dSPACE Release

# New Features and Migration

Release 2021-B – November 2021

dSPACE

## How to Contact dSPACE

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

## How to Contact dSPACE Support

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

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

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

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

## Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent AUTERA RTLib version. Visit http://www.dspace.com/go/auterartlib for software updates.

## Important Notice

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

# Contents

## Bus Manager (Stand-Alone) 49

## ConfigurationDesk 53

## ControlDesk 67

## dSPACE AUTOSAR Compare 89

# About This Document

**Content**

This document informs you about the new features of all the dSPACE software products in Release 2021-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.

**Printed document**

A printed copy of this document is available on demand. You can order it free of charge by using the following link: http://www.dspace.com/go/requestreleasematerial.

**Symbols**

dSPACE user documentation uses the following symbols:

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

**Naming conventions**

dSPACE user documentation uses the following naming conventions:

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

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

## Special folders

Some software products use the following special folders:

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

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

or

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

**Documents folder** A standard folder for user-specific documents.

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

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

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

## Accessing dSPACE Help and PDF Files

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

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

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

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

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

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

# Overview of dSPACE Release 2021-B

**Introduction**

Gives you an overview of the new key features in Release 2021-B and information about unchanged products.

**Where to go from here**

Information in this section

## General Enhancements and Changes

**Introduction**

The following new features and changes concern several dSPACE products.

**Providing legal notes for using third-party software**

Several dSPACE software products use or contain third-party software. The third-party components may be subject to additional terms and conditions or terms and conditions that deviate from the ones that apply to the dSPACE software. Copyright notices and license terms of third-party components must be adhered to. If the user is required to be explicitly informed of and to comply with the terms and conditions of third-party components, the relevant information on terms and conditions is stated in separate OSSAcknowledgements files for each dSPACE product.

You can find the OSSAcknowledgements files as follows:

- In the Legal folders accessible via the root directory on both dSPACE DVDs and the root directory of the downloaded installation files (mounted ISO images or unpacked ZIP files).
- After you install the dSPACE software in the <main installation path of dSPACE Setup>\<subfolder of InstallationSet>\Legal folder.

### RCP and HIL software: C/C++ compilers for building MATLAB MEX files

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

- MinGW (GNU Compiler Collection (GCC 6.3.0))
- Microsoft Visual Studio 2019 Professional

## New Features of dSPACE Help

### Overview video

**New Features of dSPACE Help: Release 2021-B**

This video gives you an overview of the new features of dSPACE Help.

dSPACE Help

New Features

Release 2021-B

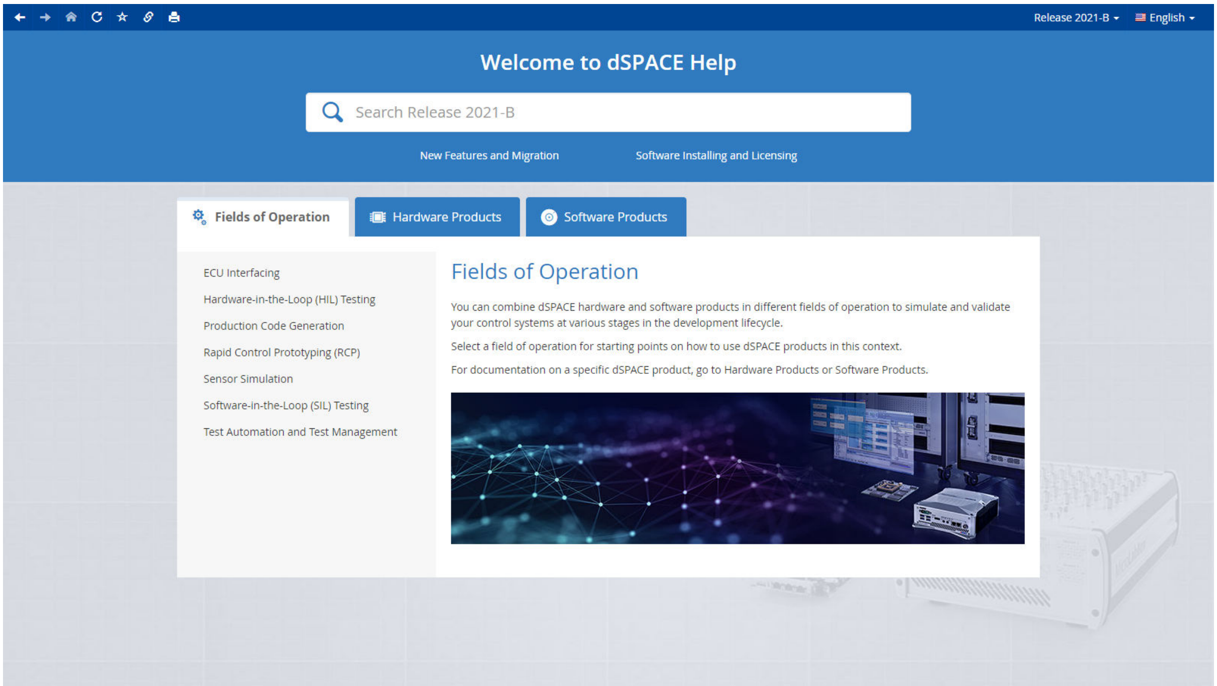
To watch this video, visit http://www.dspace.com/dspace-help/dewvF or scan the QR code.

### Enhanced home page

The home page of dSPACE Help was enhanced to improve navigation and user experience.

The home page now contains the Fields of Operation category. This category contains information on different fields of operation, which combine various dSPACE hardware and software products.

Each field of operation and product contains a start page that provides recommended starting points and information on the field of operation or product.



Release 2021-B ▾ English ▾

Welcome to dSPACE Help

Search Release 2021-B

New Features and Migration

Software Installing and Licensing

Fields of Operation | Hardware Products | Software Products

- ECU Interfacing
- Hardware-in-the-Loop (HIL) Testing
- Production Code Generation
- Rapid Control Prototyping (RCP)
- Sensor Simulation
- Software-in-the-Loop (SIL) Testing
- Test Automation and Test Management

Fields of Operation

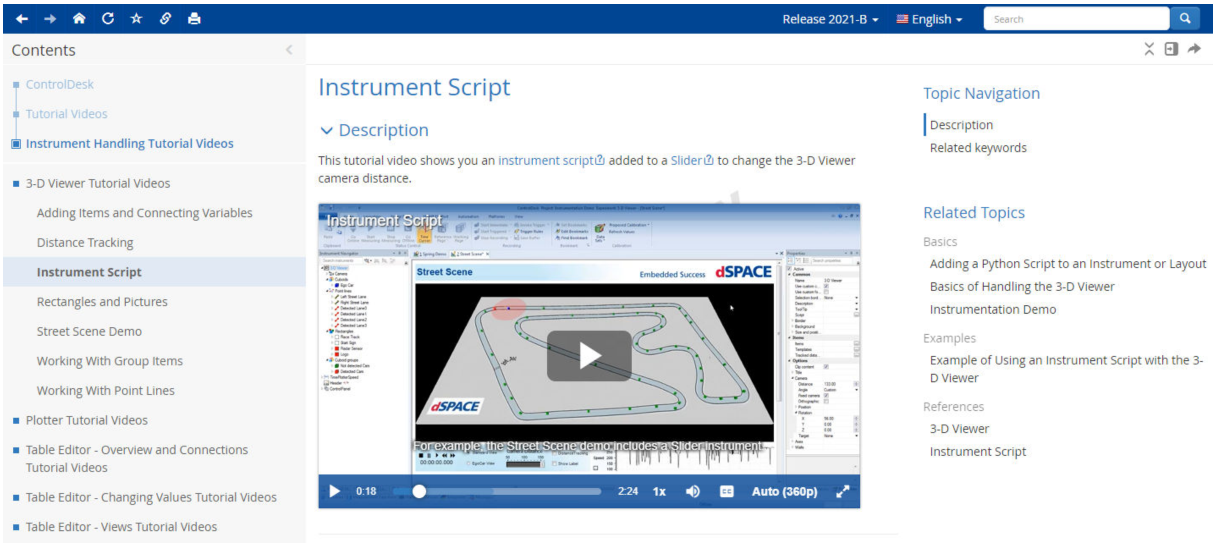
You can combine dSPACE hardware and software products in different fields of operation to simulate and validate your control systems at various stages in the development lifecycle.

Select a field of operation for starting points on how to use dSPACE products in this context.

For documentation on a specific dSPACE product, go to Hardware Products or Software Products.

## Tutorial videos in dSPACE Help topics

You can now watch tutorial videos in a video player in dSPACE Help topics. You can search for tutorial videos by using the Tutorial Videos filter. When using the dSPACE user documentation in PDF format, you can access tutorial videos via QR codes.



Release 2021-B ▾ English ▾ Search

Contents

- ControlDesk
- Tutorial Videos
- Instrument Handling Tutorial Videos
- 3-D Viewer Tutorial Videos
  - Adding Items and Connecting Variables
  - Distance Tracking
  - Instrument Script
  - Rectangles and Pictures
  - Street Scene Demo
  - Working With Group Items
  - Working With Point Lines
- Plotter Tutorial Videos
- Table Editor - Overview and Connections Tutorial Videos
- Table Editor - Changing Values Tutorial Videos
- Table Editor - Views Tutorial Videos

Instrument Script

Description

This tutorial video shows you an instrument script added to a Slider to change the 3-D Viewer camera distance.

Topic Navigation

- Description
- Related keywords

Related Topics

Basics

- Adding a Python Script to an Instrument or Layout
- Basics of Handling the 3-D Viewer
- Instrumentation Demo

Examples

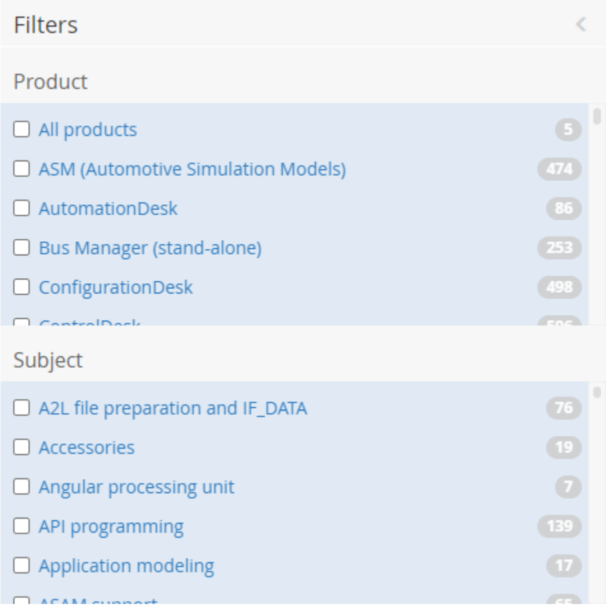
- Example of Using an Instrument Script with the 3-D Viewer

References

- 3-D Viewer
- Instrument Script

**Enhanced product search filters**

The number of product search filters has been reduced to make faceted filtering easier. For example, there now is only one filter for all ASM products.



Filters

Product

□ All products 5

□ ASM (Automotive Simulation Models) 474

□ AutomationDesk 86

□ Bus Manager (stand-alone) 253

□ ConfigurationDesk 498

Subject

□ A2L file preparation and IF_DATA 76

□ Accessories 19

□ Angular processing unit 7

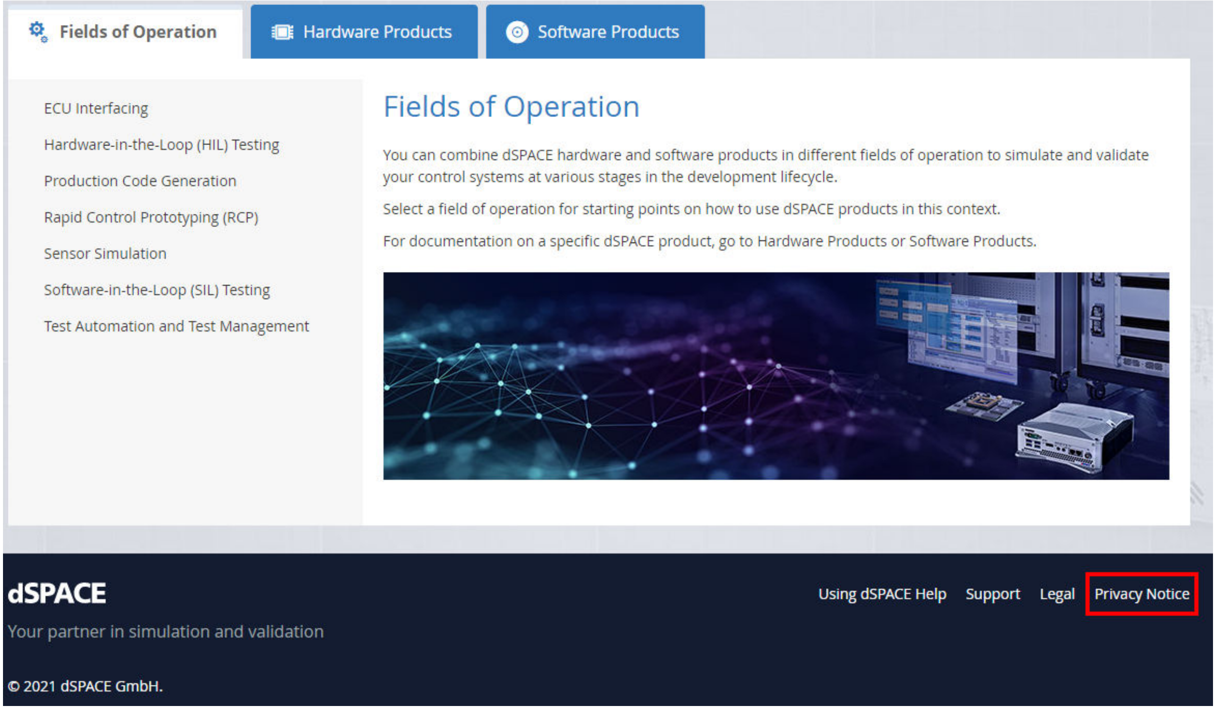
□ API programming 139

□ Application modeling 17

**Improving dSPACE Help via an analytics function**

To further improve dSPACE Help in terms of content and functionality, an analytics function has been added to dSPACE Help. The data collected is anonymous and cannot be used to identify you.

For more information, click the Privacy Notice button in dSPACE Help. You can also deactivate the transmission of user data via the Privacy Notice dialog.



ECU Interfacing

Hardware-in-the-Loop (HIL) Testing

Production Code Generation

Rapid Control Prototyping (RCP)

Sensor Simulation

Software-in-the-Loop (SIL) Testing

Test Automation and Test Management

Fields of Operation

You can combine dSPACE hardware and software products in different fields of operation to simulate and validate your control systems at various stages in the development lifecycle.

Select a field of operation for starting points on how to use dSPACE products in this context.

For documentation on a specific dSPACE product, go to Hardware Products or Software Products.

# Discontinuations

**Introduction**

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

For more end-of-life announcements, refer to https://www.dspace.com/go/discontinuation.

**Python distribution**

As of dSPACE Release 2021-A, the support of Python 3.6 has been discontinued and is replaced by Python 3.9. For more information, refer to Migrating Python Scripts from Python 3.6 to Python 3.9 on page 29.

**Planned discontinuation of software support**

**Operating system**

- As of dSPACE Release 2022-A, Windows LTSB 2016 is no longer supported.
- As of dSPACE Release 2022-A, Windows Server 2016 is no longer supported.

**Variable Editor** As of dSPACE Release 2022-A, the Variable Editor will no longer be available.

**Discontinuation of ConfigurationDesk options** The following ConfigurationDesk options will be discontinued with an upcoming dSPACE Release:

- The Generate Simulink Model Interface command group in the context menu of a Simulink model in the Model Browser

  You can use the Propagate to Simulink Model and the Generate New Simulink Model Interface commands instead.

  This affects scripts for tool automation that use the commands. Use the methods PropagateToConfigurationDeskModelInterface and PropagateToSimulink for the use case.
- The Skip Model Code Generation property of a Simulink model

  For Simulink models (SLX files) in ConfigurationDesk, Simulink® Coder$^{TM}$ checks whether more up-to-date code is available for the model and skips the code generation process if necessary. As an alternative, you can use Simulink implementation containers (SIC files) instead of SLX files. In this case, you can run a build process without generating code for a Simulink model. A MATLAB installation is therefore not required.

**Planned discontinuation of dSPACE hardware**

**PHS-bus hardware** As of December 2021, the hardware components for PHS-bus-based systems, such as the DS1006 Processor Board, the DS1007 PPC Processor Board, and all the PHS-bus I/O boards, will be discontinued. This also applies to the dSPACE Simulator Mid-Size and the dSPACE Simulator EcoLine. New Releases of dSPACE software will continue to support the PHS-bus hardware components until the end of 2023.

For new projects, we recommend that you use SCALEXIO as a modular real-time system.

## Product Version Overview

**Product versions**

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

| Product | dSPACE Release | | | |
|---|---|---|---|---|
| | 2020-A | 2020-B | 2021-A | 2021-B |
| AutomationDesk | 6.3 | 6.4 | 6.5 | 6.6<br>Refer to AutomationDesk on page 35. |
| Automotive Simulation Models | 9.5 | 9.6 | 9.7 | 10.0<br>Refer to Automotive Simulation Models (ASM) on page 39. |

| Product | dSPACE Release | | | |
|---|---|---|---|---|
| | 2020-A | 2020-B | 2021-A | 2021-B |
| Bus Manager (stand-alone) | 6.5 | 6.6 | 6.7 | 6.8<br>Refer to Bus Manager (Stand-Alone) on page 49. |
| ConfigurationDesk for RapidPro | 6.5 | 6.6 | 6.7 | – [1) |
| ConfigurationDesk | 6.5 | 6.6 | 6.7 | 6.8<br>Refer to ConfigurationDesk on page 53. |
| Container Manager | 5.1 | 5.2 | 5.2 | 5.2[2) |
| ControlDesk | 7.2 | 7.3 | 7.4 | 7.5<br>Refer to ControlDesk on page 67. |
| DCI Configuration Tool | 3.12 | 3.12 | 3.13 | 3.13 |
| dSPACE AUTOSAR Compare | – | 1.0 | 1.0 | 1.1<br>Refer to dSPACE AUTOSAR Compare on page 89. |
| dSPACE CAN API Package | 4.0.4 | 4.0.5 | 4.0.6 | 4.0.7 |
| dSPACE ECU Flash Programming Tool | 2.7 | 2.7 | 2.8 | 2.8 |
| dSPACE FlexRay Configuration Package | 4.5 | 4.6 | 4.7 | 4.8 |
| dSPACE Installation Manager | 5.5 | 5.6 | 5.7 | 5.8<br>Refer to dSPACE Installation Manager on page 91 |
| dSPACE Python Extensions | 3.3 | 3.4 | 4.0 | 4.1<br>Refer to dSPACE Python Extensions on page 93. |
| dSPACE XIL API .NET | 2020-A | 2020-B | 2021-A | 2021-B<br>Refer to dSPACE XIL API .NET on page 95. |
| ECU Interface Manager | 2.7 | 2.8 | 2.9 | 2.10<br>Refer to ECU Interface Manager on page 97. |
| Firmware Manager | 3.1 | 3.2 | 3.3 | 3.4<br>Refer to Firmware Manager on page 99. |
| Model and Sensor Interface Blockset | – | 1.0 | 1.1 | 1.2<br>Refer to Model and Sensor Interface Blockset on page 105. |
| Model Compare | 3.0 | 3.1 | 3.1 | 3.2<br>Refer to Model Compare on page 103. |
| ModelDesk | 5.3 | 5.4 | 5.5 | 5.6<br>Refer to ModelDesk on page 107. |
| Model Interface Package for Simulink | 4.3 | 4.4 | 4.5 | 4.6<br>Refer to Model Interface Package for Simulink on page 111. |
| MotionDesk | 4.6 | 4.7 | 4.8 | 4.9<br>Refer to MotionDesk on page 113. |

| Product | dSPACE Release | | | |
|---|---|---|---|---|
| | 2020-A | 2020-B | 2021-A | 2021-B |
| MotionDesk Blockset | 2.6.1 | 2.6.2 | 2.6.3 | 2.6.4<br>Refer to MotionDesk on page 113. |
| Real-Time Testing | 4.3 | 4.4 | 5.0 | 5.1<br>Refer to Real-Time Testing on page 115. |
| RTI[3) ] | 7.14 | 7.15 | 7.16 | 7.17<br>Refer to RTI/RTI-MP and RTLib on page 117. |
| RTI-MP[4)] | 7.14 | 7.15 | 7.16 | 7.17<br>Refer to RTI/RTI-MP and RTLib on page 117. |
| RTI Bypass Blockset | 3.14 | 3.15 | 3.16 | 3.17 |
| RTI CAN Blockset | 3.4.10 | 3.4.11 | 3.4.12 | 3.4.13 |
| RTI CAN MultiMessage Blockset | 5.4 | 5.5 | 5.6 | 5.7<br>Refer to RTI CAN MultiMessage Blockset on page 121. |
| RTI Electric Motor Control Blockset | 1.4.1 | 1.4.2 | 1.4.3 | 1.4.4 |
| RTI Ethernet Blockset | 1.2.3 | 1.2.4 | 1.2.5 | 1.2.6 |
| RTI Ethernet (UDP) Blockset | 1.4.3 | 1.4.4 | 1.4.5 | 1.4.6 |
| RTI FPGA Programming Blockset | 3.9 | 3.10 | 3.11 | 3.12<br>Refer to RTI FPGA Programming Blockset on page 123. |
| RTI LIN MultiMessage Blockset | 3.4 | 3.5 | 3.6 | 3.7 |
| RTI RapidPro Control Unit Blockset | 2.2.3 | 2.2.4 | 2.2.5 | 2.2.6 |
| RTI Synchronized Time Base Manager Blockset | 1.4 | 1.4.1 | 1.4.2 | 1.4.3 |
| RTI USB Flight Recorder Blockset | 1.2.2 | 1.2.3 | 1.2.4 | 1.2.5 |
| RTI Watchdog Blockset | 2.1.1 | 2.1.2 | 2.1.3 | 2.1.4 |
| Sensor Simulation | 1.3 | 1.4 | 1.5 | 1.6<br>Refer to Sensor Simulation on page 131. |
| SCALEXIO firmware | 4.6 | 5.0 | 5.1 | 5.2<br>Refer to SCALEXIO Firmware on page 129. |
| SYNECT | 2.9 | 2.10 | 2.11 | 2.12<br>Refer to SYNECT on page 139. |
| SystemDesk | 5.4 | 5.5 | 5.5 | 5.6<br>Refer to SystemDesk on page 133. |
| TargetLink | 5.0 | 5.1 | 5.1 | 5.2<br>Refer to TargetLink on page 147. |
| Variable Editor[5), 6)] | 2.4 | 2.4 | 2.4 | 2.4 |
| VEOS | 5.0 | 5.1 | 5.2 | 5.3<br>Refer to VEOS on page 193. |

[1)] As of dSPACE Release 2021-B, ConfigurationDesk for RapidPro ist no longer part of the dSPACE Release. You can download it from the dSPACE website as a separate software package with its own setup program. Refer to https://www.dspace.com/go/CFDRP.

[2] Version 5.2 will be the last version of the Container Manager, which will be discontinued as of dSPACE Release 2022-B.

[3] Including the standard I/O blocksets.

[4] Including the RTI Gigalink Blockset.

[5] The Variable Editor is not part of the dSPACE Release DVD. It is available at https://www.dspace.com/go/requestreleasedownload.

[6] As of dSPACE Release 2022-A, the Variable Editor will no longer be available.

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

## New Key Product Features

### Introduction

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

### AutomationDesk

The new key features of AutomationDesk are:

- Changed first-run behavior concerning the demo projects.
- Enhancements to the **Dialogs** library.
- Enhancements to the SYNECT interface.
- Discontinuation of the RS232 library.

For more information on the new features, refer to New Features of AutomationDesk 6.6 on page 35.

### Bus Manager (stand-alone)

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

- Simulating partial networking according to AUTOSAR 4.3.
- Creating restbus configurations, i.e., assigning all communication matrix elements that are required for the restbus simulation to a bus configuration in one step.
- Inspecting J1939-compliant PDUs.
- Enhanced bus configuration features
- Support of the DBName attribute of DBC files

For more information, refer to New Features of the Bus Manager (Stand-Alone) 6.8 on page 49.

### ConfigurationDesk

The new key features of ConfigurationDesk are:

- The **FuSa Memory Monitoring** function block is available to trigger a FuSa error if a RAM failure occurs that cannot be corrected.

- Changes to project and application management:
  - Project management synchronized with file system
  - New Project Location Manager
  - New file names and extensions
  - Simplified access to projects and applications
  - Reworked project and application creation
  - New Python script handling
- New task profiling options for calculating turnaround times of runnable functions and tasks.
- A new TRC file entry for the memory usage of a real-time application.
- Various enhancements of the Bus Manager for configuring bus communication for simulation, inspection, and manipulation purposes.

For more information, refer to ConfigurationDesk on page 53.

**ControlDesk**

The new key features of ControlDesk are:

**Platform/device enhancements**

- cmdloader: Configuring and registering dSPACE platforms using a configuration file

For more information, refer to New Features of Platform Management and Platforms/Devices (ControlDesk 7.5) on page 68.

**Variable management enhancements**

- Improved management and handling of large variable descriptions

For more information, refer to New Variable Management Features (ControlDesk 7.5) on page 68.

**Instrument enhancements**

- New Hierarchy Array

For more information, refer to New Instrument Features (ControlDesk 7.5) on page 69.

**Bus Navigator enhancements**

- Bus instruments (Bus Manager): Support for signals that use text tables
- SOME/IP service discovery: Display of decoded item names in the Decoding view
- CAN, LIN: Display of the sender ECU in the monitoring list
- CAN, LIN: Improvement for monitoring, logging, and replay in connection with Bus Manager applications
- Support of R20-11 ARXML files

For more information, refer to New Bus Navigator Features (ControlDesk 7.5) on page 69.

**Automation enhancements**

- Automating the progress bar
- Copying demo projects

For more information, refer to New Automation Features (ControlDesk 7.5) on page 71.

**dSPACE AUTOSAR Compare**

The new key features of dSPACE AUTOSAR Compare are:

- Support of AUTOSAR Release R20-11.
- dSPACE AUTOSAR Compare can be integrated into version control systems, such as Git, to view differences between different revisions of ARXML files and to assist with merge conflicts.

For more information on the new features, refer to dSPACE AUTOSAR Compare on page 89.

**dSPACE XIL API**

The new key features of dSPACE XIL API and the Platform Management API are:

- The `CmdLoader` in the XIL API on Linux installation has been enhanced.
- Enhanced user documentation

For more information on the new features, refer to New Features of dSPACE XIL API .NET 2021-B on page 95.

**ECU Interface Manager**

The new key feature of the ECU Interface Manager is:

- Support for variables with address extension ! = 0

For more information on the new feature, refer to New Features of ECU Interface Manager 2.10 on page 97.

**Firmware Manager**

The Firmware Manager has no new feature, but the QNX firmware archive for SCALEXIO systems is no longer available.

For more information, refer to New Features of Firmware Manager 3.4 on page 99.

**MicroAutoBox III firmware**

The new key features of the MicroAutoBox III firmware are:

- Extended DS1403 Processor Board support.
- Extended DS1554 Engine Control I/O Module support.

For more information on the new features, refer to New Features of the MicroAutoBox III Firmware 5.2 on page 101.

**Model and Sensor Interface Blockset**

The new key features of the Model and Sensor Interface Blockset are:

- Improved usability of the block parameters dialogs.
- A function to evaluate sample times before a model is built.
- Automatic renaming of duplicate parent object names when copying a Simulation Data Object block.
- Performance upgrades and optimizations.

For more information on the new features, refer to New Features of Model and Sensor Interface Blockset 1.2 on page 105.

**Model Compare**

The new key features of Model Compare 3.2 are:

- The new Difference Overview dialog displays a quick overview of changed model parts and allows for a preselection of specific model parts for the comparison.
- The workflow integration is improved by additional command line interface options.

For more information on the new features, refer to New Features of Model Compare 3.2 on page 103.

**ModelDesk**

The new key features of ModelDesk are:

- V2X support: Specifying signal phases and timings for traffic lights to simulate SPaT messages.

For more information on the new features, refer to New Features of ModelDesk 5.6 on page 107.

**Model Interface Package for Simulink**

The new key features of Model Interface Package for Simulink are:

- Support of MATLAB R2021b.
- Support of obfuscated source code in Simulink models for the generation of SIC files.

For more information on the new features, refer to Model Interface Package for Simulink on page 111.

**RTI, RTI-MP, and RTLib**

The new key feature of RTI, RTI-MP, and RTLib is:

- Support of MATLAB R2021b.

For more information, refer to New Features of RTI/RTI-MP and RTLib on page 117.

**RTI CAN MultiMessage Blockset**

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

- Support of complete names of frames and signals.
- Support of debounce time functionality for GTS messages.
- Reset functionality of the RTICANMM MainBlock during simulation.

For more information on the new features, refer to New Features of the RTI CAN MultiMessage Blockset 5.7 on page 121.

**RTI FPGA Programming Blockset**

The new key features of the RTI FPGA Programming Blockset 3.12 are:

- Extended Xilinx® software support.
- Enhancements to the FPGA framework for the MicroAutoBox III and SCALEXIO.

For more information on the new features, refer to New Features of the RTI FPGA Programming Blockset 3.12 on page 123.

### SYNECT

The new key features of SYNECT 2.12 are:

- The SYNECT Jira add-on for the automatic creation of Jira issues for SYNECT items.
- The Interactive Excel Report add-on for loading data on request via the OData service of the SYNECT server to create Pivot charts with Microsoft® Excel®.

For more information on the new features, refer to New Features of SYNECT 2.12 on page 140.

### SystemDesk

The new key features of SystemDesk 5.6 are:

- Support of the AUTOSAR R20-11 release.
- Exporting V-ECU FMUs.

For more information on the new features, refer to New Features of SystemDesk 5.6 on page 134.

### Targetlink

The new key features of TargetLink 5.2 are:

- **Arrays of struct**: The dynamic access to array-of-struct variables via special modeling patterns using different blocks., e.g., Data Store Read/Write, Bus Inport/Outport, Bus Selector/Selector and Assignment, is now supported.
- **Data types**: 64-bit integer data types are now supported in different modeling situations.
- **Adaptive AUTOSAR**: Modeling and simulating error handling is now possible.
- **Classic AUTOSAR**: It is now possible to generate and update frame models of composition SWCs for simulation purposes.
- **Usability improvements**: The Data Dictionary Manager provides several usability improvements.

For more information on all new features, refer to New Features of TargetLink 5.2 on page 148.

For more information on the TargetLink migration aspects (TargetLink, TargetLink AUTOSAR Module, TargetLink Data Dictionary), refer to Migrating to TargetLink 5.2 and TargetLink Data Dictionary 5.2 on page 158.

### VEOS

The new key features of VEOS are:

- Classic V-ECU enhancements
- VPU access by several XCP masters
- Building V-ECUs for 32-bit Linux operating systems
- Command line interface extension: VEOS Model Console

For more information on the new features, refer to New Features of VEOS 5.3 on page 193.

# Aspects of Migrating from Previous Releases

**Introduction**

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

## Migrating to dSPACE Release 2021-B

**Introduction**

After you install Release 2021-B, some additional steps might be required.

**Migrating from dSPACE Release 2021-A**

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

**Migrating from dSPACE Release 2020-B or earlier**

To migrate from dSPACE Release 2020-B or earlier to Release 2021-B, you also have to perform the migration steps of the intervening dSPACE Releases. All of the required migration steps can be performed using software from dSPACE Release 2021-B.

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

**Previous Release documents**

The PDF files of previous Releases are called `NewFeaturesAndMigrationxx.pdf`, where xx represents the Release number.

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

- In the installation folder of the current dSPACE Help. Refer to `C:\Program Files\Common Files\dSPACE\Help 2021-B\Print\PreviousReleases`.
- On the dSPACE DVDs. Refer to `\Doc\PreviousReleases`.

- At www.dspace.com/go/migration for download. Here, you can also find *New Features and Migration* documents for very early Releases.

# Migrating Python Scripts from Python 3.6 to Python 3.9

**Where to go from here**

Information in this section

## Main Changes in Python 3.9

**Main reason for using Python 3.9 instead of Python 3.6**

The end of life of Python 3.6 is scheduled for the end of December 2021. Therefore, dSPACE switched to Python 3.9 with dSPACE Release 2021-A.

**Related documents available on the Python website**

For a short overview of the main changes in Python 3.9, refer to Technical Changes on page 32. For information on all the changes in the Python language and environment from Python 3.6 to Python 3.9, refer to www.python.org.

The Python Software Foundation provides the following documents:

- What's New from Python 3.6 to 3.7: https://docs.python.org/3.7/whatsnew/3.7.html
- What's New from Python 3.7 to 3.8: https://docs.python.org/3.8/whatsnew/3.8.html
- What's New from Python 3.8 to 3.9: https://docs.python.org/3.9/whatsnew/3.9.html

**Related topics**

Basics

## Main Changes in Handling Python 3.9 with dSPACE Software

**Introduction**

With Python 3.9, some aspects of handling Python with dSPACE software have changed.

**Libraries**

The libraries and components used with Python 3.9 and distributed on dSPACE DVDs have changed as shown in the following table.

| Package | Python 3.6 (Release 2020-B) | Python 3.9 (Release 2021-B) |
|---|---|---|
| comtypes | 1.1.7 | 1.1.10 |
| Core | 3.6.8 | 3.9.5 |
| cycler | 0.10.0 | 0.10.0 |
| future | 0.18.2 | 0.18.2 |
| grpcio | 1.29.0 | 1.38.0 |
| grpcio_tools | 1.29.0 | 1.38.0 |
| kiwisolver | 1.2.0 | 1.3.1 |
| lxml | – | 4.6.3 |
| matplotlib | 3.2.1 | 3.4.2 |
| numpy | 1.18.4 | 1.20.3 |
| pillow | 7.1.2 | 8.2.0 |
| pip | 20.1.1 | 21.1.2 |
| protobuf | 3.12.2 | 3.17.3 |

| Package | Python 3.6<br>(Release 2020-B) | Python 3.9<br>(Release 2021-B) |
|---|---|---|
| pycparser | – | 2.20 |
| pyglet | 1.5.5 | 1.5.17 |
| pyparsing | 2.4.7 | 2.4.7 |
| pypubsub | 4.0.3 | 4.0.3 |
| Python-dateutil | 2.8.1 | 2.8.1 |
| pythonnet | 2.4.2 | 2.5.3 |
| pytz | 2020.1 | 2021.1 |
| pywin32 | 227.10 | 301.10 |
| scipy | 1.4.1 | 1.6.3 |
| six | 1.15.0 | 1.16.0 |
| wxPython | 4.1.0 | 4.1.1 |
| yapsy | 1.12.2 | 1.12.2 |

## General Information on Using Python Installations

**Introduction**

If you work with dSPACE software from dSPACE Release 2020-B or earlier, which supports Python 3.6, and dSPACE software from dSPACE Release 2021-A or later, which supports Python 3.9, both Python versions are installed on the PC and can be used in parallel.

**Limitations when using Python 3.6 and Python 3.9 in parallel**

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

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

  **Note**

  If you install dSPACE Release 2020-B or earlier after the installation of dSPACE Release 2021-B, Python 3.6 will be installed last. Therefore, PythonWin 3.6 is registered to open Python files. To set the file associations to PythonWin 3.9, execute the `PythonInstaller.exe` from `<DVDRoot>\Products\Common\Python3.9` again.

  To open a Python file with PythonWin 3.9 temporarily, use the **`Edit with PythonWin 3.9`** command on the context menu of the file.

- Environment variables are used by both Python versions. You must set their values, for example, for PYTHONHOME, to the Python installation you want to work with. For an overview of environment variables set by Python, refer to: https://docs.python.org/3.9/using/windows.html.

**Note**

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

**Using dSPACE test automation with both Python versions in parallel**

If a test automation script uses dSPACE Python Modules and you do not want to migrate the script, you have to work with both Python versions in parallel. The dSPACE Python Modules for Python 3.6 are available up to and including Release 2020-B.

**Note**

It is recommended to use only dSPACE software from the same Release when using Python scripts. Using both Python versions in the same application context, such as automating the access to an older version of ControlDesk via AutomationDesk or using AutomationDesk with an activated Real-Time Testing version that does not support Python 3.9, might cause a conflict.

## Technical Changes

**Introduction**

Migrating from Python 3.6 to Python 3.9 has caused minor changes in the Python API.

**Note**

Independently of the number of required manual modifications, you must test the migrated code.

