dSPACE Release

New Features and Migration

Release 2015-B – November 2015

Embedded Success dSPACE

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

To contact dSPACE if you have problems and questions, fill out the support request form provided on the website at http://www.dspace.com/go/supportrequest.

The request form helps the support team handle your difficulties quickly and efficiently.

In urgent cases contact dSPACE via phone: +49 5251 1638-941 (General Technical Support)

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

Contents

About This Document 11

Document Symbols and Conventions 11

Accessing Online Help and PDF Files 12

Overview of dSPACE Release 2015-B 15

General Enhancements and Changes 15

64-Bit Version of RCP and HIL Software 19

Product Version Overview 20

New Product Key Features 24

Aspects of Migrating from Previous Releases 33

Migrating to dSPACE Release 2015-B 33

Changes to TRC File Generation 35

Basics on the TRC File Changes 35

Migrating Changes in Software that Generates TRC Files 41

Migrating Changes in Software That Uses TRC Files 42

Changes to the Python 2.7 Distribution 45

Main Changes in Python 2.7 45

Main Changes of the dSPACE Python Distribution 46

General Information on Using Python Installations 46

Enhancements to the Standard Python 2.7 Distribution 47

AutomationDesk 49

New Features of AutomationDesk 5.1 49

Automotive Simulation Models (ASM) 53

ASM Base InCylinder Blockset 54

New Features of ASM Base InCylinder Blockset 2.1 54

Migrating to ASM Base InCylinder Blockset 2.1 54

ASM Diesel Engine Blockset.. 55

New Features of ASM Diesel Engine Blockset 2.2.................. 55

Migrating to ASM Diesel Engine Blockset 2.2........................ 57

ASM Diesel Exhaust Blockset.. 59

Migrating to ASM Diesel Exhaust Blockset 2.1.1.................... 59

ASM Diesel InCylinder Blockset.. 60

New Features of ASM Diesel InCylinder Blockset 2.1.............. 60

Migrating to ASM Diesel InCylinder Blockset 2.1.................... 61

ASM Drivetrain Basic Blockset.. 62

New Features of ASM Drivetrain Basic Blockset 4.1.1............. 62

Migrating to ASM Drivetrain Basic Blockset 4.1.1................... 62

ASM Electric Components Blockset.. 63

New Features of ASM Electric Components Blockset 3.1........ 63

Migrating to ASM Electric Components Blockset 3.1.............. 64

ASM Environment Blockset.. 66

New Features of ASM Environment Blockset 4.3.................... 66

Migrating to ASM Environment Blockset 4.3.......................... 66

ASM Gasoline Engine Basic Blockset.. 67

New Features of ASM Gasoline Engine Basic Blockset 2.0.2... 67

Migrating to ASM Gasoline Engine Basic Blockset 2.0.2......... 68

ASM Gasoline Engine Blockset... 70

New Features of ASM Gasoline Engine Blockset 3.2............... 70

Migrating to ASM Gasoline EngineBlockset 3.2...................... 72

ASM Gasoline InCylinder Blockset.. 74

New Features of ASM Gasoline InCylinder Blockset 2.1.......... 74

Migrating to ASM Gasoline InCylinder Blockset 2.1................ 75

ASM Optimizer.. 76

New Features of ASM Optimizer 1.7...................................... 76

Migrating to ASM Optimizer Blockset 1.7.............................. 77

ASM Traffic Blockset.. 78

New Features of ASM Traffic Blockset 3.3............................. 78

Migrating to ASM Traffic Blockset 3.3................................... 79

ASM Trailer Blockset.. 80

New Features of ASM Trailer Blockset 2.4............................. 80

Migrating to ASM Trailer Blockset 2.4................................... 81

ASM Truck Blockset... 82

New Features of ASM Truck Blockset 2.3.............................. 82

Migrating to ASM Truck Blockset 2.3.................................... 83

ASM Turbocharger Blockset........ 84
New Features of ASM Turbocharger Blockset 3.1.1........ 84
Migrating to ASM Turbocharger Blockset 3.1.1........ 85
ASM Vehicle Dynamics Blockset........ 87
New Features of ASM Vehicle Dynamics Blockset 3.2........ 87
Migrating to ASM Vehicle Dynamics Blockset 3.2........ 88

ConfigurationDesk 89
ConfigurationDesk – Implementation........ 90
New Features of ConfigurationDesk 5.4 (Implementation Version)........ 90
Migrating to ConfigurationDesk 5.4........ 95

Container Management 97
New Features of Container Management........ 97

ControlDesk Next Generation 99
New Features of ControlDesk Next Generation (ControlDesk 5.5)........ 100
New Features of Platform Management and Platforms/Devices (ControlDesk 5.5)........ 100
New Variable Management Features (ControlDesk 5.5)........ 103
New Visualization and Instrument Features (ControlDesk 5.5)........ 105
New Measurement and Recording Features (ControlDesk 5.5)........ 107
New Bus Navigator Features (ControlDesk 5.5)........ 108
New Data Set Management Features (ControlDesk 5.5)........ 109
New Signal Editor Features (ControlDesk 5.5)........ 109
New Electrical Error Simulation Features (ControlDesk 5.5).. 110
New Automation Features (ControlDesk 5.5)........ 111
Migrating to ControlDesk Next Generation (ControlDesk 5.5)........ 113
Discontinuations in ControlDesk........ 113
Migrating to ControlDesk Next Generation (ControlDesk 5.5)........ 116

DCI Configuration Tool 123
New Features of the DCI Configuration Tool 3.5........ 123

dSPACE FlexRay Configuration Package 125

New Features of dSPACE FlexRay Configuration Package 3.6 125

dSPACE HIL API .NET 127

New Features of dSPACE HIL API .NET 2.0 127

dSPACE Python Extensions 129

New Features of dSPACE Python Extensions 2.0 129

dSPACE XIL API 133

New Features of dSPACE XIL API 2015-B 133

ECU Interface Manager 135

New Features of ECU Interface Manager 1.7 135

Migrating to ECU Interface Manager 1.7 136

Firmware Manager 139

New Features of Firmware Manager 2.0 139

Model Compare 141

New Features of Model Compare 2.6 141

Migration to Model Compare 2.6 143

ModelDesk 145

New Features of ModelDesk 4.2 145

Model Interface Package for Simulink 147

Features of the Model Interface Package for Simulink 3.1 147

MotionDesk 149

New Features of MotionDesk 3.7 149

Migrating to MotionDesk 3.7 150

Real-Time Testing 151

New Features of Real-Time Testing 2.6 151

RTI/RTI-MP and RTLib 153

New Features of RTI/RTI-MP and RTLib 153

Migration Aspects of RTI/RTI-MP and RTLib 156

RTI Bypass Blockset 157

New Features of the RTI Bypass Blockset 3.5 157

Migrating to RTI Bypass Blockset 3.5 158

RTI CAN MultiMessage Blockset 161

New Features of the RTI CAN MultiMessage Blockset 4.2 161

Migrating to RTI CAN MultiMessage Blockset 4.2 162

RTI Electric Motor Control Blockset 165

New Features of RTI Electric Motor Control Blockset 1.2 165

RTI FPGA Programming Blockset 167

New Features of the RTI FPGA Programming Blockset 3.0 167

Migrating to RTI FPGA Programming Blockset 3.0 170

RTI LIN MultiMessage Blockset 173

New Features of the RTI LIN MultiMessage Blockset 2.5.1 173

Migrating to RTI LIN MultiMessage Blockset 2.5.1 173

SCALEXIO Firmware 175

New Features of the SCALEXIO Firmware 3.3 175

SystemDesk 177

New Features of SystemDesk 4.5 178

New General Features 178

Modeling Software Architectures 179

Working with Diagrams 179

Configuring ECUs 181

Simulating Systems 187

Automating SystemDesk 187

Variation Binding 188

Migrating to SystemDesk 4.5 190
Migrating to SystemDesk 4.5 190

TargetLink

TargetLink 191
New Features of TargetLink 4.1 and TargetLink Data Dictionary 4.1 192
Modeling in Simulink or Stateflow 192
Newly Supported Simulink Blocks 193
Improvements to Custom Look-up Tables 193
Support of Simulink's Simplified Mode and IC Structures 194
Support of Structures in Stateflow Action Language 194
Code Generation Core Functionality 195
MISRA-C Compliance 195
Improved Code Efficiency 196
Component-Based Development 199
Improvements to Function Reuse 199
AUTOSAR 200
Supported AUTOSAR Releases 200
Support of NvData Communication 201
Data Transformation 201
Activation Reasons of Runnables 201
Port-Defined Argument Values 202
Miscellaneous AUTOSAR Features 202
Target Simulation (PIL) 203
Changes in the Target Simulation Modules 203
Folder for TSM Extensions 205
Data Dictionary and Data Management 206
Improvements to the Data Dictionary 206
New DD MATLAB API Functions 207
Referencing DD CodegenOptions Objects at TargetLink Main Dialog Block 208
Code Generator Options 208
New Code Generator Options 208
Tool Chain Integration 209
Exporting Functional Mock-up Units 210
Requirement Information in the Data Dictionary 210
Other 211
General Enhancements and Changes 211
TargetLink Demos 213

API Functions and Hook Functions.. 215
New API Functions.. 215
New Hook Functions... 216
Migrating to TargetLink 4.1 and TargetLink Data Dictionary 4.1... 218
Upgrade of Models, Libraries, and Data Dictionaries................. 219
Migrating to TargetLink 4.1... 219
How to Upgrade a Data Dictionary with Included DD Files... 220
How to Manually Upgrade Libraries and Models Via the API... 222
Code Generator Options.. 223
Migration Aspects Regarding Code Generator Options........ 223
API Functions and Hook Functions.. 226
Changes in TargetLink and TargetLink Data Dictionary API Functions... 226
AUTOSAR-Related Migration Aspects.................................... 226
AUTOSAR-Related Migration Aspects................................. 226
Code Changes.. 227
Code Changes.. 227
Other.. 242
Various Migration Aspects... 242
Obsolete... 244
Obsolete Limitations... 244
Changes in Future TargetLink Versions.................................. 245
Features to Be Discontinued.. 245
API Functions to Be Discontinued...................................... 246

VEOS 247

New Features of VEOS 3.5.. 247
Migrating to VEOS 3.5.. 250

Compatibility Information 253

Supported MATLAB Releases... 254
Operating System.. 255
Run-Time Compatibility of dSPACE Software...................... 257
Limitations for 64-Bit Windows Operating Systems in Combination with 32-Bit dSPACE Software........................ 258

Products on the 64-Bit dSPACE DVD Set and Their MATLAB Support 259

General Limitations for Windows 7 261

Index 263

About This Document

Contents

This document informs you about the new features of all the dSPACE software products in Release 2015-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

Document Symbols and Conventions	11
Accessing Online Help and PDF Files	12

Document Symbols and Conventions

Symbols

The following symbols may be used in this document.

⚠	Indicates a general hazard that may cause personal injury of any kind if you do not avoid it by following the instructions given.
⚡	Indicates the danger of electric shock which may cause death or serious injury if you do not avoid it by following the instructions given.
⚠	Indicates a hazard that may cause material damage if you do not avoid it by following the instructions given.
🛑	Indicates important information that should be kept in mind, for example, to avoid malfunctions.

💡 Indicates tips containing useful information to make your work easier.

Naming conventions

The following abbreviations and formats are used in this document:

%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.

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

📝 Indicates that a link refers to another document, which is available in dSPACE HelpDesk.

Special folders

Some software products, for example, ControlDesk Next Generation and AutomationDesk, use the following special folders:

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

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

Documents folder A standard folder for user-specific documents.

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

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

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

Accessing Online Help and PDF Files

Objective

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

Online help

You can access the online help, dSPACE HelpDesk, as follows:

Windows Start menu Select Start – (All) Programs – <ProductName> – dSPACE HelpDesk (<ProductName>) to open dSPACE HelpDesk with the start page of the selected product displayed. You can also navigate and search in the user documentation of any other installed software product and its supported hardware.

Context-sensitive Press the **F1** key or click the Help button in the dSPACE software to get help on the currently active context.

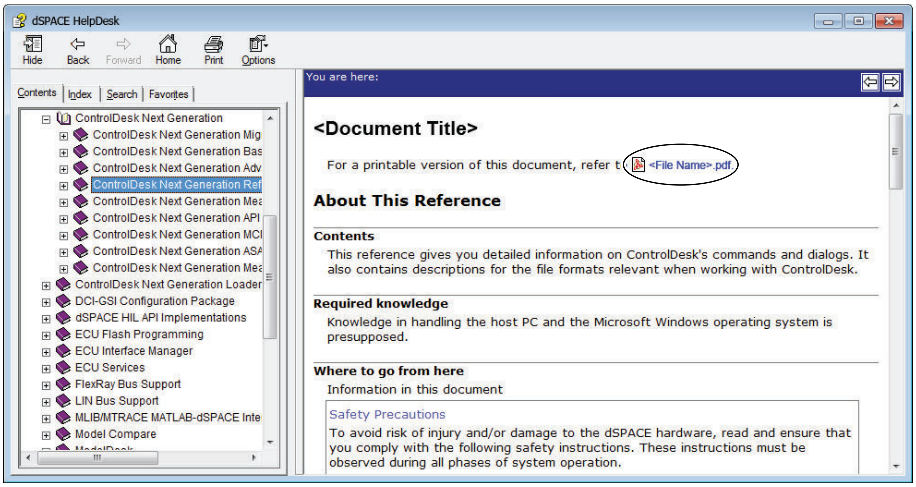
In some software products, context-sensitive help is not available.

Help menu in the dSPACE software On the menu bar, select Help – Contents or Help – Search (not available in all software products) to open dSPACE HelpDesk. It opens at the start page of the currently active product. You can also navigate and search in the user documentation of any other installed software product and its supported hardware.

PDF files

You can access the PDF files as follows:

dSPACE HelpDesk Click the PDF link at the beginning of a document:


dSPACE HelpDesk
Hide
Back
Forward
Home
Print
Options
Contents
Index
Search
Favorites
ControlDesk Next Generation
ControlDesk Next Generation Mig
ControlDesk Next Generation Bas
ControlDesk Next Generation Adv
ControlDesk Next Generation Ref
ControlDesk Next Generation Mea
ControlDesk Next Generation API
ControlDesk Next Generation MCI
ControlDesk Next Generation ASA
ControlDesk Next Generation Mea
ControlDesk Next Generation Loader
DCI-GSI Configuration Package
dSPACE HIL API Implementations
ECU Flash Programming
ECU Interface Manager
ECU Services
FlexRay Bus Support
LIN Bus Support
MLIB/MTRACE MATLAB-dSPACE Inte
Model Compare
You are here:

<Document Title>

For a printable version of this document, refer t <File Name>.pdf

About This Reference

Contents

This reference gives you detailed information on ControlDesk's commands and dialogs. It also contains descriptions for the file formats relevant when working with ControlDesk.

Required knowledge

Knowledge in handling the host PC and the Microsoft Windows operating system is presupposed.

Where to go from here

Information in this document

Safety Precautions

To avoid risk of injury and/or damage to the dSPACE hardware, read and ensure that you comply with the following safety instructions. These instructions must be observed during all phases of system operation.

Overview of dSPACE Release 2015-B

Objective

Gives you an overview of the new key features in Release 2015-B and information about unchanged products.

Where to go from here

Information in this section

General Enhancements and Changes	15
64-Bit Version of RCP and HIL Software	19
Product Version Overview	20
New Product Key Features	24

General Enhancements and Changes

Objective

The following new features and changes concern several dSPACE products.

Support of new dSPACE hardware

With dSPACE Release 2015-B, new dSPACE hardware is introduced:

- *DS2655M2 I/O Module*

  This SCALEXIO module can be mounted on a DS2655 FPGA Base Board and is supported by the RTI FPGA Programming Blockset, refer to *New Features of the RTI FPGA Programming Blockset 3.0* on page 167.

Distribution of 32-bit and 64-bit software

The dSPACE software is distributed on two DVD sets, each with the same content but with the following differences:

- One set with two 32-bit DVDs containing only 32-bit dSPACE software products (e.g., to support 32-bit MATLAB versions).
- One set with two 64-bit DVDs containing:
  - All MATLAB-related dSPACE products which support 64-bit MATLAB versions.
  - All 32-bit dSPACE products which also support 64-bit MATLAB versions.
  - All 32-bit dSPACE products that do not relate to MATLAB (e.g., ControlDesk Next Generation).

  > You can therefore install dSPACE software from the 64-bit DVD set without changing to the 32-bit DVD set during the installation procedure.

  For a list of all dSPACE products contained on the 64-bit dSPACE DVD set and their MATLAB support, refer to *Products on the 64-Bit dSPACE DVD Set and Their MATLAB Support* on page 259.

Contents of DVD sets

The dSPACE software is provided on two disks for each DVD set (32-bit and 64-bit). The disks contain the following dSPACE software packages and main products:

- Disk 1:
  - AutomationDesk 5.1
  - ControlDesk Next Generation (ControlDesk 5.5)
  - TargetLink 4.1
  - Model Compare 2.6

    > **Product use prohibited in United States**
    >
    > You are not licensed to use Model Compare in the United States. You are not allowed to use or permit others to use this product in the United States or in any way that violates the laws of the United States.
  - SystemDesk 4.5 (supports AUTOSAR 4.x)
  - VEOS 3.5
  - Various other dSPACE software tools

- Disk 2:
  - RCP and HIL software

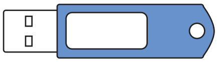
    RCP and HIL software is a generic term for a software package containing several dSPACE software products, such as RTI, ConfigurationDesk, MotionDesk, and ModelDesk.

    Disk 2 does not contain any other dSPACE software products.

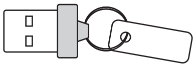
New hardware dongles for dongle licenses

As of dSPACE Release 2014-B, the hardware dongle for dongle licenses is now a CodeMeter instead of a WibuKey. Both are products of WIBU-SYSTEMS and are shown below.

WibuKey dongle



CodeMeter dongle



With dSPACE Release 2014-B, the new CodeMeter hardware dongles are shipped with new dSPACE systems for the first time.

Keep the following compatibility information in mind:

- In general, you can use dSPACE Release 2015-B with an already delivered WibuKey dongle. As of dSPACE Release 2014-B, the drivers for both dongle versions are installed on your host PC. The driver software automatically detects which dongle is used. No further user action is necessary.
- If you want to use dSPACE Release 2014-A and earlier with the new CodeMeter dongle, you have to install dSPACE Installation Manager 3.8 (or later) on your host PC. This version contains the driver for the new dongle. You can download the latest version of dSPACE Installation Manager from http://www.dspace.com/go/imupdate.
- dSPACE Release 6.3 and earlier versions have not been tested for the new CodeMeter dongle. If necessary, contact dSPACE Support.

Restrictions when working with dSPACE HelpDesk

dSPACE HelpDesk is installed in release-specific folders in `C:\Program Files\Common Files\dSPACE` on a 32-bit operating system and in `C:\Program Files(x86)\Common Files\dSPACE` on a 64-bit operating system. For example, if you have installed products from dSPACE Release 2015-A and products from dSPACE Release 2015-B, two dSPACE HelpDesks are available.

Note the following restrictions:

Links to documents might not work and might return the following error message: *Selection is not associated with any topics*. The possible reasons are:

- The documents for the product are not installed, because the product is not included in your license key.
- The documents for the product are installed in another dSPACE HelpDesk. For example, if a product in the current dSPACE Release is unchanged, its user documentation is installed in the dSPACE HelpDesk version that the product setup was created for.

  After you install dSPACE Release 2015-B, you can find the user documentation in dSPACE HelpDesk 2015-A for the following products:

  - dSPACE ECU Flash Programming Tool 2.2.6
  - SYNECT Server 1.4.1

  If you are not sure where to find the user documentation for your product, use the product-specific dSPACE HelpDesk shortcut in the Windows Start menu to open the online help.

Printed user documentation

With dSPACE Release 2015-B, the printed user documentation is not delivered automatically. You can now decide which of the available printed documents you want to have. To order printed documentation, refer to http://www.dspace.com/go/requestreleasematerial.

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

Software support discontinuation

Planned discontinuation of 32-bit software support dSPACE Release 2015-B is the last release supporting 32-bit operating systems and 32-bit MATLAB variants. As of dSPACE Release 2016-A, dSPACE software supports only 64-bit operating systems and only 64-bit MATLAB variants.

Planned discontinuation of MicroAutoBox software support dSPACE Release 2015-B is the last release supporting MicroAutoBox with its variants 1401/1501, 1401/1504, 1401/1505/1506, 1401/1505/1507, and 1401/1507. As of dSPACE Release 2016-A, dSPACE software supports only MicroAutoBox II with its variants 1401/1507, 1401/1511, 1401/1513, 1401/1511/1514, and 1401/1513/1514.

64-Bit Version of RCP and HIL Software

Objective

The RCP and HIL software products support 64-bit MATLAB versions.

Product support in RCP and HIL (64-bit) software

In general, the RCP and HIL (64-bit) software contains the same products as the RCP and HIL software available on dSPACE Release 2015-B (32-bit) DVD.

For an overview of RCP and HIL and all other dSPACE software products concerning the 64-bit MATLAB support, refer to *Products on the 64-Bit dSPACE DVD Set and Their MATLAB Support* on page 259.

Supported MATLAB versions

The RCP and HIL (64-bit) software supports:

- MATLAB R2014a (64-bit)
- MATLAB R2014b (64-bit)
- MATLAB R2015a (64-bit)
- MATLAB R2015b (64-bit)

See also *Supported MATLAB Releases* on page 254.

Supported MEX compiler

For building MEX functions, the RCP and HIL (64-bit) software supports only Microsoft Windows SDK 7.1.

This compiler is a free download from Microsoft. The compiler requires .NET framework 4.0, which is also available at no charge from Microsoft. Download links and instructions for the compiler and framework can be found at http://www.mathworks.com/support/compilers/R2015a/index.html.

You must install this compiler and configure it as a MEX compiler in MATLAB if you intend to use RCP and HIL products that require a MEX compiler such as:

- RTI CAN MultiMessage Blockset
- RTI LIN MultiMessage Blockset
- Automotive Simulation Models
- MotionDesk Blockset

System requirements

The RCP and HIL (64-bit) software requires Windows 7 Enterprise (64-bit version) with Service Pack 1. Other 64-bit operating systems (Windows XP and Windows Vista) are not supported.

The host PC main memory must be at least 4 GB RAM. 8 GB RAM or more is recommended.

See also *Operating System* on page 255.

Product Version Overview

Objective

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			
	2014-A	2014-B	2015-A	2015-B
AutomationDesk	4.1	4.1	5.0	5.1 See *AutomationDesk* on page 49.
Automotive Simulation Models	6.0	7.0	8.0	8.1 See *Automotive Simulation Models (ASM)* on page 53.
ConfigurationDesk	5.1	5.2	5.3	5.4 See *ConfigurationDesk* on page 89.
Container Manager	4.2	4.3	4.3	4.4 See *Container Management* on page 97.
ControlDesk Next Generation	5.2	5.3	5.4	5.5 See *ControlDesk Next Generation* on page 99.
DCI Configuration Tool	3.2.2	3.3	3.4	3.5 See *DCI Configuration Tool* on page 123.
dSPACE CAN API	2.7.1	2.7.1	2.7.1	2.7.4
dSPACE ECU Flash Programming Tool	2.2.5	2.2.5	2.2.6	2.2.6

Product	**dSPACE Release**			
	2014-A	**2014-B**	**2015-A**	**2015-B**
dSPACE FlexRay Configuration Package	3.3	3.4	3.5	3.6 See *dSPACE FlexRay Configuration Package* on page 125.
dSPACE HIL API .NET	1.6	1.6	1.8	2.0 See *dSPACE HIL API .NET* on page 127.
dSPACE Python Extensions	1.6	1.7	1.8	2.0 See *dSPACE Python Extensions* on page 129.
dSPACE XIL API	–	2.0	2015-A	2015-B See *dSPACE XIL API* on page 133.
ECU Interface Manager	1.4.1	1.5	1.6	1.7 See *ECU Interface Manager* on page 135.
Firmware Manager	1.1	1.2	1.3	2.0 See *Firmware Manager* on page 139.
Model Compare	2.4	2.5	2.5	2.6 See *Model Compare* on page 141.
ModelDesk	3.2	4.0	4.1	4.2 See *ModelDesk* on page 145.
Model Interface Package for Simulink	–	–	3.0	3.1 See *Model Interface Package for Simulink* on page 147.
MotionDesk	3.4	3.5	3.6	3.7 See *MotionDesk* on page 149.
MotionDesk Blockset	2.3	2.3.1	2.3.2	2.4 See *MotionDesk* on page 149.
Real-Time Testing	2.3	2.4	2.5	2.6 See *Real-Time Testing* on page 151.

Product	**dSPACE Release**			
	2014-A	**2014-B**	**2015-A**	**2015-B**
RTI[1]	7.2	7.3	7.4	7.5 See *RTI/RTI-MP and RTLib* on page 153.
RTI-MP[2]	7.2	7.3	7.4	7.5 See *RTI/RTI-MP and RTLib* on page 153.
RTI AUTOSAR Package	1.3.1	–	–	–
RTI Bypass Blockset	3.2	3.3	3.4	3.5 See *RTI Bypass Blockset* on page 157.
RTI CAN Blockset	3.2	3.3	3.4	3.4.1
RTI CAN MultiMessage Blockset	3.0	4.0	4.1	4.2 See *RTI CAN MultiMessage Blockset* on page 161.
RTI Electric Motor Control Blockset	–	1.0	1.1	1.2 See *RTI Electric Motor Control Blockset* on page 165.
RTI Ethernet Blockset	1.0	1.1	1.2	1.2
RTI Ethernet (UDP) Blockset	1.3	1.3	1.4	1.4
RTI FPGA Programming Blockset	2.7	2.8	2.9	3.0 See *RTI FPGA Programming Blockset* on page 167.
RTI LIN MultiMessage Blockset	2.3	2.4	2.5	2.5.1 See *RTI LIN MultiMessage Blockset* on page 173.
RTI RapidPro Control Unit Blockset	2.2	2.2	2.2.1	2.2.1
RTI USB Flight Recorder Blockset	1.1	1.2	1.2	1.2
RTI Watchdog Blockset	1.0	1.0	1.0	1.0
SCALEXIO firmware	3.0	3.1	3.2	3.3 See *SCALEXIO Firmware* on page 175.
SYNECT server	1.3.1	1.4	1.4.1	1.4.1
SystemDesk 4.x[3]	4.2	4.3	4.4	4.5 See *SystemDesk* on page 177.

Product	dSPACE Release			
	2014-A	**2014-B**	**2015-A**	**2015-B**
TargetLink/TargetLink Data Dictionary	3.5	4.0	4.0	4.1 See *TargetLink* on page 191.
Variable Editor	1.8	1.8	2.1	2.2
VEOS	3.2	3.3	3.4	3.5 See *VEOS* on page 247.

1) Including the standard I/O blocksets.

2) Including the RTI Gigalink Blockset.

3) Supporting AUTOSAR 4.x

If you have not updated regularly, 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

Objective

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

Information in this topic

AutomationDesk on page 24
ConfigurationDesk (Implementation Version) on page 25
Container management on page 25
ControlDesk Next Generation on page 25
DCI Configuration Tool on page 26
dSPACE FlexRay Configuration Package on page 27
dSPACE HIL API .NET on page 27
dSPACE XIL API on page 27
ECU Interface Manager on page 27
Firmware Manager on page 27
Model Compare on page 27
ModelDesk on page 28
MotionDesk on page 28
Python Extensions on page 28
Real-Time Testing on page 28
RTI, RTI-MP and RTLib on page 28
RTI Bypass Blockset on page 29
RTI CAN MultiMessage Blockset on page 29
RTI Electric Motor Control Blockset on page 29
RTI FPGA Programming Blockset on page 29
RTI LIN MultiMessage Blockset on page 30
SCALEXIO firmware on page 30
SystemDesk on page 30
TargetLink on page 30
VEOS on page 31

AutomationDesk

The new key features of AutomationDesk are:

- Enhancements to signal-based testing, such as the new time tag feature
- Enhancements to the AutomationDesk API
- Usability enhancements: e.g., new editors for specifying expressions and conditions
- New Variable Browser supporting the enhanced trace file generation

- Enhanced Signal Generator support for the DS1007 PPC Processor Board and MicroLabBox

For details on the new features, refer to *New Features of AutomationDesk 5.1* on page 49.

ConfigurationDesk (Implementation Version)

The new key features of ConfigurationDesk are:

- Support of the Simulink Coder features concerning parameter handling introduced with MATLAB R2014a
- Support of precompiled SIC files

For details on the new features, refer to *ConfigurationDesk – Implementation* on page 90.

Container management

The new key feature of container management is:

- Improved assignment of elements to container files in SystemDesk 4.5

For details on the new features, refer to *New Features of Container Management* on page 97.

ControlDesk Next Generation

The new key features of ControlDesk Next Generation (ControlDesk 5.5) are:

- Platform/device enhancements:
  - Support of the new DCI-CAN2 (including CAN FD support)
  - CAN Bus Monitoring device: Support for FIBEX and AUTOSAR system description files
  - FlexRay Bus Monitoring device: Support for AUTOSAR system description files
  - Further platforms that support automatic reconnect
  - Specifying the master for a simulation time group
  - Improvements to virtual validation scenarios
- Variable management enhancements:
  - New Variable Browser
  - Support of structs and struct arrays
  - Display of the original connection path

- Instrument and visualization enhancements:
  - Customizing the connection assignment of variables to instruments
  - Time Plotter and Index Plotter enhancements:
    - Saving Time Plotter data as a new measurement (Time Plotter only)
    - Synchronization of the x-axis
  - Multiswitch enhancement
- Measurement and recording enhancements:
  - ASAM MDF 4.x new default exchange format
  - Handling a large number of measurement rasters
  - Measurement Data API enhancements
- Data set management enhancement:
  - CDFX new default exchange format
- Bus Navigator enhancements:
  - Bus Instrument generation for Bus Manager configurations
  - CAN bus communication replay via MicroLabBox
  - CAN FD support for SCALEXIO and DCI-CAN2
  - CAN Bus Monitoring device: Support for FIBEX and AUTOSAR system description files
- Signal Editor enhancements:
  - Full support of the DS1007
  - Support of DS1202 MicroLabBox
- Electrical error simulation (failure simulation) enhancement:
  - New XIL API EESPort graphical user interface (successor to ControlDesk's Failure Simulation Module)
- Automation enhancements:
  - Measuring and recording look-up tables (maps and curves)
  - Event when adding signals to or removing signals from the measurement signal list

For details on the new features, refer to *New Features of ControlDesk Next Generation (ControlDesk 5.5)* on page 100.

DCI Configuration Tool

The new key feature of the DCI Configuration Tool is:

- Improved A2L file adaptation

For details on the new features, refer to *New Features of the DCI Configuration Tool 3.5* on page 123.

dSPACE FlexRay Configuration Package

The new key features of the dSPACE FlexRay Configuration Tool are:

- Support of AUTOSAR System Template 4.2.1
- Support of opaque byte order format

For details on the new features, refer to *New Features of dSPACE FlexRay Configuration Package 3.6* on page 125.

dSPACE HIL API .NET

The new key features of dSPACE HIL API .NET are:

- Support of the enhanced trace file generation
- Enhanced stimulus support

For details on the new features, refer to *dSPACE HIL API .NET* on page 127.

dSPACE XIL API

The new key features of dSPACE XIL API are:

- Support of the enhanced trace file generation
- Enhanced stimulus support

For details on the new features, refer to *New Features of dSPACE XIL API 2015-B* on page 133.

ECU Interface Manager

The new key features of the ECU Interface Manager are:

- Support of built-in dSPACE Calibration and Bypassing Service configured for DPMEM plug-on devices (PODs)
- Specifying the behavior when execution control insertion fails

For details on the new features, refer to *New Features of ECU Interface Manager 1.7* on page 135.

Firmware Manager

The new key feature of the Firmware Manager is:

- Support of SCALEXIO systems

For details on the new feature, refer to *New Features of Firmware Manager 2.0* on page 139.

Model Compare

The new key features of Model Compare are:

- Hook mechanism that lets you control the creation of dump files
- Copying property values in the Property Inspector
- Model Compare Quick Guide which summarizes the most important features

For details on the new features, refer to *New Features of Model Compare 2.6* on page 141.

ModelDesk

The new key features of ModelDesk are:

- Support of new simulation platforms: DS1007 PPC Processor Board and MicroLabBox
- Tool automation extended for processing and plotting

For details on the new features, refer to *New Features of ModelDesk 4.2* on page 145.

MotionDesk

The new key features of MotionDesk are:

- New instrument feature: Instrument Panel
- New supported platform: MicroLabBox
- Instruments can be assigned to several observers
- Tool automation extended to handle observers

For details on the new features, refer to *New Features of MotionDesk 3.7* on page 149.

Python Extensions

The new key features of dSPACE HIL API Python Implementation for MAPort are:

- Support of the enhanced trace file generation
- Enhanced stimulus support

The new key features of matlablib2 are:

- New and enhanced methods and properties for MATLAB and MATFile instances.

For details on the new features, refer to *dSPACE Python Extensions* on page 129.

Real-Time Testing

The new key features of Real-Time Testing are:

- New supported platform: MicroLabBox

For details on the new features, refer to *New Features of Real-Time Testing 2.6* on page 151.

RTI, RTI-MP and RTLib

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

- Support of the new Simulink Coder features introduced with MATLAB R2015b, reflected by the enhanced trace file generation

- Enhanced RTI support for MicroLabBox
- Support of nonvolatile data handling (NVDATA) for MicroLabBox and DS1007

For details on the new features, refer to *New Features of RTI/RTI-MP and RTLib* on page 153.

RTI Bypass Blockset

The new key features of the RTI Bypass Blockset are:

- Support of XCP 1.3
- CAN FD support

For details on the new features, refer to *New Features of the RTI Bypass Blockset 3.5* on page 157.

RTI CAN MultiMessage Blockset

The new key features of the RTI CAN MultiMessage Blockset are:

- CAN FD support for SCALEXIO systems
- Support of ISO CAN FD protocol
- Sample points for the arbitration phase and data phase of CAN FD messages
- Support of opaque byte order format

For details on the new features, refer to *New Features of the RTI CAN MultiMessage Blockset 4.2* on page 161.

RTI Electric Motor Control Blockset

The new key feature of the RTI Electric Motor Control Blockset is:

- Support of EnDat interface-based encoders for position measurement

For details on the new features, refer to *New Features of RTI Electric Motor Control Blockset 1.2* on page 165.

RTI FPGA Programming Blockset

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

- Extended Xilinx® software support
- Enhancements to the FPGA framework for a DS2655 FPGA Base Board
- New FPGA frameworks for the DS2655M2 Digital I/O Module.
- Enhancement to the FPGA frameworks for the DS2655M1 Multi-I/O Module.

For details on the new features, refer to *New Features of the RTI FPGA Programming Blockset 3.0* on page 167.

RTI LIN MultiMessage Blockset

The new key feature of the RTI LIN MultiMessage Blockset is:

- Support of opaque byte order format

For details on the new features, refer to *New Features of the RTI LIN MultiMessage Blockset 2.5.1* on page 173.

SCALEXIO firmware

The new key features of the SCALEXIO firmware are:

- Support of the DS2655M2 Digital I/O Module

For details on the new features, refer to *New Features of the SCALEXIO Firmware 3.3* on page 175.

SystemDesk

The new key features of SystemDesk 4.5 are:

- Support of AUTOSAR 4.2.2, 4.2.1, 4.1.3, 4.1.2, 4.1.1, and 4.0.3.
- Improvements to SystemDesk's diagrams for graphical modeling
- Support for the basic software module description template according to AUTOSAR
- Improved RTE generation
- Support for the NVRAM manager for virtual validation
- Variation binding of variant-rich models

For details on the new feature, refer to *New General Features* on page 178.

TargetLink

The new key features of TargetLink are:

- AUTOSAR
  - Revision 4.2.1 support
  - NvData communication
  - Data transformation
  - Port-defined argument values
  - Activation reasons
- Improved function reuse now supports referenced models (with multiple instances) and incremental subsystems
- FMU export of TargetLink subsystems to simulate the generated production code in FMI-compliant tools
- Support for Simulink's simplified initialization mode, including initial condition structures for buses
- Support for buses in Stateflow

- Support for storing requirement information in the Data Dictionary via DD RequirementInfo objects that can be referenced from TargetLink blocks and Stateflow Chart objects

For details on all the new features, refer to *New Features of TargetLink 4.1 and TargetLink Data Dictionary 4.1* on page 192.

For details on the TargetLink migration aspects (TargetLink, TargetLink AUTOSAR module, TargetLink Data Dictionary), refer to *Migrating to TargetLink 4.1 and TargetLink Data Dictionary 4.1* on page 218.

VEOS

The new key features of VEOS are:

- More intuitive user interface
- Enabling/disabling the generation of debug information (MSVC, GCC)
- Accessing call stack information in case of an exception

For details on the new features, refer to *VEOS* on page 247.

Aspects of Migrating from Previous Releases

Objective

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

Where to go from here

Information in this section

Migrating to dSPACE Release 2015-B 33

Information in other sections

Changes to TRC File Generation 35

Migrating to dSPACE Release 2015-B

Objective

After you install Release 2015-B, some additional steps might be necessary.

Migrating from dSPACE Release 2015-A

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

Migrating from dSPACE Release 2014-B or earlier

To migrate from dSPACE Release 2014-B or earlier to Release 2015-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 2015-B installed.

For more details 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 at the following locations:

- In the installation folder of the current dSPACE HelpDesk, refer to `C:\Program Files<(x86)>\Common Files\dSPACE\HelpDesk 2015-B\Print\PreviousReleases`.
- On the dSPACE DVDs, refer to `\Doc\Print\PreviousReleases`.
- Download them from www.dspace.com/go/migration. Here you can also find *New Features and Migration* documents for very early releases.

Changes to TRC File Generation

Where to go from here

Information in this section

Basics on the TRC File Changes Basics on the changes of the TRC file generation.	35
Migrating Changes in Software that Generates TRC Files Information on required manual migration.	41
Migrating Changes in Software That Uses TRC Files Information on required manual migration.	42

Basics on the TRC File Changes

Objective

The enhanced code generation leads to improvements for the simulation behavior of the executable application. To profit from these improvements in dSPACE software, the TRC file generation was enhanced.

Enhancements in the generated TRC file

With the MATLAB/Simulink R2014a release, the enhanced code generation by Simulink® Coder^{TM} was introduced to optimize the simulation behavior. It provides a simpler behavior for tuning all parameters and support for referenced models. Additional Simulink Coder functions introduced with MATLAB R2015b now allow dSPACE to fully support these new features via the enhanced TRC file generation.

The main advantages of the enhanced TRC file generation are:

- Same view on model parameters in MATLAB workspace and TRC file

  All tunable model parameters defined by MATLAB workspace variables are available in the top-level Tunable Parameters group in the TRC file. This lets you access global parameters very quickly and independently of the model hierarchy. Modifying the model hierarchy later on will not affect the variable path already specified for layout connections or test scripts.

- Working with MATLAB structures

  If a MATLAB structure is tunable according to the Simulink Coder rules, the structure levels and structure fields are generated into the code.

  This means:

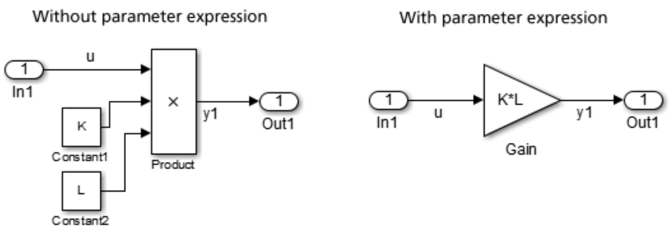
  - Structured parameters are available in the TRC file
  - Non-virtual Simulink buses are represented more efficiently in the TRC file.
  - Bus arrays are available in the TRC file

- Higher performance

  For non-virtual Simulink buses, the performance of code generation and compiling will be highly increased.

- More compact models by using tunable parameter expressions

  Complex workarounds for modelling parameter expressions can be simplified, for example, as shown in the model below. The MATLAB workspace variables K and L are automatically generated as tunable parameters.



- Handling of global parameters for Default parameter behavior = Tunable or Inlined (formerly Inline Parameters option off and on)

The mapping between the configured tunable workspace variables and Simulink.Parameter objects, and variables in the generated code does not depend on the Default parameter behavior option (formerly Inline Parameters option).

- Improved model referencing support

  Simulink referenced models were restricted to using the Inline Parameters option set to On (MATLAB R2015b: Default parameter behavior set to Inlined). Now, the dSPACE toolchain also supports the Default parameter behavior option set to Tunable for referenced models when MATLAB R2015b is used.

- Support of Simulink mask parameters

  Simulink mask parameters are now available in the TRC file and can be accessed by dSPACE software, such as ControlDesk Next Generation.

- Same behavior of Simulink simulation and simulations running on dSPACE platforms

  As a result of the above mentioned enhancements for consistent parameter tuning, the behavior of a Simulink simulation and a simulation on dSPACE platforms will be the same.

Support of Simulink Coder enhancements

For the support of the coder enhancements in the generated TRC file, MathWorks and dSPACE together developed additional build functionality which was released with MATLAB R2015b. The resulting additions of the TRC file syntax required complex modifications in all the TRC file-generating and TRC file-consuming dSPACE products.

The full support of these enhancements is realized with dSPACE Release 2015-B used with MATLAB R2015b. If you use dSPACE Release 2015-B with an earlier MATLAB version, the code generation mainly remains the same.

No migration is required, if you change the dSPACE Release but keep the MATLAB Release.

