings.
dSPACE_Sim/Lin	Generate Lin Code	Lets you generate code for the dSPACE LIN driver MCAL module according to the parameters in the Lin module configuration.

Specifying memory sections

You can now specify memory sections in the simulator abstraction module. This lets you define the memory segments to use in the variable description of the V-ECU and the mapping to linker sections. For more information, refer to [New Features of VEOS 4.3](#) on page 259.

Further reading

Refer to [Configuring ECUs](#) ([PDF](#) [SystemDesk Manual](#)).

Managing V-ECUs

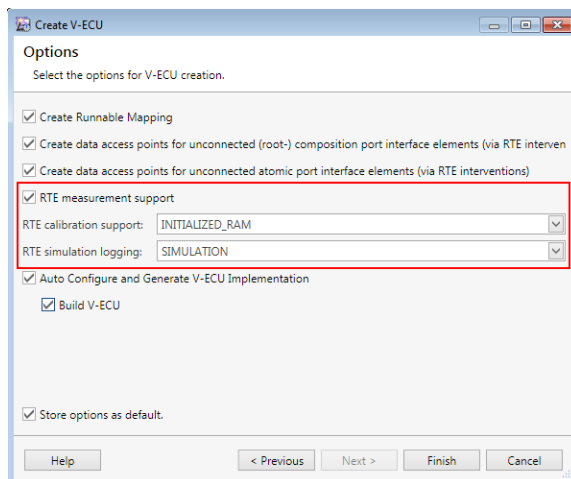
Improved Create V-ECU wizard

The Create V-ECU wizard is a simple method to generate a V-ECU for the simulation of software components (SWCs) or compositions on the virtual functional bus (VFB) level.

With this version, you can configure the wizard to activate measurement support for all the software components with measurement access. You can specify the type of calibration support and the code generation with regard to the measurement and stimulation of variables in a VEOS simulation.

The default V-ECU wizard options give you access to all V-ECU variables for virtual validation.

The following illustration shows the Options page of the Create V-ECU wizard.

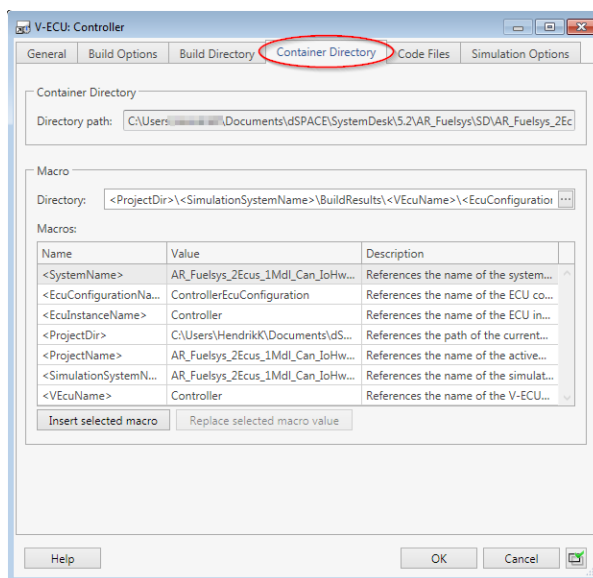


Improved setting of breakpoints

You can now let SystemDesk set breakpoints in OS tasks automatically without opening all the V-ECU files when you debug simulations. Visual Studio is ready to debug a VEOS simulation faster when not all of the V-ECU code files are opened.

Specifying the location of implementation containers

You can now specify the location of V-ECU implementation containers that you built with VEOS on the Container Directory page of the V-ECU Properties dialog. This helps you locate implementation containers that were used for a V-ECU build.



Further reading

Refer to [Creating Simulation Systems for Virtual Validation](#) ([📖 SystemDesk Manual](#)).

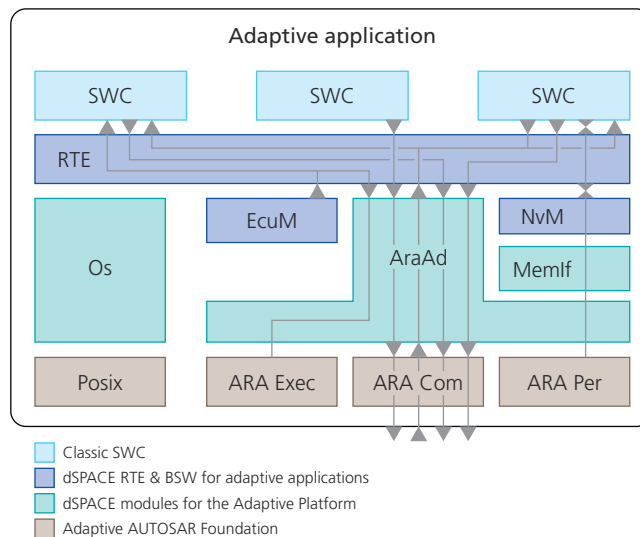
Creating Adaptive Applications

Introduction

With this version, SystemDesk lets you create adaptive applications to execute Classic Platform software components on the Adaptive Platform.

Adapters for Classic Platform software components

The external communication of the Classic Platform software components is performed via the AUTOSAR runtime environment for adaptive applications communication API (ARA COM) that can transfer data via the Ethernet bus. The operating system (OS) and the NVRAM manager are mapped from the Classic Platform to APIs of the Adaptive Platform.



dSPACE provides the following basic software modules to create adaptive applications on the basis of Classic Platform SWCs:

- AraAd: To use the ARA COM API for external communication of Classic Platform SWCs and to provide an adapter for the ARA EXEC and ARA PER API.
- Os: To provide the Classic Platform API of the operating system on the basis of the POSIX interface.
- MemIf: To map memory accesses of the NVRAM manager to the memory interface of the AraAd.

Note

The support of SystemDesk 5.2 for creating adaptive applications based on Classic Platform SWCs is intended only for evaluation projects.

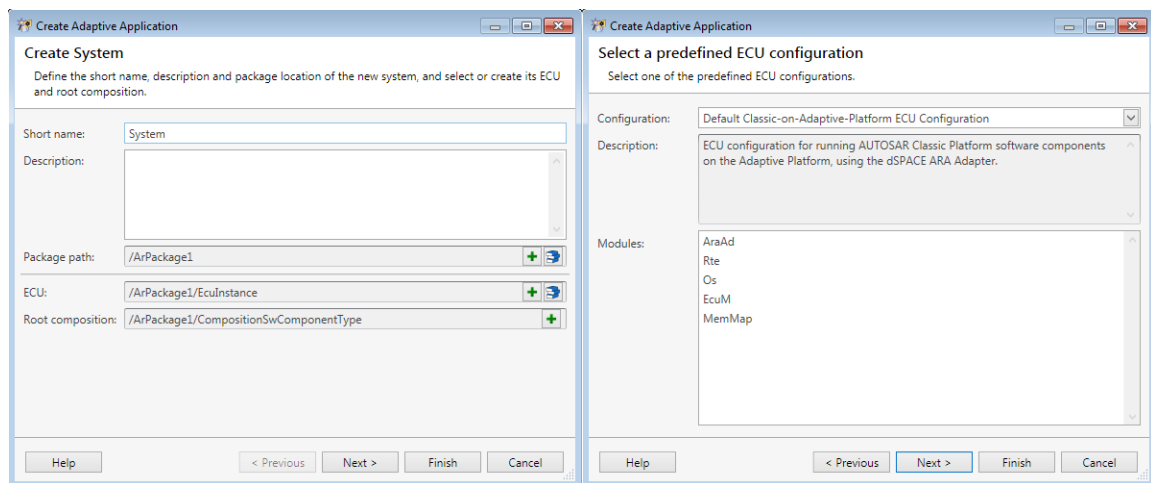
Creating adaptive applications based on Classic Platform software components

You can perform the following steps to create an adaptive application on the basis of Classic Platform SWCs:

1. Specify the Classic Platform software components that you want to execute on the Adaptive Platform and create an ECU configuration for this use case.
2. Configure and generate the required dSPACE RTE, basic software for adaptive applications, and the dSPACE modules for executing Classic Platform SWCs on the Adaptive Platform.
3. Create an adaptive application with executables that are based on a classic-on-adaptive ECU configuration. You can assign build scripts to the executables and build them.

Create Adaptive Application wizard

SystemDesk provides a wizard for creating adaptive applications. If you selected a Classic Platform software component or composition, the wizard lets you create the required elements, such as a system, an ECU configuration, and an adaptive V-ECU. This simplifies step 1 for creating an adaptive application based on Classic Platform SWCs (see above). However, you have to configure the dSPACE ARA adapter and configure and generate the classic-on-adaptive ECU configuration.



Further reading

Refer to [Basics on Adaptive Applications](#) (SystemDesk Manual).

Migrating to SystemDesk 5.2

Migrating to SystemDesk 5.2

Automatic migration

SystemDesk 5.2 automatically migrates SystemDesk 5.0, and 5.1 SDP project files when it loads.

Note

You are recommended to install the most recent patch for SystemDesk 5.0 or 5.1. Then, save the SDP project files you want to migrate before opening them in SystemDesk 5.2.

Migrating projects with Dap module configurations You have to perform manual steps to migrate projects that contain Dap module configurations with DAP blocks with signals in both directions due to a platform compatibility issue regarding VPU ports.

To do so, you have to perform the following steps:

1. Load the project with SystemDesk 5.2.
2. Execute the `Auto configure and generate` command on the ECU configuration or V-ECU that contains the Dap module configuration you have to migrate.

The command adds separate DAP blocks for the signals and updates the data access function references of the DAP user. The names of related VPU ports are changed.

3. Connect the updated VPU ports in VEOS Player.

Python distribution

The support of Python 2.7 is discontinued with dSPACE Release 2018-B and is replaced by Python 3.6.

You have to migrate automation scripts that you run with an external Python interpreter of the dSPACE Python distribution of dSPACE Release 2018-B, such as PythonWin.

For more information, refer to [Migrating Python Scripts from Python 2.7 to Python 3.6](#) on page 33.

Migrating from SystemDesk 5.1

Migrating scripts for automating SystemDesk The SystemDesk API was changed with SystemDesk 5.2. Some interfaces were added with respect to SystemDesk 5.1. A number of interfaces were changed as well.

For information, refer to [API Changes from SystemDesk 5.1 to SystemDesk 5.2](#) ([📖 SystemDesk API Reference](#)).


TargetLink

Where to go from here

Information in this section

New Features of TargetLink 4.4 and TargetLink Data Dictionary 4.4.....	208
Migrating to TargetLink 4.4 and TargetLink Data Dictionary 4.4	223
Changes in Future TargetLink Versions	255

Information in other sections

 [TargetLink New Features and Migration Guide](#)
Provides information on new features, migration steps, discontinuations and code changes of the different TargetLink releases.

New Features of TargetLink 4.4 and TargetLink Data Dictionary 4.4

Where to go from here

Information in this section

Tool Chain Support	208
Modeling in Simulink or Stateflow	209
Code Generation Core Functionality	212
AUTOSAR	213
Target Simulation (PIL)	216
Data Dictionary and Data Management	217
Code Generator Options	218
API Functions and Hook Scripts	219
Other	220

Tool Chain Support

Executing TargetLink Code on dSPACE Real-Time Hardware

TargetLink can now generate SIC files to interoperate with ConfigurationDesk. This lets you execute the production code generated by TargetLink on dSPACE hardware for real-time simulations.

Offline simulation on VEOS is also possible.

Related documentation

- [Interoperating with ConfigurationDesk](#) ( [TargetLink Interoperation and Exchange Guide](#))

Modeling in Simulink or Stateflow

Where to go from here

Information in this section

Support of Arrays of Struct	209
Bus Support of the Constant Block	209
Other MATLAB/Simulink/Stateflow Features	210
TargetLink Module for MATLAB® Code	212

Support of Arrays of Struct

DD-based code generation

TargetLink supports arrays of struct in the generated code as follows:

- Specifying array-of-struct variables in the Data Dictionary
- Specifying initial values in the Data Dictionary
- Generating C modules with global array-of-struct interface variables from the Data Dictionary
- Accessing array elements and leaf struct components in the model via access functions or custom code
- Address-based logging of leaf struct components during SIL/PIL simulation

Related documentation

- [Generating Modules with Global Array-of-Struct Interface Variables](#)
(📖 [TargetLink Customization and Optimization Guide](#))

Bus Support of the Constant Block

Bus-capable Constant Block

TargetLink now supports bus signals in Constant blocks.

You can easily generate a complete Simulink bus by creating a Simulink.Bus object and specifying it as the data type of a Constant block. During model initialization, the Constant block outputs a Simulink bus.

You can also create predefined structs in the TargetLink Data Dictionary, and assign these DD Typedef objects or DD Variable objects at Constant blocks. The mapping is done in the same way as for all bus-capable blocks. For more information, refer to [Mapping Entire Buses to Explicit Structs in the Code](#) (📖 [TargetLink Customization and Optimization Guide](#)).

To specify an initial condition, you can set the **Constant** value on the **Output** page of the **Constant** block to one of the following options:

- To use the default value for every signal, enter **0** (or a MATLAB expression that evaluates to 0).

For numerical data types, the default value is always '0'.

For enumerations, the default value (if specified) or the first value is used.

- To specify a specific initial condition for each bus element, specify a complete initial condition structure. Partial IC structures are not supported at **Constant** blocks. For more information, refer to [Basics on Initializing Buses via Initial Condition Structures](#) ([TargetLink Customization and Optimization Guide](#)).

Obsolete limitation A limitation concerning **Constant** blocks has become obsolete. Refer to [Obsolete Limitations](#) on page 253.

Constant blocks whose data type is specified by using a `Simulink.Bus` object are now supported by TargetLink.

Related documentation

- [Basics on Defining Buses via Constant Blocks](#) ([TargetLink Customization and Optimization Guide](#))
- [Constant Block](#) ([TargetLink Model Element Reference](#))
- [Basics on the Representation of Buses in the Production Code](#) ([TargetLink Customization and Optimization Guide](#))
- [Basics on Initializing Buses via Initial Condition Structures](#) ([TargetLink Customization and Optimization Guide](#))
- `tlSimulinkBusObject`

Related topics

References

[tlSimulinkBusObject](#) ([TargetLink API Reference](#))

Other MATLAB/Simulink/Stateflow Features

Fast Restart mode for iterative model simulations

In **Fast Restart** mode, you can simulate a model without initializing it again. For more information on **Fast Restart**, refer to the Simulink documentation.

TargetLink now supports **Fast Restart** for all three simulation modes (MIL, SIL and PIL), including signal logging, Min/Max logging, plotting, and overflow detection.