**Migration issues**

The following changes in Python 3.9 require manual migration of your Python scripts.

**Python.NET: Return value of IList type is converted to PyList** Since Python.NET 2.4, return values of `SystemCollections.Generic.IList` type have been converted to the `PyList` type. Because some methods and properties of an `IList` object are not available with a `PyList` object, dSPACE provided the modified Python.NET 2.4.1 and 2.4.2 packages, which suppressed the conversion.

As of Python.NET 2.5.3, the modifications by dSPACE are no longer provided. You must therefore migrate the methods and properties of an `IList` object to the related methods and properties of a `PyList` object. For example, you must use `len(MyObject)` instead of `MyObject.Count`.

**Note**

The created Python list is a copy of the .NET object. Modifications to the Python list therefore have no effect on the associated .NET object. To apply the modifications to the .NET object, assign the Python list afterward.

**ctypes: Loading a library after changing the PATH variable** Loading a library via `ctypes` can fail if the path to the library was added to the PATH variable beforehand.

To load a library in a safe manner, use `ctypes.WinDLL(NameOfDll, winmode=8)` instead of `ctypes.windll.LoadLibrary(NameOfDll)`.

## Product-Specific Information on Migrating

**Products with editable Python content**

Python scripts are used explicitly or implicitly in the following dSPACE products:

- AutomationDesk
- Bus Manager (stand-alone)
- ConfigurationDesk
- ControlDesk
- ModelDesk
- dSPACE Platform API Package (Python Extensions, XIL API .NET, Platform Management API)
- Real-Time Testing
- SYNECT
- SystemDesk

No other dSPACE products are affected by the Python migration.

You must check all custom scripts of the above products for required migration, e.g., custom user functions. For further product-specific information, see below.

**Using Real-Time Testing**

In Real-Time Testing, Python is used on the host PC and on the real-time platform.

The interpreters support different Python versions:

- Python 3.9:
  - Interpreter on the host PC
- Python 3.6:
  - Interpreter used with VEOS as of dSPACE Release 2019-B
  - Interpreter on MicroAutoBox III as of dSPACE Release 2019-B
  - Interpreter on SCALEXIO/DS6001 as of dSPACE Release 2020-B
- Python 2.7.11:
  - All other real-time platforms

You must migrate your custom scripts to Python 3.9 manually. Real-Time Testing does not provide any features for the automatic migration of custom scripts from Python 3.6 to Python 3.9.

The activated Real-Time Testing version and the dSPACE software using Real-Time Testing, e.g., AutomationDesk, must support the same Python version.

### Using dSPACE XIL API .NET

All dSPACE XIL API .NET methods and properties returning `IList` types must be migrated to the `PyList` type except for the `ASAM.XIL.Interfaces.Testbench.Common.Capturing.ICapture.Variables` property, which uses the `ObservableList` type.

### Using SYNECT

**Changed list handling of Python plug-ins** The handling of lists in Python plug-ins changed. However, you can import the `ListAsRawEncoder` module to migrate scripts.

This is due to changes of list handling that were introduced with Python.NET 2.5.3. For details, refer to Technical Changes on page 32.

**Setting Min/Max properties of variables** The handling of variables when exchanging data via Python plug-ins changed. You now have to convert strings to double values if you specify the Min/Max properties of variables.

*Old* listing:

```
Variable = SignalsAndParameterManagement.Variable()
variable.Min = '-10'
variable.Max = '10'
```

*New* listing:

```
Variable = SignalsAndParameterManagement.Variable()
variable.Min = float('-10')
variable.Max = float('10')
```

For details on the variable class for exchanging signals and parameters, refer to Variable (SYNECT Guide 🕮).

### Using SystemDesk

SystemDesk uses an external Python interpreter. It is therefore not affected by the Python migration. Only the demos depend on the installed Python distribution. They are updated with dSPACE Release 2021-B.

# AutomationDesk

**Where to go from here**

Information in this section

## New Features of AutomationDesk 6.6

**General enhancements**

**Changed first-run behavior concerning the demo projects** Up to AutomationDesk 6.5, the AutomationDesk demo projects were automatically copied to the user-specific `Documents` folder at the first start of AutomationDesk. Now, you can decide whether and when to copy the demo projects. The AutomationDesk Options dialog provides the **`Copy Demos`** command to explicitly copy the demos to your `Documents` folder.

For more information, refer to User Settings Page (AutomationDesk Basic Practices 🕮).

**Enhancements to the SYNECT interface** You can now edit and synchronize the implementation state attribute of a SYNECT test case in AutomationDesk via user interface and AutomationDesk API.

For more information, refer to View Properties (AutomationDesk Basic Practices 🕮) and ImplementationState (AutomationDesk Automation 🕮).

For more information on the implementation state attribute in SYNECT, refer to Basics on a State Model for Test Implementation (SYNECT Guide 🕮).

**Tutorial video** The tutorial video that provides basic information on signal-based testing is now directly referenced in dSPACE Help. Refer to Overview on the Test Concept and the Involved Components (AutomationDesk Implementing Signal-Based Tests 🕮).

**Enhancements to the Dialogs library**

**Timeout settings for the blocks in the Dialogs library** The AutomationDesk Options dialog now provides settings to configure the behavior of the Dialogs library blocks. By default, the input and message dialogs wait for user interaction. With the new timeout settings, you can specify the duration after which the dialogs are closed automatically and return the specified offline values. You can also specify an action triggered by the timeout, e.g., writing a message to the log, generating a verdict, or throwing an exception.

For more information, refer to General Page (AutomationDesk Basic Practices 🕮).

**New input dialog block** The library has been enhanced by the SelectMultipleValues block. It lets you select multiple values from a predefined list via checkboxes as input values for the current execution.

For more information, refer to SelectMultipleValues (AutomationDesk Basic Practices 🕮).

## Migrating to AutomationDesk 6.6

**General migration aspects**

If you open an AutomationDesk project with a later AutomationDesk version, the software automatically detects whether migration is required. You must click OK in the message dialog to start the migration. Save the migrated project to another path or name.

An AutomationDesk project loaded and saved with AutomationDesk 6.3 and later can be opened in AutomationDesk 6.2. The different schema versions are then displayed in a warning message. It might not be possible to execute the project with the earlier AutomationDesk version.

**Note**

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

- You must create backups of the project and of the linked custom libraries.
- AutomationDesk must be running properly. The Log Viewer must not display any error messages.
- The built-in libraries, required custom libraries, and other packages must be loaded properly.

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

For more information, refer to Migrating AutomationDesk (AutomationDesk Introduction And Overview 🕮).

**Migration to the new serialization format introduced with AutomationDesk 6.2**

If you open an AutomationDesk project or a custom library created with AutomationDesk 6.1 or earlier, the data is automatically migrated to the new XML format. The migration considers the standard serialization format, exported projects and custom libraries in ZIP format, and exported projects and custom libraries in the legacy XML format.

If you use a version control system, you can get and open projects and custom libraries in the legacy formats. They are automatically migrated to the new serialization format and lose their connections to the version control system. You must add the migrated projects and custom libraries to new version control projects to avoid mixing the formats in the repository. For more information, refer to How to Migrate Projects or Custom Libraries Under Version Control (AutomationDesk Basic Practices 📖).

**Project import restricted since Release 2021-A**

Since dSPACE Release 2021-A, AutomationDesk has supported the direct import only of projects last saved with one of the previous seven AutomationDesk versions.

**Discontinued libraries and blocks**

If you open a project containing discontinued elements in AutomationDesk 6.4, the discontinued data objects are replaced by **Discontinued data object** data objects and the discontinued blocks are replaced by **Discontinued block** blocks during the automatic project update. This lets you load your projects and search for blocks and data objects to be migrated. If you execute a project containing elements of a discontinued library via AutomationDesk or an API script, project execution will stop with an exception.

The following blocks and data objects are affected.

**RS232 library** The RS232 library is no longer included in AutomationDesk 6.6. A custom library that provides the same functionality is available on demand. Refer to https://www.dspace.com/go/AUD-RS232-CustLib.

Python scripts using the `rs232lib2` Python module in an **Exec** or **ExecFile** block are also affected by the discontinuation. As an alternative, you can use the standard Python serial port extension `pyserial`.

For more information on migration aspects, refer to Migrating AutomationDesk (AutomationDesk Introduction And Overview 📖).

**Discontinuation in ControlDesk**

As of ControlDesk 7.5, the ControlDesk MCD 3 automation interface no longer supports the access to calculated variables. As a consequence, you have to adapt sequences that involve the access to calculated variables using the ControlDesk MCD 3 automation interface. Refer to Discontinuations in ControlDesk on page 74.

**Planned discontinuations**

**Automation Server** As of December 2022, the *Automation Server* will be discontinued. New Releases of the AutomationDesk Automation Server will only be available for customers with a Software Maintenance Service Contract until June 2022.

The *AutomationDesk Basic* license provides the technical equivalent to the Automation Server.

For more information, refer to http://www.dspace.com/go/discontinuation.

# Automotive Simulation Models (ASM)

**Where to go from here**

Information in this section

# All ASM Products

## New Features of All ASM Blocksets

**Tutorial videos**

There are new ASM tutorial videos. The videos describe the first steps with ASM.

The tutorial videos are available on the dSPACE website and now also in dSPACE Help.

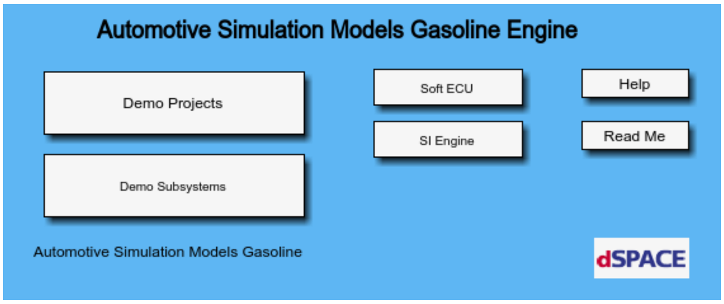
- To access the videos in dSPACE Help, go to ASM Tutorial Videos (ASM User Guide 🕮).
- To access the videos on dSPACE website, go to https://www.dspace.com/en/pub/home/medien/videos.cfm.

**Demo subsystems access**

The top layers of the ASM libraries have been restructured. You can now directly access demo subsystems via the **Demo Subsystems** subsystem.

Demo subsystems can be used as variants for some components inside the demo models.

The following illustration shows the top layer of the ASM Gasoline Engine library as an example:


Automotive Simulation Models Gasoline Engine
Demo Projects
Soft ECU
Help
SI Engine
Read Me
Demo Subsystems
Automotive Simulation Models Gasoline
dSPACE

# ASM Drivetrain Basic

## New Features of ASM Drivetrain Basic Blockset 5.5

**PLANETARY_GEAR block**

There is a new block to simulate the planetary gear set consisting of sun gear, planet gear carrier, planet gears, and ring gear. It uses the torques acting on the individual gears as well as the gears' geometric and dynamic parameters to calculate the speeds of the corresponding gears.

# ASM Electric Components

## Migrating to ASM Electric Components Blockset 3.12

**TRQ_REQUEST_ COORDINATION_BEV block**

A new SOC_Battery[%] inport and a new Const_SOC_Min[%] parameter have been added to switch off the torque request when the state of charge (SOC) reaches a minimum value.

**Related topics**

Basics

Migrating ASM Models (ASM User Guide 📖)

# ASM Fuel Cell

## New Features of ASM Fuel Cell Blockset 1.0

**New library and new demo model**

There is a new ASM library: ASM Fuel Cell. It can be used to create a simulation model of a proton exchange membrane (PEM) fuel cell stack with its balance-of-plant components like air compressor, valves, intercooler, humidifier, and recirculation blower. There is also a new demo project. It comes with a fully functional simulation model of a fuel cell electric vehicle (FCEV), a ControlDesk project to operate, and a ModelDesk project to parameterize the model.

For more information on ASM Fuel Cell, refer to http://www.dspace.com/go/asm_fuelcell.

# ASM Turbocharger

## Migrating to ASM Turbocharger Blockset 3.2.9

**COMPRESSOR block**

The license check of the internally used Euler blocks to be inherited from the COMPRESSOR block has been changed.

**Related topics**

Basics

Migrating ASM Models (ASM User Guide 📖)

# ASM Environment

## New Features of ASM Environment Blockset 5.0

**ROAD block**

A new signal has been added to the INFO output of the road. This signal provides a pointer to *V2X* information. This is relevant for MAP and SPaT messages, for example.

# ASM KnC

## New Features of ASM KnC 8.2

| | |
|---|---|
| **User interface** | The ASM KnC user interface now uses the version 3.6 of the OpenSceneGraph. This does not affect the functionality of ASM KnC. |

# ASM Traffic

**Where to go from here**

Information in this section

## New Features of ASM Traffic Blockset 4.3

**OBJECT_SENSOR_3D_ CALCULATION block**

Several changes have been made to the block:

**History information** The block now provides *history* information about objects in the sensor scope:

- Current age of an object in the scope starting from the time of entering the scope.
- Number of times an object enters the scope during a maneuver.
- Time of first entry of an object into the scope related to maneuver start.

**Output mode** The block provides a new output mode: The detected objects are output at a fixed position within the vectorized output as long as the objects are in the scope.

**Time stamps** The block now provides two time stamps for the object list:

- Time of sensor calculation related to maneuver start.
- Current sample time of incoming object related to maneuver start.

**Nearest point information** The block now provides information about the nearest point of an object in the sensor system of coordinates.

**V2X_TRAFFIC_LIGHT_ CONTROLLER block**

This block is new. It calculates vehicle-to-X (V2X)-specific data such as the current state and the corresponding start and end time for each signal group of a junction with activated V2X flag. The calculated information is provided via the V2X data pointer from the ROAD block.

## Changes in the ASM Traffic Demo Model

**Extended object information from the object sensor 3-D**

The ASM_Traffic/Environment/Sensors/UserInterface/DISP_Plant/Object_Sensor_3D/IdxSelect_3DSensor subsystem now provides more comprehensive information for objects detected by the object sensor 3-D.

**New V2X Traffic Light Controller block**

The new V2X_TRAFFIC_LIGHT_CONTROLLER block has been added to the Traffic submodule inside the Environment module. The block calculates dynamic *V2X* data, e.g., light states of a signal group.

## Migrating to ASM Traffic Blockset 4.3

**OUTPUT_INTERFACE_ACC block**

The existing inport for the total gear ratio has been vectorized. It now contains a total gear ratio of the front and rear driveline.

An additional inport regarding the torque distribution ratio of the front driveline has also been added. These new inport signals are used to calculate a new vectorized output signal for the engine torque request. This output provides a front and a rear torque request that can be used in a BEV model with separate engines for the front and rear axle.

**OBJECT_SENSOR_3D_CALCULATION block**

There are two additional inports:

- ManeuverTime[s]
- ManeuverTime_Sampled[s]

These signals are used for calculation of the new time stamps and the new history information.

**Related topics**

Basics

Migrating ASM Models (ASM User Guide 🕮)

# Bus Manager (Stand-Alone)

**Where to go from here**

Information in this section

## New Features of the Bus Manager (Stand-Alone) 6.8

**Changes to project and application management**

The project and application management of the Bus Manger (stand-alone) is based on the project and application management of ConfigurationDesk. For ConfigurationDesk, the project and application management is enhanced. These enhancements also apply to the Bus Manger (stand-alone). For more information, refer to New General Features of ConfigurationDesk 6.8 on page 53.

**Simulating partial networking**

The Bus Manager (stand-alone) supports partial networking according to AUTOSAR 4.3. When a communication matrix specifies partial networking settings, you can use the **Partial Network Cluster Enable** feature to implement partial network clusters in the executable application. Additionally, the feature lets you enable and disable the partial network clusters at run time. However, to simulate partial networking, you also have to use blocks from the AUTOSAR Network Management Solution.

For more information, refer to Implementing Partial Networking in Executable Applications (Bus Manager (Stand-Alone) Implementation Guide 📖)

This feature is also available with Bus Manager 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

### Creating restbus configurations

The Bus Manager (stand-alone) lets you create restbus configurations. Creating a restbus configuration is useful if you want to perform a restbus simulation, e.g., for ECUs that are available as real ECUs. When creating a restbus configuration, all communication matrix elements that are required for the restbus simulation are assigned to a bus configuration in one step. You can also specify to exclude specific ECUs from the restbus configuration.

For more information, refer to Creating Restbus Configurations (Bus Manager (Stand-Alone) Implementation Guide 🕮).

This feature is also available with Bus Manager 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

### Inspecting J1939-compliant PDUs

The Bus Manager (stand-alone) lets you inspect J1939-compliant PDUs with up to 1,785 bytes. You can assign such PDUs to the Inspection part of a bus configuration and inspect the PDUs and their ISignals by using bus inspection features.

For an overview of available bus inspection features, refer to Basics on Bus Configuration Features (Bus Manager (Stand-Alone) Implementation Guide 🕮).

This feature is also available with Bus Manager 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

### Enhanced bus configuration feature

The following bus configuration features are enhanced:

**PDU Length feature** The PDU Length feature is also available as a bus inspection feature. For more information, refer to Accessing the Payload Length of CAN PDUs (Bus Manager (Stand-Alone) Implementation Guide 🕮).

**Communication Controller Enable feature** If a LIN master communication controller is disabled and re-enabled via the Communication Controller Enable feature, the active LIN schedule table always starts from the beginning. For more information, refer to Enabling and Disabling Communication Controllers (Bus Manager (Stand-Alone) Implementation Guide 🕮).

These feature enhancements are also available with Bus Manager 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

### Support of the DBName attribute of DBC files

The Bus Manager (stand-alone) supports the `DBName` attribute of DBC files. The `DBName` attribute lets you specify a name for the communication cluster that the DBC file describes. If the `DBName` attribute specifies a name, the Bus Manager (stand-alone) uses this name as the communication cluster name. If the `DBName` attribute does not specify a name, the name of the DBC file is used as the communication cluster name.

This feature is also available with Bus Manager 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

**Accessing the CAN identifier of J1939-compliant PDUs via the Bus Custom Code interface**

The Bus Manager (stand-alone) supports accessing the CAN identifier of J1939-compliant PDUs via the `DsBusCustomCodePdu_getCanFrameTriggering` function of the Bus Custom Code interface.

For more information, refer to DsBusCustomCodePdu_getCanFrameTriggering (Bus Custom Code Interface Handling 🕮).

This feature is also available with Bus Manager 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

**New examples for automating Bus Manager features**

The user documentation now provides new examples for automating Bus Manager features. The examples are compact code snippets that focus on automation tasks that are frequently performed when configuring bus communication by using the Bus Manager.

For more information, refer to Examples of Automating Bus Manager Features (ConfigurationDesk Automating Tool Handling 🕮).

## Migrating to the Bus Manager (Stand-Alone) 6.8

**Project import restricted**

As of dSPACE Release 2021-A, the Bus Manager (stand-alone) supports the direct import only of projects last saved with one of the previous seven Bus Manager (stand-alone) versions.

**Changes to the tool automation interface that might cause code malfunctions**

Some changes to the tool automation interface affect the data model and can cause code from previous Releases to malfunction. For more information, refer to New Features and Changes to the Automation Interface for Release 2021-B (ConfigurationDesk Automating Tool Handling 🕮).

# ConfigurationDesk

**Introduction**

With ConfigurationDesk, you can implement real-time applications for the SCALEXIO hardware or the MicroAutoBox III hardware.

**Where to go from here**

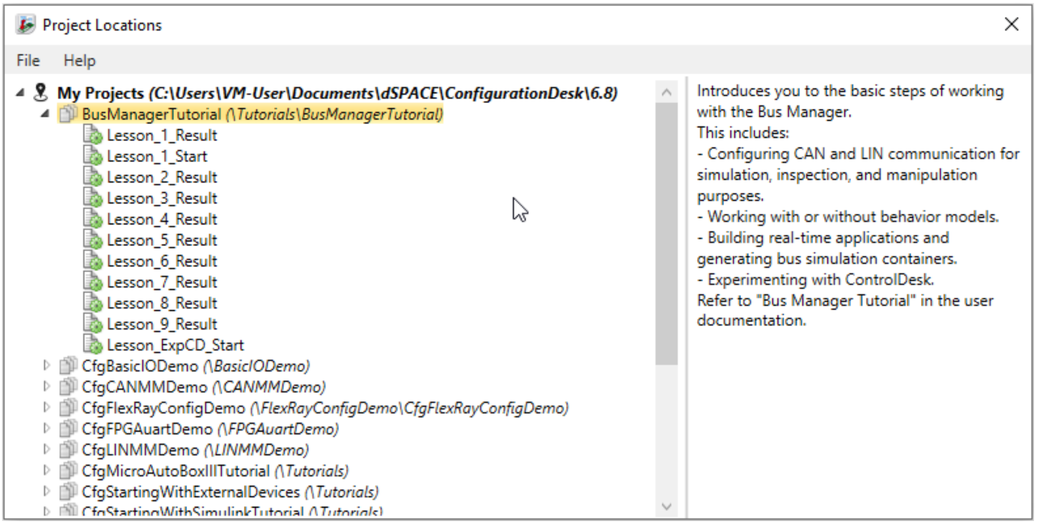
Information in this section

## New General Features of ConfigurationDesk 6.8

**Changes to project and application management**

**Project management synchronized with file system** ConfigurationDesk projects and applications are now synchronized with their folders in the file system. This means that any changes in a project or application folder are immediately displayed in the ConfigurationDesk Project Manager. You can even add a ConfigurationDesk application to a project by copying the application folder to the project folder.

**Project Location Manager** ConfigurationDesk now stores project and application folders in designated project locations, similar to project root folders in previous ConfigurationDesk versions. Projects, applications, and project locations are managed in the new Project Location Manager, which opens when you create or open a project.



My Projects (C:\Users\VM-User\Documents\dSPACE\ConfigurationDesk\6.8)
BusManagerTutorial (\Tutorials\BusManagerTutorial)
Lesson_1_Result
Lesson_1_Start
Lesson_2_Result
Lesson_3_Result
Lesson_4_Result
Lesson_5_Result
Lesson_6_Result
Lesson_7_Result
Lesson_8_Result
Lesson_9_Result
Lesson_ExpCD_Start
CfgBasicIODemo (\BasicIODemo)
CfgCANMMDemo (\CANMMDemo)
CfgFlexRayConfigDemo (\FlexRayConfigDemo\CfgFlexRayConfigDemo)
CfgFPGAuartDemo (\FPGAuartDemo)
CfgLINMMDemo (\LINMMDemo)
CfgMicroAutoBoxIIITutorial (\Tutorials)
CfgStartingWithExternalDevices (\Tutorials)
CfgStartingWithSimulinkTutorial (\Tutorials)

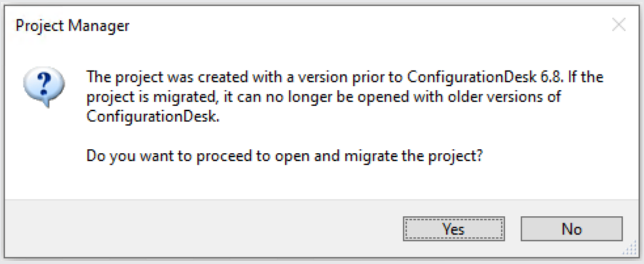
Introduces you to the basic steps of working with the Bus Manager.
This includes:
- Configuring CAN and LIN communication for simulation, inspection, and manipulation purposes.
- Working with or without behavior models.
- Building real-time applications and generating bus simulation containers.
- Experimenting with ControlDesk.
Refer to "Bus Manager Tutorial" in the user documentation.

**New file names and extensions** The file names and file name extensions of ConfigurationDesk project and application files as well as project and application backups were changed:

| File Type | Previous File Name | New File Name |
|---|---|---|
| ConfigurationDesk project | `<project name>.CDP` | `Data.CfgDeskProject` |
| ConfigurationDesk application | `<application name>.CDL` | `Data.CfgDeskApp` |
| ConfigurationDesk project backup | `<backup name>.ZIP` | `<backup name>.zip`<br>*optional:*<br>`<backup name>.CfgDeskProject.zip` |
| ConfigurationDesk application backup | `<backup name>.DSA` | `<backup name>.CfgDeskApp.zip` |

**Note**

- You can still open project and application files and backups from previous ConfigurationDesk versions. ConfigurationDesk prompts you to confirm the migration when you open a project:

  Project Manager

  The project was created with a version prior to ConfigurationDesk 6.8. If the project is migrated, it can no longer be opened with older versions of ConfigurationDesk.

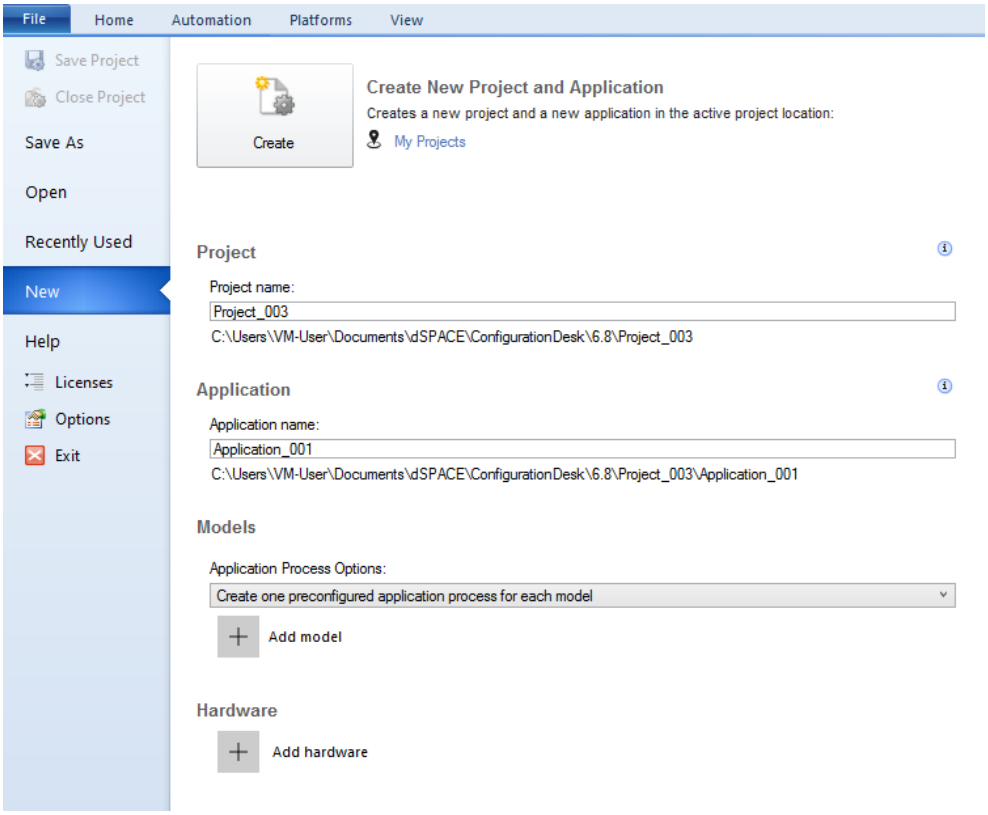
  Do you want to proceed to open and migrate the project?

  Click Yes to migrate the project and all its applications to the new file formats. Afterwards you can no longer open them with previous ConfigurationDesk versions.
- The new project and application files are stored in the respective project and application folders. They are identified via a GUID. Do not move or rename them.

**Simplified access to projects and applications** You can now access ConfigurationDesk projects and applications in additional simple ways:

- Drag project or application files to the ConfigurationDesk window to open them.
- Double-click project or application files in the file system to open them. For this purpose, you have to associate the file extension with ConfigurationDesk the first time you double-click a project or application file.

**Reworked project and application creation** The File – New backstage was adjusted to the changes in project and application management and reworked to provide a better overview:



Create New Project and Application

Creates a new project and a new application in the active project location:

My Projects

Project

Project name: Project_003

C:\Users\VM-User\Documents\dSPACE\ConfigurationDesk\6.8\Project_003

Application

Application name: Application_001

C:\Users\VM-User\Documents\dSPACE\ConfigurationDesk\6.8\Project_003\Application_001

Models

Application Process Options: Create one preconfigured application process for each model

Hardware

**Changed Python script handling** The handling of Python script files that are part of ConfigurationDesk projects or applications was changed:

- Python script files in project or application folders are now automatically part of the project and do not have to be imported.
- The script files can be deleted or renamed only in the file system.
- Fewer ribbon commands are required to handle Python script files.

**Adjusted tool automation** The tool automation API was adjusted to the project and application handling changes. Refer to New Features and Changes to the Automation Interface for Release 2021-B (ConfigurationDesk Automating Tool Handling 🕮).

### Create ControlDesk Experiment now works for multicore and multi-PU applications

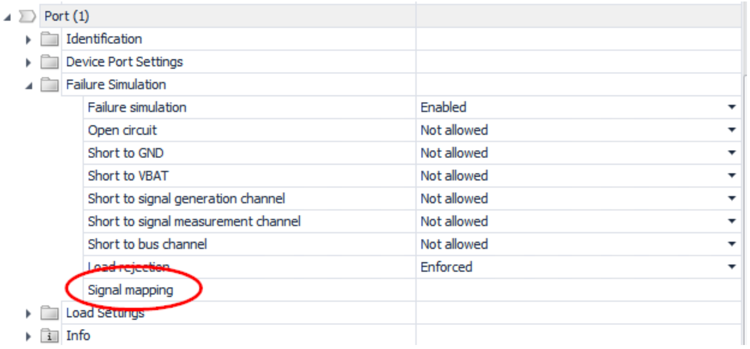
The Create ControlDesk Experiment command introduced with ConfigurationDesk 6.7 is no longer limited to ConfigurationDesk applications that contain a single application process.

Refer to How to Create a ControlDesk Experiment from a ConfigurationDesk Application (ConfigurationDesk Real-Time Implementation Guide 🕮).

This feature is also available with ConfigurationDesk 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

## Signal mapping property for device ports

ConfigurationDesk provides the new **Signal mapping** property for device ports to specify failure simulation.



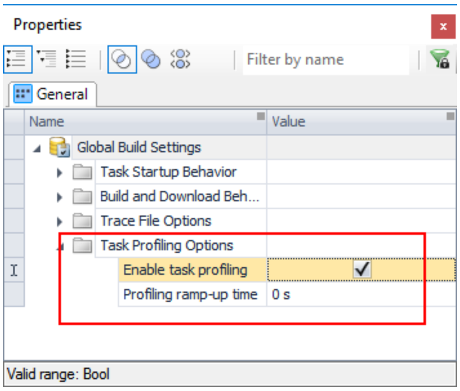
| Name | Value |
|---|---|
| Port (1) | |
| Identification | |
| Device Port Settings | |
| Failure Simulation | |
| Failure simulation | Enabled |
| Open circuit | Not allowed |
| Short to GND | Not allowed |
| Short to VBAT | Not allowed |
| Short to signal generation channel | Not allowed |
| Short to signal measurement channel | Not allowed |
| Short to bus channel | Not allowed |
| Load rejection | Enforced |
| Signal mapping | |
| Load Settings | |
| Info | |

Here you can enter a string-based description for the failure simulation signal. Multiple devices can have different port names but identical functionality from a failure simulation perspective. The description can help you to identify a unique failure simulation signal regardless of the device port names.

This feature is also available with ConfigurationDesk 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

## New task profiling options

**Options for calculating turnaround times of runnable functions and tasks** ConfigurationDesk provides the **Enable task profiling** and **Profiling ramp-up time** build settings.



| Name | Value |
|---|---|
| Global Build Settings | |
| Task Startup Behavior | |
| Build and Download Beh... | |
| Trace File Options | |
| Task Profiling Options | |
| Enable task profiling | ✓ |
| Profiling ramp-up time | 0 s |

Valid range: Bool

If you select the **Enable task profiling** checkbox, ConfigurationDesk now generates the following additional entries in the TRC file during the build process:

- The maximum turnaround time of each task
- The turnaround time of each runnable function assigned to a task
- The maximum turnaround time of each runnable function assigned to a task

The **Profiling ramp-up time** lets you specify a time during which the maximum task turnaround time is not calculated. The task profiling options can be helpful, for example, to find out the cause of task overruns via experiment software. For more information, refer to Specifying Options for the Build Process (ConfigurationDesk Real-Time Implementation Guide 📖).

This feature is also available with ConfigurationDesk 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

**New TRC file entry for memory usage** ConfigurationDesk now generates the `MemoryUsage` variable for the memory usage of a real-time application in the `Diagnostics` group of the TRC file. Refer to Diagnostics Group (ConfigurationDesk Real-Time Implementation Guide 🕮).

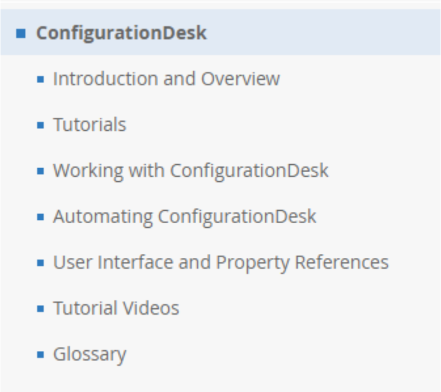
**New behavior when adding signal chain elements to working views**

When you add signal chain elements to working views via a CAFX file, the IDs of the added model port blocks are preserved. You can then resolve the model port blocks by analyzing the related Simulink model. If there are already model port blocks with the same IDs in the application, this would cause a conflict due to duplicate IDs. In this case, model port blocks are added as new, unresolved blocks with new IDs. These blocks are not displayed in the Model Browser.

This feature is also available with ConfigurationDesk 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

**Improved documentation structure**

The structure of the ConfigurationDesk documentation was improved to provide easier access to the required information for new and experienced users:



- **ConfigurationDesk**
  - Introduction and Overview
  - Tutorials
  - Working with ConfigurationDesk
  - Automating ConfigurationDesk
  - User Interface and Property References
  - Tutorial Videos
  - Glossary

## New Features Concerning I/O Functionality and Hardware Support

**New function block types**

**FuSa Memory Monitoring** The FuSa Memory Monitoring function block is a memory-related monitor that enhances functional safety in your application. If you use this function block, a functional safety (FuSa) error can be triggered directly by the monitor if a memory error occurs.

For more information, refer to FuSa Memory Monitoring (ConfigurationDesk I/O Function Implementation Guide 🕮).

**Enhanced function block types**

**Lambda Probe In** The Lambda Probe In function block type now supports LSU 5.1 lambda probes.

For more information, refer to Lambda Probe In (ConfigurationDesk I/O Function Implementation Guide 🕮).

**CAN** The CAN function block lets you virtualize CAN controllers. This lets you simulate bus communication even if no board or module providing a CAN-compatible channel type is installed in your SCALEXIO or MicroAutoBox III system, or if no other CAN controller is connected.

For more information, refer to Configuring the Basic Functionality (CAN) (ConfigurationDesk I/O Function Implementation Guide 🕮).

This feature is also available with ConfigurationDesk 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

### Changing the name of the Power On Signal In function block type

The Power On Signal In function block type was renamed Shutdown Request to better express the functionality of the block.

After migrating an existing ConfigurationDesk application containing a Power On Signal In function block, you have to observe the following points:

- After migrating a project, existing function block names are not changed. For example: Even if you use the former default name for the function block (Power On Signal In (1)), this name is still used in the migrated ConfigurationDesk application.
- After you build a real-time application with ConfigurationDesk 6.8, you might have to update your ControlDesk experiment manually. This means, if you use the tunable properties of the Shutdown Request function block in the ControlDesk layout (via a TRC file), you have to manually reassign the updated variables to the instruments in the layout.

This feature is also available with ConfigurationDesk 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

### New features concerning hardware support

**SCALEXIO** If you create a hardware topology from scratch, you can now specify the variant of the SCALEXIO Processing Unit. This is useful for SCALEXIO Processing Units that provide more Ethernet controllers and/or processor cores than the default SCALEXIO Processing Unit. The default SCALEXIO Processing Unit provides one Ethernet controller and four processor cores (three cores available for real-time applications).

## New Features of the Bus Manager in ConfigurationDesk

### Simulating partial networking

The Bus Manager supports partial networking according to AUTOSAR 4.3. When a communication matrix specifies partial networking settings, you can use the Partial Network Cluster Enable feature to implement partial network clusters in the executable application. Additionally, the feature lets you enable and disable the partial network clusters at run time. However, to simulate partial networking, you also have to use blocks from the AUTOSAR Network Management Solution.

For more information, refer to Implementing Partial Networking in Executable Applications (ConfigurationDesk Bus Manager Implementation Guide 📖)

This feature is also available with ConfigurationDesk 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

### Creating restbus configurations

The Bus Manager lets you create restbus configurations. Creating a restbus configuration is useful if you want to perform a restbus simulation, e.g., for ECUs that are available as real ECUs. When creating a restbus configuration, all communication matrix elements that are required for the restbus simulation are assigned to a bus configuration in one step. You can also specify to exclude specific ECUs from the restbus configuration.