For an overview of the different behavior, refer to the following table:

dSPACE Release 2015-B used with MATLAB ...			
R2014a	**R2014b**	**R2015a**	**R2015b**
Same behavior of Simulink Coder code generation and dSPACE TRC file generation as with MATLAB R2013b and earlier.	Sharing the same parameter variable across several blocks is supported for block parameters that are defined with an *unstructured* MATLAB workspace variable and *without* expressions. All other block parameter definitions have the previous behavior.		Full support of the above mentioned Simulink coder features.
The behavior must be enforced by executing the Simulink command `revertInlineParametersOffToR2013b` before you use RTI or the Model Interface Package for Simulink.	Internal adaptions to the coder changes are automatically done.		The standard Simulink Coder behavior is used.
Using the Inline Parameters option set to `Off` for referenced models is not supported.			Using the Inline Parameters option set to `Off` for referenced models is supported.[1]

[1] With MATLAB R2015b the setting is similar to Default parameter behavior set to `Tunable`.

Details on the TRC file changes introduced with MATLAB R2015b

The following changes are done with dSPACE Release 2015-B and MATLAB R2015b.

Model Root group The entries in the Model Root group have changed as follows:

- To improve performance and usability, entries for virtual Simulink buses and muxed signals (e.g., `Out1{SubArray1}`), are no longer generated into the variable description.

  This also applies to the labels of these signals.

  This is an incompatible change that requires manual migration, refer to *Migrating Changes in Software that Generates TRC Files* on page 41.

- Entries for non-virtual Simulink buses are now generated as one structured variable in the variable description, e.g., `Out1{MyField}` has changed to `Out1.MyField`.

  This also applies to the labels of non-virtual Simulink buses.

This is an incompatible change that requires manual migration, refer to *Migrating Changes in Software that Generates TRC Files* on page 41.

- Simulink mask parameters are now generated into the variable description at the entries of the related masked subsystems.
- Input signals of signal sink blocks are now generated into the variable description also when you use ConfigurationDesk or VEOS for the build process.
- The Include states and Include derivatives options are now also available for ConfigurationDesk and VEOS.

Tunable Parameters group The entries in the Tunable Parameters group have changed as follows:

- MATLAB workspace variables and Simulink.Parameter objects, which are used as block parameters in the model, are now generated as global variables in the Tunable Parameters group. Internal optimizations during code generation might be the reason that a variable will not be generated into the variable description.

  If a block's parameter definition contains an expression, the local block parameter is no longer available. This is an incompatible change that requires manual migration, refer to *Migrating Changes in Software that Generates TRC Files* on page 41.
- Structured workspace variables and Simulink.Parameter objects that are used as block parameters in the model are now generated as global structured parameters in the Tunable Parameters group.

  The structure has to fulfill the Simulink Coder conditions for a tunable structured parameter.
- Previously, each referenced model of a model referencing hierachy had its own Tunable Parameters group. These groups are no longer generated.

  All global parameters referenced in the top-level model or in the referenced models are generated into the Tunable Parameters group of the top-level model.

  This is an incompatible change that requires manual migration, refer to *Migrating Changes in Software that Generates TRC Files* on page 41.

Handling n-D Lookup Tables With dSPACE Release 2015-B Lookup Table blocks with a dimension higher than 2, such as a 4x3x2 matrix, are no longer automatically divided into two-dimensional slices.

This is an incompatible change that requires manual migration, refer to *Migrating Changes in Software that Generates TRC Files* on page 41.

Data Stores group To improve performance and data consistency with other blocks, the Data Stores group is no longer generated into the variable description.

This is an incompatible change that requires manual migration, refer to *Migrating Changes in Software that Generates TRC Files* on page 41.

Structured variables Structured variables, such as non-virtual buses or tunable structured parameters, are generated into the code and represented in the variable description as a `struct` element. The hierarchy of fields and members in a structured element is described in dot notation, for example, `myStruct.mySubstruct.myValue[0][1]`..

References A variable description now contains block parameters as references. The source of a reference can be a global parameter: e.g., a MATLAB workspace variable available in the Tunable Parameters group, or a mask parameter. For structured parameters, the reference can specify a field of a structure.

For the support of structures and references, the following keywords have been added to the TRC file syntax:

- `array-incr`
- `offs`
- `struct`
- `endstruct`
- `refvar`
- `refgroup`
- `refelem`
- `DEPRECATED`

If you have used one of these keywords as a variable name, it is detected during the generation of the TRC file and not added to the file. There might be definitions in user code that you must check. Otherwise, there might be an error in the software that uses the TRC file.

Up-to-date information

For further information on TRC file generation and the latest migration instructions, refer to the dSPACE website: http://www.dspace.com/go/trc.

Migrating Changes in Software that Generates TRC Files

Objective

Despite the complex changes in the code generation, only a few manual migrations are required. Most of the changes based on the enhancements are automatically migrated by the dSPACE products.

Using MATLAB R2014a with dSPACE Release 2015-B

The behavior of Simulink Coder code generation and dSPACE TRC file generation is the same as with MATLAB R2013b and earlier. This means that none of the Simulink Coder changes are available.

The support is based on the Simulink command `revertInlineParametersOffToR2013b` that must be run before generating code with RTI, or the Model Interface Package for Simulink used with ConfigurationDesk and VEOS. You can run the command either manually after MATLAB started, or by adding it as a call in the `startup.m` or `dsstartup.m` scripts.

Using the Inline Parameters option set to `Off` for referenced models is not supported.

Using MATLAB R2014b with dSPACE Release 2015-B

There is no manual migration required. The variable description contains further global parameters in the Tunable Parameters group for unstructured workspace variables. These global parameters are shared with the corresponding block parameters if the block parameter is not defined with an expression. Writing a new value to one of the global parameters changes the related block parameters too.

Using the Inline Parameters option set to `Off` for referenced models is not supported.

Using MATLAB R2015a with dSPACE Release 2015-B

Same notes as with MATLAB R2014b.

Using MATLAB R2015b with dSPACE Release 2015-B

If you use MATLAB R2015b, the new Simulink Coder features are fully supported. The incompatible changes require migration steps that are described below in general. Detailed instructions are not given, because they depend on various conditions such as the complexity of your model, the software you are using, and the internal structure of your test scripts, for example. There are therefore only some basic examples to show a general way to migrate.

For more details, refer to http://www.dspace.com/go/trc.

Migration steps required in TRC file generating software

dSPACE products that generate TRC files such as RTI, ConfigurationDesk or VEOS, support the new Simulink Coder enhancements as is. There are only the following changes that might require a manual migration to provide information in the variable description.

Update assertion mode (only RTI) The rtiAssertionMode variable is no longer generated into the variable description. The Assertion mode setting on the RTI simulation options page is still available for configuring the mode before you start the build process.

Update access to Data Stores group The Data Stores group is no longer generated into the variable description. Instead of using Data Store Memory blocks, you have to use Data Store Read blocks for read access or the combination of Constant blocks with Data Store Write blocks for write access. Instead of the entries in the Data Stores group you then find entries of the Data Store Read blocks or the Constant blocks in the Model Root group.

For this migration step, it is not required to use dSPACE Release 2015-B. You can also do it with earlier dSPACE releases.

Migrating Changes in Software That Uses TRC Files

Objective

Products that use TRC files, such as ControlDesk Next Generation, use the generated variable description to connect elements in the software with variables in the simulation application. Most of the variable path modifications caused by the TRC file changes can be automatically migrated by the dSPACE products, but for some changes you have to do manual migration in your software product.

Migration steps required in TRC file consuming software

If you have already used ControlDesk, AutomationDesk, or test scripts of any kind that are accessing variables via their variable paths and you rebuild the simulation application with MATLAB R2015b, you have to check whether the variable paths have been discontinued or changed in the variable description.

If you are using ControlDesk you get support for finding inconsistent connections: for example, via specially marked instruments or the `Check Mapping` command in the Signal Editor. In AutomationDesk, variable access is realized via variable aliases. Therefore, modifications in the variable description cannot be automatically recognized.

However, if you are using a variable pool in your project, it is sufficient to update this.

To graphically support the new TRC file features, such as structures and references, ControlDesk and AutomationDesk provide the new Variable Browser.

For the changes that were not able to be migrated automatically in the software, you have to perform the following manual migration.

Issue	Migration Step
Update variable paths of parameters with expressions	Update any connections in ControlDesk or variables defined in test scripts that contain expressions with MATLAB workspace variables, mask parameters, or Simulink Parameter objects. Usually, it is not sufficient to change the variable path to the generated global parameter only to get the required variable access for controlling. You also have to consider any element of the expression or the resulting variable of the block.
Update variable paths of virtual Simulink buses	Update any connections in ControlDesk and test scripts accessing signals within a virtual Simulink bus to directly accessed signal source blocks. As an alternative, you can add a Bus Selector block to your model and then connect the block's output variables.
Update variable paths of non-virtual Simulink buses	Update connections in ControlDesk and test scripts that access signals within a non-virtual Simulink bus to the corresponding field of the structured variable. The formerly generated measurement arrays in the variable description are now represented by struct elements. The syntax of structured elements has changed from `Out1{myField.mySubField}` to `Out1.myField.mySubField`. This might conflict with variable names containing dots.
Update variable paths of Tunable Parameters groups for referenced models	Update any connections in ControlDesk or variables defined in test scripts that refer to tunable parameters of referenced models. The variable path of such a variable must be changed to the top-level Tunable Parameters group.
Update access to Data Stores group	Update any connections in ControlDesk and test scripts that refer to the variables of the discontinued Data Stores group to the variables of the inserted Data Store Read or Write blocks in your model.
Update connections to lookup tables	■ ControlDesk does not recognize all the lookup tables in a TRC file. As a result, these lookup tables are not available as maps or curves, for example, in ControlDesk's Variable Browser. The recognition of lookup tables does not work in the following cases: ■ The table data of the lookup table is contained in a structured parameter. ■ The table data of the lookup table references mask parameters. ■ The lookup table has three or more dimensions. To update the connection to such a lookup table, connect the individual variables of the lookup table in these cases. ■ ControlDesk no longer provides a map or curve for the `tableData` parameter of a lookup table if the `tableData` parameter was parameterized with numeric values. To update the connection to such a parameter, connect the `LookUpTableData` variable of the lookup table (instead of the `tableData` parameter).

ControlDesk automatically migrates variable connections after you rebuild a simulation application with MATLAB R2015b and reload the application's variable description. However, if you then reload the variable description of the simulation application built with a MATLAB Release earlier than R2014a, the migrated variable connections are lost, and you have to update these connections manually.

For more details on TRC file changes, refer to *Basics on the TRC File Changes* on page 35.

Changes to the Python 2.7 Distribution

Objective

Gives you information on the changes in the Python distribution provided by dSPACE.

If you want to migrate from an earlier version of Python to Python 2.7, refer to the migration steps described in the *New Features and Migration* document for dSPACE Release 2013-B.

You can also find this information on the dSPACE website, refer to http://www.dspace.com/go/Python27Migration.

Where to go from here

Information in this section

Main Changes in Python 2.7	45
Main Changes of the dSPACE Python Distribution	46
General Information on Using Python Installations	46
Enhancements to the Standard Python 2.7 Distribution	47

Main Changes in Python 2.7

Objective

Provides information on the changes in the Python 2.7 distribution that comes with dSPACE Release 2015-B.

What's New documentation from the Python Software Foundation

The *What's New* document for the updated Python version is available from the Python Software Foundation:

- What's New for Python 2.7
  http://docs.python.org/2.7/whatsnew/2.7.html

Main Changes of the dSPACE Python Distribution

Objective

The Python distribution provided by dSPACE contains some dSPACE-specific changes.

Components of the dSPACE Python distribution

The Python 2.7 distribution on the dSPACE DVD provides the following Python components.

Python Component	Version
Python core	2.7.9
PyWin32	219.10
Numpy	1.7.1
Matplotlib	1.2.1
WxPython	2.9.4.0
Py2exe	0.6.9
Comtypes	0.6.2
PIL	1.1.7
Python for .NET	2.0p3

General Information on Using Python Installations

Objective

The following information is relevant if you want to use both Python versions on your computer.

Using Python 2.5 and Python 2.7 in parallel

Both Python versions can be used in parallel on your computer, with the following restrictions:

- The file associations for PY and PYW files can only be set to one Python version. This is usually the latest installed Python version.

- Environment variables are used by both Python versions. Their values (e.g., for `PYTHONHOME`), must be set to the Python installation you want to work with. For an overview of the environment variables set by Python, refer to http://docs.python.org/2/using/cmdline.html.

Using dSPACE test automation with both Python versions in parallel

If your test automation scripts use dSPACE Python modules distributed either via the dSPACE Python 2.5 setup or via the dSPACE Python Extensions setup available up to dSPACE Release 2013-A, and you do not want to migrate your scripts, you have to work with both Python versions.

Enhancements to the Standard Python 2.7 Distribution

Objective

There are some dSPACE-specific enhancements to the standard Python 2.7. These either ensure the same behavior as before or solve known bugs. The following enhancements are available with dSPACE Release 2015-B.

Enhancements to solve known Python bugs

The following changes have been made to solve a known bug from Python 2.7:

- Changes to the PyWin32 package from the former versions were adopted.
- The Python for .NET package was fixed to run with .NET 4.5.2.
- The Python for .NET package was fixed to run with WPF user interfaces.

For the latest information on bugs in Python 2.7 and their solutions, refer to http://bugs.python.org.

To identify the PyWin32 files changed by dSPACE, the version number of the files was changed from 219.0 to 219.10.

AutomationDesk

New Features of AutomationDesk 5.1

Information in this topic

General enhancements on page 49

Enhanced user interface on page 49

Support of the enhanced trace file generation on page 50

Enhancements to the libraries on page 50

Dialogs library on page 50

Signal-Based Testing library on page 51

XIL API library on page 50

Enhancements to the COM API on page 51

Discontinuations for future versions on page 51

General enhancements

Enhanced user interface The following user interface enhancements facilitate working with AutomationDesk:

- The new Expression Editor is used to specify conditions according to the ASAM General Expression Syntax (GES): e.g., trigger conditions for capture tasks defined with the XIL API library.
- The Condition Editor has been updated to the same look&feel of the new Expression Editor.
- The Signal Editor comes with some enhancements:
  - New Time Tag feature to graphically specify logical time tags. These tags can then be connected to segments for configuring trigger conditions.
  - Properties of the Signal Editor have been updated.
  - Inherited properties are displayed in a different color.

  - The commands of the editor are now provided by a separate ribbon.
  - Undo and Redo actions are supported by the editor.
- The Library Favorites Viewer comes with some enhancements:
  - You can now add an entire library folder with its contained blocks and data objects to the favorites list.
  - You can import and export the favorites in XML format.
  - Favorites that refer to elements and that are not available in the Library Browser are now detected and displayed by another icon and in red text.
- You can now open a custom library by dragging its related ADL file from a file explorer to the Library Browser.

Support of the enhanced trace file generation AutomationDesk libraries that access a simulation application via a variable description support the new features coming with the enhanced trace file generation. For more details, refer to *Changes to TRC File Generation* on page 35. The Variable Browser used o parameterize the Platform data object and the MAPortConfiguration data object has been updated to support the new TRC file features.

Enhancements to the libraries

The following libraries have been enhanced:

Dialogs library The library supports the following new methods that you can use in an Exec block:

- `EnterGESExpression`

  To open the Expression Editor.
- `EnterPythonExpression`

  To open the Condition Editor.

For further information, refer to *Dialogs* (📖 *AutomationDesk Library Reference*).

XIL API library The XIL API library now provides the Mapping data object. It can be used to provide the variables in your simulation application in XIL API-based format. The XIL API mapping contains the variable name as an abstract identifier and the variable path as the concrete identifier (Alias). To create a XIL API mapping file, you can use the new Mapping Editor, which you can execute from the context menu of the new Mapping Viewer that opens when you double-click a Mapping data object.

There are the following enhancements to stimulus generation using the XIL API SignalGenerator elements:

- Support of MicroLabBox
- If you use a multicore or multiprocessor application on a DS1007 PPC Processor Board, model variables can now be stimulated on each connected application processor.

For more details, refer to *XIL API* (🕮 *AutomationDesk Library Reference*).

Signal-Based Testing library The implementation blocks of the Signal-Based Testing library have been internally enhanced to support the new time tag feature of the Signal Editor.

For more details, refer to *Signal-Based Testing Library* (🕮 *AutomationDesk Library Reference*).

Enhancements to the COM API

The AutomationDesk COM API provides the following enhancement:

- New methods to import and export library favorites.
- New method to import a XIL API mapping to a Mapping data object.

For more details, refer to 🕮 *AutomationDesk API Reference*.

Discontinuations for future versions

The following libraries, automation blocks and data objects will be discontinued in future versions of AutomationDesk:

- Test Framework library

  You should migrate your projects based on the Test Framework library to the Test Builder library. For migration help, refer to http://www.dspace.com/go/TestBuilderMigration.

- Platform Access library

  The Platform Access library is delivered for the last time with dSPACE Release 2016-A. You should migrate your projects based on the Platform Access library to the XIL API library or to the XIL API Convenience library. This provides the MAPort for reading, writing and stimulating variables of a connected platform.

  For migration help, refer to http://www.dspace.com/go/pscta.

- Failure simulation automation blocks in the ControlDeskNG Access library

  ControlDesk's Failure Simulation Module is delivered for the last time with dSPACE Release 2016-A. To prepare electrical error simulation via automation, use the Electrical Error Simulation Port (EESPort) in the XIL API library or in the XIL API Convenience library instead of the failure simulation blocks in the ControlDeskNG Access library.

  For migration help, refer to http://www.dspace.com/go/pscta.

- InitCaptureResultIDFReader and InitCaptureResultIDFWriter automation blocks in the XIL API library

  The InitCaptureResultIDFReader and InitCaptureResultIDFWriter automation blocks are delivered for the last time with dSPACE Release 2016-A. Because the IDF format will be discontinued in future versions, you should replace these automation blocks with the CaptureResultReader and CaptureResultWriter data objects which support the MDF format. For more details, refer to CaptureResultReader (Data Object) (📖 *AutomationDesk Library Reference*) and CaptureResultWriter (Data Object) (📖 *AutomationDesk Library Reference*).

The elements that are planned to be discontinued are specially marked in the Library Browser.

Automotive Simulation Models (ASM)

Where to go from here

Information in this section

ASM Base InCylinder Blockset	54
ASM Diesel Engine Blockset	55
ASM Diesel Exhaust Blockset	59
ASM Diesel InCylinder Blockset	60
ASM Drivetrain Basic Blockset	62
ASM Electric Components Blockset	63
ASM Environment Blockset	66
ASM Gasoline Engine Basic Blockset	67
ASM Gasoline Engine Blockset	70
ASM Gasoline InCylinder Blockset	74
ASM Optimizer	76
ASM Traffic Blockset	78
ASM Trailer Blockset	80
ASM Truck Blockset	82
ASM Turbocharger Blockset	84
ASM Vehicle Dynamics Blockset	87

Information in other sections

Migrating ASM Models (📖 *ASM User Guide*)
Provides general information on the migration of ASM models.

ASM Base InCylinder Blockset

Where to go from here

Information in this section

New Features of ASM Base InCylinder Blockset 2.1	54
Migrating to ASM Base InCylinder Blockset 2.1	54

New Features of ASM Base InCylinder Blockset 2.1

ENGINE_SETUP

The ENGINE_SETUP block now has additional switches for parameterization with ModelDesk:

- Const_max_num_PortInjector_PressureDrop
- Sw_Turbo_Stage[1SingleStage|2TwoStage]
- Sw_Turbo_Model [1phys|2LUT]
- Sw_ExhMan [1phys|2simple|3LUT]
- Sw_InMan[1phys|2LUT]

Until now, these were parameterized by the CPT structure.

The Sw_Turbo_Model_[1phys|2LUT] and Sw_Turbo_Stage_[1SingleStage|2TwoStage] outports are new.

These changes have an effect in the engine model if the turbocharger model contains the Turbo_Adv model and/or the Turbo_2Stage model.

Migrating to ASM Base InCylinder Blockset 2.1

ENGINE_SETUP

The new parameters and inputs are set to dummy values. In migrated models, the original CPT variables are still used for the switches.

ASM Diesel Engine Blockset

Where to go from here

Information in this section

New Features of ASM Diesel Engine Blockset 2.2	55
Migrating to ASM Diesel Engine Blockset 2.2	57

New Features of ASM Diesel Engine Blockset 2.2

Model start script go.m file

The standard start script go.m file for the ASM demo models has been adapted in order to take the changes of the supported simulation platforms into account.

The revised go.m file now contains the following platform options:

- 'RTI': DS1005, DS1006, DS1007, MicroLabBox
- 'SCALEXIO': SCALEXIO hardware
- 'VEOS': VEOS (offline simulation platform)

For more information on the calling procedures of the go.m file, refer to the corresponding file header or *Model Initialization* (📖 *ASM User Guide*).

The start script go.m file also takes the changes to the VEOS target into account. With Release 2015-B, VEOS uses the DSRT target instead of the DSOffSim target (refer to *Migrating to VEOS 3.5* on page 250). The go.m file in the ASM demo projects has been adapted to trigger the correct compiler settings.

ENGINE_SETUP

The ENGINE_SETUP block now has additional switches for parameterization with ModelDesk:

- Const_max_num_PortInjector_PressureDrop
- Sw_Turbo_Stage[1SingleStage|2TwoStage]
- Sw_Turbo_Model [1phys|2LUT]

Until now, this parameterization was done by the SWITCHES_TURBO block.

The Sw_Turbo_Model_[1phys|2LUT] and Sw_Turbo_Stage_[1SingleStage|2TwoStage] outports are new. These changes have an effect in the engine model if the turbocharger model contains the Turbo_Adv model and/or the Turbo_2Stage model.

The turbo model can now be switched from the ModelDesk Engine Setup page if the turbocharger model included in the engine model contains the components.

COMMON_ENGINE_PARAMETERS

The Const_kappa_Fuel parameter has been added for calculations with gaseous fuel.

For this, the block has been expanded by the corresponding outports:

- kappa_Fuel[]
- cv_Fuel[J|[kgK]]
- cp_Fuel[J|[kgK]]

INJECTOR

The block has been fixed for use with 20 cylinders. There was a bug assigning the injection to cylinder 19 instead of to cylinder 20, where it was supposed to be.

UNIT_INJECTOR

The block has been fixed for use with 20 cylinders. There was a bug assigning the injection to cylinder 19 instead of to cylinder 20, where it was supposed to be.

PORTINJECTOR_TIMING

The block can now calculate a higher injection time to account for a lower fuel quantity if the pressure difference between the injector fuel supply and the intake manifold is too low.

The Map_mdot_Fuel_Gain_Red parameter has been added. Two new inports for fuel pressure difference calculation have been created: p_InMan[Pa] and p_Fuel[Pa].

The PORTINJECTOR_TIMING block can be used with multiple air intake engine setups, such as V-engines. The Map_Inj2Cyl and Const_max_num_PortInjector_PressureDrop parameters are used to assign the intake manifold, injection and cylinder.

Test cycles

The Ftp_75 test cycle has been updated with an engine switch-off phase. After a total duration of 1369 seconds, the engine is switched off for 10 minutes. This is then followed by a warm engine restart.

Moreover, the new test cycles WLTC (Worldwide Harmonized Light Vehicle Test Procedure) with three classes (depending on the power-to-mass ratio) have been implemented. The new test cycles can be found in the test cycles folder of the new engine demos.

Migrating to ASM Diesel Engine Blockset 2.2

RAIL_CONTROL_ CRANKBASED

The Const_disable_FMU_q_Inj, Const_enable_FMU_n_Engine, and Const_enable_FMU_q_Inj parameters have been removed, as these thresholds have no functional influence anymore. The change has no functional effect on the model behavior.

The Map_eta_DeltaAngle parameter has been renamed to Map_phi_FMU_energized. The change has no functional effect on the model behavior.

The Map_phi_FMU_FF parameter has been added in order to improve the control strategy by means of a feedforward table.

The Sw_phi_FMU_Update parameter has been added in order to improve the pulsed control mode with a defined pump angle update of the control variable within a pump cycle.

The Map_ValveDelay parameter has been added in order to account for delays in the valve actuation.

The inactive elements in the phi_FMU_energized output vector are now replaced by 999, in parallel to the I/O behavior. A crank angle of 999 will be interpreted as invalid in the high pressure pump plant model. Consequently, the corresponding pump cycle will not deliver any fuel to the high pressure rail.

DPF_REGENERATION

A data type conversion block has been inserted in order to account for data type consistency checks in MATLAB/Simulink. The change has no functional effect and just prevents warnings in MATLAB during Update diagram.

HPP_CRANKBASED

The block has been modified to also account for negative TDC offset definitions.

In order to take pump cycles without FMU actuation into account, the vector containing the FMU actuation information will now be reordered. Based on that, the pump block will consider only the actuation signals that belong to the current pump cycle.

ENGINE_SETUP

The block's parameters will be initialized with dummy values. The new outports will be terminated.

COMMON_ENGINE_ PARAMETERS

The parameters will be initialized with dummy values. The new outports will be terminated.

INJECTOR	The block has been fixed for use with 20 cylinders. The injection that is supposed to be in cylinder 20 will now be correctly assigned to cylinder 20.
UNIT_INJECTOR	The block has been fixed for use with 20 cylinders. The injection that is supposed to be in cylinder 20 will now be correctly assigned to cylinder 20.

ASM Diesel Exhaust Blockset

Migrating to ASM Diesel Exhaust Blockset 2.1.1

DIESEL_OXIDATION_CATALYST

The PT1 term has been moved to apply a delay on not only the pressure drop information but also the output pressure. This modification affects the simulation behavior. Hence, during migration, the link is changed to a former version of the block.

ASM Diesel InCylinder Blockset

Where to go from here

Information in this section

New Features of ASM Diesel InCylinder Blockset 2.1	60
Migrating to ASM Diesel InCylinder Blockset 2.1	61

New Features of ASM Diesel InCylinder Blockset 2.1

Model start script go.m file

The standard start script go.m file for the ASM demo models has been adapted in order to take the changes of the supported simulation platforms into account.

The revised go.m file now contains the following platform options:

- 'RTI': DS1005, DS1006, DS1007, MicroLabBox
- 'SCALEXIO': SCALEXIO hardware
- 'VEOS': VEOS (offline simulation platform)

For more information on the calling procedures of the go.m file, refer to the corresponding file header or *Model Initialization* (📖 *ASM User Guide*).

The start script go.m file also takes the changes to the VEOS target into account. With Release 2015-B, VEOS uses the DSRT target instead of the DSOffSim target (refer to *Migrating to VEOS 3.5* on page 250). The go.m file in the ASM demo projects has been adapted to trigger the correct compiler settings.

Test cycles

The Ftp_75 test cycle has been updated with an engine switch-off phase. After a total duration of 1369 seconds, the engine is switched off for 10 minutes. This is then followed by a warm engine restart.

Moreover, the new test cycles WLTC (Worldwide Harmonized Light Vehicle Test Procedure) with three classes (depending on the power-to-mass ratio) have been implemented. The new test cycles can be found in the test cycles folder of the new engine demos.

Migrating to ASM Diesel InCylinder Blockset 2.1

DPF_REGENERATION

A data type conversion block has been inserted in order to account for data type consistency checks in MATLAB/Simulink. The change has no functional effect.

ASM Drivetrain Basic Blockset

Where to go from here

Information in this section

New Features of ASM Drivetrain Basic Blockset 4.1.1	62
Migrating to ASM Drivetrain Basic Blockset 4.1.1	62

New Features of ASM Drivetrain Basic Blockset 4.1.1

CYCLES

The CYCLES block now accepts the definitions of test cycles with an engine switch-off phase. The engine can now be switched off and restarted during the test to test a warm restart. In the test cycle definition file a new variable (Sw_Engine) can be added. This variable has a value of 0 to switch the engine off and 1 to switch it on.

For an example, refer to the Ftp_75 test cycle in the new Engine demo models. A lower engine speed is also implemented as a new parameter to prevent the test bench from switching the engine off during the execution of a dynamometer test cycle.

Migrating to ASM Drivetrain Basic Blockset 4.1.1

CYCLES

During migration, the new Const_n_Engine_Min parameter for the lower engine speed is added. This parameter will have a default value of 0 to keep the old behavior unchanged.

ASM Electric Components Blockset

Where to go from here

Information in this section

New Features of ASM Electric Components Blockset 3.1	63
Migrating to ASM Electric Components Blockset 3.1	64

New Features of ASM Electric Components Blockset 3.1

Model start script go.m file

The standard start script go.m file for the ASM demo models has been adapted in order to take the changes of the supported simulation platforms into account.

The revised go.m file now contains the following platform options:

- 'RTI': DS1005, DS1006, DS1007, MicroLabBox
- 'SCALEXIO': SCALEXIO hardware
- 'VEOS': VEOS (offline simulation platform)

For more information on the calling procedures of the go.m file, refer to the corresponding file header or *Model Initialization* (📖 *ASM User Guide*).

The start script go.m file also takes the changes to the VEOS target into account. With Release 2015-B, VEOS uses the DSRT target instead of the DSOffSim target (refer to *Migrating to VEOS 3.5* on page 250). The go.m file in the ASM demo projects has been adapted to trigger the correct compiler settings.

ALTERNATOR

The block has a new outport for the execution current: I_Excitation.

ASM_EC_STARDELTA

This block is new. It is used for current (or voltage) conversion from three-phase supply variables to three-phase machine variables, and in the other direction.

PMSM_D_Q_NONLINEAR

A new parameter has been added to switch the machine configuration. The machine can now be modeled in either a star or a delta configuration.

PMSM_MAGNET_ SYNCHRONOUS_ MACHINE_D_Q	A new parameter has been added to switch the machine configuration. The machine can now be modeled in either a star or a delta configuration.
SQUIRREL_CAGE_ ASYNCHRONOUS_ MACHINE_D_Q	A new parameter has been added to switch the machine configuration. The machine can now be modeled in either a star or a delta configuration.
PMSM_CONTROLLER	A new parameter has been added to control the machine according to its configuration.
PMSM_CONTROLLER_ THREE_LEVEL	A new parameter has been added to control the machine according to its configuration.
SCIM_CONTROLLER	A new parameter has been added to control the machine according to its configuration.

Migrating to ASM Electric Components Blockset 3.1

SOFT_ECU_HYBRID_ MANAGER

The block is split in the subcomponent blocks BRAKE_CONTROL, DRIVE_MANAGEMENT, TRQ_REQUEST_COORDINATION, KEY_SIGNALS_ICE, STARTER_ICE, CLUTCH_CONTROL and HYBRID_VEHICLE_SWITCH. The change has no functional effect but it lets you to modify the SoftECU Hybrid Manager easier.

Renamed inports and outports

There are changes in the names of inports and outports:

- v_Stator[a;b;c][V] is renamed to v[a;b;c][V]
- i_Stator[a;b;c][V] to i[a;b;c][V]

This applies to the following blocks:

- PMSM_D_Q_NONLINEAR
- BRUSHLESS_DC_MACHINE_ALPHA_BETA
- PMSM_MAGNET_SYNCHRONOUS_MACHINE_D_Q
- SQUIRREL_CAGE_ASYNCHRONOUS_MACHINE_D_Q
- PMSM_CONTROLLER
- PMSM_CONTROLLER_BASIC
- PMSM_CONTROLLER_THREE_LEVEL
- SCIM_CONTROLLER

- SCIM_CONTROLLER_BASIC
- THREE_PHASE_INVERTER
- THREE_LEVEL_THREE_PHASE_INVERTER
- THREE_PHASE_DCM_INVERTER

Renamed inports

The v_Stator[a;b;c][V] inport is renamed to v[a;b;c][V].

This applies to the following blocks:

- SPACE_VECTOR_MODULATOR
- THREE_LEVEL_SPACE_ VECTOR_MODULATOR

The i_Stator[a;b;c][V] inport is renamed to i[a;b;c][V].

This applies to the following blocks:

- BLDC_CONTROLLER
- BLDC_CONTROLLER_BASIC

ASM Environment Blockset

Where to go from here	Information in this section

New Features of ASM Environment Blockset 4.3	66
Migrating to ASM Environment Blockset 4.3	66

New Features of ASM Environment Blockset 4.3

LATERAL_CONTROL1/ LATERAL_CONTROL2	The steering wheel angle will be reset at a global reset.

Migrating to ASM Environment Blockset 4.3

LATERAL_CONTROL1/ LATERAL_CONTROL2	The steering wheel angle will be reset at a global reset.
ROAD	The distance calculations are executed for a dynamic number of traffic objects. Thus, the corresponding Simulink ports of the ROAD block have a dynamically sized width. There are changes due to Simulink diagnostics.
BASIC_ROAD	There are changes due to Simulink diagnostics.
MANEUVER_SCHEDULER	There are changes due to Simulink diagnostics.
GEAR_SHIFTER	The bevahior after reset has been corrected.

ASM Gasoline Engine Basic Blockset

Where to go from here

Information in this section

New Features of ASM Gasoline Engine Basic Blockset 2.0.2	67
Migrating to ASM Gasoline Engine Basic Blockset 2.0.2	68

New Features of ASM Gasoline Engine Basic Blockset 2.0.2

Model start script go.m file

The standard start script go.m file for the ASM demo models has been adapted in order to take the changes of the supported simulation platforms into account.

The revised go.m file now contains the following platform options:

- 'RTI': DS1005, DS1006, DS1007, MicroLabBox
- 'SCALEXIO': SCALEXIO hardware
- 'VEOS': VEOS (offline simulation platform)

For more information on the calling procedures of the go.m file, refer to the corresponding file header or *Model Initialization* (📖 *ASM User Guide*).

The start script go.m file also takes the changes to the VEOS target into account. With Release 2015-B, VEOS uses the DSRT target instead of the DSOffSim target (refer to *Migrating to VEOS 3.5* on page 250). The go.m file in the ASM demo projects has been adapted to trigger the correct compiler settings.

WALL_FILM

The Reset inport to this block had no effect and was fixed to work correctly.

PORTINJECTOR

The block can now calculate a lower fuel quantity if the pressure difference between the fuel supply and the intake manifold is low. This is used in compressed natural gas engines.

The Map_mdot_Fuel_Gain_Red parameter has been added. Two new inports for fuel pressure difference calculation have been created: p_InMan[Pa] and p_Fuel[Pa].

COMMON_ENGINE_PARAMETERS

The Const_kappa_Fuel parameter has been added for calculations with gaseous fuel.

For this, the block has been expanded by the corresponding outports:

- kappa_Fuel[]
- cv_Fuel[J|[kgK]]
- cp_Fuel[J|[kgK]]

PORTINJECTOR_TIMING

The block can now calculate a higher injection time to account for a lower fuel quantity if the pressure difference between the injector fuel supply and the intake manifold is too low.

The Map_mdot_Fuel_Gain_Red parameter has been added. Two new inports for fuel pressure difference calculation have been created: p_InMan[Pa] and p_Fuel[Pa].

The PORTINJECTOR_TIMING block can be used with multiple air intake engine setups, such as V-engines. The Map_Inj2Cyl and Const_max_num_PortInjector_PressureDrop parameters are used to assign the intake manifold, injection and cylinder.

Test cycles

The Ftp_75 test cycle has been updated with an engine switch-off phase. After a total duration of 1369 seconds, the engine is switched off for 10 minutes. This is then followed by a warm engine restart.

Moreover, the new test cycles WLTC (Worldwide Harmonized Light Vehicle Test Procedure) with three classes (depending on the power-to-mass ratio) have been implemented. The new test cycles can be found in the test cycles folder of the new engine demos.

Migrating to ASM Gasoline Engine Basic Blockset 2.0.2

WALL_FILM

The reset will now work as intended.

PORTINJECTOR

The new parameters and inputs will be set up so that the new functionality has no effect.

ENGINE_SETUP

The parameters will be initialized with dummy values. The new outports will be terminated.

COMMON_ENGINE_PARAMETERS	The parameters will be initialized with dummy values. The new outports will be terminated.
PORTINJECTOR_TIMING	The new parameters and inputs will be set up so that the new functionality has no effect.

ASM Gasoline Engine Blockset

Where to go from here

Information in this section

New Features of ASM Gasoline Engine Blockset 3.2	70
Migrating to ASM Gasoline EngineBlockset 3.2	72

New Features of ASM Gasoline Engine Blockset 3.2

Model start script go.m file

The standard start script go.m file for the ASM demo models has been adapted in order to take the changes of the supported simulation platforms into account.

The revised go.m file now contains the following platform options:

- 'RTI': DS1005, DS1006, DS1007, MicroLabBox
- 'SCALEXIO': SCALEXIO hardware
- 'VEOS': VEOS (offline simulation platform)

For more information on the calling procedures of the go.m file, refer to the corresponding file header or *Model Initialization* (📖 *ASM User Guide*).

The start script go.m file also takes the changes to the VEOS target into account. With Release 2015-B, VEOS uses the DSRT target instead of the DSOffSim target (refer to *Migrating to VEOS 3.5* on page 250). The go.m file in the ASM demo projects has been adapted to trigger the correct compiler settings.

WALL_FILM

The Reset inport to this block had no effect and was fixed to work correctly.

PORTINJECTOR

The block can now calculate a lower fuel quantity if the pressure difference between the fuel supply and the intake manifold is low. This is used in compressed natural gas engines.

The Map_mdot_Fuel_Gain_Red parameter has been added. Two new inports for fuel pressure difference calculation have been created: p_InMan[Pa] and p_Fuel[Pa].

DIRECTINJECTOR

The block has been fixed for use with 20 cylinders. There was a bug assigning the injection supposed to be in cylinder 20 to cylinder 19.

ENGINE_SETUP

The ENGINE_SETUP block now has additional switches for parameterization with ModelDesk:

- Const_max_num_PortInjector_PressureDrop
- Sw_Turbo_Stage[1SingleStage|2TwoStage]
- Sw_Turbo_Model [1phys|2LUT]

Until now, this parameterization was done by the SWITCHES_TURBO block.

The Sw_Turbo_Model_[1phys|2LUT] and Sw_Turbo_Stage_[1SingleStage|2TwoStage] outports are new. These changes have an effect in the engine model if the turbocharger model contains the Turbo_Adv model and/or the Turbo_2Stage model.

The turbo model can now be switched from the ModelDesk Engine Setup page if the turbocharger model included in the engine model contains the components.

COMMON_ENGINE_PARAMETERS

The Const_kappa_Fuel parameter has been added for calculations with gaseous fuel.

For this, the block has been expanded by the corresponding outports:

- kappa_Fuel[]
- cv_Fuel[J|[kgK]]
- cp_Fuel[J|[kgK]]

PORTINJECTOR_TIMING

The block can now calculate a higher injection time to account for a lower fuel quantity if the pressure difference between the injector fuel supply and the intake manifold is too low.

The Map_mdot_Fuel_Gain_Red parameter has been added. Two new inports for fuel pressure difference calculation have been created: p_InMan[Pa] and p_Fuel[Pa].

The PORTINJECTOR_TIMING block can be used with multiple air intake engine setups, such as V-engines. The Map_Inj2Cyl and Const_max_num_PortInjector_PressureDrop parameters are used to assign the intake manifold, injection and cylinder.

New blocks

The following blocks are new:

- CNG_PRESSURE_REGULATOR
- CNG_SHUTOFF_VALVE
- CNG_HIGH_PRESSURE_LINE
- CNG_TANK
- CNG_RAIL

Test cycles

The Ftp_75 test cycle has been updated with an engine switch-off phase. After a total duration of 1369 seconds, the engine is switched off for 10 minutes. This is then followed by a warm engine restart.

Moreover, the new test cycles WLTC (Worldwide Harmonized Light Vehicle Test Procedure) with three classes (depending on the power-to-mass ratio) have been implemented. The new test cycles can be found in the test cycles folder of the new engine demos.

Migrating to ASM Gasoline EngineBlockset 3.2

RAIL_CONTROL_ CRANKBASED

The Const_disable_FMU_q_Inj, Const_enable_FMU_n_Engine, and Const_enable_FMU_q_Inj parameters have been removed, as these thresholds have no functional influence anymore. The change has no functional effect on the model behavior.

The Map_eta_DeltaAngle parameter has been renamed to Map_phi_FMU_energized. The change has no functional effect on the model behavior.

The Map_phi_FMU_FF parameter has been added in order to improve the control strategy by means of a feedforward table.

The Sw_phi_FMU_Update parameter has been added in order to improve the pulsed control mode with a defined pump angle update of the control variable within a pump cycle.

The Map_ValveDelay parameter has been added in order to account for delays in the valve actuation.

The inactive elements in the phi_FMU_energized output vector are now replaced by 999, in parallel to the I/O behavior. A crank angle of 999 will be interpreted as invalid in the high pressure pump plant model. Consequently, the corresponding pump cycle will not deliver any fuel to the high pressure rail.

HPP_CRANKBASED

The block has been modified to also account for negative TDC offset definitions.

In order to take pump cycles without FMU actuation into account, the vector containing the FMU actuation information will now be reordered. Based on that, the pump block will consider only the actuation signals that belong to the current pump cycle.

PORTINJECTOR_TIMING

The new parameters and inputs will be set up so that the new functionality has no effect.

PORTINJECTOR

The new parameters and inputs will be set up so that the new functionality has no effect.

Related topics

Basics

- *Migrating ASM Models* (📖 *ASM User Guide*)

ASM Gasoline InCylinder Blockset

Where to go from here

Information in this section

New Features of ASM Gasoline InCylinder Blockset 2.1	74
Migrating to ASM Gasoline InCylinder Blockset 2.1	75

New Features of ASM Gasoline InCylinder Blockset 2.1

Model start script go.m file

The standard start script go.m file for the ASM demo models has been adapted in order to take the changes of the supported simulation platforms into account.