The Simulink limitations regarding the modification of models between two simulation runs also apply to TargetLink. However, in MIL simulation mode, you can change TargetLink data, such as the scaling or the data type, before you start a new simulation.

To enable or disable the **Fast Restart** mode, click the **Fast Restart** button on the Simulink Editor toolbar or execute the following API commands:

- `set_param(<model>, 'FastRestart', 'on')`
- `set_param(<model>, 'FastRestart', 'off')`

Obsolete limitation A limitation concerning the fast restart behavior has become obsolete. Refer to [Obsolete Limitations](#) on page 253.

Simulink no longer automatically disables the Fast Restart mode when starting a simulation if at least one TargetLink subsystem is in SIL or PIL mode.

Related documentation

- None

String array support

In addition to character arrays (short sequence of characters (text) enclosed in single quotes), the TargetLink API now also supports text processing for string arrays in MATLAB®. A string array is a container used for larger text sequences (enclosed in double quotes). For details on working with text as data, refer to the MathWorks documentation: [Represent Text with Character and String Arrays](#).

TargetLink 4.4
<pre>t1_get(<block>, "output.type") OR t1_set(<block>, 'output.description', "My description")</pre>

Related documentation

- None

Initialization of function return values for iterated subsystems

You can now specify the block variables of TargetLink OutPort blocks of iterated subsystems as function return values that are initialized variables as well. To achieve this, the variable must have a variable class with the Scope property of the DD VariableClass object set to `fcn_return` and the `InitAtDefinition` property to `on`.

TargetLink 4.4 generates initialized variables that are returned by the function. In TargetLink ≤ 4.3, the `InitAtDefinition` property was ignored and uninitialized variables were returned. Furthermore, TargetLink displayed message [E06514](#) ([TargetLink Message Reference](#)) informing about an invalid DD VariableClass object.

TargetLink ≤ 4.3	TargetLink 4.4
<pre>Int16 Sa2_IterExtern(Int16 Sa2_In1) { Int16 Sa2_Out1; Int32 Sa2_For_Iterator_it; for (Sa2_For_Iterator_it = 1; Sa2_For_Iterator_it <= ((Int32) Sa2_In1); Sa2_For_Iterator_it++) { Sa2_Out1 = [...]; } return Sa2_Out1; }</pre>	<pre>Int16 Sa2_IterExtern(Int16 Sa2_In1) { Int16 Sa2_Out1 = 0; Int32 Sa2_For_Iterator_it; for (Sa2_For_Iterator_it = 1; Sa2_For_Iterator_it <= ((Int32) Sa2_In1); Sa2_For_Iterator_it++) { Sa2_Out1 = [...]; } return Sa2_Out1; }</pre>

Related documentation

- None

TargetLink Module for MATLAB® Code

MATLAB code



MATLAB® code provides functions for a script-based implementation of function algorithms, which is useful for coding algorithms on a textual basis (procedural language) instead of a graphical language. However, not all MATLAB functions are supported by TargetLink. All functions that are fully or partially supported can be used for implementing the function algorithm.

The TargetLink Module for MATLAB Code provides the following features:

- Generate code from MATLAB Function blocks in Simulink® models to extend model-based development with procedural language functions and operators
- Generate code from Stateflow® MATLAB Functions in Stateflow to easily access MATLAB expressions from Stateflow charts
- Full specification of fixed-point and other code properties to generate optimized fixed-point code that complies with the applicable coding guidelines
- AUTOSAR support to create standard-compliant AUTOSAR Software Components from models containing MATLAB code components
- Tight integration with TargetLink features, so you do not have to modify existing workflows
- Call legacy functions from MATLAB code to reuse C code that has already been verified and tested

All features are natively supported by TargetLink, so no separate TargetLink block is required.

Related documentation

- [Basics on Supporting MATLAB® Code Using TargetLink](#) ( [TargetLink Code Generation Guide for MATLAB® Code in Simulink® Models](#))
- [Supported MATLAB® Code Functions](#) ( [TargetLink Code Generation Reference for MATLAB® Code in Simulink® Models](#))

Code Generation Core Functionality

New Value Copy Access Functions

TargetLink now lets you specify value copy access functions as either system inputs or system outputs.

Example

```

Int16 Aux_MyInput[56];
Int16 Aux_MyOutput[20];

GetMyInput(Aux_MyInput);
/* copies contents of MyInput to Aux_MyInput */
...
SetMyOutput(Aux_MyOutput);
/* copies contents of Aux_MyOutput to MyOutput */

```

Related documentation

- [Encapsulating Access to Data Signals and Parameters via Access Functions](#) ([TargetLink Customization and Optimization Guide](#))
- [AccessFunction](#) ([TargetLink Data Dictionary Reference](#))
- [Overview of Predefined Access Function Templates](#) ([TargetLink Data Dictionary Reference](#))

AUTOSAR

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

Supported AUTOSAR Releases	213
Support of Provide-Require Ports for Sender-Receiver Communication	214
Support of Data Store Blocks for Sender-Receiver Communication	214
Support of Explicit NvData Communication	215
Support of Rte_IWrite and Rte_IWriteRef in Sender-Receiver and NvData Communication	215
Exporting Splittable Elements	215

Supported AUTOSAR Releases

Supported AUTOSAR Releases

The following AUTOSAR Releases are supported:

AUTOSAR Release	Revision
4.3	4.3.1 ¹⁾ 4.3.0

AUTOSAR Release	Revision
4.2	4.2.2
	4.2.1
4.1	4.1.3
	4.1.2
	4.1.1
4.0	4.0.3
	4.0.2
3.2	3.2.3
	3.2.2
	3.2.1
3.1	3.1.5
	3.1.4
	3.1.2
	3.1.0
3.0	3.0.7
	3.0.6
	3.0.4
	3.0.2
2.1	2.1.4

¹⁾ New in TargetLink 4.4

Support of Provide-Require Ports for Sender-Receiver Communication

You can now use provide-require ports when modeling sender-receiver communication.

Related documentation

- [Modeling Sender-Receiver Communication](#) ( TargetLink AUTOSAR Modeling Guide)

Support of Data Store Blocks for Sender-Receiver Communication

You can now use data store blocks to model sender-receiver communication.

Related documentation

- [Modeling Sender-Receiver Communication](#) ( TargetLink AUTOSAR Modeling Guide)

Support of Explicit NvData Communication

You can now model explicit NvData communication.

Related documentation

- [Modeling NvData Communication](#) ( [TargetLink AUTOSAR Modeling Guide](#))

Support of Rte_IWrite and Rte_IWriteRef in Sender-Receiver and NvData Communication

TargetLink now supports the `Rte_IWrite` and `Rte_IWriteRef` RTE API functions for both SenderReceiver and NvData communication. You can select the API to generate in the related block dialog.

Related documentation

- [Modeling NvData Communication](#) ( [TargetLink AUTOSAR Modeling Guide](#))
- [Modeling Sender-Receiver Communication](#) ( [TargetLink AUTOSAR Modeling Guide](#))

Exporting Splittable Elements

TargetLink now lets you export the following elements to separate AUTOSAR (ARXML) files:

- `<AR-TYPED-PER-INSTANCE-MEMORYS>`
- `<CONSTANT-MEMORYS>`
- `<EXPLICIT-INTER-RUNNABLE-VARIABLES>`
- `<SWC-INTERNAL-BEHAVIOR>`
- `<IMPLICIT-INTER-RUNNABLE-VARIABLES>`
- `<PER-INSTANCE-PARAMETERS>`
- `<SHARED-PARAMETERS>`
- `<STATIC-MEMORYS>`

Related documentation

- [Introduction to Importing and Exporting AUTOSAR Files](#) ( [TargetLink Interoperation and Exchange Guide](#))
- [AR_POSCONTROL](#) ( [TargetLink Demo Models](#))

Target Simulation (PIL)

Where to go from here

Information in this section

Support for Virtual Evaluation Boards	216
Changes in the Target Simulation Modules	216

Support for Virtual Evaluation Boards

Virtual evaluation boards

TargetLink lets you run a processor-in-the-loop simulation on virtual evaluation boards (instruction set simulators (ISS)), such as the TRACE32 Simulator by Lauterbach.

Related documentation

- [Processor-in-the-Loop Simulation Mode](#) ([📖 TargetLink Preparation and Simulation Guide](#))
- [Virtual Evaluation Boards](#) ([📖 Evaluation Board Reference](#))

Changes in the Target Simulation Modules

New and discontinued compiler versions

The following table shows the compiler versions that are now supported by TargetLink 4.4. Refer to the New and No changes columns. Compiler versions that are no longer supported are listed in the Discontinued column.

Microcontroller Family	Compiler	New	No Changes	Discontinued
ARM CortexM3	Keil	—	5.2	—
C16x	TASKING	—	8.6	—
MPC57xxVLE	Diab	—	5.9	—
	GreenHill	2018	—	2016
MPC560xVLE	Diab	—	5.9	—
	GreenHill	2018	—	2016
RH850	GreenHill	2018	—	2016
S12X	Cosmic	—	4.8	—
	Metrowerk	—	5.1	—
SH2	Renesas	—	9.3	—
SH2A-FPU	Renesas	—	9.4	—

Microcontroller Family	Compiler	New	No Changes	Discontinued
TriCore17xx	TASKING	—	3.2, 6.2	—
TriCore2xx	TASKING	—	6.2	—
	GCC	4.9	—	4.6
V850	GreenHill	2018	—	2016
XC22xx	TASKING	—	3.0	—

For more information on the evaluation boards supported by TargetLink, refer to [Combinations of Evaluation Boards and Compilers](#) ([📄 Evaluation Board Reference](#)).

Note

For more PIL support combinations that are part of a valid Software Maintenance Service (SMS) contract, refer to dSPACE's [TargetLink PIL Support](#) website at the [TargetLink Product Support Center](#).

Data Dictionary and Data Management

Where to go from here

Information in this section

Support of Show Masked Content and Show Library Content in the Property Manager	217
New Import and Export of View Sets in the Property Manager	218
Support for User Mode per Workspace in the Data Dictionary Manager	218

Support of Show Masked Content and Show Library Content in the Property Manager

Show masked content and Show library content in the Property Manager

The TargetLink Property Manager now supports showing or hiding the content of the following model elements in the Model Navigator and the properties of the underlying model elements in the Property View:

- The content of masked subsystems and charts.
- The content of model elements with active library links, including LinkCharts.

Related documentation

- [Show Masked Content](#) ([📖 TargetLink Tool and Utility Reference](#))
- [Show Library Content](#) ([📖 TargetLink Tool and Utility Reference](#))
- [Basics on Working with Libraries](#) ([📖 TargetLink Preparation and Simulation Guide](#))

New Import and Export of View Sets in the Property Manager

Import and export of view sets

The TargetLink Property Manager now supports the import and export of custom property view sets via View Set XML (VSML) files.

Related documentation

- [Import View Sets Dialog](#) ([📖 TargetLink Tool and Utility Reference](#))
- [Export View Sets Dialog](#) ([📖 TargetLink Tool and Utility Reference](#))
- [How to Create Property View Sets](#) ([📖 TargetLink Preparation and Simulation Guide](#))

Support for User Mode per Workspace in the Data Dictionary Manager

One user mode and password per workspace

TargetLink lets you assign a user mode and password per workspace in the Data Dictionary.

Related documentation

- [Users](#) ([📖 TargetLink Data Dictionary Reference](#))
- [GetUser](#) ([📖 TargetLink Data Dictionary Reference](#))
- [SetUser](#) ([📖 TargetLink Data Dictionary Reference](#))
- [SetPassword](#) ([📖 TargetLink Data Dictionary Reference](#))

Code Generator Options

New Code Generator Options

Overview of new Code Generator options

The following new Code Generator options are available with TargetLink 4.4.

- `InitializeVariablesViaRestartFunction`
Lets you globally change restart initialization behavior.

- **RealTimePlatformSupport**
Causes the Code Generator to consider specifications for use with dSPACE ConfigurationDesk and VEOS Player.

Related documentation

- [InitializeVariablesViaRestartFunction](#) (📖 TargetLink Model Element Reference)
- [RealTimePlatformSupport](#) (📖 TargetLink Model Element Reference)

For reference information on all Code Generator options, refer to [Alphabetical List of Code Generator Options](#) (📖 TargetLink Model Element Reference).

Migration aspects of Code Generator options

For more information, refer to [Migration Aspects Regarding Code Generator Options](#) on page 232.

API Functions and Hook Scripts

Where to go from here

Information in this section

New API Functions	219
New Hook Scripts	220
Improved Hook Scripts	220

New API Functions

List of new API functions

API Function	Purpose
<code>tlGenerateSic</code>	Generates a dSPACE Simulink implementation container using the TargetLink Code Generator.

Related documentation:

- [tlGenerateSic](#) (📖 TargetLink API Reference)
- [Interoperating with ConfigurationDesk](#) (📖 TargetLink Interoperation and Exchange Guide)

With TargetLink 4.4, API functions have changed. Refer to [Changes in API Functions](#) on page 235.

New Hook Scripts

No new hook scripts were introduced with TargetLink 4.4.

Improved Hook Scripts

`tl_pre_compile_host`

TargetLink supports parallel compiling for SIL simulation. You can disable or enable the parallel compilation by means of the `EnableParallelCompiling` option of the `tl_pre_compile_host` hook script.

Related documentation

- `tl_pre_compile_host_hook` ([📖 TargetLink File Reference](#))

Other

Where to go from here

Information in this section

General Enhancements and Changes	220
TargetLink Demos	221
Synchronizing Simulink and TargetLink	222

General Enhancements and Changes

Blocks not supporting matrix signals

In TargetLink 4.4, blocks that do not support matrix signals reduce a [1 x 1] signal to a scalar. In this case, code generation is now possible.

Related documentation

- [Block-Specific Limitations](#) ([📖 TargetLink Limitation Reference](#))

Simulink parameter-based differences between MIL and SIL simulations

If you enable the `Use algorithms optimized for row-major array layout` Simulink configuration parameter to enable efficient algorithms, this might result in numerical differences between simulation and generated code. However, TargetLink does not support the `Use algorithms optimized for row-major array layout` Simulink configuration parameter.

Block and Object Reference renamed and reorganized

The *Block and Object Reference* user documentation was renamed to *Model Element Reference*. Additionally, it now focuses on the TargetLink properties of TargetLink blocks and Stateflow objects. TargetLink blocks and Stateflow objects are model elements that have TargetLink-specific properties. These properties influence how TargetLink generates production code from the model.

TargetLink Demos

New demo models

The following new demo models are available for TargetLink 4.4:

MATLAB_CODE_STATEFLOW The MATLAB_CODE_STATEFLOW demo model demonstrates fixed-point code generation of MATLAB functions.