For more information, refer to Creating Restbus Configurations (ConfigurationDesk Bus Manager Implementation Guide 📖).

This feature is also available with ConfigurationDesk 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

### Inspecting J1939-compliant PDUs

The Bus Manager lets you inspect J1939-compliant PDUs with up to 1,785 bytes. You can assign such PDUs to the Inspection part of a bus configuration and inspect the PDUs and their ISignals by using bus inspection features.

For an overview of available bus inspection features, refer to Basics on Bus Configuration Features (ConfigurationDesk Bus Manager Implementation Guide 📖).

This feature is also available with ConfigurationDesk 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

### Enhanced bus configuration feature

The following bus configuration features are enhanced:

**PDU Length feature** The PDU Length feature is also available as a bus inspection feature. For more information, refer to Accessing the Payload Length of CAN PDUs (ConfigurationDesk Bus Manager Implementation Guide 📖).

**Communication Controller Enable feature** If a LIN master communication controller is disabled and re-enabled via the Communication Controller Enable feature, the active LIN schedule table always starts from the beginning. For more information, refer to Enabling and Disabling Communication Controllers (ConfigurationDesk Bus Manager Implementation Guide 📖).

These feature enhancements are also available with ConfigurationDesk 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

### Support of the DBName attribute of DBC files

The Bus Manager supports the `DBName` attribute of DBC files. The `DBName` attribute lets you specify a name for the communication cluster that the DBC file describes. If the `DBName` attribute specifies a name, the Bus Manager uses this name as the communication cluster name. If the `DBName` attribute does not specify a name, the name of the DBC file is used as the communication cluster name.

This feature is also available with ConfigurationDesk 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

**Accessing the CAN identifier of J1939-compliant PDUs via the Bus Custom Code interface**

The Bus Manager supports accessing the CAN identifier of J1939-compliant PDUs via the `DsBusCustomCodePdu_getCanFrameTriggering` function of the Bus Custom Code interface.

For more information, refer to DsBusCustomCodePdu_getCanFrameTriggering (Bus Custom Code Interface Handling 📖).

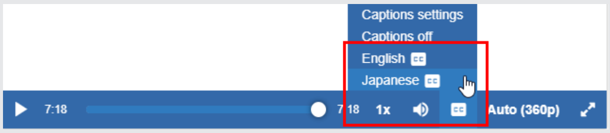
This feature is also available with ConfigurationDesk 6.7p1 (Release 2021-A Implementation Software Service Pack 1).

**Tutorial video introducing the Bus Manager**

A new tutorial video is available that introduces you to the Bus Manager. It shows the basic steps for configuring CAN and LIN communication, working with a behavior model, specifying the hardware access, and building a real-time application. You can access the video in dSPACE Help, on the **Start Page** of ConfigurationDesk, and on the dSPACE website.

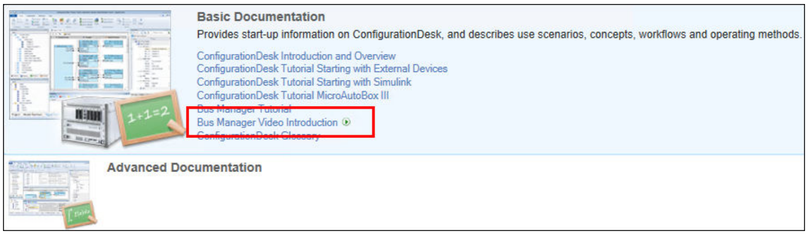
**Tip**

In dSPACE Help, you can switch between English and Japanese video subtitles. You can switch the subtitle language on the controlbar of the video player.



Refer to:

- dSPACE Help: Overview of the Bus Manager (ConfigurationDesk Bus Manager Implementation Guide 📖)
- **Start Page** of ConfigurationDesk: **Basic Documentation - Bus Manager Video Introduction**



Basic Documentation

Provides start-up information on ConfigurationDesk, and describes use scenarios, concepts, workflows and operating methods.

ConfigurationDesk Introduction and Overview
ConfigurationDesk Tutorial Starting with External Devices
ConfigurationDesk Tutorial Starting with Simulink
ConfigurationDesk Tutorial MicroAutoBox III
Bus Manager Tutorial
Bus Manager Video Introduction
ConfigurationDesk Glossary

Advanced Documentation

- dSPACE website: www.dspace.com/go/BusManagerCfgDesk_Introduction

**New examples for automating Bus Manager features**

The user documentation now provides new examples for automating Bus Manager features. The examples are compact code snippets that focus on automation tasks that are frequently performed when configuring bus communication by using the Bus Manager.

For more information, refer to Examples of Automating Bus Manager Features (ConfigurationDesk Automating Tool Handling 🕮).

## Supported Container File Versions

**Supported SIC file versions** ConfigurationDesk 6.8 supports SIC file versions as listed below:

| SIC Files Created With ... | MATLAB Release |
|---|---|
| dSPACE Release 2021-B:<br>▪ Model Interface Package for Simulink 4.6<br>▪ TargetLink 5.2 | R2021b, R2021a, R2020b, R2020a |
| dSPACE Release 2021-A:<br>▪ Model Interface Package for Simulink 4.5 | R2021a, R2020b, R2020a, R2019b |
| dSPACE Release 2020-B:<br>▪ Model Interface Package for Simulink 4.4<br>▪ TargetLink 5.1 | R2020b, R2020a, R2019b, R2019a |
| dSPACE Release 2020-A:<br>▪ Model Interface Package for Simulink 4.3 | R2020a, R2019b, R2019a, R2018b |
| dSPACE Release 2019-B:<br>▪ Model Interface Package for Simulink 4.2 | R2018a |
| dSPACE Release 2019-A:<br>▪ Model Interface Package for Simulink 4.1 | R2017b |
| dSPACE Release 2018-B:<br>▪ Model Interface Package for Simulink 4.0 | R2017a |
| dSPACE Release 2018-A:<br>▪ Model Interface Package for Simulink 3.6 | R2016b |
| dSPACE Release 2017-B:<br>▪ Model Interface Package for Simulink 3.5 | R2016a |
| dSPACE Release 2017-A:<br>▪ Model Interface Package for Simulink 3.4 | R2015b |

**Note**

You can use only SIC files that were generated with the Model Interface Package for Simulink for the `dsrt.tlc` system target file to build real-time applications with ConfigurationDesk. SIC files generated for the `dsrt64.tlc` system target file can be used in ConfigurationDesk only to generate BSC files for VEOS running on a Linux operating system.

**Limitations for earlier SIC file versions in ConfigurationDesk scenarios** SIC files created with the Model Interface Package for Simulink version 3.4 ... 4.1 are not supported in the following ConfigurationDesk scenarios:

- In multimodel application processes.
- For building real-time applications that use Real-Time Testing.

**Limitations for Simulink behavior models underlying earlier SIC files versions** The following limitations apply to Simulink behavior models underlying SIC files created with the Model Interface Package for Simulink version 3.4 … 4.2:

- The Simulink behavior model must not contain blocks from the following blocksets:
  - Blocks of ASM

    Real-time applications that contain such SIC files cannot be used with ModelDesk.
  - RTI FPGA Programming Blockset
  - MotionDesk Blockset

    Real-time applications that contain such SIC files cannot be used with MotionDesk.
  - Blocks of any dSPACE Solution.
- Compiled objects contained in SIC files must match the target platform. SIC files with compiled objects that were created at a time when the current target platforms did not exist are not supported. In this case, it is not possible to use compiled objects, e.g., for IP protection scenarios.

**Supported BSC file versions**

ConfigurationDesk 6.8 supports BSC files that were generated with the Bus Manager of the current Release, i.e., BSC file version 1.10.

**Supported V-ECU implementation container versions**

ConfigurationDesk 6.8 supports V-ECU implementation container versions as listed below:

| V-ECU Implementations Created With... | V-ECU Implementation Version |
|---|---|
| dSPACE Release 2021-B:<br>▪ SystemDesk 5.6<br>▪ TargetLink 5.2 | 3.0 |
| dSPACE Release 2021-A:<br>▪ SystemDesk 5.5 | 3.0 |
| dSPACE Release 2020-B:<br>▪ SystemDesk 5.5<br>▪ TargetLink 5.1 | 3.0<br>As of version 3.0, *VECU* is the new V-ECU implementation file format. |
| dSPACE Release 2020-A:<br>▪ SystemDesk 5.4 | 2.10 |

**Supported Functional Mock-up Unit versions**

ConfigurationDesk 6.8 supports Functional Mock-up Units (FMUs) that comply with the following versions of the FMI standard:

- 2.0
- 2.0.1

**Supported EIC file versions**

ConfigurationDesk 6.8 supports EIC file versions as listed below:

| EIC Files Created With ... | EIC Version |
|---|---|
| dSPACE Release 2021-B<br>(ECU Interface Manager 2.10) | 4.0.0 |
| dSPACE Release 2021-A<br>(ECU Interface Manager 2.9) | 4.0.0 |
| dSPACE Release 2020-B<br>(ECU Interface Manager 2.8) | 4.0.0 |
| dSPACE Release 2020-A<br>(ECU Interface Manager 2.7) | 4.0.0 |
| dSPACE Release 2019-B<br>(ECU Interface Manager 2.6) | 4.0.0 |
| dSPACE Release 2019-A<br>(ECU Interface Manager 2.5) | 3.0.0 |
| dSPACE Release 2018-B<br>(ECU Interface Manager 2.4) | 3.0.0 |
| dSPACE Release 2018-A<br>(ECU Interface Manager 2.3) | 2.0.0 |
| dSPACE Release 2017-B<br>(ECU Interface Manager 2.2) | 1.0.0 |
| dSPACE Release 2017-A<br>(ECU Interface Manager 2.1) | 1.0.0 |
| dSPACE Release 2016-B<br>(ECU Interface Manager 2.0p1) | 1.0.0 |

However, MicroAutoBox III systems only support EIC files as of version 4.0.0.

## Migrating to ConfigurationDesk 6.8

**Project import restricted**

As of dSPACE Release 2021-A, ConfigurationDesk supports the direct import only of projects last saved with one of the previous seven ConfigurationDesk versions.

**Discontinuation of SCALEXIO Ethernet Solution**

The SCALEXIO Ethernet Solution is discontinued as follows:

- The end-of-life date was January 31, 2021.
- Customers with a Software Maintenance Service contract who work with dSPACE Release 2018-B will be automatically migrated to the new ConfigurationDesk UDP/TCP function blocks.

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

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

### Changes to serial communication

The following changes in dSPACE Release 2021-A affect serial communication.

**SENT communication** An invalid value at the Sequence port for selecting sequence definitions from the application-specific protocol now stops the transmission/receipt of SENT messages. Until Release 2020-B, invalid values were ignored and the function block used the last valid value.

**SPI communication** The new Word separation property lets you select the method for separating consecutive data words within an SPI cycle. Until Release 2020-A, you selected the method by setting the Time between words property to 0 s (chip select inactive between words) or to a value greater than 0 s (transmission pause between words).

For more information on the property, refer to Configuring SPI Cycles (SPI Master) (ConfigurationDesk I/O Function Implementation Guide 🕮).

### Changes to I/O Ethernet communication

The following changes in dSPACE Release 2020-B affect I/O Ethernet communication.

**Receiving UDP messages** The Received Bytes function port of the UDP Receive function block now continuously provides the number of valid data bytes that are provided by the Data Vector function port. Until Release 2020-A, the Received Bytes function port provided the number of valid data bytes only during the model step in which the function block provides new data.

**Unsupported I/O Ethernet feature** A SCALEXIO real-time application can suppress Internet group management protocol (IGMP) messages for I/O Ethernet communication.

With dSPACE Release 2020-B, dSPACE introduced a Linux-based operating system for the SCALEXIO Processing Unit and DS6001 Processor Board. The Linux-based operating system does not support the suppression of IGMP messages.

### FPGA custom function blocks with APU functionality

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

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

**Changes to the tool automation interface that might cause code malfunctions**

Some changes to the tool automation interface affect the data model and can cause code from previous Releases to malfunction. For more information, refer to New Features and Changes to the Automation Interface for Release 2021-B (ConfigurationDesk Automating Tool Handling 🕮).

# ControlDesk

**Where to go from here**

Information in this section

# New Features of ControlDesk 7.5

**Where to go from here**

Information in this section

## New Features of Platform Management and Platforms/Devices (ControlDesk 7.5)

**cmdloader: Configuring and registering dSPACE platforms using a configuration file**

The `cmdloader` command line tool now lets you configure and register dSPACE platforms. For this purpose, you have to specify a JSON file with the required platform configuration by using the `cmdloader` tool.

For more information, refer to Registering Platforms and Handling Applications via cmdloader (ControlDesk Platform Management 📖).

## New Variable Management Features (ControlDesk 7.5)

**Improved management and handling of large variable descriptions**

The management and handling of large variable descriptions in ControlDesk 7.5 is significantly faster than in ControlDesk 7.4 and earlier.

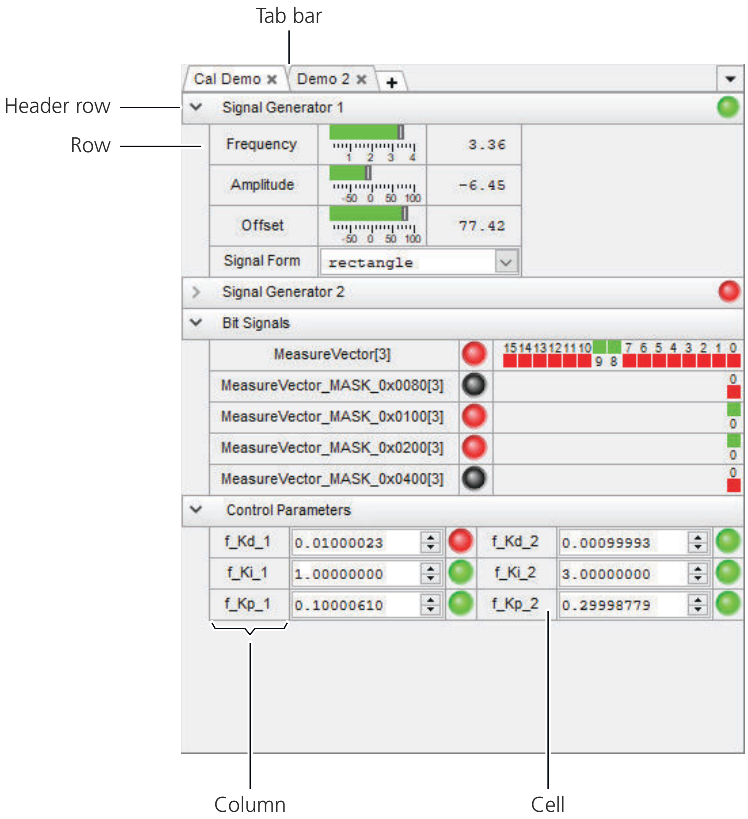
However, main memory usage will increase, especially when you import such large variable descriptions. A main memory of 32 GB RAM or more is therefore recommended.

# New Instrument Features (ControlDesk 7.5)

**New Hierarchy Array**

The new Hierarchy Array lets you calibrate scalar parameters and display scalar measurement variable values. In the instrument, you can group the variable display freely on different hierarchy levels. You can select different instrument types to visualize the individual variables, such as the Alphanumeric Input, Slider, or the Display.

The following illustration shows an example:



Frequency 3.36
Amplitude -6.45
Offset 77.42
Signal Form rectangle

f_Kd_1 0.01000023
f_Ki_1 1.00000000
f_Kp_1 0.10000610
f_Kd_2 0.00099993
f_Ki_2 3.00000000
f_Kp_2 0.29998779

For more information, refer to Basics of Handling the Hierarchy Array (ControlDesk Instrument Handling 🕮).

# New Bus Navigator Features (ControlDesk 7.5)

**Bus instruments (Bus Manager): Support for signals that use text tables**

Bus instruments for bus communication modeled with the Bus Manager now support signals that use text tables or a combination of a linear scaling and text tables. These instruments display converted values, i.e., alphanumerical values are used for linear scalings and text strings for text table scalings.

For more information, refer to Bus Instruments (Bus Manager) (ControlDesk Bus Navigator 🕮).

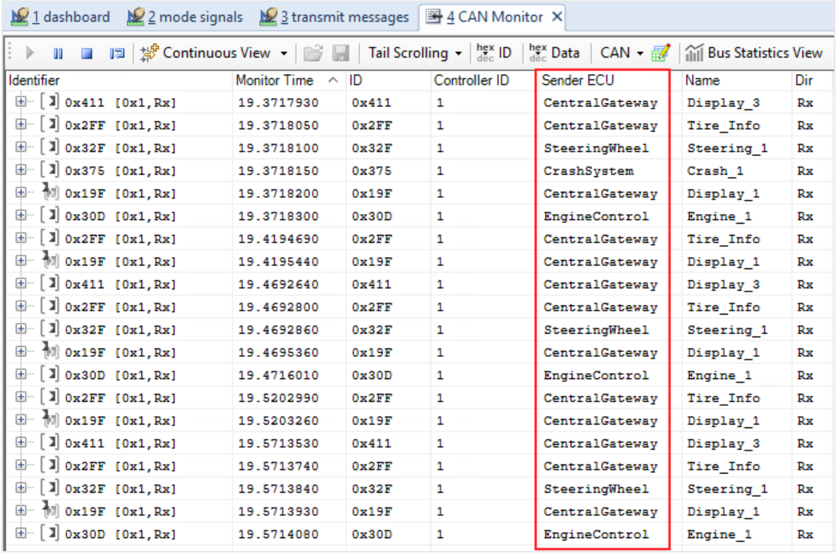
### SOME/IP service discovery: Display of decoded item names in the Decoding view

The Ethernet Decoding view, which is part of the monitoring list, now also displays decoded names of services, methods, events, and event groups.

Refer to Monitoring List (ControlDesk Bus Navigator 🕮).

### CAN, LIN: Display of the sender ECU in the monitoring list

For CAN and LIN, the Bus Monitoring list now lets you display the Sender ECU.



| Identifier | Monitor Time | ID | Controller ID | Sender ECU | Name | Dir |
|---|---|---|---|---|---|---|
| 0x411 [0x1,Rx] | 19.3717930 | 0x411 | 1 | CentralGateway | Display_3 | Rx |
| 0x2FF [0x1,Rx] | 19.3718050 | 0x2FF | 1 | CentralGateway | Tire_Info | Rx |
| 0x32F [0x1,Rx] | 19.3718100 | 0x32F | 1 | SteeringWheel | Steering_1 | Rx |
| 0x375 [0x1,Rx] | 19.3718150 | 0x375 | 1 | CrashSystem | Crash_1 | Rx |
| 0x19F [0x1,Rx] | 19.3718200 | 0x19F | 1 | CentralGateway | Display_1 | Rx |
| 0x30D [0x1,Rx] | 19.3718300 | 0x30D | 1 | EngineControl | Engine_1 | Rx |
| 0x2FF [0x1,Rx] | 19.4194690 | 0x2FF | 1 | CentralGateway | Tire_Info | Rx |
| 0x19F [0x1,Rx] | 19.4195440 | 0x19F | 1 | CentralGateway | Display_1 | Rx |
| 0x411 [0x1,Rx] | 19.4692640 | 0x411 | 1 | CentralGateway | Display_3 | Rx |
| 0x2FF [0x1,Rx] | 19.4692800 | 0x2FF | 1 | CentralGateway | Tire_Info | Rx |
| 0x32F [0x1,Rx] | 19.4692860 | 0x32F | 1 | SteeringWheel | Steering_1 | Rx |
| 0x19F [0x1,Rx] | 19.4695360 | 0x19F | 1 | CentralGateway | Display_1 | Rx |
| 0x30D [0x1,Rx] | 19.4716010 | 0x30D | 1 | EngineControl | Engine_1 | Rx |
| 0x2FF [0x1,Rx] | 19.5202990 | 0x2FF | 1 | CentralGateway | Tire_Info | Rx |
| 0x19F [0x1,Rx] | 19.5203260 | 0x19F | 1 | CentralGateway | Display_1 | Rx |
| 0x411 [0x1,Rx] | 19.5713530 | 0x411 | 1 | CentralGateway | Display_3 | Rx |
| 0x2FF [0x1,Rx] | 19.5713740 | 0x2FF | 1 | CentralGateway | Tire_Info | Rx |
| 0x32F [0x1,Rx] | 19.5713840 | 0x32F | 1 | SteeringWheel | Steering_1 | Rx |
| 0x19F [0x1,Rx] | 19.5713930 | 0x19F | 1 | CentralGateway | Display_1 | Rx |
| 0x30D [0x1,Rx] | 19.5714080 | 0x30D | 1 | EngineControl | Engine_1 | Rx |

For more information, refer to Monitoring List (ControlDesk Bus Navigator 🕮).

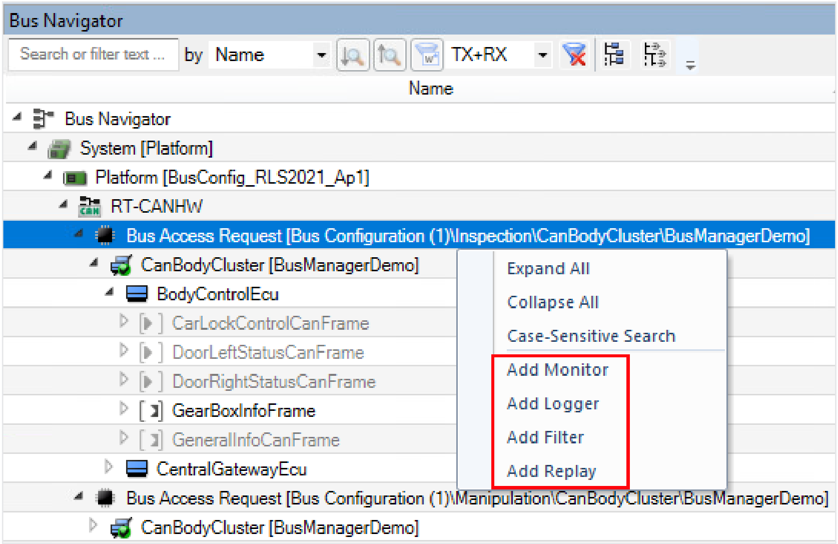
### CAN, LIN: Improvement for monitoring, logging, and replay in connection with Bus Manager applications

Up to and including ControlDesk 7.4, to monitor, log, and replay bus communication modeled with the Bus Manager, it was necessary to use a monitoring device in connection with a bus interface of SCALEXIO, MicroAutoBox III, or VEOS.

As of ControlDesk 7.5, you can also create filtered/unfiltered monitor, logger, and replay nodes directly for a controller of SCALEXIO, MicroAutoBox III, or VEOS, without having to use a monitoring device. You can also display and log bus statistics this way.

The improvement is available in the graphical user interface and in the tool automation of the Bus Navigator.

The following illustration shows the corresponding context menu commands of a CAN controller of VEOS as an example.


Bus Navigator
Search or filter text ...
by
Name
TX+RX
Name
Bus Navigator
System [Platform]
Platform [BusConfig_RLS2021_Ap1]
RT-CANHW
Bus Access Request [Bus Configuration (1)\Inspection\CanBodyCluster\BusManagerDemo]
CanBodyCluster [BusManagerDemo]
BodyControlEcu
CarLockControlCanFrame
DoorLeftStatusCanFrame
DoorRightStatusCanFrame
GearBoxInfoFrame
GeneralInfoCanFrame
CentralGatewayEcu
Bus Access Request [Bus Configuration (1)\Manipulation\CanBodyCluster\BusManagerDemo]
CanBodyCluster [BusManagerDemo]
Expand All
Collapse All
Case-Sensitive Search
Add Monitor
Add Logger
Add Filter
Add Replay

For more information, refer to Dependencies Regarding Monitoring, Logging, and Filtering Bus Communication (ControlDesk Bus Navigator 🕮) and Basics on Replaying CAN and Ethernet Bus Communication (ControlDesk Bus Navigator 🕮).

### Support of R20-11 ARXML files

The ControlDesk Bus Navigator now also supports AUTOSAR system description (ARXML) files according to the R20-11 version of the AUTOSAR system template.

Refer to Variable Descriptions Supported by ControlDesk (ControlDesk Variable Management 🕮).

### Related topics

References

Monitoring List (ControlDesk Bus Navigator 🕮)

## New Automation Features (ControlDesk 7.5)

### Automating the progress bar

For automated determinate and indeterminate actions in ControlDesk, you can create a progress bar that visualizes the action progress. This is especially useful for time-consuming actions, letting you provide status information to users who are actively working with ControlDesk. The ControlDesk automation interface now provides the `IXaProgressBarManagement` and `IXaProgressBarManagementEvents` interfaces for this purpose.

The ControlDesk installation also provides the `ProgressbarHandling.py` demo script that shows you how to automate the progress bar handling.

For more information, refer to Automating Progress Bar Handling (ControlDesk Automation 🕮).

### Copying demo projects

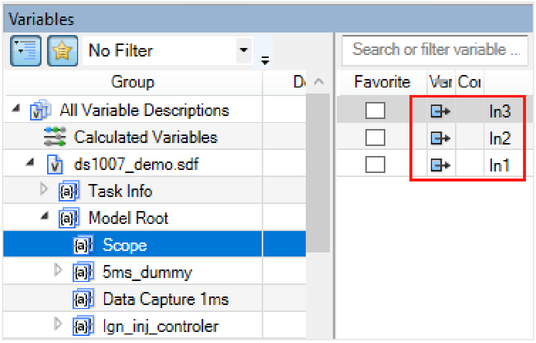
The `UserSettings / IXaUserSettings <<Interface>>` automation interface now provides the `CopyDemos` method.

As of ControlDesk 7.5, you have to copy the ControlDesk demo projects to the user-specific Documents folder manually since this is no longer done automatically during the first start of ControlDesk after installation. For more information, refer to Changed availability of demo projects after installation on page 79.

### Adding multiple signals to the measurement signal list

The `MeasurementSignals / IXaMeasurementSignals <<Collection>>` automation interface now provides the `AddMultiple` method to add a list of signals to the measurement signal list at once.

**Example** Your experiment contains a platform named `Platform` to which the `ds1007_demo.sdf` variable description file is assigned. You want to add the variables in the Scope group to the measurement signal list.



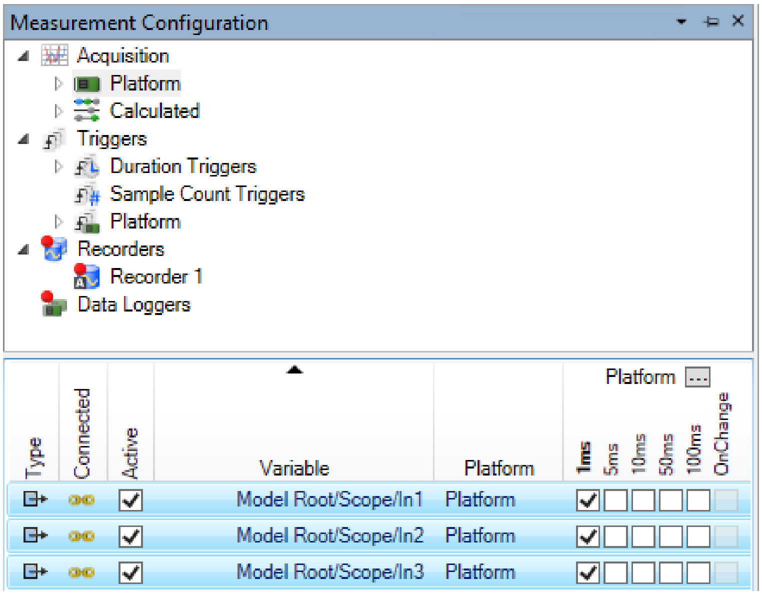
| Favorite | Var | Col |
|---|---|---|
| □ | | In3 |
| □ | | In2 |
| □ | | In1 |

Enter the following command in the ControlDesk Interpreter controlbar:

```
Application.MeasurementDataManagement.MeasurementConfiguration.Signals.AddMultiple(['Platform()://Model Root/Scope/In1',
                                                                                   'Platform()://Model Root/Scope/In2',
                                                                                   'Platform()://Model Root/Scope/In3'])
```

As a result:

- The signals are added to the measurement signal list as shown in the following illustration. Because no measurement raster was specified when adding the variables to the list, they are added to the default raster.

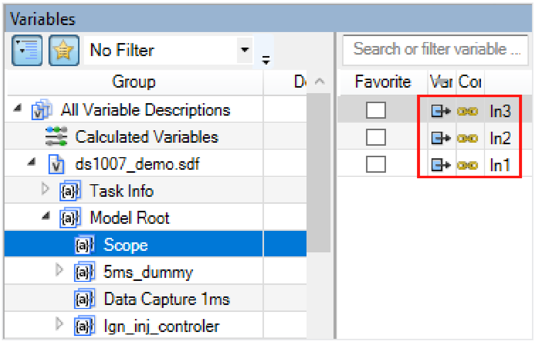


Measurement Configuration

- Acquisition
  - Platform
  - Calculated
- Triggers
  - Duration Triggers
  - Sample Count Triggers
  - Platform
- Recorders
  - Recorder 1
- Data Loggers

| Type | Connected | Active | Variable | Platform | Platform 1ms | 5ms | 10ms | 50ms | 100ms | OnChange |
|---|---|---|---|---|---|---|---|---|---|---|
|  |  | ☑ | Model Root/Scope/In1 | Platform | ☑ | □ | □ | □ | □ |  |
|  |  | ☑ | Model Root/Scope/In2 | Platform | ☑ | □ | □ | □ | □ |  |
|  |  | ☑ | Model Root/Scope/In3 | Platform | ☑ | □ | □ | □ | □ |  |

- The Variables controlbar also indicates that the signals are now part of the measurement signal list.



Variables

No Filter

Search or filter variable ...

- All Variable Descriptions
  - Calculated Variables
  - ds1007_demo.sdf
    - Task Info
    - Model Root
      - Scope
      - 5ms_dummy
      - Data Capture 1ms
      - Ign_inj_controler

| Favorite | Var Coı |  |
|---|---|---|
| □ |  | In3 |
| □ |  | In2 |
| □ |  | In1 |

- The Interpreter controlbar displays the return value of the `AddMultiple` method. The return value is the number of entries in the method's parameter list that ControlDesk has processed without errors.

## Related topics

Basics

Demo Projects (ControlDesk Introduction and Overview 📖)

References

MeasurementSignals / IXaMeasurementSignals <<Collection>> (ControlDesk Automation 📖)
ProgressBarManagement / IXaProgressBarManagement <<Interface>> (ControlDesk Automation 📖)
ProgressBarManagementEvents / IXaProgressBarManagementEvents <<EventInterface>> (ControlDesk Automation 📖)
UserSettings / IXaUserSettings <<Interface>> (ControlDesk Automation 📖)

# Migrating to ControlDesk 7.5

**Where to go from here**

Information in this section

## Discontinuations in ControlDesk

**Discontinuations as of ControlDesk 7.5**

**Access to calculated variables via MCD-3 automation** As of Version 7.5, the ControlDesk MCD-3 automation interface no longer supports the access to calculated variables. As a consequence, you have to adapt MCD-3 automation scripts that involve the access to calculated variables.

**Downsampling when saving displayed Time Plotter data** As of Version 7.5, ControlDesk no longer lets you specify downsampling for saving data currently displayed in a Time Plotter to a new measurement data file.

For more information, refer to Save Displayed Data as New Measurement (ControlDesk Instrument Handling 🕮).

**Discontinuations in ControlDesk (dSPACE Release 2022-A)**

**Variable Editor** As of dSPACE Release 2022-A, the Variable Editor will no longer be available.

**End of software support for discontinued dSPACE hardware**

For information on the end of software support for discontinued dSPACE hardware, refer to Discontinuations on page 17.

# Migrating to ControlDesk 7.5

### Introduction

To migrate from ControlDesk 7.4 to ControlDesk 7.5 and reuse existing experiments, you might have to carry out the following migration steps.

> **Note**
>
> To migrate to ControlDesk 7.5 from versions earlier than 7.4, you also have to perform the migration steps of the intervening ControlDesk versions.

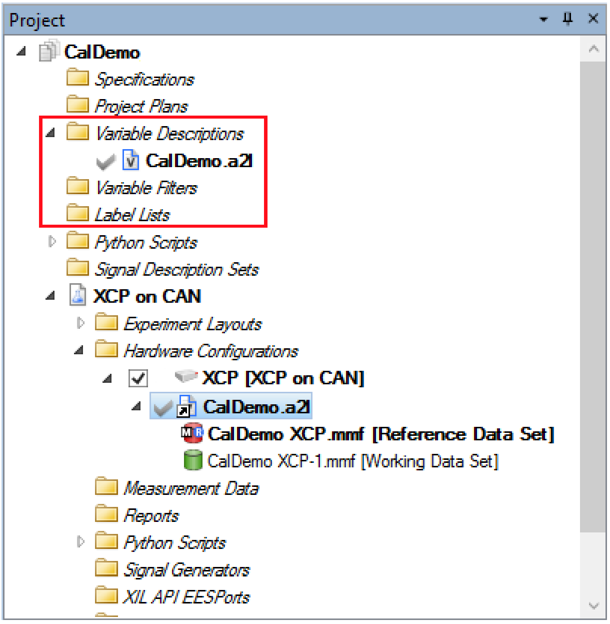
### Changed handling of variable descriptions

There are some changes to the handling and management of variable descriptions in ControlDesk 7.5. The changes are due to the improvements to the handling of large variable descriptions:

- Up to and including ControlDesk 7.4, variable descriptions were managed in the **Variable Descriptions** folder of *the project*. A variable description assigned to a platform/device was actually a reference to the variable description of the project. Such a reference was marked by an additional ↗ symbol in the **Project** controlbar.

  Consequently, the **Variable Filters** and the **Label Lists** folders were also part of the project.

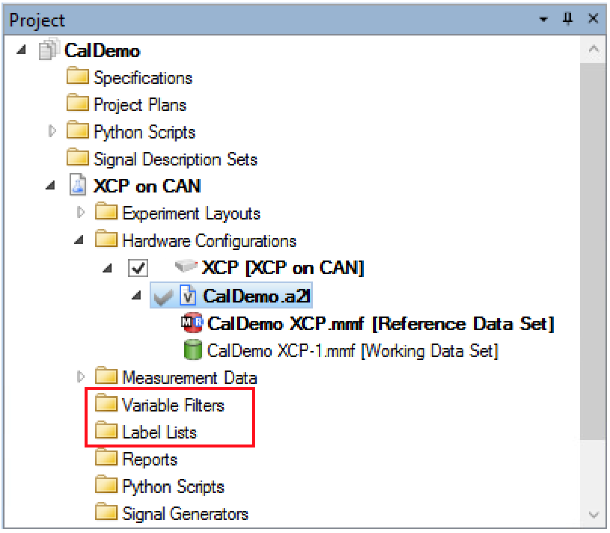
  The following illustration shows an example:

- As of ControlDesk 7.5, the project no longer contains a **Variable Descriptions** folder. Instead, variable descriptions belong to a platform or device of an *experiment*. In the file system, variable descriptions are stored in the `Variable Descriptions` folder of the experiment.

  The **Variable Filters** and the **Label Lists** folders are also part of the experiment.

The following illustration shows an example:



- CalDemo
  - Specifications
  - Project Plans
  - Python Scripts
  - Signal Description Sets
  - XCP on CAN
    - Experiment Layouts
    - Hardware Configurations
      - ☑ XCP [XCP on CAN]
        - CalDemo.a2l
          - CalDemo XCP.mmf [Reference Data Set]
          - CalDemo XCP-1.mmf [Working Data Set]
    - Measurement Data
    - Variable Filters
    - Label Lists
    - Reports
    - Python Scripts
    - Signal Generators

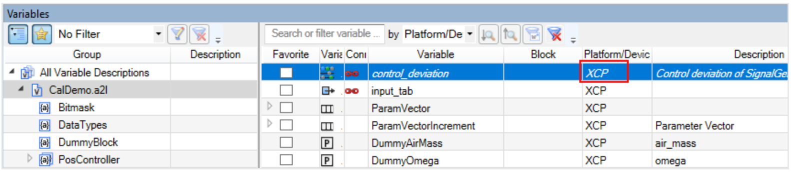
For more information on the new component for variable management, refer to New Variable Management Features (ControlDesk 7.5) on page 68.