The revised go.m file now contains the following platform options:

- 'RTI': DS1005, DS1006, DS1007, MicroLabBox
- 'SCALEXIO': SCALEXIO hardware
- 'VEOS': VEOS (offline simulation platform)

For more information on the calling procedures of the go.m file, refer to the corresponding file header or *Model Initialization* (📖 *ASM User Guide*).

The start script go.m file also takes the changes to the VEOS target into account. With Release 2015-B, VEOS uses the DSRT target instead of the DSOffSim target (refer to *Migrating to VEOS 3.5* on page 250). The go.m file in the ASM demo projects has been adapted to trigger the correct compiler settings.

PORTINJECTOR_TIMING

The block can now calculate a higher injection time to account for a lower fuel quantity if the pressure difference between the injector fuel supply and the intake manifold is too low.

The Map_mdot_Fuel_Gain_Red parameter has been added. Two new inports for fuel pressure difference calculation have been created: p_InMan[Pa] and p_Fuel[Pa].

The PORTINJECTOR_TIMING block can be used with multiple air intake engine setups, such as V-engines. The Map_Inj2Cyl and Const_max_num_PortInjector_PressureDrop parameters are used to assign the intake manifold, injection and cylinder.

PORTINJECTOR	The block can now calculate a lower fuel quantity if the pressure difference between the fuel supply and the intake manifold is low. This is used in compressed natural gas engines. The Map_mdot_Fuel_Gain_Red parameter has been added. Two new inports for fuel pressure difference calculation have been created: p_InMan[Pa] and p_Fuel[Pa].
Test cycles	The Ftp_75 test cycle has been updated with an engine switch-off phase. After a total duration of 1369 seconds, the engine is switched off for 10 minutes. This is then followed by a warm engine restart. Moreover, the new test cycles WLTC (Worldwide Harmonized Light Vehicle Test Procedure) with three classes (depending on the power-to-mass ratio) have been implemented. The new test cycles can be found in the test cycles folder of the new engine demos.

Migrating to ASM Gasoline InCylinder Blockset 2.1

PORTINJECTOR_TIMING	The new parameters and inputs will be set up so that the new functionality has no effect.
PORTINJECTOR	The new parameters and inputs will be set up so that the new functionality has no effect.

ASM Optimizer

Where to go from here

Information in this section

New Features of ASM Optimizer 1.7	76
Migrating to ASM Optimizer Blockset 1.7	77

New Features of ASM Optimizer 1.7

Separation of data, execution, and postprocessing

The settings for data, execution, and postprocessing are separated more clearly now. Because of this, several inputs are now located on a different page. The axis definition is moved from the CalcEOPC script to a seperate settings file in the postprocessing.

The General Settings file of ModelDesk can be reused.

Improved update behavior

Update processes are automatically performed. For example, changes in the mapping automatically trigger a regeneration of the OPData structure.

Import of ModelDesk measurement data files

ModelDesk measurement data files (*.md) can be used as the source for mean value measurement and mapping import.

Improved restart behavior

The results of gradient optimization algorithms depend on the start conditions. Now, each operation point of each task is either loaded or optimized. This ensures that repeated execution of an optimization reproduces the same results.

Redesigned postprocessing

The postprocessing pages are redesigned similar to the optimization task handling. The Manage Postprocessing page is introduced to keep global postprocessing settings. For each optimization task, a separate Postprocessing Task page is created.

Post-optimization function

A function can be added after each task. These functions can be uesd to modify the OPData. The task results are also available for further plots, for example.

Migrating to ASM Optimizer Blockset 1.7

General

All settings are transferred to the new location. With the existing axis definitions of the EOPV calculation, a Settings file is created. For existing post commands, a Post Optimization Function is generated. Due to changes in the result structure, existing results are deleted if an optimization is started.

ASM Traffic Blockset

Where to go from here

Information in this section

New Features of ASM Traffic Blockset 3.3	78
Migrating to ASM Traffic Blockset 3.3	79

New Features of ASM Traffic Blockset 3.3

Model start script go.m file

The standard start script go.m file for the ASM demo models has been adapted in order to take the changes of the supported simulation platforms into account.

The revised go.m file now contains the following platform options:

- 'RTI': DS1005, DS1006, DS1007, MicroLabBox
- 'SCALEXIO': SCALEXIO hardware
- 'VEOS': VEOS (offline simulation platform)

For more information on the calling procedures of the go.m file, refer to the corresponding file header or *Model Initialization* (📖 *ASM User Guide*).

The start script go.m file also takes the changes to the VEOS target into account. With Release 2015-B, VEOS uses the DSRT target instead of the DSOffSim target (refer to *Migrating to VEOS 3.5* on page 250). The go.m file in the ASM demo projects has been adapted to trigger the correct compiler settings.

FUEL_CONSUMPTION

The FUEL_CONSUMPTION block has been added to the ENGINE subsystem to simulate the fuel consumption and the carbon dioxide emissions. The calculations are approximated, since the focus of the demo is the vehicle dynamics. The fuel consumption is calculated on the basis of a brake-specific fuel consumption map, which is a measurement of the fuel efficiency.

The block is implemented in the new demo only and will not be automatically added during the migration. To use the block, drag it from the library and connect the related ports.

Migrating to ASM Traffic Blockset 3.3

CUSTOM_SENSOR_ SCOPEZONE	The implementation has been changed. Demux blocks are used instead of Selector blocks.
CUSTOM_SENSOR_OUTPUT	The implementation has been changed. Demux blocks are used instead of Selector blocks.
TRAFFIC_SCHEDULER	The ASMSignalBus has been changed. For details, refer to *Traffic Scheduler* (📖 *ASM Traffic Reference*).
FELLOW_PARAMETERS	The width of the PosVec_mainPnt_Fellows[x;y;z][m] outport is now defined by a workspace variable.
SENSOR_POSITION	There are changes due to Simulink diagnostics.
COORDINATE_ TRANSFORMATION	There are changes due to Simulink diagnostics.
NEAREST POINT	There are changes due to Simulink diagnostics.
FELLOW_POSITION	There are changes due to Simulink diagnostics.
NEAREST SURFACE	There are changes due to Simulink diagnostics.
RADARSENSOR_3D	There are changes due to Simulink diagnostics.

ASM Trailer Blockset

Where to go from here

Information in this section

New Features of ASM Trailer Blockset 2.4	80
Migrating to ASM Trailer Blockset 2.4	81

New Features of ASM Trailer Blockset 2.4

Model start script go.m file

The standard start script go.m file for the ASM demo models has been adapted in order to take the changes of the supported simulation platforms into account.

The revised go.m file now contains the following platform options:

- 'RTI': DS1005, DS1006, DS1007, MicroLabBox
- 'SCALEXIO': SCALEXIO hardware
- 'VEOS': VEOS (offline simulation platform)

For more information on the calling procedures of the go.m file, refer to the corresponding file header or *Model Initialization* (📖 *ASM User Guide*).

The start script go.m file also takes the changes to the VEOS target into account. With Release 2015-B, VEOS uses the DSRT target instead of the DSOffSim target (refer to *Migrating to VEOS 3.5* on page 250). The go.m file in the ASM demo projects has been adapted to trigger the correct compiler settings.

SUSCOMP_RIGID_SYM_xxx

A new parameter has been added to the block: Map_Angle_Gamma_Axle_F_y. It takes the axle rotation due to lateral force into account.

Migrating to ASM Trailer Blockset 2.4

SUSCOMP_RIGID_SYM_xxx	A new parameter has been added: Map_Angle_Gamma_Axle_F_y.
SUSKIN_RIGID_SYM_xxx	There has been a bug fix for the Taylor series arcsin calculation (wrong sign at the third Taylor series).

ASM Truck Blockset

Where to go from here

Information in this section

New Features of ASM Truck Blockset 2.3	82
Migrating to ASM Truck Blockset 2.3	83

New Features of ASM Truck Blockset 2.3

Model start script go.m file

The standard start script go.m file for the ASM demo models has been adapted in order to take the changes of the supported simulation platforms into account.

The revised go.m file now contains the following platform options:

- 'RTI': DS1005, DS1006, DS1007, MicroLabBox
- 'SCALEXIO': SCALEXIO hardware
- 'VEOS': VEOS (offline simulation platform)

For more information on the calling procedures of the go.m file, refer to the corresponding file header or *Model Initialization* (📖 *ASM User Guide*).

The start script go.m file also takes the changes to the VEOS target into account. With Release 2015-B, VEOS uses the DSRT target instead of the DSOffSim target (refer to *Migrating to VEOS 3.5* on page 250). The go.m file in the ASM demo projects has been adapted to trigger the correct compiler settings.

SUSCOMP_RIGID_SYM_xxx

A new parameter has been added to the block: Map_Angle_Gamma_Axle_F_y. It takes the axle rotation due to lateral force into account.

Drivetrain

The Drivetrain model has been changed to a 6x6 configuration with one front-driven axis and two rear-driven axes. In the new demo, the rigid drivetrain approach has been used. Two central, one front, and two rear differentials build the new drivetrain configuration. For this purpose, the multi-instance feature of the ASM blocks has been used.

The new configuration will only be available in the new demo. However, the old flexible drivetrain configuration from the vehicle dynamics demo model can still be used.

Migrating to ASM Truck Blockset 2.3

SUSCOMP_RIGID_SYM_xxx	A new parameter has been added: Map_Angle_Gamma_Axle_F_y.
SUSKIN_RIGID_SYM_xxx	There has been a bug fix for the Taylor series arcsin calculation (wrong sign at the third Taylor series).

ASM Turbocharger Blockset

Where to go from here

Information in this section

New Features of ASM Turbocharger Blockset 3.1.1	84
Migrating to ASM Turbocharger Blockset 3.1.1	85

New Features of ASM Turbocharger Blockset 3.1.1

POSTTURBHPMAN

The block has new outports for mass fraction consistency when two turbochargers are used:

- Xsi_Fuel_PostTurbHPMan[0_1]
- Xsi_Air_PostTurbHPMan[0_1]
- Xsi_Exh_PostTurbHPMan[0_1]

TURBINE

The Sw_TurbineType inport has been removed. The Sw_Ctrl_VTG_On parameter has been added to set up whether the turbine is controlled via variable turbine geometry (VTG) actuation. The inport for the VTG actuation signal now awaits a position signal.

The power balance has been fixed for backflow through the turbine.

TURBINE_HP

The Sw_TurbineType inport has been removed. The Sw_Ctrl_VTG_On parameter has been added to set up whether the high-pressure turbine is controlled via variable turbine geometry (VTG) actuation. The inport for the VTG actuation signal now awaits a position signal.

The power balance has been fixed for backflow through the turbine.

TURBINE_SAEJ922

The block is now also available for ASM Gasoline Engine. The functions have been modified for higher compatibility with different turbine data.

The Sw_TurbineType inport has been removed. The Sw_Ctrl_VTG_On parameter has been added to set up whether the high-pressure turbine is controlled via variable turbine geometry (VTG) actuation. The inport for the VTG actuation signal now awaits a position signal.

The Sw_mdot_Conv, Const_ConvScaling_mdot, Sw_omega_TC_Conv, and Const_ConvScaling_omega parameters were added to customize the conversion scaling.

The Map_HeatLoss_Turb_Rel parameter is new to take the heat flux at the turbine casing into account.

WASTEGATE_VALVE

The Sw_TurbineType inport was removed.

The Sw_Ctrl_WGate_On parameter has been added to set up whether the wastegate is controlled or not. The inport for the wastegate actuation signal now awaits a position signal.

WASTEGATE_VALVE_HP

The Sw_TurbineType inport was removed.

The Sw_Ctrl_WGate_On parameter has been added to set up whether the wastegate is controlled or not. The inport for the wastegate actuation signal now awaits a position signal.

TURBO_BASIC

The MAPS_TC block has been renamed to TURBO_BASIC.

The Sw_Ctrl_TC inport has been removed and added as the Sw_Ctrl_TC parameter to decide whether to use the wastegate, VTG, or no actuation signal.

TURBO_BASIC_2STAGE

The MAPS_TC_2STAGE block has been renamed to TURBO_BASIC_2STAGE.

The Sw_Ctrl_TC inport has been removed and added as the Sw_Ctrl_TC parameter to decide if the system uses the incoming actuation signal.

Migrating to ASM Turbocharger Blockset 3.1.1

Former version

The following systems are moved to a former version during migration:

- POSTTURBHPMAN
- TURBINE
- TURBINE_HP
- TURBINE_SAEJ922
- WASTEGATE_VALVE
- WASTEGATE_VALVE_HP

- TURBO_BASIC
- TURBO_BASIC_2STAGE
- SWITCHES_TURBO_2_0

ASM Vehicle Dynamics Blockset

Where to go from here

Information in this section

New Features of ASM Vehicle Dynamics Blockset 3.2	87
Migrating to ASM Vehicle Dynamics Blockset 3.2	88

New Features of ASM Vehicle Dynamics Blockset 3.2

Model start script go.m file

The standard start script go.m file for the ASM demo models has been adapted in order to take the changes of the supported simulation platforms into account.

The revised go.m file now contains the following platform options:

- 'RTI': DS1005, DS1006, DS1007, MicroLabBox
- 'SCALEXIO': SCALEXIO hardware
- 'VEOS': VEOS (offline simulation platform)

For more information on the calling procedures of the go.m file, refer to the corresponding file header or *Model Initialization* (📖 *ASM User Guide*).

The start script go.m file also takes the changes to the VEOS target into account. With Release 2015-B, VEOS uses the DSRT target instead of the DSOffSim target (refer to *Migrating to VEOS 3.5* on page 250). The go.m file in the ASM demo projects has been adapted to trigger the correct compiler settings.

SUSCOMP_RIGID_SYM_xxx

A new parameter has been added to the block: Map_Angle_Gamma_Axle_F_y. It takes the axle rotation due to lateral force into account.

FUEL_CONSUMPTION

The FUEL_CONSUMPTION block has been added to the ENGINE subsystem to simulate the fuel consumption and the carbon dioxide emissions. The calculations are approximated, since the focus of the demo is the vehicle dynamics. The fuel consumption is calculated on the basis of a brake-specific fuel consumption map, which is a measurement of the fuel efficiency.

The block is implemented in the new demo only and will not be automatically added during the migration. To use the block, drag it from the library and connect the related ports.

Migrating to ASM Vehicle Dynamics Blockset 3.2

VEHICLE_MOVEMENT_INFO_CAR	The "Not a number avoidance" at the calculation of the vehicle sideslip angle has been removed. The calculation now uses the atan2 function from Simulink. Therefore, the division by zero is already avoided.
SUSCOMP_RIGID_SYM_xxx	A new parameter has been added: Map_Angle_Gamma_Axle_F_y.
SUSKIN_RIGID_SYM_xxx	There has been a bug fix for the Taylor series arcsin calculation (wrong sign at the third Taylor series).
SOFT_ECU_POWERSTEERING	The Bus2Vector block has been added to avoid the "Bus signal treated as vector" warning.
STEERING_3DOF_VARIABLE_RATIO	The Bus2Vector block has been added to avoid the "Bus signal treated as vector" warning. The calculation of the exponential spring friction element at the steering rod and steering column has been corrected to avoid "Not a Number" generation.
STEERING	The Bus2Vector block has been added to avoid the "Bus signal treated as vector" warning.
STEERING_VARIABLE_RATIO	The Bus2Vector block has been added to avoid the "Bus signal treated as vector" warning.

ConfigurationDesk

Objective

ConfigurationDesk is a tool that can be applied in many different scenarios. You can use it to implement real-time applications and configure RapidPro hardware.

ConfigurationDesk – Implementation

Where to go from here

Information in this section

New Features of ConfigurationDesk 5.4 (Implementation Version)	90
Migrating to ConfigurationDesk 5.4	95

New Features of ConfigurationDesk 5.4 (Implementation Version)

Support of the enhanced trace file generation

ConfigurationDesk supports the Simulink Coder features concerning parameter handling introduced with MATLAB R2014a.

For more details, refer to *Changes to TRC File Generation* on page 35

Support of precompiled SIC files

ConfigurationDesk lets you add Simulink implementation container files (SIC files) to a ConfigurationDesk application. An SIC file is a container file containing the behavior model code. ConfigurationDesk now provides a method for you to precompile an SIC file. The following variants are possible:

- Precompiled SIC files without readable source files

  ConfigurationDesk lets you convert SIC files with source files into SIC files without source files but with a SCALEXIO-compatible library file, which might be desirable for IP protection.

- Precompiled SIC files that contain the original source files

  ConfigurationDesk lets you create precompiled SIC files that still contain the original source files. This can be useful if you want to use them in VEOS Player.

The build process is faster when you use such a precompiled SIC file. You can add the converted SIC files to your ConfigurationDesk application. For details, refer to *Creating Precompiled SIC Files* (📖 *ConfigurationDesk Real-Time Implementation Guide*).

Support of protected Simulink models

ConfigurationDesk lets you build real-time applications that contain Simulink behavior models and/or Simulink implementation containers that contain protected referenced models. Refer to *Features of the Model Interface Package for Simulink 3.1* on page 147.

New features of the FMU support

Support of several FMUs with identical data port names
ConfigurationDesk now identifies each port of an FMU via the FMU's model name and the name of the port, including its hierarchical structure as defined in the FMU's model description file. Thus, ports in different FMUs have different identities, even if they have the same names. This means that:

- You can add different FMUs with identical port names to your ConfigurationDesk application without creating a Model port block: Duplicate ID conflict.
- If you replace an FMU with an FMU with the same port names, but with a different model name, all ports that are provided by the original FMU and are used in the ConfigurationDesk application become unresolved.

FMU import: Support of enumerations ConfigurationDesk now supports the enumeration data type in FMUs. This means that:

- FMU inputs and outputs with the enumeration data type are available as data port in ConfigurationDesk.
- FMU variables with the enumeration data type are available in the TRC file.

For details, refer to *Integrating Functional Mock-up Units in ConfigurationDesk* (📖 *ConfigurationDesk Real-Time Implementation Guide*).

New features of the V-ECU support

Supported V-ECU Implementation container versions The following table shows the tool versions that export V-ECU implementation containers, and the related container versions:

V-ECU Implementations Created With Products of...	V-ECU Implementation Version
dSPACE Release 2013-B and earlier: ■ SystemDesk 3.x ■ TargetLink 3.5	1.0
dSPACE Release 2014-A: ■ SystemDesk 4.2	2.0
dSPACE Release 2014-B: ■ SystemDesk 4.3 ■ TargetLink 4.0	2.1

V-ECU Implementations Created With Products of...	V-ECU Implementation Version
dSPACE Release 2015-A: ■ SystemDesk 4.4	2.2
dSPACE Release 2015-B: ■ SystemDesk 4.5 ■ TargetLink 4.1	2.3

Support of several V-ECU implementations with identical port names ConfigurationDesk now identifies each V-ECU implementation port via the V-ECU implementation name, the port name, and the path of the port within the V-ECU implementation. Thus, ports in different V-ECU implementations have different identities, even if they have the same name and the same path within the V-ECU implementation. This means that:

- You can add different V-ECU implementations with identical port names to your ConfigurationDesk application without creating a Model port block: Duplicate ID conflict.
- If you replace a V-ECU implementation with a V-ECU implementation that has the same port names but a different V-ECU implementation name, all ports that are provided by the original V-ECU implementation and are used in the ConfigurationDesk application become unresolved.

Data type transformation when extending the signal chain

Data type transformation for digital function ports For the global Extend signal chain options on the Configuration page of the ConfigurationDesk Options dialog, the following applies:

If you set the Model port data type setting to Inherited, a model port with the Boolean data type is created for digital function ports that have an integer data type with the value range [0|1] during signal chain extension, for example, digital signals.

User-specific saturation value range When you use the Extend signal chain command, the following applies if you specified your own saturation values:

- The specified value range is written to the Unit property of a model port generated with the Extend Signal Chain command.
- The specified value range is written to the value range of the function block in the TRC file generated during the build process.

Saturation rules for Extend signal chain ConfigurationDesk lets you specify user saturation values for function ports. If you use System min/max values, a data type transformation between the model port and the function port is performed automatically if necessary. If you specify your own User min/max values for function ports and then use the Extend Signal Chain - Create Suitable Model Port Block command, specific saturation rules apply to the value ranges of function inports and function outports.

Refer to *Specifying Global Options for Extending the Signal Chain* (🕮 *ConfigurationDesk Real-Time Implementation Guide*)

Enhanced function block types

Current In, Voltage In The Current In and Voltage In function blocks provide the following new features:

- Capturing of angle positions. The angle position values are captured synchronously to the measured current/voltage values.
- Enabling or disabling the capturing of time stamp and angle position data to save computation time

For details, refer to *Configuring the Basic Functionality (Current In)* (🕮 *ConfigurationDesk I/O Function Implementation Guide*) or *Configuring the Basic Functionality (Voltage In)* (🕮 *ConfigurationDesk I/O Function Implementation Guide*).

Current Signal Capture, Voltage Signal Capture The Current Signal Capture and Voltage Signal Capture function blocks provide the following new features:

- Capturing of angle positions for all samples of all the captured sequences in a model step
- Capturing of start angle positions for each complete captured sequence
- Enabling or disabling the capturing of angle positions and start angle positions to save computation time

For details, refer to Configuring the Basic Functionality (Current Signal Capture) or *Configuring the Basic Functionality (Voltage Signal Capture)* (🕮 *ConfigurationDesk I/O Function Implementation Guide*).

CAN The CAN function block now supports the CAN FD mode for the DS2671 Bus Board. The CAN FD frames must either comply with the *Non-ISO CAN FD protocol* developed by Bosch or the *ISO CAN FD protocol* according to the upcoming ISO 11898-1:2015 standard (expected release in late 2015).

For details, refer to *CAN* (🕮 *ConfigurationDesk I/O Function Implementation Guide*)

New features of the Bus Manager

Communication matrix modification The Bus Manager now lets you specify user-defined settings for communication matrix elements. The defined settings apply only to the active ConfigurationDesk application. This allows you to work with modified communication matrix elements without changing the original communication matrix, for example. For details, refer to *Specifying User-Defined Settings for Communication Matrix Elements* (📖 *ConfigurationDesk Bus Manager Implementation Guide*).

LIN Schedule Table function ports for LIN masters The Bus Manager now provides a LIN Schedule Table function port for each LIN master that is assigend to a bus configuration. The LIN Schedule Table function port lets you specify an initial schedule table and enable to switch the active schedule table during run time. For details, refer to *Working With LIN Schedule Tables* (📖 *ConfigurationDesk Bus Manager Implementation Guide*).

New filter features

Preconfigured view sets After you installed ConfigurationDesk, the view sets 1 and 2 are preconfigured in the following way:

- Standard

  This view set provides the default screen arrangement of ConfigurationDesk, which is suitable for most use cases and screen resolutions.

- Small Screen Resolution

  This view set provides a screen arrangement that is most useful for signal chain configuration in the working area on displays with a relatively small screen resolution, such as a notebook display.

New features of the automation interface

ConfigurationDesk's automaton interface is enhanced and supports further features of ConfigurationDesk. For example, now you can access Bus Manager elements via XPath expressions. The XPath expressions must comply with XPath 1.0.

For detailed information on the enhancements and changes, refer to *Changes to the Automation Interface from Release 2015-A* (📖 *ConfigurationDesk Automating Tool Handling*)

New documentation features

Glossary with ConfigurationDesk terminology The glossary briefly explains the most important expressions and naming conventions used in the ConfigurationDesk documentation. Refer to *ConfigurationDesk Glossary* 📝.

Custom function block documentation You can find all the information on creating custom function blocks in a separate new document: 📖 *ConfigurationDesk Custom I/O Function Implementation Guide*.

The document contains:

- An introduction to custom function blocks and basics on defining custom function blocks and their elements
- Examples of implementing custom function blocks for serial and Ethernet communication
- Current limitations for custom function block implementation
- An XML element reference describing the structure and purpose of all elements that you can use in a custom function XML file

Migrating to ConfigurationDesk 5.4

Changes in TRC file generation

You have to note some modifications on TRC file generation in ConfigurationDesk. Refer to *Changes to TRC File Generation* on page 35.

Container Management

New Features of Container Management

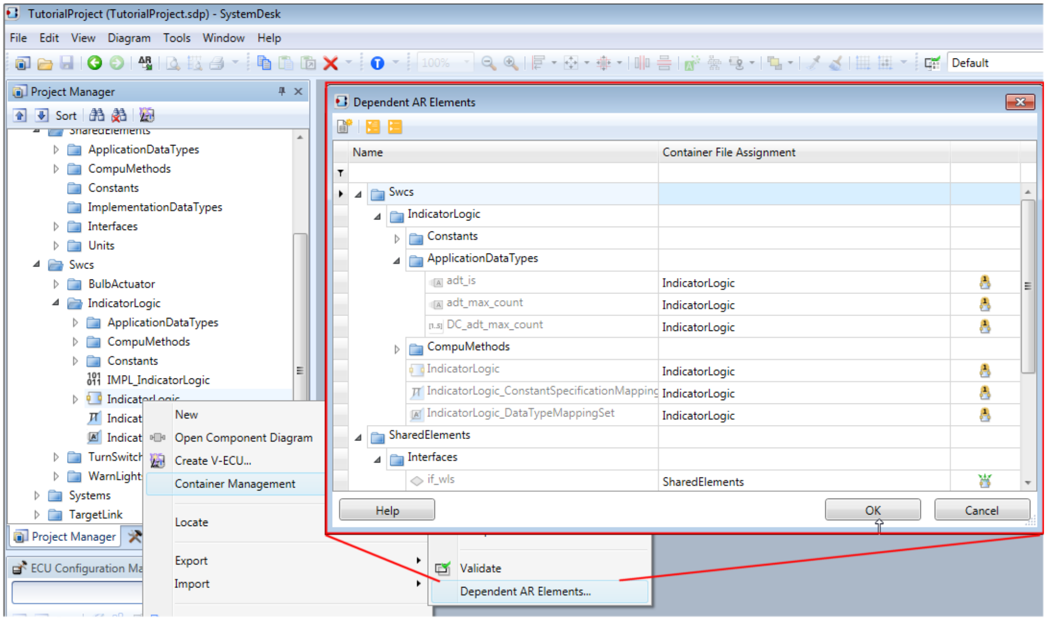
Improved assignment of elements to container files in SystemDesk 4.5

With this version of the Container Manager, assignment of AUTOSAR elements to container file assignments has been improved for SystemDesk 4.5. The file assigments are required for exchanging software components between SystemDesk and TargetLink.

With the Dependent AR elements command that is available from the Container Management context menu of atomic software component types in SystemDesk, you can assign elements easily.

The command opens the Dependent AR elements dialog which lets you manage file assignments for all the AUTOSAR elements that are referenced by the selected software component type. The command suggests an assignment for all the elements that you have not assigned yet. You can create new file assignments and change file assignments for each element or complete packages. The dialog assists you by displaying if an element is used by the selected software component type only ( ) or by several software component types ( ).

The following illustration shows an example from SystemDesk 4.5:



Name	Container File Assignment
Swcs	
IndicatorLogic	
Constants	
ApplicationDataTypes	
adt_is	IndicatorLogic
adt_max_count	IndicatorLogic
DC_adt_max_count	IndicatorLogic
CompuMethods	
IndicatorLogic	IndicatorLogic
IndicatorLogic_ConstantSpecificationMapping	IndicatorLogic
IndicatorLogic_DataTypeMappingSet	IndicatorLogic
SharedElements	
Interfaces	
if_wls	SharedElements

Further reading

For more details on assigning elements to container file assignments in SystemDesk 4.5, refer to *How to Assign AR Elements to Container Files* (📖 *Container Management Document*).

ControlDesk Next Generation

Where to go from here

Information in this section

New Features of ControlDesk Next Generation (ControlDesk 5.5)	100
Migrating to ControlDesk Next Generation (ControlDesk 5.5)	113

Information in other sections

🕮 *ControlDesk Next Generation Migration Guide* Explains migration from ControlDesk 3.x, CalDesk and prior versions of ControlDesk Next Generation to ControlDesk 5.5.
🕮 *ControlDesk Next Generation Migration of ControlDesk 3.x Automation* Explains migration from ControlDesk 3.x automation to ControlDesk Next Generation automation.

New Features of ControlDesk Next Generation (ControlDesk 5.5)

Where to go from here

Information in this section

New Features of Platform Management and Platforms/Devices (ControlDesk 5.5)	100
New Variable Management Features (ControlDesk 5.5)	103
New Visualization and Instrument Features (ControlDesk 5.5)	105
New Measurement and Recording Features (ControlDesk 5.5)	107
New Bus Navigator Features (ControlDesk 5.5)	108
New Data Set Management Features (ControlDesk 5.5)	109
New Signal Editor Features (ControlDesk 5.5)	109
New Electrical Error Simulation Features (ControlDesk 5.5)	110
New Automation Features (ControlDesk 5.5)	111

New Features of Platform Management and Platforms/Devices (ControlDesk 5.5)

Information in this topic

Support of the new DCI-CAN2 on page 101

CAN FD support on page 101

CAN Bus Monitoring device: Support for FIBEX and AUTOSAR system description files on page 101

FlexRay Bus Monitoring device: Support for AUTOSAR system description files on page 102

Further platforms that support automatic reconnect on page 102

Specifying the master for a simulation time group on page 102

Viewing application process information on page 102

Improvements to virtual validation scenarios on page 103

Support of the new DCI-CAN2

ControlDesk supports the new DCI-CAN2. The DCI-CAN2 lets you connect your host PC to a CAN FD or CAN network.

You can use the new DCI-CAN2 with the following ControlDesk devices:

- CAN Bus Monitoring
- CCP
- XCP on CAN

For details on the new DCI-CAN2, refer to the 🕮 *DCI-CAN2 Feature Reference*.

CAN FD support

The following devices now support CAN FD in connection with the new DCI-CAN2:

- CAN Bus Monitoring
- CCP
- XCP on CAN

> The current version of the dSPACE CAN API does not support CAN FD.

For details on the CAN FD-specific device settings, refer to *CAN Settings Properties* (🕮 *ControlDesk Next Generation Reference*).

CAN Bus Monitoring device: Support for FIBEX and AUTOSAR system description files

The CAN Bus Monitoring device now also supports the following variable description file formats in addition to DBC:

- FIBEX:
  - Version 4.1.0, 4.1.1
  - Version 3.1.0, 3.1.1
  - Version 3.0.0
- AUTOSAR system description files according to the AUTOSAR system template:
  - Version 4.2.1
  - Version 4.1.1 ... 4.1.3
  - Version 4.0.3
  - Version 3.2.1 ... 3.2.3
  - Version 3.1.4

> Keep the migration aspects in mind when you reuse an experiment that was originally created with ControlDesk 5.4 or earlier and contains a CAN Bus Monitoring device. Refer to *Migrating to ControlDesk Next Generation (ControlDesk 5.5)* on page 116.

FlexRay Bus Monitoring device: Support for AUTOSAR system description files

The FlexRay Bus Monitoring device now also supports AUTOSAR system description files according to the AUTOSAR system template:

- Version 4.2.1
- Version 4.1.1 ... 4.1.3
- Version 4.0.3
- Version 3.2.1 ... 3.2.3
- Version 3.1.4

Further platforms that support automatic reconnect

ControlDesk now also supports the automatic reconnect feature (→ *Automatic Reconnect* (📖 *ControlDesk Next Generation Basic Practices Guide*)) for the following platforms:

- DS1007 PPC Processor Board
- DS1202 MicroLabBox
- SCALEXIO
- XIL API MAPort

For details on the *automatic reconnect* feature, refer to *Reconnecting to Platform/Device Hardware Automatically* (📖 *ControlDesk Next Generation Basic Practices Guide*).

Specifying the master for a simulation time group

You can now specify the master for a simulation time group (→ *Simulation time group* (📖 *ControlDesk Next Generation Basic Practices Guide*)). This is useful if one of the members of the simulation time group has a low latency and high synchronization accuracy.

Refer to *Configure Simulation Time Group* (📖 *ControlDesk Next Generation Reference*).

Viewing application process information

You can now view information on the hardware that is assigned to the members of multiprocessor/multicore systems, and state information on the application process that is currently loaded to the hardware.

Refer to *Online Details Properties* (📖 *ControlDesk Next Generation Reference*).

Improvements to virtual validation scenarios

Reloading variable descriptions for multiple platforms/devices For virtual validation scenarios, ControlDesk offers a simplified way to reload the active application and variable descriptions of all platforms and devices that refer to a virtual validation scenario performed on VEOS or SCALEXIO.

Refer to *Reload System* (📖 *ControlDesk Next Generation Reference*).

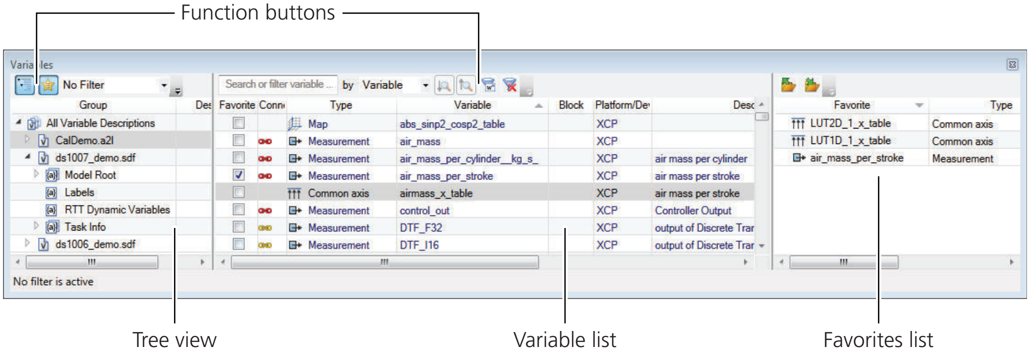
Automation: adding applications instead of variable descriptions ControlDesk's automation interface now allows you to add an application file instead of a variable description file to a VEOS or SCALEXIO platform in the experiment. This can be useful if you work with an application *without an environment model*.

Refer to *Adding applications instead of variable descriptions* on page 111.

New Variable Management Features (ControlDesk 5.5)

New Variable Browser

The Variable Browser has been renewed.



- All Variable Descriptions
  - CalDemo.a2l
  - ds1007_demo.sdf
    - Model Root
    - Labels
    - RTT Dynamic Variables
    - Task Info
  - ds1006_demo.sdf

Favorite	Conn	Type	Variable	Block	Platform/De	Desc
□		Map	abs_sinp2_cosp2_table		XCP	
□	⚭	Measurement	air_mass		XCP	
□	⚭	Measurement	air_mass_per_cylinder__kg_s_		XCP	air mass per cylinder
☑	⚭	Measurement	air_mass_per_stroke		XCP	air mass per stroke
□		Common axis	airmass_x_table		XCP	air mass per stroke
□	⚭	Measurement	control_out		XCP	Controller Output
□	⚭	Measurement	DTF_F32		XCP	output of Discrete Trar
□	⚭	Measurement	DTF_I16		XCP	output of Discrete Trar

Favorite	Type
LUT2D_1_x_table	Common axis
LUT1D_1_x_table	Common axis
air_mass_per_stroke	Measurement

For details, refer to *Basics of the Variable Browser* (📖 *ControlDesk Next Generation Basic Practices Guide*).

Support of structs and struct arrays

ControlDesk's Variable Browser now supports the display of structs and struct arrays.

	Icon	Description
Struct	▦	A structured list of variables that can have various data types
Struct array	▦	An array of homogeneous structs

If you drag a struct onto a Variable Array, all the variables it contains are connected, with exception to contained arrays and struct arrays. If you want to connect the contained variables to different instruments, you can customize the connection assignment. Refer to *Customizing the connection assignment of variables to instruments* on page 105.

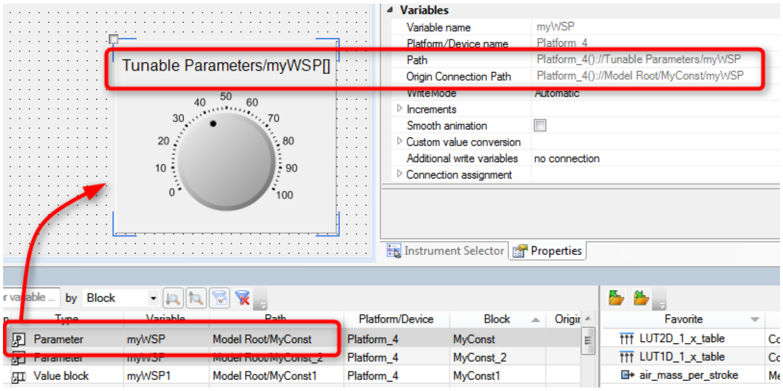
Display of the original connection path

When you place a variable on an instrument, ControlDesk now also displays the path of the variable that was originally connected to the instrument. This lets you identify the originally connected variable if it was replaced by the variable it references.

Referenced variables The icons of variables displayed under a subnode in the variable list can have an additional ↗ symbol. The arrow indicates that this variable is a reference to a variable residing in another node of the variable description. For example, variables in an A2L file always reside under the root node.

When you drag a reference to an instrument, the referenced variable is connected, not the reference itself.

Path and origin connection path The path of the connected variable and the *origin connection path* of the dragged variable are displayed in the instrument properties.



Variables

Variable name	myWSP
Platform/Device name	Platform_4
Path	Platform_4()://Tunable Parameters/myWSP
Origin Connection Path	Platform_4()://Model Root/MyConst/myWSP
WriteMode	Automatic
Increments	
Smooth animation	□
Custom value conversion	
Additional write variables	no connection
Connection assignment	

by Block

Type	Variable	Path	Platform/Device	Block	Origi
Parameter	myWSP	Model Root/MyConst	Platform_4	MyConst	
Parameter	myWSP	Model Root/MyConst_2	Platform_4	MyConst_2	
Value block	myWSP1	Model Root/MyConst1	Platform_4	MyConst1	

Favorite	
LUT2D_1_x_table	Cc
LUT1D_1_x_table	Cc
air_mass_per_stroke	Me

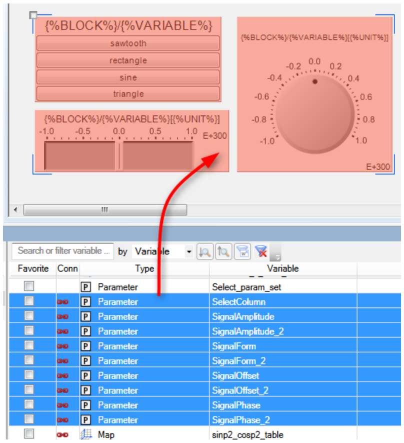
For details, refer to *Basics of Placing Variables on a Layout* (📖 *ControlDesk Next Generation Basic Practices Guide*).

New Visualization and Instrument Features (ControlDesk 5.5)

Customizing the connection assignment of variables to instruments

ControlDesk now lets you customize the connection assignment. This lets you create special variable-instrument connections, such as connecting a group of variables to multiple instruments in one step, for example.

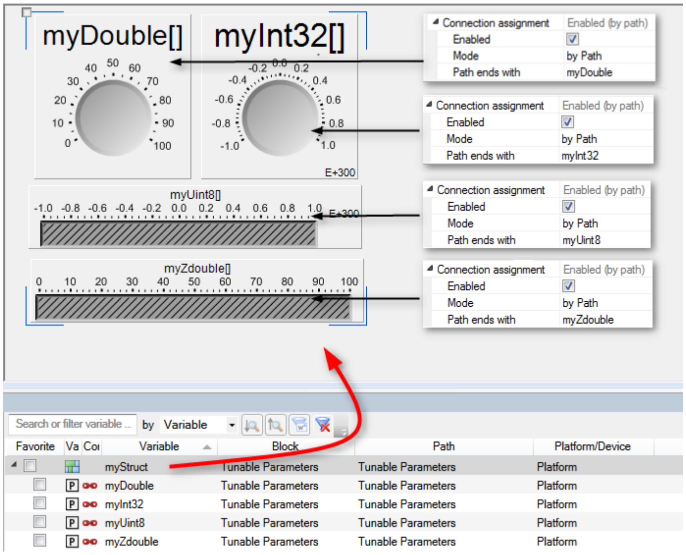
The illustration below shows the connection of several variables in different instruments in one go.



Favorite	Conn	Type	Variable
□		P Parameter	Select_param_set
□	⚯	P Parameter	SelectColumn
□	⚯	P Parameter	SignalAmplitude
□	⚯	P Parameter	SignalAmplitude_2
□	⚯	P Parameter	SignalForm
□	⚯	P Parameter	SignalForm_2
□	⚯	P Parameter	SignalOffset
□	⚯	P Parameter	SignalOffset_2
□	⚯	P Parameter	SignalPhase
□	⚯	P Parameter	SignalPhase_2
□	⚯	Map	sinp2_cosp2_table

Connection assignment is especially useful for connecting the individual members of struct variables. To let ControlDesk perform a connection assignment, you have to specify a text string that lets ControlDesk search for a matching variable in the struct.

The following illustration shows the connection of the members of a struct in different instruments in one go.



Connection assignment	Enabled (by path)
Enabled	☑
Mode	by Path
Path ends with	myDouble

Connection assignment	Enabled (by path)
Enabled	☑
Mode	by Path
Path ends with	myInt32

Connection assignment	Enabled (by path)
Enabled	☑
Mode	by Path
Path ends with	myUint8

Connection assignment	Enabled (by path)
Enabled	☑
Mode	by Path
Path ends with	myZdouble

Favorite	Va	Coı	Variable	Block	Path	Platform/Device
☐			myStruct	Tunable Parameters	Tunable Parameters	Platform
☐	P		myDouble	Tunable Parameters	Tunable Parameters	Platform
☐	P		myInt32	Tunable Parameters	Tunable Parameters	Platform
☐	P		myUint8	Tunable Parameters	Tunable Parameters	Platform
☐	P		myZdouble	Tunable Parameters	Tunable Parameters	Platform

For instructions, refer to *How to Customize the Connection Assignment of Variables to Instruments* (📖 *ControlDesk Next Generation Basic Practices Guide*).