Refer to [MATLAB_CODE_STATEFLOW](#) ([📖 TargetLink Demo Models](#)).

MATRIX_INVERSE The MATRIX_INVERSE demo model shows two possibilities for matrix inversion:

- A block with only one output, which simply calculates and returns a matrix - without any error check.
- A block with an additional second output that returns a status indicating whether it was possible to calculate the inverse matrix.

Refer to [MATRIX_INVERSE](#) ([📖 TargetLink Demo Models](#)).

REAL_TIME_PLATFORM_SUPPORT The REAL_TIME_PLATFORM_SUPPORT demo model shows how to prepare a TargetLink subsystem so you can use it as a behavior model (i.e., a SIC file) in ConfigurationDesk. In particular, it shows how to model the subsystem's interface via model port variables, how to model multiple SIC runnable functions that can be connected to different tasks (asynchronous or time-triggered) in ConfigurationDesk, and how to ensure a secure communication between the tasks in a real-time simulation.

Refer to [REAL_TIME_PLATFORM_SUPPORT](#) ([📖 TargetLink Demo Models](#)).

Extended demo models

The following demo models have been extended for TargetLink 4.4:

AR_POSCONTROL New feature: Settings for AUTOSAR export, i.e., files are exported as splittables.

Refer to [AR_POSCONTROL](#) ([📖 TargetLink Demo Models](#)).

BUS_CC New feature: The specification of array-of-struct variables as DD-based objects is explained. You can access field elements of array-of-struct variables by means of access functions for a model-based code generation unit.

Refer to [BUS_CC](#) ([📖 TargetLink Demo Models](#)).

Synchronizing Simulink and TargetLink

Preparing, synchronizing, and clearing Simulink models

TargetLink 4.4 lets you synchronize referenced Simulink library blocks with TargetLink data from a specific library block instances.

Related documentation

- [System Preparation Tool](#) (📖 [TargetLink Tool and Utility Reference](#))
- [How to Synchronize Simulink and TargetLink Data](#) (📖 [TargetLink Preparation and Simulation Guide](#))

Migrating to TargetLink 4.4 and TargetLink Data Dictionary 4.4

Upgrade process Carefully read all of the following information and modify the tool chain accordingly.

Where to go from here	Information in this section
	<ul style="list-style-type: none"> General Migration Information 223 Migrating from TargetLink 4.3 to 4.4 232 Code Changes 236 Discontinuations 252

General Migration Information

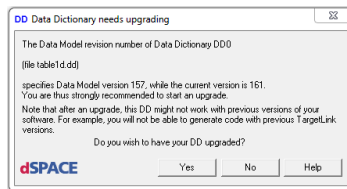
Upgrading Models, Libraries, and Data Dictionaries

Where to go from here	Information in this section
	<ul style="list-style-type: none"> Basics on Migrating Between TargetLink Versions 223 How to Upgrade a Data Dictionary with Included DD Files 226 How to Manually Upgrade Libraries and Models via the API 228 Migrating Data Dictionaries to CodeDecorationSets 229

Basics on Migrating Between TargetLink Versions

Automatic upgrade from TargetLink 3.1 or later TargetLink 4.4 automatically upgrades models, TargetLink-compliant libraries, and Data Dictionaries if they were created with TargetLink 3.1 or later. You are

prompted for the automatic upgrade when the Data Dictionary is opened with TargetLink 4.4 for the first time. For example:



The automatic upgrade comprises all the steps required by the intervening TargetLink versions. For example, an automatic upgrade from TargetLink 4.0 to TargetLink 4.4 comprises the steps 4.0 to 4.1 to 4.2 to 4.3 to 4.4.

Note

Check the TargetLink migration documentation of the different TargetLink versions to see whether user interaction is required.

User interaction required In the following cases, for example, the automatic upgrade requires additional user interaction:

- Libraries must be TargetLink-compliant. Otherwise, an upgrade is not possible at all.
- To upgrade DD files with included partial DD files, refer to [How to Upgrade a Data Dictionary with Included DD Files](#) on page 226.
- Style sheets for code generation are version-specific and subject to change from one TargetLink version to another. Thus, modified style sheets of older TargetLink versions have to be updated to match the current version (reapplying the modifications as intended).
- Custom code S-functions built with 32-bit TargetLink versions do not work with 64-bit versions of TargetLink and vice versa.

Initiate a rebuild of all custom code S-functions using the


`tlUpgrade('Model', <MyModel>, 'CheckModel', 'FixIssues')` API function.

Making new libraries TargetLink-compliant

Libraries that you create from scratch and that consist of TargetLink blocks, must be made upward compatible so that you can upgrade them to a newer TargetLink version in the future. Otherwise, an upgrade will not be possible.

Note

A library does not become a TargetLink library just because it contains TargetLink blocks. The library itself must be TargetLink-compliant.

Refer to [How to Make TargetLink User Libraries Upgrade-Capable](#) ( TargetLink Orientation and Overview Guide).

Making old libraries TargetLink-compliant

The following two approaches let you make libraries created with older TargetLink versions compliant with the current TargetLink version 4.4:

The old TargetLink version is available Use the old TargetLink version which the library was created with to make the library TargetLink-compliant. Refer to the TargetLink migration documentation of the old TargetLink version. You can then use this library with all the later TargetLink versions because TargetLink automatically performs an upgrade. The library can still be used with TargetLink versions earlier than TargetLink 4.4 because the automatic upgrade does not save a library in the newer TargetLink version.

Only the current TargetLink version 4.4 is available Use TargetLink version 4.4 and the `tlUpgrade` API command to make the library TargetLink-compliant. Refer to [How to Manually Upgrade Libraries and Models via the API](#) on page 228. If you follow the instructions, the library is saved in TargetLink version 4.4 and, hence, cannot be used with TargetLink versions earlier than TargetLink 4.4.

Manual upgrade from TargetLink 2.x or 3.0.x

Models, libraries, and Data Dictionaries created with TargetLink versions 2.x or 3.0.x have to be upgraded manually to the highest TargetLink version 3.x (3.1...3.5) you have. Afterwards, an automatic upgrade is possible.

Using TargetLink models with earlier TargetLink versions

It is not possible to use models in the format of newer TargetLink versions.

Data model filter rule files

Existing data model filter rule files can contain invalid elements, because the data model of the TargetLink Data Dictionary changed. The following files that were shipped with previous TargetLink versions can be affected:

- `DD_Filter_Admin.xml`
- `DD_Filter_AR_User.xml`
- `DD_Filter_NonAR_NonRTOS_User.xml`

You can check filter rule files via the API in the MATLAB Command Window:

Checking a Single File	Checking Filter Rule Sets ¹⁾
<pre>dsdd_free; dsdd('ReadFilterRuleSet', 'file', '<myFile>.xml'); ds_error_register(dsdd('GetMessageList')); ds_msgdlg('update');</pre>	<pre>dsdd_free; dsdd('ReloadFilterRuleSets'); ds_error_register(dsdd('GetMessageList')); ds_msgdlg('update');</pre>

¹⁾ All the files contained in the directory defined in Data Dictionary - Filter Rules in the Preferences Editor.

TargetLink informs you about errors in TargetLink's Message Browser. Each error contains the following information so that you can fix it in an XML-capable editor of your choice:

- File name
- Row number
- Column number

Related topics

Basics

[Basics on Code Formatting](#) (📖 TargetLink Customization and Optimization Guide)

HowTos

[How to Make TargetLink User Libraries Upgrade-Capable](#) (📖 TargetLink Orientation and Overview Guide)
[How to Manually Upgrade Libraries and Models via the API](#) 228

References

[tlUpgrade](#) (📖 TargetLink API Reference)

How to Upgrade a Data Dictionary with Included DD Files

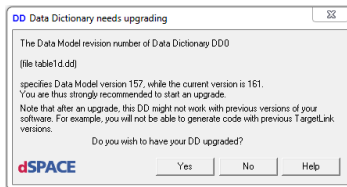
Objective

If you open a TargetLink model with an old Data Dictionary file that was not upgraded, you have to upgrade the Data Dictionary file.

Method

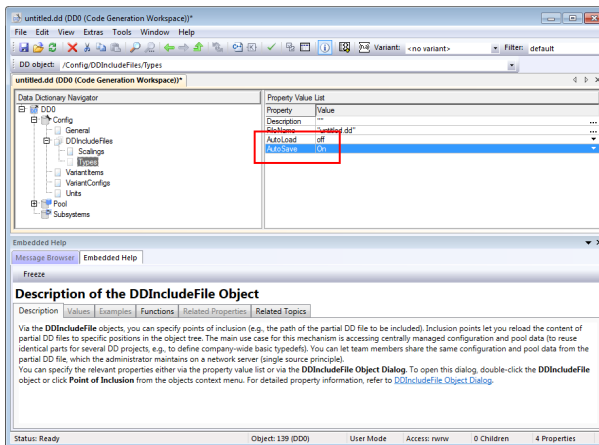
To upgrade a Data Dictionary with included DD files

- 1 Open the model and the referenced TargetLink Data Dictionary, or type `dsdd('Open', <DDFile>)` in the MATLAB Command Window. The Data Dictionary needs upgrading dialog automatically opens if an earlier DD version is involved.



- 2 Select No in the upgrade dialog.

- 3 Under /Config/DDIncludeFiles, set the AutoLoad and AutoSave properties for each included DD file as shown in the following screenshot.

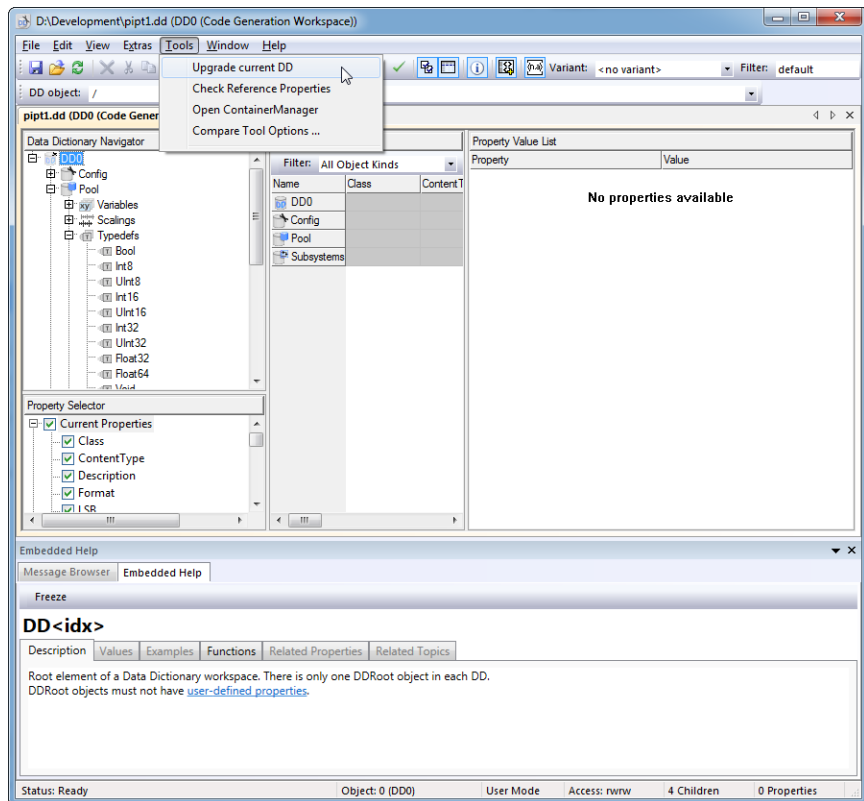


This ensures that after the Data Dictionary and the included DD files were upgraded, the included DD files that were upgraded are saved at the same time the Data Dictionary is saved. You can set these properties for a large number of included DD files via the Object Explorer.

Tip

You can also use the Point of Inclusion dialog to set the included DD file properties.

- 4 Start the Data Dictionary upgrade (with the included DD files) via Tools – Upgrade current DD in the Data Dictionary Manager, or enter `dsdd('Upgrade')` in the MATLAB Command Window.



- 5 Save the Data Dictionary with write permission to the relevant DD file. This completes the upgrade of the DD file and the included partial DD files.

Result

When you open the DD file again, the upgrade dialog does not open, because the DD file and the included partial DD files are up to date. After the files are properly upgraded, you might want to restore the old settings for the included DD files.

How to Manually Upgrade Libraries and Models via the API

Objective

To prepare a central upgrade of libraries and models in a tool chain scenario with several users, for example.

Preconditions

The model or library files are available on MATLAB's search path but they are not open.

The required and upgraded DD project file has been opened, for example, via `dsdd_manage_project('Open', '<name>.dd')`. Upgrading DD project files is possible via `dsdd('Upgrade'[, <DD_Identifier>])`.

Method

To manually upgrade libraries and models via the API

1 Type the following API command in the MATLAB Command Window:

```
tlUpgrade('Model', '<Model|Library>.mdl',
'CheckModel', 'FixIssues')
```

The model or library is upgraded.

Note

When upgrading models and libraries, first upgrade models or libraries that do not reference any other libraries, i.e., the blocks and subsystems they contain have no links to other libraries. Start with the bottom library and then upgrade the libraries above it in ascending order.

2 Save the upgraded model or library files, e.g., **Library.mdl**.

3 Repeat steps 1 and 2 for all other models or libraries.

Result

You upgraded your models and libraries.

Related topics

References

[tlUpgrade](#) (📖 TargetLink API Reference)

[Upgrade](#) (📖 TargetLink Data Dictionary Reference)

Migrating Data Dictionaries to CodeDecorationSets

Introduction of CodeDecorationSet and CodeDecoration objects

TargetLink 4.3 introduced DD CodeDecorationSet and CodeDecoration objects.

Additionally, several properties were removed from the Data Dictionary's data model:

DD Object	Change	Replacement
FunctionClass	Removal of the DeclarationStatements and SectionName properties.	The DeclarationStatements and SectionName properties of the DD CodeDecoration.Settings object.
VariableClass		
VariableClassTemplate.Filter	Removal of the WidthSpec property.	The WidthSpec property of the DD CodeDecoration.Filter object.

Automatic upgrade by TargetLink

Limitation TargetLink no longer supports width-specific type prefixes for variable classes. The automatic upgrade of the Data Dictionary fails if the original

Data Dictionary contains variable class templates used to derive variable classes that have width-specific type prefixes.

Use declaration statements instead.

When opening a Data Dictionary whose data model is older than the latest revision, TargetLink prompts you to perform an automatic upgrade.