## Changed handling of calculated variables

The handling and management of calculated variables was changed in ControlDesk 7.5:

- Up to and including Version 7.4, ControlDesk assigned a calculated variable to the variable description to which it was initially added. This was done even if the input signals of the calculated variable were derived from another variable description.

  The following illustration shows a calculated variable assigned to the `CalDemo.a2l` variable description as an example. The A2L file belongs to the device named XCP, and the calculated variable was implicitly assigned to this device:



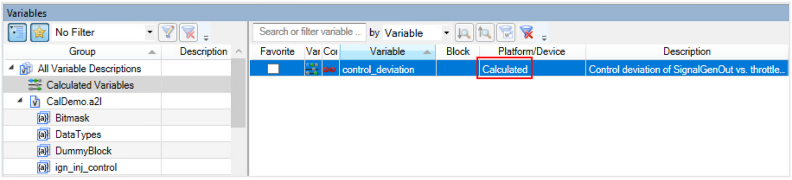
| Group | Description |
|---|---|
| All Variable Descriptions | |
| CalDemo.a2l | |
| Bitmask | |
| DataTypes | |
| DummyBlock | |
| PosController | |

| Favorite | Varia | Conı | Variable | Block | Platform/Devic | Description |
|---|---|---|---|---|---|---|
| □ | | | control_deviation | | XCP | Control deviation of SignalGe |
| □ | | | input_tab | | XCP | |
| □ | | | ParamVector | | XCP | |
| □ | | | ParamVectorIncrement | | XCP | Parameter Vector |
| □ | P | | DummyAirMass | | XCP | air_mass |
| □ | P | | DummyOmega | | XCP | omega |

- As of Version 7.5, ControlDesk no longer assigns calculated variables to variable descriptions. Instead, they are grouped separately from the variable descriptions in the current experiment.

  This change reflects the fact that calculated variables are generally independent of platforms/devices and the related variable descriptions.

The following illustration shows a calculated variable as an example. For calculated variables, the Platform/Device column always displays 'Calculated'.



| Group | Description |
|---|---|
| All Variable Descriptions | |
| Calculated Variables | |
| CalDemo.a2l | |
| Bitmask | |
| DataTypes | |
| DummyBlock | |
| ign_inj_control | |

| Favorite | Var | Coi | Variable | Block | Platform/Device | Description |
|---|---|---|---|---|---|---|
| □ | | | control_deviation | | Calculated | Control deviation of SignalGenOut vs. throttle... |

You also have to consider the following behavior changes:

**Exporting/Importing formulas for calculated variables** As of ControlDesk 7.5, exporting/importing formulas for calculated variables is no longer supported. To transfer formulas from one experiment to another, export the calculated variables that reference these formulas, and import them to the other experiment.

For instructions, refer to How to Export and Import Calculated Variables (ControlDesk Variable Management 📖).

**Creating and editing calculated variables requires online calibration to be stopped**

- Up to and including ControlDesk 7.4, measurement had to be stopped to create a new calculated variable or edit an existing one.
- As of ControlDesk 7.5, online calibration must be stopped for this.

**No description for calculated variables when saving displayed data as a new measurement**

- Up to and including ControlDesk 7.4, when saving the data that is displayed in a Time Plotter as a new measurement, a description including information on the formula and the input signals was added to calculated variables that were saved in this way.
- As of Version 7.5, ControlDesk no longer adds such information to calculated variables that are saved this way.

You also have to consider the migration aspect. Refer to Migrating experiments containing calculated variables on page 80.

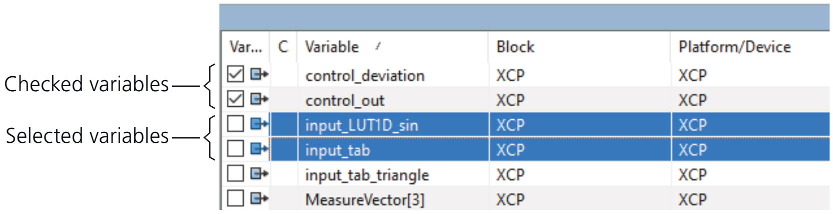
## Changed handling of the Measurement Data Pool

There are some changes to the Measurement Data Pool controlbar.

**Applying commands to selected variables** The handling of the Measurement Data Pool controlbar has been aligned to that of the Variables controlbar.

As a consequence, ControlDesk applies variable-specific commands, such as Visualize Variables and Remove Variables, only to the *selected* variables in the Measurement Data Pool controlbar, but no longer to *checked* variables.

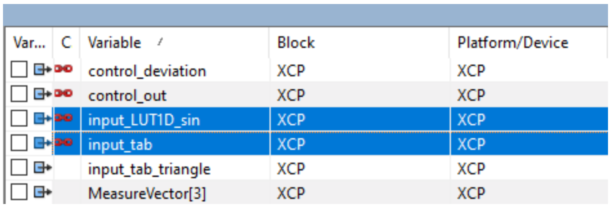
The following illustration shows *two checked variables* and *two selected variables* as an example:



| Var... | C | Variable | Block | Platform/Device |
|---|---|---|---|---|
| ☑ | | control_deviation | XCP | XCP |
| ☑ | | control_out | XCP | XCP |
| ☐ | | input_LUT1D_sin | XCP | XCP |
| ☐ | | input_tab | XCP | XCP |
| ☐ | | input_tab_triangle | XCP | XCP |
| ☐ | | MeasureVector[3] | XCP | XCP |

- Up to and including Version 7.4, ControlDesk applied variable-specific commands to both the *selected* variables and the *checked* variables.

  The following illustration shows the result of using the Visualize Variables command in ControlDesk 7.4 and earlier in the example shown above:

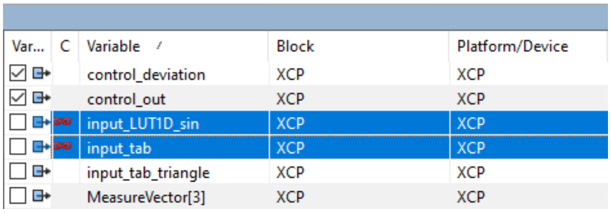


| Var... | C | Variable | Block | Platform/Device |
|---|---|---|---|---|
| ☐ | 🔗 | control_deviation | XCP | XCP |
| ☐ | 🔗 | control_out | XCP | XCP |
| ☐ | 🔗 | input_LUT1D_sin | XCP | XCP |
| ☐ | 🔗 | input_tab | XCP | XCP |
| ☐ | | input_tab_triangle | XCP | XCP |
| ☐ | | MeasureVector[3] | XCP | XCP |

  All checked and selected variables have a red chain symbol, indicating their visualization in an instrument. The previously checked variables are no longer checked.

- As of Version 7.5, ControlDesk applies variable-specific commands only to the *selected* variables, no longer to the *checked* variables.

  The following illustration shows the result of using the Visualize Variables command in ControlDesk 7.5 and later in the example shown above:



| Var... | C | Variable | Block | Platform/Device |
|---|---|---|---|---|
| ☑ | | control_deviation | XCP | XCP |
| ☑ | | control_out | XCP | XCP |
| ☐ | 🔗 | input_LUT1D_sin | XCP | XCP |
| ☐ | 🔗 | input_tab | XCP | XCP |
| ☐ | | input_tab_triangle | XCP | XCP |
| ☐ | | MeasureVector[3] | XCP | XCP |

  Only the selected variables have a red chain symbol, indicating their visualization in an instrument. The previously checked variables remain checked.

**No combined filter** You can no longer filter the measurement signal list by the combined filter. However, you can filter the list by the standard filter and the wildcard filter.

For more information, refer to Standard Filter (ControlDesk Variable Management 🕮) and Wildcard Filter (ControlDesk Variable Management 🕮).

### Changed display of variable limits

The display of variable limits changed in ControlDesk 7.5:

- Up to and including ControlDesk 7.4, the *data type limits* for a variable were displayed in the **Limits** column of the **Variables** controlbar if the variable description contained no limit specification for this variable.
- As of ControlDesk 7.5, *no limits* for a variable are displayed in the **Limits** column of the **Variables** controlbar if the variable description contains no limit specification for this variable.

### Changed behavior in connection with ambiguous table scalings

As of ControlDesk 7.5, when converting variable values from converted to source mode, ControlDesk uses the *first* table entry in connection with ambiguous table scalings.

The following listing shows a table scaling from an A2L file as an example. The scaling is ambiguous because it contains the entries `1600 1` and `3400 1`.

ControlDesk uses the first entry (`1600 1`) when converting values from converted to source mode. Any further table entries for the conversion to `1` are ignored.

```
/begin COMPU_TAB acceleration_s_COMPU_VTAB
   ""
   TAB_INTP 7 /* Number of pairs */
   0 -3
   700 -2
   1600 1
   2500 0
   3400 1
   4300 2
   5000 3
   DEFAULT_VALUE "0"
/end COMPU_TAB
```

### Changed availability of demo projects after installation

The availability of demo projects after installation was changed in ControlDesk 7.5:

- Up to and including ControlDesk 7.4, all the demo projects were automatically copied to the user-specific Documents folder the first time ControlDesk was started after being installed.
- As of ControlDesk 7.5, the demo projects are no longer copied automatically. As a consequence, you have to copy the demo projects manually.

  To copy the demo projects, you have two options:

  - Click **Copy Demos** on the **User Settings** page of the **ControlDesk Options** dialog.

For more information on the **User Settings** page, refer to User Settings Page (ControlDesk User Interface Handling 📖).

- As an alternative, you can use the ControlDesk automation interface. If you use the ControlDesk **Interpreter** controlbar, enter the following command:

```
Application.UserSettings.CopyDemos()
```

> **Tip**
>
> This step is required only once per user after installing ControlDesk.

### Migrating experiments containing calculated variables

Calculated variables from ControlDesk 7.4 or earlier are *not* included in automatic project migration.

As a consequence, to reuse calculated variables in experiments originally created with ControlDesk 7.4 or earlier, you have to do the following:

1. Export the calculated variables using ControlDesk 7.4 or earlier, then import them to ControlDesk 7.5 or later.

   For instructions, refer to How to Export and Import Calculated Variables (ControlDesk Variable Management 📖).

   As an alternative, define the calculated variables again. For instructions, refer to How to Define a Calculated Variable (ControlDesk Variable Management 📖).
2. If necessary, restore the variable connections of the calculated variables.

> **Tip**
>
> You have to perform the steps described above only once, i.e., calculated variables migrated to ControlDesk 7.5 or later will be included in automatic project migration again.

You also have to consider the behavior changes with regard to the handling of calculated variables. Refer to Changed handling of calculated variables on page 76.

### Migrating experiments containing an XIL API MAPort platform

When you open a project and experiment containing an XIL API MAPort platform in ControlDesk 7.5 or later, ControlDesk removes the variable description and data sets from the platform if the experiment was created with ControlDesk 7.4 or earlier due to automatic project migration.

To reuse the experiment, perform the following steps in ControlDesk 7.5 or later:

1. Call the **Add Variable Description** command.

   ControlDesk adds the variable description to the platform and reestablishes the variable connections.
2. If you worked with data sets in connection with the XIL API MAPort platform, call the Import Data Set(s) command.

   ControlDesk reimports the data sets to the variable description.

> **Tip**
>
> The variable description and the data sets are available in the backup of the original project. The backup is created during project migration and stored in the project root directory.

For more information on configuring a XIL API MAPort platform, refer to How to Register and Configure an XIL API MAPort Platform (ControlDesk Platform Management 📖).

### Migrating experiments containing CAN/LIN Bus Monitoring devices

There is a point to note for migration if your experiment contains one of the following devices:

- CAN Bus Monitoring device to which the variable description was added in ControlDesk 5.4 or earlier.
- LIN Bus Monitoring device to which the variable description was added in ControlDesk 5.5 or earlier.

During automatic project migration, ControlDesk automatically reloads the variable description of these devices.

However, due to different variable paths, ControlDesk cannot restore the original variable connections and creates a corresponding entry in the Messages controlbar.

You can use the `MigrateBusMonitoringDevices` Python script to repair the variable connections. For instructions, refer to Repairing variable connections (ControlDesk Introduction and Overview 📖).

### Limit specifications not migrated during project migration

If parameter limits are specified in a project created with ControlDesk 7.4 or earlier, these specifications are not migrated during automatic project migration to ControlDesk 7.5 or later. You must specify custom limits again afterwards.

For more information on limit specifications, refer to Handling Parameter Range Limits (ControlDesk Calibration and Data Set Management 📖).

### Tool automation changes

**Changed access to variable description files** As of ControlDesk 7.5, variable description files are no longer accessible via the `ActiveProject.Files` property.

You can access them via the `ActiveExperiment.Files` property.

**Changes to the IXaVariablesManagement interface** In ControlDesk 7.5, the following changes were made to the `VariablesManagement / IXaVariablesManagement <<Interface>>` interface:

- The `Compress` method was removed.
- The `AskUserToTransferCalculatedVariablesEnabled` property was removed.
- The `VariableDescriptionFileFormats` collection was replaced by the `ImportOptions` property.

For example, to allow for the generation of raw byte variables for the bus import, specify the following:

```
Application.VariablesManagement.ImportOptions.Bus.GenerateRawBytesEnabled = True
```

In addition, the following import options were removed:

- A2L: `SupportIFDataAtVariablesEnabled`
- BUS: `AutorenameFormat`
- SDF: `IdenticalAddressLoggingEnabled`
- SDF: `ValidationEnabled`
- SDF: `VerboseErrorOutputEnabled`

**Change to the IXaVariableDescription interface** In ControlDesk 7.5, the `TypeName` and `CreatedBy` properties were removed from the `VariableDescription / IXaVariableDescription <<Interface>>` interface.

**Changes to the IXaActiveVariableDescription interface** In ControlDesk 7.5, the following changes were made to the `ActiveVariableDescription / IXaActiveVariableDescription <<Interface>>` (ControlDesk Automation 🕮) interface:

- The following properties are `NULL` for the top-level variable description of multicore (MC) and multiprocessor (MP) applications:
  - `RootGroup`
  - `Scalings`
  - `TypeDefinitions`
  - `Variables`
- Up to and including ControlDesk 7.4, when you used the `Replace` method to replace the A2L variable description file, ControlDesk also added the original ECU Image file when replacing variable descriptions.

  As of ControlDesk 7.5, ControlDesk no longer adds the original ECU Image file when you use the `Replace` method. As a result, there is no reference data set after replacing variable descriptions. To maintain the data sets, use the `ReplaceWithImage` method instead.

**Changes to variable-specific interfaces** In ControlDesk 7.5, the following changes were made to variable-specific interfaces:

- The `IsDeprecated` property was removed.
- The `Name` and `Description` properties were made read-only.
- The `Address` property was changed to 64-bit.
- The `AddressOffset` property was changed to null/none if the offset is not defined in the variable description.
- The type of the `Bitmask` property was changed to `Object`. If the bitmask is set, the data type is always `UInt64`.
- The `StartBit` and `NumberOfBits` properties were removed.

These changes apply to the following variable-specific interfaces:

- AddressInformation / IXaAddressInformation <<Interface>>
- BitMask / IViBitMask <<Interface>>
- CalculatedVariable / IXaCalculatedVariable <<Interface>>
- CommonAxisVariable / IXaCommonAxisVariable <<Interface>>
- CurveVariable / IXaCurveVariable <<Interface>>
- MapVariable / IXaMapVariable <<Interface>>
- MeasurementArrayVariable / IXaMeasurementArrayVariable <<Interface>>
- MeasurementVariable / IXaMeasurementVariable <<Interface>>
- ParameterVariable / IXaParameterVariable <<Interface>>
- StringVariable / IXaStringVariable <<Interface>>
- StructArrayVariable / IXaStructArrayVariable <<Interface>>
- StructVariable / IXaStructVariable <<Interface>>
- ValueBlockVariable / IXaValueBlockVariable <<Interface>>

**Changes to the definition and handling of calculated variables** The creation and handling of calculated variables was changed in ControlDesk 7.5:

- Up to and including Version 7.4, ControlDesk assigned a calculated variable to the variable description to which it was initially added. This was done even if the input signals of the calculated variable were derived from another variable description.

  Automation interfaces relevant to calculated variables were part of the `ActiveVariableDescription / IXaActiveVariableDescription <<Interface>>` interface.

- As of Version 7.5, ControlDesk no longer assigns calculated variables to variable descriptions. Instead, they are grouped separately from the variable descriptions in the current experiment.

  Consequences:

  - Automation interfaces relevant to calculated variables were moved to the new `CalculatedVariablesConfiguration / IXaCalculatedVariablesConfiguration <<Interface>>` interface.
  - Newly created formulas are empty by default, i.e., you have to specify formula parameters and the formula itself.
  - Type definitions for calculated variables can no longer be created. Instead, you have to select a predefined type definition.

    See below for an example:

    ```
    doubleTypeDefinition =
    calculatedVariablesConfiguration.TypeDefinitions.Float64
    ```

  - Scalings for calculated variables can no longer be created. As a consequence, the `Scaling` property always returns `Null`. The `ValueConverted` and `ValueSource` properties always return the Source value.
  - The connection string for a calculated variable has changed:

    The following example shows a connection string for a calculated variable as of ControlDesk 7.5:

    ```
    Calculated(5ms)://control_deviation
    ```

The following example shows a connection string for a calculated variable assigned to a device named XCP up to and including ControlDesk 7.4:

```
XCP(5ms)://control_deviation
```

Examples:

The following listing shows how to create a new formula as of ControlDesk 7.5 as an example:

```
calculatedVariablesConfiguration = Application.ActiveExperiment.CalculatedVariablesConfiguration
newFormula = calculatedVariablesConfiguration.Formulas.Add("TestFormula")
newFormula.Formula = "a + b"
newFormula.FormulaParameters.Add("a")
newFormula.FormulaParameters.Add("b")
newFormula.Description = "A new test formula."
```

The following listing shows how to create a new calculated variable as of ControlDesk 7.5 as an example:

```
newCalculatedVariable = calculatedVariablesConfiguration.RootGroup.Variables.AddCalculatedVariable("Test_CV",
                                                                                                    doubleTypeDefinition,
                                                                                                    newFormulaInstance)
```

For a detailed listing that shows how to define a calculated variable, refer to Creating calculated variables (ControlDesk Variable Management 🕮).

**Change to the IXaFormulaParameter interface** In ControlDesk 7.5, the `BufferSize` property of the `FormulaParameter / IXaFormulaParameter <<Interface>>` interface was removed.

**Changes to the Scaling interface of variable descriptions** In ControlDesk 7.5, the `Add` and `Remove` methods of the `Scaling` interface for variable descriptions were removed.

**Changed access to subelements of multidimensional variables** The following change applies to multidimensional variables, i.e., to value blocks, measurement arrays, and to the data point arrays of maps.

- Up to and including ControlDesk 7.4, when a subelement of a multidimensional variable was accessed, ControlDesk did not check whether the index of a dimension was consistent with the dimension length. For example, when a value block with the dimensions [2][3] was accessed via `ValueBlock.SubElements[0][5]`, no exception was returned.
- As of ControlDesk 7.5, the `ArgumentOutOfRangeException` exception is returned in this case.

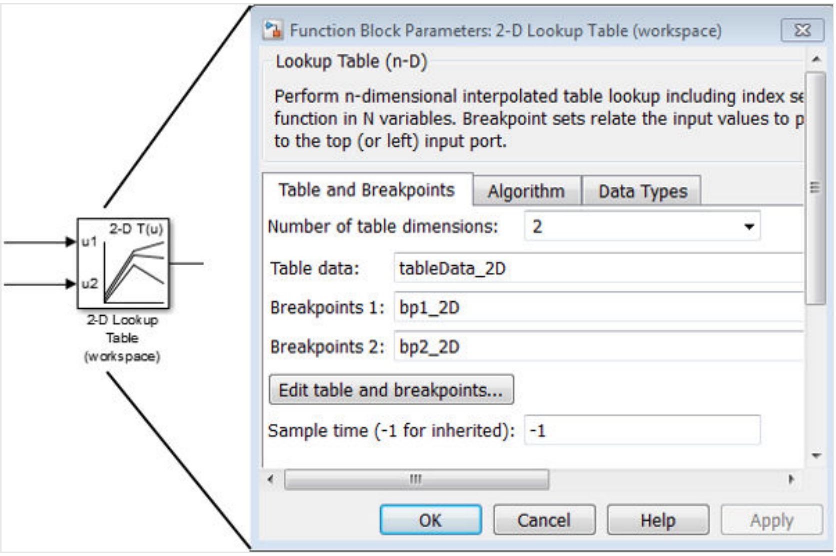
**Changed exceptions for collections** In ControlDesk 7.5, the exceptions returned by collections such as `IXaVariables` and `IXaGroupVariables` were harmonized. As an example, the `IndexOutOfRangeException` exception in ControlDesk 7.4 or earlier was changed to `ArgumentOutOfRangeException` in ControlDesk 7.5.

**Changed access to struct elements**

- Up to and including ControlDesk 7.4, the `Item` method of the `Variables / IXaVariables <<Collection>>` interface returned individual struct elements.
- As of ControlDesk 7.5, , use the `ItemByPath` method to return individual struct elements.

**Corrected treatment of look-up tables (SDF files only)** The following change applies to look-up tables that contain references to other variables. It applies to look-up tables in SDF files only.

As an example, the following illustration shows the parameterization of a Simulink look-up table that references workspace variables, such as `tableData_2D`.



Function Block Parameters: 2-D Lookup Table (workspace)

Lookup Table (n-D)

Perform n-dimensional interpolated table lookup including index se function in N variables. Breakpoint sets relate the input values to p to the top (or left) input port.

Number of table dimensions: 2

Table data: tableData_2D

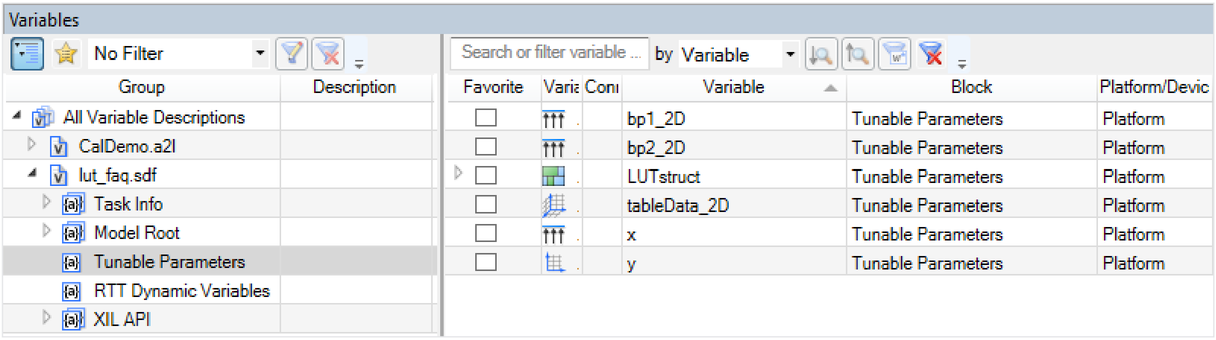
Breakpoints 1: bp1_2D

Breakpoints 2: bp2_2D

Sample time (-1 for inherited): -1

- Up to and including ControlDesk 7.4, referenced variables were represented with a wrong variable type.

  The following illustration shows the Tunable Parameters variable group in ControlDesk. The group contains the referenced `tableData_2D` variable from the Simulink example above. The variable is represented incorrectly as a map.

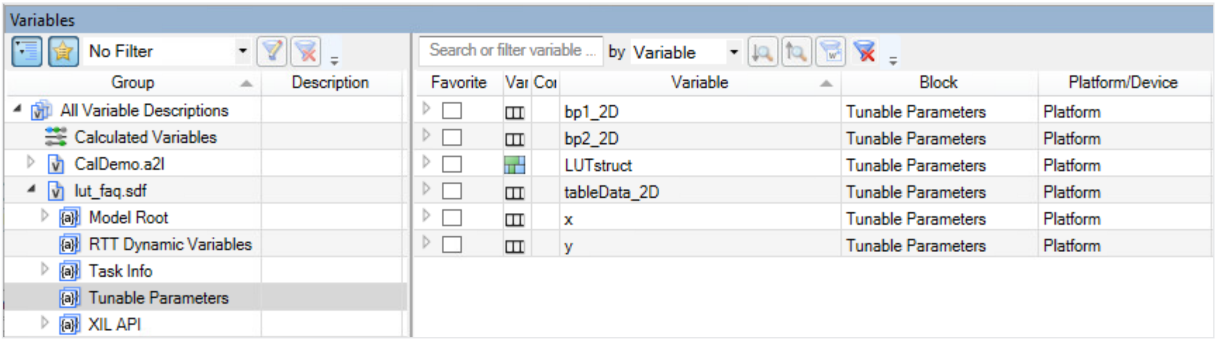


| Group | Description |
|---|---|
| All Variable Descriptions | |
| CalDemo.a2l | |
| lut_faq.sdf | |
| Task Info | |
| Model Root | |
| Tunable Parameters | |
| RTT Dynamic Variables | |
| XIL API | |

| Favorite | Varia | Conı | Variable | Block | Platform/Devic |
|---|---|---|---|---|---|
| □ | | | bp1_2D | Tunable Parameters | Platform |
| □ | | | bp2_2D | Tunable Parameters | Platform |
| □ | | | LUTstruct | Tunable Parameters | Platform |
| □ | | | tableData_2D | Tunable Parameters | Platform |
| □ | | | x | Tunable Parameters | Platform |
| □ | | | y | Tunable Parameters | Platform |

- As of ControlDesk 7.5, referenced variables are represented with the correct variable type in the Variables controlbar.

  The following illustration shows the Tunable Parameters variable group in ControlDesk. The group contains the referenced `tableData_2D` variable from the Simulink example above. The variable is represented correctly as a value block.



Variables

No Filter

| Group | Description |
|---|---|
| All Variable Descriptions | |
| Calculated Variables | |
| CalDemo.a2l | |
| lut_faq.sdf | |
| Model Root | |
| RTT Dynamic Variables | |
| Task Info | |
| Tunable Parameters | |
| XIL API | |

Search or filter variable ... by Variable

| Favorite | Var | Col | Variable | Block | Platform/Device |
|---|---|---|---|---|---|
| □ | | | bp1_2D | Tunable Parameters | Platform |
| □ | | | bp2_2D | Tunable Parameters | Platform |
| □ | | | LUTstruct | Tunable Parameters | Platform |
| □ | | | tableData_2D | Tunable Parameters | Platform |
| □ | | | x | Tunable Parameters | Platform |
| □ | | | y | Tunable Parameters | Platform |

**Changed use of data type limits (SDF files only)** The following change applies to variables in SDF files only.

- Up to and including ControlDesk 7.4, variable limits resulting from the data type were used as the *weak limits* of that variable.
- As of ControlDesk 7.5, variable limits resulting from the data type are used as the *hard limits*.

With this change, the use of data type limits for SDF file variables has been aligned to that for variables of all the other variable description file formats supported by ControlDesk.

**Changed base type names (SDF files only)** The following table compares the base type names of variables from SDF files:

| BaseType | Up to and Including ControlDesk 7.4 | As of ControlDesk 7.5 |
|---|---|---|
| 1 | FltIeee32Type | float32 |
| 2 | FltIeee64Type | float64 |
| 3 | IntType | int32 |
| 4 | Int8Type | int8 |
| 5 | Int16Type | int16 |
| 6 | Int64Type | int64 |
| 7 | UInt8Type | uint8 |
| 8 | UInt16Type | uint16 |
| 9 | UInt32Type | uint32 |
| 10 | UInt64Type | uint64 |

**Changed TypeDefinition for values and value blocks (A2L files only)** Up to and including ControlDesk 7.4, a structured type definition was used for values and value blocks in A2L files. The first component of the type definition described the data type of the parameter.

The following listing shows an example:

```
ActiveVarDesc = Application.ActiveExperiment.Platforms[0].ActiveVariableDescription
ActiveVarDesc.Variables['XCP()://DummyAirMass'].TypeDefinition.BaseType 12
ActiveVarDesc.Variables['XCP()://DummyAirMass'].TypeDefinition.Name 'SWORD_COL_DIRECT'
ActiveVarDesc.Variables['XCP()://DummyAirMass'].TypeDefinition.Components[0].Name 'FNC_VALUES'
ActiveVarDesc.Variables['XCP()://DummyAirMass'].TypeDefinition.Components[0].Type.BaseType 5
ActiveVarDesc.Variables['XCP()://DummyAirMass'].TypeDefinition.Components[0].Type.Name 'SWORD'
```

As of ControlDesk 7.5, the type definition of the data type for values and value blocks in A2L files is used.

The following listing shows an example:

```
ActiveVarDesc = Application.ActiveExperiment.Platforms[0].ActiveVariableDescription
ActiveVarDesc.Variables['XCP()://DummyAirMass'].TypeDefinition.BaseType 5
ActiveVarDesc.Variables['XCP()://DummyAirMass'].TypeDefinition.Name 'SWORD'
```

**Changed behavior when setting the value of float/double parameters to infinity** The following change applies to parameters with the `float` or `double` data type:

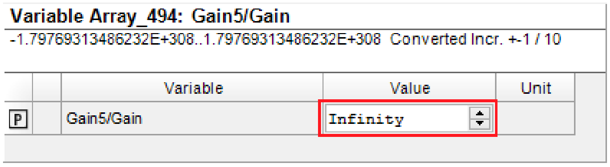
- Up to and including ControlDesk 7.4, setting the value of such parameters to infinity was impossible both via the ControlDesk automation interface and via the graphical user interface.
- As of ControlDesk 7.5, setting the value of such parameters to infinity is possible via the ControlDesk automation interface and via the graphical user interface.

*Example*:

The following Python listing shows how to set a parameter to infinity via the ControlDesk automation interface. A double parameter of the RT application demo project is used as an example:

```
v = ActiveVarDesc.Variables['Platform()://Model Root/Tables_Vectors_Bitmasks/DummyBlock_100ms/Gain5/Gain']
v.ValueConverted = float('inf')
```

The following illustration shows the variable in an instrument.



**Variable Array_494: Gain5/Gain**

-1.79769313486232E+308..1.79769313486232E+308 Converted Incr. +-1 / 10

| | | Variable | Value | Unit |
|---|---|---|---|---|
| P | | Gain5/Gain | Infinity | |

## MC3 automation change

As of ControlDesk 7.5, the `Accuracy` and `Resolution` properties of IMCDbMeasurement return 0 or 1.

## Migrating from earlier ControlDesk versions

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

## Related topics

Basics

Basics on Migrating from Prior Versions of ControlDesk (ControlDesk Introduction and Overview 🕮)
Demo Projects (ControlDesk Introduction and Overview 🕮)

References

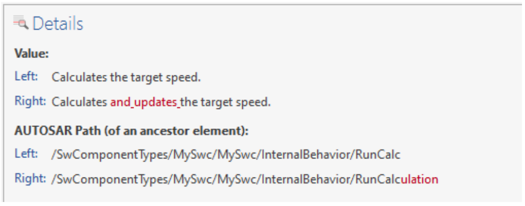
Save Displayed Data as New Measurement (ControlDesk Instrument Handling 🕮)
User Settings Page (ControlDesk User Interface Handling 🕮)
UserSettings / IXaUserSettings <<Interface>> (ControlDesk Automation 🕮)

# dSPACE AUTOSAR Compare

## New Features of dSPACE AUTOSAR Compare 1.1

**General improvements**

- The new version also supports AUTOSAR Release R20-11.
- You can now specify an AUTOSAR path to go to elements on a comparison page.
- The added **Details** pane highlights the differences of selected AUTOSAR elements and displays the full AUTOSAR path.



Details

**Value:**

Left: Calculates the target speed.

Right: Calculates and_updates_the target speed.

**AUTOSAR Path (of an ancestor element):**

Left: /SwComponentTypes/MySwc/MySwc/InternalBehavior/RunCalc

Right: /SwComponentTypes/MySwc/MySwc/InternalBehavior/RunCalculation

**Integration into version control systems**

dSPACE AUTOSAR Compare can be integrated into version control systems, such as Git, to view differences between different revisions of ARXML files and to assist with merge conflicts. The following sections for *adding dSPACE AUTOSAR Compare*, *viewing differences*, and *solving merge conflicts* focus on Git.

**Adding dSPACE AUTOSAR Compare**

You can add dSPACE AUTOSAR Compare as a diff and a merge tool in Git. You can use dSPACE AUTOSAR Compare as the default tool for the aforementioned purposes.

**Viewing differences**

You can use the following command to inspect local changes to a file that has not been committed:

```
git difftool -t darc <file>
```

dSPACE AUTOSAR Compare opens a session in which you can compare the file contents between the last commit and your local changes.

You can omit '-t darc' if you set up dSPACE AUTOSAR Compare as the default Git diff tool. If you do not specify a file name, a session will be opened for all changed files.

### Solving merge conflicts

When merging or rebasing branches, Git notifies you of any merge conflicts with this message:

```
Auto-merging <file>
CONFLICT (content): Merge conflict in <file>
Automatic merge failed; fix conflicts and then commit the result.
```

You can use the following command to solve the merge conflicts with dSPACE AUTOSAR Compare:

```
git mergetool -t darc <file>
```

You can omit '-t darc' if you set up dSPACE AUTOSAR Compare as the default Git merge tool.

If mergetool.prompt is set to 'true' in your Git configuration, Git now shows further information and lets you solve merge conclicts manually in dSPACE AUTOSAR Compare. Otherwise, the merge conclicts are solved automatically.

### Related topics

Basics

Integration in Other Tools (dSPACE AUTOSAR Compare Manual 📖)

# dSPACE Installation Manager

**Where to go from here**

Information in this section

## New Features of dSPACE Installation Manager 5.8

**New features for managing installations**

Now, the dSPACE Installation Manager allows you to save the existing configuration of dSPACE installations on the local PC to an XML file. The contents of the file can then be displayed and edited in a specific user dialog. For example, you can change configuration settings, such as activating dSPACE installations or creating MATLAB connections. This changed configuration can then be applied to the local PC. For details, refer to Changing the Configuration of dSPACE Installations via Configuration File (Managing dSPACE Software Installations 📖).

**Enhanced API functionality**

The API of the dSPACE Installation Manager now also supports license management functions, for example, license activation, license updating, license borrowing. Refer to Features of the API and Important Notes (dSPACE Installation Manager API Reference 📖).

**New command line utility for the API**

The dSPACE Installation Manager now provides the `InstallationManager.Console`command line utility. This utility is a Command Line Interface (CLI) wrapper for the API of the dSPACE Installation Manager. With dSPACE Release 2021-B, the `InstallationManager.Console` command line utility supports only part of the API functionality: license management. For more information, refer to Using the

InstallationManager.Console Command Line Utility (dSPACE Installation Manager API Reference 🕮).

## Migrating to dSPACE Installation Manager 5.8

### Using CmDongles

If you want to work with licenses on CmDongles in combination with dSPACE Installation Manager 5.8, for example, to activate, deactivate, or update licenses, the dongles must have at least firmware version 4.10.

To use CmDongles shipped for Releases earlier than dSPACE Release 2019-A, a firmware update is required. CmDongles shipped for dSPACE Release 2019-A and later contain the required minimum firmware version.

dSPACE Installation Manager checks if the firmware of a connected dongle matches the required minimum firmware version and displays if an update is necessary.

For instructions on updating the firmware, refer to How to Update the Firmware of a CmDongle (Working with CodeMeter Licensing Technology 🕮).

# dSPACE Python Extensions

## New Features of dSPACE Python Extensions 4.1

**New features**

The dSPACE Python Extensions do not have new features.

**Discontinuation of rs232lib2**

The `rs232lib2` Python module is discontinued with Release 2021-B.

As an alternative, you can use the standard Python serial port extension `pyserial`.

# dSPACE XIL API .NET

## New Features of dSPACE XIL API .NET 2021-B

**New features**

The dSPACE XIL API.NET 2021-B has the following new feature:

- Improved documentation structure for the model access port (MAPort) and the electrical error simulation port (EESPort). You will now find the specific information for your port implementation in separate documents. Refer to dSPACE XIL API MAPort Implementation 📖 and dSPACE XIL API EESPort Implementation 📖 .

**dSPACE XIL API .NET on Linux**

The `CmdLoader` command line tool in the XIL API .NET on Linux installation lets you configure and register dSPACE platforms using a JSON configuration file. For more information, refer to CmdLoader (dSPACE XIL API MAPort Implementation 📖).

# ECU Interface Manager

**Where to go from here**

Information in this section

## New Features of ECU Interface Manager 2.10

**Support for variables with address extension ! = 0**

The ECU Interface Manager supports A2L files describing variables with address extensions != 0.

For the XCP on CAN and XCP on Ethernet ECU interfaces, 16-bit address extensions are converted to 8 bits.