Time Plotter and Index Plotter enhancements

Saving Time Plotter data as a new measurement (Time Plotter only) You can now save the data of a Time Plotter to a new measurement data file.

For instructions, refer to *How to Save Plotter or Time Plotter Data as a New Measurement* (📖 *ControlDesk Next Generation Basic Practices Guide*).

Legend enhancements The Time Plotter's legend now provides some enhancements:

- New legend columns such as x Delta
- Support of multiple selection
- Searching and filtering in the legend

Synchronization of the x-axis You can now synchronize the x-axis of the Time Plotter and the Index Plotter in continuous visualization mode. In this case, all synchronized Time Plotters share the same time range and all Index Plotters share the same index range. Zooming and moving actions then affect all synchronized Plotters in the same way.

For details, refer to *Zooming and Moving the Chart (Time Plotter)* (📖 *ControlDesk Next Generation Basic Practices Guide*) and *Zooming and Moving the Chart (Index Plotter)* (📖 *ControlDesk Next Generation Basic Practices Guide*).

New line styles The Index Plotter and the Time Plotter now also support the following line styles:

Line Styles	Description
Area	Data points are connected by a direct area line. The area between the base line and the area line is filled with the signal color(s).
Area Staircase	Data points are connected by a staircase area line. The area between the base line and the area line is filled with the signal color(s).
Staircase	Data points are connected by a staircase line.

For details, refer to *Axes and Signal Properties (Time Plotter/Index Plotter)* (📖 *ControlDesk Next Generation Reference*).

Switchable label orientation You can now display the y-axes labels horizontally or vertically.

For details, refer to *View Properties (Time Plotter/Index Plotter)* (📖 *ControlDesk Next Generation Reference*).

Multiswitch enhancement

You can now specify whether the currently selected element is marked by a dotted frame.

For details, refer to *Multiswitch Properties* (📖 *ControlDesk Next Generation Reference*).

New Measurement and Recording Features (ControlDesk 5.5)

ASAM MDF 4.x new default exchange format

The ASAM MDF 4.x file format now is ControlDesk's default format for measurement data files.

DSSIGCONV tool: Support of the ASAM MDF 4.x file format The DSSIGCONV tool, which lets you extract time sections or signals from a measurement data file, now also supports the ASAM MDF 4.x file format.

If you want to decrease the amount of data in a measurement data file, you can extract time sections or signals from it, or split the file into parts.

For instructions, refer to *How to Extract Data from a Measurement Data File* (🕮 *ControlDesk Next Generation Basic Practices Guide*).

Handling a large number of measurement rasters

In specific scenarios, such as bypassing or platform access via XIL API, you might have to handle a large number of measurement rasters. This might reduce system performance. By default, the number of measurement rasters that are displayed in the Measurement Configuration for a platform/device is limited to about 20 rasters.

You can change the number of displayed rasters according to your needs, which might enhance system performance.

For details, refer to *Handling a Large Number of Measurement Rasters* (🕮 *ControlDesk Next Generation Basic Practices Guide*).

Measurement Data API enhancements

Support of look-up tables (maps and curves) and common axes The Measurement Data API now supports look-up tables (maps and curves) and common axes.

For details, refer to the 🕮 *ControlDesk Next Generation Measurement Data API Reference*.

Specifying the variable type when creating new signals When you create a new signal using the Measurement Data API, you can now specify the signal's variable type.

Refer to *VariableTypeConstants* (🕮 *ControlDesk Next Generation Measurement Data API Reference*).

New Bus Navigator Features (ControlDesk 5.5)

Bus Instrument generation for Bus Manager configurations

ControlDesk's Bus Navigator now lets you generate Bus Instruments based on EXPSWCFG configuration data files created by ConfigurationDesk's Bus Manager for use with SCALEXIO.

Instrument generation is supported for the following communication protocols:

- CAN
- CAN FD
- LIN

CAN bus communication replay via MicroLabBox

You can now replay CAN bus communication in connection with MicroLabBox.

CAN FD support for SCALEXIO and the DCI-CAN2

The Bus Navigator now supports CAN FD (CAN with Flexible Data Rate) messages:

- On the SCALEXIO platform, if CAN bus communication is modeled via the RTI CAN MultiMessage Blockset in the behavior model
- For the DCI-CAN2

> The replay of CAN FD messages on the SCALEXIO platform and the DCI-CAN2 is not supported yet.

Refer to *Features of the Bus Navigator Specific for CAN* (📖 *ControlDesk Next Generation Advanced Practices Guide*).

CAN Bus Monitoring device: Support for FIBEX and AUTOSAR system description files

The Bus Navigator now also supports FIBEX and AUTOSAR system description files as variable description file formats in connection with the CAN Bus Monitoring device. Refer to *CAN Bus Monitoring device: Support for FIBEX and AUTOSAR system description files* on page 101.

FlexRay Bus Monitoring device: Support for AUTOSAR system description files

The Bus Navigator now also supports AUTOSAR system description files as variable description file formats in connection with the FlexRay Bus Monitoring device. Refer to *FlexRay Bus Monitoring device: Support for AUTOSAR system description files* on page 102.

New Data Set Management Features (ControlDesk 5.5)

CDFX new default exchange format

The Calibration Data File (CDFX) ASAM file format (ASAM Calibration Data Format V2.0) now is ControlDesk's default format for data sets.

New Signal Editor Features (ControlDesk 5.5)

Full support of the DS1007

ControlDesk's Signal Editor now fully supports stimulus generation on the DS1007 PPC Processor Board platform: i.e., one signal generator for a DS1007-based MP/MC application can stimulate variables of other application processes.

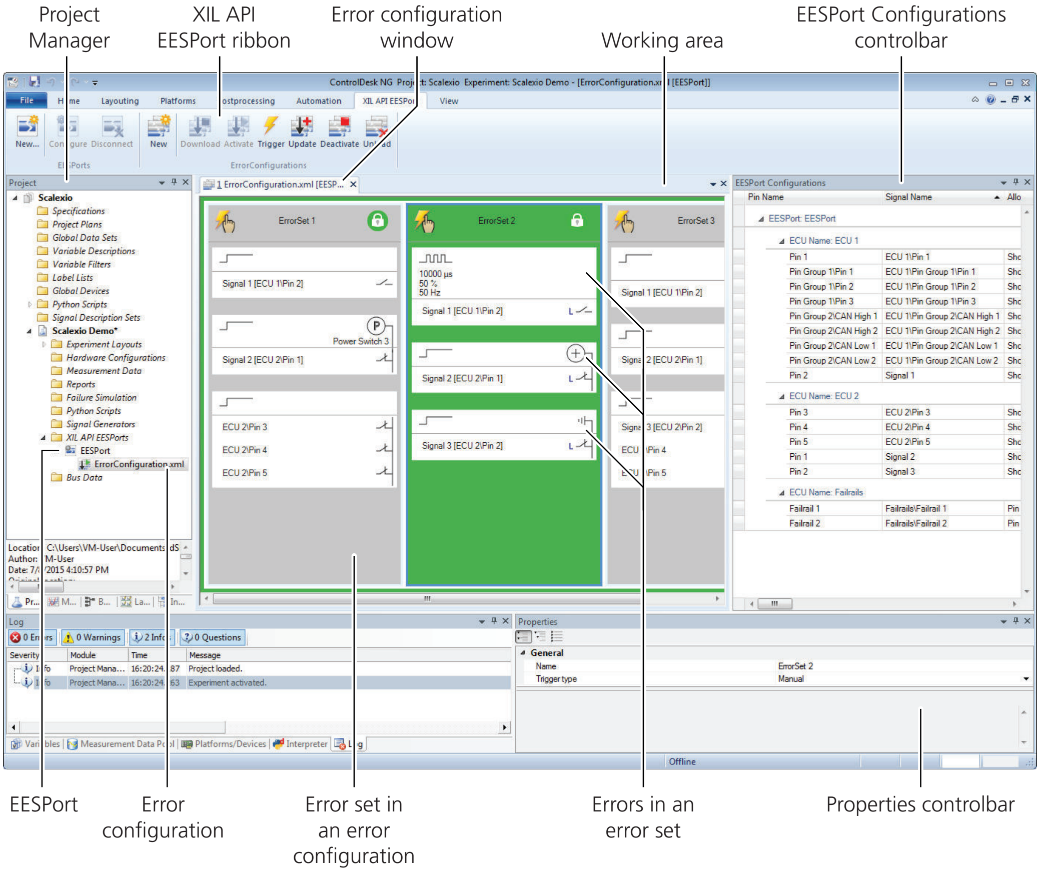
Support of DS1202 MicroLabBox

ControlDesk's Signal Editor now also supports stimulus generation on the DS1202 MicroLabBox platform.

New Electrical Error Simulation Features (ControlDesk 5.5)

New XIL API EESPort graphical user interface

To prepare electrical error simulation, ControlDesk now provides the XIL API EESPort graphical user interface.



Location: C:\Users\VM-User\Documents\dS
Author: M-User
Date: 7/ /2015 4:10:57 PM

Log

0 Errors | 0 Warnings | 2 Infos | 0 Questions

```
Severity  Module          Time          Message
Info      Project Mana... 16:20:24.87   Project loaded.
Info      Project Mana... 16:20:24.63   Experiment activated.
```

EESPort Configurations

Pin Name	Signal Name	Allo
EESPort: EESPort		
ECU Name: ECU 1		
Pin 1	ECU 1\Pin 1	Shc
Pin Group 1\Pin 1	ECU 1\Pin Group 1\Pin 1	Shc
Pin Group 1\Pin 2	ECU 1\Pin Group 1\Pin 2	Shc
Pin Group 1\Pin 3	ECU 1\Pin Group 1\Pin 3	Shc
Pin Group 2\CAN High 1	ECU 1\Pin Group 2\CAN High 1	Shc
Pin Group 2\CAN High 2	ECU 1\Pin Group 2\CAN High 2	Shc
Pin Group 2\CAN Low 1	ECU 1\Pin Group 2\CAN Low 1	Shc
Pin Group 2\CAN Low 2	ECU 1\Pin Group 2\CAN Low 2	Shc
Pin 2	Signal 1	Shc
ECU Name: ECU 2		
Pin 3	ECU 2\Pin 3	Shc
Pin 4	ECU 2\Pin 4	Shc
Pin 5	ECU 2\Pin 5	Shc
Pin 1	Signal 2	Shc
Pin 2	Signal 3	Shc
ECU Name: Failrails		
Failrail 1	Failrails\Failrail 1	Pin
Failrail 2	Failrails\Failrail 2	Pin

Properties

General

Name	ErrorSet 2
Trigger type	Manual

For details and instructions, refer to *Simulating Electrical Errors via XIL API EESPort* (📖 *ControlDesk Next Generation Advanced Practices Guide*).

> ControlDesk's XIL API EESPort graphical user interface is the successor to ControlDesk's Failure Simulation Module, which is being delivered for the last time with dSPACE Release 2016-A.
>
> Keep in mind that electrical error configurations of ControlDesk’s Failure Simulation Module are not compatible with XIL API EESPort configurations.
>
> For migration information, refer to *Migrating to ControlDesk Next Generation (ControlDesk 5.5)* on page 116.

New Automation Features (ControlDesk 5.5)

Measuring and recording look-up tables (maps and curves)

ControlDesk's automation interface now lets you measure and record look-up tables (maps and curves).

Event when adding signals to or removing signals from the measurement signal list

ControlDesk's automation interface now supports the following events when adding signals to or removing signals from the measurement signal list:

- `MeasurementSignalAdded`
- `MeasurementSignalRemoving`

Refer to *MeasurementDataManagementEvents / IXaMeasurementDataManagementEvents <<Events>>* (📖 *ControlDesk Next Generation API Reference*).

Adding applications instead of variable descriptions

ControlDesk's automation interface now allows you to add an application file instead of a variable description file to a VEOS or SCALEXIO platform in the experiment. This can be useful if you work with an application *without an environment model*.

Refer to *VEOSPlatform / IPmVEOSPlatform <<Interface>>* (📖 *ControlDesk Next Generation API Reference*) and *SCALEXIOPlatform / IPmSCALEXIOPlatform <<Interface>>* (📖 *ControlDesk Next Generation API Reference*).

Getting folder locations via automation

You can extend the functionality of ControlDesk and customize ControlDesk's ribbon by using Python *extension scripts*. Depending on the location, an extension script is executed for a specific user or for all users.

For example, as of ControlDesk 5.5, you can get the `<DocumentsFolder>` location using the properties of the `ApplicationEnvironment / IAeApplicationEnvironment <<Interface>>` interface. Refer to *ApplicationEnvironment / IAeApplicationEnvironment <<Interface>>* (📖 *ControlDesk Next Generation API Reference*).

Migrating to ControlDesk Next Generation (ControlDesk 5.5)

Where to go from here

Information in this section

Discontinuations in ControlDesk	113
Migrating to ControlDesk Next Generation (ControlDesk 5.5)	116

Information in other sections

🕮 *ControlDesk Next Generation Migration Guide*

Explains migration from ControlDesk 3.x, CalDesk and prior versions of ControlDesk Next Generation to ControlDesk 5.5.

🕮 *ControlDesk Next Generation Migration of ControlDesk 3.x Automation*

Explains migration from ControlDesk 3.x automation to ControlDesk Next Generation automation.

Discontinuations in ControlDesk

Information in this topic

Discontinuations for ControlDesk as of dSPACE Release 2016-A on page 113

Discontinuations for ControlDesk as of dSPACE Release 2016-B on page 114

Discontinuations for ControlDesk as of dSPACE Release 2016-A

ControlDesk's ASAP3 interface ControlDesk's ASAM ASAP3-compatible interface is being delivered for the last time with ControlDesk 5.5 (dSPACE Release 2015-B).

To automate calibration and measurement tasks, you can alternatively use:

- ControlDesk's automation interface. Refer to *Automating ControlDesk* (🕮 *ControlDesk Next Generation Advanced Practices Guide*).
- ControlDesk's ASAM MCD-3-compatible interface. Refer to the 🕮 *ControlDesk Next Generation MCD-3 Automation Guide*.

CDF import/export The Calibration Data File (CDF) format used to import/export data sets is being supported for the last time with ControlDesk 5.5 (dSPACE Release 2015-B).

To exchange calibration data, use one of the other file formats supported by ControlDesk such as CDFX (ASAM Calibration Data File 2.0), DCM, or DSV. The CDFX format is ControlDesk's default exchange format for data sets.

Refer to *Exporting and Converting Data Sets* (📖 *ControlDesk Next Generation Basic Practices Guide*).

User-defined databases (UDDBs) User-defined databases (UDDBs) are being supported for the last time with ControlDesk 5.5 (dSPACE Release 2015-B).

As a consequence, you have to change the real-time model to manipulate the communication of a CAN controller on dSPACE real-time hardware.

LDF (format version 1.2 and earlier) LDF files (format version 1.2 and earlier) are being supported by the LIN Bus Monitoring device for the last time with ControlDesk 5.5 (dSPACE Release 2015-B).

MAT file (version 6) export ControlDesk 5.5 and earlier creates version 6 MAT files that can be loaded in MATLAB Versions 5 (R8) or later.

As of dSPACE Release 2016-A, ControlDesk creates version 7.3 MAT files that can be loaded in MATLAB Versions 7.3 (R2006b) or later.

Discontinuations for ControlDesk as of dSPACE Release 2016-B

ControlDesk Failure Simulation Module ControlDesk's Failure Simulation Module is being delivered for the last time with dSPACE Release 2016-A.

- To prepare electrical error simulation via the graphical user interface (GUI), use the ControlDesk XIL API EESPort GUI, which is introduced with ControlDesk 5.5 (dSPACE Release 2015-B).
- To prepare electrical error simulation via automation, use the dSPACE XIL API .NET implementation supporting the Electrical Error Simulation Port (EESPort).

For migration aspects, refer to *Migrating to ControlDesk Next Generation (ControlDesk 5.5)* on page 116.

Plotter The Plotter is being delivered for the last time with dSPACE Release 2016-A.

Use one of the following instruments instead:

- Index Plotter
- Time Plotter
- XY Plotter

For information on the differences between the different plotter types, refer to *Differences Between Plotter, Time Plotter, Index Plotter, and XY Plotter* (📖 *ControlDesk Next Generation Basic Practices Guide*).

Table Editor The Table Editor is being delivered for the last time with dSPACE Release 2016-A.

It will be replaced by an enhanced Table Editor.

MDF (format versions 2.0 and 3.0) export The export of MDF measurement data files (MDF file format versions 2.0 and 3.0) is being supported for the last time with dSPACE Release 2016-A.

> Support for *importing* MDF files (format versions 2.0 and 3.0) continues.

To export measurement data, use one of the other file formats supported by ControlDesk. Refer to *How to Configure the Storage Settings for Recording* (📖 *ControlDesk Next Generation Basic Practices Guide*).

Migrating ControlDesk 3.x experiments The migration of ControlDesk 3.x experiments for reuse in ControlDesk Next Generation is being supported for the last time with dSPACE Release 2016-A.

> To reuse a ControlDesk 3.x experiment in ControlDesk from dSPACE Release 2016-B or later:
>
> 1. Migrate the ControlDesk 3.x experiment using ControlDesk from dSPACE Release 2016-A or earlier. Refer to *Migrating from ControlDesk 3.x to ControlDesk Next Generation* (📖 *ControlDesk Next Generation Migration Guide*).
> 2. Migrate the project from ControlDesk from dSPACE Release 2016-A or earlier to ControlDesk from dSPACE Release 2016-B or later. Refer to *Migrating from Prior Versions of ControlDesk Next Generation* (📖 *ControlDesk Next Generation Migration Guide*).

Migrating CalDesk projects The migration of CalDesk projects for reuse in ControlDesk Next Generation is being supported for the last time with dSPACE Release 2016-A.

> To reuse a CalDesk project in ControlDesk from dSPACE Release 2016-B or later:
>
> 1. Migrate the CalDesk project using ControlDesk from dSPACE Release 2016-A or earlier. Refer to *Migrating from CalDesk to ControlDesk Next Generation* (📖 *ControlDesk Next Generation Migration Guide*).
> 2. Migrate the project from ControlDesk from dSPACE Release 2016-A or earlier to ControlDesk from dSPACE Release 2016-B or later. Refer to *Migrating from Prior Versions of ControlDesk Next Generation* (📖 *ControlDesk Next Generation Migration Guide*).

Migrating to ControlDesk Next Generation (ControlDesk 5.5)

To migrate from ControlDesk 5.4 to ControlDesk 5.5 and reuse existing experiments, you might have to carry out the following migration steps.

> To migrate to ControlDesk 5.5 from versions earlier than 5.4, you might also have to perform the migration steps of the intervening ControlDesk versions.

Information in this topic

Changed TRC file generation on page 117
Failure Simulation Module: Discontinuation and migration on page 117
CAN Bus Monitoring device: changed DBC import on page 118
Change to the default behavior when downloading an incompatible SCALEXIO application on page 119
Change to the Path property on page 119
Change to the evaluation of recording triggers with multiscaling variables on page 120
Changed handling of checked variables and label lists on page 121
Tool automation changes on page 121
    Changes to variable management automation interfaces on page 121
    Change to the VideoDisplayStyleConstants enumeration on page 121
    Change to the storage of recorded multidimensional arrays on page 121
Changes to the Measurement Data API on page 122
Migrating from prior ControlDesk Next Generation versions on page 122

Changed TRC file generation

You have to note some modifications on TRC file generation in connection with MATLAB R2015b.

Refer to *Migrating Changes in Software That Uses TRC Files* on page 42.

Failure Simulation Module: Discontinuation and migration

ControlDesk's Failure Simulation Module is being delivered for the last time with dSPACE Release 2016-A.

- To prepare electrical error simulation via the graphical user interface (GUI), use the ControlDesk XIL API EESPort GUI, which is introduced with ControlDesk 5.5 (dSPACE Release 2015-B).

  To use the ControlDesk XIL API EESPort GUI, the Failure Simulation Package is required, which is based on XIL API's EESPort. The implementation is based on dSPACE XIL API .NET.

  Keep in mind that electrical error configurations of ControlDesk's Failure Simulation Module are not compatible with XIL API EESPort configurations.

  For migration, you can use the `FailureSimulationExportTool` to export information from an existing ControlDesk failure simulation system (FSN) file to the following files:

  - A hardware-dependent port configuration (PORTCONFIG) file

    You can use the file to create a new EESPort. For instructions, refer to *How to Create a New EESPort* (📖 *ControlDesk Next Generation Advanced Practices Guide*).

- One error configuration XML file for each failure pattern

  You can use the files to create and configure electrical errors, refer to *How to Create and Configure an Electrical Error* (📖 *ControlDesk Next Generation Advanced Practices Guide*).

The `FailureSimulationExportTool` version to use depends on the installed version of ControlDesk and dSPACE XIL API .NET as shown in the following table:

Installed ControlDesk Version	Installed dSPACE XIL API .NET Version	Required FailureSimulationExportTool Version
5.3	2.0	2014-B
5.4	2015-A	2015-A
5.5	2015-B	2015-B

You can download the `FailureSimulationExportTool` including a ReadMe file containing user documentation from the *ControlDesk Next Generation Product Support Center* at http://www.dspace.com/cdngpsc.

- To prepare electrical error simulation via automation, use the dSPACE XIL API .NET implementation supporting the Electrical Error Simulation Port (EESPort).

CAN Bus Monitoring device: changed DBC import

As of version 5.5, ControlDesk's DBC file import in connection with the CAN Bus Monitoring device has been changed to support FIBEX and AUTOSAR system description files. As a consequence, paths to variables in a DBC file are different depending on whether you import the DBC file in ControlDesk 5.4 (or earlier) and ControlDesk 5.5 (or later).

When you reuse an experiment originally created with ControlDesk 5.4 or earlier, you can continue working with the device and layouts/instruments based on the originally imported DBC file as usual.

The following limitations apply:

- Replacing and reloading the originally imported DBC file is blocked.
- When you add a new version of the DBC file to the CAN Bus Monitoring device, ControlDesk activates this DBC variable description and tries to restore the original variable connections. Due to the changed DBC file import, however, the paths to the variables in the newly added DBC file are different, so ControlDesk cannot restore any variable connection even if you added the same DBC file.

Generate new layouts based on the variable paths of the newly added DBC file after you add a new version of the DBC file to the CAN Bus Monitoring device.

Change to the default behavior when downloading an incompatible SCALEXIO application

When you download a real-time application to a SCALEXIO system, ControlDesk detects incompatibilities such as differences between the SCALEXIO system I/O and the I/O as required by the real-time application.

ControlDesk's default behavior on detecting such incompatibilities has been changed:

- Up to and including ControlDesk 5.4, ControlDesk simulated the access to the diverging I/O channels by default.
- As of ControlDesk 5.5, ControlDesk activates the access to the diverging I/O channels by default.

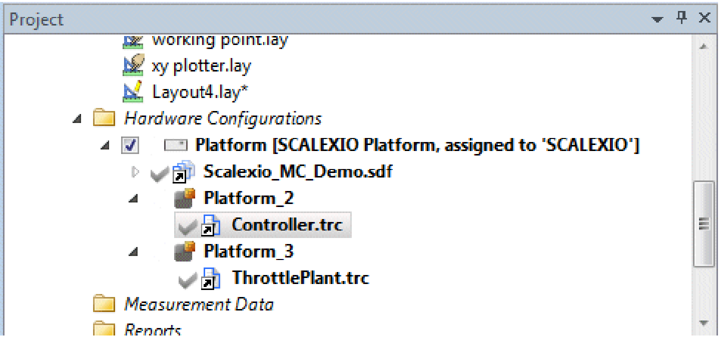
  To let ControlDesk simulate the access to the diverging I/O channels, you have to deselect the default behavior explicitly.

The default behavior was changed for downloading an application via ControlDesk's graphical user interface. It was *not* changed for downloading an application via ControlDesk's automation interface.

Change to the Path property

The Path property of visualized variables of multiprocessor and multicore real-time applications displayed in the Properties controlbar has changed in ControlDesk 5.5.

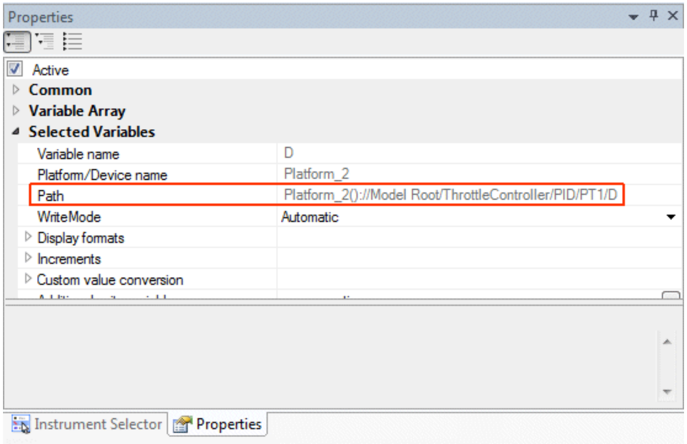
The following illustration shows a SCALEXIO platform with a variable description related to multicore application as an example:



- Path in ControlDesk 5.4 or earlier:

  `[PlatformName()://ModelRoot/...]`

  The following illustration shows the path to a variable from the multicore application as displayed in the Properties controlbar (ControlDesk 5.4 or earlier) as an example:



Properties

☑ Active

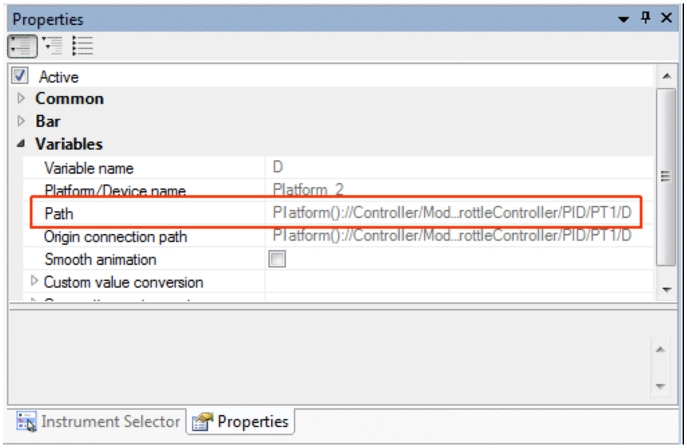
- Common
- Variable Array
- Selected Variables

Variable name	D
Platform/Device name	Platform_2
Path	Platform_2()://Model Root/ThrottleController/PID/PT1/D
WriteMode	Automatic
Display formats	
Increments	
Custom value conversion	

- Path in ControlDesk 5.5 or later:

  ```
  [MasterPlatform()://SubPlatform/ModelRoot/...]
  ```

  The illustration below shows the path to the same variable as displayed in the Properties controlbar (ControlDesk 5.5 or later) as an example:



Properties

☑ Active

- Common
- Bar
- Variables

Variable name	D
Platform/Device name	Platform_2
Path	Platform()://Controller/Mod..rottleController/PID/PT1/D
Origin connection path	Platform()://Controller/Mod..rottleController/PID/PT1/D
Smooth animation	□
Custom value conversion	

This change does not require any manual migration steps.

For reference information, refer to *Variables Properties* (📖 *ControlDesk Next Generation Reference*).

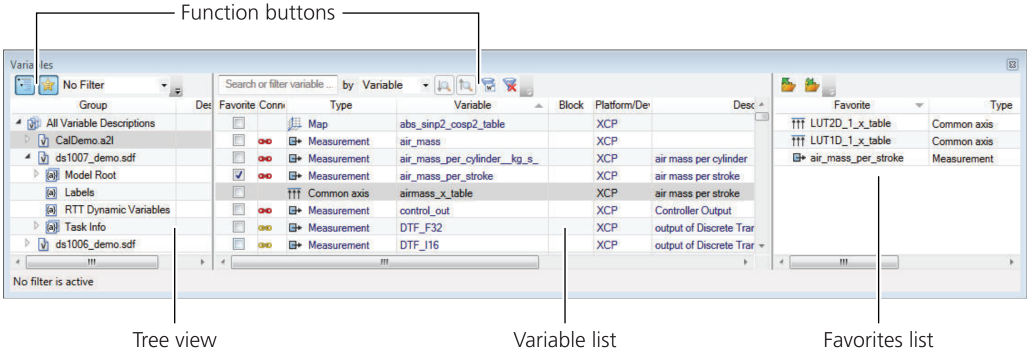
Change to the evaluation of recording triggers with multiscaling variables

As of ControlDesk 5.5, if a variable using a multiscaling table is used within a recording trigger rule, the *variable's source value* is used for trigger evaluation.

In ControlDesk 5.4, the *variable's converted value* was used for trigger evaluation.

Changed handling of checked variables and label lists

The Variable Browser has been renewed in ControlDesk 5.5. The Variable Browser's new Favorite list integrates the Checked variables list and the Label list:



Varia les

No Filter

Group	Des
All Variable Descriptions	
CalDemo.a2l	
ds1007_demo.sdf	
Model Root	
Labels	
RTT Dynamic Variables	
Task Info	
ds1006_demo.sdf	

No filter is active

Search or filter variable ... by Variable

Favorite	Conn	Type	Variable	Block	Platform/De	Desc
□		Map	abs_sinp2_cosp2_table		XCP	
□		Measurement	air_mass		XCP	
□		Measurement	air_mass_per_cylinder__kg_s_		XCP	air mass per cylinder
☑		Measurement	air_mass_per_stroke		XCP	air mass per stroke
□		Common axis	airmass_x_table		XCP	air mass per stroke
□		Measurement	control_out		XCP	Controller Output
□		Measurement	DTF_F32		XCP	output of Discrete Trar
□		Measurement	DTF_I16		XCP	output of Discrete Trar

Favorite	Type
LUT2D_1_x_table	Common axis
LUT1D_1_x_table	Common axis
air_mass_per_stroke	Measurement

For an overview of the Variable Browser, refer to *Basics of the Variable Browser* (📖 *ControlDesk Next Generation Basic Practices Guide*). For details on the Favorite list, refer to *Favorites List* (📖 *ControlDesk Next Generation Reference*).

Tool automation changes

Changes to variable management automation interfaces As of version 5.5, ControlDesk provides an enhanced Variable Browser.

As a consequence, the following properties of the VariablesManagement / IXaVariablesManagement <<Interface>> are no longer required and have been removed from the automation interface:

- `Application.VariablesManagement.OperationButtonsVisible`
- `Application.VariablesManagement.SubsystemFirstEnabled`

Change to the VideoDisplayStyleConstants enumeration In ControlDesk 5.5, the `Strech` value of the *VideoDisplayStyleConstants <<Enumeration>>* (📖 *ControlDesk Next Generation API Reference*) enumeration has been corrected to `Stretch`.

The numerical value of the `Stretch` enumeration value is unchanged.

Change to the storage of recorded multidimensional arrays Up to and including ControlDesk 5.4, recorded multidimensional arrays were not stored as arrays but as nested vectors.

As of ControlDesk 5.5, recorded multidimensional arrays are stored as arrays. As a consequence, you may have to adapt automation scripts accordingly.

> No script adaptation is required if you use Python to automate ControlDesk.

Changes to the Measurement Data API

- The data type of the `Size` attribute of the `FileDescription` interface has been changed from `Int` to `Float`.

  Refer to *Class Description (FileDescription)* (📖 *ControlDesk Next Generation Measurement Data API Reference*).
- The `BitOffset` and `NumberOfBits` attributes of the `Signal` interface can no longer be *set*. You can only *get* these attribute values.

  Refer to *Class Description (Signal)* (📖 *ControlDesk Next Generation Measurement Data API Reference*).

Migrating from prior ControlDesk Next Generation versions

To migrate from prior ControlDesk Next Generation versions and reuse existing experiments, you might have to carry out additional migration steps. For more details on the migration steps, refer to *Migrating to ControlDesk Next Generation* (📖 *ControlDesk Next Generation Migration Guide*).

DCI Configuration Tool

New Features of the DCI Configuration Tool 3.5

Improved A2L file adaptation

The DCI Configuration Tool comes with improvements related to the adaptation of an existing A2L file for use with a DCI-GSI2.

Refer to *A2L File Page* (📖 *DCI Configuration*).

dSPACE FlexRay Configuration Package

New Features of dSPACE FlexRay Configuration Package 3.6

FlexRay Configuration Tool

Support of AUTOSAR System Template 4.2.1 The FlexRay Configuration Tool now supports the AUTOSAR System Template based on AUTOSAR Release 4.2.1 for describing FlexRay networks.

Support of opaque byte order format The FlexRay Configuration Tool now supports signals with opaque byte order, which can be defined in an AUTOSAR system description file. Data of signal instances with the opaque byte order format is interpreted as dynamic uint8[n] arrays, where n depends on the signal length. The data is transmitted without endianness conversion, and the start bits must be byte-aligned.

For more details, refer to *Handling Configuration Projects* (📖 *FlexRay Configuration Tool Guide*).

Discontinuation of the FlexRay Replay Script Generator

The FlexRay Replay Script Generator is delivered for the last time with dSPACE Release 2015-B.

The FlexRay Replay Script Generator supports you in mapping logged FlexRay signals (stored in a MAT file) to TX signals of the replay model (contained in a TRC file). A Python script can be generated from the mapped signals. The script can then be replayed in Real-Time Testing via the Python interpreter.

As of RLS2016-A, it is still possible for you to integrate the Python interpreter into a FlexRay timetable task. This lets you still replay user-created Python scripts time-synchronously to the FlexRay bus.

dSPACE HIL API .NET

New Features of dSPACE HIL API .NET 2.0

Support of the enhanced trace file generation

dSPACE HIL API .NET supports the new features coming with the enhanced trace file generation, refer also to *Changes to TRC File Generation* on page 35.

Variable paths in your test scripts might be invalid. For further information, refer to *Migrating Changes in Software That Uses TRC Files* on page 42.

Stimulus support

The MAPort stimulus support for DS1007 has been enhanced. If you use a DS1007 as a multicore or multiprocessor system, you can now also stimulate variables that are contained in another subapplication that the signal generator is running in.

The MAPort stimulus now also supports MicroLabBox.

New default variable path format

The MAPort property `VariableNames` now returns the variable paths in ControlDesk Next Generation format. The ControlDesk 3.x format for variable paths is still supported.

The table below shows some examples:

Type	ControlDesk 3.x Format of Variable Paths	ControlDesk Next Generation Format of Variable Paths
Scalar	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1/Out1'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1/Out1'[1]
Vector	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1x3/Out1[1,3]'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1x3/Out1[2]'[2]

Type	ControlDesk 3.x Format of Variable Paths	ControlDesk Next Generation Format of Variable Paths
Matrix	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain2x3/Value[2,3]'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain2x3/Value[1][2]'
Structure[3]	–	'Platform()://[MP_SubApplicationName/]Model Root/Structures/Struct.SubStruct.DoubleArray[0]'

[1] 'Platform' is replaced by the relevant platform name, for example, 'ds1006'. The platform name has no effect on the model path itself. If you are using a multiprocessor model, the names of the subapplications are available in the model path.

[2] In ControlDesk Next Generation format, an array index starts with 0, in ControlDesk 3.x format, it starts with 1.

[3] Introduced with dSPACE Release 2015-B.

dSPACE Python Extensions

New Features of dSPACE Python Extensions 2.0

Support of the enhanced trace file generation

dSPACE HIL API Python and rtplib2 support the new features coming with the enhanced trace file generation, refer also to *Changes to TRC File Generation* on page 35.

Variable paths in your test scripts might be invalid. For further information, refer to *Migrating Changes in Software That Uses TRC Files* on page 42.

Stimulus support

The MAPort stimulus support for DS1007 has been enhanced. If you use a DS1007 as a multicore or multiprocessor system, you can now also stimulate variables that are contained in another subapplication that the signal generator is running in.

The MAPort stimulus now also supports MicroLabBox.

New default variable path format

The MAPort property `VariableNames` now returns the variable paths in ControlDesk Next Generation format. The ControlDesk 3.x format for variable paths is still supported.

The table below shows some examples:

Type	ControlDesk 3.x Format of Variable Paths	ControlDesk Next Generation Format of Variable Paths
Scalar	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1/Out1'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1/Out1'[1)
Vector	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1x3/Out1[1,3]'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1x3/Out1[2]'[2)

Type	ControlDesk 3.x Format of Variable Paths	ControlDesk Next Generation Format of Variable Paths
Matrix	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain2x3/Value[2,3]'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain2x3/Value[1][2]'
Structure[3]	–	'Platform()://[MP_SubApplicationName/]Model Root/Structures/Struct.SubStruct.DoubleArray[0]'

[1] 'Platform' is replaced by the relevant platform name, for example, 'ds1006'. The platform name has no effect on the model path itself. If you are using a multiprocessor model, the names of the subapplications are available in the model path.
[2] In ControlDesk Next Generation format, an array index starts with 0, in ControlDesk 3.x format, it starts with 1.
[3] Introduced with dSPACE Release 2015-B.

Test Automation Python modules

The `rtplib2` Python module comes with the following changed method:

- The `GetVarNames` method now returns the variable paths in ControlDesk Next Generation format. The ControlDesk 3.x format for variable paths is still supported. For examples, see above.

The `matlablib2` Python module comes with some new properties and enhanced methods.

The new properties are:

- `WatchdogMethod`

  To get or set the method for observing the MATLAB process.
- `Visible`

  To get or set the visibility of the MATLAB user interface.
- `ProcessArchitecture`

  To get the process architecture (32-bit or 64-bit) of the connected MATLAB instance.
- `ProcessID`

  To get the process ID of the connected MATLAB instance.
- `ExecutablePath`

  To get the path to the executable of the connected MATLAB instance.
- `Version`

  To get the version of the connected MATLAB instance.
- `IsMUMatlabOpen`

To get the flag that shows whether the connected MATLAB instance is opened for multiple use.

The new method is:

- `MaximizeCommandWindow`

  To maximize the MATLAB Command Window.

The enhanced methods are:

- MATLAB instance: `Open`

  With the new optional `MLInstallDir` parameter you can configure to start the MATLAB instance from the specified installation directory.

- MATLAB instance: `Close`

  With the new optional `DisconnectOnly` parameter you can configure whether the MATLAB instance is to be closed when you disconnect your automation access from the instance.

- MATFile instance: `Open`

  The `Mode` parameter now supports the `w7.3` option to create a MAT file in HDF5-based format that lets you store objects greater than 2 GB.

dSPACE XIL API

New Features of dSPACE XIL API 2015-B

Support of the enhanced trace file generation

dSPACE XIL API supports the new features coming with the enhanced trace file generation, refer also to *Changes to TRC File Generation* on page 35.

Variable paths in your test scripts might be invalid. For further information, refer to *Migrating Changes in Software That Uses TRC Files* on page 42.

Stimulus support

The MAPort stimulus support for DS1007 has been enhanced. If you use a DS1007 as a multicore or multiprocessor system, you can now also stimulate variables that are contained in another subapplication that the signal generator is running in.

The MAPort stimulus now also supports MicroLabBox.

New default variable path format

The MAPort property `VariableNames` now returns the variable paths in ControlDesk Next Generation format. The ControlDesk 3.x format for variable paths is still supported.

The table below shows some examples:

Type	ControlDesk 3.x Format of Variable Paths	ControlDesk Next Generation Format of Variable Paths
Scalar	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1/Out1'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1/Out1'[1]
Vector	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1x3/Out1[1,3]'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain1x3/Out1[2]'[2]

Type	ControlDesk 3.x Format of Variable Paths	ControlDesk Next Generation Format of Variable Paths
Matrix	'[MP_SubApplicationName/]Model Root/double/Subsystem/Gain2x3/Value[2,3]'	'Platform()://[MP_SubApplicationName/]Model Root/double/Subsystem/Gain2x3/Value[1][2]'
Structure[3]	–	'Platform()://[MP_SubApplicationName/]Model Root/Structures/Struct.SubStruct.DoubleArray[0]'

[1] 'Platform' is replaced by the relevant platform name, for example, 'ds1006'. The platform name has no effect on the model path itself. If you are using a multiprocessor model, the names of the subapplications are available in the model path.

[2] In ControlDesk Next Generation format, an array index starts with 0, in ControlDesk 3.x format, it starts with 1.

[3] Introduced with dSPACE Release 2015-B.

ECU Interface Manager

Where to go from here

Information in this section

New Features of ECU Interface Manager 1.7	135
Migrating to ECU Interface Manager 1.7	136

New Features of ECU Interface Manager 1.7

Support of built-in dSPACE Calibration and Bypassing Service configured for DPMEM POD

The ECU Interface Manager now also supports the dSPACE Calibration and Bypassing Service if it is already integrated in an ECU application and configured for use with a dSPACE DPMEM plug-on device (POD). With the ECU Interface Manager, you can add additional service function calls to such a built-in dSPACE Calibration and Bypassing Service, which allows you to prepare service-based bypassing using a DPMEM POD.

Refer to *Preparing Code Items for Bypassing* (📖 *ECU Interface Manager Guide*).

Specifying the behavior when execution control insertion fails

Inserting an execution control requires additional memory space.

If you try to insert an execution control *for all the instances of a code item*, and if there is insufficient memory space for the insertion of at least one execution control, the ECU Interface Manager opens a dialog for you to select one of the following behaviors:

- Skip the insertion of only the problematic instances
- Disable the execution of all instances *permanently*
- Skip the insertion for all instances

Refer to *Basics on Disabling Code Items* (🕮 *ECU Interface Manager Guide*).