Object Kind	Trigger	Upgrade Action
VariableClass FunctionClass	DeclarationStatements or SectionName properties are set.	<ol style="list-style-type: none"> 1. Creating a DD CodeDecorationSet object. 2. Creating a single DD CodeDecoration object for each DD CodeDecorationSet object. The settings of the CodeDecoration object and its child objects match the settings of the original objects. 3. Referencing the CodeDecorationSet object at the original object.
SubStructTemplate	Filter.VariableClass is set.	Transfer the values of the following properties from the variable class to the SubStructTemplate object's filter: <ul style="list-style-type: none"> ▪ DeclarationStatements ▪ SectionName ▪ TypePrefix
VariableClassTemplate	<ul style="list-style-type: none"> ▪ Filter.FilterCondition property is set to ALL_TRUE. ▪ Settings.VariableClass references a DD VariableClass object whose DeclarationStatements or SectionName properties are set. ▪ The Filter.WidthSpec property is set for this DD VariableClassTemplate object or for another VariableClassTemplate object whose Filter.VariableClassSpec property has the same value. 	<ol style="list-style-type: none"> 1. Create a new DD VariableClass object in /Pool/VariableClasses/Templates. 2. Create a new DD CodeDecorationSet object in /Pool/CodeDecorations/Templates. 3. For each VariableClassTemplate object with the same value at the Filter.VariableClassSpec property, adding a CodeDecoration object to the CodeDecorationSet object. 4. Specifying the CodeDecoration object as required. 5. Referencing the CodeDecorationSet object at the VariableClass object created in step 1. 6. Referencing the VariableClass object created in step 1 via the VariableClassTemplate.Settings.VariableClass property.

Special considerations for variable class templates

If you specified DD VariableClassTemplate objects whose Filter.FilterCondition property is set to ALWAYS or NEVER, TargetLink deletes the object's Filter.WidthSpec property during upgrade without replacement.

If you want to keep the property's value, set the DD VariableClassTemplate object's Filter.FilterCondition property to ALL_TRUE before upgrading the Data Dictionary.

Limitation TargetLink does not upgrade DD VariableClassTemplate objects whose Filter.FilterCondition property is set to ONE_OR_MORE or ALL_FALSE.

Cleaning

The automatic upgrade retains the functionality that was specified in the old Data Dictionary. You can clean it manually to reduce the number of objects in the new Data Dictionary.

Merging width-specific variable classes If your old Data Dictionary contained width-specific `VariableClassTemplate/VariableClass` objects, your new Data Dictionary still contains all these variable classes.

Because the width-specific information is now stored in DD `CodeDecoration` objects, you can manually reduce the number of `VariableClass` objects in the Data Dictionary. For example, if you used variable classes in the form of `<Name>_<width>` you can replace them by a single variable class `<Name>` that references a suitable code decoration set.

Two methods are possible:

- Merging code decoration sets:
 1. Copy all the DD `CodeDecoration` objects that were generated during the upgrade for each variable class called `<Name>_<width>` to a single `CodeDecorationSet` object.
 2. Make each `CodeDecoration` object width-specific via its filter.
 3. Reference the resulting `CodeDecorationSet` object at the variable class `<Name>`.
- Using a code decoration set created for variable class templates:
 1. If the original `<Name>_<width>` variable classes were referenced by variable class templates, the DD upgrade automatically creates a width-specific code decoration set in `/Pool/CodeDecorationSets/Templates` for you to use.
 2. You can reference this code decoration set at the resulting variable class called `<Name>`.

Note

Replace references from model elements to the variable classes called `<Name>_<width>` with references to `<Name>`.

Retarget variable class templates After you merged the old width-specific variable classes, you can use them again as the target of your variable class templates. You can then delete all the variable classes contained in `/Pool/VariableClasses/Templates` that were created during the upgrade.

Simplifying user-specified scope reduction chains (SRC) If you used a user-specified SRC to specify declaration statements or section names for variables with specific scopes, you can do the following:

1. Adjust the `Filter.ScopeSpec` property of the code decoration that belongs to the set referenced by the first variable class in the SRC (highest scope) as required.
2. Delete the other variable classes of the SRC.
3. If you also used the SRC to prevent static local variables, you can now use the `AvoidStaticLocalScope` Code Generator option instead.

Remove obsolete variable class templates Find DD VariableClassTemplate objects with the same value of the Filter.VariableClassSpec property and delete all but one.

Changes in the generated production code

Changes in CodeDecoration objects can influence the generated production code mainly in the following respects:

- Changed code comments
- Sorting of variable definitions

Refer to [Code Changes](#) on page 236.

Migrating from TargetLink 4.3 to 4.4

Where to go from here

Information in this section

Code Generator Options	232
API Functions and Hook Scripts	235
Messages	235
Other Migration Aspects	236

Code Generator Options

Migration Aspects Regarding Code Generator Options

Removed Code Generator option

The following Code Generator options were removed from TargetLink:

Removed Option	Additional Information
CreateRestartFunctions	As a replacement, you can use the new InitializeVariablesViaRestartFunction option.

Changed Code Generator options

Code Generator Option	Old Default	New Default
None	-	-

Recommended compatibility settings

CreateRestartFunctions option	InitializeVariablesViaRestartFunction option	Additional information
off	Never	-
on	AsSpecified	This is the default setting.

Basics on changed defaults

The settings of the Code Generator options are stored with the model (model-based option storage). In addition, you can store user-defined sets of Code Generator options in DD CodegenOptionSet objects (DD-based option storage). You can use DD CodegenOptionSet objects as a central source for overwriting and replacing the model-based option settings that had been used since TargetLink 4.1.

If a model-based option value equals the old default value, it is automatically changed to the new default value during the upgrade. If a DD-based option value equals the old default value, it is not changed to the new default value during the upgrade but keeps the old value.

Option value = old default If Code Generator options were set to default values in the former TargetLink version, and the new TargetLink version uses modified default values, note the following points:

- Model-based option:
 - If you want to keep the old default values, you must reset them manually.
- DD-based option:
 - If you want to use the new default values, you must adjust them manually.

The following table is an example describing the impact of a TargetLink upgrade (TargetLink_{Old} to TargetLink_{New}) on three arbitrary option values: 9, 11, and 13. The table illustrates two basic migration scenarios:

- Scenario #1: New default = old default
 - The default value of a Code Generator option has not changed in the new TargetLink version, i.e., the default value remains 9.
 - None of the option values is changed.
- Scenario #2: New default ≠ old default
 - The default value of a Code Generator option changed with the new TargetLink version, i.e., the default value changed to 11.

Option Storage	Option Value (TargetLink _{Old}) Default = 9	Option Value (≤ TargetLink _{New})	
		Default = 9 (Scenario #1)	Default = 11 (Scenario #2)
Model-based	9 ¹⁾	9 ¹⁾	11 ²⁾
	11	11	11 ¹⁾
	13	13	13
DD-based	9	9	9 ³⁾
	11	11	11
	13	13	13

¹⁾ The option value is not stored with the model because it equals the default.

²⁾ Manual reset might be necessary.

³⁾ Manual adjustment might be necessary.

Option value = new default If the Code Generator options were not set to default values in the former TargetLink version (A) but are in the new TargetLink version (B), TargetLink assumes that you intentionally specified the default value in the new TargetLink version. The same applies if the default changes again in the next TargetLink version (C).

Note

Upgrading TargetLink_A ⇒ TargetLink_B ⇒ TargetLink_C and upgrading TargetLink_A ⇒ TargetLink_C can cause different option values. Refer to the following table.

If the default values for TargetLink versions A, B, and C read 9, 11, and 13, and an option was set to 11 in version A, an upgrade to version C changes the option value as follows:

Upgrade Strategy	Option Value TargetLink _A Default = 9	Option Value TargetLink _B Default = 11	Option Value TargetLink _C Default = 13
A ⇒ B ⇒ C	11 (≠ default)	11 (= default) ¹⁾	13 (= default) ¹⁾
A ⇒ C	11 (≠ default)	—	11 (≠ default)

¹⁾ The option value is not stored with the model because it equals the default.

New Code Generator options

For more information on new Code Generator options, refer to [New Code Generator Options](#) on page 218.

API Functions and Hook Scripts

Where to go from here

Information in this section



Changes in API Functions	235
Changes in Hook Scripts	235

Changes in API Functions

tl_get_config_path

Migration issue All paths must be defined as absolute paths. Do not use .. in conjunction with MATLAB's fullfile and addpath API commands.

Related documentation

- [tl_get_config_path](#) ( [TargetLink API Reference](#))
- [How to Define TargetLink's Search Path for M-API-Related Customization Files](#) ( [TargetLink Customization and Optimization Guide](#))

With TargetLink 4.4, new API functions were introduced. Refer to [New API Functions](#) on page 219.

Changes in Hook Scripts

No hook scripts need to be migrated to TL 4.4.

Messages

Message changes

Changed message type

The following messages changed their type to **Advice** in TargetLink 4.4 because they only inform you about your responsibility for the initialization of external variables:

Old Message Number	New Message Number
W17055	A17055 (📖 TargetLink Message Reference)

Other Migration Aspects

Various Migration Aspects

Rate Transition block default settings

With TargetLink 4.4, the default settings of the Rate Transition block have changed. The checkboxes for Ensure data integrity during data transfer and for Ensure deterministic data transfer (maximum delay) are now cleared by default if you use the Rate Transition block via the TargetLink block library.

TargetLink does not support these settings. Before starting code generation, TargetLink checks if the checkboxes are selected. If so, TargetLink displays message W03839.

Migration issue None

Related documentation [Code-Relevant Simulink Blocks](#) ([📖 TargetLink Model Element Reference](#))

Non-TargetLink-compliant libraries and referenced models

You must not use libraries or referenced models that are not TargetLink-compliant. If you do use them, a message occurs in the System Checker.

Related documentation E03887

[Basics on Migrating Between TargetLink Versions on page 223](#)

Code Changes

Where to go from here

Information in this section

Access Functions	237
AUTOSAR	239
Block-Specific Code Changes	240

Efficiency	244
Inheritance of Struct Data Types	245
Mixed Operations (Floating-Point and Fixed-Point Types)	247
Stateflow	248
Other	249

Access Functions

Names of auxiliary variables for access functions and NvData communication

For certain access-function-related code patterns, TargetLink generates variables of global scope and names them according to this naming schema: `Aux_<OriginalVariableName>_I$R`. TargetLink 4.4, generates fewer auxiliary variables of global scope and uses the following naming schema `Aux_<OriginalVariableName>$R`:

TargetLink ≤ 4.3	TargetLink 4.4
<code>MyType Aux_OriginalVariableName_a;</code>	<code>MyType Aux_OriginalVariableName;</code>

Additionally, TargetLink 4.4 sometimes generates auxiliary variables for vectors as `_tmpu_<Originalvariablename>$R` instead of `_tmpd_<Originalvariablename>$R`.

Reason Code efficiency

Migration issue None

Initialization of auxiliary variables

The code changed in contexts where auxiliary variables are used for encapsulation. The auxiliary variables now more frequently have the following characteristics:

- Their storage duration is static.
- They are initialized in the same way as the encapsulated variable.

This code difference is primarily visible in unoptimized code.

Reason Code efficiency

Migration issue None

Optimization of variables with restart initialization or access function template

The code changed for user-specified scope reduction chains whose DD VariableClass objects have different settings regarding the ERASABLE value of

their **Optimization** property. This inconsistent specification can result in the addition or removal of dead code in the form of:

- Restart initializations
- Assignments to variables that are never used

Note

This inconsistency can occur only if you reference a DD `VariableClass` object whose `ScopeReducedClass` property is set, because the `Optimization` property of all internal default variable classes is set to `ERASABLE` and `MOVABLE`.

Example Consider a variable `x` that has the variable class `G` with the scope-reduced class `M`. They are specified as follows:

Property	G	M
Scope	global	global
Storage	default	static
Optimization	ERASABLE, SCOPE_REDUCEABLE	SCOPE_REDUCEABLE
ScopeReducedClass	M	-
RestartFunctionName	default	default

The unoptimized code looks like this:

```
Int16 x;

void RestartFoo(void) {
    x = 42;
}

void foo(void) {
    if (cond) {
        x = 17;
    } else {
        x = 35;
    }
    o = x;
}
```

TargetLink ≤ 4.3	TargetLink 4.4
<pre>void foo(void) { if (cond) { o = 17; } else { o = 35; } }</pre>	<pre>static Int16 x; void RestartFoo(void) { x = 42; } void foo(void) { if (cond) { o = 17; } else { o = 35; } }</pre>

If you switch the ERASABLE value between M and G, this results in the opposite behavior.

If the UserName property of either G or M is set to on, a macro definition might be added to or removed from `t1_defines_$I.h`.

Reason Increased consistency

Migration issue None

Reduction of the dimension of access-function-related auxiliary variables

The code has changed for access-function-related auxiliary variables in the context of dimension reduction:

TargetLink ≤ 4.3	TargetLink 4.4
Vector and matrix variables with an automatic storage duration might be substituted by a scalar variable:	
<pre>Int16 _tmpd_B[10]; for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) { _tmpd_B[Aux_S32] = GetA (Aux_S32); SetBIndexed(Aux_S32, _tmpd_B[Aux_S32]); out1[Aux_S32][index] = _tmpd_B[Aux_S32] + 1; }</pre>	<pre>Int16 Aux_S16; for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) { Aux_S16 = GetA (Aux_S32); SetBIndexed(Aux_S32, Aux_S16); out1[Aux_S32][index] = Aux_S16 + 1; }</pre>
Vector and matrix variables that were reduced to a scalar might now be static and initialized:	
<pre>Int16 Aux_S16; for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) { Aux_S16 = GetA (index, Aux_S32); SetBIndexed(Aux_S32, index, Aux_S16); out1[Aux_S32][Aux_S32_a] = Aux_S16 + 1; }</pre>	<pre>static Int16 _tmpd_B[10][10] = {...}; for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) { _tmpd_B[Aux_S32][index] = GetA (index, Aux_S32); SetBIndexed(Aux_S32, index, _tmpd_B[Aux_S32][index]); out1[Aux_S32][index2] = _tmpd_B[Aux_S32][index] + 1; }</pre>

Reason Correct run-time behavior

Migration issue None

AUTOSAR

Loops in production code for NvData communication with optimized writing

For optimized writing in NvData communication via Data Store Write blocks, TargetLink respects the LoopUnrollThreshold Code Generator option for generating production code with loops. This enables the generation of rolled code.

Reason Code size

Migration issue None

Improved scheduling of pointer initialization

The placement of return pointers of AUTOSAR RTE API functions with read access was improved for the following conditions :

- The RTE API function returning the pointer is called in another function.
- The corresponding RTE API function with write access is called in another function.