## Compatibility of ECU Interface Manager 2.10

**Compatibility in general**

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

**Compatibility between EIC files and ConfigurationDesk**

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

| | EIC Files Created with ECU Interface Manager ... | | | |
|---|---|---|---|---|
| | Version 2.7[1] | Version 2.8[2] | Version 2.9[3] | Version 2.10[4] |
| ConfigurationDesk 6.8[4] | ✓ | ✓ | ✓ | ✓ |
| ConfigurationDesk 6.7[3] | ✓ | ✓ | ✓ | ✓ |
| ConfigurationDesk 6.6[2] | ✓ | ✓ | ✓ | ✓ |
| ConfigurationDesk 6.5[1] | ✓ | ✓ | ✓ | ✓ |

[1] dSPACE Release 2020-A

[2] dSPACE Release 2020-B

[3] dSPACE Release 2021-A

[4] dSPACE Release 2021-B

## Migrating to ECU Interface Manager 2.10

**Automatic migration of projects**

You can reuse projects in the ECU Interface Manager 2.10 if the projects were last saved with the ECU Interface Manager 2.0 p1 or later.

When you open the projects in the ECU Interface Manager 2.10, they are migrated automatically.

**Note**

In the ECU Interface Manager 2.10, you cannot reuse projects that were last saved with ECU Interface Manager 2.0 or earlier.

**Additional migration steps in some cases**

To migrate to the ECU Interface Manager 2.10 from versions earlier than the ECU Interface Manager 2.7, you might also have to perform the migration steps of the intervening ECU Interface Manager versions.

# Firmware Manager

## New Features of Firmware Manager 3.4

**Discontinued firmware archive**

The firmware archive for SCALEXIO systems with a QNX operating system is no longer available.

# MicroAutoBox III Firmware

## New Features of the MicroAutoBox III Firmware 5.2

**DS1403 Processor Board**

The firmware now supports the following new features and improvements:

- You can activate error-correcting code (ECC) that can detect and correct memory errors. Refer to How to Activate Error-Correcting Code for the RAM (MicroAutoBox III Hardware Installation and Configuration 🕮).
- Functional safety monitoring is enhanced with a RAM memory monitor. This monitor can be used to trigger a functional safety (FuSa) error if a memory error occurs.

  For more information, refer to FuSa Memory Monitoring (ConfigurationDesk I/O Function Implementation Guide 🕮).
- The USB port B that is connected to the real-time processor can now be used to access USB mass storage devices with the real-time application.

  For more information, refer to Accessing USB Mass Storage Devices with Real-Time Applications (MicroAutoBox III Hardware Installation and Configuration 🕮).
- The web interface is improved for a more convenient operation. Refer to Web Interface Reference (MicroAutoBox III Hardware Installation and Configuration 🕮).

**DS1554 Engine Control I/O Module**

The firmware now supports the control of LSU 5.1 lambda probes.

For the electrical characteristics of the lambda interface, refer to Lambda In 1 Characteristics (MicroAutoBox III Hardware Installation and Configuration 🕮).

For implementing the lambda interface, refer to Lambda Probe In (ConfigurationDesk I/O Function Implementation Guide 🕮).

# Model Compare

**Where to go from here**

**Information in this section**

## New Features of Model Compare 3.2

**New Difference Overview**

Model Compare 3.2 provides a new Difference Overview dialog, which gives you a quick overview of changed model parts. It lets you select specific model parts for the comparison instead of using the entire model. This leads to faster comparison results.

**Related documentation**

- Difference Overview (Model Compare Reference 📖)
- How to Create Comparison Sessions Starting with the Difference Overview Dialog (Model Compare Guide 📖)

**Improved workflow integration**

Model Compare 3.2 provides improved workflow integration by means of additional command line interface options for the use of Model Compare with the command line interface or in version control systems:

- `-ComparisonSettings`
- `-ConflictRes`
- `-ImportComparisonSettings`
- `-ImportUserPreferences`
- `-Session`

**Related documentation**

- The ModelCompare [Options] API Command (Model Compare Reference 📖)

- Basics on the Interaction with Version Control Systems (Model Compare Guide 🕮)

**New documentation: Using Model Compare with Git**

The new documentation Using Model Compare with Git provides all information required to set up Model Compare as a difference and merge tool in your Git environment.

Refer to Using Model Compare with Git.

**New tutorial video: Model Compare Quick Guide**

The new tutorial video Model Compare Quick Guide introduces you to Model Compare and its key features.

Refer to Model Compare Quick Guide.

## Migration to Model Compare 3.2

**Obsolete API command options**

With Model Compare 3.2, the `-settings` option is no longer supported.

Use `-ImportComparisonSettings` or `-ComparisonSettings` instead.

Refer to The ModelCompare [Options] API Command (Model Compare Reference 🕮).

# Model and Sensor Interface Blockset

**Where to go from here**

Information in this section

## New Features of Model and Sensor Interface Blockset 1.2

**Model and Sensor Interface Blockset**

As of dSPACE Release 2020-B, the Model and Sensor Interface Blockset (MSI Blockset) must be used for simulation with sensors in a software-in-the-loop (SIL) or in a hardware-in-the-loop (HIL) test environment.

In Sensor Simulation, the blockset is required to calculate simulation data and transmit it to the MotionDesk PC for visualization and to the SensorSim applications that produce the sensor composition and raw data for connected sensors.

The data is transmitted in data streams of sequenced data elements using the TCP/IP protocol. In the Simulink model, the blockset must be connected to Ethernet functions to communicate with the selected supported simulation platform and to Sensor Simulation systems and applications using the TCP/IP protocol.

For each of the moving objects in the simulation, simulation data objects can be defined in kinematic chains in single object blocks using a parent-child relationship. Blocks can also be added to retrieve status and statistical information on the connected system.

You can connect an Environment Sensor Interface Unit. The unit splits the sensor raw data coming from the SensorSim application and inserts it into the relevant sensor hardware, for example, into imaging sensor interfaces of camera sensors. You can add blocks to the Simulink model to receive feedback on the connected sensors from the Environment Sensor Interface Unit. You can also configure sensor failures for specific sensors, for example, a pixel error on a camera sensor.

If you adapted ASM models using the Model and Sensor Interface Solution, you can migrate these models to use the Model and Sensor Interface Blockset.

**New features**

The following new features for the Model and Sensor Interface Blockset 1.2 are available:

- Improved usability of the block parameters dialogs.
- A dsmsi_analyze function to evaluate the simulation model before building. For example, to evaluate block sample times, parameters and block connections.
- Copy a Simulation Data Object and automatically append a number to the top level objects in the block.

  It is not possible to build a model that contains Simulation Data Objects blocks where duplicate root objects names exist across the Simulation Data Objects blocks in the model.
- Performance upgrades and optimizations.

**Related topics**

Basics

Model and Sensor Interface Blockset Manual
New Features of MotionDesk 4.9 ........................................ 113
Sensor Simulation Overview

## Migrating to Model and Sensor Interface Blockset 1.2

**MicroAutobox III**

The MicroAutobox III platform is not supported by the Model and Sensor Interface Blockset.

For more information on using MicroAutobox III with the blockset, contact dSPACE Support.

**Migrating from previous Releases**

**Migration from previous releases as of dSPACE Release 2020-B** Models created with previous Releases of the Model and Sensor Interface Blockset as of version 1.0 in dSPACE Release 2020-B are migrated automatically when you open the model. Messages are displayed in the MATLAB Command Window and in the migration log to confirm successful migration or notify you of failures.

**Migration from the Model and Sensor Interface Blockset Solution** Models created with the Model and Sensor Interface Blockset Solution that was available prior to dSPACE Release 2020-B must be manually migrated. When you open a model that uses blocks from the blockset solution, a message is displayed with instructions on how to migrate the model to use the blocks of the latest Release of the Model and Sensor Interface Blockset.

# ModelDesk

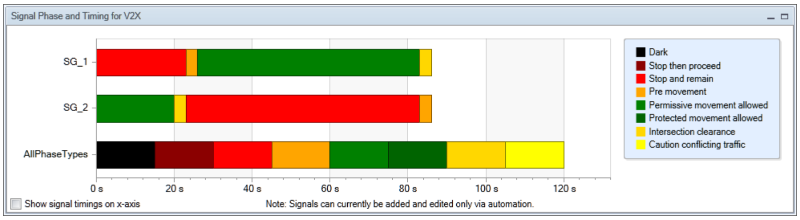
**Where to go from here**

**Information in this section**

## New Features of ModelDesk 5.6

**Road Generator**

**V2X support** ASM is extended by a traffic light controller so that SPaT (signal phase and timing) messages can be sent to the ego-vehicle. In this version of ModelDesk, you can specify the chronological order of movement events for the traffic light. These movement events are defined in signal groups that contain the signals of the traffic lights. The signal groups are properties of junctions and can be specified via the user interface. Signals must be specified via tool automation. The user interface has a new pane that gives you an overview of the signal groups and their signals.


Note: Signals can currently be added and edited only via automation.

Signal Phase and Timing for V2X
SG_1
SG_2
AllPhaseTypes
0 s
20 s
40 s
60 s
80 s
100 s
120 s
□ Show signal timings on x-axis
⬛ Dark
🟫 Stop then proceed
🟥 Stop and remain
🟧 Pre movement
🟩 Permissive movement allowed
🟩 Protected movement allowed
🟨 Intersection clearance
🟨 Caution conflicting traffic

You can assign signal groups to lanes. Each signal group has a unique ID that you can select in the properties of the lanes that are part of a road connected to the junction with a traffic light.

**Working in Road Network 2-D Preview** The handling for working with multiple road network elements in the **Road Network 2-D Preview** is improved:

- You can move all selected road network elements together by specifying new values of the center point in the property grid.

- You can rotate all selected road network elements together by calling a command of the context menu and specifying the rotation angle and rotation center point.

  You can rotate all selected network elements by specifying a rotation angle in the Properties pane. In this case, all selected network elements are rotated around the common center point.

**Shortcuts** New shortcuts have been implemented for various commands.

**Visualization** The Road 1-D Preview and Junction 2-D Preview display the direction of the road at the connection of the road network element.

The Road 1-D Preview and Junction 2-D Preview display the lane indices.

**Traffic object management**

You can now change the 3-D object of an existing traffic object via tool automation.

**Scenario Editor**

**Shortcuts** New shortcuts have been implemented for various commands.

**Alias support**

In the previous ModelDesk versions, aliases were supported only for scalar parameters with a double data type. Now, you can also use aliases for strings, Booleans, and enumeration data types.

**Parameterization**

The navigation buttons are disabled if model parts are not available.

ModelDesk lets you to edit a parameter set in the user interface to the maximum extent possible in the ASM models. In previous ModelDesk versions, if some ASM model parts were not included in a model, all respective input fields were unavailable in the user interface, but you were able to access them via navigation buttons. Now, these navigation buttons are disabled automatically.

**Related topics**

HowTos

How to Parameterize a V2X Traffic Light Controller (ModelDesk Road Creation 📖)

## Migration to ModelDesk 5.6

**Project migration**

As of ModelDesk 5.6, you can migrate only projects created with ModelDesk version 4.6 (dSPACE Release 2017-B) and later.

**Using an ASM of Release 2020-B in a migrated project**

When you migrate a ModelDesk project and the corresponding ASM model, all signals in the layouts used for plotting are still connected. If you replace the migrated ASM model with an ASM model created with Release 2020-B, you must connect the signals again. ASM models created with Release 2020-B use ASMSignalInterface blocks that can be used in different parts of the model.

**Triggering plots**

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

**Tip**

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

**ASM project**

Creating new ModelDesk projects on the basis of ASM projects is no longer supported for DS1006 target platforms. However, DS1006 platforms are still supported by ModelDesk. You can still parameterize real-time applications running on DS1006 platforms.

# Model Interface Package for Simulink

**Where to go from here**

Information in this section

## New Features of the Model Interface Package for Simulink 4.6

**Support of obfuscated source code in Simulink models**

As of the Model Interface Package for Simulink 4.5p1, the Model Interface Package for Simulink supports the Simulink code obfuscation feature for top-level models for which SIC files are generated. You can activate this option via the `ObfuscateCode` arguments of the `dsrt_build()` API command. For more information, refer to dsrt_build (Model Interface Package for Simulink API Reference 📖).

**Preservation of signal IDs when assigning Simulink.Bus objects to untyped bus signals**

You can assign a Simulink.Bus signal with a matching signal configuration to model port blocks with structured data ports that have untyped bus signals by using the model port block dialogs. As of the Model Interface Package for Simulink 4.5p1, the Model Interface Package for Simulink preserves the signal IDs during this operation. Thus, the model interface to ConfigurationDesk is preserved.

# Migrating to the Model Interface Package for Simulink 4.6

**Limitations when using MATLAB R2021b**

**Allow multiple tasks to access inputs and outputs is not supported** If the `Allow multiple tasks to access inputs and outputs` option is selected, a root level In block or Out block can be called by multiple tasks. The Model Interface Blockset for Simulink does not support this feature.

**Known problems when using model port blocks in variant scenarios** Due to a bug in MATLAB R2021b, MATLAB may stop working if you use model port blocks in variant scenarios. Refer to https://www.mathworks.com/support/bugreports/details/2570862. For working with variants, it is recommended to install MAtLAB R2021b Update 1, which is expected to fix this bug.

# MotionDesk

**Where to go from here**

Information in this section

## New Features of MotionDesk 4.9

**MotionDesk**

There are no new features in MotionDesk 4.9.

## Migrating to MotionDesk 4.9

**Discontinuations**

There are no new discontinuations in MotionDesk 4.9. Development has focused on a number of user interface and background fixes and improvements.

**Using endless ground plate and horizon**

In MotionDesk 4.0 and earlier, the virtual world of a scene was built using ground plate and dome 3-D objects.

If you want to use the endless ground plate and endless sky in an old scene, you first have to delete these 3-D objects.

In advanced lighting mode, the static objects used for domes are not suitable for building the virtual world. You must use the endless sky.

**Migrating 3-D custom objects**

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

**Migrating from previous Releases**

MotionDesk supports experiments from the previous four versions of MotionDesk.

# Real-Time Testing

**Where to go from here**

Information in this section

## New Features of Real-Time Testing 5.1

**No new features**

There are no new features in Real-Time Testing 5.1.

## Migrating to Real-Time Testing 5.1

**Incompatible BCG files**

BCG files generated on a Windows PC cannot be used on a Linux PC and vice versa. Generate the BCG files under the operating system that you want to use for managing real-time tests.

The BCG files generated with Real-Time Testing 4.0 or earlier cannot be used for Real-Time Testing 5.1. You must create the BCG file of the RTT sequence again.

Only for SCALEXIO as of Real-Time Testing 4.4 and VEOS as of Real-Time Testing 4.2: The internal Python interpreter version changed from 2.7.11 to 3.6.4. To use older scripts in the syntax of Python 2.7.11, you must migrate them to the syntax of Python 3.6.4. For more information on migrating Python scripts, refer to http://www.dspace.com/go/Python36Migration.

# RTI/RTI-MP and RTLib

**Where to go from here**

Information in this section

## New Features of RTI/RTI-MP and RTLib

**New features of RTI/RTI-MP**

RTI and RTI-MP has the following new feature:

- Support of MATLAB R2021b.

## Migration Aspects of RTI/RTI-MP and RTLib

**Modified features in later MATLAB versions**

**Switching to a later MATLAB version** If you install a new MATLAB version, some settings are adopted from previously installed MATLAB versions. To prevent unexpected behavior by the Simulink models when you switch to a later MATLAB version or dSPACE Release, always reset the MATLAB and Simulink preferences to their default values before you start using the models.

If you change the MATLAB version and/or the dSPACE Release, configuration sets stored in a MAT file of an earlier version might cause problems. Therefore, you are recommended to create these configuration sets again when you change the Release version.

**Note**

Simulink.ConfigSet objects that were saved in MAT files using dSPACE Release 2013-B or older cannot be migrated to Release 2021-B.

# RTI Bypass Blockset

## Migrating to RTI Bypass Blockset 3.17

**Models containing a DCI-GSI1-based interface**

Since dSPACE Release 2020-A, the DCI-GSI1 has no longer been supported. As a consequence, the RTI Bypass Blockset no longer supports the following DCI-GSI1-based bypass interface types:

- DCI_GSI1
- dSPACE_on_JTAG_NEXUS
- dSPACE_on_JTAG_OCDS
- dSPACE_on_JTAG_SDI
- dSPACE_on_NBD_AUD
- dSPACE_on_NEXUS_READI
- cPATCH_on_JTAG_NEXUS
- cPATCH_on_JTAG_OCDS
- cPATCH_on_JTAG_SDI
- cPATCH_on_NBD_AUD
- cPATCH_on_NEXUS_READI

When you open a model with one or more of these DCI-GSI1-based bypass interfaces specified in the imported database files in the RTI Bypass Blockset 3.14 or later, a message informing you about the discontinued support is displayed, and the DCI-GSI1-based interfaces are removed from the Setup block. You can then continue working with the model as follows:

- If you did not select any of the listed interfaces in the model, you can continue working with the model as usual. No migration steps are required.
- If you selected one of the above interfaces, you must select a different bypass interface type offered in the Setup block and reconfigure all affected blocks accordingly.
- If you selected one of the above interfaces but no bypass interface type is offered for selection in the Setup block, you must import another database file with suitable interface definitions (`IF_DATA` entries) in the Setup block.

For more information, refer to Migrating Models for DCI-GSI1-Based Interfaces (RTI Bypass Blockset Reference 🕮).

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

The current Release contains RTI Bypass Blockset 3.17, 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 from the prior RTI Bypass Blockset versions. If you have a Simulink model that was built with RTI Bypass Blockset 2.5 or earlier and you open it with RTI Bypass Blockset 3.16, the old Data Dictionary file (with the file name extension .dd) is replaced by a new Data Dictionary file (.vdb) using the information stored in the Setup block. This step is performed automatically when you open and close the Setup block dialog by clicking **OK**, or when you open the Read, Write, Upload, or Download block dialog and click **Fill Variable Selector** on the Variables page.

  If you have a model that was saved with RTI Bypass Blockset 3.17 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 step is performed when you update the A2L files in the Setup block, or when you open the Read, Write, Upload, or Download block and click **Fill Variable Selector** on the Variables page. The Data Dictionary file created under RTI Bypass Blockset 3.17 (.vdb) remains on the disk.

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

- *Working with models from RTI Bypass Blockset 2.6 up to and including RTI Bypass Blockset 3.16*

  If a Simulink model was built with RTI Bypass Blockset 2.6 up to RTI Bypass Blockset 3.16 and you open it with RTI Bypass Blockset 3.17, 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 and including RTI Bypass Blockset 3.16, you have to create a suitable database in the earlier RTI Bypass Blockset version by reimporting the database files (A2L files) specified in the Setup block.

# RTI CAN MultiMessage Blockset

**Where to go from here**

Information in this section

## New Features of the RTI CAN MultiMessage Blockset 5.7

**Support of complete names of frames and signals**

The RTI CAN MultiMessage Blockset now supports the import of complete names of frames and signals from the communication matrix. This applies when you use a DBC file, FIBEX file, or AUTOSAR system description file as the database. In previous versions of RTI CAN MultiMessage Blockset, the length of the names of imported frames and signals was limited to 56 characters. The RTI CAN MultiMessage Blockset shortened the names during import, if necessary.

By default, the RTI CAN MultiMessage Blockset now uses the long names. However, you can switch to using short names. The RTI CAN MultiMessage Blockset provides the MATLAB *name_shortening* preference for this purpose. To use the shortened names, you must enter the following command in the MATLAB Command Window: `rtimmsu_private('fcnlib', 'name_shortening', 'can', 'enable');`

**Support of debounce time functionality for GTS messages**

The RTI CAN MultiMessage Blockset supports the debounce time functionality when sending global time synchronization (GTS) messages.

For more information, refer to Aspects of Miscellaneous Supported AUTOSAR Features (RTI CAN MultiMessage Blockset Reference 🕮).

**Reset functionality of the RTICANMM MainBlock during simulation**

If an Enable "Reset" checkbox is selected on the Code Options page, the RTI CAN MultiMessage Blockset now resets all settings of the RTICANMM MainBlock to their default values during run time, i.e., also the outport values.

For more information, refer to Code Options Page (RTICANMM MainBlock) (RTI CAN MultiMessage Blockset Reference 🕮).

## Migrating to RTI CAN MultiMessage Blockset 5.7

**Working with models from earlier RTI CAN MultiMessage Blockset versions**

To reuse a model created with an earlier RTI CAN MultiMessage Blockset version, you must update the S-functions for all the RTICANMM blocks and save the model before modifying the CAN configuration.

To create new S-functions for all the RTICANMM blocks in a model in one step, you can perform one of the following actions after opening the model:

- In the MATLAB Command Window, enter `rtimmsu_update('System', bdroot)`.

  For more information on the command and its options, enter `help rtimmsu_update` in the MATLAB Command Window.
- Select the Create S-Function for all CAN Blocks command from the Options menu of the RTICANMM GeneralSetup block.

For more information, refer to Limitations with RTICANMM (RTI CAN MultiMessage Blockset Reference 🕮).

**Additional migration steps in some cases**

To migrate to the RTI CAN MultiMessage Blockset 5.7 from versions earlier than the RTI CAN MultiMessage Blockset 5.6 and reuse existing models, you might have to carry out additional migration steps. For more migration information, refer to Migration (RTI CAN MultiMessage Blockset Reference 🕮).

# RTI FPGA Programming Blockset

**Where to go from here**

Information in this section

## New Features of the RTI FPGA Programming Blockset 3.12

**Extended Xilinx® support**

The RTI FPGA Programming Blockset now supports the following products and versions of the Xilinx design tools:

| Xilinx Design Tools Version | MATLAB Version[1] | Operating System |
|---|---|---|
| Vivado 2021.1[2] | ▪ MATLAB R2020a<br>▪ MATLAB R2020b<br>▪ MATLAB R2021a | Windows operating system that is supported by the RCP and HIL software of the current Release.<br>For a list of supported operating systems, refer to Operating System on page 202. The listed Windows Server operating systems are not officially supported by Xilinx, but tested by dSPACE. |

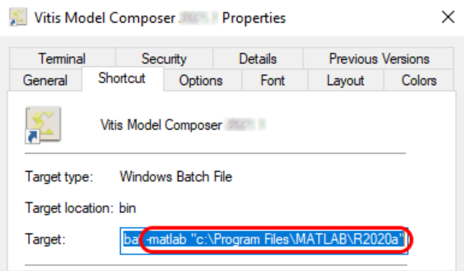
[1] The Processor Interface sublibrary of the RTI FPGA Programming Blockset also supports MATLAB R2021b.

[2] The Vivado ML Standard Edition of the Xilinx design tools also support the DS2655 (7K160) and DS6601 FPGA base boards. A separate license for the Xilinx Vitis Model Composer is required for modeling FPGA applications with the RTI FPGA Programming Blockset.

As of Vivado 2021.1 the Xilinx System Generator for DSP is part of the Vitis Model Composer. This is the unified Xilinx add-on for MATLAB Simulink. For licensing issues related to this change, refer to https://xilinx.com/support/answers/76039.html.

**Tip**

To use the Vitis Model Composer with a specific MATLAB version, the `-matlab "<matlabdir>"` parameter must be specified. You can add this parameter to the Vitis Model Composer icon on the desktop, for example.



Target type: Windows Batch File

Target location: bin

Target: ba -matlab "c:\Program Files\MATLAB\R2020a"

## Framework enhancements

**New function blocks for the processor communication** The new Bus In/Bus Out blocks for the MicroAutoBox III/SCALEXIO frameworks replace the transfer mode of the Buffer64 In/Buffer64 Out blocks.

The new Blocks support the following new features:

- You can copy a Simulink bus topology from a corresponding Data Inport/Data Outport block of the processor model.
- A bus editor lets you change the bus topology, including the bus signals of the Simulink bus used.

For more information, refer to Using Simulink Buses for Modeling the Processor Communication (RTI FPGA Programming Blockset Guide 📖).

**New ports for the MGT interfaces** The Aurora 64b66b In and Aurora 64b66b 128 Bit In blocks now provide a Ready port to specify that the communication channel is ready to read new data. The new port lets you prevent data overflow.

Furthermore, the MGT In block now provides an Opto_Ready port. The port indicates whether the MGT module is ready for data exchange.

*Handcoding interface only:* The Aurora 64b66b I/O functions provide a `mgt_user_clk_156` clock port. You can use the clock port to implement parts of your FPGA application synchronously with the clock frequency of the MGT module, i.e., with 156.25 MHz.

Working with synchronous frequencies has the following benefits:

- Less latency because there is no clock transition between the clock frequencies of the FPGA and the MGT module.
- No need for clock compensation due to drifting clocks. The Aurora protocol automatically synchronizes the different clocks.

## Enhancements to the script interface

For more information, refer to Script Functions Supporting the FPGA Interface Sublibrary (RTI FPGA Programming Blockset Script Interface Reference 📖).

**New script function for MicroAutoBox III/SCALEXIO** The script interface is enhanced with a script function to configure a Bus In/BusOut block with the bus configuration of the coresponding processor block.

For more information, refer to CopyProcFPGAXDATABus (RTI FPGA Programming Blockset Script Interface Reference 📖).

**Related topics**

Basics

# Migrating to the RTI FPGA Programming Blockset 3.12

**Introduction**

There are various ways to migrate an existing model, depending on the blockset version used.

**Migrating from previous RTI FPGA Programming Blocksets**

If you implemented an FPGA application with the RTI FPGA Programming Blockset Version 1.1 and later and want to use it with the RTI FPGA Programming Blockset 3.12, the framework automatically updates itself to the current framework version.

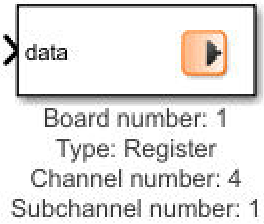
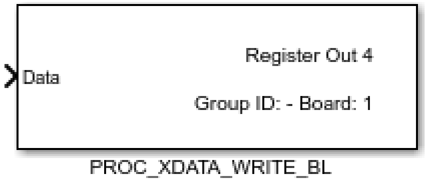
The update affects all the subsystems in the model/subsystem. The parameters of the blocks stay the same after updating to the current framework version.

You can also use a script to migrate processor interface blocks, for example, to migrate RTI processor models without the FPGA model. For more information, refer to MigrateToModelPortBlocks (RTI FPGA Programming Blockset Script Interface Reference 📖).

**Bus In blocks replace Buffer64 blocks** As of RTI FPGA Programming Blockset 3.12, the Bus In/Bus Out blocks for the MicroAutoBox III/SCALEXIO frameworks replace the transfer mode of the Buffer64 In/Buffer64 Out blocks. The update process automatically replaces the Buffer64 In/Buffer64 Out blocks that use the bus transfer mode with Bus In/Bus Out blocks.

**Display of migrated processor interfaces** With the RTI FPGA Programming Blockset 3.4 ... 3.8, you modeled the processor interface of a SCALEXIO system/MicroAutoBox III with processor interface blocks of the Processor Interface sublibrary.

If you migrate a model with processor interface blocks of the Processor Interface sublibrary, the update process migrates these blocks to the model port blocks of the Model Interface Package for Simulink. The following illustrations provide an example.

| Display After Migration | Display Before Migration |
|---|---|
|  data<br>Board number: 1<br>Type: Register<br>Channel number: 4<br>Subchannel number: 1 |  Data<br>Register Out 4<br>Group ID: - Board: 1<br>PROC_XDATA_WRITE_BL |

### ConfigurationDesk custom functions incompatible with current dSPACE Release

FPGA custom function block types that are not built with the RTI FPGA Programming Blockset 3.12 might be incompatible with the current ConfigurationDesk version.

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

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

### Using different dSPACE hardware

Using an FPGA model on different dSPACE hardware requires some model modifications. Refer to Migrating to Different FPGA Hardware (RTI FPGA Programming Blockset Guide 📖).

# RTI LIN MultiMessage Blockset

## Migrating to RTI LIN MultiMessage Blockset 3.7

**Working with models from earlier RTI LIN MultiMessage Blockset versions**

To reuse a model created with an earlier RTI LIN MultiMessage Blockset version, you must update the S-functions for all the RTILINMM blocks and save the model before modifying the LIN configuration.

To create new S-functions for all the RTILINMM blocks in a model in one step, you can perform one of the following actions after opening the model:

- In the MATLAB Command Window, enter `rtimmsu_update('System', bdroot)`.

  For more information on the command and its options, enter `help rtimmsu_update` in the MATLAB Command Window.
- Select the **Create S-Function for all LIN Blocks** command from the **Options** menu of the RTILINMM GeneralSetup block.

For more information, refer to Limitations of RTI LIN MultiMessage Blockset (RTI LIN MultiMessage Blockset Reference 🕮).

# SCALEXIO Firmware

**Where to go from here**

Information in this section

## New Features of the SCALEXIO Firmware 5.2

**Hypervisor Extension**

The SCALEXIO Hypervisor Extension adds a hypervisor to the new Linux-based operating system. The hypervisor is based on KVM and lets you create one or more virtual machines that run on any Linux distribution (Ubuntu 18.04 is tested and recommended).

**HypervisorShmem software package** As a supplement to the SCALEXIO Hypervisor Extension, the HypervisorShmem software package is now available. It allows for easy communication between applications on different operating systems, especially between SCALEXIO real-time applications and Linux applications.

The HypervisorShmem software package provides specific hypervisor function block types for ConfigurationDesk and supports the following methods of communication:

- Communication via shared memory.
- Communication via interrupts.

Additionally, the HypervisorShmem software package automatically manages synchronized data transfer.

**Communication between operating systems using the 9p protocol** The SCALEXIO Hypervisor Extension now supports the 9p protocol as an additional method for communication between operating systems via shared file systems.

A shared folder can be used to exchange data between operating systems as follows:

- Between Linux applications running on virtual machines and SCALEXIO real-time applications running under the SCALEXIO real-time operating system.
- Between Linux applications running on different virtual machines.

Using the 9p protocol is now the recommended method for communication between operating systems via shared file systems. As an alternative method, communication via a network file system (NFS) and an NFS server is still supported.

**Performance page for the Hypervisor page of the SCALEXIO Web Interface** The Hypervisor page of the SCALEXIO Web Interface now includes the Performance page. The Performance page provides graphical displays of the CPU time currently used by each running virtual machine and of the current data transfer rate of the Ethernet devices of each running virtual machine.

## Migrating to SCALEXIO Firmware 5.2

**Migrating to the Linux-based operating system**

With dSPACE Release 2020-B, dSPACE changed the SCALEXIO firmware from a QNX-based to a Linux-based distribution. The following items built for dSPACE Release 2020-A and earlier are no longer compatible with the SCALEXIO system and must be (re-)built from source code based on dSPACE Release 2020-B or later:

- Real-time applications
- Binary libraries contained in model containers (i.e., SIC, BSC, FMU, and CTLGZ files)
- Binary libraries referenced by Simulink models
- Binary libraries referenced by ConfigurationDesk applications via custom code settings or custom I/O functions

Source code is expected to be reusable in most cases. Cases that might require an adaptation of C code are related to custom code that uses special features (e.g., OS-specific functions) created by your company or third-party suppliers.

**Hypervisor Extension**

If the new SCALEXIO Hypervisor Extension is installed, SCALEXIO real-time applications as well as Linux real-time and non-real-time applications created for dSPACE Release 2020-A and older can no longer be used. You must rebuild the SCALEXIO real-time applications and adapt Linux real-time and non-real-time applications to the new hypervisor, especially regarding interrupts, shared memory, and the assignment of hardware resources.

# Sensor Simulation

## New Features of Sensor Simulation 1.6

**SensorSim application**

There are no new features in the SensorSim application for Sensor Simulation 1.6.

**Related topics**

Basics

MotionDesk Sensor Simulation Control
Sensor Simulation Manual
Sensor Simulation Overview

# SystemDesk

**Where to go from here**

Information in this section

# New Features of SystemDesk 5.6

**Where to go from here**

Information in this section

## New General Features

**Classic Platform support by SystemDesk 5.6**

**AUTOSAR release for modeling** SystemDesk lets you model Classic Platform software and system architectures with a data model according to the AUTOSAR R20-11 Release. However, SystemDesk also lets you exchange data of other AUTOSAR releases.

**Data exchange support** SystemDesk supports AUTOSAR R20-11, R19-11, 4.4.0, 4.3.1, 4.3.0, 4.2.2, 4.2.1, 4.1.3, 4.1.2, 4.1.1, 4.0.3, and 4.0.2 for data exchange.

**Adaptive Platform support**

SystemDesk now supports AUTOSAR R20-11 for developing Adaptive Platform software. For exchanging data, AUTOSAR R20-11, R19-11, and R19-03 are supported.

## Exporting V-ECU FMUs

**Introduction**

SystemDesk now lets you export V-ECUs according to the FMI standard.

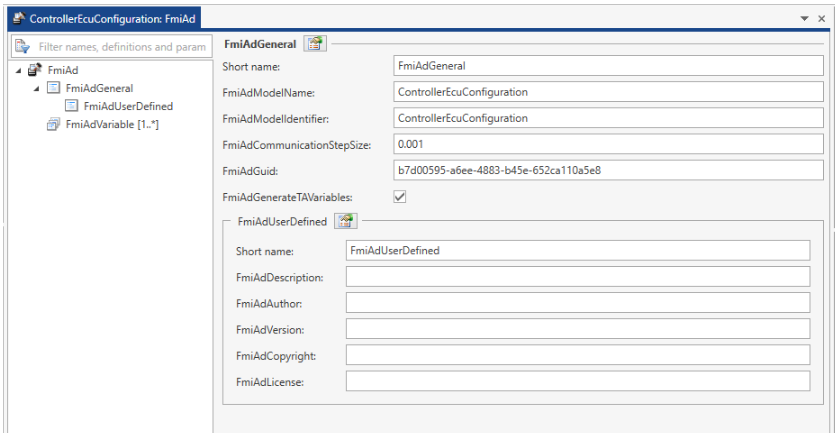
V-ECU FMUs are intended to support OEM-supplier scenarios where not all partners use the dSPACE SIL solution. Although the FMI standard does not support all features of the dSPACE SIL solution, you can use V-ECU FMUs for scenarios where the limitations of the FMI standard are acceptable.

For detailed and up-to-date compatibility information, refer to the following website: http://www.dspace.com/go/FMI-Compatibility.

## Configuring ECUs for FMU export

To execute ECU software on a platform, you have to configure its basic software. AUTOSAR has specified ECU configuration parameters that basic software vendors support for configuration on real hardware. dSPACE has adapted ECU configuration parameter definitions of the MCAL modules and provides related BSW modules to support the execution of ECU software on the dSPACE simulator.

dSPACE now provides adapted MCAL modules that support the execution of ECU software on FMI-compliant simulators. The BSW configuration parameter definitions are identical with those for the dSPACE simulator in most cases. However, bus communication requires specific handling. In addition, you have to use an FMI adapter module for the definition of exchanged FMI variables and FMI specifications, such as the **FMI Model Name** and **FMI Step Size**.


ControllerEcuConfiguration: FmiAd
Filter names, definitions and param
FmiAd
FmiAdGeneral
FmiAdUserDefined
FmiAdVariable [1..*]

FmiAdGeneral

Short name: FmiAdGeneral

FmiAdModelName: ControllerEcuConfiguration

FmiAdModelIdentifier: ControllerEcuConfiguration

FmiAdCommunicationStepSize: 0.001

FmiAdGuid: b7d00595-a6ee-4883-b45e-652ca110a5e8

FmiAdGenerateTAVariables: ☑

FmiAdUserDefined

Short name: FmiAdUserDefined

FmiAdDescription:

FmiAdAuthor:

FmiAdVersion:

FmiAdCopyright:

FmiAdLicense:

Refer to Basics on FMI-Compliant ECU Configuration (SystemDesk Manual 📖).

## Exporting V-ECU FMUs

After creating a V-ECU with ECU software that is configured for execution on an FMI-compliant simulator, you can export it to an FMU. SystemDesk validates the V-ECU before export, e.g., to detect missing files.

You can specify the simulation platform during export, i.e., the supported simulation hardware and compiler. SystemDesk lets you export FMUs for x86 and 64-bit processors that can be compiled with GCC or MSVC. Linux is not supported.

Refer to How to Export V-ECU FMUs (SystemDesk Manual 📖).

## Related topics

Basics

Basics on V-ECU FMUs (SystemDesk Manual 📖)

# Configuring ECUs

**COM stack support**

The CAN interface and LIN interface modules now support the MCAL CAN and LIN driver modules, i.e., dSPACE now provides a comprehensive COM stack for SIL testing. This supports scenarios where application software and third-party BSW is integrated in V-ECUs.

**Signal-to-frame converter (Sfc)**

The dSPACE-specific signal-to-frame converter module now supports group signals.