Migrating to ECU Interface Manager 1.7

Migrating projects last saved with a former version of ECU Interface Manager

In ECU Interface Manager 1.7, you can reuse projects that were last saved with a former version of the ECU Interface Manager.

When you open such a project for the first time, you are asked whether to update it:

- When you start the update, you can continue working with the project with ECU Interface Manager 1.7.
- When you postpone the update, actions are blocked except for exporting the application. You can update the project later.
- When you save the project, you are asked whether to overwrite the old project file:
  - When you overwrite the old project, you can no longer use it with a former version of the ECU Interface Manager.
  - When you do not overwrite the old project, you have to specify another location and/or name for the project file. This lets you keep a version of the project that you can work with in the former version of ECU Interface Manager.

New software module description file schema

As of ECU Interface Manager 1.6, ECU suppliers can now use a generic schema to create a software module description file (→ *Software module description file* (🕮 *ECU Interface Manager Guide*)).

You can also import software module description files based on the dSPACE-specific schema, which was originally introduced with ECU Interface Manager 1.0.

- The dSPACE-specific schema is supported for downward compatibility reasons only. It will be replaced by the generic schema in the next dSPACE releases.
- Multicore support and further developments are not available with the dSPACE-specific schema.

  Use the *generic schema* instead.

For details on the generic schema, refer to *Generic Schema of Software Module Description Files* (📖 *ECU Interface Manager Reference*).

Firmware Manager

New Features of Firmware Manager 2.0

Enhanced platform support

The Firmware Manager supports SCALEXIO systems.

The support is restricted to the firmware archive installed with dSPACE Release 2015-B. If you want to update the firmware version on a SCALEXIO system to an earlier version, you have to use ControlDesk Next Generation or ConfigurationDesk from an earlier dSPACE release.

Usability improvements

The Firmware Management pane provides an additional pane that lists the available firmware archive versions with some additional information.

For more details, refer to *Firmware Manager Reference* (📖 *Firmware Manager Document*).

Changed Archive Format

The archive format for DS1007 and MicroLabBox changed with Firmware Archives 2.0 contained in dSPACE Release 2015-B. Earlier versions of these archives cannot be loaded with the Firmware Manager 2.0.

Model Compare

Product use prohibited in United States

You are not licensed to use Model Compare in the United States. You are not allowed to use or permit others to use this product in the United States or in any way that violates the laws of the United States.

Where to go from here

Information in this section

New Features of Model Compare 2.6	141
Migration to Model Compare 2.6	143

New Features of Model Compare 2.6

Dump file customization via hooks

Model Compare provides a hook mechanism that allows you to customize dump files. In detail, a hook defines which blocks and which block details are dumped, and in which format. If a subsystem contains other subsystems, you can omit the inner ones. You can write your own hooks. A demo hook is available.

Related documentation

- *Basics on XML Dump Files* (📖 *Model Compare Guide*)
- *How to Customize the XML Dump via Hooks* (📖 *Model Compare Guide*)

Quicker highlighting of leaf model elements

As a quicker alternative to the highlighting option in the context menu of model elements in the Model Navigator, you can now simply double-click a leaf model element to highlight it in Simulink.

Related documentation

- *How to Visualize Model Differences in MATLAB* (📖 *Model Compare Guide*)

Enhanced search

With this version of Model Compare you can search strings in property values. Furthermore, Model Compare provides advanced find options such as regular expressions, for example.

Related documentation

- *General Page* (📖 *Model Compare Reference*)
- *Find Bar* (📖 *Model Compare Reference*)

Saving models

You can configure that all the backup files created when a model is being saved are kept. Furthermore, you can now save a copy of the model by using a different name and path.

Related documentation

- *Merge Settings Page* (📖 *Model Compare Reference*)
- *Save Model Copy* (📖 *Model Compare Reference*)

Copying property names and values

By double-clicking a property in the Property Inspector, a separate window opens which lets you copy the property name as well as the property value.

Related documentation

- *Property Inspector* (📖 *Model Compare Reference*)

New quick guide and demo models

Model Compare now provides the new Model Compare Quick Guide which summarizes the most important features of Model Compare. This document is accessible via the Help menu. Furthermore, Model Compare is now delivered with demo models, which you can extract to your current working folder.

Related documentation

- *Quick Guide* (📖 *Model Compare Reference*)
- *Extract Demos* (📖 *Model Compare Reference*)

Migration to Model Compare 2.6

No adaptation necessary

You can migrate from Model Compare 2.5 to Model Compare 2.6 without any adaptations.

ModelDesk

New Features of ModelDesk 4.2

New supported platform

ModelDesk supports new simulation platforms:

- DS1007 PPC Processor Board
- MicroLabBox

Processing

Automation You can automize the essential steps of processing via tool automation:

- Creating and editing measurement types
- Creating and editing measurement data
- Accessing the processing properties of parameters

Automized mapping The mapping of the variables of raw data to the variables of a measurement type is improved. When you use the auto map function, all variables with the same name are mapped.

Plotting

Automation You can automize the plotting via tool automation:

- Managing the files for plotting the layouts, for example
- Adding and removing signals in the layouts
- Saving the values of the signals
- Starting and stopping plotting

Parameterizing

Automotive Simulation Models You can parameterize the Automotive Simulation Models in this release. For details on the Automotive Simulation Models, refer to *Automotive Simulation Models (ASM)* on page 53.

Model Interface Package for Simulink

Features of the Model Interface Package for Simulink 3.1

Support of the enhanced trace file generation

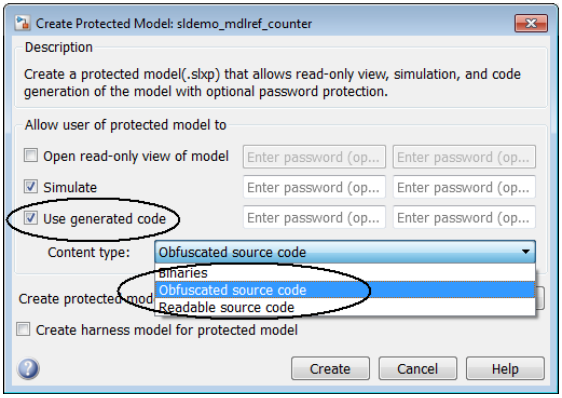
The Model Interface Package for Simulink supports the Simulink Coder features concerning parameter handling introduced with MATLAB R2014a for the generation of Simulink implementation container files.

For more details, refer to *Changes to TRC File Generation* on page 35

Support of protected Simulink models

The Model Interface Package for Simulink supports Simulink behavior models and/or Simulink implementation containers that contain protected referenced models. To create a protected referenced model, you have to select the Use generated code checkbox in the Simulink model's Create protected models dialog. The Model Interface Package for Simulink supports the following options from the Content type list (including password-protection):

- Obfuscated source code
- Readable source code



Create Protected Model: sldemo_mdlref_counter

Description

Create a protected model(.slxp) that allows read-only view, simulation, and code generation of the model with optional password protection.

Allow user of protected model to

☐ Open read-only view of model

☑ Simulate

☑ Use generated code

Content type: Obfuscated source code

Create protected mod

☐ Create harness model for protected model

For details, refer to the Simulink® Coder™ documentation.

New access to Copy/Paste with Identity commands

You can now select the Copy with Identity and Paste with Identity commands from the context menu of a model port block. Additionally, the Model Interface Package provides the following keyboard shortcuts for the commands:

- Copy with Identity: `Ctrl+Alt+C`
- Paste with Identity: `Ctrl+Alt+V`

MotionDesk

Where to go from here

Information in this section

New Features of MotionDesk 3.7	149
Migrating to MotionDesk 3.7	150

New Features of MotionDesk 3.7

New supported platform

MotionDesk supports a new simulation platform:

- MicroLabBox

Instruments

Instrument Panel MotionDesk provides the Instrument Panel, a tool that groups several instruments to one panel so that you can handle all the assigned instruments in common. This lets you minimize/maximize them or connect them to one or more observers, for example. You can use Instrument Panels to visualize a dash board in the scene.

Instruments and observers You can assign instruments to several observers by specifying an instrument property or via mouse operation in the Scene Navigator. So the same instrument can be displayed by different observers.

In the Scene Navigator, all the instruments are listed below the observer to which they are assigned.

Copy & paste You can copy & paste instruments to/from the Windows Clipboard to create duplicates of instruments.

Tool automation

MotionDesk tool automation is extended. Now you can handle observers via tool automation:

- Create observers
- Delete observers
- Create default observers
- Configure observers

Migrating to MotionDesk 3.7

Migrating from MotionDesk 2.2.1 and earlier

In MotionDesk 2.2.1 and earlier, MotionDesk uses 3-D objects in VRML format. To use the scenes and custom 3-D objects used in these MotionDesk versions, they must be migrated so that they can be used in MotionDesk 3.7. For details, refer to *Migrating from MotionDesk 2.2.1 and Lower* (📖 *MotionDesk Guide*).

Migrating from MotionDesk experiments in MDX file format

MotionDesk 3.7 cannot read old MotionDesk experiments in the MDX file format any longer. It is therefore not possible to migrate from a MotionDesk experiment with a version earlier than 2.2.

If you want to migrate such old experiments, you can migrate using MotionDesk 3.0 up to MotionDesk 3.6.

Real-Time Testing

New Features of Real-Time Testing 2.6

New supported platforms

Real-Time Testing supports MicroLabBox.

The following Real-Time Testing modules are not supported for MicroLabBox:

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

DS1007 multiprocessor systems

Now it is possible for RTT sequences running on a node to access TRC variables of remote nodes in DS1007 multiprocessor systems.

MAT file for data replay

Real-Time Testing is now independent of installed MATLAB versions because it can use MATLAB DLLs redistributed by dSPACE.

New class

The `rttlib.utilities` module has the `SequenceProperties` class which you can use to read properties of an RTT sequence (name, description, file name, priority) and to read/write the SequenceChannel property.

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	156

New Features of RTI/RTI-MP and RTLib

Support of the enhanced trace file generation

RTI and RTI-MP support the Simulink Coder features concerning parameter handling introduced with MATLAB R2014a.

For more details, refer to *Changes to TRC File Generation* on page 35

MicroLabBox

Enhanced software support The following features are now supported:

- Nonvolatile data handling (NVDATA)

  You can store data from your real-time application to the board's nonvolatile memory. This data is available again after a shutdown of your hardware. You can manage up to 64 KB data via RTLib, RTI, and also via the board's web interface.

  For more details, refer to *Nonvolatile Data Handling (NVDATA)* (📖 *MicroLabBox Features*).

- Enhanced RTI support

  The FPGA I/O Type 1 library provides the new Extras library containing the following new blocks:

  - LED_BLx

    To control the customizable LEDs.

  - Buzzer

    To control the board's buzzer.

  For more details, refer to *Basic Functions* (🕮 *MicroLabBox RTI Reference*).

- Enhanced setting in the DAC_CL1_BLx block

  You can configure the Termination mode for the specified channels. If the setting is enabled, you can set the termination value individually for each channel. If the setting is disabled, all the specified channels are set to high impedance at termination.

  For more details, refer to *DAC_CLASS1_BLx* (🕮 *MicroLabBox RTI Reference*).

- Enhanced support of electric motor control

  For more details, refer to *New Features of RTI Electric Motor Control Blockset 1.2* on page 165.

- Real-Time Testing support

  The Enable real-time testing option on the RTI simulation options page is set by default and is not changeable.

For more details, on the board's features, refer to 🕮 *MicroLabBox Features*.

DS1007

Enhanced software support The following features are now supported:

- Nonvolatile data handling (NVDATA)

  You can store data from your real-time application to the board's nonvolatile memory. This data is available again after a shutdown of your hardware. You can manage up to 64 KB data via RTLib, RTI, and also via the board's web interface.

  For more details, refer to *Nonvolatile Data Handling (NVDATA)* (🕮 *DS1007 Features*).

For more details, on the board's features, refer to 🕮 *DS1007 Features*.

MicroAutoBox

Pulse width measurement MicroAutoBox II variants with a DS1511 I/O Board or a DS1513 I/O Board support pulse width measurement (PW2D) via RTLib and RTI. You can either measure the high pulses or the low pulses of the connected signal in the range 3.3 µs … 53.6 s.

For more details, refer to *Pulse Width Measurement (PW2D)* (📖 *MicroAutoBox Features*).

Discontinuation of MicroAutoBox I/O boards The following MicroAutoBox II variants will be available only until end of 2015:

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

The software support continues at least up to end of 2018. We recommend that you use the successor variants of MicroAutoBox II with the I/O boards DS1511, DS1512, DS1513 and DS1514. The MicroAutoBox II variant 1401/1507 will still remain available.

For more details, on the board's features, refer to 📖 *MicroAutoBox Features*.

Unsupported new features of MATLAB R2015b

If you use the Evenly spaced breakpoints option for look-up tables, the parameters of the related block are not generated into the variable description. Accessing these parameters during run time, for example, via the Table Editor in ControlDesk Next Generation, is not possible.

Limitations when using MATLAB R2015b

Note the following limitation when you use RTI/RTI-MP with MATLAB R2015b:

- Absolute time support in triggered subsystems

  The scaling of the absolute time might be incorrect. The reason for this behavior is still unknown, but MathWorks is planning to publish a patch for Simulink versions R2015a and R2015b.

Discontinued batch file

The `build.bat` batch file was an alternative to the `rtimp_build` command for building a multiprocessor application. The `build.bat` batch file has been discontinued now.

Migration Aspects of RTI/RTI-MP and RTLib

Changes in TRC file generation

You have to note some modifications on TRC file generation in RTI and RTI-MP. Refer to *Changes to TRC File Generation* on page 35.

Modified features in MATLAB R2015b

The following changes were made in the Simulink Configuration Parameters dialog:

- The settings on the Hardware Implementation page have been changed. The production hardware has to be specified slightly differently.

  The related RTI settings were also adjusted. It is therefore recommended to reset the Simulink Preferences in all MATLAB installations, that were previously connected with an older RTI version.
- On the Optimization - Signals and Parameters page, the Inline parameters setting has been changed to the Default parameter behavior setting.

Real-time testing support always enabled for DS1007

The Enable real-time testing option on the RTI simulation options page is now always enabled for the DS1007 PPC Processor Board. This is the default behavior for the new dSPACE platforms DS1007 and MicroLabBox. Existing configurations are automatically updated by RTI and RTI-MP.

RTI Bypass Blockset

Where to go from here

Information in this section

New Features of the RTI Bypass Blockset 3.5	157
Migrating to RTI Bypass Blockset 3.5	158

New Features of the RTI Bypass Blockset 3.5

RTI Bypass Blockset

Support of XCP 1.3 The RTI Bypass Blockset supports A2L files containing XCP-specific `IF_DATA` entries based on the XCP 1.3 standard.

Support of CAN FD Besides the classic CAN protocol, the RTI Bypass Blockset now also supports CAN FD (CAN with Flexible Data Rate) as XCP transport layer. This means that communication between the ECU and the prototyping tool can be via CAN and/or CAN FD. Compared with the classic CAN protocol, the CAN FD protocol allows data rates higher than 1 MBit/s and payloads of up to 64 bytes per message.

Currently, there are two CAN FD protocols on the market, which are not compatible with each other: the *non-ISO CAN FD protocol* (representing the original CAN FD protocol from Bosch) and the *ISO CAN FD protocol* (representing the CAN FD protocol according to the upcoming ISO 11898-1:2015 standard, expected release in late 2015). The RTI Bypass Blockset supports both CAN FD protocols.

For service-based bypassing via XCP on CAN FD, the RTI Bypass Blockset uses the already established XCP on CAN bypass interface type. After importing a database file containing descriptions in the CAN FD format, you must enable CAN FD support and make some CAN FD-specific settings in the Setup block.

Refer to *Options Page (RTIBYPASS_SETUP_BLx for XCP on CAN)* (🕮 *RTI Bypass Blockset Reference*).

The blockset supports service-based bypassing via XCP on CAN FD only for dSPACE platforms equipped with DS4342 CAN FD Interface Modules.

RTI Bypass Blockset MATLAB API

Support of enhancements to RTI Bypass Blockset The RTI Bypass Blockset MATLAB API supports the enhancements to the RTI Bypass Blockset.

Refer to the 🕮 *RTI Bypass Blockset MATLAB API Reference*.

Migrating to RTI Bypass Blockset 3.5

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

The current release contains RTI Bypass Blockset 3.5, 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 open it with RTI Bypass Blockset 3.4, the old data dictionary file (with 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 via OK, or open the Read, Write, Upload or Download block dialog and click the Fill Variable Selector button on the Variables page.

  If you have a model that was saved with RTI Bypass Blockset 3.4 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 as soon as you update the A2L files in the Setup block or open the Read, Write, Upload or Download block and click the Fill Variable

Selector button on the Variables page. The data dictionary file created under RTI Bypass Blockset 3.5 (*.vdb) remains on disk.

To make the RTI Bypass Blockset able 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 3.4*

  If you have a Simulink model built with RTI Bypass Blockset 2.6 up to RTI Bypass Blockset 3.4 and open it with RTI Bypass Blockset 3.5, 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 3.4, 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 4.2	161
Migrating to RTI CAN MultiMessage Blockset 4.2	162

New Features of the RTI CAN MultiMessage Blockset 4.2

Enhancements in connection with CAN FD

The RTI CAN MultiMessage Blockset provides the following enhancements in connection with working with CAN FD messages:

CAN FD support for SCALEXIO systems The blockset now supports working with CAN FD messages for SCALEXIO systems with a DS2671 Bus Board.

Refer to *Basics on Working with CAN FD* (📖 *RTI CAN MultiMessage Blockset Reference*).

Support of ISO CAN FD protocol Besides the already supported original CAN FD protocol from Bosch, the blockset also supports the ISO CAN FD protocol (according to the upcoming ISO 11898-1:2015 standard, expected release in late 2015). You can select the CAN FD protocol to be used in the RTICANMM ControllerSetup block.

Refer to *Basics on Working with CAN FD* (📖 *RTI CAN MultiMessage Blockset Reference*).

Sample points for arbitration phase and data phase If working with CAN FD messages is enabled, you can specify the sample points for the arbitration phase and the data phase of CAN FD message transmission as a percentage of the CAN bit period. The sample points are used for synchronization purposes.

Refer to *Setup Page (RTICANMM ControllerSetup)* (📖 *RTI CAN MultiMessage Blockset Reference*).

Support of opaque byte order format

The RTI CAN MultiMessage Blockset now also supports signals with opaque byte order, which can be defined in an AUTOSAR system description file.

Migrating to RTI CAN MultiMessage Blockset 4.2

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 your 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', gcs)`.

  For more details 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 details 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 containing the phrase <<`argument of type "can_tpl_canChannel *" is incompatible with parameter of type "DsTCanCh"`>> will appear during the build process of your simulation model. This is due to a modified data type. These warnings can be ignored and will disappear after you regenerate the RTICANMM code by using the current blockset version.

Using existing checksum algorithms

Checksum algorithms originally developed for an application containing CAN messages cannot be reused for applications containing CAN FD messages, because CAN FD includes new message types and longer data fields. Existing checksum algorithms can still be used for applications that just contain classic CAN messages. For CAN FD applications, you must adapt the checksum algorithms.

RTI Electric Motor Control Blockset

New Features of RTI Electric Motor Control Blockset 1.2

New block

The RTI Electric Motor Control Blockset provides a new block:

- EMC_ENDAT_BLx to use an absolute encoder connected to an EnDat interface as an input sensor for motor control.

For more details, refer to 🕮 *RTI Electric Motor Control Blockset Reference*.

RTI FPGA Programming Blockset

Where to go from here

Information in this section

New Features of the RTI FPGA Programming Blockset 3.0	167
Migrating to RTI FPGA Programming Blockset 3.0	170

New Features of the RTI FPGA Programming Blockset 3.0

Xilinx® support

The RTI FPGA Programming Blockset now supports the following products and versions of the Xilinx design tools.

Xilinx Design Tools Version	Operating System	MATLAB Version
Vivado 2015.2 (64-bit version)	Windows 7 Business, Ultimate, and Enterprise SP1 (64-bit version)	64-bit versions of: ■ MATLAB R2014a ■ MATLAB R2014b ■ MATLAB R2015a

Supported dSPACE platforms

The following dSPACE platforms are supported by the RTI FPGA Programming Blockset 3.0:

- MicroAutoBox (MicroAutoBox II 1401/1511/1514 and MicroAutoBox II 1401/1513/1514)
- MicroLabBox
- Modular system (DS5203 (7K325) and DS5203 (7K410))
- SCALEXIO (DS2655)

The following hardware is not supported by Xilinx Vivado. The RTI FPGA Programming Blockset 3.0 therefore supports only the building of the processor interface for existing FPGA model INI files:

- DS5203 FPGA Board (SX95)
- DS5203 FPGA Board (LX50)
- MicroAutoBox II 1401/1511/1512
- MicroAutoBox II 1401/1512/1513

Only the RTI FPGA Programming Blockset up to version 2.9 supports Xilinx ISE and the FPGA modeling for the DS5203 (SX95) and DS5203 (LX50) boards, and the DS1512 I/O Board (MicroAutoBox II 1401/1511/1512 and MicroAutoBox II 1401/1512/1513). Due to the introduction of Vivado, Xilinx no longer supports the Xilinx System Generator for DSP in combination with the ISE Design Suite after MATLAB Release R2013b.

Enhancements to the DS2655 FPGA Base Board framework

The framework for the DS2655 FPGA Base Board provides the following enhancements.

Multiple clock periods You can use up to 10 individual clock periods for modeling specific parts of your FPGA design. The clock periods are in the range 1.6e-7 s ... 6.25e-10 s (6.25 MHz ... 1600 MHz).

The FPGA_SETUP_BL block provides the Subsystems Clocks page to configure the clock period for subsystems with an individual clock period.

Enhancements to the DS2655M1 Multi-I/O Module framework

The framework for the DS2655M1 Multi-I/O Module provides the following enhancement:

- The Digital In function of the FPGA_IO_READ_BLx block and the Digital InOut function of the FPGA_IO_Write_BLx block have a new port. The Threshold Ack port outputs a flag that indicates whether the I/O channel currently updates its threshold voltage or a new threshold configuration can be set.

Support of new DS2655M2 Digital I/O Module

The RTI FPGA Programming Blockset now provides the new *DS2655M2 I/O Module* framework for the DS2655M2 Digital I/O Module.

The DS2655M2 Digital I/O Module has 32 versatile digital I/O channels that can handle bit-wise data or serial communication. The main features of the framework are I/O functions that use one or more digital channels to implement a specific I/O functionality.

The following I/O functions can be used:

- Digital In

  Up to 32 digital input functions that provide bit-wise access.
- Digital Out

  Up to 32 digital output functions that provide bit-wise access.
- Digital Out-Z

  Up to 16 digital output functions that provide bit-wise access and a high-impedance output state (tristate).
- RS232 Rx

  Up to 8 serial functions that receive data values from RS232 networks.
- RS232 Tx

  Up to 8 serial functions that transmit data values to RS232 networks.
- RS485 Rx

  Up to 8 serial functions that receive data values from RS485 networks in simplex mode.
- RS485 Rx/Tx

  Up to 8 serial functions that exchange data values with RS485 networks in half-duplex mode.
- RS485 Tx

  Up to 8 serial functions that transmit data values to RS485 networks in simplex mode.

Enhancements to the FPGA1401Tp1 with Multi-I/O Frameworks

The frameworks for MicroAutoBox provide the following enhancements.

Multiple clock periods You can use up to 10 individual clock periods for modeling specific parts of your FPGA design. The clock periods are in the range 1.6e-7 s ... 6.25e-10 s (6.25 MHz ... 1600 MHz).

The FPGA_SETUP_BL block provides the Subsystems Clocks page to configure the clock period for subsystems with an individual clock period.

Related topics

Basics

- *Migrating to RTI FPGA Programming Blockset 3.0* on page 170

Migrating to RTI FPGA Programming Blockset 3.0

Objective

There are different ways to migrate an existing model, depending on the blockset version used.

Migrating from RTI FPGA Programming Blockset 1.0 to 3.0

Because the RTI FPGA Programming Blockset 1.0 (released with dSPACE Release 6.4) was not fully implemented, a model that you implemented with it must be migrated manually. You must replace each block of the RTI FPGA Programming Blockset with a new one to make the model compatible with the current dSPACE RTI environment for modeling, building and executing.

> The update function of the script interface does not support RTI FPGA Programming Blockset 1.0.

Migrating from RTI FPGA Programming Blockset 1.1 and higher to 3.0

If you have implemented your FPGA application using RTI FPGA Programming Blockset Version 1.1 and later, and want to use it with RTI FPGA Programming Blockset 3.0, you must update the FPGA framework. You can use the script interface for this, refer to *Updating the FPGA framework using the script interface* on page 170.

You also have to update the framework if you have updated from MATLAB R2008b or earlier to MATLAB R2011b or later.

Updating the FPGA framework using the script interface

> It is recommended to back up your model before starting migration.

The script interface provides the `FPGAFrameworkUpdate` method to update a framework. You can decide whether to set the block parameters to their initial values or leave them unchanged.

To update the FPGA framework without changing the values of the block parameters

```
rtifpga_scriptinterface('FPGAFrameworkUpdate',
 <SimulinkHandle>)
```

The script handles all the subsystems in the model/subsystem that is specified by the Simulink handle. The parameters of the blocks are unchanged after updating to the current framework version.

Example: The following script updates the FPGA framework for any FPGA subsystem in the processor model called *MyProcModel*. The specified values of the block parameters are not changed.

```
ProcModelHandle = get_param('MyProcModel','handle')
rtifpga_scriptinterface('FPGAFrameworkUpdate',
 ProcModelHandle)
```

To update the FPGA framework and reset the values of the block parameters to their initial values

```
rtifpga_scriptinterface('FPGAFrameworkUpdate',
 <SimulinkHandle>, 'ReInit')
```

The script handles all the subsystems in the model/subsystem that is specified by the Simulink handle. The parameters of the blocks are reset to their initial values after updating to the current framework version.

```
ProcModelHandle = get_param('MyProcModel','handle')
rtifpga_scriptinterface('FPGAFrameworkUpdate',
 ProcModelHandle,'ReInit')
```

ConfigurationDesk custom functions incompatible with dSPACE Release 2015-B

Relevant for SCALEXIO systems with a DS2655 FPGA Base Board and a DS2655M1 Multi-I/O Module

A custom function generated by using RTI FPGA Programming Blockset 2.5 from dSPACE Release 2013-A and the real-time applications (*.rta) containing the custom function are incompatible with dSPACE Release 2015-B. To produce a usable custom function you have to rebuild the FPGA model by using RTI FPGA Blockset 3.0 from dSPACE Release 2015-B.

RTI LIN MultiMessage Blockset

Where to go from here

Information in this section

New Features of the RTI LIN MultiMessage Blockset 2.5.1	173
Migrating to RTI LIN MultiMessage Blockset 2.5.1	173

New Features of the RTI LIN MultiMessage Blockset 2.5.1

Support of opaque byte order format

The RTI LIN MultiMessage Blockset now also supports signals with opaque byte order, which can be defined in an AUTOSAR system description file.

Migrating to RTI LIN MultiMessage Blockset 2.5.1

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 your 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', gcs)`.

For more details 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 details refer to *Limitations of RTI LIN MultiMessage Blockset* (📖 *RTI LIN MultiMessage Blockset Reference*).

SCALEXIO Firmware

New Features of the SCALEXIO Firmware 3.3

New supported hardware

The SCALEXIO firmware supports the new SCALEXIO hardware module: DS2655M2 Digital I/O Module.

SystemDesk

Where to go from here

Information in this section

New Features of SystemDesk 4.5	178
Migrating to SystemDesk 4.5	190

New Features of SystemDesk 4.5

Where to go from here

Information in this section

New General Features	178
Modeling Software Architectures	179
Working with Diagrams	179
Configuring ECUs	181
Simulating Systems	187
Automating SystemDesk	187
Variation Binding	188

New General Features

Objective

SystemDesk 4.5 has the following new general features.

AUTOSAR Releases supported by SystemDesk 4.5

Modeling support for the AUTOSAR 4.2.1 Release
SystemDesk 4.5 supports the modeling of software and system architectures according the AUTOSAR 4.2.1 Release.

Compatiblity to recent AUTOSAR Releases SystemDesk also supports AUTOSAR Releases 4.1.3, 4.1.2, 4.1.1, and 4.0.3 for exchanging AUTOSAR files.

Compatibility to the newest AUTOSAR Release SystemDesk 4.5 supports importing AUTOSAR files according to the newest AUTOSAR 4.2.2 Release. However, elements that are introduced with this AUTOSAR Release are not imported to SystemDesk.

You can export AUTOSAR files from SystemDesk acording to the AUTOSAR 4.2.2 Release.

Modeling Software Architectures

Improvements

With this version of SystemDesk, modeling software architectures now contains the following improvements:

- SystemDesk now lets you specify the naming of automatically created constants for initial values of communication specifications.

  For reference information on the naming, refer to *General page (preferences)* (📖 *SystemDesk 4.x Reference*).

- SystemDesk now provides a Start Page that lets you quickly access user documentation and additional SystemDesk material.

- You can now navigate in SystemDesk's controlbars such as the Project Manager using Navigate back ⬅ and Navigate forward ➡ commands from SystemDesk's menu.

  For reference information on the commands, refer to *Navigate Forward* (📖 *SystemDesk 4.x Reference*) and *Navigate Backward* (📖 *SystemDesk 4.x Reference*).

- The improved Project Manager search displays an element's AUTOSAR type and ECU configuration if available. It provides quick access to an element's Properties dialog.

  For reference information on SystemDesk's Project Manager and searching for elements, refer to *Project Manager* (📖 *SystemDesk 4.x Reference*).

- The improved import of AUTOSAR templates provides quick access to templates for creating standardized SystemDesk projects with standard AUTOSAR types and a standardized package structure.

  For reference information on importing templates, refer to *Import AUTOSAR Templates* (📖 *SystemDesk 4.x Reference*).

Working with Diagrams

Improving SystemDesk's diagrams for graphical modeling

With this version of SystemDesk, graphical modeling has been improved. Improvements apply to all of SystemDesk's diagrams: i.e., composition diagrams for modeling and connecting software components, component diagrams for modeling an atomic software component's ports and interfaces, and service connection diagrams for connecting basic software with aplication software.

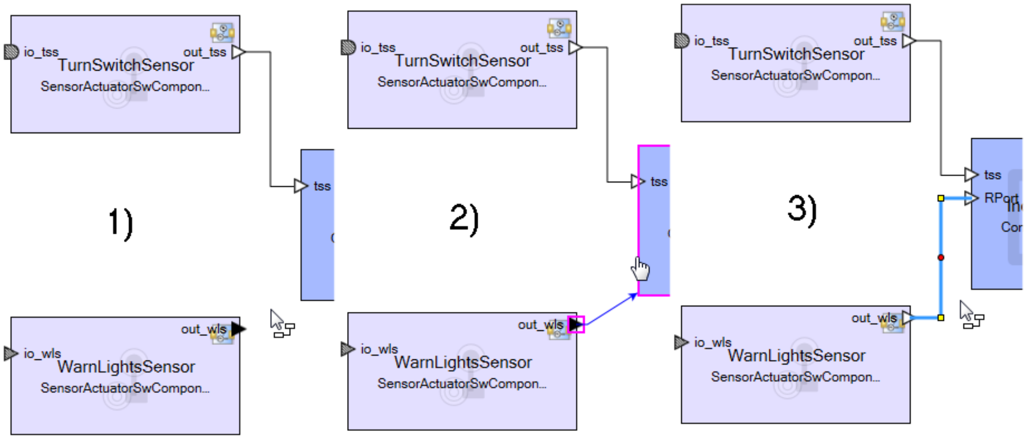
Edit and Connection modes

With this version, SystemDesk introduces the *Edit* and *Connection* modes for simplifying element handling and element connection in composition and service connection diagrams.

In the Edit mode that is active when you open a diagram, you can add and arrange elements.

When you switch to the Connection mode, which is indicated by a special cursor (↖), you can connect software components. To do so, you can connect software components via drag & drop. You can connect two ports, a port and an SWC or two SWCs. SystemDesk then adds ports and, where possible, interfaces to the connected SWCs.

The illustration below shows an example where you connect a port with an assigned interface to another software component. SystemDesk adds a port to the targeted software component and assigns the same interface to the added port as in the starting port.



The following additional improvements have been made:

Additional improvements comprise:

- Improved auto-layout of diagrams and connections
- Navigating between parent and child composition diagrams like in Simulink
- Searching for software components, ports, and interfaces in composition and service connection diagrams by name
- Displaying error messages for incompatible connections

- Selecting elements to add to a diagram more conveniently
- Improved configuration of element visibility in diagrams: e.g., the visibility of ports and connections in composition diagrams
- Specifying default appearances for software components, connections, ports, and interfaces in SystemDesk's preferences

Further reading

For more details on working with SystemDesk's diagrams, refer to *Working with Diagrams* (📖 *SystemDesk 4.x Guide*).

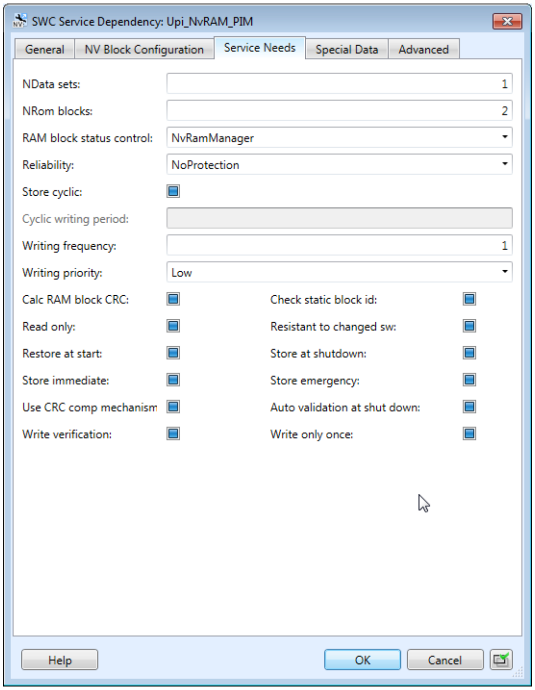
Configuring ECUs

NVRAM manager

SystemDesk now supports the NVRAM manager basic software component.

Modeling software architectures You can comfortably model NV block service needs of atomic SWCs for requesting read/write operations of the NVRAM manager. Additionally, you can model an NV block software component for managing the specified NV block service needs.

The following illustration shows an example of specifying an NV block service need.


SWC Service Dependency: Upi_NvRAM_PIM

NData sets:	1		
NRom blocks:	2		
RAM block status control:	NvRamManager		
Reliability:	NoProtection		
Store cyclic:	▣		
Cyclic writing period:			
Writing frequency:	1		
Writing priority:	Low		
Calc RAM block CRC:	▣	Check static block id:	▣
Read only:	▣	Resistant to changed sw:	▣
Restore at start:	▣	Store at shutdown:	▣
Store immediate:	▣	Store emergency:	▣
Use CRC comp mechanism	▣	Auto validation at shut down:	▣
Write verification:	▣	Write only once:	▣

Configuring ECUs You can add an NvM basic software module to an ECU configuration.

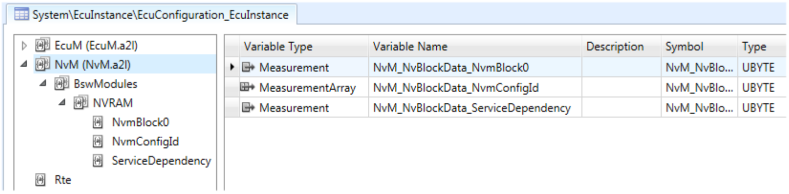
SystemDesk provides the following commands to auto-configure and generate the NVRAM manager:

- Updating the NvM module configuration according to the specified service needs.
- Generating a basic software component for the NVRAM manager. This includes automatically connecting the basic software component with the application software components that request services.
- Generating code to simulate the NVRAM manager for virtual validation.

Simulating systems You can experiment using A2L variables that are generated for the NVRAM manager.

For this, SystemDesk generates an array that describes the NVRAM for monitoring it.

The following illustration shows A2L variables that are generated for an example NVRAM manager.



System\EcuInstance\EcuConfiguration_EcuInstance

- EcuM (EcuM.a2l)
- NvM (NvM.a2l)
  - BswModules
    - NVRAM
      - NvmBlock0
      - NvmConfigId
      - ServiceDependency
- Rte

Variable Type	Variable Name	Description	Symbol	Type
Measurement	NvM_NvBlockData_NvmBlock0		NvM_NvBlo...	UBYTE
MeasurementArray	NvM_NvBlockData_NvmConfigId		NvM_NvBlo...	UBYTE
Measurement	NvM_NvBlockData_ServiceDependency		NvM_NvBlo...	UBYTE

Basic software descriptions

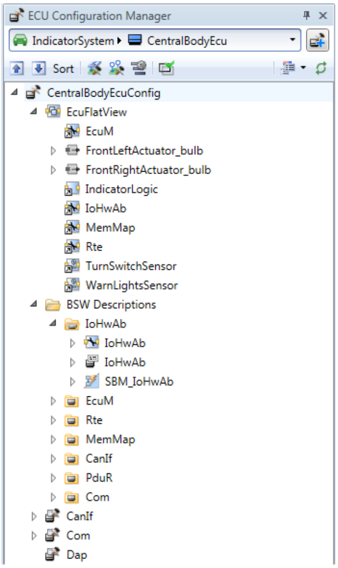
SystemDesk now uses the basic software module description template for the elements of basic software components according to AUTOSAR. Therefore, configuring ECUs with SystemDesk is now fully compatible with AUTOSAR. This lets you exchange basic software modules with third-party tools according to AUTOSAR.

The following table explains elements in SystemDesk's ECU Configuration Manager:

Type				Description
ECU configuration				The entire configuration of the basic software and the RTE of a single ECU.
	ECU flat view			Contains prototypes of all the atomic software components that are mapped to an ECU instance, and also software components for the basic software and the RTE. The ECU flat view is flat in the sense that the hierarchy that is introduced by composition SW components in the application software is resolved by removing the compositions and connecting the atomic software components directly. However, SystemDesk also shows the delegation ports.
		SW component prototype		Prototype of an application or basic software component.
		...		
		Port		Delegation port of a composition SW component.
			Interface	Interface that is assigned to a delegation port.
		...		
	BSW Descriptions			The elements below BSW Descriptions are managed automatically by SystemDesk. Typically, a user only modifies parameters of module configurations and the BSW descriptions are automatically adapted to the configuration.
		BSW description		
			SW component type	Defines the AUTOSAR interface of the basic software module to the application software.

Type			Description
		BSW module description	Defines the standardized interface of the basic software module to other basic software modules.
		SWC-BSW mapping	Maps elements of the *SW component type* to elements of the *BSW module description*.
		…	
	Module configuration		Specifies configuration parameters of the basic software component for generating the basic software module code and the *BSW description*.
	…		

The following illustration shows an example ECU configuration in the ECU Configuration Manager.



- CentralBodyEcuConfig
  - EcuFlatView
    - EcuM
    - FrontLeftActuator_bulb
    - FrontRightActuator_bulb
    - IndicatorLogic
    - IoHwAb
    - MemMap
    - Rte
    - TurnSwitchSensor
    - WarnLightsSensor
  - BSW Descriptions
    - IoHwAb
      - IoHwAb
      - IoHwAb
      - SBM_IoHwAb
    - EcuM
    - Rte
    - MemMap
    - CanIf
    - PduR
    - Com
  - CanIf
  - Com
  - Dap

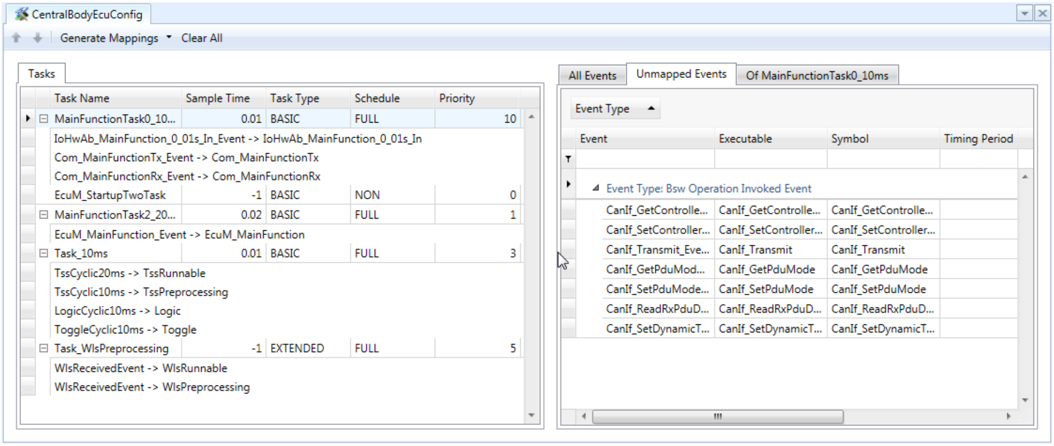
Changed runnable entity to OS task mapping

With this version of SystemDesk, you can map runnable and BSW schedulable entities according to AUTOSAR via their RTE events.

The Runnable Mapping Editor provides lists of all the relevant RTE events that trigger executables, i.e., runnable entities and BSW schedulable entities.

In the editor, you can create OS tasks and map executables to them via their triggering events. For each OS task, you can specify the order of executable activation.

The following illustration shows the Runnable Mapping Editor for an example ECU configuration.