TargetLink ≤ 4.3	TargetLink 4.4
<pre>const sint32 * p_IRV_Vector_W5; /* Data store read: Subsystem/SWC/Run2/Read_IRV_Vector_W5 */ p_IRV_Vector_W5 = Rte_IrvIRead_Run2_IRV_Vector_W5(); ... X_Store = p_IRV_Vector_W5[CurrIndex];</pre>	<pre>const sint32 * p_IRV_Vector_W5; ... /* Data store read: Subsystem/SWC/Run2/Read_IRV_Vector_W5 */ p_IRV_Vector_W5 = Rte_IrvIRead_Run2_IRV_Vector_W5(); X_Store = p_IRV_Vector_W5[CurrIndex];</pre>

Reason Readability, matches user expectations

Migration issue None

Block-Specific Code Changes

Sqrt block

In some cases, TargetLink used to generate an unreachable else branch in the production code for the Sqrt block. This occurred in the following cases, for example:

- The Generate optimized code Code Generator option was disabled.
- The global input variable referenced a VariableClass object whose Optimization property was not set to ERASABLE

This is no longer the case.

TargetLink ≤ 4.3	TargetLink 4.4
<pre>if (in <= 0.) { out = 0.; } else { if (in >= 0.) { out = sqrt(in); } else { out = -sqrt(-in); } } }</pre>	<pre>if (in < 0.) { out = 0.; } else { out = sqrt(in); }</pre>

Furthermore, this change results in `out = sqrt(in)`; for positive only ranges (including zero) of the input signal and `out = 0`; for negative only ranges.

Reason MISRA C compliance, code efficiency

Migration issue None

Delay block

Changes in the evaluation of ranges lead to more efficient code for the Delay block.

TargetLink ≤ 4.3

```
T1AuxVar_a =
(UInt8) (((Int16) ((Int16) X_Sa2_Delay_Index) +
(Int16) (30 - ((UInt16) ((UInt16) T1AuxVar) << 1)))) % 30;
```

TargetLink 4.4

```
T1AuxVar_a =
((UInt8) (X_Sa2_Delay_Index +
(UInt8) (30 - ((UInt8) (T1AuxVar << 1))))) % 30;
```

Additionally, the auxiliary calculation for the pointer initialization has been moved into the conditionally executed control flow branch.

Unoptimized

```
Aux_U8 = ((UInt8) (X_Sa2_Delay_Index +
((UInt8) (6 - ((UInt8) (C_U8SATI16_SATb(Sa1_DelayLength, 2, 1) * 3))))) % 6;
pAuxPtr_a = &(X_Delay[Aux_U8]);
if (Sa1_Enable > 0) {
... pAuxPtr_a[0] ...
}
```

Optimized TargetLink ≤ 4.3

```
Aux_U8 = ((UInt8) (X_Sa2_Delay_Index + ((UInt8) (6 - ((UInt8) (C_U8SATI16_SATb(Sa1_DelayLength,
2, 1) * 3))))) % 6;
if (Sa1_Enable > 0) {
pAuxPtr_a = &(X_Delay[Aux_U8]);
... pAuxPtr_a[0] ...
}
```

Optimized TargetLink 4.4

```
if (Sa1_Enable > 0) {
Aux_U8 = Aux_U8 = ((UInt8) (X_Sa2_Delay_Index + ((UInt8) (6 - ((UInt8) (C_U8SATI16_SATb(Sa1_DelayLength,
2, 1) * 3))))) % 6;
pAuxPtr_a = &(X_Delay[Aux_U8]);
... pAuxPtr_a[0] ...
}
```

Reason Code efficiency

Migration issue None

Direct Look-Up Table (n-D) block

The following code patterns of the Direct Look-Up Table (n-D) block have changed with TargetLink 4.4:

- The loop control variable is now named `Aux_*` instead of `i*`, as is the case for other blocks.

- The LoopUnrollThreshold is respected, i.e., the generation of unrolled code is now possible.
- The loop control variable is always defined in the first section of a function.

Reason Increased consistency

Migration issue None

Discrete-Time Integrator block

Saturation of the code generated from the Discrete-Time Integrator block changed. For the Trapezoidal integration method, TargetLink now saturates both additions contained in the underlying formula:

TargetLink ≤ 4.3

```

/* Discrete Integrator: Subsystem/Discrete-Time Integrator */
if (Sa1_Subsystem_FirstRun) {
    /* Discrete Integrator: first run initialization */
    T1AuxVar = 0;
}
else {
    /* Discrete Integrator: integration */
    T1AuxVar =
        ((Int32) X_Sa1_Discrete_Time_Integrator) + ((Int32) (((Int32) Sa1_InPort) +
            ((Int32) U_Sa1_Discrete_Time_Integrator)));
}

/* Discrete Integrator: saturation */
X_Sa1_Discrete_Time_Integrator =
    C_I8SATI32_SATb(T1AuxVar, 100 /* 50. */, -100 /* -50. */);

/* Discrete Integrator: Subsystem/Discrete-Time Integrator */
Sa1_Discrete_Time_Integrator =
    (Int16)X_Sa1_Discrete_Time_Integrator;

```

TargetLink 4.4

```

/* Discrete Integrator: Subsystem/Discrete-Time Integrator */
if (Sa1_Subsystem_FirstRun) {
    /* Discrete Integrator: first run initialization */
    T1AuxVar_b = 0;
}
else {
    /* Discrete Integrator: integration */
    T1AuxVar = ((Int32) X_Sa1_Discrete_Time_Integrator) +
        ((Int32) U_Sa1_Discrete_Time_Integrator);

    /* Discrete Integrator: saturation */
    T1AuxVar_a = C_I32SATI32_SATb(T1AuxVar, 100 /* 50. */, -100 /* -50. */);
    T1AuxVar_b = T1AuxVar_a + ((Int32) Sa1_InPort);
}

/* Discrete Integrator: saturation */
X_Sa1_Discrete_Time_Integrator =
    C_I8SATI32_SATb(T1AuxVar_b, 100 /* 50. */, -100 /* -50. */);

```

Reason Resolves differences in MIL/SIL/PIL simulation modes, matches user expectations

Migration issue None

Math block (pow function)

TargetLink's code pattern for the initial code of the Math block is different, if all of the following conditions are fulfilled:

- The block is specified to implement the **pow** function.
- The block is driven by Constant blocks that do not specify a signal or variable.

TargetLink ≤ 4.3

```
if (( in1 < 0) && (in2 != floor(in2)))
{
    out = -(pow(-in1, in2));
}
else
{
    out = pow(in1, in2);
}
```

TargetLink 4.4

TargetLink now evaluates the conditions (**in1 <0**) and (**in2 != floor(in2)**) when generating the initial code. If **in1** and/or **in2** are constants, either of the two **if** conditions or the whole if-else statement might not be generated.

Reason Code efficiency

Migration issue None

Math block (mod/rem function)

The code of the Math block is different, if all the following conditions are fulfilled:

- The block is specified to implement the **mod** or **rem** function.
- At least one of the block's inputs or its result has a floating-point type.

TargetLink ≤ 4.3

Code for mod Function

```
Sa1_I16_F32F64_ = (Int16) (((Float64) Sa1_F32NumInInt) -
(Sa1_F64DenomInInt *
((Float64) (Int32) (((Float64) Sa1_F32NumInInt) /
Sa1_F64DenomInInt))));
if (Sa1_I16_F32F64_ != 0) {
    if (((Sa1_F32NumInInt < 0.F) &&
(Sa1_F64DenomInInt >= 0.)) ||
((Sa1_F32NumInInt >= 0.F) &&
(Sa1_F64DenomInInt < 0.))) {
        Sa1_I16_F32F64_ += ((Int16) Sa1_F64DenomInInt);
    }
}
```

TargetLink 4.4

```
Sa1_I16_F32F64_ = (Int16) (((Float64)
Sa1_F32NumInInt) -
(Sa1_F64DenomInInt * floor(((Float64)
Sa1_F32NumInInt) / Sa1_F64DenomInInt)));
```

TargetLink ≤ 4.3	TargetLink 4.4
Code for rem Function	
<pre>Sa1_F64_F32I16_ = ((Float64) Sa1_F32NumIn) - (((Float64) Sa1_I16DenomIn) * ((Float64) (Int32) (((Float64) Sa1_F32NumIn) / ((Float64) Sa1_I16DenomIn))));</pre>	<pre>Aux_F64 = ((Float64) Sa1_F32NumIn) / ((Float64) Sa1_I16DenomIn); if (Aux_F64 > 0.) { Aux_F64 = floor(Aux_F64); } else { Aux_F64 = ceil(Aux_F64); } Sa1_F64_F32I16_ = ((Float64) Sa1_F32NumIn) - (((Float64) Sa1_I16DenomIn) * Aux_F64);</pre>

Reason Resolves differences in MIL/SIL/PIL simulation modes, matches user expectations

Migration issue None

Efficiency

Implicit variables in min/max operations

TargetLink used superfluous implicit variables in unoptimized code for min/max operations if all of the following was true:

- The operands had different data types.
- The result type was different from the data type resulting from the worst-case ranges of the operands.
- One of the operands had the same data type as the worst-case range.

This is no longer the case.

TargetLink ≤ 4.3	TargetLink 4.4
<pre>Int16 Aux_, Aux__a; Aux_ = Ca1_I16In; Aux__a = (Int16) Ca1_U8In; if (Aux_ > Aux__a) { Ca1_I80Out = (Int8)Aux_; } else { Ca1_I80Out = (Int8)Aux__a; }</pre>	<pre>Int16 Aux_; Aux_ = (Int16) Ca1_U8In; if (Ca1_I16In > Aux_) { Ca1_I80Out = (Int8)Ca1_I16In; } else { Ca1_I80Out = (Int8)Aux_; }</pre>
<pre>Aux_S32 = (Int32) Sa1_UInt16In; if (((Int32) Sa1_Int16In) < Aux_S32) { Sa1_I16MinI16U16_ = Sa1_Int16In; } else { Sa1_I16MinI16U16_ = (Int16) Aux_S32; }</pre>	<pre>if (((Int32) Sa1_Int16In) < ((Int32) Sa1_UInt16In)){ Sa1_I16MinI16U16_ = Sa1_Int16In; } else { Sa1_I16MinI16U16_ = (Int16) Sa1_UInt16In; }</pre>

Reason Code efficiency

Migration issue None

Result ranges

TargetLink 4.4 takes the available range information for a multiplication into account in more cases. Therefore, a smaller data type might be used.

TargetLink ≤ 4.3
<pre>F_I64MULI32U32(SActRet1_Rescaler, (tUI32) x_FunctionCoordinationFuncFact_ldu16, &Aux_lds32, &Aux_ldu32); C_I16DIVI64U32(Aux_lds32, Aux_ldu32, (tUI32) 1024, x_ActiveRetFuncFactor_lds16); Sa1_out_2 = (Int16) (UInt8) (SReferenced1_OutPort3 * ((UInt8) MyScalarDDMacro));</pre>
TargetLink 4.4
<pre>x_ActiveRetFuncFactor_lds16 = (tSI16) ((SActRet1_Rescaler * ((tSI32) x_FunctionCoordinationFuncFact_ldu16)) / 1024); Sa1_out_2 = (Int16) (((Int8) SReferenced1_OutPort3) * ((Int8) MyScalarDDMacro));</pre>

If possible, TargetLink performs the multiplication with the same signedness as the result type. This can lead to different casts during the multiplication:

TargetLink ≤ 4.3
<pre>Aux_U16_b = (UInt16) (Int16) (((Int16) Aux_U8) * Aux_S16_b) >> 1;</pre>
TargetLink 4.4
<pre>Aux_U16_b = (UInt16) (((UInt16) (((UInt16) Aux_U8) * ((UInt16) Aux_S16_b))) >> 1);</pre>

Reason Code efficiency

Migration issue None

Addition with constants

The code has changed for Sum blocks whose List of signs property is set to +- and whose first inport is driven by a constant:

TargetLink ≤ 4.3	TargetLink 4.4
<pre>Out = -In1 + 17;</pre>	<pre>Out = 17 - In1;</pre>

Reason Code efficiency

Migration issue None

Inheritance of Struct Data Types

SPI blocks

In TargetLink 4.4, a block can inherit a struct data type from a preceding block if the following conditions are met:

- The block's Inherit properties checkbox is selected.
- The preceding block references a root struct DD variable.

This was already the case for block variables referencing one of the following DD objects:

- a DD Typedef object whose BaseType property is set to Struct
- the DD Typedef object IMPLICIT_STRUCT.

However, this did not apply to structured variables.

The resulting code then contains:

TargetLink ≤ 4.3	TargetLink 4.4
Multiple non-struct variables: One for each leaf signal of the inheriting block.	One struct variable for the inheriting block.

Reason Increased consistency, code efficiency

Migration issue None

Ports of atomic subsystems

An unenhanced inport or outport of an atomic subsystem can inherit a struct data type from a preceding block in the following cases.

- A bus signal is applied to the port.
- The preceding block references a DD Variable object that is specified as a formal function parameter passing a pointer to a struct.

This was already the case for block variables referencing one of the following DD objects:

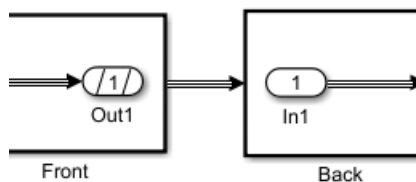
- a DD Typedef object whose BaseType property is set to Struct
- the DD Typedef object IMPLICIT_STRUCT.

However, this did not apply to structured variables.

The unenhanced port is then implemented as follows:

TargetLink ≤ 4.3	TargetLink 4.4
Implemented as multiple global non-struct variables.	Implemented as a function parameter of the atomic subsystem with type 'pointer to struct'.

Example Consider the following modeling situation:



The described changes lead to the following differences in the code for this model:

TargetLink ≤ 4.3	TargetLink 4.4
<pre> 1 StructType DDStruct; 2 Int16 Sa2_Out1; 3 Int16 Sa2_Out1_a; 4 void Sa2_Back(void) 5 { 6 Sa2_Out1 = DDStruct.a; 7 Sa2_Out1_a = DDStruct.b; </pre>	<pre> 1 StructType DDStruct; 2 void Sa2_Back(StructType * Sa2_In1, StructType * Sa2_Out1) 3 { 4 *Sa2_Out1 = *Sa2_In1; 5 } 6 void Code(void) 7 { </pre>

TargetLink ≤ 4.3	TargetLink 4.4
<pre> 8 } 9 void Code(void) 10 { 11 Sa3_Front(&DDStruct); 12 Sa2_Back(); 13 [...] 14 }</pre>	<pre> 8 StructType Sa2_Out1; 9 Sa3_Front(&DDStruct); 10 Sa2_Back(&DDStruct, &Sa2_Out1); 11 [...] 12 }</pre>

Remarks This code change can occur if you used a Bus Output block at the interface of an incremental code generation unit and if the following conditions are fulfilled:

- One of the Bus Output block's formal parameters passes a pointer to a struct.
- This formal parameter can be specified via a DD VariableClass object whose Scope property is set to `ref_param`.