You can add the signal-to-frame converter module to an ECU. The module lets you connect V-ECUs with modeled bus communication to environment models or other V-ECUs that use signal-based communication, such as SIC and FMU files, or buses that are not supported. The bus signals can be transferred. However, bus timings cannot be simulated using the signal-to-frame converter.

Refer to Signal-to-Frame Converter (Sfc) (SystemDesk Manual 📖).

**Resource file manager**

The resource file manager lets you use resource files for the persistent storage of EEPROM, Flash, and NVRAM memory in simulations. You can define load configurations for loading memory from a resource file when a simulation is started. Save configurations let you store memory in a resource file when a simulation is stopped.

Refer to Resource File Manager (Rfm) (SystemDesk Manual 📖).

# Generating V-ECUs

**Specifying preprocessor definitions**

You can now store preprocessor definitions in `SoftwareDescription.xml` files of V-ECU implementation containers (VECU). The definitions are used to build V-ECUs.

Refer to Basics on Preprocessor Definitions (SystemDesk Manual 📖).

**Support for external files**

You can now add external files to V-ECUs that are referenced in exported V-ECU container files.

**Related topics**

Basics

Creating V-ECUs for SIL Testing (SystemDesk Manual 📖)

# Migrating to SystemDesk 5.6

**Where to go from here**

Information in this section

## Discontinuations as of dSPACE Release 2022-B

**Discontinuation of SWC container exchange**

As of dSPACE Release 2022-B, exchanging SWC containers between TargetLink and SystemDesk will no longer be supported. However, you can still exchange software components using ARXML files. With dSPACE AUTOSAR Compare, you can compare and merge these files.

The discontinued software parts include:

- In SystemDesk, the Container File Explorer and the commands for preparing and exchanging SWC containers.
- In TargetLink, the Data Dictionary commands for exchanging SWC containers.
- The Container Manager, which is a stand-alone tool for managing the contents of SWC containers. It will no longer be available from TargetLink or SystemDesk.

**Discontinuation of views on ECU configurations**

As of dSPACE Release 2022-B, the ECU Configuration Manager will no longer support different views.

You will no longer be able to switch to the AUTOSAR View or Dependency View. However, all SystemDesk features for configuring ECUs will still be available in the default Flat List View.

# Migrating to SystemDesk 5.6

### Automatic migration of projects

SystemDesk 5.6 automatically migrates SystemDesk 5.4, and 5.5 SDP project files when it starts.

> **Note**
>
> You are recommended to install the most recent patch for SystemDesk 5.4 or 5.5. Then, save the SDP project files you want to migrate before opening them in SystemDesk 5.6.

### Replacing the DsIdBusIf module configuration

As of dSPACE Release 2021-B, the DsIdBusIf module configuration is discontinued. You now have to use the Sfc module configuration instead. It is recommended to manually replace the DsIdBusIf module configuration in older SystemDesk projects.

For more information, refer to Signal-to-Frame Converter (Sfc) (SystemDesk Manual 🕮).

### Migrating scripts for automating SystemDesk

The SystemDesk API was changed as of SystemDesk 5.6. Some interfaces were added compared to SystemDesk 5.5 and certain interfaces were changed.

For more information, refer to *API Changes from SystemDesk 5.5 to SystemDesk 5.6* in the *SystemDesk API Reference*, which is available only in dSPACE Help.

# SYNECT

**Where to go from here**

Information in this section

# New Features of SYNECT 2.12

**Where to go from here**

Information in this section

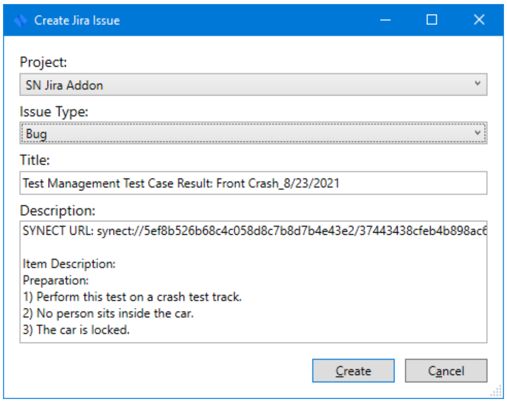
## New General Features of SYNECT

**SYNECT Jira add-on**

You can now connect Atlassian Jira software to handle issues, such as defects or tasks, with SYNECT.

The SYNECT Jira add-on lets you create Jira issues for specific SYNECT items. This lets you integrate SYNECT in a tool chain for issue handling. In such a tool chain, Jira is used to handle issues, while you can use SYNECT for their identification.

Refer to the following illustration, which shows how to create a Jira issue for a test case result.



Project:
SN Jira Addon

Issue Type:
Bug

Title:
Test Management Test Case Result: Front Crash_8/23/2021

Description:
SYNECT URL: synect://5ef8b526b68c4c058d8c7b8d7b4e43e2/37443438cfeb4b898ac6

Item Description:
Preparation:
1) Perform this test on a crash test track.
2) No person sits inside the car.
3) The car is locked.

**Improved user management**

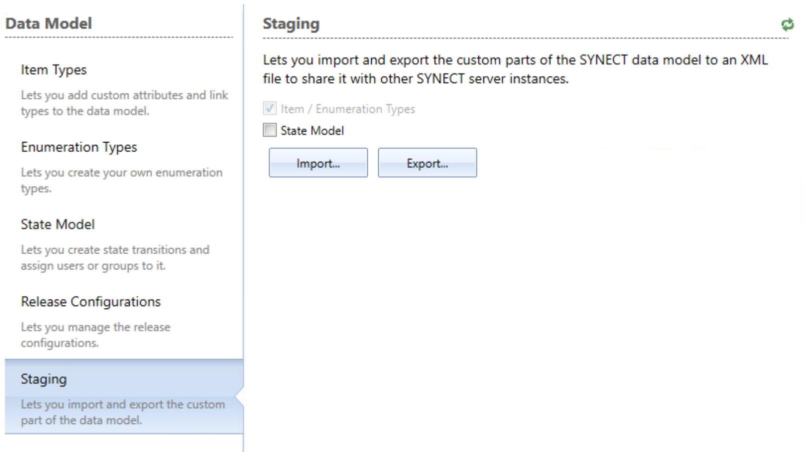
The user management of SYNECT has been improvided to ease the work of administrators.

- Managing SYNECT user licenses can now be performed via the API.

- SYNECT users for external authentication providers are now automatically created in the deactivated state when an authenticated user without a SYNECT account tries to log on to SYNECT. A SYNECT user administrator only has to activate the SYNECT account. This applies to Windows Users and OpenID Connect-compliant users, such Keycloak users.

## Improvements for state models

- Editing items with an underlying state model has been improved. The data grids of SYNECT only display the state transitions that are valid and can be switched with your user login.
- A state model for the implementation of test cases was added to the Central Locking demo. The state model supports a development process between test developers using AutomationDesk and test managers using SYNECT.
- Exchanging the data model of SYNECT servers has been improved. You can now exchange state models in addition to exchanging custom attributes and enumeration types. This lets you develop state models on test servers and roll them out on production servers.



Data Model

Item Types

Lets you add custom attributes and link types to the data model.

Enumeration Types

Lets you create your own enumeration types.

State Model

Lets you create state transitions and assign users or groups to it.

Release Configurations

Lets you manage the release configurations.

Staging

Lets you import and export the custom part of the data model.

Staging

Lets you import and export the custom parts of the SYNECT data model to an XML file to share it with other SYNECT server instances.

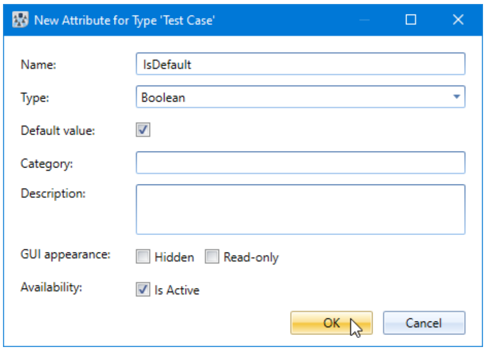
☑ Item / Enumeration Types

□ State Model

## Default values of custom attributes

You can now specify default values for custom attributes when you extend the data model of SYNECT. This applies to custom attributes of base types, i.e., Boolean, integer, double, string, and enumeration types.

Refer to the following illustration.



New Attribute for Type 'Test Case'

Name: IsDefault

Type: Boolean

Default value: ☑

Category:

Description:

GUI appearance: □ Hidden □ Read-only

Availability: ☑ Is Active

**Limited support for system model containers**

SYNECT model management supports the integration of system models and lets you build system models for SIL testing in VEOS. A system model contains different components that can be imported to SYNECT from container files.

With dSPACE Release 2020-B, the container format for V-ECU implementation containers changed. The new VECU format improves the data exchange between ECU software suppliers and OEMs and provides multiplatform support.

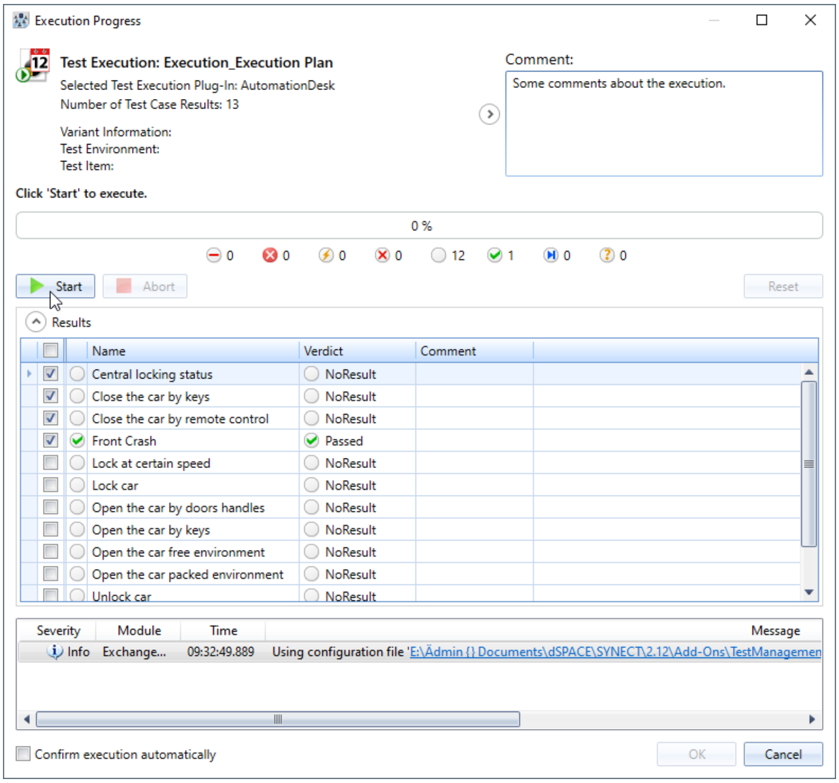
SYNECT does not support VECU containers. You cannot integrate system models with dSPACE products of dSPACE Release 2020-B or later. This affects the import of the dSPACE-specific VECU, SIC, and BSC containers as well as the exchange of system models with VEOS using SMC containers. However, the FMU 2.0 support and the support for the dSPACE-specific model containers of earlier dSPACE Releases has not changed.

## New Features of Test Management

**Execute with manual start**

You can now start the execution of executions and evaluations in the Progress dialog. The dialog lets you edit test cases and evaluation functions and select them for execution.

Refer to the following illustration, that shows the selection of four test cases for execution.



**Test Execution: Execution_Execution Plan**
Selected Test Execution Plug-In: AutomationDesk
Number of Test Case Results: 13

Variant Information:
Test Environment:
Test Item:

Comment: Some comments about the execution.

**Click 'Start' to execute.**

0 %

⊖ 0 ✖ 0 ⚡ 0 ✖ 0 ○ 12 ✓ 1 ⏭ 0 ? 0

| ☐ | | Name | Verdict | Comment |
|---|---|---|---|---|
| ☑ | ○ | Central locking status | ○ NoResult | |
| ☑ | ○ | Close the car by keys | ○ NoResult | |
| ☑ | ○ | Close the car by remote control | ○ NoResult | |
| ☑ | ✓ | Front Crash | ✓ Passed | |
| ☐ | ○ | Lock at certain speed | ○ NoResult | |
| ☐ | ○ | Lock car | ○ NoResult | |
| ☐ | ○ | Open the car by doors handles | ○ NoResult | |
| ☐ | ○ | Open the car by keys | ○ NoResult | |
| ☐ | ○ | Open the car free environment | ○ NoResult | |
| ☐ | ○ | Open the car packed environment | ○ NoResult | |
| ☐ | ○ | Unlock car | ○ NoResult | |

| Severity | Module | Time | Message |
|---|---|---|---|
| Info | Exchange... | 09:32:49.889 | Using configuration file 'E:\Ädmin {} Documents\dSPACE\SYNECT\2.12\Add-Ons\TestManagemen |

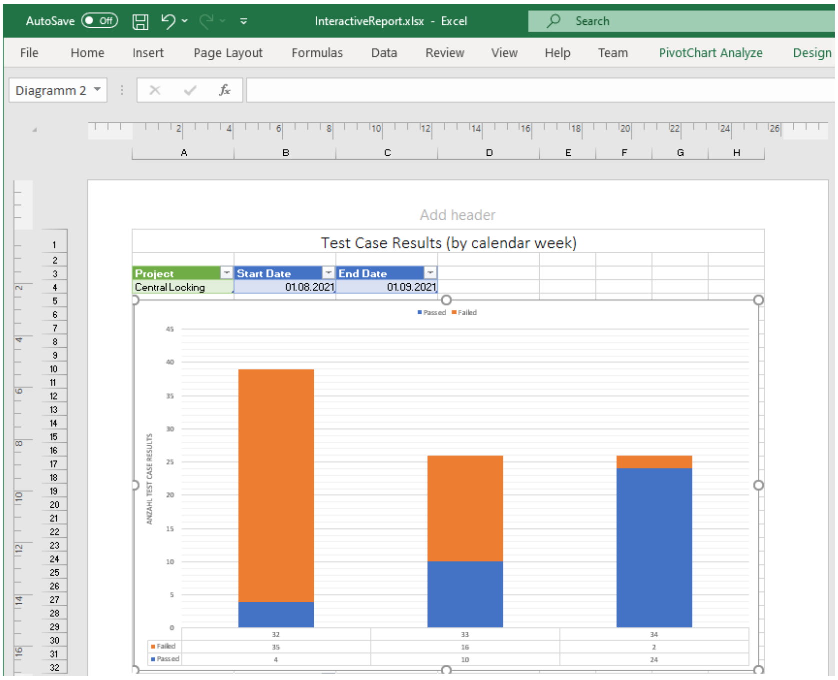
☐ Confirm execution automatically

### Interactive Excel Report add-on

SYNECT now provides the Interactive Excel Report add-on, which lets you load data on request via the OData service of the SYNECT server to create Pivot charts with Microsoft® Excel®.

The add-on opens a preconfigured Excel file that provides a demo report on the progress of test management projects with Pivot charts.

You can customize the layout of the Pivot charts as required. Refer to the following illustration.



Test Case Results (by calendar week)

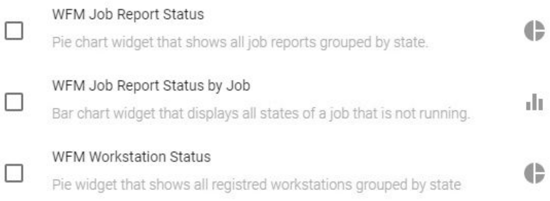
| Project | Start Date | End Date |
|---|---|---|
| Central Locking | 01.08.2021 | 01.09.2021 |

| | 32 | 33 | 34 |
|---|---|---|---|
| Failed | 35 | 16 | 2 |
| Passed | 4 | 10 | 24 |

## New Features of Workflow Management

### Dashboard widgets

SYNECT now provides widgets for dashboards that you can use to monitor job reports and the availability of workstations.



☐ WFM Job Report Status
Pie chart widget that shows all job reports grouped by state.

☐ WFM Job Report Status by Job
Bar chart widget that displays all states of a job that is not running.

☐ WFM Workstation Status
Pie widget that shows all registred workstations grouped by state

# Migrating to SYNECT 2.12

**Where to go from here**

Information in this section

## Migrating Databases

**Introduction**

To use the data from previous SYNECT versions with SYNECT 2.12, you have to migrate the SYNECT database.

To migrate databases from SYNECT Versions 2.0 - 2.11 to SYNECT 2.12, use the Database Migrator of SYNECT 2.12.

**Note**

Contact dSPACE Support if you want to migrate SYNECT versions earlier than SYNECT 2.0. Refer to www.dspace.com/go/supportrequest.

For basic information and instructions on migrating databases, refer to Migrating Databases from Previous SYNECT Versions (The SYNECT Server Guide 📖).

## Migrating from SYNECT 2.11

**Migrating server scripts**

The generic EnumerationType property that was defined on the AttributeType class was removed.

You can use the EnumerationType property of the EnumAttributeType class instead.

**Old listing** Up to Synect 2.11, each instance of an AttributeType implemented the EnumerationType property, even if the instance was no EnumAttributeType.

Refer to the following listing, which uses the generic EnumerationType property in line number 4.

```
1 workspaceType = context.GetItemTypeByName('Workspace')
2 enumerationTypes = []
3 for attributeType in workspaceType.AttributeTypes:
4     if attributeType.EnumerationType is not None:
5         enumerationTypes.append(attributeType.EnumerationType)
```

**New listing** You can test if an instance of AttributeType is also an instance of EnumAttributeType before using the EnumerationType property. Refer to line number 5 of the following listing.

```
1 from dSPACE.Synect.Server.Extension.MetaModelEditor import EnumAttributeType
2 workspaceType = context.GetItemTypeByName('Workspace')
3 enumerationTypes = []
4 for attributeType in workspaceType.AttributeTypes:
5     if isinstance(attributeType, EnumAttributeType):
6         enumerationTypes.append(attributeType.EnumerationType)
```

Refer to EnumAttributeType (SYNECT Server API Reference 🕮).

## Using client events

You can use client events to handle situations, such as importing test cases. The event handler provides an argument with situation-specific properties for this. Refer to the following listing from the client event demo, which you can find in the demos of SYNECT at `Demos\TestManagement\Programming\ClientEventDemo Add-On\Source\Client\ClientEventDemo.py`.

```
class TestCasesImported(object):
    """Class: Test Cases Imported."""
    def OnInvoke(self, eventSource, arguments):
        """Called after test cases are imported."""
        arguments = Dispatch(arguments)
        projectId = arguments.projectId
        importedFile = arguments.FullPath
        createdTestCases = arguments.CreatedTestCases
        updatedTestCases = arguments.UpdatedTestCases
        unmodifiedTestCases = arguments.UnmodifiedTestCases
        plugin = Dispatch(arguments.Plugin)
```

**Changes of the argument properties** Up to SYNECT 2.11 all Imported event handlers erronously provided the CreatedTestCases, UpdatedTestCases, and UnmodifiedTestCases properties of the event handler argument.

As of SYNECT 2.12 the mentioned properties are available only for the TM.TestCasesImported event. They were removed from all the other Imported events, such as TM.TestCaseResultsImported or RM.DocumentsImported events.

**Required script migration** You must not use the CreatedTestCases, UpdatedTestCases, and UnmodifiedTestCases properties for events other than TM.TestCasesImported.

## Data Model Changes from SYNECT 2.11 to SYNECT 2.12

**No changes**

The data model did not change from SYNECT 2.11 to SYNECT 2.12.

# TargetLink

**Where to go from here**

## Information in this section

## Information in other sections

TargetLink New Features and Migration Guide
Provides information on new features, migration steps, discontinuations, and code changes of the different TargetLink releases.

# New Features of TargetLink 5.2

**Where to go from here**

Information in this section

# Modeling in Simulink or Stateflow

## Improved Array-of-Struct Support

**Dynamic access to array-of-struct variables**

TargetLink 5.2 supports the dynamic access to array-of-struct variables via special modeling patterns consisting of the following blocks:

- Data Store Read and Data Store Write blocks
- Bus Inport and Simulink Inport blocks
- Bus Outport and Simulink Outport blocks
- Bus Selector and Selector blocks
- Assignment block

**Related documentation:**

- Overview of Modeling Patterns for Accessing Arrays of Buses (TargetLink Preparation and Simulation Guide 🕮)
- How to Model the Dynamic Access to Array-of-Struct Variables via Data Stores (TargetLink Preparation and Simulation Guide 🕮)
- Example of Modeling the Dynamic Access to a Plain-Array-of-Bus Signal via Bus Port Blocks (TargetLink Preparation and Simulation Guide 🕮)
- Example of Modeling the Dynamic Access to a Bus-with-Array-of-Bus-as-Payload signal via Bus Port Blocks (TargetLink Preparation and Simulation Guide 🕮)

# Adaptive AUTOSAR

## Enhanced Support for Adaptive AUTOSAR

**Error handling**

With TargetLink 5.2, you can model and simulate error handling as described by ara::com and ara::per.

**Related documentation**

- Basics on Modeling and Simulating Error Handling (Adaptive AUTOSAR) (TargetLink Adaptive AUTOSAR Modeling Guide 🕮)
- Introduction to Using TargetLink With Adaptive AUTOSAR (TargetLink Adaptive AUTOSAR Modeling Guide 🕮)

**Supported Adaptive AUTOSAR releases**

TargetLink 5.2 supports the AUTOSAR AP R20-11 release.

# Classic AUTOSAR

**Where to go from here**

Information in this section

## Supported Classic AUTOSAR Releases

**Supported Classic AUTOSAR Releases**

The following Classic AUTOSAR Releases are supported:

| Classic AUTOSAR Release | Revision |
|---|---|
| R20-11 | 20-11 [1)] |
| R19-11 | 19-11 |
| 4.4 | 4.4.0 |

| Classic AUTOSAR Release | Revision |
|---|---|
| 4.3 | 4.3.1<br>4.3.0 |
| 4.2 | 4.2.2<br>4.2.1 |
| 4.1 | 4.1.3<br>4.1.2<br>4.1.1 |
| 4.0 | 4.0.3<br>4.0.2 |

[1] New in TargetLink 5.2

## Classic AUTOSAR SWC Compositions

**Generating and updating AUTOSAR SWC compositions**

With TargetLink 5.2, it is possible to generate and update frame models for composition SWCs for simulation purposes. A composition SWC is represented as Simulink or TargetLink subsystem.

**Related documentation:**

- Basics on AUTOSAR Development Approaches and Frame Model Generation in TargetLink (TargetLink Classic AUTOSAR Modeling Guide 📖)
- How to Generate a Frame Model from Classic AUTOSAR Data (TargetLink Classic AUTOSAR Modeling Guide 📖)
- How to Update an Existing Frame Model from Classic AUTOSAR Data (TargetLink Classic AUTOSAR Modeling Guide 📖)
- `AR_COMPOSITION` (TargetLink Demo Models 📖) `demo model`

## Improved Array-Of-Struct Support

**Dynamically selecting elements of array-of-bus signals via data stores in Classic AUTOSAR**

With TargetLink 5.2, you can now use data store blocks to dynamically select elements from an array-of-bus signal.

**Related documentation**

- Basics on Working with Arrays of Structs When Modeling Classic AUTOSAR in TargetLink (TargetLink Classic AUTOSAR Modeling Guide 📖)
- Overview of Modeling Patterns for Accessing Arrays of Buses (TargetLink Preparation and Simulation Guide 📖)

**Modeling array-of-struct operations via bus port blocks in Classic AUTOSAR**

With TargetLink 5.2, you can now use bus port blocks to model server runnables and operation subsystems with operation arguments that have an array-of-struct data type.

**Related documentation**

- Basics on Working with Arrays of Structs When Modeling Classic AUTOSAR in TargetLink (TargetLink Classic AUTOSAR Modeling Guide 📖)
- Overview of Modeling Patterns for Accessing Arrays of Buses (TargetLink Preparation and Simulation Guide 📖)

# Target Simulation (PIL)

## Changes in the Target Simulation Modules

**New and discontinued compiler versions**

The following table shows the compiler versions that are now supported by TargetLink 5.2. Refer to the New and No changes columns. Compiler versions that are no longer supported are listed in the Discontinued column.

| Microcontroller Family | Compiler | New | No Changes | Discontinued |
|---|---|---|---|---|
| ARM CortexM3 | Keil | — | 5.2 | — |
| C16x | TASKING | — | 8.6 | — |
| MPC57xxVLE | Diab | — | 5.9 | — |
|  | GreenHill | 2021 |  | 2019 |
| RH850 | GreenHill | 2021 |  | 2019 |
| S12X | Cosmic | — | 4.8 | — |
|  | Metrowerk | — | 5.1 | — |
| SH2 | Renesas | — | 9.3 | — |
| SH2A-FPU | Renesas | — | 9.4 | — |
| TriCore17xx | TASKING | — | 3.2 | — |
| TriCore2xx | TASKING | — | 6.3 | — |
|  | TASKING | — | 4.2[1)] | — |
|  | GCC | — | 4.9 | — |
| TMS570 (ARM) | CCS | — | 7.0 | — |
| V850 | GreenHill | 2021 |  | 2019 |
| XC22xx | TASKING | — | 3.0 | — |

[1)] For Lauterbach simulator.

For more information on the evaluation boards supported by TargetLink, refer to Combinations of Evaluation Boards and Compilers (Evaluation Board Reference 🕮).

**Note**

For more PIL support combinations that are part of a valid Software Maintenance Service (SMS) contract, refer to the dSPACE TargetLink PIL Support website at the TargetLink Product Support Center.

# Usability

**Where to go from here**

Information in this section

## Data Dictionary

**Usability improvements**

TargetLink 5.2 provides several usability improvements in the Data Dictionary Manager. For example:

- Hyperlinks to almost all mentioned DD objects in the Message Browser.
- Close buttons for DD workspaces, i.e., to close the active DD workspace, and a context menu command to close multiple DD workspaces at once.
- Mechanisms to change the view of the Property Value List.

**Related documentation**

- Message Browser (TargetLink Data Dictionary Manager Reference 🕮)
- Property Value List (TargetLink Data Dictionary Manager Reference 🕮)

**Data Dictionary upgrade**

With TargetLink 5.2, the upgrade behavior of the Data Dictionary changed. A Data Dictionary upgrade is performed automatically. If a Data Dictionary is saved after upgrade, a backup of any original DD file is saved to the same directory as the DD file. This is also true for included DD files.

**Related documentation**

- Basics on Migrating Between TargetLink Versions on page 159

- How to Upgrade and Save a Data Dictionary with Included DD Files on page 161
- tl_pref (TargetLink API Reference 🕮)

**Improved load and save behavior for partial DD files**

Loaded partial DD files are embedded in a complete Data Dictionary. With TargetLink 5.2, the name of the DD workspace is identical to the name of the loaded partial DD file. When you save the DD workspace, the DD object tree into which the partial DD was loaded is saved instead of the complete Data Dictionary.

**Related documentation**

- Basics on Opening and Handling DD Files (TargetLink Data Dictionary Basic Concepts Guide 🕮)

## Block-Dialog-Related

**Creating struct variables**

It is now possible to create struct variables in the Data Dictionary directly from the block dialog of bus port blocks.

If the bus port block references a Simulink.Bus object *and* you enter a nonexistent name in the Variable edit field, TargetLink automatically creates a struct variable that matches the Simulink.Bus object, when you click Apply or OK. This is especially convenient when working with array-of-struct variables.

**Related documentation**

- How to Create DD Variable Objects via Block Dialog (TargetLink Preparation and Simulation Guide 🕮)

# Code Generator Options

## New Code Generator Options

**Overview of new Code Generator options**

The following new Code Generator options are available with TargetLink 5.2:

- EnableSharingOfSequentialControlVariables

  Lets you enable variable sharing for variables used for sequential control, e.g., to controll conditions of cyclic counters or first-run logic. This eliminates superfluous variables and allows for further control flow optimization.

**Related documentation**

- EnableSharingOfSequentialControlVariables (TargetLink Model Element Reference 🕮)

For reference information on all Code Generator options, refer to Alphabetical List of Code Generator Options (TargetLink Model Element Reference 📖).

**Migration aspects of Code Generator options**

For more information, refer to Migration Aspects Regarding Code Generator Options on page 167.

# API Functions and Hook Scripts

## New API Functions

**List of new API functions**

| API Function | Purpose |
|---|---|
| `ddExc` | The command-line interface of the exception-based Data Dictionary MATLAB API. |
| `tlARACoreError` | Provides simulation functionality to simulate error handling for different errors in Adaptive AUTOSAR. |
| `tlBatchMode` | Gets or sets the TargetLink batch mode. |
| `dsdd('SetDDModified')` | The status of the Data Dictionary workspace is set to saved/not modified or to unsaved/modified. |

**Related topics**

References

ddExc (TargetLink API Reference 📖)
SetDDModified (TargetLink Data Dictionary Reference 📖)
tlARACoreError (TargetLink API Reference 📖)
tlBatchMode (TargetLink API Reference 📖)

# Other

**Where to go from here**

Information in this section

## General Enhancements and Changes

**Probe block**

With TargetLink 5.2, the Probe block can be used as external Iteration Limit Source at the inport of the For Iterator block.

**Support for 64-bit integer data types**

With TargetLink 5.2, 64-bit integer data types can be used on various blocks.

**Related documentation**

- Basics on 64-Bit Integer Data Types (TargetLink Customization and Optimization Guide 🕮)

**Copy DD object references**

With TargetLink 5.2, DD object references in block dialogs can be copied to the clipboard via the Copy Full Path context menu command.

This also works for DD object references of DD Typedef objects displayed in AUTOSAR <Port> block dialogs. The copied strings are intended to be used as input in other model element dialogs.

**Related documentation**

- How to Modify Block Properties via Block Dialogs (TargetLink Preparation and Simulation Guide 🕮)
- Common TargetLink Context Menu Options (TargetLink Model Element Reference 🕮)

**Cleaning current working directory**

With TargetLink 5.2, the `tl_clean.bat` batch file prompts for user confirmation to delete the following folders if they exist:

- `.\CodeCoverageReport`
- `.\CodeViewFiles`
- `.\TLBuild`
- `.\TLProj`
- `.\TLSim`

The `.\doc` folder is not deleted.

You can skip the user confirmation prompt by calling `tl_clean.bat` with the `all` argument from a Command Prompt or using the system command in the MATLAB® Command Window.

**Related documentation**

- Files Generated During Frame Generation (TargetLink File Reference 🕮)
- tl_clean (TargetLink API Reference 🕮)

### Improved performance

There are performance improvements in TargetLink 5.2.

**Code generation** Code generation performance is improved for models that contain one of the following:

- Large busses at the root interfaces
- Many **DD Typedef** objects
- Large structured variables

**Data Dictionary API** The performance of the Data Dictionary API is improved. This also improves the performance of Data Dictionary validation and the Data Dictionary Manager, especially in the situations described above.

**Related documentation**

- Basics on Validating Data Dictionaries (TargetLink Data Dictionary Basic Concepts Guide 🕮)
- Validate

## TargetLink Demos and Examples

### New example

| New Example | Description |
|---|---|
| AR_COMPOSITION | The live script shows you how to generate and update a frame model from ARXML files that contain SWC compositions. |

### Modified demo models

| Changed Demo Model | Description |
|---|---|
| AAR_COMMUNICATION (TargetLink Demo Models 🕮) | The `AAR_COMMUNICATION` demo model shows you how to model communication and error handling for communication according to Adaptive AUTOSAR in TargetLink. |
| AUTOSAR_FUELSYS | The demo model now shows communication via NvM and EcuM service interfaces. |
| BUS_CC | The BUS_CC demo model shows Custom Code (type II) blocks that work with bus signals and the corresponding struct types as Custom Code interface variables. It now shows custom code access to a data store memory containing a complex array-of-bus signal. |
| BUS_STRUCT | The `bus_struct` demo model shows you the different modeling styles that TargetLink offers for implementing buses associated with struct types. |

| Changed Demo Model | Description |
|---|---|
| STRUCT_INTERFACES | The struct_interfaces demo model shows struct interfaces that do not correspond to buses in the model, and especially struct pointer function parameters. It now shows additional function parameters of struct pointer type. |

# Migrating to TargetLink 5.2 and TargetLink Data Dictionary 5.2

**Upgrade process**

Carefully read all of the following information and modify the tool chain accordingly.

**Where to go from here**

Information in this section

# General Migration Information

**Where to go from here**

Information in this section

# Upgrading Models, Libraries, and Data Dictionaries

**Where to go from here**

Information in this section

## Basics on Migrating Between TargetLink Versions

### Automatic upgrade from TargetLink 3.1 or later

TargetLink 5.2 automatically upgrades models and TargetLink-compliant libraries if they were created with TargetLink 3.1 or later.

The automatic upgrade includes all the steps required by the intervening TargetLink versions. For example, an automatic upgrade from TargetLink 4.0 to TargetLink 5.2 comprises the steps 4.0 to 4.1 to 4.2 to 4.3 to 4.4 to 5.0 to 5.1 to 5.2.

> **Note**
>
> Check the TargetLink migration documentation of the different TargetLink versions to see whether user interaction is required.

**User interaction required** In the following cases, for example, the automatic upgrade requires additional user interaction:

- Libraries must be TargetLink-compliant. Otherwise, no upgrade can be performed.
- Style sheets for code generation are version-specific and subject to change from one TargetLink version to another. Thus, modified style sheets of older TargetLink versions have to be updated to match the current version (reapplying the modifications as intended). Refer to Basics on Code Formatting (TargetLink Customization and Optimization Guide 📖).
- Custom code S-functions built with 32-bit TargetLink versions do not work with 64-bit versions of TargetLink.

  Initiate a rebuild of all custom code S-functions using the **`tlUpgrade('Model',<MyModel>,'CheckModel','FixIssues')`** API function.

### Upgrading Data Dictionaries

If you open a DD file that was last saved with a TargetLink version < 5.2, it must be upgraded. The upgrade is executed automatically. The **Subsystems** area and the **<Application>** area are no longer compatible and thus removed. To change the upgrade behavior, refer to tl_pref (TargetLink API Reference 📖).

If you save the upgraded DD file, the original DD file is automatically saved to a backup DD file in the same directory as the DD file before it is overwritten with the upgraded version. To disable the automatic backup, refer to **Data Dictionary Upgrade** in Topic Navigator (TargetLink Tool and Utility Reference 📖).

To upgrade DD files with included DD files, refer to How to Upgrade and Save a Data Dictionary with Included DD Files on page 161.

### Making new libraries TargetLink-compliant

Libraries that you create from scratch and that consist of TargetLink blocks must be made upward compatible so that you can upgrade them to a newer TargetLink version in the future. Otherwise, no upgrade can be performed.

> **Note**
>
> A library does not automatically become a TargetLink library if it contains TargetLink blocks. The library itself must be TargetLink-compliant.

Refer to How to Make User Libraries Upgrade-Capable (TargetLink Orientation and Overview Guide 🕮).

**Making existing libraries TargetLink-compliant**

The following two approaches let you make libraries created with earlier TargetLink versions compliant with the current TargetLink version 5.2:

**The earlier TargetLink version is available** Use the TargetLink version with which the library was created to make the library TargetLink-compliant. Refer to the TargetLink documentation of the earlier TargetLink version. You can then use this library with all later TargetLink versions because TargetLink automatically performs an upgrade. The library can still be used with TargetLink versions earlier than TargetLink 5.2 because the automatic upgrade does not save a library in the newer TargetLink version.

**Only the current TargetLink version 5.2 is available** Use TargetLink version 5.2 and the `tlUpgrade` API command to make the library TargetLink-compliant. Refer to How to Manually Upgrade Libraries and Models via the API on page 162. If you follow the instructions, the library is saved in TargetLink version 5.2. Therefore, it cannot be used with TargetLink versions earlier than TargetLink 5.2.

**Manual upgrade from TargetLink 2.x or 3.0.x**

Models and libraries created with TargetLink versions 2.x or 3.0.x have to be upgraded manually to the latest TargetLink version 3.x (3.1...3.5) you have. Afterwards, automatic upgrade is possible.

**No downward compatibility**

You cannot use models, libraries, and Data Dictionaries in the format of newer TargetLink versions in earlier TargetLink versions.

**Data model filter rule files**

Existing data model filter rule files can contain invalid elements because the data model of the TargetLink Data Dictionary changed. The following files that were shipped with previous TargetLink versions can be affected:

- `DD_Filter_Admin.xml`
- `DD_Filter_AR_User.xml`
- `DD_Filter_NonAR_NonRTOS_User.xml`

You can check filter rule files via the API in the MATLAB Command Window:

| Checking a Single File | Checking Filter Rule Sets[1)] |
|---|---|
| `dsdd_free;` | `dsdd_free;` |
| `dsdd('ReadFilterRuleSet', 'file', '<myFile>.xml');` | `dsdd('ReloadFilterRuleSets');` |
| `ds_error_register(dsdd('GetMessageList'));` | `ds_error_register(dsdd('GetMessageList'));` |
| `ds_msgdlg('update');` | `ds_msgdlg('update');` |

[1] All the files contained in the directory defined in Data Dictionary - Filter Rules in the Preferences Editor.