Task Name	Sample Time	Task Type	Schedule	Priority
MainFunctionTask0_10...	0.01	BASIC	FULL	10
IoHwAb_MainFunction_0_01s_In_Event -> IoHwAb_MainFunction_0_01s_In				
Com_MainFunctionTx_Event -> Com_MainFunctionTx				
Com_MainFunctionRx_Event -> Com_MainFunctionRx				
EcuM_StartupTwoTask	-1	BASIC	NON	0
MainFunctionTask2_20...	0.02	BASIC	FULL	1
EcuM_MainFunction_Event -> EcuM_MainFunction				
Task_10ms	0.01	BASIC	FULL	3
TssCyclic20ms -> TssRunnable				
TssCyclic10ms -> TssPreprocessing				
LogicCyclic10ms -> Logic				
ToggleCyclic10ms -> Toggle				
Task_WlsPreprocessing	-1	EXTENDED	FULL	5
WlsReceivedEvent -> WlsRunnable				
WlsReceivedEvent -> WlsPreprocessing				

Event	Executable	Symbol	Timing Period
Event Type: Bsw Operation Invoked Event			
CanIf_GetControlle...	CanIf_GetControlle...	CanIf_GetControlle...	
CanIf_SetController...	CanIf_SetController...	CanIf_SetController...	
CanIf_Transmit_Eve...	CanIf_Transmit	CanIf_Transmit	
CanIf_GetPduMod...	CanIf_GetPduMode	CanIf_GetPduMode	
CanIf_SetPduMode...	CanIf_SetPduMode	CanIf_SetPduMode	
CanIf_ReadRxPduD...	CanIf_ReadRxPduD...	CanIf_ReadRxPduD...	
CanIf_SetDynamicT...	CanIf_SetDynamicT...	CanIf_SetDynamicT...	

Improved Generate Mappings command

With this version, SystemDesk's feature for generating a mapping of runnables to OS tasks has been improved.

The Generate Mappings command that is available via the Runnable Mapping Editor now lets you generate a mapping for unmapped events to OS tasks. The command maps the events either to existing OS tasks or creates new OS tasks if required.

For reference information on the Generate Mappings command, refer to *Runnable Mapping Editor* (📖 *SystemDesk 4.x Reference*).

Improved RTE generation

With this version of SystemDesk, RTE generation has been improved for a more complete support of the RTE API.

Contact dSPACE support for detailed information on SystemDesk's support of the RTE API.

Improved RTE interventions

RTE interventions for delegation ports You can now create RTE interventions for read/write access to data elements of delegation ports. This lets you test compostion software components in simulations for virtual validation.

Configuring RTE interventions more easily The RTE intervention editor now lets you configure RTE interventions more easily. It provides the Select elements for DAP interventions points command to create RTE interventions that can be accessed externally of the V-ECU. The Select elements for RTE service ports intervention points command lets you create RTE interventions that can be accessed internally of the V-ECU via RTE service ports.

Quickly generating V-ECUs with the V-ECU wizard

The V-ECU wizard replaces SystemDesk's System wizard and improves the quick generation of a V-ECU for a given atomic software component or composition software component.

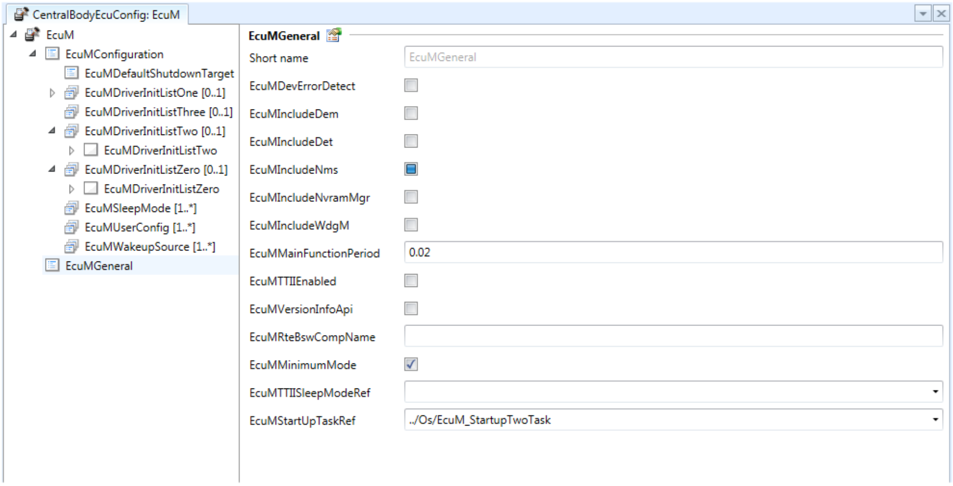
The wizard lets you create a system, add or create an ECU instance, create an ECU configuration, and automatically configure it. As a result, SystemDesk generates a V-ECU implementation and lets you build it for VEOS.

This way you can quickly simulate a software component on the virtual functional bus (VfB) level. SystemDesk maps the selected SWC to a single ECU, which acts as a *virtual functional bus* (VFB).

Improved editor for configuring basic software modules

The BSW Module Editor for configuring basic software module configurations has been redesigned for improved clarity and usability.

The illustration below shows the new layout.



CentralBodyEcuConfig: EcuM

- EcuM
  - EcuMConfiguration
    - EcuMDefaultShutdownTarget
    - EcuMDriverInitListOne [0..1]
    - EcuMDriverInitListThree [0..1]
    - EcuMDriverInitListTwo [0..1]
      - EcuMDriverInitListTwo
    - EcuMDriverInitListZero [0..1]
      - EcuMDriverInitListZero
    - EcuMSleepMode [1..*]
    - EcuMUserConfig [1..*]
    - EcuMWakeupSource [1..*]
  - EcuMGeneral

EcuMGeneral

Short name	EcuMGeneral
EcuMDevErrorDetect	□
EcuMIncludeDem	□
EcuMIncludeDet	□
EcuMIncludeNms	■
EcuMIncludeNvramMgr	□
EcuMIncludeWdgM	□
EcuMMainFunctionPeriod	0.02
EcuMTTIIEnabled	□
EcuMVersionInfoApi	□
EcuMRteBswCompName	
EcuMMinimumMode	☑
EcuMTTIISleepModeRef	
EcuMStartUpTaskRef	../Os/EcuM_StartupTwoTask

Further reading

For more details on configuring ECUs and generating V-ECU implementation with SystemDesk, refer to *Configuring ECUs and Generating V-ECU Implementations* (📖 *SystemDesk 4.x Guide*).

Simulating Systems

Improvement for debugging V-ECUs

You can now specify to build a V-ECU either in Release or in Debug configuration. Building a V-ECU in Debug configuration allows you to debug the V-ECU as the required information is generated during the V-ECU build.

The availability of the Debug configuration depends on the selected simulation target.

Further reading

For more details on debugging V-ECUs, refer to *Debugging and Analyzing Simulations* (📖 *SystemDesk 4.x Guide*).

Automating SystemDesk

Improvement for creating lists of elements via automation

SystemDesk's automation feature has been improved to enable creating lists of elements via automation. For each automation interface with an addNew method, the automation now provides an additional addNewRange method. The newly introduced addNewRange method provides improved performance for creating large numbers of elements.

Consider the following example for creating package elements via automation:

```
def AddNewPackages(rootPackage, amount):
    packages = []
    for i in range(0, amount):
        package = rootPackage.ArPackages.AddNew()
        package.ShortName = "package_" + str(i)
        packages.append(package)
    return packages
```

Using the newly introduced addNewRange method as in the following listing provides improved performance:

```
def AddNewRangeOfPackages(rootPackage, amount):
    shortNames = []
    for i in range(0, amount):
        shortName = "package_" + str(i)
        shortNames.append(shortName)
    packages = rootPackage.ArPackages.AddNewRange(shortNames)
    return packages
```

Further reading

For more details on automating SystemDesk, refer to *Programming SystemDesk Automation* (📖 *SystemDesk 4.x Guide*).

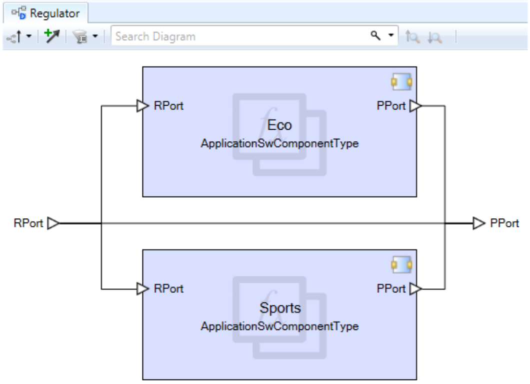
Variation Binding

Objective

SystemDesk lets you bind variant-rich models to work with a selected variant of a variant-rich model.

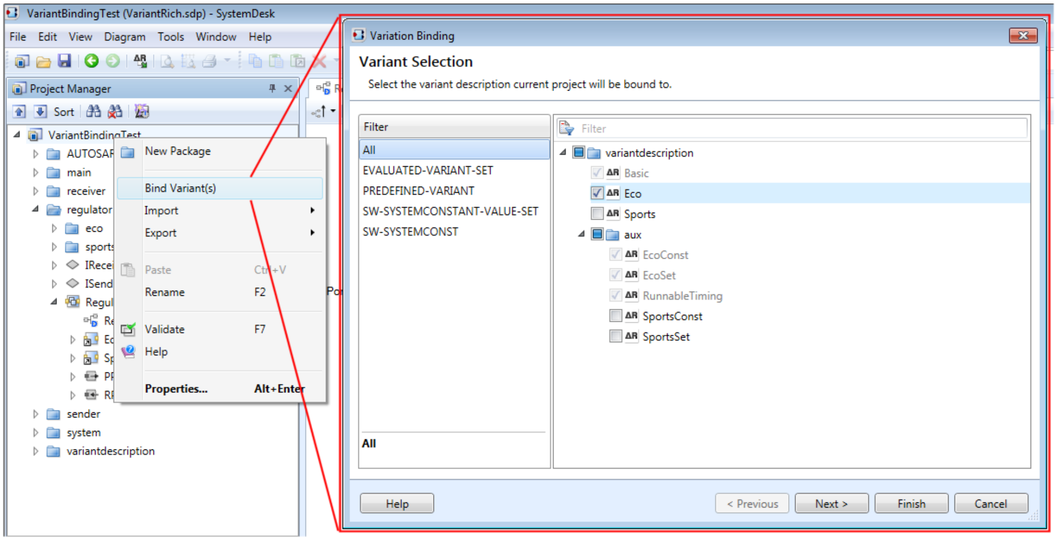
Importing variant-rich models

The illustration below shows the composition diagram of an imported variant-rich model in SystemDesk. The illustrated Regulator composition is modeled for different variants.



Variation binding

SystemDesk lets you bind the imported variant-rich model. The illustration below shows selecting the variant to bind to.



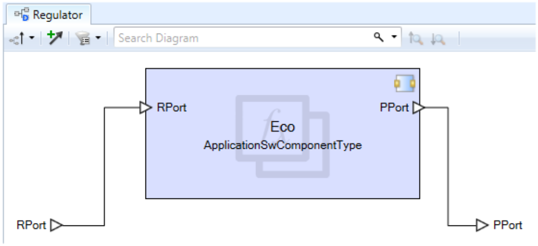
Variant Selection

Select the variant description current project will be bound to.

- variantdescription
  - ☑ Basic
  - ☑ Eco
  - □ Sports
  - aux
    - ☑ EcoConst
    - ☑ EcoSet
    - ☑ RunnableTiming
    - □ SportsConst
    - □ SportsSet

Working with the resulting variant

As a result, SystemDesk binds the variant-rich model to the selected variant. The following illustration shows the composition diagram of the Regulator composition that is now bound to the Eco variant. You can now start with working with the Eco variant in SystemDesk.



Further reading

For more details on variation binding with SystemDesk, refer to *Variation Binding* (📖 *SystemDesk 4.x Guide*).

Migrating to SystemDesk 4.5

Migrating to SystemDesk 4.5

SystemDesk 4.5 automatically migrates SystemDesk 4.3 and 4.4 SDP project files upon loading.

> You are recommended to install the most recent patch for SystemDesk 4.3 or 4.4. Then, save the SDP project files you want to migrate before opening them in SystemDesk 4.5.

Migrating BSW modules

With this version, SystemDesk describes basic software using the basic software module template according to AUTOSAR. Therefore, SystemDesk migrates basic software to the new description when you load SystemDesk 4.3 and 4.4 SDP project files in SystemDesk 4.5. To export V-ECU implementations or simulate V-ECUs of migrated SDP files, you have to perform *auto-configure and generate* on the respective ECU configurations and export the V-ECU implementations or build them as required.

However, if you want to migrate a SystemDesk project with basic software that is not supported for dSPACE virtual validation, you have to migrate that basic software manually. To do so, you have to export them to the AUTOSAR format in SystemDesk 4.3 or 4.4, create according modules in SystemDesk 4.5 and import the exported AUTOSAR files.

TargetLink

Where to go from here

Information in this section

New Features of TargetLink 4.1 and TargetLink Data Dictionary 4.1	192
Migrating to TargetLink 4.1 and TargetLink Data Dictionary 4.1	218

New Features of TargetLink 4.1 and TargetLink Data Dictionary 4.1

Where to go from here

Information in this section

Modeling in Simulink or Stateflow	192
Code Generation Core Functionality	195
Component-Based Development	199
AUTOSAR	200
Target Simulation (PIL)	203
Data Dictionary and Data Management	206
Code Generator Options	208
Tool Chain Integration	209
Other	211
API Functions and Hook Functions	215

Modeling in Simulink or Stateflow

Where to go from here

Information in this section

Newly Supported Simulink Blocks	193
Improvements to Custom Look-up Tables	193
Support of Simulink's Simplified Mode and IC Structures	194
Support of Structures in Stateflow Action Language	194

Newly Supported Simulink Blocks

Supported Simulink blocks

TargetLink now supports the following Simulink blocks:

- Signal Conversion block:

  TargetLink supports the Simulink Signal Conversion block, which can be used to rescale bus signals, for example.

- Bus Assignment block:

  A supported Simulink block that simplifies modeling with Simulink buses.

Related documentation

- *Signal Conversion Block* (📖 *TargetLink Block and Object Reference*)
- *Code-Relevant Simulink Blocks* (📖 *TargetLink Block and Object Reference*)

Improvements to Custom Look-up Tables

Variables inherit an input signal's dimension

Variables specified via the `tlscript` command can inherit their dimension from an input signal. This allows, for example, the implementation of a last index state variable for the Local search table search method.

Related documentation

- *Basics on Using Custom Look-Up Functions* (📖 *TargetLink Preparation and Simulation Guide*)

Support of vectors and matrices

In addition to scalar signals, the custom look-up script mechanism now supports vector and matrix input signals.

Related documentation

- *Obsolete Limitations* on page 244

New demo model

The new demo model TABLE1D_USR_LOCAL shows how custom look-up tables process vector and matrix input signals and how you can implement a local search.

Related documentation

- *Lookup Tables, User-Written Lookup Functions* (📖 *TargetLink Demo Models*)
- *TABLE1D_USR_LOCAL* (📖 *TargetLink Demo Models*)

Support of Simulink's Simplified Mode and IC Structures

In order to determine initial values that are not specified in the model, TargetLink supports Simulink's simplified initialization mode, which you can select by setting Simulink's Underspecified initialization detected parameter to `Simplified`.

Additionally, in this mode you can initialize bus-capable blocks by using Simulink's initial condition (IC) structures. These structs are now also supported by TargetLink 4.1.

Related documentation *Initializing Buses Via Initial Condition Structures* (📖 *TargetLink Customization and Optimization Guide*)

Support of Structures in Stateflow Action Language

Bus signals at Stateflow charts

TargetLink now supports Simulink data objects using the `Simulink.Bus` data type for Stateflow variables, and Simulink buses at the inputs and outputs of Stateflow charts and at Stateflow-internal data. To access those signals, TargetLink supports structures in the Stateflow action language. The associated data objects must reference either a structured Typedef or a structured DD Variable object.

Related documentation

- *Basics on the Representation of Buses in the Production Code* (📖 *TargetLink Customization and Optimization Guide*)
- *Basics on the Compatibility of Buses and Predefined Structs* (📖 *TargetLink Customization and Optimization Guide*)

Code Generation Core Functionality

Where to go from here

Information in this section

MISRA-C Compliance	195
Improved Code Efficiency	196

MISRA-C Compliance

Improvements to TargetLink's Fixed-Point Library

Several improvements were made to TargetLink's Fixed-Point Library so it complies with MISRA-C. The improvements include the following:

- The value of an expression of integer type shall not be implicitly converted to a different underlying type if a) it is not a conversion to a wider integer type of the same signedness, b) the expression is complex, c) the expression is not constant and is a function argument, d) the expression is not constant and is a return expression.
- The unary minus operator shall not be applied to an operand whose underlying type is unsigned.
- Before preprocessing, a null statement shall only occur on a line by itself. It may be followed by a comment, provided that the first character following the null statement is a white-space character.
- Removed superfluous division by zero protection: The code protection handling division by zero operations is generated by the Code Generator when needed and is not additionally contained in TargetLink's Fixed-Point Library.

Code Generator improvements

Code Generator improvements to comply with MISRA-C:

- TargetLink does not generate bit-wise XOR operations in look-up table code anymore.
- TargetLink no longer converts subtractions with unsigned integer results into additions using the compute-through-overflow (CTO) calculation method. Instead, the subtraction is generated into the production code, improving readability and leading to MISRA compliance.

≤ TargetLink 4.0	TargetLink 4.1
U8 = U8 + 255;	U8 = U8 - 1;

- Dereferences in logical operations are now written in parentheses.

≤ TargetLink 4.0	TargetLink 4.1
Sa1_OutPort1 = (Int16) (*In2 || *In1);	Sa1_OutPort1 = (Int16) **(**(*In2) || (*In1)**)**;

- There are code changes regarding *Assignments of Stateflow State IDs* and *Assignments to bitfields*. For details, refer to *Assignments to bitfields* on page 237.

Further improvements

Further improvements to comply with MISRA-C:

- By default, TargetLink now generates the `void` data type instead of the `Void` typedef.

  Also refer to *No Void typedef* on page 229.

Improved Code Efficiency

Dimension downgrade

TargetLink can now replace accesses to the same index range of a matrix variable by a scalar variable.

Additionally, elements of vector and matrix variables can now be replaced if they are based on the same non-loop-variable expression:

TargetLink < 4.1	TargetLink 4.1

vector2[e-d] = c[e-d]*7;
g(&**vector2[e-d]**);
	Aux_S16_a = c[e-d]*7;
	g(&**Aux_S16_a**);

This also works for vectors and matrices as well as index access ranges with relative offsets.

Related documentation

- *Basics on Optimizing Variables* (📖 *TargetLink Customization and Optimization Guide*)
- *Basics on Eliminating Temporary Variables* (📖 *TargetLink Customization and Optimization Guide*)

Copy propagation

TargetLink can now remove variables in more contexts.

≤ TargetLink 4.0	TargetLink 4.1

if (cond) {
   b = a + 1;
}
a = **b**;
	if (cond) {
	   a = a + 1;
	}

<table>
<tr><th>≤ TargetLink 4.0</th><th>TargetLink 4.1</th></tr>
<tr><td>

```
if (cond1) {
   if (cond2) {
      S = In1;
   } else {
      S = X;
   }
} else {
   S = In2;
}
X = S;
```

</td><td>

```
if (cond1) {
   if (cond2) {
      X = In1;
   }
} else {
   X = In2;
}
```

</td></tr>
</table>

Related documentation

- *ERASABLE* (🕮 *TargetLink Customization and Optimization Guide*)

Moving code into conditionally executed branches

TargetLink can now move more code into conditionally executed branches. This is especially visible for vector or matrix code:

<table>
<tr><th>≤ TargetLink 4.0</th><th>TargetLink 4.1</th></tr>
<tr><td>

```
if (Cond1[0]) {
      S1[0] = Input1[0];
   } else {
      S1[0] = Input2;
   }

…

   if (Cond1[3]) {
      S1[3] = Input1[3];
   } else {
      S1[3] = Input2;
   }

   /* Switch: CascSw/S2
      CascSw/S2: Omitted comparison with constant. */
   if (Cond2[0]) {
      Output[0] = (Int16) (-S1[0]);
   } else {
      Output[0] = 1;
   }

…

   if (Cond2[3]) {
      Output[3] = (Int16) (-S1[3]);
   } else {
      Output[3] = 4;
   }
```

</td><td>

```
   /* Switch: CascSw/S2 [0]
      CascSw/S2: Omitted comparison with constant. */
   if (Cond2[0]) {
      if (Cond1[0]) {
      Output[0] = (Int16) (-Input1[0]);
      } else {
         Output[0] = (Int16) (-Input2);
      }
   } else {
      /* # combined # TargetLink outport: CascSw/Output */
      Output[0] = 1;
   }

…

   /* Switch: CascSw/S2 [3]
      CascSw/S2: Omitted comparison with constant. */
   if (Cond2[3]) {
      if (Cond1[3]) {
         Output[3] = (Int16) (-Input1[3]);
      } else {
         Output[3] = (Int16) (-Input2);
      }
   } else {
      Output[3] = 4;
   }
```

</td></tr>
</table>

Related documentation

- *MOVABLE* (🕮 *TargetLink Customization and Optimization Guide*)

Better optimization of Assignment blocks in iterated subsystems

The production code generated for Assignment blocks that reside in iterated subsystems and whose Omit dispensable initializations checkbox is cleared was improved:

≤ TargetLink 4.0	TargetLink 4.1

for (Sa3_For_Iterator_it = 0; Sa3_For_Iterator_it <= 9;
Sa3_For_Iterator_it++) {
   if (Sa3_For_Iterator_it == 0) {
      for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) {
         /* Assignment: … - output initialization */
         Ass_For[Aux_S32] = X_UD_For[Aux_S32];
      }
   }
 	
   /* Assignment: … - output calculation
      # combined # Product: … */
   Ass_For[Sa3_For_Iterator_it] =
                Op(Sa3_For_Iterator_it);
   for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) {
      /* Unit delay: … */
      X_UD_For[Aux_S32] = Ass_For[Aux_S32];
    }
}
for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) {
   /* TargetLink outport: … */
   OutFor[Aux_S32] = Ass_For[Aux_S32];
}
	for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) {
	   /* Assignment: … - output initialization */
	   Ass_For[Aux_S32] = X_UD_For[Aux_S32];
	}
	for (Sa3_For_Iterator_it = 0; Sa3_For_Iterator_it <= 9;
	Sa3_For_Iterator_it++) {
	   /* Assignment: … - output calculation
	      # combined # Product: Direct_Init/For/Product */
	   Ass_For[Sa3_For_Iterator_it] =
	                Op(Sa3_For_Iterator_it);
	   for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) {
	      /* Unit delay: …_For */
	      X_UD_For[Aux_S32] = Ass_For[Aux_S32];
	   }
	}
	
	for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) {
	   /* TargetLink outport: … */
	   OutFor[Aux_S32] = Ass_For[Aux_S32];
	}
	Which can be simplified as follows:
	for (Aux_S32 = 0; Aux_S32 < 10; Aux_S32++) {
	   /* TargetLink outport: …
	      # combined # Assignment: … - output calculation
	      # combined # Product: … */
	   OutFor[Aux_S32] = **Op(Aux_S32);**
	}

Optimization of Rte_IRead() pointer return variables

TargetLink no longer generates unnecessary return variables for `Rte_IRead()` and Rte_IWriteRef():

≤ TargetLink 4.0	TargetLink 4.1

if (cond1) {
   p_DE = Rte_IRead_Run_DE();
   … /* Use p_DE */
 	
}
 	
…
 	
p_DE_a = Rte_IRead_Run_DE();
 	
… /* Use p_DE_a */
	p_DE = Rte_IRead_Run_DE();
	if (cond1) {
	   … /* Use p_DE */
	}
	…
	… /* Use p_DE */

More algebraic simplifications for auxiliary variables used for code patterns

TargetLink can now perform further optimizations for auxiliary variables used for accumulation in code patterns for complex operations:

≤ TargetLink 4.0	TargetLink 4.1

`aux = 42;`
`aux += 0;`
`aux -= 0;`
`aux *= 1;`
`aux /= 1;`
	`aux = 42;`

Component-Based Development

Improvements to Function Reuse

Function reuse for incremental subsystems and referenced models

With this TargetLink version, function reuse is possible not only for simple atomic subsystems but also for subsystems configured for incremental code generation and referenced models (with multiple instances).

Related documentation

- *Basics on Function Reuse* (📖 *TargetLink Customization and Optimization Guide*)
- *MULTIPLE_INSTANCES_REFMODEL* (📖 *TargetLink Demo Models*)

Variable propagation for function reuse

TargetLink now lets you reuse variables of predecessor and successor blocks of the reusable system definition without generating additional interface variables.

Related documentation

- *Basics on Reusing Variables of Preceding and Subsequent Blocks* (📖 *TargetLink Customization and Optimization Guide*)
- *FUNCTION_REUSE* (📖 *TargetLink Demo Models*)

AUTOSAR

Where to go from here

Information in this section

Supported AUTOSAR Releases	200
Support of NvData Communication	201
Data Transformation	201
Activation Reasons of Runnables	201
Port-Defined Argument Values	202
Miscellaneous AUTOSAR Features	202

Supported AUTOSAR Releases

Supported AUTOSAR Releases

The following AUTOSAR Releases are supported:

AUTOSAR Release	Revision
4.2	4.2.1[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 4.1.

Support of NvData Communication

NvData communication

TargetLink supports implicit NvData communication as described by AUTOSAR. This includes the support of provide-require ports.

You can model access to the NVRAM via port blocks, data store memory blocks, and block parameters.

Via special modeling styles, TargetLink lets you reduce accesses to the NVRAM.

Related documentation

- *Modeling NvData Communication* (📖 *TargetLink AUTOSAR Modeling Guide*)
- *AR_NVDATA_TRANSFORMER* (📖 *TargetLink Demo Models*)

Data Transformation

Modeling and simulating error logic

TargetLink lets you use data transformation as described by AUTOSAR to model and simulate related error logic, e.g., for end-to-end protection for safety-critical applications or for Automotive Ethernet and SOME/IP.

Related documentation

- *Basics on Data Transformation* (📖 *TargetLink AUTOSAR Modeling Guide*)
- *How to Model and Simulate Transformation Error Logic in Sender-Receiver Communication* (📖 *TargetLink AUTOSAR Modeling Guide*)
- *tlTransformerError* (📖 *TargetLink API Reference*)
- *AR_NVDATA_TRANSFORMER* (📖 *TargetLink Demo Models*)

Activation Reasons of Runnables

Activation reasons

TargetLink lets you use activation reasons for your runnables as described by AUTOSAR:

TargetLink lets you specify DD ActivationReason objects in a Runnable object's subtree.

To model activation reasons, you use TargetLink InPort blocks.

Related documentation

- *Basics on Activation Reasons* (📖 *TargetLink AUTOSAR Modeling Guide*)
- *How To Model a Runnable's Activation Reasons* (📖 *TargetLink AUTOSAR Modeling Guide*)
- *AR_POSCONTROL* (📖 *TargetLink Demo Models*)

Port-Defined Argument Values

Port-defined argument values

TargetLink supports port-defined argument values as described by AUTOSAR.

In the model, you reference a DD PortDefinedArgument object at the TargetLink InPort block that represents the port-defined argument value.

In production code, TargetLink generates port-defined argument values as formal arguments within the runnable function's signature.

Related documentation

- *Basics on Port-Defined Argument Values* (📖 *TargetLink AUTOSAR Modeling Guide*)
- *How to Model Port-Defined Argument Values* (📖 *TargetLink AUTOSAR Modeling Guide*)

Miscellaneous AUTOSAR Features

Support of Rte_IWriteRef for NvData communication

TargetLink supports the `Rte_IWriteRef` API function for NvData communication.

Transformation ComSpecs

TargetLink can import and export communication specifications for data transformations and `TRANSFORMER_HARD_ERROR_EVENT`.

ImplementationPolicy of parameter prototypes

TargetLink can import and export implementation policies for parameter data prototypes.

The data is stored in the DD ImplementationPolicy property of parameter prototypes.

StepSize property

TargetLink can import and export a step size specified for measurement and calibration tools.

The data is stored in the StepSize property of primitive application data types or data prototypes.

ConstrLevel property

TargetLink can import and export constraint levels specified for measurement and calibration tools

The data is stored in the ConstrLevel property of primitive application data types or data prototypes.

For implementation data types, it is stored in the Constraints subtree.

Scaling for LINEAR category

TargetLink can import and export default values of scalings whose category is set to `LINEAR`

The data is stored in the DefaultValue property of DD Scaling objects.

Data status of DataReceiverComSpec objects

TargetLink supports the HandleDataStatus property at DD DataReceiverComSpec objects.

Target Simulation (PIL)

Where to go from here

Information in this section

Changes in the Target Simulation Modules	203
Folder for TSM Extensions	205

Changes in the Target Simulation Modules

New and discontinued compiler versions

The following table shows the compiler versions that are now supported by TargetLink 4.1, refer to the New and No Changes columns. Compiler versions that are no longer supported are listed in the Discontinued column.

Target	Compiler	New	No Changes	Discontinued
ARM CortexM3	Keil	5.1	—	—
C16x	TASKING	—	8.6	8.7
HCS12	Cosmic	—	—	4.8
	Metrowerk	—	—	5.1
M32R	Gaio	—	—	11, 9
	Renesas	—	—	5.1
MC56F83	Metrowerk	—	—	8.3
MPC55xx	Diab	—	—	5.9
	GreenHill	—	—	2013
	GNU	—	—	4.1
	Metrowerk	—	—	2.8
MPC55xxVLE	Diab	—	—	5.9
	GreenHill	—	—	2013
	Metrowerk	—	—	2.8
MPC57xxVLE	Diab	5.9	—	—
	GreenHill	2014	—	—
MPC560xVLE	Diab	—	5.9	—
	GreenHill	2014	2012	2013
RH850	GreenHill	2014	—	—
S12X[1]	Cosmic	—	4.8	—
	Metrowerk	—	5.1	—
SH2	Renesas	—	9.3	9.4
SH2A-FPU	Renesas	—	9.4	—
TriCore17xx	TASKING	5.0	3.2	4.3
	GNU	—	—	3.4
TriCore2xx	TASKING	5.0	—	—
	GNU	4.6	—	—
V850	GreenHill	2014	—	2013
	NEC	—	—	3.40
XC22xx	TASKING	—	3.0	—

[1] Freescale S12XEVB/S12XEVB_USB is replaced by the new Freescale EVB9S12XEP100.

For more details on the evaluation boards supported by TargetLink, refer to 🕮 *TargetLink Evaluation Board Hardware Reference*.

For further 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.

Discontinued boards

No longer supported, no longer distributed The following boards are no longer supported by TargetLink and no longer distributed by dSPACE:

- Freescale 56F83xx
- Freescale HCS12
- Freescale PowerPC MPC5500
- Freescale PowerPC MPC5500VLE
- NEC V850ES
- Renesas M32R

If you want to use the unsupported evaluation boards with TargetLink 4.1, contact dSPACE Support.

Still supported, no longer distributed The following boards are still supported by TargetLink but no longer distributed by dSPACE:

- Freescale S12XEVB/S12XEVB_USB is replaced by the new Freescale EVB9S12XEP100

Folder for TSM Extensions

Specifying the folder via the API

You can now specify the folder for TSM extensions not only via the TargetLink Preferences Editor but also via the API, for example:

```
TlTsmManager.exe -SetTsmExtensionFolder -Folder:C:\exp
```

Related documentation

- *How to Clone Target/Compiler Combinations to Outside the TargetLink Installation* (📖 *TargetLink Customization and Optimization Guide*) (the Preconditions in particular)

Data Dictionary and Data Management

Where to go from here

Information in this section

Improvements to the Data Dictionary	206
New DD MATLAB API Functions	207
Referencing DD CodegenOptions Objects at TargetLink Main Dialog Block	208

Improvements to the Data Dictionary

Predefined filter rule sets for different DD views

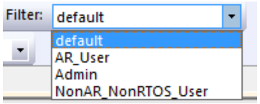
TargetLink comes with new predefined filter rule sets that can hide specific objects and properties in the DD tree. This makes it easier to view the relevant data. The filter rule sets are designed for typical user groups.

The following predefined filter rule sets are available:

- Admin - A filter rule set for administrators.
- AR_User - A filter rule set for AUTOSAR users.
- NonAR_NonRTOS_User - A filter rule set for users that use neither AUTOSAR nor RTOS.

For further information on the filter rule sets, refer to *DD_FILTER* (📖 *TargetLink Demo Models*).

You can change the filter rule set in the Filter list of the Data Dictionary Manager's toolbar.



Related documentation

- *Basics on Filter Rule Sets for the Data Model* (📖 *TargetLink Data Dictionary Basic Concepts Guide*)
- *How to Create Filter Rule Sets via DD Files* (📖 *TargetLink Data Dictionary Basic Concepts Guide*)

Filter views for the Object Comparison Navigator

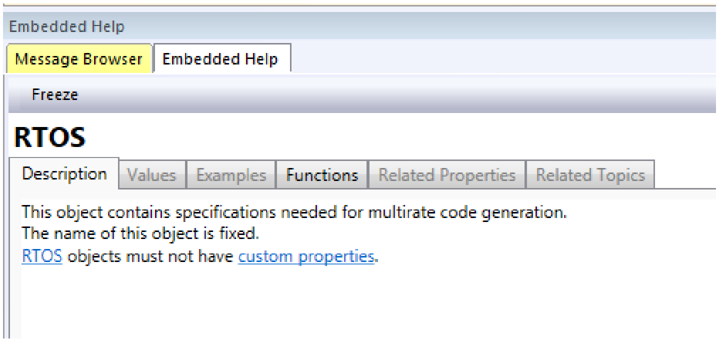
You can use the Filter list in the Object Comparison Navigator to hide specific objects and properties of no interest. This makes it easier to compare a high number of DD objects. You can either select one of the predefined filter rule sets or build your own.

Related documentation

- *Basics on Filter Rule Sets for the Data Model* (📖 *TargetLink Data Dictionary Basic Concepts Guide*)
- *Comparing and Merging Data Dictionary Objects* (📖 *TargetLink Data Dictionary Basic Concepts Guide*)

Message alerts

If the Message Browser pane is opened but hidden because of an active Embedded Help pane, the tab of the Message Browser indicates that there are new messages by changing color.



This object contains specifications needed for multirate code generation.
The name of this object is fixed.
RTOS objects must not have custom properties.

- A new Info message - blue tab
- A new Warning message - yellow tab
- A new Error message - red tab

Related documentation

- *Overview of the User Interface* (📖 *TargetLink Data Dictionary Basic Concepts Guide*)

New DD MATLAB API Functions

New DD API functions

TargetLink provides several new DD MATLAB API functions that are listed below. For more details, refer to the 📖 *TargetLink Data Dictionary Reference*.

GetAutosarVersion

```
[version, errorCode] = dsdd('GetAutosarVersion'[,<DD_identifier>]);
```

retrieves the Autosar version as specified with the `/Pool/Autosar/Config.AutosarVersion` property of the specified DD.

GetRenameBaseType

```
[version, errorCode] = dsdd('GetAutosarVersion'[,<DD_identifier>]);
```

retrieves the Typedef object which specifies the rename rule for a base data type.

Referencing DD CodegenOptions Objects at TargetLink Main Dialog Block

Referencing DD CodegenOptions options

For centralized handling and easier specification of consistent options, TargetLink 4.1 lets you directly reference DD CodegenOptions objects at the TargetLink Main Dialog block.

Related documentation *Basics on Configuring the Code Generator for Production Code Generation* (📖 *TargetLink Customization and Optimization Guide*)

Code Generator Options

New Code Generator Options

Overview of new Code Generator options

The following new Code Generator options are available with TargetLink 4.1.

- AvoidNestedVariablePropagationPointerAccess

  Generates additional pointers in reuse structures if these reuse structures access variables for which variable propagation is specified. For details, refer to *AvoidNestedVariablePropagationPointerAccess* (📖 *TargetLink Block and Object Reference*).

- ReportFailedFunctionReuseVariablePropagation

  Enables an optional report indicating problems that occurred in function reused systems during variable propagation. For details, refer to *ReportFailedFunctionReuseVariablePropagation* (📖 *TargetLink Block and Object Reference*).

- ReplaceUnrolledVectorsAndMatricesByScalar

  Controls the replacement of unrolled vector and matrix accesses by scalars. For details, refer to *ReplaceUnrolledVectorsAndMatricesByScalars* (📖 *TargetLink Block and Object Reference*).

For reference information on all Code Generator options, refer to *Alphabetical List of Code Generator Options* (📖 *TargetLink Block and Object Reference*).

Migration aspects of Code Generator options

Migration aspects include:

- Removed Code Generator option
- Changed Code Generator options
- Recommended compatibility settings
- Basics on changed defaults

For details, refer to *Migration Aspects Regarding Code Generator Options* on page 223.

Tool Chain Integration

Where to go from here

Information in this section

Exporting Functional Mock-up Units	210
Requirement Information in the Data Dictionary	210

Exporting Functional Mock-up Units

Functional Mock-up Units

TargetLink lets you generate Functional Mock-up Units (FMUs) for TargetLink subsystems. These FMUs are based on the FMI 2.0 standard in order to execute TargetLink code in FMI-compliant simulation environments.

These include VEOS, SCALEXIO and a large range of third-party FMI-compliant tools (https://www.fmi-standard.org/tools).

Related documentation

- *Definition of the FMI Standard and FMUs* (📖 *TargetLink Interoperation and Exchange Guide*)
- *Basics on Exporting FMUs from TargetLink* (📖 *TargetLink Interoperation and Exchange Guide*)

Requirement Information in the Data Dictionary

Storing requirement information as DD objects

TargetLink lets you store requirement information in the Data Dictionary as RequirementInfo objects. These objects act as a proxy to your requirements management system.

In the model, you can reference these objects at TargetLink blocks and from Stateflow objects. This instructs TargetLink to add requirement information as comments to the generated production code and the generated documentation.

You can programmatically handle the block data for requirement information via the `tlRequirementInfo()` API function, which is used instead of `tl_set()` or `tl_get()` for this data.

Related documentation

- *Basics on Requirement Information in the Generated Code* (📖 *TargetLink Interoperation and Exchange Guide*)
- *Basics on Using DD Based Requirement Information* (📖 *TargetLink Interoperation and Exchange Guide*)
- *tlRequirementInfo* (📖 *TargetLink API Reference*)

Other

Where to go from here

Information in this section

General Enhancements and Changes	211
TargetLink Demos	213

General Enhancements and Changes

TargetLink context menu

The context menu of TargetLink blocks provides two new options:

- Create reference (incr. subsystem to ref. model)
- Disable reference (ref. model to incr. subsystem)

Before, these options were available only in the dialog of the TargetLink Main block. You can use them to replace a subsystem configured for incremental code generation with a referenced model, and vice versa.

Related documentation

- *Common TargetLink Context Menu Options* (📖 *TargetLink Block and Object Reference*)

Improved simulation performance

TargetLink's simulation performance has been improved. The following types of models are now simulated with better performance in MIL simulation mode:

- Models that contain many scaling-invariant subsystems
- Models that use many workspace variables

> To maximize simulation performance for TargetLink models, it is best to use one of the following methods to start the simulation:
>
> - Via the **`tl_sim(model, parameters)`** (📖 *TargetLink API Reference*) API function instead of Simlunk's own `sim()` function
> - Via the TargetLink Main Dialog block
> - Via a TargetLink plot dialog

Related documentation

- None

Improved synchronization of function system signatures

The following improvements apply to the specification and synchronization of function system signatures:

- In addition to the DD Function object, now you can link a DD Signature object in a TargetLink Function block. DD Signature objects contain interface specifications (port data).
- You can perform consistency checks between the interface (port data) specified in the DD Signature object and the modeled interface of the Function system (Check ports).
- You can update/synchronize the interfaces in the model with the specifications of the DD Signature object (DD to Model).
- The SyncSystemSignature object in the `/Pool/ModelDesign/Config/` object tree no longer contains string properties. Instead, it contains typed data types like `Boolean` or `Enum`. This makes it easier to configure them in the Data Dictionary Manager.
- If you do not have any DD Signature object specifications but want to transfer your modeled interface data to the Data Dictionary, you can also synchronize the port data to the Data Dictionary (Model to DD).
- An HTML report is generated whenever you check or synchronize ports. The report contains detailed information about differences in the specification of the DD Signature object and the interface of the Function system.

Related documentation

- *Centrally Specifying and Synchronizing Function System Signatures* (🕮 *TargetLink Customization and Optimization Guide*)

Setting MEX and SIL compilers independently

You can now set a SIL compiler for the production code DLL independently of the MEX compiler for the simulation S-function. This lets you use the free MSVC Express Edition for compiling the production code and SIL debugging.

Related documentation

- *How to Set or Change MEX and SIL Compilers* (🕮 *TargetLink Preparation and Simulation Guide*)
- *How to Debug in SIL Simulation Mode* (🕮 *TargetLink Preparation and Simulation Guide*)
- *tlProductionCodeSILCompiler* (🕮 *TargetLink API Reference*)

TargetLink Demos

New demos

The following new demos come with TargetLink 4.1

Ar_nvdata_transformer This new demo shows two features:

- Read and Write access to nonvolative RAM via NVData Interfaces
- Use of AUTOSAR transformers: e.g., to model end-to-end communication protection for safety-critical applications

DD_filter The demo shows how to create XML filter rule sets based on DD files with the required specification. The demo contains the `tl_example_CreateDDFilterBasedOnDDFile` script and three generic DD files with the specification for typical filter use cases. The script generates an XML filter rule set.

> This demo does not contain any model.