With TargetLink ≤ 4.3, the successor block of the unenhanced Inport block reads from the structured actual parameter that was generated for the Bus Output block. This is shown in lines 6 and 7 in the table above. The `Sa2_Back` function does not have any parameter. This is shown in line 12.

With TargetLink 4.4, the unenhanced Inport block is implemented as a formal pointer to struct parameter. This parameter has the same structured actual parameter as the Bus Output block. Optimization can possibly reduce the parameter's scope to `local`. This is shown in lines 4 and 10.

Reason Increased consistency, code efficiency

Migration issue None

Mixed Operations (Floating-Point and Fixed-Point Types)

Binary floating-point operations

Binary floating-point operations with a floating-point result, a floating-point operand, and an integer or fixed-point operand are now calculated in the largest occurring floating-point type. This does not apply to additions and subtractions resulting from a Sum block.

TargetLink ≤ 4.3

```
Sa1_F64Out = (Float64) (((Float64) Sa1_I16In) * Sa1_F32In);
```

TargetLink 4.4

```
Sa1_F64Out = ((Float64) Sa1_I16In) * ((Float64) Sa1_F32In);
```

Reason MISRA C compliance

Migration issue None

Stateflow

Log macro calling

TargetLink 4.4 calls log macros later. This results in improved optimization possibilities, including merged loop code resulting from vector or matrix allocations as shown in the following pseudo-code:

TargetLink ≤ 4.3	TargetLink 4.4
<pre>for() { out[...] = test[...] +1; } LOG_VEC(out); for() { merge[...] = out[...]; } LOG_VEC(merge);</pre>	<pre>for() { out[...] = test[...] +1; merge[...] = out[...]; } LOG_VEC(out); LOG_VEC(merge);</pre>

Reason Improved loop merging can lead to better optimization.

Migration issue None

Floating-point expressions cast to integers

Index expressions that contain floating-point operands that are explicitly cast to integers by the user are now always calculated as floating-points. Consider the code for the Stateflow expression `inC[int16(inC[0]/2*2.11)]`:

TargetLink ≤ 4.3 p1	TargetLink 4.4
<pre>Ca1_inC[(Int16) ((Ca1_inC[0] / 2) * 2 /* 2.11 */)]</pre>	<pre>Ca1_inC[(Int16) (((Float32) (Int32) (Ca1_inC[0] >> 1)) * 2.11F)]</pre>

Reason Bug fix

Migration issue None

Auxiliary variables for struct parameters

Generally, TargetLink generates auxiliary variables for input and output parameters of graphical functions in Stateflow charts that are not scalar or complex.

This was not the case when all of the following conditions were fulfilled:

- An Input or Output parameter was a bus.
- The function parameter was specified via a structured variable in the Data Dictionary.
- The function parameter was not called by reference.

In this case, TargetLink 4.4 now generates auxiliary variables as follows:

For a graphical function `F(StructFormal)` with the input parameter `StructFormal` of the type `'Bus'` that is specified by the `DD_StructFormal` DD Variable object whose `Scope` property is set to anything but `ref_param`, *unoptimized* code looks like this:

TargetLink ≤ 4.3	TargetLink 4.4
<pre>StructFormal. ... = StructActual. ...; ... StructFormal. ... = StructActual. ...; F();</pre>	<pre>Aux_StructFormal. ... = StructActual. ...; ... Aux_StructFormal. ... = StructActual. ...; StructFormal. ... = Aux_StructFormal. ...; ... StructFormal. ... = Aux_StructFormal. ...; F();</pre>

Reason Bug fix

Migration issue None

Values of state IDs for Stateflow states

TargetLink now sorts the state IDs of Stateflow states by their Ca number:

TargetLink ≤ 4.3	TargetLink 4.4
<pre>#define Ca5_A_id 4 #define Ca6_B_id 3</pre>	<pre>#define Ca5_A_id 3 #define Ca6_B_id 4</pre>

Reason Bug fix, Increased consistency

Migration issue None

Other

Changed order of function parameters

The order of function parameters might be changed if one parameter of the function is removed during optimization.

This is known to occur for the following functions:

- Step functions of reused subsystems or charts that do not have a reuse-instance-structure as a parameter.

Reason Bug fix – Known Problem Report KPR.2017.08.03.004

Migration issue None

Naming of variable vector width auxiliary variables

TargetLink uses auxiliary variables for the initialization of variables with ExchangeableWidth in a restart function. The identifiers of these auxiliary variables have changed:

TargetLink ≤ 4.3	TargetLink 4.4
Temp__a, Temp__c, Temp__e, ...	Temp_, Temp__a, Temp__b, ...

Reason Matches user expectations

Migration issue None

Scalar products with floating-point operands and integer results

Scalar products of floating-point operands with integer results are now calculated in floating-point notation and then cast to integers. Usually, such products result from using Product, FIR Filter, Discrete Filter, Discrete Transfer Fcn and Discrete State-Space blocks.

TargetLink ≤ 4.3

```
Sa1_Product2D2D[3][3] = (Int32) (((Int32) (((Float32) Sa1_Matrix2D[3][0]) * Sa1_Matrix2D2[0][3])) +
  ((Int32) (((Float32) Sa1_Matrix2D[3][1]) * Sa1_Matrix2D2[1][3])) +
  ((Int32) (((Float32) Sa1_Matrix2D[3][2]) * Sa1_Matrix2D2[2][3])) +
  ((Int32) (((Float32) Sa1_Matrix2D[3][3]) * Sa1_Matrix2D2[3][3])) +
  ((Int32) (((Float32) Sa1_Matrix2D[3][4]) * Sa1_Matrix2D2[4][3]));
```

TargetLink 4.4

```
Sa1_Product2D2D[3][3] = (Int32) (((Float32) Sa1_Matrix2D[3][0]) * Sa1_Matrix2D2[0][3]) +
  (((Float32) Sa1_Matrix2D[3][1]) * Sa1_Matrix2D2[1][3]) +
  (((Float32) Sa1_Matrix2D[3][2]) * Sa1_Matrix2D2[2][3]) +
  (((Float32) Sa1_Matrix2D[3][3]) * Sa1_Matrix2D2[3][3]) +
  (((Float32) Sa1_Matrix2D[3][4]) * Sa1_Matrix2D2[4][3]);
```

Reason Precision

Migration issue None

Changes in identifiers

The identifiers of the following code elements might change:

- Pointers and substructs in the context of indirect function reuse.
- State IDs of Stateflow states

Additionally, the names of auxiliary loop variables changed. When determining their names for initial code, TargetLink now starts with `Aux_` instead of `Aux_S32_a`. During optimization, their names might change.

Reason Increased consistency

Migration issue None

Changes in function inlining

When determining whether to inline a function, TargetLink no longer takes into account simple variable definitions. This can result in the following:

- Change in inlining:
 - Small functions are more frequently inlined.
 - Different order of variable definitions within the function body.
- Change in optimization order:
 - Different order of `#combined#` comments.
 - More auxiliary variables.

Reason Matches user expectations

Migration issue None

Memory sections for TriCore 17xx/Task32 optimization module

TargetLink now generates statements for memory sections that comply with the compiler's manual. Additionally, TargetLink 4.4 no longer automatically generates prefixes for section names, which allows for more customization. This applies to the sections `bss`, `data`, `rom`, and `code`. The following example applies to the `bss` section:

TargetLink ≤ 4.3	TargetLink 4.4
<pre>#pragma section nearbss = "near_<name>" #pragma section farbss = "far_<name>"</pre>	<pre>#pragma section nearbss "<name>" #pragma section farbss "<name>"</pre>

Reason Compliance with compiler's manual

Migration issue None

Removal of suffix from INT32MIN constants

The L suffix was removed from INT32MIN constants.

TargetLink ≤ 4.3	TargetLink 4.4
<pre>(-2147483647L -1L)</pre>	<pre>(-2147483647 -1)</pre>

Reason Code efficiency

On targets whose long type was a 64bit type, the suffix led to inefficient code because the constant unnecessarily used 64 bits. For all other targets, the suffix was superfluous, because not using it resulted in the same integer length as using it.

Migration issue None

Changes in code comments

Several code comments changed:

- Comments preceding the code of Rescaler blocks now contain the string `Rescaler:`.
- Comments for variables with the `CONST` qualifier no longer contain range information. This can result in additional line breaks.

Reason Increased consistency, readability

Migration issue None

Copy propagation and call by reference/vector/matrix call parameters

Copy propagation changed for call parameters of the following kinds:

- Call by reference parameters
- Vector call parameters
- Matrix call parameters

TargetLink no longer propagates the following in these kinds of call parameters:

- Bitfields
- Macros

- Variables qualified as `const` or `volatile`
- Variables qualified via a type prefix

TargetLink ≤ 4.3	TargetLink 4.4
<pre> const Type* MyType = Rte_IRead_SWC_MyType(); x = S.bitfield; loop i = 0..9 y[i] = MyType[i]; foo(&S.bitfield, MyType /* Type[10] */); </pre>	<pre> const Type* MyType = Rte_IRead_SWC_MyType(); x = S.bitfield; loop i = 0..9 y[i] = MyType[i]; foo(&x¹, y²)/* Type[10] */); </pre>

- 1) Not replaced by `S.bitfield`, because addresses of bitfields are not supported.
 2) Not replaced by `MyType`, because `const` is not supported in contexts that are not `const`.

Reason Bug fix

Migration issue If you used type prefixes to make non-compiling code compilable, check if this is still necessary.

In AUTOSAR contexts, check the value of the `/Pool/AUTOSAR/Config.RteUsePointerToConstForNonScalarReturnValues` property.

Discontinuations

Where to go from here

Information in this section

Discontinued TargetLink Features	253
Obsolete API Functions	253
Obsolete Limitations	253

Discontinued TargetLink Features

No discontinued TargetLink features

No features have been discontinued with TargetLink 4.4.

Obsolete API Functions

No obsolete API functions

No API functions have been discontinued with TargetLink 4.4.

Obsolete Limitations

Obsolete with TargetLink 4.4

With TargetLink 4.4, the following limitations of previous TargetLink versions were removed:

General limitations

Fast restart behavior

Before a simulation run starts, TargetLink checks whether Simulink's Fast Restart mode is active. If yes, an error occurs in TargetLink and the simulation features are disabled, i.e., logging and overflow detection. The following limitation applies:

- Simulink automatically disables the Fast Restart mode when starting a simulation if at least one TargetLink subsystem is in SIL or PIL mode.

TargetLink subsystem IDs

Each TargetLink subsystem must have a unique ID string with up to 10 characters. You can enter the ID on the Code Generation page of the TargetLink Main Dialog.

ModelDataLogs logging format

If the Simulink ModelDataLogs logging format is active, signals in referenced models that are instantiated multiple times cannot be logged. Use the DataSet logging format instead.

Block-specific limitations**Constant block**

Constant blocks whose data type is specified by using a Simulink.Bus object are not supported by TargetLink.

Code generation limitations**Structured variables**

TargetLink does not support arrays of structured variables.

The following AUTOSAR limitations no longer exist:**Import of ARRAY-SPECIFICATION elements modeled as arrays of struct**

When importing value specifications that are specified as 1-D or 2-D arrays of struct, TargetLink only imports the first array element that contains a RECORD-SPECIFICATION into the Data Dictionary as a struct.

No NvData communication

For the following access points, only sender-receiver communication is supported but not NvData communication:

- DataSendPoint
- DataReceivePointByArgument

The following Data Dictionary limitation no longer exists:**DD objects with write protection**

If a DD object is write-protected, i.e., `Access="rwr-", "rw--", "r-r-", or "r---"`, its properties cannot be modified and its children cannot be deleted. However, it is possible to add new children to a write-protected DD object.

Changes in Future TargetLink Versions

Where to go from here

Information in this section

Features to Be Discontinued	255
API Functions to Be Discontinued	256
Deprecated Code Generator Options	257

Features to Be Discontinued

RTOS/OSEK code generation mode

Support for TargetLink's RTOS/OSEK Code Generation Modes will be discontinued in future TargetLink versions.

Target Optimizations Module (TOM)

Support for TargetLink's Target Optimizations Module (TOM) will be discontinued in future TargetLink versions.

Clean code and Do not log anything

Variables selected for logging cannot be fully optimized. When generating code with the Global logging option **Do not log anything** or **Log according to block data**, TargetLink does not fully optimize the code to facilitate testing. This means the code only differs with regard to the log macros. This contrasts the Clean code checkbox on the Code Generation page of the TargetLink Main Dialog block, which always activates full code optimization.

The special **Do not log anything** behavior will be removed in future TargetLink versions.

User state flags in Stateflow

Support for TargetLink's own user state flags feature in Stateflow will be discontinued in future TargetLink versions, because the Stateflow Active State data is similar and more convenient to use.

Simulink's classic initialization mode

Support for Simulink's classic initialization mode will be discontinued in future TargetLink versions.

Dynamic components

Support for specifying dynamic components for DD Variable objects will be discontinued in future TargetLink versions.

Code generation for special OSEK versions	The code generation for special OSEK versions, such as OsCan, will be discontinued in future TargetLink versions.
Automatic interpretation of Boolean	TargetLink's automatic interpretation of certain integer data types as Boolean will be discontinued in future TargetLink versions.
Support of AUTOSAR 2.x and 3.x	TargetLink's support of AUTOSAR 2.x and AUTOSAR 3.x will be discontinued in future TargetLink versions.
Unit Delay Reset Enabled	TargetLink's support of the Unit Delay Reset Enabled block will be discontinued in future TargetLink versions.
TargetLink Blockset (stand-alone)	TargetLink's setup routine of the TargetLink Blockset (stand-alone) will be discontinued in future TargetLink versions.
Model change detection via checksum	TargetLink's support of model change detection via checksum will be discontinued in future TargetLink versions.