TargetLink informs you about errors in the TargetLink Message Browser. Each error contains the following information so that you can fix it in any XML-capable editor:

- File name
- Row number
- Column number

**Related topics**

Basics

Basics on Code Formatting (TargetLink Customization and Optimization Guide 📖)

HowTos

How to Make User Libraries Upgrade-Capable (TargetLink Orientation and Overview Guide 📖)
How to Manually Upgrade Libraries and Models via the API.................................................. 162

References

tlUpgrade (TargetLink API Reference 📖)

## How to Upgrade and Save a Data Dictionary with Included DD Files

**Precondition**

In the main DD file to be loaded, the AutoLoad property of the DD DDIncludeFiles objects is set to on.

**Method**

**To upgrade and save a Data Dictionary with included DD files**

**1** Open the Data Dictionary Manager and the main DD file via File - Open.

The main DD file and all included DD files are automatically upgraded to the latest data model revision.

**2** Set the AutoSave property of all desired `/Config/DDIncludeFiles/<DDIncludeFile>` objects to On to save the included DD files together with the main DD file.

**3** Save the main DD file via File - Save.

**Result**

You upgraded and saved the main DD file and all included DD files. TargetLink adjusted the revision number of every upgraded DD file to the latest data model revision. Backups of the original DD files are also saved to the same directory as the DD file.

### Related topics

Basics

Basics on Opening and Handling DD Files (TargetLink Data Dictionary Basic Concepts Guide 🕮)

HowTos

How to Include Partial Data Dictionary Files (TargetLink Data Dictionary Basic Concepts Guide 🕮)

References

Point of Inclusion (TargetLink Data Dictionary Manager Reference 🕮)

## How to Manually Upgrade Libraries and Models via the API

### Objective

To prepare a central upgrade of libraries and models in a tool chain with several users, for example.

### Preconditions

The model or library files are available on the MATLAB search path but they are not open.

The required DD project file has been opened and automatically upgraded, for example, via `dsdd_manage_project('Open','<name>.dd')` or `dsdd('Open', '<name.dd>')`.

### Method

**To manually upgrade libraries and models via the API**

**1** Type the following API command in the MATLAB Command Window:

```
tlUpgrade('Model', '<Model|Library>.slx',
'CheckModel','FixIssues')
```

The model or library is upgraded.

> **Note**
>
> When you upgrade models and libraries, first upgrade models or libraries that do not reference any other libraries, i.e., the libraries whose blocks and subsystems have no links to other libraries. Start with the bottom library and then upgrade the libraries above it in ascending order.

**2** Save the upgraded model or library files, e.g., `Library.slx`.

**3** Repeat steps 1 and 2 for all other models and libraries.

**Result**

You upgraded the models and libraries.

**Related topics**

References

tlUpgrade (TargetLink API Reference 📖)

## Migrating Data Dictionaries to CodeDecorationSets

**Introduction of CodeDecorationSet and CodeDecoration objects**

TargetLink 4.3 introduced DD CodeDecorationSet and CodeDecoration objects.

Additionally, several properties were removed from the Data Dictionary data model:

| DD Object | Change | Replacement |
|---|---|---|
| FunctionClass | Removal of the DeclarationStatements and SectionName properties. | The DeclarationStatements and SectionName properties of the DD CodeDecoration.Settings object. |
| VariableClass | | |
| VariableClassTemplate.Filter | Removal of the WidthSpec property. | The WidthSpec property of the DD CodeDecoration.Filter object. |

**Automatic upgrade by TargetLink**

**Limitation** TargetLink no longer supports width-specific type prefixes for variable classes. The automatic upgrade of the Data Dictionary fails if the original Data Dictionary contains variable class templates used to derive variable classes that have width-specific type prefixes.

Use declaration statements instead.

When you open a Data Dictionary whose data model is older than the latest revision, TargetLink prompts you to perform an automatic upgrade.

| Object Kind | Trigger | Upgrade Action |
|---|---|---|
| VariableClass | DeclarationStatements or SectionName properties are set. | 1. Create a DD CodeDecorationSet object.<br>2. Create a single DD CodeDecoration object for each DD CodeDecorationSet object.<br>The settings of the CodeDecoration object and its child objects match the settings of the original objects.<br>3. Reference the CodeDecorationSet object at the original object. |
| FunctionClass | | |
| SubStructTemplate | Filter.VariableClass is set. | Transfer the values of the following properties from the variable class to the SubStructTemplate object's filter:<br>▪ DeclarationStatements<br>▪ SectionName<br>▪ TypePrefix |

| Object Kind | Trigger | Upgrade Action |
|---|---|---|
| VariableClassTemplate | ▪ Filter.FilterCondition is set to `ALL_TRUE`.<br>▪ Settings.VariableClass references a DD VariableClass object whose DeclarationStatements or SectionName property is set.<br>▪ The Filter.WidthSpec property is set for this DD VariableClassTemplate object or for another VariableClassTemplate object whose Filter.VariableClassSpec property has the same value. | 1. Create a new DD VariableClass object in `/Pool/VariableClasses/Templates`.<br>2. Create a new DD CodeDecorationSet object in `/Pool/CodeDecorations/Templates`.<br>3. For each DD VariableClassTemplate object with the same value at the Filter.VariableClassSpec property, add a DD CodeDecoration object to the CodeDecorationSet object.<br>4. Specify the DD CodeDecoration object as required.<br>5. Reference the DD CodeDecorationSet object at the VariableClass object created in step 1.<br>6. Reference the DD VariableClass object created in step 1 via the VariableClassTemplate.Settings.VariableClass property. |

### Special considerations for variable class templates

If you specified DD VariableClassTemplate objects whose Filter.FilterCondition property is set to `ALWAYS` or `NEVER`, TargetLink deletes the object's Filter.WidthSpec property during the upgrade without replacement.

If you want to keep the property value, set the DD VariableClassTemplate object's Filter.FilterCondition property to `ALL_TRUE` before upgrading the Data Dictionary.

**Limitation** TargetLink does not upgrade DD VariableClassTemplate objects whose Filter.FilterCondition property is set to `ONE_OR_MORE` or `ALL_FALSE`.

### Cleaning

The automatic upgrade retains the functionality that was specified in the previous Data Dictionary. You can clean it manually to reduce the number of objects in the new Data Dictionary.

**Merging width-specific variable classes** If the previous Data Dictionary contained width-specific VariableClassTemplate/VariableClass objects, the new Data Dictionary still contains all these variable classes.

Because the width-specific information is now stored in DD CodeDecoration objects, you can manually reduce the number of VariableClass objects in the Data Dictionary. For example, if you used variable classes in the form of `<Name>_<Width>`, you can replace them by a single `<Name>` variable class that references a suitable code decoration set.

Two methods are possible:

- Merging code decoration sets:
  1. Copy all the DD CodeDecoration objects that were generated during the upgrade for each variable class called `<Name>_<Width>` to a single CodeDecorationSet object.
  2. Make each CodeDecoration object width-specific via its filter.
  3. Reference the resulting CodeDecorationSet object at the `<Name>` variable class.

- Using a code decoration set created for variable class templates:
  1. If the original `<Name>_<Width>` variable classes were referenced by variable class templates, the DD upgrade automatically creates a width-specific code decoration set in `/Pool/CodeDecorationSets/Templates`.
  2. You can reference this code decoration set at the resulting variable class called `<Name>`.

> **Note**
>
> Replace references from model elements to the variable classes called `<Name>_<Width>` with references to `<Name>`.

**Retarget variable class templates** After you merged the previous width-specific variable classes, you can use them again as the target of the variable class templates. You can then delete all the variable classes contained in `/Pool/VariableClasses/Templates` that were created during the upgrade.

**Simplifying user-specified scope reduction chains (SRC)** If you used a user-specified SRC to specify declaration statements or section names for variables with specific scopes, you can do the following:

1. Adjust the Filter.ScopeSpec property of the code decoration that belongs to the set referenced by the first variable class in the SRC (highest scope) as required.
2. Delete the other variable classes of the SRC.
3. If you also used the SRC to prevent static local variables, you can now use the AvoidStaticLocalScope Code Generator option instead.

**Remove obsolete variable class templates** Find DD VariableClassTemplate objects with the same value of the Filter.VariableClassSpec property and delete all but one.

### Changes in the generated production code

Changes in DD CodeDecoration objects can influence the generated production code in the following respects:

- Changed code comments (TargetLink New Features and Migration Guide 🕮)
- Sorting of variable definitions (TargetLink New Features and Migration Guide 🕮)

Refer to Code Changes Between TargetLink 4.3 and TargetLink 4.4 (TargetLink New Features and Migration Guide 🕮).

# Changes in Rounding Behavior for Exactly Representable Floating-Point Numbers (Microsoft)

## Changes in Rounding Behavior for Exactly Representable Floating-Point Numbers (Microsoft)

**Introduction**

Microsoft changed the rounding behavior of the `printf` family of functions. Starting in Windows 10 version 2004 (build 19041), these functions now print exactly representable floating point numbers as described in IEEE 754-2019.

**Impact on code generation with TargetLink**

This change made by Microsoft impacts code generation with TargetLink. Depending on the Windows version, the rounding behavior can differ between machines. Note, that this is independent of TargetLink.

| Windows 10 < 2004 (build 19041) | Windows 10 ≥ 2004 (build 19041) |
|---|---|
| Exactly representable floating point numbers ending in 5 would always round up. | Banker's rounding as described in IEEE 754-2019. |

**Mitigation**

To use the legacy floating point rounding behavior, you can link with legacy_stdio_float_rounding.obj, as described by Microsoft in the documentation of `printf`.

# Migrating from TargetLink 5.1 to 5.2

**Where to go from here**

Information in this section

# Code Generator Options

## Migration Aspects Regarding Code Generator Options

**Basics on changed defaults**

The settings of the Code Generator options are stored with the model (model-based option storage). In addition, you can store user-defined sets of Code Generator options in DD **CodegenOptionSet** objects (DD-based option storage). You can use DD **CodegenOptionSet** objects as a central source for overwriting and replacing the model-based option settings that have been used since TargetLink 4.1.

If a model-based option value is identical to the old default value, it is automatically changed to the new default value during the upgrade. If a DD-based option value is identical to the old default value, it is not changed to the new default value during the upgrade but keeps the old value.

**Option value = old default** If Code Generator options were set to default values in the earlier TargetLink version, and the new TargetLink version uses modified default values, note the following points:

- Model-based option:

  If you want to keep the old default values, you must reset them manually.
- DD-based option:

  If you want to use the new default values, you must adjust them manually.

The following table is an example describing the impact of a TargetLink upgrade (TargetLink$_{\text{Old}}$ to TargetLink$_{\text{New}}$) on three option values: 9, 11, and 13. The table illustrates two basic migration scenarios:

- Scenario #1: New default = old default

  The default value of a Code Generator option has not changed in the new TargetLink version, i.e., the default value remains 9.

  None of the option values is changed.
- Scenario #2: New default ≠ old default

  The default value of a Code Generator option changed with the new TargetLink version, i.e., the default value changed to 11.

| Option Storage | Option Value (TargetLink$_{\text{Old}}$) | Option Value (≤ TargetLink$_{\text{New}}$) | |
|---|---|---|---|
| | 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] The option value is not stored with the model because it is identical to the default.

[2] Manual reset might be necessary.

[3] Manual adjustment might be necessary.

**Option value = new default** If the Code Generator options were not set to default values in the former TargetLink version (A) but are in the new TargetLink version (B), TargetLink assumes that you intentionally specified the default value in the new TargetLink version. The same applies if the default changes again in the next TargetLink version (C).

> **Note**
>
> Upgrading TargetLink$_A$ ⇒ TargetLink$_B$ ⇒ TargetLink$_C$ and upgrading TargetLink$_A$ ⇒ TargetLink$_C$ can result in different option values. Refer to the following table.

If the default values for TargetLink versions A, B, and C are 9, 11, and 13, and an option was set to 11 in version A, an upgrade to version C changes the option value as follows:

| **Upgrade Strategy** | **Option Value**<br>**TargetLink$_A$**<br>**Default = 9** | **Option Value**<br>**TargetLink$_B$**<br>**Default = 11** | **Option Value**<br>**TargetLink$_C$**<br>**Default = 13** |
|---|---|---|---|
| A ⇒ B ⇒ C | 11 (≠ default) | 11 (= default)[1] | 13 (= default)[1] |
| A ⇒ C | 11 (≠ default) | — | 11 (≠ default) |

[1] The option value is not stored with the model because it is identical to the default.

### New Code Generator options

For more information on new Code Generator options, refer to New Code Generator Options on page 153.

# API Functions and Hook Scripts

## Changes in TargetLink and TargetLink Data Dictionary API Functions

### tl_access_logdata

**New properties**

| **Property** | **Description** |
|---|---|
| `realworldvalue` | Converts scaled values stored in SIL or PIL simulations to their real world value.<br>Default: 'on' |

For more information, refer to Various Migration Aspects on page 169.

**tl_clean**

**New behavior** Calls the `tl_clean.bat` batch file to delete files generated by TargetLink and asks the user for confirmation.

The following folders are deleted:

- `.\CodeCoverageFiles`
- `.\CodeViewFiles`
- `.\TLBuild`
- `.\TLProj`
- `.\TLSim`

The `.\doc` folder is no longer deleted.

# Other Migration Aspects

## Various Migration Aspects

**New name=value syntax in MATLAB®**

As of R2021a, MATLAB supports a new syntax for passing name-value arguments.

This feature does not work with option names that contain a point, e.g., 'output.type'. Thus, it cannot be used with the TargetLink API.

For example, `tl_set(gcbh, output.type = 'Float32')` results in an error:

```
Error: Incorrect use of '=' operator. To compare values for
equality, use '=='. To specify name value arguments, check that
name is a valid identifier with no quotes.
```

**New selection and interaction with ports in Simulink®**

As of R2021a, Simulink provides a new method to select ports and interact with them by pointing to the region of a connected port.

Using this method with the ports of a TargetLink subsystem leads to a corrupted simulation frame.

**Boolean types with integer base types**

Up to TargetLink 5.1, it was possible to create additional Boolean types by creating a DD Typedef object with the following properties: BaseType = arbitrary integer; Constraints: Min = 0.0, Max = 1.0; Scaling = arbitrary scaling with Lsb = 1.0 und Offset = 0.0. The dsdd_advanced_master.dd template used to contain the Bool8 typedef to show this functionality.

As of TargetLink 5.2, such types are not treated as Boolean types. Only types whose BaseType = Bool are considered Boolean. The Bool8 typedef was removed from the dsdd_advanced_master.dd template.

If a type is set to Bool8, this change causes problems with the blocks that require Boolean data types:

- Relational Operator
- Logical Operator
- S-R Flip-Flop
- J-K Flip-Flop
- !Q of D Latch
- !Q of D Flip-Flop

**Solution 1** You can replace the type at the affected blocks in the Property Manager

1. Open the Property Manager.
2. Enable Show Descendants.
3. Filter the Type column for Bool8.
4. Filter the Type (ModelElement) column for the affected blocks.
5. Replace the type with one whose BaseType = Bool, e.g., `/Pool/Typedefs/Bool`.

**Solution 2** You can use the TargetLink API to search for the affected blocks and replace Bool8 by Bool, for example:

```
hTLBlocks = tl_get_blocks(hSys, 'TL_RelationalOperator');
tl_set(hTLBlocks, 'output.type', 'Bool');
```

**Solution 3** You can set Basetype = Bool at Bool8 and any user-defined Boolean data type with integer base type.

If you used Bool8 to protect single outputs of the Relational Operator against the effect of the Use Global Bitfields option, i.e., prevent them from being moved to the global bitfield, you can use a different typedef whose BaseType = Bool, e.g., MyBool.

For information on related code changes, refer to Changes for Boolean types on page 184.

### Internal change in building the TargetLink simulation frame

Building the TargetLink simulation frame for SIL and PIL simulation modes was changed internally.

### Possible differences between MIL and SIL/PIL simulations when using signal injection and tunneling

The internal change in building the TargetLink simulation frame (see above) might result in differences between MIL and SIL/PIL simulations when using signal injection and tunneling. If the execution order is not specified by the user but the default execution order is determined by Simulink, the execution order in MIL mode on the one hand and SIL/PIL modes on the other hand might differ. This might result in differences between MIL and SIL/PIL simulations when you use signal injection and tunneling.

This affects the following blocks: Data Store Write (writing values for signal injection), TargetLink subsystem (performing the calculation of new values) and Data Store Read (reading results via signal tunneling).

If the priority is not specified, the results of the SIL/PIL simulation might be offsetfrom those of the MIL simulation by one sampling step.

To ensure that the execution order is identical in MIL and SIL/PIL simulation modes, it is important to specify priorities for the affected blocks outside the TargetLink subsystem. Refer to block priority in the Simulink user documentation by MathWorks.

**Related documentation**

- Basics on Injecting or Tunneling Signals During Simulation (TargetLink Preparation and Simulation Guide 🕮)

### Obsolete option of the tl_generate_swc_model API command

As of TargetLink 5.2, the `UseOneTlSubsystemForAllSwcs` option of the `tl_generate_swc_model` API command is obsolete but still supported. However, the `UseTLSubsystemsForAtomicSoftwareComponents` option should be used instead.

`UseTLSubsystemsForAtomicSoftwareComponents = 'off'` (default) corresponds to `UseOneTlSubsystemForAllSwcs = 'on'`: Atomic software components are modeled as virtual Simulink subsystems.

`UseTLSubsystemsForAtomicSoftwareComponents = 'on'` corresponds to `UseOneTlSubsystemForAllSwcs = 'off'`: Atomic software components are modeled as TargetLink subsystems.

Options defined under `/Pool/Autosar/Config/FrameModelGeneration` are not modified during DD upgrade. To avoid warnings during AUTOSAR frame model generation, delete the `UseOneTlSubsystemForAllSwcs` option and replace it with `UseTLSubsystemsForAtomicSoftwareComponents` and set the option's value accordingly.

**Related documentation**

- tl_generate_swc_model (TargetLink API Reference 🕮)

### New default behavior of tl_generate_swc_model when SoftwareComponents is not specified

The default behavior of the `tl_generate_swc_model` API command when the `SoftwareComponents` option is not specified has changed.

Up to TargetLink 5.1, an AUTOSAR frame model was generated/updated for all the atomic software components in the Data Dictionary or specified in the ARXML file.

As of TargetLink 5.2, TargetLink also supports the import of and frame model generation for composition software components.

If`tl_generate_swc_model` is called without explicitly specifying `SoftwareComponents` but compositions are defined in the Data Dictionary or ARXML file, frame model generation/update is performed for the compositions.

If you do not want to generate a frame model for the compositions but only for the atomic software components defined in the Data Dictionary or ARXML file, you have to explicitly specify them.

If no composition software components are defined in the Data Dictionary or ARXML file, the default behavior is unchanged.

**Related documentation**

- tl_generate_swc_model (TargetLink API Reference 🕮)

### Simplified representation of DD struct components in custom look-up scripts

To use a component of a DD struct variable in a custom look-up script, such as lookup1d_script.m, you previously had to recreate the entire structured variable in the script.

As of TargetLink 5.2, you can use a plain variable in the custom look-up script and map it to the struct component via the **Variable** property. TargetLink then uses the specified struct component. It is possible to define a plain variable in the Data Dictionary and use it as a template for the struct components in the script.

**Initial values** Generally speaking, if an initial value is specified at a DD variable, it is used in the generated code. Values specified differently in the script are ignored.

The initial value of a DD struct component can originate either from the Data Dictionary or the script. If it is specified in the script, it has to be identical for all blocks.

The initial value of a plain DD (template) variable can originate either from the Data Dictionary or the script. If it is specified in the script, it can be block-specific.

**Related documentation**

- Basics on Using Custom Look-Up Functions (TargetLink Preparation and Simulation Guide 🕮)

### Changed behavior of the tl_access_logdata function

The `tl_access_logdata('GetLoggedSignal', ...)` command returns simulation data for one or more TargetLink blocks (MIL simulation) or variables (SIL/PIL).

Up to TargetLink 5.1, all simulation values were converted to double. Additionally, the simulation values for scaled integer variables stored in SIL/PIL were converted to their real-world values, thus allowing for the direct comparison with the corresponding signal of the MIL simulation.

However, this behavior has some drawbacks as it is not possible to read the simulation value executed in the production code via the TLDS API, and simulation values of Boolean or single (Float32) type are cast to double as well.

As of TargetLink 5.2, the simulation values are no longer converted to double but retain their original data types.

The `realworldvalue` option allows you to switch off the conversion of scaled integer variables into their real-world values. By default, the option is set to 'on'. Thus, the default behavior does not change for scaled integer variables.

**Related documentation**

- tl_access_logdata (TargetLink API Reference 🕮)
- Changes in TargetLink and TargetLink Data Dictionary API Functions on page 168

### New Data Dictionary validation for code generation

TargetLink 5.2 introduces the new **CheckConformanceForCodeGeneration** Data Dictionary validation. It ensures that the DD objects required for code generation are completely specified and valid.

In contrast to the standard DD validation, the **CheckConformanceForCodeGeneration** validation does not check the entire Data Dictionary but only the DD objects required for code generation. It is performed each time code is generated and cannot be switched off. You can also start it via the context menu of any DD **SoftwareComponent** object.

**Related documentation**

- CheckConformanceForCodeGeneration

### Using 64-bit data types in Adaptive AUTOSAR

With TargetLink 5.2, 64-bit data types can be used in various situations, e.g., when modeling and simulating error handling in Adaptive AUTOSAR.

To support handling errors specified by Adaptive AUTOSAR, TargetLink uses the following structured data types:

- `int32_t` for the error value
- `uint64_t` for the error domain

When upgrading to TargetLink 5.2 from previous versions, you must create the 64-bit integer data types, i.e., `uint_64_t` and `int64_t`, in the Data Dictionary manually

**Related documentation**

- Introduction to Data Types in Adaptive AUTOSAR (TargetLink Adaptive AUTOSAR Modeling Guide 🕮)

### Names of leaf elements at bus ports synchronized with names of DD Typedef components

Up to TargetLink 5.1, the names of the leaf bus elements at bus ports were not synchronized with the names of the components of the referenced DD **Typedef** objects.

As of TargetLink 5.2, if the root variable at a bus port block is defined as an explicit struct and a structured DD **Typedef** is referenced, TargetLink synchronizes the names of the leaf bus elements with the names of the typedef components.

The Code Generator uses these names in the generated code.

If the reference to the DD **Typedef** at the root variable is removed, the block retains the synchronized properties, including the names of the leaf bus elements. This corresponds to the behavior with DD **Variable** objects.

However, if the names for the components are not specified carefully, this might result in unexpected code and in name ambiguity errors.

**Related documentation**

- How to Explicitly Specify Structured Data Types (TargetLink Customization and Optimization Guide 🕮)
- How to Create Explicitly Specified Structured Variables (TargetLink Customization and Optimization Guide 🕮)

**Removed block properties (flags) of look-up table blocks**

With TargetLink 5.2, the following block properties (flags) of Look-Up Table and Look-Up Table (2D) blocks were removed from the data model:

- `Interpolate`
- `Extrapolate`

This means that you no longer can use these flags directly via the API but have to derive them from the `LookupMethEnum` as follows:

-
  ```
  Interpolate = any(tl_get(<block>, ‘lookupmeth’) == [1 2])
  ```

-
  ```
  Extrapolate = tl_get(<block>, ‘lookupmeth’) == 1)
  ```

If you used the API to access any of these properties, adapt your scripts accordingly.

# Code Changes Between TargetLink 5.1 and TargetLink 5.2

**Where to go from here**

Information in this section

## Adaptive AUTOSAR

**No detailed documentation of changes**

**Note**

Due to the high volatility of the Adaptive AUTOSAR standard and the large amount of changes, no Adaptive-AUTOSAR-related code changes are documented for TargetLink 5.2.
If you need more information, contact dSPACE Support.

## Array-of-Struct-Related

**Removal of intermediate variables**

TargetLink can now remove more intermediate variables in the context of struct accesses and array-of-struct accesses.

| TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|
| `Sa2_Selector = A2[4];`<br><br>`Aux_ = Sa2_Selector.m11;`<br>`Aux_ -= Sa2_Selector.m13;` | <br><br>`Aux_ = A2[4].m11;`<br>`Aux_ -= A2[4].m13;` |

**Reason** Code efficiency

## Classic AUTOSAR

**Memory mapping of runnables**

DD Runnable objects that do not reference a DD FunctionClass object now automatically use the preconfigured DD CodeDecorationSet `/Pool/CodeDecorationSets/AUTOSAR/MemorySections`. If you make changes to this code decoration set, these changes affect every DD Runnable object that does not reference a DD FunctionClass object.

This can result in a different sorting order of runnable code.

**Reason** Increased consistency

**Generalization of if condition for efficient write pattern of NvData communication**

With TargetLink 5.2 the if condition in the code pattern for efficient rewrites in NvData communication was generalized. It no longer depends on the modeling style.

| TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|
| With Assignment block:<br>`if (<ReadPointer> != <input>) ....`<br>Without Assignment block:<br>`if (<input> != <ReadPointer>) ...` | Now always:<br>`if (<ReadPointer> != <input>) ....` |

**Reason** Increased consistency.

**Efficient read/write access via data stores**

TargetLink now offers new code patterns for efficiently reading and writing via data stores, refer to Efficient read/write access via data stores on page 178. This results in more efficient code. The following table shows an example of sender-receiver communication:

**TargetLink ≤ 5.1**

```
/* Data store write: Subsystem/Subsystem/DSW */
p_new_DataElement_a = Rte_IWriteRef_new_Runnable_new_SenderReceiverPort_new_DataElement();

/* Data store read: Subsystem/Subsystem/DSR */
p_new_DataElement = Rte_IRead_new_Runnable_new_SenderReceiverPort_new_DataElement();

/* Assignment: Subsystem/Subsystem/Assignment: output initialization */
 Sa2_Assignment[0] = p_new_DataElement[0];
 Sa2_Assignment[1] = p_new_DataElement[1];
 Sa2_Assignment[2] = p_new_DataElement[2];

/* Assignment: Subsystem/Subsystem/Assignment: output calculation */
 Sa2_Assignment[1] = 2;

/* Data store write: Subsystem/Subsystem/DSW */
 p_new_DataElement_a[0] = Sa2_Assignment[0];
 p_new_DataElement_a[1] = Sa2_Assignment[1];
 p_new_DataElement_a[2] = Sa2_Assignment[2];
```

**TargetLink 5.2**

```
/* Data store write: Subsystem/Subsystem/DSW */
p_new_DataElement = Rte_IWriteRef_new_Runnable_new_SenderReceiverPort_new_DataElement();

/* Data store write: Subsystem/Subsystem/DSW
      # combined # Assignment: Subsystem/Subsystem/Assignment: output calculation */
p_new_DataElement[1] = 2;
```

**Reason**

- Code efficiency.
- Code stability.

### Dimension reduction of scalar auxiliary variables

Scalar auxiliary variables created for array variables that have an AUTOSAR array type now have the scalar type that is referenced at the AUTOSAR array type.

**TargetLink ≤ 5.1**

```
sint16 Aux_S16;

   for (Aux_S32 = 0; Aux_S32 < 7; Aux_S32++)
   {
      /* Abs: TL_Root/SS3/SS3_Abs */
      if (p_NvData_V_7_S16[Aux_S32] >= 0) {
         /* Vector 'SS3_Abs' replaced by 'Aux_S16' */
         Aux_S16 = p_NvData_V_7_S16[Aux_S32];
      }
```

**TargetLink 5.2**

```
MyType Aux_S16;

   for (Aux_S16 = 0; Aux_S16 < 7; Aux_S16++)
   {
      /* Abs: TL_Root/SS3/SS3_Abs */
      if (p_NvData_V_7_S16[Aux_S16] >= 0) {
         /* Vector 'SS3_Abs' replaced by 'Aux_S16' */
         Aux_S16 = p_NvData_V_7_S16[Aux_S16];
      }
```

**Reason** Matches user expectations.

### Protection of Classic AUTOSAR operations

Classic AUTOSAR operations can potentially be called from different code generation units (CGUs). Accordingly, TargetLink does not remove outputs and related calculations, even if they are not used in the actual CGU.

This mainly affects simulation code and is related to the following code change: Removal of unused assignments for pointer initialization on page 185.

**Reason** Code efficiency

### Change in the initializiation of data prototypes without defined initial value

The RTE-internal initialization changed for data prototypes for which no initial value is specified in the AUTOSAR data or in the model. In the RTE, such data prototypes are now initialized by `0` (internal value) instead of `0` (physical value) because `0` as a physical value might not be representable in the selected scaling:

| TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|
| `#define Rte_InitValue_ReceiverPort_data_scalar 4 /* 0 */` | `#define Rte_InitValue_ReceiverPort_data_scalar 0 /* -2 */` |

This does not have any impact on simulation behavior.

**Reason** To avoid errors in cases where `0` as a physical value is not representable in the selected scaling.

## Conditional Control Flows and Branches

### Moving into conditional control flows improved

TargetLink can now move more statements into conditional control flows. The following statements can now be moved:

- Several statements that write to the same variable.
- Statements that write to a part of the range used by the variable in the conditional control flow.
- Statements that contain indeterminate indices.
- Loops containing the write statement.
- Write statements belonging to read accesses inside loops inside the conditionally executed control flow branch.

Examples:

<table>
<tr><th>TargetLink ≤ 5.1</th><th>TargetLink 5.2</th></tr>
<tr><td><pre><code>for (i = 0; i < 10; i++) {
   v[i] = ...
   w[i] = ...
}
if (cond) {
   f(v, w);
}</code></pre></td><td><pre><code>if (cond) {
   for (i = 0; i < 10; i++) {
      v[i] = ...
      w[i] = ...
   }
   f(v, w);
}</code></pre></td></tr>
<tr><td><pre><code>h(x); /* Defines all of x */
if (cond) {
   while (...) {
      g(x[j]);
   }
}</code></pre></td><td><pre><code>if (cond) {
   h(x); /* Defines all of x */
   while (...) {
      g(x[j]);
   }
}</code></pre></td></tr>
</table>

**Reason** Code efficiency.

### Conditional control flows and preprocessor directives

TargeLink can now move statements into blocks that are controlled via preprocessor directives for conditional compilation (`#if`, `#ifdef`, `#ifndef`). Moreover, if these directives have a closing `#else` directive, TargetLink now treats these code sections in the same way as control flow statements with the `else` and `default` branch when analyzing whether they can be moved into conditionally executed branches or conditionally compiled code sections.

**Reason** Code efficiency.

## Initial Code

### Increased consistency of struct-related code

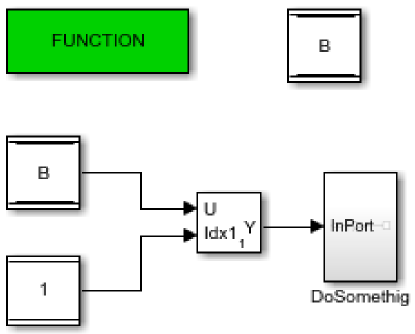
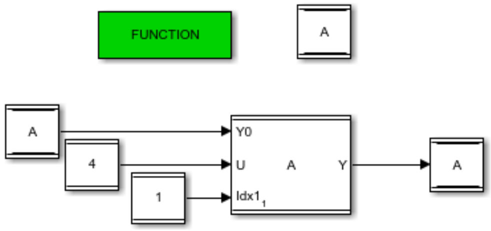
The consistency of initial code was improved regarding struct-related code. As a consequence, the number of struct assignments in the optimized code might change in the following areas:

- Loop Merging on page 179.
- Conditional Control Flows and Branches on page 177.
- Variable Sharing on page 180.

**Reason** Increased consistency.

### Efficient read/write access via data stores

TargetLink now offers new code patterns for efficiently reading and writing via data stores. The following illustration shows the modeling style for read/write access:

| Modeling Efficient Read Access | Modeling Efficient Write Access |
|---|---|
| <br> FUNCTION<br>B<br>B<br>1<br>U<br>Idx1<br>Y<br>InPort<br>DoSomethig | <br> FUNCTION<br>A<br>A<br>4<br>1<br>Y0<br>U<br>A<br>Y<br>Idx1<br>A |

| TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|
| **Read Access** | |
| /* Data store read: Subsystem/Subsystem/DSR */<br>Sa2_DSR[0] = myArray[0];<br>Sa2_DSR[1] = myArray[1];<br>Sa2_DSR[2] = myArray[2];<br><br>/* Selector: Subsystem/Subsystem/Selector */<br>Sa2_Selector = Sa2_DSR[Sa2_Constant1 - 1]; | /* Selector: Subsystem/Subsystem/Selector<br>   # combined # Data store read: Subsystem/Subsystem/DSR<br>*/<br>Sa2_Selector = myArray[Sa2_Constant1 - 1]; |
| **Write Access** | |
| /* Data store read: Subsystem/Subsystem/DSR */<br>Sa2_DSR[0] = myArray[0];<br>Sa2_DSR[1] = myArray[1];<br>Sa2_DSR[2] = myArray[2];<br><br>/* Assignment: Subsystem/Subsystem/Assignment: output<br>calculation */<br>DD_Sa2_Assignment[Sa2_Constant1 - 1] = 4;<br><br>/* Data store write: Subsystem/Subsystem/DSW */<br>myArray[Sa2_Constant1 - 1] =<br>DD_Sa2_Assignment[Sa2_Constant1 - 1]; | /* Data store write: Subsystem/Subsystem/DSW<br>    # combined # Assignment:<br>Subsystem/Subsystem/Assignment: output calculation */<br>myArray[Sa2_Constant1 - 1] = 4; |

**Reason**

- Code efficiency.
- Code stability.

## Loop Merging

**Improved loop merging**

TargetLink can now merge more loops. This pertains to the following contexts:

- Loops containing function calls.
- Loops that are generated for iterated subsystems with other loops, if data flow allows it.
- Loops containing struct accesses.

**Reason** Code efficiency.

# Stateflow-Related

### Only relevant state machines in file header

The information on state machines in file headers changed. With TargetLink 5.2, the numbering of required state machines no longer contains gaps:

<table>
<tr><th>TargetLink ≤ 5.1</th><th>TargetLink 5.2</th></tr>
<tr><td><pre>
*** CCC0           LowLevelCCFunctions_tl
*** CCC3           TopLevelCCFunctions_tl
</pre></td><td><pre>
*** CCC0           LowLevelCCFunctions_tl
*** CCC1           TopLevelCCFunctions_tl
</pre></td></tr>
</table>

**Reason** Matches user expectations.

### Saturation of large negative numerical values

Large negative numerical values are no longer saturated on `INT32MIN`, e.g., the following assignment in a Stateflow® chart:

```
Int32Var = int32(-2147483649) // -21474836489 == (INT32MIN - 1)
```

<table>
<tr><th>TargetLink ≤ 5.1</th><th>TargetLink 5.2</th></tr>
<tr><td><pre>
Int32Var = (int32)-2147483648;
</pre></td><td><pre>
Int32Var = (int32)-2147483649;
</pre></td></tr>
</table>

**Reason**

- Easier debugging.
- Increased consistency.

# Variable Sharing

### Downsampling counter and FirstRun variables

TargetLink can now combine the following variables used for sequential control:

- `FirstRun` variables generated for the initialization of Assignment block outputs in iterated subsystems.
- Downsampling counter (for equal step sizes).

You can control this behavior via the EnableSharingOfSequentialControlVariables Code Generator option.