Related Documentation

- *DD_FILTER* (🕮 *TargetLink Demo Models*)
- *Basics on Filter Rule Sets for the Data Model* (🕮 *TargetLink Data Dictionary Basic Concepts Guide*)

DD_ML_API This demo shows two examples of how to use the DD MATLAB API:

- The `simple.m` script creates a DD variable object and a user-defined generic DD object via dsdd commands.
- The `findCalVariables.m` script demonstrates how to find all variable objects with STATIC_CAL class in a DD file.

Related Documentation

- *DD_ML_API* (🕮 *TargetLink Demo Models*)

DD_ML_ImportExport This demo contains various M scripts that show how to import objects to the Data Dictionary from XLS and XML files.

Related Documentation

- *DD_ML_IMPORTEXPORT* (🕮 *TargetLink Demo Models*)

Function_reuse This demo shows the function reuse feature that is applied to subsystems with instance-specific initial parameter values by defining mask parameters.

Related Documentation

- *FUNCTION_REUSE* (📖 *TargetLink Demo Models*)
- *Basics on Function Reuse* (📖 *TargetLink Customization and Optimization Guide*)

Multiple_instances_refmodel This demo shows the function reuse feature that is applied to referenced models with instance-specific initial parameter values by defining model arguments.

Related Documentation

- *MULTIPLE_INSTANCES_REFMODEL* (📖 *TargetLink Demo Models*)
- *Basics on Function Reuse* (📖 *TargetLink Customization and Optimization Guide*)

Table1d_usr_local This demo shows how to replace TargetLink look-up table code that uses a local search algorithm with nonscalar inputs by custom look-up functions.

Related Documentation

- *TABLE1D_USR_LOCAL* (📖 *TargetLink Demo Models*)
- *Basics on Using Custom Look-Up Functions* (📖 *TargetLink Preparation and Simulation Guide*)

Variable_vector_width This demo model shows how to work with vectorized variables that have width variants. Using preprocessor macros for vector widths, the same model and generated production code can be used for all widths. The code of complete subsystems can become width-varying at code compile time.

Related Documentation

- *VARIABLE_VECTOR_WIDTH* (📖 *TargetLink Demo Models*)
- *Introduction to Variable Vector Widths* (📖 *TargetLink Customization and Optimization Guide*)

Improved demos

The following demos contain new feature demonstrations:

Ar_poscontrol This demo simulates activation reasons with a Stateflow chart triggering various RTE events. Signals are automatically adjusted to match the values specified in the DD.

Related Documentation

- *AR_POSCONTROL* (📖 *TargetLink Demo Models*)
- *Basics on Activation Reasons* (📖 *TargetLink AUTOSAR Modeling Guide*)

Poscontrol The demo now shows the connection of TargetLink function blocks with DD Function and DD Signature objects in the Data Dictionary to ensure interface consistency between functions and the modeled subsystems.

Related Documentation

- *POSCONTROL* (📖 *TargetLink Demo Models*)
- *Centrally Specifying and Synchronizing Function System Signatures* (📖 *TargetLink Customization and Optimization Guide*)

API Functions and Hook Functions

Where to go from here

Information in this section

New API Functions	215
New Hook Functions	216

New API Functions

Exporting FMUs

The `tl_generate_fmu(propertyName, propertyValue, ...)` API function lets you export functional mock-up units (FMUs) to use in FMI-compliant tools.

Related documentation

- *Definition of the FMI Standard and FMUs* (📖 *TargetLink Interoperation and Exchange Guide*)
- *Basics on Exporting FMUs from TargetLink* (📖 *TargetLink Interoperation and Exchange Guide*)
- *How to Generate an FMU to use in an FMI-Compliant Tool* (📖 *TargetLink Interoperation and Exchange Guide*)
- *TargetLink FMU Export* (📖 *TargetLink Tool and Utility Reference*)

Switching SIL compilers

The `tlProductionCodeSILCompiler` API function lets you set or change SIL compilers.

Related documentation

- *How to Set or Change MEX and SIL Compilers* (📖 *TargetLink Preparation and Simulation Guide*)

Setting requirement information at blocks

The **`tlRequirementInfo`** API function lets you manage DD-based requirement information at TargetLink blocks and Stateflow objects. **`tlRequirementInfo`** is used instead of **`tl_set()`** or **`tl_get()`** for this data.

Related documentation

- *Basics on Using DD Based Requirement Information* (📖 *TargetLink Interoperation and Exchange Guide*)

Preparing the simulation of transformation error logic

The **`tlTransformerError`** API function helps you prepare AUTOSAR models for simulation that contain transformation error logic.

Related documentation

- *Basics on Data Transformation* (📖 *TargetLink AUTOSAR Modeling Guide*)
- *How to Model and Simulate Transformation Error Logic in Sender-Receiver Communication* (📖 *TargetLink AUTOSAR Modeling Guide*)

New Hook Functions

Generating and synchronizing system signatures

TargetLink provides the following new hook functions to customize subsystem generation from the DD:

- **`tl_pre_add_ddsignatureport_hook`** This hook function is called before a new DD SignaturePort object is added to the specified DD Signature object.
- **`tl_post_add_ddsignatureport_hook`** This hook function is called after a new DD SignaturePort object is added to the specified DD Signature object.
- **`tl_pre_sync_systemsignatureport_hook`** This hook function is called before an existing Port block is synchronized with the corresponding DD Signature object.
- **`tl_post_sync_systemsignatureport_hook`** This hook function is called after an existing Port block is synchronized with the corresponding DD Signature object.

Related documentation *Basics on Using Hook Functions* (📖 *TargetLink Customization and Optimization Guide*)

Centrally Specifying and Synchronizing Function System Signatures (📖 *TargetLink Customization and Optimization Guide*)

Initializing buses via DD Variable objects

TargetLink provides the following customization file to customize the initialization of buses via Simulink IC structures:

- `tlGetBusStructMapping` This customization file gets the bus struct mappings (DD Variable objects to Simulink.Bus objects).

This customization file lets you you to map DD-based struct variables to Simulink.Bus objects.

Related documentation *How to Manually Create a Mapping Between a DD Variable and a Simulink Bus* (📖 *TargetLink Customization and Optimization Guide*)

Migrating to TargetLink 4.1 and TargetLink Data Dictionary 4.1

Upgrade process

To upgrade to a new TargetLink version you have to adjust the following:

- Your data dictionaries
- Your models
- Your scripts and hook functions

To migrate libraries/models from TargetLink versions older than 4.0, you also have to perform the migration steps of the TargetLink versions in between. Refer to the previous TargetLink Migration Guides available on your DVD.

You can launch an upgrade manually by using the **`tlUpgrade`** API function. For detailed instructions, refer to *How to Manually Upgrade Libraries and Models Via the API* on page 222.

Carefully read all of the following information and modify your tool chain accordingly.

Where to go from here

Information in this section

Upgrade of Models, Libraries, and Data Dictionaries	219
Code Generator Options	223
API Functions and Hook Functions	226
AUTOSAR-Related Migration Aspects	226
Code Changes	227
Other	242
Obsolete	244
Changes in Future TargetLink Versions	245

Upgrade of Models, Libraries, and Data Dictionaries

Where to go from here

Information in this section

Migrating to TargetLink 4.1	219
How to Upgrade a Data Dictionary with Included DD Files	220
How to Manually Upgrade Libraries and Models Via the API	222

Migrating to TargetLink 4.1

Indirect upgrade from TargetLink 2.x

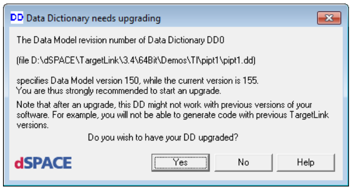
Libraries, models, and DD files from TargetLink versions prior to TargetLink 3.1 cannot be upgraded directly.

However, you can perform an indirect upgrade. First, migrate older libraries, models, and DD files to TargetLink 3.5. Then you can upgrade them to TargetLink 4.1.

Direct upgrade from TargetLink 3.1 or higher

TargetLink 4.1 automatically upgrades models, libraries, and Data Dictionaries created with TargetLink 3.1 or higher.

The following dialogs can appear during the upgrade process of the Data Dictionary file:



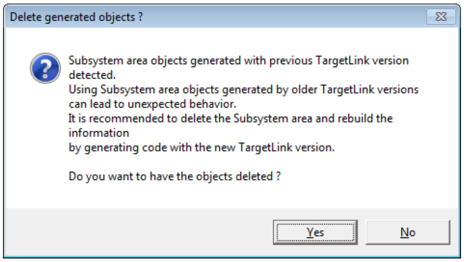
The Data Model revision number of Data Dictionary DD0

(file D:\dSPACE\TargetLink\3.4\64Bit\Demos\TI\pipt1\pipt1.dd)

specifies Data Model version 150, while the current version is 155.
You are thus strongly recommended to start an upgrade.

Note that after an upgrade, this DD might not work with previous versions of your software. For example, you will not be able to generate code with previous TargetLink versions.

Do you wish to have your DD upgraded?



Subsystem area objects generated with previous TargetLink version detected.
Using Subsystem area objects generated by older TargetLink versions can lead to unexpected behavior.
It is recommended to delete the Subsystem area and rebuild the information
by generating code with the new TargetLink version.

Do you want to have the objects deleted ?

User interaction In the following cases, user interaction is required:

- *Legacy libraries never prepared for TargetLink* on page 220
- *Migrating from TargetLink 32-bit to TargetLink 64-bit* on page 220
- *DD files with included partial DD files* on page 220
- *How to Manually Upgrade Libraries and Models Via the API* on page 222

Legacy libraries never prepared for TargetLink

Libraries created with TargetLink 3.x or 4.0 that were never prepared using the **`tl_prepare_system(propertyName, propertyValue, ...)`** (📖 *TargetLink API Reference*) API function cannot be upgraded automatically by TargetLink 4.1.

Solution

1. Open the library in the prior TargetLink version and prepare it for the upgrade by using **`tl_prepare_system(propertyName, propertyValue, ...)`** (📖 *TargetLink API Reference*).
2. Save the library.
3. Open the library with TargetLink 4.1.

Related documentation

- *How to Make TargetLink User Libraries Upgrade-Capable* (📖 *TargetLink Orientation and Overview Guide*)

Migrating from TargetLink 32-bit to TargetLink 64-bit

Custom code S-functions built with 32-bit TargetLink versions do not work with 64-bit versions of TargetLink and vice versa.

Solution Initiate a rebuild of all custom code S-functions using the **`tlUpgrade('Model',<MyModel>,'CheckModel','FixIssues')`** (refer to *tlUpgrade(propertyName, propertyValue, ...)* (📖 *TargetLink API Reference*)) API function.

DD files with included partial DD files

To upgrade DD files with included partial DD files, refer to *How to Upgrade a Data Dictionary with Included DD Files* on page 220.

How to Upgrade a Data Dictionary with Included DD Files

Objective

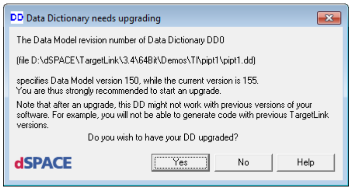
If you open a TargetLink model with an old Data Dictionary file that was not upgraded, you have to upgrade the Data Dictionary file.

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.



Data Dictionary needs upgrading

The Data Model revision number of Data Dictionary DD0

(file D:\dSPACE\TargetLink\3.4\64Bit\Demos\TI\pipt1\pipt1.dd)

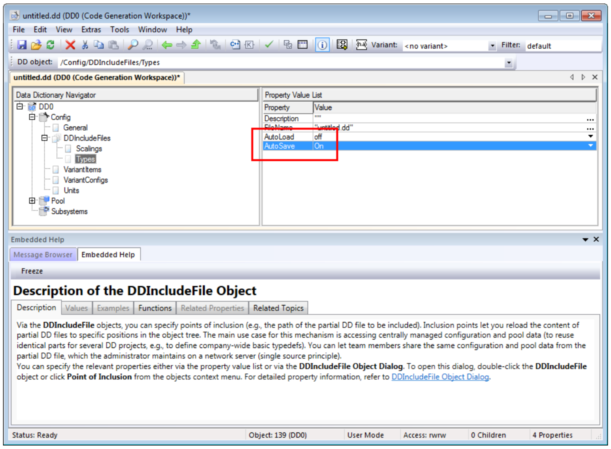
specifies Data Model version 150, while the current version is 155.
You are thus strongly recommended to start an upgrade.

Note that after an upgrade, this DD might not work with previous versions of your software. For example, you will not be able to generate code with previous TargetLink versions.

Do you wish to have your DD upgraded?

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.



DD object: /Config/DDIncludeFiles/Types

Property	Value
Description	""
FileName	"untitled.dd"
AutoLoad	off
AutoSave	On

Description of the DDIncludeFile Object

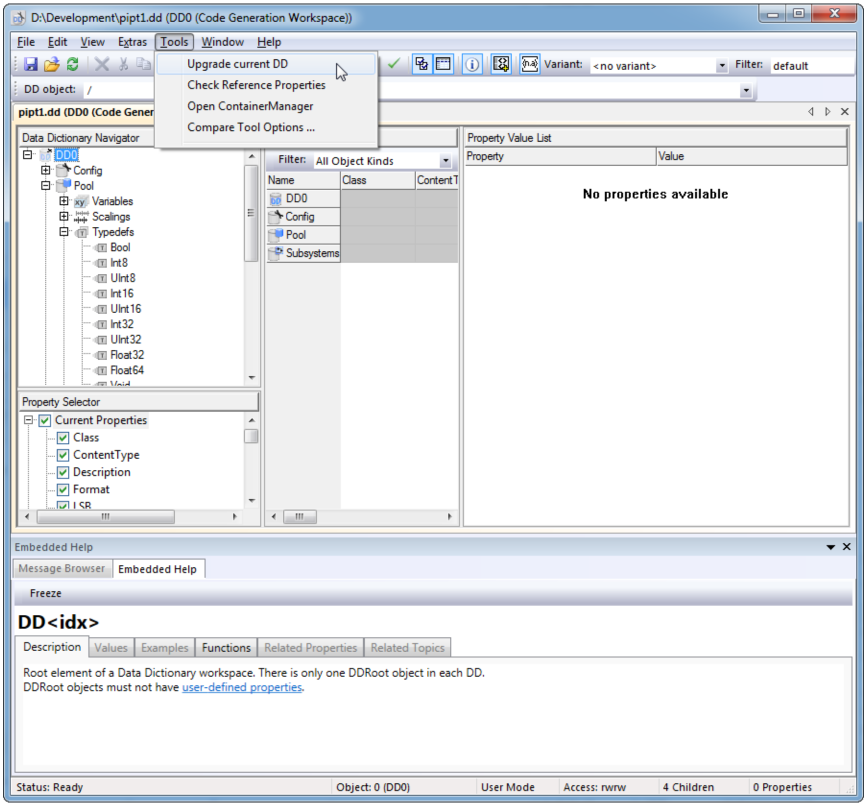
Via the **DDIncludeFile** objects, you can specify points of inclusion (e.g., the path of the partial DD file to be included). Inclusion points let you reload the content of partial DD files to specific positions in the object tree. The main use case for this mechanism is accessing centrally managed configuration and pool data (to reuse identical parts for several DD projects, e.g., to define company-wide basic typedefs). You can let team members share the same configuration and pool data from the partial DD file, which the administrator maintains on a network server (single source principle).
You can specify the relevant properties either via the property value list or via the **DDIncludeFile Object Dialog**. To open this dialog, double-click the **DDIncludeFile** object or click **Point of Inclusion** from the objects context menu. For detailed property information, refer to DDIncludeFile Object Dialog.

Status: Ready | Object: 139 (DD0) | User Mode | Access: rwrw | 0 Children | 4 Properties

This ensures that after the Data Dictionary and the included DD files were upgraded, the upgraded included DD files are saved when the Data Dictionary is saved. You can set these properties for a large number of included DD files via the Object Explorer.

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 DD Manager, or enter `dsdd('Upgrade')` in the MATLAB Command Window.



No properties available

DD<idx>

Root element of a Data Dictionary workspace. There is only one DDRoot object in each DD.
DDRoot objects must not have user-defined properties.

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 were 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

You can manually update libraries and models via the **`tlUpgrade(propertyName, propertyValue, ...)`** (📖 *TargetLink API Reference*) API function and save them afterwards, e.g., to prepare a central upgrade of libraries and models in a tool chain scenario with several users.

> When upgrading models and libraries, first upgrade models or libraries that do not reference any other libraries, i.e., the blocks/subsystems they contain have no links to other libraries. Start with the bottom library and then upgrade the libraries above it in ascending order.

For details on libraries that were never prepared, refer to *Legacy libraries never prepared for TargetLink* in *Migrating to TargetLink 4.1* on page 219.

Method

To manually upgrade libraries and models via the API

1. In the MATLAB Command Window, type **`dsdd_manage_project('Open','<name>.dd')`** to load the required and already upgraded DD project file (one way to upgrade DD project files is to use the **`dsdd('Upgrade'[,<DD_Identifier>])`** command, refer to **`Upgrade`** (📖 *TargetLink Data Dictionary Reference*)).
2. Type **`tlUpgrade('Model', '<Model|Library>.mdl', 'CheckModel','FixIssues')`** to upgrade single models or libraries.
3. Save the upgraded model or library files, e.g., `Library.mdl`.
4. Repeat steps 2 and 3 for all other models or libraries.

Result

You upgraded your models and libraries.

Code Generator Options

Migration Aspects Regarding Code Generator Options

Removed Code Generator option

The MPC5xx-specific (TOM-specific) EnableLogicalOperationOptimisation Code Generator option was removed from TargetLink:

Removed Option	Replacement Option	Compatibility Setting
EnableLogicalOperationOptimisation	None	None

With TargetLink 4.1, this option was removed together with MPC5xx-support.

Changed Code Generator options

The following Code Generator options changed with TargetLink 4.1:

- None

Recommended compatibility settings

Make the following settings for new TargetLink 4.1 Code Generator options to ensure best possible downward compatibility:

- None

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 CodegenOptions objects (DD-based option storage). You can use DD CodegenOptions-objects as a central source for overwriting model-based option settings.

If a model-based option value equals the old default value, it is automatically changed to the new default value during upgrade. If a DD-based option value equals the old default value, it is not changed to the new default value during upgrade but keeps the old value.

Option value = old default If Code Generator options equal 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 (TL_{Old} to TL_{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 (TL_{Old})	Option Value (≤ TL_{New})	
	Default = 9	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] 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 Code Generator options did not equal default values in the former TargetLink version (A) but do 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).

Upgrading TL_A ⇒ TL_B ⇒ TL_C and upgrading TL_A ⇒ TL_C can cause different option values (see the following table).

Suppose the default values for TargetLink versions A, B, and C read 9, 11, and 13. If an option value equaled 11 in version A, an upgrade to version C would change the option value as follows:

Upgrade Strategy	Option Value TL_A Default = 9	Option Value TL_B Default = 11	Option Value TL_C Default = 13
A ⇒ B ⇒ C	11 (≠ default)	11 (= default)[1]	13 (= default)[1]
A ⇒ C	11 (≠ default)	—	11 (≠ default)

[1] Option value is not stored with the model, because it equals the default.

New Code Generator options

For more details on new Code Generator options, refer to *New Code Generator Options* on page 208.

Related topics

References

- *Code Generator Options* (📖 *TargetLink Block and Object Reference*)

API Functions and Hook Functions

Changes in TargetLink and TargetLink Data Dictionary API Functions

Custom look-up functions

For the `tlscript` API command, a new property is available which relates to the dimension of input signals. This lets you implement a last index state variable for the Local search table search method, for example.

- `InheritDimensionFromInput`

Related documentation

- *Permissible Properties for Variables* (📖 *TargetLink API Reference*)
- *Basics on Using Custom Look-Up Functions* (📖 *TargetLink Preparation and Simulation Guide*)

AUTOSAR-Related Migration Aspects

AUTOSAR-Related Migration Aspects

Name macros in name templates of runnable objects

With TargetLink 4.1, the NameTemplate property of Runnable objects must not contain name macros other than `$D`.

Removed options in import/export

The Merge and EnablePackageSupport options are obsolete.

The corresponding properties in `/Pool/Autosar/Config/ImportExport` were removed.

TargetLink now always does the following:

- Merges Data Dictionary elements that belong to one software component but reside in different subsystems into one merged file (Merge = `On`).
- Imports and exports the package information provided in AUTOSAR files and the Data Dictionary (EnablePackageSupport = `On`).

Modify your scripts accordingly.

RTE error code macros in the Data Dictionary

The variables representing RTE error code macros in the `/Pool/Variables/AUTOSAR/Std_ReturnType` variable group and the predefined variable class `/Pool/VariableClasses/AUTOSAR/RTE/RTE_ERROR_CODE` now have their ModuleRef property set to `Rte_Frame`.

This is in accordance with the AUTOSAR standard, which requires these macros to be defined in `Rte.h`.

The new value of each variables' ModuleRef property is set automatically during a Data Dictionary upgrade.

This can result in an `#include Rte.h` in production code if one of these variables is referenced at a block in your model.

Code Changes

Code Changes

Code generated from Stateflow charts

In order to follow special Stateflow semantics more precisely (e.g., to avoid MIL/SIL differences for certain modeling styles), code generated from Stateflow charts might be less efficient in the following Stateflow chart scenarios:

- Graphical functions imported from other charts are called.
- Function call output events occur.

As a result, constants might not be propagated, or unused code fragments might not be removed.

To change this, assign a function class whose SIDE_EFFECT_FREE optimization flag is enabled to the imported graphical functions and to the subsystems/charts that are triggered by function call output events.

You must guarantee that the function is actually side-effect-free. For example, side-effect-free functions do not:

- Modify global variables
- Call functions that are not side-effect-free

Assignment blocks in iterated subsystems

If multiple Assignment blocks reside in an iterated subsystem, only one first iteration flag is implemented, which increases code efficiency. As a result, the name of such flags changes, for example:

TargetLink ≤ 4.0	TargetLink 4.1
`{Subsystem_AssignmentBlock}_FirstIter`	`{Subsystem_IterationBlock}_FirstIter`

The flag is explicitly initialized at the beginning of the iteration loop:

…`_FirstIter = 1`

The flag is reset at the end of the loop:

…`_FirstIter = 0`

Assignment blocks in nested subsystems If at least one Assignment block resides in an atomic subsystem and this atomic subsystems resides in an iterated subsystem, the following variable scopes are assigned to the iteration variable:

- *Global* if the function of the atomic subsystem is *not inlined*.
- *Local* if the function of the atomic subsystem is *inlined*.

Code patterns for saturated additions or subtractions

For TargetLink 4.1, compute-through-overflow (CTO) code patterns are never used in saturation code of additions or subtractions, if the ExploitComputeThroughOverflow Code Generator option is set to NEVER. The following example shows an addition `out = in + Const` with `I16Out = I16In +1`:

TargetLink ≤ 4.0	TargetLink 4.1

`if (I16In > 32766) { /* Max(Result type) - Const */`
`      I16Out = 32767;    }`
`else {`
`      I16Out = (Int16) (( /* CTO */ (UInt16) I16In) + 1);    }`
	`if (I16In > 32766) { /* Max(Result type) - Const */`
	`      I16Out = 32767;    }`
	`else {`
	`      I16Out = (Int16) (I16In + ((Int16) 1));    }`

If the ExploitComputeThroughOverflow Code Generator option is set to `Always` or to `Optimized`, the production code remains unchanged compared to previous versions.

No Void typedef

For improved MISRA-C compliance, TargetLink by default no longer generates the `Void` typedef within `tl_basetypes.h`. It uses the standard C `void` data type instead.

You can change this back to the old behavior (using `Void`) by editing the `TargetConfig.xml` file that belongs to your target-compiler combination.

This file is located in `<TL_InstRoot>\Matlab\Tl\TargetConfiguration\<MicrocontrollerFamily>\<CompilerFamily>` or similar in the TSM extension folder.

To instruct TargetLink, to generate the `Void` base type in `tl_basetypes.h` again, do the following:

1. Open the `TargetConfig.xml` file that belongs to your target-compiler combination.
2. Locate the `ddObj` XML element whose name attribute is set to `Void`.
3. Locate the child `ddProperty` XML element whose name attribute is set to `CodedType`.
4. Change its value from `Use standard C void type` to `void`.
5. Save the `TargetConfig.xml` file and generate code.

Additional default scaling code comments

For readability, TargetLink now places additional code comments concerning default scalings at variable definitions:

TargetLink ≤ 4.0	TargetLink 4.1
`Int16 SX1_OutPort1_FR_Actual;`	`Int16 SX1_OutPort1_FR_Actual /* LSB: 2^0 OFF:  0 MIN/MAX:  -32768 .. 32767 */;`

Encapsulation of preprocessor #IF statements

TargetLink now encapsulates preprocessor #include directives by #IF directives only if all of the included file's definitions and declarations are encapsulated by #IF directives.

As an example, consider the following `FuncDefModule.h` header file:

```
extern GLOBAL Int16 SEnc1_Out1;

#if FLAG
extern void Encapsulated(void);
#endif
```

TargetLink's generated code changes as follows:

TargetLink ≤ 4.0	TargetLink 4.1

`#if FLAG`
`#include "FuncDefModule.h"`
`#endif`
 	
…
 	
`void TL_Root(void)`
`{`
`   #if FLAG`
`      Encapsulated(Sa1_InPort);`
`   #endif`
 	
`   SEnc1_Out1 = …`
`}`
	`#include "FuncDefModule.h"`
	
	…
	
	`void TL_Root(void)`
	`{`
	`   #if FLAG`
	`      Encapsulated(Sa1_InPort);`
	`   #endif`
	
	`   SEnc1_Out1 = …`
	
	`}`

Code comments at indices of vector or matrix variables

For improved readability, the code comments concerning the initial values of vector or matrix variables have changed:

TargetLink ≤ 4.0	TargetLink 4.1
Vectors	

`Int16 MyVar[3] =`
`{`
`   /*[0..2]*/ 61, 52, 43`
`};`
	`Int16 MyVar[3] =`
	`{`
	`   /* [0..2] */ 61, 52, 43`
	`};`

Matrices	

`Int16 MyVar[2][3] = {`
`   {`
`      /*[0..2]*/ 61, 52, 43`
`   },`
`   {`
`      /*[0..2]*/ 34, 25, 16`
`   }`
`};`
	`Int16 MyVar[2][3] = {`
	`   {`
	`      /* [0][0..2] */ 61, 52, 43`
	`   },`
	`   {`
	`      /* [1][0..2] */ 34, 25, 16`
	`   }`
	`};`

Identifier of implicitly generated struct types

TargetLink's identifiers for implicitly generated struct types now comply with the AUTOSAR standard.They now comply with the following regular expression:

```
[a-zA-Z]([a-zA-Z0-9]|_[a-zA-Z0-9])*_?
```

For typedef identifiers that you specified by using the `$C`, `$R`, or `$S` name macros, TargetLink now does the following:

- Removes underscores at the beginning of the identifier of implicitly generated typedefs
- Replaces double underscores by a single one

TargetLink ≤ 4.0	TargetLink 4.1
/* update(s) for inport Subsystem/Func/In1_with_super_long_name_to_break_limit */ Rte_Pim_ACP_a(instance)->Sa2_In1_with_s**__**to_break_limit = (sint16) DataElement;	/* update(s) for inport Subsystem/Func/In1_with_super_long_name_to_break_limit */ Rte_Pim_ACP_a(instance)->Sa2_In1_with_s**_**to_break_limit = (sint16) DataElement;

Identifiers that consist only out of underscores or that begin with an underscore immediately followed by a numeral are not changed. These typedefs are not generated into an autogenerated per instance memory (PIM):

TargetLink ≤ 4.0	TargetLink 4.1
/* update(s) for inport Subsystem/Func/In1 */ Rte_Pim_ACP_a(instance)->**_1Var** = (sint16) DataElement;	/* update(s) for inport Subsystem/Func/In1 */ **_1Var** = (sint16) DataElement;

IF-variable for outports of conditionally executed systems

To eliminate possible differences between MIL and SIL simulations, TargetLink now generates an additional variable, `IF_<Suffix>`, for unenhanced outports of conditionally executed subsystems if these are preceded by one of the following blocks:

- Data Type Conversion
- (Matrix) Concatenate
- Permute Dimensions
- Reshape
- Selector
- Bus Assignment
- Zero Order Hold
- Rate Transition

Code comments for optimization

For improved readability, TargetLink's code comments for optimization changed. The chain `A replaced by … replaced by … X` is now replaced by `A replaced by X`:

TargetLink ≤ 4.0	TargetLink 4.1
/* Gain: foo/Gain **Variable 'Sa1_Gain' replaced by 'Aux_f'** **Variable 'Aux_f' replaced by 'Aux_F32_e' */**	/* Gain: foo/Gain **Variable 'Sa1_Gain' replaced by 'Aux_F32_e' */**

No access functions for auxiliary variables

You can no longer define access function templates (AFTs) for auxiliary variables that result from access functions. The former behavior created more access functions than desired or even caused a near-infinite loop during code generation.

New CTO avoidance macros for additions and subtractions

To improve compliance with MISRA-C, TargetLink's Fixed-Point Library provides new macros for additions and subtractions for operands ≤ 16 bit. These macros are generated only when you suppress CTO code patterns (via the ExploitComputeThroughOverflow Code Generator option). The macro names always end with `_PROT` and contain an `I16` or `U16`.

TargetLink ≤ 4.0	TargetLink 4.1
`C__I32ADDI32U32_PROT((Int32) Sa1_I16InPort, Sa1_U32InPort)`	`C__I32ADDI16U32_PROT(Sa1_I16InPort, Sa1_U32InPort)`

Writing dereferences in logical operations in parentheses

Dereferences in logical operations are now written in parentheses to comply with MISRA-C.

TargetLink ≤ 4.0	TargetLink 4.1
`Sa1_OutPort1 = (Int16) (*In2 || *In1);`	`Sa1_OutPort1 = (Int16) ((*In2) || (*In1));`

MinMax block code pattern

For MinMax blocks with more than two inputs or an input which has more than two signals, the code pattern changes when the first two inputs or signals do not fit in the output. This can mean one of the following:

- The order of the comparison changes.
- The output is initialized with minimum or maximum values before comparing it with each signal.

Status of RTE_Invalidate

TargetLink can now evaluate the return value of the `Rte_Invalidate` RTE API function. The following table shows a code sample of a `Rte_Invalidiate` call for a SenderComSpec block with enabled Invalidate and Status ports:

TargetLink ≤ 4.0	TargetLink 4.1

`if(InvalidateCondition) {`
`   Rte_Invalidate_x_y();`
`   `**`status = 0;`**
`}`
`else {`
`   status = Rte_Write_x_y();`
`}`
	`if(InvalidateCondition) {`
	`   `**`status = Rte_Invalidate_x_y();`**
	`}`
	`else {`
	`   status = Rte_Write_x_y();`
	`}`

Changed subsystem naming for incremental code generation

The code of subsystems configured for incremental code generation that were converted from referenced models might now be stored at a different location than in previous versions. For details, refer to *Various Migration Aspects* on page 242.

Base types for logging variables

If an auxiliary variable is created for a Sink block for logging, it always gets a TargetLink base type (in AUTOSAR: a platform type). However, logging code is not production code.

Abs pattern changes

For better readability, condensed fixed-point Abs patterns are no longer supported. Instead, if-else expressions are used (e.g., an unscaled `Int16` Abs with unscaled `Int16` input).

TargetLink ≤ 4.0	TargetLink 4.1

`Sa1_OutPort = Sa1_InPort;`
`if (Sa1_InPort < 0) {`
`   Sa1_OutPort = -Sa1_OutPort;`
`}`
	`if (Sa1_InPort >= 0) {`
	`   Sa1_OutPort = Sa1_InPort;`
	`}`
	`else {`
	`   Sa1_OutPort = (Int16) (-Sa1_InPort);`
	`}`

The following changes might apply:

- Subsequent code patterns might change because another (usually better) range information is inherited: e.g., after the paragraph is followed by a sign, its negative branch is omitted, because the output of the Abs is always a positive value.
- With saturation, additional 64-bit operations are generated when `KeepSaturationStatements` is set.
- In Stateflow expressions like `FloatOutVar = abs (IntVar)`, another code pattern is generated (with the `?` operator).

Additional casts in nonscalar AUTOSAR or indirect function reuse

Whenever a pointer with indices on the left side is accessed, there might be additional casts that now better conform with TargetLink's general casting style:

TargetLink ≤ 4.0	TargetLink 4.1
`pISV->pISV_SL1_0_tp->pSL1_ImplicitOut1[Aux_S32] = 0;`	`pISV->pISV_SL1_0_tp->pSL1_ImplicitOut1[Aux_S32] =` **`(sint16)`** `0;`

In this particular case, the cast is created because the ground symbol was originally on the right side.

Overflow-free, unary minus

The following example shows code for an Abs block with `Int16` input and `UInt16` output:

TargetLink ≤ 4.0	TargetLink 4.1

`if(I16In >= 0) {`
`   …`
`} else {`
`   U16Out = (UInt16)(-I16In)`
`}`
This pattern is critical, because the value -32768 (INT16MIN) might cause an undefined overflow. 32768 does not fit in a 16-bit-platform `int` and the minus is calculated in `int`.
	`if(I16In >= 0) {`
	`   …`
	`} else {`
	`   if (I16In == -32768) {`
	`      U16O7ut = 32768;`
	`   } else {`
	`      U16Out = (UInt16)(-I16In)`
	`   }`
	`}`
	Now the smallest negative value is used.

The second *if-else* might be replaced by a *?* operator.

```
… (I16In == -32768) ? 32768 : (UInt16)(-I16In)
```

This is always the case when the unary minus is not the last operation on the right side of an assignment. A typical example is an Abs block with a different input/output scaling. The minus is then the operand of the rescaling (shift/division). As a result, the `?` operator is used.

The following expressions and blocks can be affected:

- Stateflow expressions with unary minus
- Abs blocks
- Gain blocks with negative gain values
- Product blocks with a negative constant default
- Sum blocks with the corresponding settings

Saturation macros:FIT macro calls replace SAT macro calls

There are two kinds of saturation macros in TargetLink's Fixed-Point Library:

- FIT macros that perform saturation on range limits (implemented range)
- SAT macros that perform saturation on user-defined limits (e.g., in the Saturation block or in Stateflow)

Prior to TargetLink 4.1, in rare cases a SAT macro was called although a saturation on range limits was performed. This is now changed for consistency and for improved MISRA-C compliance.

The following example shows saturation in Stateflow with

```
UInt16 U16Out ; // LSB = 0.002 and checkmax = 1

U16Out = U16Out + I8In;
```

TargetLink ≤ 4.0	TargetLink 4.1

`Aux_I32 = …`
`U16Out = C__U16SATI32_SATb(Aux_I32, 65535 /* 131.07 */, 0)`
	`Aux_I32 = …`
	`U16Out = C__U16FITI32_SAT(Aux_I32, 65535 /* 131.07 */ )`

Saturation macros

When calling SAT macros that are generated for the Saturation block or Stateflow, the type of the limits (parameter 2,3) is adjusted to match the type of the output if the limits have a variable class that is different from the default.

TargetLink ≤ 4.0	TargetLink 4.1

`GLOBAL Int32 Sb1_Saturation_lower = 0;`
`GLOBAL Int32 Sb1_Saturation_upper = 100;`
 	
`Int16 Aux_S16;`
`Int16 Aux_S16_a;`
 	
`Aux_S16 = (Int16) Sb1_Saturation_upper;`
`Aux_S16_a = (Int16) Sb1_Saturation_lower;`
 	
`Sb1_OutPort = C__U16SATI16_SATb(Sb1_InPort, Aux_S16,`
`Aux_S16_a);`
	`GLOBAL Int32 Sb1_Saturation_lower = 0;`
	`GLOBAL Int32 Sb1_Saturation_upper = 100;`
	
	`UInt16 Aux_U16;`
	`UInt16 Aux_U16_a;`
	
	`Aux_U16 = (UInt16) Sb1_Saturation_upper;`
	`Aux_U16_a = (UInt16) Sb1_Saturation_lower;`
	
	`Sb1_OutPort = C__U16SATI16_SATb(Sb1_InPort,Aux_U16,`
	`Aux_U16_a);`

Accessing matrix variables within loops

For code efficiency, accessing matrix variables within loops can now be replaced by scalar variables. In certain cases, matrix and vector variables are replaced by one or more scalar variables, even if they are accessed from outside the loops.

TargetLink ≤ 4.0	TargetLink 4.1

`Int16 vec[…];`
`vec[0] = …`
`   … = v[0];`
 	
`loop (i = 1:n) {`
`   v[i] = …`
`      = v[i];`
`}`
	`Int16 scalar;`
	`scalar = …`
	`   … = scalar;`
	
	`loop (i = 1:n) {`
	`   scalar = …`
	`      … = scalar;`
	`}`
	Or, if more scalar variables are introduced:
	`Int16 scalar1;`
	`Int16 scalar2;`
	
	`scalar1 = …`
	`   … = scalar1;`
	
	`loop (i = 1:n) {`
	`   scalar2 = …`
	`      … = scalar2;`
	`}`

Accessing matrix variables outside of loops

Accessing matrix variables from outside the loops can now be replaced by one or more scalar variables. The combination of accesses within and outside of loops can also now be optimized. In addition, unknown accesses for indices can also be optimized if the index expression accesses is built up identically.

During replacement by scalar variables, memory savings typically emerge (in the aggregate). However, in some situations it is not possible to reduce memory consumption: i.e., the new scalar variables use as much memory as the initial multidimensional variable.

This takes place only for completely unrolled code, precisely if:

- (For a vector variable) `Number of elements < LoopUnrollThreshold`
- (For a matrix variable) `Number of elements < LoopUnrollThreshold`^{2}

Accesses from ...	TargetLink ≤ 4.0	TargetLink 4.1

Outside of loops, analog for vectors	M[0][0] = …;
	…
	… = M[0][0];
		Aux = …;
		…
		… = Aux;

Within and outside (of partially unrolled) loops, analog for vectors	M[0][0] = …;
	loop {
	   M[1][i] = …
	   M[2][i] = …
	      …
	      … = M[1][i];
	      … = M[2][i];
	}
		Aux = …;
		loop {
		   Aux_a = …
		   Aux_b = …
		      …
		      … = Aux_a;
		      … = Aux_b;
		}

Outside of loops, analog for matrices	V[<expr>] = …; … … = V[<expr>];	Aux = …; … … = Aux;

Tangents code patterns

The tangents code pattern from within Stateflow and for the TargetLink Trigonometric and Math blocks was modified.

Tangents within Stateflow Up to TargetLink 4.1, saturation was erroneously omitted, especially for saturated expressions.

Now the following rules apply:

- If a tangent is used in an assignment such as `out = tan(expr)`, then the tangent is calculated in fixed-point when `out` has a fixed-point type.
- In all other cases, e.g., complex expressions, TargetLink performs tangent calculations in floating-point. If a pure fixed-point context is detected during code generation, an error message occurs.

Tangents for the Trigonometric and Math blocks Now, unnecessary saturations are omitted by default if the result is either 32-bit or 16-bit with LSB >= 2^{-14}. Therefore, *Omitted Saturation* comments are dropped in some cases.

Assignments to bitfields

Assignment of Stateflow state IDs If Use bitfields in state machines = On and there is a multi-valued state variable, casts to `unsigned int` are now superfluous for Stateflow state IDs:

TargetLink ≤ 4.0	TargetLink 4.1
`SIBFS_Chart_a.Ca1_Chart_ns = (unsigned int) Ca5_OFF_id;`	`SIBFS_Chart_a.Ca1_Chart_ns = Ca5_OFF_id;`

In addition, the class default settings of Stateflow state IDs has changed. Stateflow state IDs become global macros in the generated code without the initial value being cast.

Bitfield semantic for genuine bitfields TargetLink evaluates whether a bitfield is a Boolean bitfield (UseGlobalBitfieldsForBooleans = On) or a genuine bitfield (Base type = Bitfield). For TargetLink versions ≤ 4.0, a bool semantic is applied to both, Boolean bitfields and genuine bitfields. If required, `!= 0` is added. As of TargetLink version 4.1, a bitfield semantic is applied to genuine bitfields:

TargetLink ≤ 4.0	TargetLink 4.1

`BooleanBitfield = (Int16Var != 0)`
`GenuineBitfield = (Int16Var != 0)`
	`BooleanBitfield = (Int16Var != 0)`
	`GenuineBitfield = (Int16Var & 1)`

The bitfield semantic can bealso applied to constants, for example:

```
GenuineBitfield = 3 & 1
```

Code optimization is feasible only if a constant equals the values 0 or 1.

Immediate assignment to bitfields An immediate assignment to a bitfield is generated if the right side is one of the following:

- A bitfield
- A bool expression
- A & 1 operation
- A Stateflow state ID macro

> For the assignment of floating-point or scaled operands to genuine bitfields, TargetLink no longer generates `!= 0` but returns an error.

Better Z/N values for rescaling

The algorithm for calculating the Z and N values has been improved. This leads to code that is either more precise or more efficient with calculations based on smaller bit widths:

TargetLink ≤ 4.0	TargetLink 4.1
`Sb1_PRODUCT = (Int16) (((Int32) ((((Int32) Op1) * ((Int32) Op2)) << 2)) / 39);`	`Sb1_PRODUCT = (Int16) (((Int32) ((((Int32) OP1) * ((Int32) Op2)) << 6)) / 625);`

Parameter tolerance is now more frequently used to increase code efficiency. This means that a parameter tolerance greater than 0 can result in code that is less precise. If required, you can change this by reducing the parameter tolerance.