API Functions to Be Discontinued

Discontinued API functions The following API functions are deprecated and will be removed in future TargetLink versions:

Function	Deprecated Since	Replacement Function
<code>t1_adapt_dd_references</code>	TargetLink 4.0	<code>t1MoveDDObject</code>
<code>t1_extract_subsystem</code>	TargetLink 4.0	<code>t1ExtractSubsystem</code>
<code>t1_find_dd_references</code>	TargetLink 4.0	<code>t1FindDDReferences</code>
<code>t1_get_blockset_mode</code>	TargetLink 4.0	<code>t1OperationMode</code>
<code>t1_sim_interface</code>	TargetLink 4.0	<code>t1SimInterface</code>
<code>t1_switch_blockset</code>	TargetLink 4.0	<code>t1OperationMode</code>
<code>t1_upgrade</code>	TargetLink 4.0	<code>t1Upgrade</code>
<code>generate_ASAP2</code>	TargetLink 2.x	<code>dsdd_export_a2l_file</code>
<code>t1_upgrade_libmapfile</code>	TargetLink 4.0	-

Note

See the help contents on the new API functions to adjust your user scripts accordingly.

Related topics

References

[tSimInterface](#)  [TargetLink API Reference](#)

Deprecated Code Generator Options

List of deprecated Code Generator options

The following Code Generator options are deprecated and will be removed in future TargetLink versions:

- SideEffectFreeAnalysisThreshold
- TreatAllForcedAtomicSubsystemsAsWeakAtomic
- DisableFunctionsAsAnalysisBoundaries

VEOS

Where to go from here

Information in this section



New Features of VEOS 4.3	259
Gives an overview of the new features of VEOS 4.3.	
Compatibility of VEOS 4.3	261
Provides information on the compatibility of VEOS 4.3.	
Migrating to VEOS 4.3	263
To migrate from VEOS 4.2 to VEOS 4.3, you might have to carry out the following migration steps.	
Discontinuations in VEOS	264







New Features of VEOS 4.3

Support of memory segments and page switching

The calibration memory segments of classic V-ECUs generated as of VEOS 4.3 are divided into two calibration pages. You can use ControlDesk to switch the active V-ECU calibration page at run time.

During the build process, VEOS also generates an ECU Image file that contains the initial parameter values of the V-ECU application. Adding the ECU Image file to the XCP on Ethernet device that represents the V-ECU in a ControlDesk experiment lets you perform offline calibration on the V-ECU with ControlDesk, for example.

For more information on handling calibration pages with ControlDesk, refer to [Basics on Memory Pages](#) ( [ControlDesk Calibration and Data Set Management](#)) and [How to Activate the Working or Reference Page](#) ( [ControlDesk Calibration and Data Set Management](#)).

Simulation in soft real time	<p>VEOS now supports simulation in <i>soft real time</i>.</p> <p>For more information, refer to Simulation in soft real time ( VEOS Manual).</p>
Measuring and calibrating variables of 64-bit applications	<p>As of VEOS 4.3, you can measure and calibrate variables of 64-bit applications executed on VEOS using dSPACE experiment software, such as ControlDesk.</p>
Building 64-bit VPUs from CTLGZ files	<p>VEOS now lets you build 64-bit VPUs from CTLGZ files. This lets you integrate CTLGZ files that contain 64-bit libraries.</p> <p>Refer to Simulation Target Manager ( VEOS Manual).</p>
Support of VEOS Ethernet channels for bus monitoring and logging	<p>Ethernet channels on VEOS are now supported as Ethernet interfaces for bus monitoring.</p> <p>For more information, refer to Basics on Monitoring, Logging, and Replaying Bus Communication ( ControlDesk Bus Navigator).</p>
Support of the LIN communication driver module for LIN masters	<p>VEOS 4.3 lets you build and simulate V-ECUs that contain a configuration and implementation of the LIN communication driver (<code>Lin</code>) module for LIN masters. The module provides hardware access to the LIN bus.</p> <p>Refer to Basic Software Module Support for V-ECUs ( Virtual Validation Overview).</p>
Automatic connection of communication controllers to communication clusters	<p>VEOS now lets you automatically connect all unconnected communication controllers to communication clusters.</p> <p>For more information, refer to Autoconnect ( VEOS Manual).</p>
Python 3.6 support	<p>As for all other products of dSPACE Release 2018 B, the Python demo scripts for automating VEOS Player have been adapted to Python 3.6.</p> <p>For information on the differences between the previously supported Python 2.7 and Python 3.6 and for general migration information, refer to Migrating Python Scripts from Python 2.7 to Python 3.6 on page 33.</p>
Data type information for signals	<p>VEOS Player's Properties pane now displays the data type of signals selected in the Port Topology pane.</p> <p>For more information, refer to Signal ( VEOS Manual).</p>

Compatibility of VEOS 4.3

Where to go from here

Information in this topic

Compatibility in general	261
Supported compiler versions	261
Real-Time Testing compatibility	261
File import compatibility	261
BSC compatibility	261
CTLGZ compatibility	262
FMU compatibility	262
OSA compatibility	262
SIC compatibility	262
SMC compatibility	263

Compatibility in general

dSPACE recommends using only software products from the same dSPACE Release. This ensures maximum run-time compatibility.

Supported compiler versions

For information on supported compiler versions, refer to [Basics on Integrating the Simulation System](#) (📖 [VEOS Manual](#)).

Real-Time Testing compatibility

To use RTT in connection with VEOS and ControlDesk, the Real-Time Testing (RTT) version used by the VEOS Simulator that runs the simulation system and the RTT version that is active on the PC must be identical.

The following table shows the VEOS Simulator version and the corresponding RTT version:

VEOS Simulator	Version
... from VEOS 4.3	Real-Time Testing Version 4.0
... from VEOS 4.2	Real-Time Testing Version 3.4
... from VEOS 4.1	Real-Time Testing Version 3.3
... from VEOS 4.0	Real-Time Testing Version 3.2

ControlDesk 6.4 automatically uses the VEOS Simulator of VEOS 4.3. You can therefore use RTT in connection with VEOS and ControlDesk if RTT 4.0 is active on the PC.

File import compatibility

BSC compatibility VEOS 4.3 is compatible with bus simulation container (BSC) files created with the Bus Manager of dSPACE Release 2018-B (BSC version 1.5).

CTLGZ compatibility The following table shows the compatibility between VEOS 4.3 and CTLGZ files (V-ECU implementations):

V-ECU Implementations Created With...	V-ECU Implementation Version
dSPACE Release 2018-B: <ul style="list-style-type: none"> ▪ SystemDesk 5.2 ▪ TargetLink 4.4 	2.8 ¹⁾
dSPACE Release 2018-A: <ul style="list-style-type: none"> ▪ SystemDesk 5.1 	2.7 ¹⁾
dSPACE Release 2017-B: <ul style="list-style-type: none"> ▪ SystemDesk 5.0 ▪ TargetLink 4.3 	2.6 ¹⁾
dSPACE Release 2017-A: <ul style="list-style-type: none"> ▪ SystemDesk 4.8 	2.5 ¹⁾

¹⁾ There is a migration issue for VEOS if the container file to be imported contains static libraries. For more information, refer to [Migrating from VEOS 4.1 to 4.2 \(VEOS Manual\)](#).

FMU compatibility VEOS supports:

- Functional Mock-up Units (FMUs) that comply with the FMI 2.0 standard
- Only the FMI for Co-Simulation interface, but not the FMI for Model Exchange interface

For detailed and up-to-date compatibility information on dSPACE's FMI support, refer to:

<http://www.dspace.com/go/FMI-Compatibility>.

OSA compatibility The following table shows the compatibility between VEOS 4.3 and offline simulation application (OSA) files:

OSA Files Created with Products Of ...	OSA Version
dSPACE Release 2018-B	4.3 ¹⁾
dSPACE Release 2018-A	4.2 ²⁾
dSPACE Release 2017-B	4.1 ²⁾
dSPACE Release 2017-A	4.0 ²⁾

¹⁾ OSA files created or modified with VEOS 4.3 cannot be loaded in earlier VEOS versions.

²⁾ You cannot modify the properties of VPU's contained in an OSA file if you open the OSA file in a later VEOS version than the version with which the OSA file was originally created. However, port and network connections can be edited. As a consequence, it is recommended to rebuild the binary OSA files from existing model implementation container files (CTLGZ, SIC, BSC, FMU) when you migrate from one VEOS version to another.

SIC compatibility The following table shows the compatibility between VEOS 4.3 and Simulink implementation container (SIC) files:

SIC Files Created With ...	SIC Version
dSPACE Release 2018-B: <ul style="list-style-type: none"> ▪ Model Interface Package for Simulink 4.0 ▪ TargetLink 4.4 	1.5 ¹⁾

SIC Files Created With ...	SIC Version
dSPACE Release 2018-A: ▪ Model Interface Package for Simulink 3.6	1.4 ^{1), 2)}
dSPACE Release 2017-B: ▪ Model Interface Package for Simulink 3.5	1.3 ^{1), 2)}
dSPACE Release 2017-A: ▪ Model Interface Package for Simulink 3.4	1.2.1 ^{1), 2)}

¹⁾ There is a migration issue for VEOS if the container file to be imported contains static libraries. For more information, refer to [Migrating from VEOS 4.1 to 4.2](#) ([PDF VEOS Manual](#)).

²⁾ If the SIC file is created with a previous dSPACE Release and if the SIC file contains an ASM model, you cannot simulate the model in VEOS 4.3 (dSPACE Release 2018-B). For more information, refer to [Migrating ASM models](#) on page 264.

SMC compatibility The following table shows the compatibility between VEOS 4.3 and system model container (SMC) files:

SMC Files Created With ...	SMC Version
dSPACE Release 2018-B: ▪ SYNECT 2.6 ▪ VEOS 4.3	1.1
dSPACE Release 2018-A: ▪ SYNECT 2.5 ▪ VEOS 4.2	1.1
dSPACE Release 2017-B: ▪ SYNECT 2.4	1.0

You also have to consider the following compatibility restrictions of the individual container files contained in the SMC file to be imported: If the SMC file contains a container of an unsupported version, VEOS 4.3 imports neither the unsupported container nor the connections to the application process based on the unsupported container.

SMC files exported with VEOS 4.3 have file version 1.1.

Migrating to VEOS 4.3

Introduction

To migrate from VEOS 4.2 to VEOS 4.3, you might have to carry out the following migration steps.




Note

To migrate to VEOS 4.3 from versions earlier than 4.2, you might also have to perform the migration steps of the intervening VEOS versions.


Migrating ASM models

You cannot simulate an ASM model on VEOS 4.3 (dSPACE Release 2018-B) if the model is contained in an OSA or SIC file that was created with a previous dSPACE Release.

To simulate an ASM model that was last saved with a dSPACE Release earlier than Release 2018-B on VEOS 4.3, perform the following steps:

1. Migrate the ASM model to dSPACE Release 2018-B.
For information on migrating ASM models, refer to [Migrating ASM Models](#) ( [ASM User Guide](#)).
2. Generate a Simulink implementation container (SIC) file on the basis of the ASM model by using the *Model Interface Package for Simulink*.
For instructions, refer to [Generating Simulink Implementation Containers](#) ( [Model Interface Package for Simulink - Modeling Guide](#)).
3. Import the SIC file to the VEOS Player of VEOS 4.3.
For instructions, refer to [How to Import Simulink Implementations](#) ( [VEOS Manual](#)).

Migrating from prior VEOS versions

To migrate from prior VEOS versions and reuse existing offline simulation applications, you might have to carry out additional migration steps. For more information on the migration steps, refer to [Migrating from Prior Versions of VEOS](#) ( [VEOS Manual](#)).

Discontinuations in VEOS

Discontinuations as of VEOS 4.3

Discontinuation of XCP stimulation As of dSPACE Release 2018-B, VEOS no longer supports XCP stimulation.

Up to and including VEOS 4.2, enabling XCP stimulation was possible in connection with the import of CTLGZ files. The relevant setting was available on the XCP Service page of the VEOS Import dialog.

The relevant `XcpStimulationEnabled` property of the `IImportSettings` automation interface is deprecated.

Compatibility Information

Where to go from here

Information in this section

Supported MATLAB Releases	265
Operating System	266
Run-Time Compatibility of dSPACE Software	268
Limitations for Using Windows Features	269

Supported MATLAB Releases

MATLAB®/Simulink®

Working with various dSPACE products requires that MATLAB is installed on your host PC.

Tip

For system requirements of MathWorks® software, refer to <http://www.mathworks.com/support/sysreq.html>.

MATLAB Release...	...Is Supported by dSPACE Release 2018-B					
	RCP and HIL Software ^{1), 2)}	AutomationDesk 6.0 ³⁾	TargetLink 4.4	Model Compare 2.9	dSPACE Python Extensions 3.0 ⁴⁾	XIL API .NET MAPort 2018-B
R2018b	✓ ⁵⁾	✓	✓	✓	✓	✓
R2018a	✓	✓	✓	✓	✓	✓
R2017b	✓	✓	✓	✓	✓	✓
R2017a	✓ ⁶⁾	✓	✓	✓	✓	✓

- 1) 'RCP and HIL software' is a generic term for a software package containing several dSPACE software products, for example, ASM, RTI, ConfigurationDesk, MotionDesk and ModelDesk. These software products are installed in a common folder.
- 2) MATLAB/Simulink Student Suite does not support Automotive Simulation Models (ASM).
- 3) The AutomationDesk MATLAB Access library requires MATLAB.
- 4) matlablib2 of dSPACE Python Extensions requires MATLAB.
- 5) R2018b is not supported by the RTI FPGA Programming Blockset – FPGA Interface.
- 6) R2017a is not supported by ASM KnC.

For up-to-date information on additional MATLAB releases that can be used in combination with dSPACE software, refer to <http://www.dspace.com/go/MATLABCompatibility>.

Operating System

Operating system on host PC

The dSPACE products of dSPACE Release 2018-B support the following operating systems:

- Windows 7 Professional, Ultimate, and Enterprise with Service Pack 1 (64-bit versions)
Only the listed editions are supported. The Windows 7 Home and Starter editions are not supported.
- The following editions, channels and servicing options of Windows 10:
 - Windows 10 Professional, Education, and Enterprise (64-bit versions)

The Windows 10 Home, Mobile, and Windows 10 S editions are not supported.

- Long-Term Servicing Branch: LTSB 2016
- Semi Annual Channel (formerly known as Current Branch (CB)): The compatibility statement of Microsoft applies. This means that newer versions released in this channel should be compatible with all previous versions. dSPACE used the 1803 version of the Semi Annual Channel for testing.

Some limitations apply when you use dSPACE software in conjunction with features of Windows. Refer to [Limitations for Using Windows Features](#) on page 269.