Comparison of UInt32 and Int32 variables

TargetLink now checks if the signed variable is negative and can avoid 64-bit macros when comparing UInt32 and Int32 variables:

<table>
<tr><th>TargetLink ≤ 4.0</th><th>TargetLink 4.1</th></tr>
<tr><td>C__I64COPYU32(U32Var, I64Var_hi, I64Var_lo);
C__I64COPYI32(I32Var, I64Var2_hi, I64Var2_lo);
if(C__LE64(I64Var_hi, I64Var_lo, I64Var2_hi, I64Var2_lo))</td><td>For <=, <, and !=:
if(I32Var < 0 || ((UInt32)I32Var <op> U32Var))
For >, >=, and ==:
if(I32Var >= 0 && ((UInt32)I32Var <op> U32Var))</td></tr>
</table>

As an effect, superfluous logical branches can be eliminated from the generated code by subsequent optimization.

Comparison of 64-bit variables

When comparing two 64-bit variables that both have an internally calculated worst-case range of ≤ 32-bit, TargetLink now also considers the upper 32 bits. For example, Int64Var could become negative:

TargetLink ≤ 4.0	TargetLink 4.1
if(Int64Var_lo < Int64Var2_lo)	if(C__LT(Int64Var_hi, Int64Var_lo, Int64Var2_hi, Int64Var2_lo))

This change is particularly relevant for the Discrete-Time Integrator block.

Elimination of intermediate variables

TargetLink can now perform intermediate variable elimination in such a way that a computation followed by an index selection is no longer performed for the complete width, only for the desired element.

TargetLink ≤ 4.0	TargetLink 4.1
VectorVar[<iteration range>] = expression(<iteration range>) … = VectorVar[cnst]	… = expression(cnst)

The following conditions must be met:

- VectorVar[cnst] is the only usage of VectorVar[<iteration range>].
- Index cnst is a subset of the iteration range. This is fulfilled if <iteration range> covers all elements.

Assignment blocks in For Iterator subsystems

If an Assignment block resides in an atomic subsystem that resides in a For Iterator subsystem, the following applies:

If the block property Omit dispensable initializations is set to off, the initialization via the Y0 signal is performed only during the first sample step. This is in accordance with the Simulink behavior.

Division-by-zero check for floating-points

TargetLink improves compliance with MISRA-C:

No superfluous rescaling TargetLink avoids irrelevant rescaling:

TargetLink ≤ 4.0	TargetLink 4.1

if (((Float32) Sa1_ScaledDenom) * 0.25F) != 0) {
   Sa1_OutPort1 = Sa1_F32Num_ / (((Float32) Sa1_ScaledDenom)
* 0.25F);
} else { …
	if (Sa1_ScaledDenom != 0) {
	   Sa1_OutPort = Sa1_F32Num / (((Float32)
	Sa1_ScaledDenom) * 0.25F);
	} else { …

Auxiliary variables for offsets TargetLink uses auxiliary variables:

TargetLink ≤ 4.0	TargetLink 4.1

if (((((Float32) Sa1_ScaledDenomWithOffset) * 0.25F) +
42.F) != 0) {
 	
   Sa1_OutPort1 = Sa1_F32Num_ / ((((Float32)
Sa1_ScaledDenomWithOffset) * 0.25F) + 42.F);
 	
} else { …
	Aux_F32 = (((Float32) Sa1_ScaledDenomWithOffset) *
	0.25F) + 42.F;
	
	if (Aux_F32 != 0.F) {
	   Sa1_OutPort1 = Sa1_F32Num_ / Aux_F32;
	} else { …

Subtractions with unsigned result

For TargetLink ≤ 4.0, subtractions with an unsigned result were turned into additions: e.g., `U8Var = U8Var2 + 255;`.

For improved readability with TargetLink 4.1, a subtraction remains a subtraction if the constant fits the result type, `U8Var = U8Var2 - 1;`. Calculations might be performed in smaller types.

TargetLink ≤ 4.0	TargetLink 4.1
U8Var = U8Var2 + 255;	U8Var = U8Var2 - 1;

Predefined AUTOSAR variable classes

To suppress additional `#include tl_defines.h` directives, the predefined AUTOSAR variables were changed: When updating existing DD files, TargetLink sets the UseName property of predefined AUTOSAR VariableClass objects to `off` if they reference a DD AccessFunctionTemplate object.

This results in a different declaration or definition of variables that have these variable classes:

TargetLink ≤ 4.0	TargetLink 4.1
extern **EXPLICIT_IRV** S16_LinPos Rte_Irv_Controller_LinPos	extern S16_LinPos Rte_Irv_Controller_LinPos

This might cause missing `#include tl_defines.h` directives.

Code pattern for additions and subtractions without CTO

To increase efficiency and MISRA-C compliance, the code pattern of additions and subtractions can change if all of the following conditions are true:

- The ExploitComputeThroughOverflow Code Generator option is set to `never`.
- The operands of the operation are unsigned.
- The operands of the operation have fewerl than 32 bits.

TargetLink ≤ 4.0	TargetLink 4.1
UInt8Var = (UInt8)(**(Int16)**UInt8Var + **(Int16)**UInt8Var)	UInt8Var = (UInt8)(UInt8Var + UInt8Var)
UInt8Var = (UInt8)(**(Int16)**UInt8Var - **(Int16)**UInt8Var)	UInt8Var = (UInt8)(UInt8Var - UInt8Var)

Definition of Data Store Memory block variables

To increase consistency and user control, the definitions of Data Store Memory block variables are now always made in the module that the step function of the subsystem that contains the Data Store Memory block is generated in.

Accordingly, the definitions of Data Store Memory block variables might occur in a different header file if all of the following conditions are true:

- The Data Store Memory block is placed in a subsystem that also contains a nested subsystem.
- The nested subsystem contains a Data Store Read or Data Store Write block that accesses the data store memory.
- The step functions of the subsystems are generated in different modules:
  - The step function of the subsystem containing the Data Store Memory block is generated into module A.
  - The step function of the subsystem containing the Data Store Read or Data Store Write block is generated into module B.

TargetLink ≤ 4.0	TargetLink 4.1
In certain modeling situations, TargetLink might have defined the block variable of the Data Store Memory block in module B.	TargetLink now always defines the block variable of the Data Store Memory block in module A.

Rounding of doubles in Float32 comparisons

TargetLink's rounding behavior changed for literal values in Float32 comparison. This is due to the following reasons:

- Code is easier to understand because the value is used in the same way as specified in the model.

- More user control:
  - You can specify the rounded value in the model.
  - You can use C compiler options to control the rounding and keep it consistent between TargetLink-generated production code and legacy code.

In the following example, the value `0.1` is compared with a variable whose data type is `Float32`:

```
0.1 > Float32Var
```

The following table shows TargetLink's rounding behavior:

TargetLink ≤ 4.0	TargetLink 4.1
Sa1_Relational_Operator = **0.1000000015F** > Sa1_F32In;	Sa1_Relational_Operator = **0.1F** > Sa1_F32In;

Double precision for Float32 values

For improved traceabilty between model and production code and also because of the above stated rounding control, TargetLink now generates literal values for `Float32` with double precision and by adding the `F` suffix. The following table shows an example of a `Float32` gain with `GainValue = pi`:

TargetLink ≤ 4.0	TargetLink 4.1
Sa1_OutPort = Sa1_InPort * 3.141592654F;	Sa1_OutPort = Sa1_InPort * 3.1415926535897931F;

This might cause more decimal places being printed in the generated code if the value was not exactly given in float/single precision in the model, as it was in TargetLink ≤ 4.0.

Improved code efficiency

With TargetLink 4.1, code efficiency has been improved. This might also cause changes in the generated code, in comparison to older TargetLink versions. For details, refer to *Improved Code Efficiency* on page 196.

Other

Various Migration Aspects

TargetLink Main Dialog block

The Compare with reference command has been removed from the context menu of the code generation units listed on the block's Code Generation page.

Converting referenced models

TargetLink's behavior when converting referenced models has changed.

Referenced model to incremental subsystem When converting a referenced model to a subsystem configured for incremental code generation, the created subsystem is named as shown in following table:

Referenced Model Reused?	< TargetLink 4.1	TargetLink 4.1
Yes	-	Subsystem name = Model block name
No	Subsystem name = Model block name	Subsystem name = model name

Accordingly, there is a new storage location for the code that is generated for the model configured for incremental code generation:

Referenced Model Before Conversion (TL4.0 and TL4.1)	Converted Incremental Subsystem (TL4.0)	Converted Incremental Subsystem (TL4.1)
Generated code modules: ■ <ModelName>.c[1] ■ <ModelName>.h[1] Name of the DD Subsystems object: <ModelName>	Generated code modules: ■ <ModelBlockName>.c[1] ■ <ModelBlockName>.h[1] Name of the DD Subsystems object: <ModelBlockName>	Generated code modules: ■ <ModelName>.c[1] ■ <ModelName>.h[1] Name of the DD Subsystems object: <ModelName>

[1] If you did not directly specify a module name at the model's Function block.

Incremental subsystem to referenced model TargetLink sets the simulation mode of the referenced model as shown in the following table:

< TargetLink 4.1	TargetLink 4.1
Accelerator	If the subsystem was converted from a referenced model in TargetLink 4.1, the simulation mode of the reconverted referenced model is the same as the simulation mode of the original referenced model. If the subsystem was converted with another TargetLink version or was not converted at all, the simulation mode of the referenced model is set to `Normal`.

Related documentation

- **`tl_refmodel_to_subsystem(propertyName, propertyValue, ...)`** (📖 *TargetLink API Reference*)
- **`tl_subsystem_to_refmodel(propertyName, propertyValue, ...)`** (📖 *TargetLink API Reference*)

Explicitly modeled saturation involving Float32 and fixed-point

TargetLink cannot guess your intent if you model saturation involving fixed-point or Float32 without using TargetLink's Saturation block (e.g., by using the MinMax block instead). In such cases, keep in mind to check the rounding effects of double values to single precision in Float32 comparisons by the C compiler. If needed as an alternative, either use the Saturation block or specify the correctly rounded single precision value already in the model.

Note that TargetLink's block output code saturation is not affected, because TargetLink internally calculates the correctly rounded single precision value as needed. The changes relate to user-specified values. Also refer to *Rounding of doubles in Float32 comparisons* on page 240.

Obsolete

Obsolete Limitations

With TargetLink 4.1, the following limitations of previous TargetLink versions were removed:

General limitations

Initialization of buses[1]
Bus signals can be initialized only if one of the following conditions is met: ■ All the bus signals are enumerations of the same type. ■ All the bus signals are non-enumerations. Basically, the same scalar initial value has to be applied to all the bus signals.

[1] This limitation became obsolete because TargetLink 4.1 supports Simulink's simplified mode.

Block-specific limitations

Look-up tables (vectorized)
Table look-up functions are replaced by user-provided implementations if a matching custom look-up script is available. This applies to look-up table blocks with a scalar output. The custom look-up script mechanism does not support vector/matrix input signals or uniform elements.

Merge block
It is not possible to specify the initial value of bus signal elements by setting the initial output parameter of blocks that pass the bus signal to a Simulink Initial Condition Structure.

OutPort block
It is not possible to specify the initial value of bus signal elements by setting the initial output parameter of blocks that pass the bus signal to a Simulink Initial Condition Structure.

Rate Transition block
It is not possible to specify the initial value of bus signal elements by setting the initial output parameter of blocks that pass the bus signal to a Simulink Initial Condition Structure.

Component-based development limitations

Function reuse
TargetLink does not support multiple references to the same model inside one model hierarchy. Therefore, code generated for referenced models cannot be reused.

Simulink model arguments
TargetLink does not support Simulink model arguments for referenced models.

Code generation limitations

Incremental code generation
Incrementally generated subsystems or referenced models cannot be subject to function reuse and cannot reside in reused functions.

Changes in Future TargetLink Versions

Where to go from here

Information in this section

Features to Be Discontinued	245
API Functions to Be Discontinued	246

Features to Be Discontinued

A2L import

It is planned to discontinue the A2L import in a future TargetLink version.

Generation of RTF documents

It is planned to discontinue the option to generate documentation in rich text format (RTF) in a future TargetLink version.

MISRA-C:2004 Compliance Documentation document

It is planned to discontinue the MISRA-C:2004 Compliance Documentation document in a future TargetLink version. TargetLink users will then be advised to use the MISRA-C:2012 Compliance Documentation instead.

Simulink's classic initialization mode

It is planned to discontinue support for Simulink's classic initialization method (Underspecified initialization detection parameter set to `Classic`) in a future TargetLink version.

Dynamic components

It is planned to discontinue support for specifying dynamic components for DD Variable objects in a future TargetLink version.

Code generation for special OSEK versions

It is planned to discontinue the code generation for special OSEK versions, such as OsCan in a future TargetLink version.

Signal logging format

It is planned to discontinue the support for Simulink's logging method `ModelDataLogs` (Signal logging format parameter) in a future TargetLink version.

Fully built V-ECUs as OSA

Future versions of TargetLink can no longer build V-ECUs as OSA files. TargetLink will then only export the generated production code as V-ECU implementations (CTLGZ files) and leave the platform-specific build process to VEOS for offline simulation and ConfigurationDesk for real-time simulation.

API Functions to Be Discontinued

Discontinued API functions

The following API functions are deprecated and will be removed in a future TargetLink version:

Function	Deprecated Since	Replacement Function
`tl_adapt_dd_references`	TargetLink 4.0	**`tlMoveDDObject`**
`tl_extract_subsystem`	TargetLink 4.0	**`tlExtractSubsystem`**
`tl_find_dd_references`	TargetLink 4.0	**`tlFindDDReferences`**
`tl_get_blockset_mode`	TargetLink 4.0	**`tlOperationMode`**
`tl_sim_interface`	TargetLink 4.0	**`tlSimInterface`**
`tl_switch_blockset`	TargetLink 4.0	**`tlOperationMode`**
`tl_upgrade`	TargetLink 4.0	**`tlUpgrade`**

See the help contents on the new API functions to adjust your user scripts accordingly.

VEOS

Where to go from here

Information in this section

New Features of VEOS 3.5	247
Migrating to VEOS 3.5	250

New Features of VEOS 3.5

Information in this topic

More intuitive user interface on page 247

Enabling/Disabling the generation of debug information (MSVC, GCC) on page 249

Accessing call stack information in case of an exception on page 249

Build Output dialog on page 249

Loading an offline simulation application when an application is already running on page 250

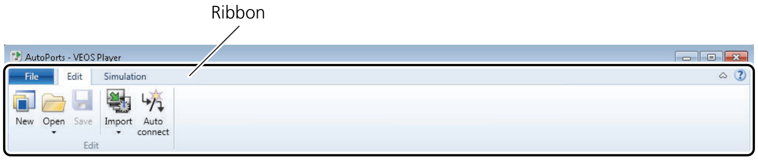
Unloading an offline simulation application on page 250

FMU import: Support of enumerations on page 250

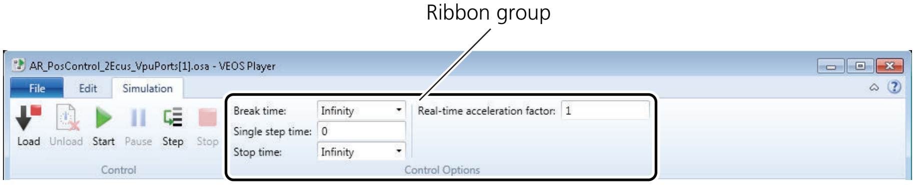
More intuitive user interface

The user interface of the VEOS Player is more intuitive: Its menu bar and toolbar have been replaced by *ribbons* and the *Backstage view* used in Microsoft Office, etc.

Ribbons VEOS Player's ribbons organize and group commands that belong together. They are located at the top of the user interface. Refer to the following illustration:



Each ribbon has *ribbon groups*, each of which provides a set of related commands:



Break time: Infinity

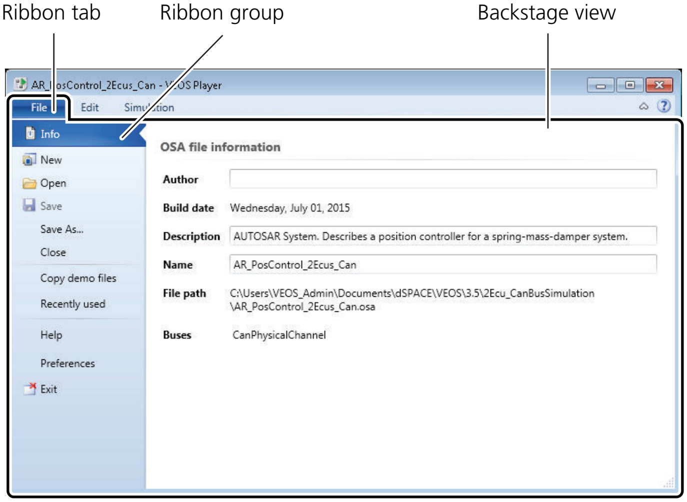
Single step time: 0

Stop time: Infinity

Real-time acceleration factor: 1

Backstage view VEOS Player's *Backstage view* provides basic commands, for example, for opening and saving OSA files. It also provides quick access to the recently used OSA files.

The following illustration shows the Backstage view with the Info ribbon group as an example:



OSA file information

Author

Build date Wednesday, July 01, 2015

Description AUTOSAR System. Describes a position controller for a spring-mass-damper system.

Name AR_PosControl_2Ecus_Can

File path C:\Users\VEOS_Admin\Documents\dSPACE\VEOS\3.5\2Ecu_CanBusSimulation \AR_PosControl_2Ecus_Can.osa

Buses CanPhysicalChannel

Context-sensitive help VEOS Player now provides *context-sensitive* help.

Press the **F1** key or click the Help button in the VEOS Player to get help on the currently active context.

For details, refer to *Basics on Ribbons* (📖 *VEOS Guide*).

Enabling/Disabling the generation of debug information (MSVC, GCC)

Up to and including VEOS 3.4 Up to and including VEOS 3.4, source code debugging:

- was *always enabled* when you imported items such as a V-ECU implementation together with the MSVC compiler, for example.
- was *always disabled* when you imported items such as a V-ECU implementation together with the GCC compiler, for example.

As of VEOS 3.5 As of VEOS 3.5, you can *enable and disable source code debugging* together with both the MSVC and the GCC compiler.

Refer to *Basics on Debugging Source Code in an Offline Simulation* (📖 *VEOS Guide*).

Accessing call stack information in case of an exception

When an exception of the model code of a specific VPU process (based on an FMU or SIC) occurs during offline simulation, VEOS outputs the following call stack information in an error message:

- The name of the function causing the exception
- The name and location of the source code file containing the function
- The call stack related to the exception

VEOS provides the call stack only if both conditions apply:

- The MSVC compiler is selected
- Source code debugging was enabled for the VPU's build process

Build Output dialog

When you import and build, for example, a V-ECU implementation, the VEOS Player now provides the following information in the separate Build Output dialog:

- Information on the build process, such as compiler and linker messages
- A history of all the error and warning messages that occur during the build process

The Build Output dialog is displayed also when you perform the import and build via the automation interface of the VEOS Player.

For instructions, refer to *How to Import VPUs, V-ECU Implementations, Simulink Implementations, and FMUs* (📖 *VEOS Guide*).

Loading an offline simulation application when an application is already running

When you load an offline simulation application to VEOS when an offline simulation application is already running, VEOS now asks you whether to:

- Connect to the running offline simulation application
- Replace the running offline simulation application

Refer to *Load* (📖 *VEOS Player Reference*).

Unloading an offline simulation application

The VEOS Player now lets you unload an offline simulation application from VEOS. This lets other products such as ControlDesk Next Generation load another offline simulation application.

Refer to *Unload* (📖 *VEOS Player Reference*).

FMU import: Support of enumerations

Enumeration data types of FMU inputs and outputs are now supported. The VEOS Player now creates VPU ports for these inputs/outputs.

Migrating to VEOS 3.5

Compatibility overview

VEOS and OSA The following table shows the compatibility between VEOS 3.5 and OSA files:

OSA Files Created with Products of...	OSA Version
dSPACE Release 2014-A	3.2
dSPACE Release 2014-B	3.3
dSPACE Release 2015-A	3.4
dSPACE Release 2015-B	3.5

- An OSA file created or modified with VEOS 3.5 cannot be loaded in earlier VEOS versions.
- There is a migration issue related to the simulation of ASM models. Refer to *Migrating ASM models* (📖 *VEOS Guide*).

VEOS and CTLGZ (V-ECU implementation) The following table shows the compatibility between VEOS 3.5 and CTLGZ files (V-ECU implementations):

V-ECU Implementations Created with Products of...	V-ECU Implementation Version
dSPACE Release 2013-B and earlier: ■ SystemDesk 3.x ■ TargetLink 3.5	1.0

V-ECU Implementations Created with Products of...	V-ECU Implementation Version
dSPACE Release 2014-A: ■ SystemDesk 4.2	2.0
dSPACE Release 2014-B: ■ SystemDesk 4.3 ■ TargetLink 4.0	2.1
dSPACE Release 2015-A: ■ SystemDesk 4.4	2.2
dSPACE Release 2015-B: ■ SystemDesk 4.5 ■ TargetLink 4.1	2.3

VEOS and SIC VEOS 3.5 is compatible with Simulink implementation container (SIC) files created with Model Interface Package for Simulink 3.1 from dSPACE Release 2015-B (Simulink Implementation Container Version 1.0.1).

> There is a migration issue related to the simulation of ASM models. Refer to *Migrating ASM models* (📖 *VEOS Guide*).

VEOS and FMU FMUs must comply with the FMI 2.0 standards for Co-Simulation. FMUs complying with the FMI 1.0 standard are not supported.

For an overview of dSPACE products and releases that support FMI, refer to:

http://www.dspace.com/go/FMI-Compatibility.

Discontinuation of the dSPACE Target for Offline Simulation and migration

Model Interface Package for Simulink The *dSPACE Target for Offline Simulation* was delivered for the last time with dSPACE Release 2015-A. To prepare a Simulink® model for offline simulation with VEOS, use the *Model Interface Package for Simulink*, which was introduced with dSPACE Release 2015-A.

The Model Interface Package for Simulink provides the `dsrt.tlc` system target file, which lets you generate a Simulink implementation container (SIC) file from a Simulink® model. The SIC file is independent of the simulation platform, so that you can integrate the same SIC file in an offline simulation application for VEOS and in a real-time application for SCALEXIO.

For details on the differences in the workflows of *dSPACE Target for Offline Simulation* and *Model Interface Package for Simulink*, refer to *Differences Between DsOffSim and the Model Interface Package for Simulink* (📖 *Model Interface Package for Simulink - Modeling Guide*).

New workflow The workflow that the *Model Interface Package for Simulink* provides for preparing a Simulink® model for offline simulation is different from the workflow provided by the *dSPACE Target for Offline Simulation*.

To prepare the offline simulation of a Simulink® model, perform the following steps:

1. In the Simulink® model, specify the `dsrt.tlc` system target file provided by the Model Interface Package for Simulink.
2. Generate code for the Simulink® model.

   Unlike the *dSPACE Target for Offline Simulation*, the *Model Interface Package for Simulink* does not generate an application for a specific simulation platform. Instead, it generates a Simulink implementation container (SIC) file containing Simulink® model code.

   For instructions, refer to *Generating Simulink Implementation Containers* (📖 *Model Interface Package for Simulink - Modeling Guide*).
3. Import the SIC file to the VEOS Player to integrate the model in an offline simulation application for VEOS.

   The VEOS Player builds the SIC file for the VEOS simulation platform.

   For instructions, refer to *How to Import VPUs, V-ECU Implementations, Simulink Implementations, and FMUs* (📖 *VEOS Guide*).

Changed automation behavior when importing a model implementation to the VEOS Player

In VEOS 3.5, there is an incompatible change of the `Import` method of the `IProject <<Interface>>` interface of the VEOS Player API. The method lets you import model implementations such as SIC and CTLGZ files to a VEOS Player project:

- Up to and including VEOS 3.4, the `Import` method provides a list of all compiler messages as a return parameter of the `<System.String[]>` data type.
- As of VEOS 3.5, the `Import` method provides build information via the `IBuildResult <<Interface>>`.

Compatibility Information

Where to go from here

Information in this section

Supported MATLAB Releases	254
Operating System	255
Run-Time Compatibility of dSPACE Software	257
Limitations for 64-Bit Windows Operating Systems in Combination with 32-Bit dSPACE Software	258
Products on the 64-Bit dSPACE DVD Set and Their MATLAB Support	259
General Limitations for Windows 7	261

Supported MATLAB Releases

Supported MATLAB releases

MATLAB Release...	...Is Supported by dSPACE Release 2015-B				
	RCP and HIL Software	AutomationDesk 5.1 [1]	TargetLink 4.1	Model Compare 2.6	dSPACE Python Extensions 2.0 [2]
R2015b (64-bit)	✓[3]	✓	✓	✓	✓
R2015b (32-bit)	–	–	–	–	–
R2015a (64-bit)	✓	✓	✓	✓	✓
R2015a (32-bit)	–	–	–	–	–
R2014b (32-bit and 64-bit)	✓[3]	✓	✓	✓	✓
R2014a (32-bit and 64-bit)	✓[3]	✓	✓	✓	✓

[1] AutomationDesk's MATLAB Access library requires MATLAB.

[2] matlablib2 of dSPACE Python Extensions requires MATLAB.

[3] R2014a (32-bit), R2014b (32-bit), R2015b (32-bit) and R2015b (64-bit) are not supported by the RTI FPGA Programming Blockset – FPGA Interface.

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/sw3rdparty.

Operating System

Operating system on host PC

The following operating systems are supported by the dSPACE products on Release 2015-B:

32-Bit dSPACE Software	64-Bit dSPACE Software
■ Windows 7 Professional, Ultimate, and Enterprise with Service Pack 1 (32-bit or 64-bit version) Only the listed editions are supported. The Windows 7 Home and Starter editions are not supported. ■ ControlDesk Next Generation can also be installed on the MicroAutoBox Embedded PC. The operating system depends on the variant as follows: ■ MicroAutoBox Embedded PC with Intel® Atom™ Processor N270: Windows 7 Ultimate, 32-bit version ■ MicroAutoBox Embedded PC with Intel® Core™ i7-3517UE Processor: Windows 7 Professional, Ultimate, and Enterprise, 64-bit version	■ Windows 7 Professional, Ultimate, and Enterprise with Service Pack 1 (64-bit version) Only the listed editions are supported. The Windows 7 Home and Starter editions are not supported.
Notes and Limitations	
■ Refer to *General Limitations for Windows 7* on page 261. ■ Support of 64-bit operating systems: 32-bit dSPACE software supports only the 64-bit version of Windows 7. Other 64-bit operating systems (Windows XP and Windows Vista) are not supported. 32-bit dSPACE software runs under 64-bit Windows operating systems in a WoW64 (Windows-on-Windows 64-bit) subsystem. WoW64 is the x86 emulator of Windows that allows 32-bit Windows-based applications to run seamlessly on 64-bit versions of Windows. This lets you use up to 4 GB of virtual memory for each 32-bit process if the application is prepared for using the large memory area. Otherwise, the virtual address space of a process is limited to 2 GB. Limitations apply when you use 32-bit dSPACE software under a 64-bit Windows operating system. Refer to *Limitations for 64-Bit Windows Operating Systems in Combination with 32-Bit dSPACE Software* on page 258.	■ Refer to *General Limitations for Windows 7* on page 261.

For some complex tasks of the following products, you are required or recommended to use the 64-bit version of Windows 7 as your operating system:

- RTI FPGA Programming Blockset: If you use the FPGA interface of this blockset, Windows 7 (64-bit) is required.
- Automotive Simulation Models: If you use the models for vehicle dynamics, trailer, truck and traffic scenario simulations, Windows 7 (64-bit) is recommended.
- ControlDesk Next Generation: If you use the video-capturing device of ControlDesk Next Generation, Windows 7 (64-bit) is recommended.
- ConfigurationDesk (Implementation Version): If you want to use more than 1000 function blocks (or several function blocks with a total of more than 1000 function ports) in your ConfigurationDesk application, Windows 7 (64-bit) is required.

Allowing communication via 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, and 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="%dspace_root%\MotionDesk\Bin\MotionDesk.exe"
  dir=in action=allow profile=any description="Allow dSPACE
  MotionDesk to receive motion data via network."
  ```

If you are running third-party firewall software on your host PC, ensure that the TCP/IP communication of dSPACE software is not blocked.

Operating system on dSPACE License Server

If you purchased floating network licenses, you have to install and configure one of the networked PCs as the dSPACE License Server.

The operating system of the dSPACE License Server must be one of the following:

- Windows XP Professional (32-bit version) with Service Pack 3
- Windows Vista Business, Ultimate, or Enterprise (32-bit or 64-bit version) with the latest Service Pack
- Windows 7 Professional, Ultimate, or Enterprise (32-bit or 64-bit version) with the latest Service Pack
- Windows Server 2003 (32-bit or 64-bit version)
- Windows Server 2008 R2
- Windows Server 2012, Windows Server 2012 R2

> The dSPACE License Server does not support non-Windows operating systems.

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 2015-B

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

Note that:

- Limitations regarding run-time compatibility in the dSPACE tool chain might occur if products from different dSPACE Releases are mixed.

  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 details on the patch and whether a patch is necessary, refer to http://www.dspace.com/go/CompPatch.

- RCP and HIL software products (on Release 2015-B) cannot be used in combination with RCP and HIL software products from earlier dSPACE releases.

Major limitations for TargetLink and Model Compare The 64-bit version of TargetLink cannot be used in combination with the 32-bit version of Model Compare and vice versa, because you can work only with a bit-compatible MATLAB version (32-bit or 64-bit).

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 further information if you have any questions.

Combining dSPACE products from earlier releases

For more details 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 64-Bit Windows Operating Systems in Combination with 32-Bit dSPACE Software

Objective

Some additional limitations apply when you use 64-bit versions of Windows 7 in combination with 32-bit dSPACE software.

Limitations of device drivers

Third-party bus interfaces (CAN, LIN, or FlexRay) are supported only if they have 64-bit drivers from the manufacturers.

TargetLink: Limitations of target compilers

For information on support for a specific target compiler, contact the respective compiler manufacturer.

MATLAB

If you install a 32-bit variant of MATLAB under Windows 7 (64-bit), the MATLAB installation program generates a message that a 64-bit variant of MATLAB is available. To install the 32-bit variant of MATLAB, click OK.

Products on the 64-Bit dSPACE DVD Set and Their MATLAB Support

Objective

The 64-bit dSPACE DVD set contains the same product versions as the 32-bit dSPACE DVD set. However, the 64-bit DVD set contains the 32-bit variants of some products.

When using the 64-bit DVD set, you should note the limitations described in the section below the table.

Products and their MATLAB support

The following table lists in detail all dSPACE products on the 64-bit dSPACE DVD set and their MATLAB support:

- All MATLAB-related dSPACE products which support 64-bit MATLAB versions
- All 32-bit dSPACE products which also support 64-bit MATLAB versions
- All 32-bit dSPACE products that do not relate to MATLAB

dSPACE Product		Product Supports 64-bit MATLAB	Product Independent of MATLAB Architecture (32-bit/64-bit)	Product Contained as 32-Bit Variant
ControlDesk Next Generation		–	✓	✓
SystemDesk		–	✓	✓
AutomationDesk		✓	–	✓
TargetLink		✓	–	–
Model Compare		✓	–	–
VEOS		–	✓	✓
Real-Time Testing		–	–	✓
Platform API Package	dSPACE Python Extensions	–[1]	✓	✓
	HIL API .NET MAPort	–	✓[2]	✓
	XIL API .NET MAPort	–[3]	✓[3]	✓
Failure Simulation API Package	XIL API .NET EESPort	–	✓	✓

dSPACE Product		Product Supports 64-bit MATLAB	Product Independent of MATLAB Architecture (32-bit/64-bit)	Product Contained as 32-Bit Variant
RCP and HIL software package	RTI and RTI-MP	✓	–	–
	RTI Gigalink Blockset	✓	–	–
	RTI CAN Blockset	✓	–	–
	RTI CAN MultiMessage Blockset	✓	–	–
	RTI LIN MultiMessage Blockset	✓	–	–
	RTI FlexRay Configuration Blockset	✓	–	–
	RTI FPGA Programming Blockset	✓	–	–
	RTI Electric Motor Control Blockset	✓	–	–
	RTI Ethernet Blockset	✓	–	–
	RTI Ethernet UDP Blockset	✓	–	–
	RTI XCP on Ethernet Blockset	✓	–	–
	RTI Watchdog Blockset	✓	–	–
	RTI RapidPro Control Unit Blockset	✓	–	–
	RTI Bypass Blockset	✓	–	–
	RTI USB Flight Recorder Blockset	✓	–	–
	ConfigurationDesk	✓	–	✓
	FlexRay Configuration Blockset	✓	–	–
	FlexRay Configuration Tool	–	✓	✓
	ModelDesk	✓	–	✓
	Automotive Simulation Models	✓	–	–
	MotionDesk	✓	–	✓
	MotionDesk Blockset	✓	–	–
	Flight Rec Data Merger	–	✓	✓

dSPACE Product		Product Supports 64-bit MATLAB	Product Independent of MATLAB Architecture (32-bit/64-bit)	Product Contained as 32-Bit Variant
	Model Interface Package for Simulink	✓	–	–
	Further products of RCP and HIL software package	–	✓	✓

[1] dSPACE Python Extensions contain the matlablib2 Python library. This library supports remote control and access of 64-bit MATLAB. matlablib2 itself is contained on the 64-bit DVD as 32-bit variant.

[2] HIL API .NET MAPort can be used from 32-bit MATLAB via MATLAB Interface for .NET, but cannot be used from 64-bit MATLAB.

[3] XIL API .NET MAPort can be used from 32-bit and 64-bit MATLAB via MATLAB Interface for .NET.

For more details on the compatibility of dSPACE products with 64-bit MATLAB versions, refer to http://www.dspace.com/go/matlab64bit.

Product-specific limitations

Restricted MAT file support ControlDesk Next Generation (ControlDesk 5.5) only supports reading and writing MAT files of file format version 5.0. MAT files of this version can be created in MATLAB by using the `save` command with the option '-v6'.

RTI-MP The `rtimpdiag` command is not functional. This command is based on dSPACE HIL API .NET, which does not support 64-bit MATLAB.

Limitations for the 64-bit variant of TargetLink

Importing an A2L file It is not possible to import an A2L file into a 64-bit variant of TargetLink. However, you can use a workaround described in *Basics of Importing A2L Files* (🕮 *TargetLink Data Dictionary A2L Import and Export*).

General Limitations for Windows 7

Objective

Some limitations apply when you use Windows 7 in combination with dSPACE software.

MATLAB support

For system requirements of MathWorks® software, refer to http://www.mathworks.com/support/sysreq/current_release.

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 procedure of Windows operating systems might cause some required processes to be aborted although they are still being used by dSPACE software. To avoid data loss, it is recommended to terminate the dSPACE software manually before performing a PC shutdown.

User Account Control

It is recommended to disable 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 run with 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

The first time that dSPACE USB devices using cables with optoisolation are connected to the PC, there might be a message that the device driver software was not installed successfully. The dSPACE device will nevertheless work properly later on.

Numerics

64-bit dSPACE DVD
limitations 259

A

ASM Base InCylinder Blockset
migrating 54
new features 54
ASM Diesel Engine Blockset
migrating 57
new features 55
ASM Diesel Exhaust Blockset
migrating 59
ASM Diesel InCylinder Blockset
migrating 61
new features 60
ASM Drivetrain Basic Blockset
migrating 62
new features 62
ASM Electric Components Blockset
migrating 64
new features 63
ASM Engine Gasoline Basic Blockset
migrating 68
new features 67
ASM Engine Gasoline Blockset
migrating 72
new features 70
ASM Environment Blockset
migrating 66
new features 66
ASM Gasoline InCylinder Blockset
migrating 75
new features 74
ASM Optimizer
new features 76
ASM Optimizer Blockset
migrating 77
ASM Traffic Blockset
migrating 79
new features 78
ASM Trailer Blockset
migrating 81
new features 80
ASM Truck Blockset
migrating 83
new features 82
ASM Turbocharger Blockset
migrating 85
new features 84
ASM Vehicle Dynamics Blockset
migrating 88
new features 87
AutomationDesk
new features 49
AUTOSAR
TargetLink-related
migration 226

C

Common Program Data folder 12
CommonProgramDataFolder 12
ControlDesk Next Generation
migration 113
new features 100

D

DCI Configuration Tool
new features 123
Documents folder 12
DocumentsFolder 12
DS1007
new features 154
dSPACE FlexRay Configuration Package
new features 125
dSPACE HIL API .NET
new features 127
dSPACE Python Extensions
new features 129
dSPACE XIL API
new features 133
DVD contents 16

E

ECU Interface Manager
migration 136
new features 135

F

Firmware Manager
new features 139

G

general enhancements and changes 15

H

host PC software
operating system 255

K

key features 24

L

Limitations
TargetLink
obsolete limitations 244
limitations for Windows 64-bit and dSPACE 64-bit software 258
limitations for Windows 7 261
Local Program Data folder 12
LocalProgramDataFolder 12

M

MATLAB
supported releases 254
MicroAutoBox
new features 155
MicroLabBox
new features 153
migrating
ASM Base InCylinder Blockset 54
ASM Diesel Engine Blockset 57
ASM Diesel Exhaust Blockset 59
ASM Diesel InCylinder Blockset 61
ASM Drivetrain Basic Blockset 62
ASM Electric Components Blockset 64
ASM Engine Gasoline Basic Blockset 68
ASM Engine Gasoline Blockset 72
ASM Environment Blockset 66
ASM Gasoline InCylinder Blockset 75
ASM Optimizer Blockset 77
ASM Traffic Blockset 79
ASM Trailer Blockset 81
ASM Truck Blockset 83
ASM Turbocharger Blockset 85
ASM Vehicle Dynamics Blockset 88
migration
ControlDesk Next Generation 113
ECU Interface Manager 136
Model Compare 143
MotionDesk 150
RTI 156
RTI Bypass Blockset 158
RTI CAN MultiMessage Blockset 162
RTI FPGA Programming Blockset 170
RTI LIN MultiMessage Blockset 173
Model Compare
migration 143
new features 141
ModelDesk
new features 145
MotionDesk
migration 150
new features 149

N

new features
ASM Base InCylinder Blockset 54
ASM Diesel Engine Blockset 55
ASM Diesel InCylinder Blockset 60
ASM Drivetrain Basic Blockset 62
ASM Electric Components Blockset 63
ASM Engine Gasoline Basic Blockset 67
ASM Engine Gasoline Blockset 70
ASM Environment Blockset 66
ASM Gasoline InCylinder Blockset 74
ASM Optimizer 76
ASM Traffic Blockset 78
ASM Trailer Blockset 80
ASM Truck Blockset 82
ASM Turbocharger Blockset 84
ASM Vehicle Dynamics Blockset 87
AutomationDesk 49
ControlDesk Next Generation 100
DCI Configuration Tool 123
DS1007 154
dSPACE FlexRay Configuration Package 125
dSPACE HIL API .NET 127
dSPACE Python Extensions 129
dSPACE XIL API 133

- ECU Interface Manager 135
- Firmware Manager 139
- MicroAutoBox 155
- MicroLabBox 153
- Model Compare 141
- ModelDesk 145
- MotionDesk 149
- Real-Time Testing 151
- RTI Bypass Blockset 157
- RTI CAN MultiMessage Blockset 161
- RTI Electric Motor Control Blockset 165
- RTI FPGA Programming Blockset 167
- RTI LIN MultiMessage Blockset 173
- RTI/RTI-MP 153
- RTLib 153
- SCALEXIO firmware 175
- SystemDesk 178
- VEOS 247

new hardware 15

not supported MATLAB features (R2015a) 155

P

- product overview 20
- Products on 64-bit dSPACE DVD 259

R

- RCP and HIL software
  - definition 16
- Real-Time Testing
  - new features 151
- requirements
  - host PC software
    - operating system 255
- RTI Bypass Blockset
  - migration 158
  - new features 157
- RTI CAN MultiMessage Blockset
  - migration 162
  - new features 161
- RTI Electric Motor Control Blockset
  - new features 165
- RTI FPGA Programming Blockset
  - migration 170
  - new features 167
- RTI LIN MultiMessage Blockset
  - migration 173
  - new features 173
- RTI/RTI-MP
  - new features 153
- RTLib
  - new features 153

S

- SCALEXIO firmware
  - new features 175
- supported MATLAB releases 254
- system requirements
  - operating system 255
- SystemDesk
  - new features 178

T

- TargetLink
  - API commands
    - changes 226
  - AUTOSAR features, new
    - supported releases 200
  - code changes
    - migration 227
  - code efficiency, improved 196
  - code generator options
    - backward compatibility 223
    - changed default value 223
  - migrating to
    - new version 218
  - migration
    - AUTOSAR-related 226
    - code changes 227
    - obsolete limitations 244
    - various aspects 242
  - new API functions 215
  - new code generator options 208
  - new features 192
    - general changes 211
    - general enhancements 211
  - new version
    - migrating to 218
  - newly supported Simulink blocks 193
  - target support
    - discontinued compiler versions 203
    - discontinued evaluation boards 203
    - new compiler versions 203
    - new evaluation boards 203
    - supported targets 203
- TargetLink Data Dictionary
  - API commands
    - changes 226
    - new commands 207
  - migrating to
    - new version 218
  - migration 219
    - discontinued documentation 219
    - manually upgrading libraries and models 222
    - upgrading existing data dictionaries 220
  - new features 192
  - new version
    - migrating to 218

V

- VEOS
  - new features 247
- version history 20

W

- Windows 64-bit
  - limitations 258
- Windows 7
  - limitations 261