Using MicroAutoBox Embedded PC as host PC

ControlDesk can also be installed on:

- MicroAutoBox Embedded PC 3rd Gen. Intel® Core™ i7-3517UE Processor, running on Windows 7 Professional, Ultimate, and Enterprise, 64-bit version
- MicroAutoBox Embedded PC 6th Gen. Intel® Core™ i7-6822EQ Processor, running on Windows 10 IOT Enterprise, LTSB 2016, 64-bit version

Operating system on SYNECT server

The SYNECT server supports the following operating systems:

- The same operating systems as listed above for all dSPACE products of dSPACE Release 2018-B.
- The 64-bit versions of Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2 and Windows Server 2016. The Windows Server Semi Annual Channel versions are not supported.

Note

Do not install the SYNECT client on a Windows server operating system, such as Windows Server 2016.

Operating system on server for floating network licenses

If you purchased floating network licenses, you have to specify one of the network PCs as a license server. Every PC with CodeMeter Runtime software can be used as a license server.

Valid for servers without dSPACE software dSPACE only tests license servers with Microsoft Windows operating systems in combination with protected dSPACE software.

Note

Non-Windows operating systems, such as Ubuntu Linux, are not tested. You can use them at your own risk. dSPACE does not provide support in this case.

Valid for servers with dSPACE Installation Manager dSPACE Installation Manager supports the same operating systems as the other dSPACE software products described above.

Allowing communication

Installing of additional firewall rules Additional Windows firewall rules are installed during the installation of various dSPACE software products. For example, one rule allows communication with a dSPACE expansion box, such as AutoBox. Another rule allows MotionDesk to receive motion data from a network channel. These example rules are created by the following commands:

- `netsh advfirewall firewall add rule name="dSPACE Net Service" service=any dir=in action=allow profile=any protocol=icmpv4:0, any description="Allow the dSPACE Net Service to connect to a dSPACE expansion box via network."`
- `netsh advfirewall firewall add rule name="dSPACE MotionDesk" program=<main installation path>\dSPACERCPHIL2018-B\MotionDesk\Bin\MotionDesk.exe" dir=in action=allow profile=any description="Allow dSPACE MotionDesk to receive motion data via network."`

Required open TCP/IP network ports If you are using third-party firewall software on your host PC, ensure that the TCP/IP communication of dSPACE software is not blocked:

- VEOS requires the following open TCP/IP network ports: 111 (TCP and UDP), 3702 (UDP), 7214 (TCP and TCP6), 8090 (TCP), 9923 (UDP), 15000 (UDP), 49152 ... 65535 (TCP, TCP6 and UDP)
- dSPACE Installation Manager and CodeMeter licensing software require the following open TCP/IP network ports:
 - 22350 (TCP and UDP) for communication in a LAN network (if not changed from the default setting).
 - 22352 (TCP and UDP): To access CodeMeter WebAdmin via http.
 - 22353 (TCP and UDP): To access CodeMeter WebAdmin via https.
- dSPACE Help requires an open TCP/IP network port for interprocess communication between its components. The default port number is 11000. If this port number is already being used, another free port is used automatically. The related processes can be identified via the following prefixes: `HelpApsLayer<xxx>`, `HelpInstaller<xxx>`.

Run-Time Compatibility of dSPACE Software

Definition

Run-time compatibility means that:

- dSPACE products can be used in parallel after software installation, even if they are installed in different folders.
- dSPACE products without interaction can run independently of each other.

Compatibility of products in dSPACE Release 2018-B

dSPACE recommends using only software products from the same dSPACE Release. This ensures maximum run-time compatibility.

Observe the following points:

- Limitations regarding run-time compatibility in the dSPACE tool chain might occur if products from different dSPACE Releases are used together.

If dSPACE products interact directly (through automation interfaces) or indirectly (through common file types like A2L), limitations might apply. For minor limitations, refer to the relevant product documentation. The major limitations are described in the following.

In rare cases, an additional patch must be installed for a product to achieve run-time compatibility. For more information on the patch and whether a patch is required, refer to <http://www.dspace.com/go/CompPatch>.

- RCP and HIL software products (of Release 2018-B) cannot be used in combination with RCP and HIL software products from earlier dSPACE Releases.

Major limitation for working with a SCALEXIO system The products for working with a SCALEXIO system must be compatible. This is guaranteed only for products delivered with the same dSPACE Release. Contact dSPACE for more information.

Compatibility of real-time applications loaded to a DS1005, DS1006, DS1103, DS1104 or MicroAutoBox platform If a real-time application is loaded to one of these platforms with a software product of dSPACE Release 2016-B or later, software products of dSPACE Release 2016-A (and earlier) do not detect that the loaded real-time application is the same as the real-time application stored on your host PC. In this case, you cannot work with the related software product without restrictions.

This also applies if you load a real-time application with a software product of dSPACE Release 2016-A or earlier and use software products of dSPACE Release 2016-B or later, for example, for experimenting.

Combining dSPACE products from earlier Releases

For more information and notes on the combined use of different products from and with earlier Releases, refer to http://www.dspace.com/go/ds_sw_combi.

Limitations for Using Windows Features

Motivation

Some limitations apply using dSPACE software in conjunction with features of Windows.

Fast user switching not supported

dSPACE software does not support the fast user switching feature of Windows.

Closing dSPACE software before PC shutdown

The shutdown process of Windows operating systems might cause some required processes to be aborted although they are still being used by dSPACE

software. To avoid a loss of data, it is recommended to close the dSPACE software manually before shutting down the PC.

User Account Control

It is recommended to disable the Windows User Account Control (UAC) during the installation of dSPACE software. If you cannot disable UAC, note the following Windows behavior: If UAC is enabled, the setup programs use the administrator account instead of the user account. Therefore, it is important that the administrator account has access to the required drives, particularly the required network drives.

USB devices

If you connect dSPACE USB devices that use cables with optoisolation to the PC for the first time, there might be a message that the device driver software was not installed successfully. However, the dSPACE device will work properly later on.

FIPS support

dSPACE software was not developed for or tested against the FIPS PUB 140-2 U.S. government computer security standard (Security Requirements for Cryptographic Modules). Therefore, dSPACE products are not guaranteed to work if the respective setting is enabled in Windows. By default, the setting is disabled. For more information on FIPS, refer to <https://technet.microsoft.com/en-us/library/security/cc750357.aspx>.

Long paths

dSPACE software does not support the long path syntax of the Windows API. If a path that exceeds 260 characters is used directly or indirectly, the behavior of the dSPACE software is not defined.

Enabling Windows 8dot3name creation option

Note

It is strongly recommended that the Windows 8dot3name creation option is enabled for all drives (drives used for installation and drives used for work) before you install third-party software, such as MATLAB®/Simulink®, and the dSPACE software.

If the option is disabled during software installation, serious errors can occur when you run the dSPACE software. For example, the build process might be aborted. To repair an installation that was installed while the 8dot3name creation option was disabled, you have to install dSPACE software and required third-party software again.

For instructions on checking the setting and enabling the option, refer to <http://www.dspace.com/faq?346> or to the Microsoft Windows documentation.

Settings in Windows for user locale and system locale must match

MATLAB reads the user locale and system locale settings that are specified in Windows operating systems. The user locale and the system locale must match. If these settings are not the same, the system might not behave as expected when working with MATLAB and dSPACE software.

For instructions on checking and changing the settings, refer to https://www.mathworks.com/help/matlab/matlab_env/setting-locale-on-windows-platforms.html?s_tid=gn_loc_drop.

This affects all MATLAB versions and all Windows operating systems, that are supported by dSPACE.

Valid for Windows 10: Microsoft .NET Framework 3.5 feature must be enabled

Microsoft .NET Framework 3.5 feature must be installed and enabled. If Microsoft .NET Framework 3.5 is not enabled, the dSPACE software installation is interrupted and an error message is displayed.

A

- ASM Base InCylinder Blockset
 - migration 53
- ASM Brake Hydraulics Blockset
 - new features 54
- ASM Diesel Engine Blockset
 - migration 55
 - new features 55
- ASM Diesel Exhaust Blockset
 - migration 57
- ASM Diesel InCylinder Blockset
 - migration 58
- ASM Drivetrain Basic Blockset
 - migration 59
 - new features 59
- ASM Electric Components Blockset
 - migration 60
 - new features 60
- ASM Engine Gasoline Basic Blockset
 - migration 65
- ASM Engine Gasoline Blockset
 - migration 66
- ASM Environment Blockset
 - migration 64
 - new features 63
- ASM Gasoline InCylinder Blockset
 - migration 67
 - new features 67
- ASM KnC
 - migration 69
- ASM Optimizer
 - new features 69
- ASM Traffic Blockset
 - new features 70
- ASM Truck Blockset
 - migration 72
- ASM Turbocharger Blockset
 - migration 74
- ASM Utils
 - new features 75
- ASM Vehicle Dynamics Blockset
 - migration 76
 - new features 76
- AutomationDesk
 - migration 82
 - new features 79

B

- Bus Manager (stand-alone)
 - migration 88
 - new features 85

C

- Common Program Data folder 14
- CommonProgramDataFolder 14
- ControlDesk
 - migration 108
 - new features 98

D

- DCI Configuration Tool
 - new features 115
- discontinuation
 - hardware 18
 - Python 2.7 19
 - software 19
 - software support 19
- Discontinuation 18
- Documents folder 14
- DocumentsFolder 14
- DS1103 PPC Controller Board
 - planned discontinuation 18
- dSPACE FlexRay Configuration Package
 - new features 117
- dSPACE Python Extensions
 - new features 119
- dSPACE XIL API
 - migration 123
 - new features 121

E

- ECU Interface Manager
 - migration 128
 - new features 127

F

- Firmware Manager
 - new features 131

G

- general enhancements and changes 17

H

- host PC software
 - MATLAB 265
 - operating system 266

K

- key features 22

L

- Limitations
 - TargetLink
 - obsolete limitations 253
- limitations for using Windows features 269
- Local Program Data folder 14
- LocalProgramDataFolder 14

M

- MATLAB
 - requirements 265
 - supported releases 265
- MicroAutoBox I/O boards
 - planned discontinuations 19
- migration
 - ASM Base InCylinder Blockset 53

- ASM Diesel Engine Blockset 55
- ASM Diesel Exhaust Blockset 57
- ASM Diesel InCylinder Blockset 58
- ASM Drivetrain Basic Blockset 59
- ASM Electric Components Blockset 60
- ASM Engine Gasoline Basic Blockset 65
- ASM Engine Gasoline Blockset 66
- ASM Environment Blockset 64
- ASM Gasoline InCylinder Blockset 67
- ASM KnC 69
- ASM Truck Blockset 72
- ASM Turbocharger Blockset 74
- ASM Vehicle Dynamics Blockset 76
- AutomationDesk 82
- Bus Manager (stand-alone) 88
- ControlDesk 108
- dSPACE XIL API 123
- ECU Interface Manager 128
- Model Compare 135
- ModelDesk 139
- MotionDesk 150
- Real-Time Testing 152
- RTI 154
- RTI Bypass Blockset 155
- RTI CAN MultiMessage Blockset 159
- RTI FPGA Programming Blockset 165
- RTI LIN MultiMessage Blockset 168

- Model Compare
 - migration 135
 - new features 133
- model management
 - new features 187
- ModelDesk
 - migration 139
 - new features 137
- MotionDesk
 - migration 150
 - new features 149

N

- new features
 - ASM Brake Hydraulics Blockset 54
 - ASM Diesel Engine Blockset 55
 - ASM Drivetrain Basic Blockset 59
 - ASM Electric Components Blockset 60
 - ASM Environment Blockset 63
 - ASM Gasoline InCylinder Blockset 67
 - ASM Optimizer 69
 - ASM Traffic Blockset 70
 - ASM Utils 75
 - ASM Vehicle Dynamics Blockset 76
 - AutomationDesk 79
 - Bus Manager (stand-alone) 85
 - ControlDesk 98
 - DCI Configuration Tool 115
 - dSPACE FlexRay Configuration Package 117
 - dSPACE Python Extensions 119
 - dSPACE XIL API 121
 - ECU Interface Manager 127
 - Firmware Manager 131
 - Model Compare 133

- model management 187
- ModelDesk 137
- MotionDesk 149
- Real-Time Testing 151
- RTI CAN MultiMessage Blockset 157
- RTI FPGA Programming Blockset 163
- RTI LIN MultiMessage Blockset 167
- RTI/RTI-MP 153
- RTLib 153
- SCALEXIO firmware 171
- signal & parameter management 187
- SystemDesk 200
- test management 184
- VEOS 259
- workflow management 190
- new general features
 - SYNECT 180
- new product
 - Sensor Simulation 175
- not supported MATLAB features (R2018b)
 - RTI/RTI-MP 153

P

- planned discontinuation
 - DS1103 PPC Controller Board 18
 - MicroAutoBox I/O boards 19
- product overview 19
- Python 2.7
 - discontinuation 19
- Python distribution 17

R

- Real-Time Testing
 - migration 152
 - new features 151
- requirements
 - host PC software
 - MATLAB 265
 - operating system 266
- RTI Bypass Blockset
 - migration 155
- RTI CAN MultiMessage Blockset
 - migration 159
 - new features 157
- RTI FPGA Programming Blockset
 - migration 165
 - new features 163
- RTI LIN MultiMessage Blockset
 - migration 168
 - new features 167
- RTI/RTI-MP
 - new features 153
 - not supported MATLAB features (R2018b) 153
- RTLib
 - new features 153

S

- SCALEXIO firmware
 - new features 171

- Sensor Simulation
 - new product 175
- signal & parameter management
 - new features 187
- supported MATLAB releases 265
- SYNECT
 - new general features 180
- system requirements
 - operating system 266
- SystemDesk
 - new features 200

T

- TargetLink
 - AUTOSAR features, new
 - supported releases 213
 - code changes
 - migration 236
 - code generator options
 - backward compatibility 232
 - changed default value 232
 - discontinued features 253
 - migrating to
 - new version 223
 - migration
 - code changes 236
 - obsolete limitations 253
 - new API functions 219
 - new code generator options 218
 - new features 208
 - general changes 220
 - general enhancements 220
 - new version
 - migrating to 223
 - target support
 - discontinued compiler versions 216
 - discontinued evaluation boards 216
 - new compiler versions 216
 - new evaluation boards 216
 - supported targets 216
- TargetLink Data Dictionary
 - migrating to
 - new version 223
 - migration 223
 - discontinued documentation 223
 - manually upgrading libraries and models 228
 - upgrading existing data dictionaries 226
 - new features 208
 - new version
 - migrating to 223
 - test management
 - new features 184

U

- user documentation
 - printed documents 18

V

- VEOS

- new features 259
- version history 19

W

- Windows
 - limitations 269
- workflow management
 - new features 190