The following table shows an example of a `FirstRun` variable:

<table>
<tr><th>TargetLink ≤ 5.1</th><th>TargetLink 5.2</th></tr>
<tr><td><pre>
Sa3_Ass2_FirstRun = 1;
   Sa3_Ass3_FirstRun = 1;
   Sa3_Ass1_FirstRun = 1;
   for (Sa2_it = 0; Sa2_it <= 4; Sa2_it++)
   {
      if (Cond) {
         if (Sa3_Ass2_FirstRun) {
            for (Aux_S32 = 0; Aux_S32 < 5; Aux_S32++)
            {
               Ass2[Aux_S32] = ...;
            }
            Sa3_Ass2_FirstRun = 0;
         }
</pre></td><td><pre>
Sa3_Ass2_FirstRun = 1;
   for (Sa2_it = 0; Sa2_it <= 4; Sa2_it++)
   {
      if (Cond) {
         if (Sa3_Ass2_FirstRun) {
            for (Aux_S32 = 0; Aux_S32 < 5; Aux_S32++)
            {
               Ass2[Aux_S32] = ...;
            }
         }
...
         if (Sa3_Ass2_FirstRun) {
            for (Aux_S32 = 0; Aux_S32 < 5; Aux_S32++)
</pre></td></tr>
</table>

**TargetLink ≤ 5.1**

```
...
   if (Sa3_Ass3_FirstRun) {
    for (Aux_S32 = 0; Aux_S32 < 5; Aux_S32++)
    {
     Ass3[Aux_S32] = X_UD[Aux_S32];
    }
    Sa3_Ass3_FirstRun = 0;
   }
...
   if (Sa3_Ass1_FirstRun) {
    for (Aux_S32 = 0; Aux_S32 < 5; Aux_S32++)
    {
     Ass1[Aux_S32] = C[Aux_S32];
    }
    Sa3_Ass1_FirstRun = 0;
   }
```

**TargetLink 5.2**

```
    {
     Ass3[Aux_S32] = X_UD[Aux_S32];
    }
   }
...
   if (Sa3_Ass2_FirstRun) {
    for (Aux_S32 = 0; Aux_S32 < 5; Aux_S32++)
    {
     Ass1[Aux_S32] = C[Aux_S32];
    }
    Sa3_Ass2_FirstRun = 0;
   }
```

Further optimized, the code looks like this:

**TargetLink 5.2 After Further Optimizations**

```
/* Variable 'Sa3_Ass2_FirstRun' replaced by 'Aux_B' */
 Aux_B = 1;
 for (Sa2_it = 0; Sa2_it <= 4; Sa2_it++)
 {
  if (Cond) {
   if (Aux_B) {
    for (Aux_S32 = 0; Aux_S32 < 5; Aux_S32++)
    {
     Ass2[Aux_S32] = ...;
     Ass3[Aux_S32] = X_UD[Aux_S32];
     Ass1[Aux_S32] = C[Aux_S32];
    }

    Aux_B = 0;
   }
...
```

The following table shows an example of downsampling code:

**TargetLink ≤ 5.1**

```
static UInt8 Ctr_Sa2 = 1;
 static UInt8 Ctr_Sa5 = 1;
 static UInt8 Ctr_Sa3 = 1;

 if ((--Ctr_Sa2) == 0) {
  Input_T3s();
  Ctr_Sa2 = 6;
 }
 if ((--Ctr_Sa5) == 0) {
  controller_T5s();
  Ctr_Sa5 = 10;
 }
 if ((--Ctr_Sa3) == 0) {
  Integrator_T3s();
  Ctr_Sa3 = 6;
 }
```

**TargetLink 5.2**

```
static UInt8 Ctr_Sa2 = 1;
 static UInt8 Ctr_Sa5 = 1;

 if ((--Ctr_Sa2) == 0) {
  Input_T3s();
 }
 if ((--Ctr_Sa5) == 0) {
  controller_T5s();
  Ctr_Sa5 = 10;
 }
 if (Ctr_Sa2 == 0) {
  Integrator_T3s();
  Ctr_Sa2 = 6;
 }
```

Further optimized, the code looks like this:

**TargetLink 5.2 After Further Optimizations**

```
static UInt8 Ctr_Sa2 = 1;
   static UInt8 Ctr_Sa5 = 1;
   if ((--Ctr_Sa2) == 0) {
      Input_T3s();

      Integrator_T3s();
      Ctr_Sa2 = 6;
   }
   if ((--Ctr_Sa5) == 0) {
      controller_T5s();
      Ctr_Sa5 = 10;
   }
```

**Reason** Code efficiency.

## Other

### Inheritance of struct types in scaling-invariant systems

If a block that is specified to inherit its properties is directly connected to a preceding OutPort block of a scaling-invariant subsystem whose block variable has a struct type, TargetLink now inherits the struct type from the OutPort block to the following block.

**TargetLink ≤ 5.1**

```
struct BS_OP_Sa2_Out1 Sa2_Out1;
static Int16 X_Sa1_Unit_Delay = 0;
static Int16 X_Sa1_Unit_Delay_a = 0;
...
Sa2_ScalingInvariant(&Sa2_Out1);
...
X_Sa1_Unit_Delay = Sa2_Out1.a;
X_Sa1_Unit_Delay_a = Sa2_Out1.b;
```

**TargetLink 5.2**

```
struct BS_OP_Sa2_Out1 Sa2_Out1;
static struct BS_OP_Sa2_Out1 X_Sa1_Unit_Delay = {
   0,
   0
};
...
Sa2_ScalingInvariant(&Sa2_Out1);
...
X_Sa1_Unit_Delay = Sa2_Out1;
```

**Reason**

- Bug fix.
- Increased consistency.

### Code comment for Fcn block with sgn function

TargetLink now correctly generates the code comment of the Fcn block if it specified as the `sgn` function in front of the first statement:

| TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|
| `TlAux_Sa1_Fcn = Sa1_InPort;`<br>`if (TlAux_Sa1_Fcn < 0) {`<br>`    TlAux_Sa1_Fcn_a = -1;`<br>`}`<br>`else {`<br>`   TlAux_Sa1_Fcn_a = TlAux_Sa1_Fcn > 0;`<br>`}`<br><br>`/* TL_Root/Fcn */`<br>`Sa1_Fcn = (Int16) TlAux_Sa1_Fcn_a;` | `/* Fcn: TL_Root/Fcn */`<br>`TlAux_Sa1_Fcn = Sa1_InPort;`<br>`if (TlAux_Sa1_Fcn < 0) {`<br>`   TlAux_Sa1_Fcn_a = -1;`<br>`}`<br>`else {`<br>`   TlAux_Sa1_Fcn_a = (Int16) (TlAux_Sa1_Fcn > 0);`<br>`}`<br><br><br>`Sa1_Fcn = (Int16) TlAux_Sa1_Fcn_a;` |

**Reason**

- Bug fix.
- Increased consistency.

### Block comment of Fcn block

The block comment generated for the Fcn block now contains the block type:

| TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|
| `/* TL_Root/Fcn */` | `/* Fcn: TL_Root/Fcn */` |

**Reason** Increased consistency.

### Improved comments for struct replacements via copy propagation

Comments used for struct replacements via copy propagation now show the complete structure hierarchy:

| TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|
| `/* Parent Struct 'n_2' replaced by 'n_2' */` | `/* Parent Struct 'Sc1_Data_Store_Read1.n_2[][]' replaced by 'MyAoSType3_3[0].m_2.n_2[][]' */` |

If an index access contains a fixed number, it is shown in the comment. Otherwise, the brackets are empty.

Additionally, a problem with lost comments was fixed.

**Reason** Bug fix.

### Code of the Fcn block with sgn function

The code pattern generated for the Fcn block with the `sgn` function changed:

- If an auxiliary variable is generated for the input parameter, it now has the data type and scaling of the parameter. In particular, if the input parameter of the function is the block variable that is generated for the preceding block, the auxiliary variable now has the data type and scaling of the block variable of the preceding block.
- The assignement in the `else` branch now contains an additional cast.

**TargetLink ≤ 5.1**

```
Int16 Sa1_InPort; /* LSB: 2^-1 OFF: 0 MIN/MAX: -16384 .. 16383.5 */
Int32 Sa1_Fcn; /* LSB: 2^-2 OFF: 0 MIN/MAX: -536870912 .. 536870911.75 */
Int32 TlAux_Sa1_Fcn_a; /* LSB: 2^-2 OFF: 0 MIN/MAX: -536870912 .. 536870911.75 */
Int16 TlAux_Sa1_Fcn; /* LSB: 2^-2 OFF: 0 MIN/MAX: -536870912 .. 536870911.75 */

TlAux_Sa1_Fcn = Sa1_InPort;
if (TlAux_Sa1_Fcn < 0) {
   TlAux_Sa1_Fcn_a = -1;
}
else {
   TlAux_Sa1_Fcn_a = TlAux_Sa1_Fcn > 0;
}
Sa1_Fcn = (Int32) TlAux_Sa1_Fcn_a;
```

**TargetLink 5.2**

```
Int16 Sa1_InPort; /* LSB: 2^-1 OFF: 0 MIN/MAX: -16384 .. 16383.5 */
Int32 Sa1_Fcn; /* LSB: 2^-2 OFF: 0 MIN/MAX: -536870912 .. 536870911.75 */
Int32 TlAux_Sa1_Fcn_a; /* LSB: 2^-2 OFF: 0 MIN/MAX: -536870912 .. 536870911.75 */
Int16 TlAux_Sa1_Fcn; /* LSB: 2^-1 OFF: 0 MIN/MAX: -16384 .. 16383.5 */

TlAux_Sa1_Fcn = Sa1_InPort;
if (TlAux_Sa1_Fcn < 0) {
   TlAux_Sa1_Fcn_a = -1;
}
else {
   TlAux_Sa1_Fcn_a = (Int32) (TlAux_Sa1_Fcn > 0);
}
Sa1_Fcn = (Int32) TlAux_Sa1_Fcn_a;
```

**Reason** Bug fix.

## Changes for Boolean types

With TargetLink 5.2, TargetLink treats only those types as Boolean whose BaseType is set to `Bool`. This can result in code changes. The following table shows examples for `UInt8` as base type of the `Bool8` DD Typedef object:

| TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|
| `if (((UInt8) Ca2_sfBool8Var) != ((UInt8) Ca2_tlBoolVar)) {` | `if (Ca2_sfBool8Var != ((UInt8) Ca2_tlBoolVar)) {` |
| `if (((UInt8) Ca2_sfBool8Var) != Ca2_tlUInt8Var) {` | `if (Ca2_sfBool8Var != Ca2_tlUInt8Var) {` |
| `Bool8Var = (UInt8) (Sa1_I16In != 0);` | `Bool8Var = (UInt8) Sa1_I16In;` |
| `MyStructVar.MyBitfield = Sa1_Bool8In;` | `MyStructVar.MyBitfield = Sa1_Bool8In & 1;` |
| `Bool8Var = (UInt8) (ScaledVarNoOffset != 0);` | `Bool8Var = (UInt8) (Int8) (ScaledVarNoOffset >> 5);` |

The following table shows a potential code change if `Bool8` is used as the `Control` input of a Switch block:

| TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|
| `if(Bool8Var != 0);` | `if(Bool8Var >= 1);` |

**Reason** Increased consistency.

**Migration issue** Refer to Boolean types with integer base types on page 169.

### Left-shift operations of Extract Bits block

TargetLink now treats left-shift operations in the code generated for the Extract Bits block the same way than other left shift operations. This can result in code differences regarding the number of casts.

| TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|
| `Sa1_ExtractBits_ = ((Int32) (10 << 16)) >> 16;` | `Sa1_ExtractBits = ((Int32) (((UInt32) 10) << 16)) >> 16;` |

**Reason** Increased consistency.

### Removal of unused assignments for pointer initialization

TargetLink now removes more pointer initialization assignments for pointers that are not used. This happens more often if the right-hand side of the assignment contains casts.

**Reason** Code efficiency.

### Delay block

The code of the Delay block no longer contains an auxiliary pointer if the delayed signal is one element.

**TargetLink ≤ 5.1**

```
Int16 * pAuxPtr;

   pAuxPtr = &(X_DelayScalar1x1[X_Sa3_DelayScalar1x1_Index]);

   Sa3_DelayScalar1x1 = pAuxPtr[0];

   Sa3_Out1 = Sa3_DelayScalar1x1;

   pAuxPtr[0] = Sa3_In1;
```

**TargetLink 5.2**

```
Sa3_DelayScalar1x1 =  X_DelayScalar1x1[X_Sa3_DelayScalar1x1_Index];

   Sa3_Out1 = Sa3_DelayScalar1x1;

   X_DelayScalar1x1[X_Sa3_DelayScalar1x1_Index] = Sa3_In1;

   X_Sa3_DelayScalar1x1_Index = ((UInt8) (X_Sa3_DelayScalar1x1_Index + 1)) % 3;
```

**Reason** Resolves differences in MIL/SIL/PIL simulation modes.

### Empty line before function inline

Inlined code that begins with a comment now always has an empty line preceding the comment:

| **TargetLink ≤ 5.1** | **TargetLink 5.2** |
|---|---|
| /* update(s) for inport Subsystem/Atomic Subsystem/Atomic Subsystem/In1 */<br>Sa5_In1 = Sa2_Gain;<br><br>/* update(s) for inport Subsystem/Atomic Subsystem/Atomic Subsystem/In2 */<br>Sa5_In2 = Sa2_Gain1;<br>/* Gain: Subsystem/Atomic Subsystem/Atomic Subsystem/Gain */<br>Sa5_Gain = Sa5_In2;<br><br>/* TargetLink outport: Subsystem/Atomic Subsystem/Atomic Subsystem/Out2 */<br>Sa5_Out2 = Sa5_Gain; | /* update(s) for inport Subsystem/Atomic Subsystem/Atomic Subsystem/In1 */<br>Sa5_In1 = Sa2_Gain;<br><br>/* update(s) for inport Subsystem/Atomic Subsystem/Atomic Subsystem/In2 */<br>Sa5_In2 = Sa2_Gain1;<br><br>/* Gain: Subsystem/Atomic Subsystem/Atomic Subsystem/Gain */<br>Sa5_Gain = Sa5_In2;<br><br>/* TargetLink outport: Subsystem/Atomic Subsystem/Atomic Subsystem/Out2 */<br>Sa5_Out2 = Sa5_Gain; |

**Reason**

- Bug fix.
- Increased consistency.

### U suffixes on constant integers

If a value of a constant in the code is less than or equal to `Int32Max`, less U suffixes after constant integers are generated.

| **TargetLink ≤ 5.1** | **TargetLink 5.2** |
|---|---|
| Sd1_OutPort_26 = 2101273897U - ((UInt32) (((UInt32) Sd1_InPort_26) * 1048576)); | Sd1_OutPort_26 = 2101273897 - ((UInt32) (((UInt32) Sd1_InPort_26) * 1048576)); |

**Reason**

- MISRA C compliance.
- Increased consistency.

### Names of variables generated for bus port blocks

The names of variables or function parameters that are generated from bus port blocks can change if all of the following conditions are fulfilled:

- The block specifies at least two variables.
  - The variable class of one of these variables is set to `Default`.
  - The variable class of the other variable is not set to `Default`.
  - The variable names are the same but contain the name macros `$R` or `$S`.
- In the model, at least one other variable is specified whose name is the same and that contains the name macros `$R` or `$S`.

In these cases, TargetLink evaluates the name macros in a different order and it might seem that the function parameters are sorted differently:

**TargetLink ≤ 5.1**

```
void Sa2_Inner(Int16 * Plain_a, Int16 * Plain_b, struct BS_IP_Sa2_StructIns * Struct_a, Int16 *
  Plain, Int16 * Plain_c, struct BS_IP_Sa2_StructIns * Struct)
{
   /* BusOutport: TL_Root/Inner/StructOuts */
   *Struct = *Struct_a;
   Struct_c = Struct_b;

   /* BusOutport: TL_Root/Inner/PlainOuts */
   *Plain = *Plain_a;
   *Plain_c = *Plain_b;
}
```

**TargetLink 5.2**

```
void Sa2_Inner(Int16 * Plain, Int16 * Plain_a, struct BS_IP_Sa2_StructIns * Struct, Int16 * Plain_b,
  Int16 * Plain_c, struct BS_IP_Sa2_StructIns * Struct_a)
{
   /* BusOutport: TL_Root/Inner/StructOuts */
   *Struct_a = *Struct;
   Struct_c = Struct_b;

   /* BusOutport: TL_Root/Inner/PlainOuts */
   *Plain_b = *Plain;
   *Plain_c = *Plain_a;
}
```

**Reason** Increased consistency.

### Inheritance of struct data types whose struct components have type qualifiers

Inheritance has changed for struct data types whose struct omponents have type qualifiers if all of the following conditions are fulfilled:

- A block whose block variable is a struct is followed by a block that is specified to inherit its block properties.
- At least one struct component has a type qualifier such as `const` or `volatile`, or a type prefix.

It depends on the way that the type qualifier is specified whether the successor block inherits the struct type:

| Type Qualifier Specified At | TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|---|
| The DD TypePrefix property of the DD VariableClass object that is referenced at the struct component. | No type qualifier at struct component | |
| | No inheritance of struct type | Inheritance of struct type |
| The DD TypePrefix property of the struct component of the DD Typedef object | Type qualifier at struct component | |
| | Inheritance of struct type | No inheritance of struct type |

**Reason** Increased consistency.

### Type prefixes and scope reduction chain

A bug has been fixed concerning type prefixes and scope reduction of variables whose DD ScopeReducedClass property is specified. For scope-reduced variables for which a scope reduction chain is specified via the ScopeReducedClass variable class property, TargetLink does the following:

| TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|
| TargetLink uses the TypePrefix from the original variable class instead of the TypePrefix from the variable class of the scope reduction chain corresponding to the resulting scope. | TargetLink uses the TypePrefix from the variable class of the scope reduction chain corresponding to the resulting scope.<br>If no DD ScopeReducedClass property is specified for the variable class of the variable, the TypePrefix of the variable remains the same during scope reduction. |

**Reason** Bug fix.

**Migration issue** Variables that were scope-reduced via variable class templates can lose their type qualifiers if the scope reduction chain is incompletely specified. Refer to the following application note: Using TargetLink Variable Class Templates.

**Removal of superfluous struct (component) definitions**

TargetLink now more often removes superfluous struct definitions or struct component definitions if they are covered by a subsequent struct definition. This also applies if the AllowStructAssignments Code Generator option is enabled.

**Reason** Code efficiency.

**D_Previous variables of D Flip-Flop blocks**

The code generated for the D Flip-Flop block has changed if all of the following conditions are fulfilled:

- The signal connected to the D port is non-scalar.
- The non-scalar signal is created via a Mux, Concatenate, or Selector block.

| TargetLink ≤ 5.1 | TargetLink 5.2 |
|---|---|
| One `D_Previous` variable that has the same width than the signal at the D port. | Several `D_Previous` variables whose names differ in their postfixes (`_a`, `_b`, etc.). |

**Reason** Increased consistency.

# Discontinuations as of TargetLink 5.2

**Where to go from here**

Information in this section

## Discontinued TargetLink Features

**Reset command in the Data Dictionary Manager**

With TargetLink 5.2, the Reset command in the Data Dictionary Manager has been removed. For more information, refer to Obsolete API Functions on page 189.

**Automatic interpretation of Boolean**

The automatic interpretation of certain integer data types as Boolean by TargetLink is discontinued as of TargetLink 5.2. For more information, refer to Various Migration Aspects on page 169.

**Unit Delay Reset Enabled**

The TargetLink support of the Unit Delay Reset Enabled block is discontinued with TargetLink 5.2.

To avoid unresolved links when migrating models containing Unit Delay Reset Enabled blocks to TargetLink 5.2, remodel the functionality using supported blocks.

**Model change detection via checksum**

The TargetLink support of model change detection via checksum is discontinued with TargetLink 5.2.

**Sample blocks**

The TargetLink support of Sample blocks is discontinued with TargetLink 5.2.

The Bit Operation blocks formerly provided in the tlsamples model are now available as part of the TargetLink Add-On Package that you can download from the dSPACE website: TargetLink Add-On Package.

## Obsolete API Functions

**Obsolete Data Dictionary MATLAB API functions**

| Function | Status | Replacement |
|---|---|---|
| `ResetToDefault` | Error[1] | - |

[1] The function was removed from TargetLink.

# Obsolete Limitations

**Multiple versions of the Data Dictionary Manager installed**

If multiple, different versions of TargetLink are installed on the same system, opening a `.dd` file via context menu or double-click on the file can lead to the stand-alone Data Dictionary Manager of the older TargetLink version being opened.

In this case, open the `.dd` file by selecting **`Open with`** on the context menu and select the correct Data Dictionary Manager Stand-Alone application in the dialog that opens. Optionally, select the checkbox Always use this app to open .dd files.

**Outputs of TargetLink subsystems**

For TargetLink subsystems, the outputs at model initialization time always emit the default values that are related to the data type, e.g., false for Boolean and 0 for numeric data types.

**Models with event buses not initializable**

Models containing a bus that consists only of event signals are not initializable and simulation in SIL mode is not possible if both of the following conditions are fulfilled:

- The bus crosses the boundary of the TargetLink subsystem on the input side.
- Inside the TargetLink subsystem, single bus signals are selected via a BusSelector block.

To avoid this problem, use the following workaround:

- Outside the TargetLink subsystem:

  Convert the bus signal to a multiplexed signal by using Bus Selector and Mux blocks.
- Inside the TargetLink subsystem:

  Select event signals by using Selector or Demux blocks (no Bus Selector block).

# Changes in Future TargetLink Versions

**Where to go from here**

Information in this section

## Features to Be Discontinued

**RTOS/OSEK code generation mode**

Support for the TargetLink RTOS/OSEK code generation modes will be discontinued in a future TargetLink version.

**Target Optimizations Module (TOM)**

Support for the TargetLink Target Optimizations Module (TOM) will be discontinued in a future TargetLink version.

**Clean code and Do not log anything**

Variables selected for logging cannot be fully optimized. When generating code with the Global logging option `Do not log anything` or `Log according to block data`, TargetLink does not fully optimize the code to facilitate testing. This means the code differs only with regard to the log macros. This contrasts the Clean code checkbox on the Code Generation page of the TargetLink Main Dialog block, which always activates full code optimization.

The special `Do not log anything` behavior will be removed in a future TargetLink version.

**Simulink classic initialization mode**

Support for the Simulink classic initialization mode will be discontinued in a future TargetLink version.

**Dynamic components**

Support for specifying dynamic components for DD Variable objects will be discontinued in a future TargetLink version.

**Discontinuation of SWC container exchange**

As of dSPACE Release 2022-B, exchanging SWC containers between TargetLink and SystemDesk will no longer be supported. However, you can still exchange software components using ARXML files. With dSPACE AUTOSAR Compare, you can compare and merge these files.

The discontinued software parts include:

- In SystemDesk, the **Container File Explorer** and the commands for preparing and exchanging SWC containers.
- In TargetLink, the Data Dictionary commands for exchanging SWC containers.
- The Container Manager, which is a stand-alone tool for managing the contents of SWC containers. It will no longer be available from TargetLink or SystemDesk.

## 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_compare_fcn_signature` | TargetLink 5.0 | - |

## Deprecated Code Generator Options

**Deprecated Code Generator options**

The following Code Generator options are deprecated and will be removed in future TargetLink versions:

- AllowStructAssignments (TargetLink Model Element Reference 📖)

# VEOS

**Where to go from here**

Information in this section

## New Features of VEOS 5.3

**Classic V-ECUs: Support of pre-startup and post-shutdown functions**

In the Sab module of a V-ECU, SystemDesk lets you specify the following:

- Pre-startup functions, i.e., functions to be executed immediately after the Sab_Init() function is called (SabPreStartupFunctions).
- Post-shutdown functions, i.e., functions to be executed immediately before the Sab_DeInit() function is called (SabPostShutdownFunctions).

VEOS supports the execution of pre-startup and post-shutdown functions.

**Adaptive V-ECUs: Support of R20-11 of the AUTOSAR Adaptive Platform**

VEOS 5.3 supports R20-11 of the AUTOSAR Adaptive Platform. Refer to Compatibility of VEOS 5.3 on page 194.

**Simultaneous VPU access from multiple XCP masters**

As of Version 5.3, VEOS supports simultaneous access to a V-ECU or controller VPU from multiple external tools. You can use the XCP on Ethernet port of the V-ECU or controller VPU for this purpose. To be accessed from multiple XCP masters, the V-ECU or controller VPU must be built with VEOS 5.3 or later.

For more information, refer to VPU (V-ECU / Environment VPU / Controller VPU / Bus VPU) (VEOS Manual 🕮).

**Building and simulating V-ECUs targeting 32-bit Linux runtimes**

VEOS 5.3 now lets you build and simulate V-ECUs that target 32-bit Linux runtimes.

For more information, refer to Simulation Target Manager (VEOS Manual 🕮).

**Command line interface extension: VEOS Model Console**

Both the Windows and Linux versions of VEOS 5.3 now provide the *VEOS Model Console*.

The new *VEOS Model Console* lets you modify offline simulation applications via the command line interface:

- Import an OSA file or selected VPUs to another OSA file.
- Add and remove elements, such as communication clusters, to/from an OSA file.
- Connect VPU signals and communication controllers in an OSA file.

You must provide the modifications in JSON format.

For more information, refer to VEOS Model Console Command Reference (VEOS Manual 🕮).

> **Tip**
>
> The demos for VEOS contain Python demo scripts that show you how to use VEOS Model Console.
> Refer to Demos for VEOS (VEOS Manual 🕮).

**Related topics**

References

VEOS Model Console Command Reference (VEOS Manual 🕮)

## Compatibility of VEOS 5.3

**General compatibility**

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

**Supported compiler versions**

For information on supported compiler versions, refer to Basics on Integrating the Simulation System (VEOS Manual 🕮).

### Supported operating systems

**Windows** For information on the Windows operating systems supported by VEOS, refer to Operating System on page 202.

**Linux** For information on the Linux distribution recommended for VEOS, refer to Introduction to VEOS on Linux (VEOS Manual 🕮).

### BSC file compatibility

VEOS 5.3 is compatible with bus simulation container (BSC) files created with the Bus Manager of dSPACE Release 2021-A and Release 2021-B (BSC version 1.10).

- If a BSC file was generated *without an SIC file* or *with an SIC file that was generated for the dsrt.tlc system target file*, you can integrate the BSC file in a VEOS simulation on Windows.
- If a BSC file was generated *with an SIC file that was generated for the dsrt64.tlc system target file*, you can integrate the BSC file in a VEOS simulation on Linux.

### CTLGZ/VECU file compatibility

The following table shows the compatibility between VEOS 5.3 and V-ECU implementation container (CTLGZ/VECU) files:

| V-ECU Implementations Created With... | V-ECU Implementation Version |
|---|---|
| dSPACE Release 2021-B:<br>▪ SystemDesk 5.6<br>▪ TargetLink 5.2 | 3.1[1] |
| dSPACE Release 2021-A and 2020-B:<br>▪ SystemDesk 5.5<br>▪ TargetLink 5.1 | 3.0[1] |
| dSPACE Release 2020-A and 2019-B:<br>▪ SystemDesk 5.4<br>▪ TargetLink 5.0 | 2.10[1] |

[1] You can integrate files of this version in a VEOS simulation on Windows and in a VEOS simulation on Linux.

### FMU file compatibility

- VEOS supports Functional Mock-up Units (FMUs) that comply with the FMI 2.0 standard.
- VEOS supports only the *FMI for Co-Simulation interface*, but not the FMI for Model Exchange interface.

You can integrate an FMU file in a VEOS simulation on Windows and in a VEOS simulation on Linux.

For detailed and up-to-date compatibility information on dSPACE FMI support, refer to:

http://www.dspace.com/go/FMI-Compatibility.

### OSA file compatibility

The following table shows the compatibility between VEOS 5.3 and offline simulation application (OSA) files:

| OSA Files Created with Products Of ... | OSA Version |
|---|---|
| dSPACE Release 2021-B | 5.3 |
| dSPACE Release 2021-A | 5.2[1] |
| dSPACE Release 2020-B | 5.1[1] |
| dSPACE Release 2020-A | 5.0[1] |

[1] You cannot modify the properties of VPUs contained in an OSA file if you open the OSA file in a later VEOS version than the version with which the OSA file was originally created. However, you can edit port and network connections. Therefore, it is recommended to rebuild the binary OSA files from existing model implementation container files (CTLGZ/VECU, SIC, BSC, FMU) when you migrate from one VEOS version to another.

### SIC file compatibility

The following table shows the compatibility between VEOS 5.3 and Simulink implementation container (SIC) files:

| SIC Files Created With ... | SIC Version[1] |
|---|---|
| dSPACE Release 2021-B:<br>▪ Model Interface Package for Simulink 4.6<br>▪ TargetLink 5.2 | 1.10 |
| dSPACE Release 2021-A:<br>▪ Model Interface Package for Simulink 4.5<br>▪ TargetLink 5.1 | 1.10 |
| dSPACE Release 2020-B:<br>▪ Model Interface Package for Simulink 4.4<br>▪ TargetLink 5.1 | 1.9[2] |
| dSPACE Release 2020-A:<br>▪ Model Interface Package for Simulink 4.3<br>▪ TargetLink 5.0 | 1.8[2] |

[1] If an SIC file was generated for the `dsrt.tlc` system target file, you can integrate it in a VEOS simulation on Windows. If an SIC file was generated for the `dsrt64.tlc` system target file, you can integrate it in a VEOS simulation on Linux. Refer to How to Generate a Simulink Implementation Container (Model Interface Package for Simulink - Modeling Guide 📖).

[2] If the SIC file is created with a previous dSPACE Release and if the SIC file contains an ASM model, you cannot simulate the model in VEOS 5.3 (dSPACE Release 2021-B). For more information, refer to Migrating ASM Models (VEOS Manual 📖).

### SMC file compatibility

The following table shows the compatibility between VEOS 5.3 and system model container (SMC) files:

| SMC Files Created With ... | SMC Version |
|---|---|
| dSPACE Release 2021-B:<br>▪ SYNECT 2.12<br>▪ VEOS 5.3 | 1.2 |

| SMC Files Created With ... | SMC Version |
|---|---|
| dSPACE Release 2021-A:<br>▪ SYNECT 2.11<br>▪ VEOS 5.2 | 1.2 |
| dSPACE Release 2020-B:<br>▪ SYNECT 2.10<br>▪ VEOS 5.1 | 1.2 |
| dSPACE Release 2020-A:<br>▪ SYNECT 2.9<br>▪ VEOS 5.0 | 1.2 |
| dSPACE Release 2019-B:<br>▪ SYNECT 2.8<br>▪ VEOS 4.5 | 1.1 |

You also have to consider the following compatibility restrictions of the individual container files contained in the SMC file to be imported: If the SMC file contains a container of an unsupported version, VEOS 5.3 imports neither the unsupported container nor the connections to the application process based on the unsupported container.

**Real-Time Testing compatibility**

To use RTT in connection with VEOS and ControlDesk, the Real-Time Testing (RTT) version used by the VEOS Simulator that runs the simulation system and the RTT version that is active on the PC must be identical.

The following table shows the VEOS Simulator version and the corresponding RTT version:

| VEOS Simulator | RTT Version |
|---|---|
| ... of VEOS 5.3 | Real-Time Testing Version 5.1 |
| ... of VEOS 5.2 | Real-Time Testing Version 5.0 |
| ... of VEOS 5.1 | Real-Time Testing Version 4.4 |
| ... of VEOS 5.0 | Real-Time Testing Version 4.3 |
| ... of VEOS 4.5 | Real-Time Testing Version 4.2 |
| ... of VEOS 4.4 | Real-Time Testing Version 4.1 |
| ... of VEOS 4.3 | Real-Time Testing Version 4.0 |
| ... of VEOS 4.2 | Real-Time Testing Version 3.4 |
| ... of VEOS 4.1 | Real-Time Testing Version 3.3 |
| ... of VEOS 4.0 | Real-Time Testing Version 3.2 |

ControlDesk 7.5 automatically uses the VEOS Simulator of VEOS 5.3. You can therefore use RTT in connection with VEOS and ControlDesk if RTT 5.0 is active on the PC.

**AUTOSAR Adaptive Platform compatibility**

For the simulation of AUTOSAR adaptive V-ECUs, VEOS 5.3 is compatible with the following release of the AUTOSAR Adaptive Platform:

- R20-11

### Related topics

Basics

Hypervisor Configuration for Adaptive V-ECU Simulation on Windows (VEOS Manual 📖)

## Migrating to VEOS 5.3

### Introduction

To migrate from VEOS 5.2 to VEOS 5.3, you might have to perform certain migration steps.

> **Note**
>
> To migrate to VEOS 5.3 from versions earlier than 5.2, you might also have to perform the migration steps of the intervening VEOS versions.

### Migrating ASM models

You cannot simulate an ASM model on VEOS if the model is contained in an OSA or SIC file that was created with a dSPACE Release earlier than the one to which your VEOS installation belongs.

To simulate an ASM model that was last saved with a dSPACE Release earlier than the dSPACE Release to which your VEOS version belongs, perform the following steps:

1. Migrate the ASM model to the dSPACE Release to which your VEOS version belongs.

   For information on migrating ASM models, refer to Migrating ASM Models (ASM User Guide 📖).
2. Generate a Simulink implementation container (SIC) file on the basis of the ASM model by using the *Model Interface Package for Simulink*.

   For instructions, refer to Generating Simulink Implementation Containers (Model Interface Package for Simulink - Modeling Guide 📖).
3. Import the SIC file to the VEOS Player of your VEOS version.

   For instructions, refer to How to Import Simulink Implementations (VEOS Manual 📖).

### Migrating from earlier VEOS versions

To migrate from earlier VEOS versions and reuse existing offline simulation applications, you might have to carry out additional migration steps. For more information on the migration steps, refer to Migrating from Prior Versions of VEOS (VEOS Manual 📖).

# Discontinuations as of VEOS 5.3

### DsIdBusIf support

As of Version 5.3, VEOS no longer supports V-ECUs or V-ECU implementations with a *dSPACE Idealized Bus Interface* (DsIdBusIf) module. The DsIdBusIf module is a dSPACE-specific module for specifying unsupported bus protocols using an idealized bus without specific arbitration.

Use the *Signal-to-Frame Converter* (Sfc) module instead. The Sfc module is a dSPACE-specific module that lets you connect V-ECUs with modeled bus communication to environment models or other V-ECUs that use signal-based communication, such as SIC and FMU, or buses that are not supported. The bus signals can be transferred.

For more information, refer to Signal-to-Frame Converter (Sfc) (SystemDesk Manual 🕮).

### Related topics

Basics

Discontinuations in VEOS (VEOS Manual 🕮)

# Compatibility Information

**Where to go from here**

Information in this section

## Supported MATLAB Releases

**MATLAB®/Simulink®**

Various dSPACE products require a MATLAB installation on the host PC.

**Tip**

For system requirements of MathWorks® software, refer to https://www.mathworks.com/support/sysreq.html.

| MATLAB Release... | ...Is Supported by dSPACE Release 2021-B | | | | | |
|---|---|---|---|---|---|---|
| | RCP and HIL Software [1], [2] | AutomationDesk 6.6[3] | TargetLink 5.2 | Model Compare 3.2 | dSPACE Python Extensions 4.1 [4] | XIL API .NET MAPort 2021-B |
| R2021b | ✓ [5] | ✓ | ✓ | ✓ | ✓ | ✓ |
| R2021a | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| R2020b | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| R2020a | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

[1] 'RCP and HIL software' is a generic term for a software package containing several dSPACE software products, for example, ASM, RTI, ConfigurationDesk, and ModelDesk. These software products are installed in a common folder.

[2] MATLAB/Simulink Student Suite is not supported by Automotive Simulation Models (ASM).

[3] The AutomationDesk MATLAB Access Library requires MATLAB.

[4] matlablib2 of dSPACE Python Extensions requires MATLAB.

[5] R2021b is 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 https://www.dspace.com/go/MATLABCompatibility.

## Operating System

**Windows operating system on host PC**

The dSPACE products of dSPACE Release 2021-B support the following operating systems:

- The following editions, channels, and servicing options of Windows 10:
  - Windows 10 Professional, Education, and Enterprise (64-bit versions)

The Windows 10 Home, Mobile, and Windows 10 S editions are not supported.

- Long-Term Servicing Branch: LTSB 2016

  > **Note**
  >
  > As of dSPACE Release 2022-A, Windows LTSB 2016 is no longer supported.

- Long-Term Servicing Channel: LTSC 2019
- Semi Annual Channel: The compatibility statement of Microsoft applies. This means that newer versions released in this channel should be compatible with all previous versions. dSPACE used the 21H1 version of the Semi Annual Channel for testing.

- Windows Server 2016 Standard and Datacenter edition, each with the Desktop Experience installation option

  Only the listed editions are supported. The Windows Server 2016 Essentials, MultiPoint Premium Server editions are not supported.

  > **Note**
  >
  > As of dSPACE Release 2022-A, Windows Server 2016 is no longer supported.

- Windows Server 2019 Standard and Datacenter editions, each with the Desktop Experience installation option

  Only the listed editions are supported. The Windows Server 2019 Essentials edition is not supported.

Some limitations apply when you use dSPACE software in conjunction with features of Windows. Refer to Limitations for Using Windows Features on page 210.

**Support of Windows Docker** You can also run some dSPACE products in a Windows Docker container. For more details, contact dSPACE Support (https://www.dspace.com/go/supportrequest).

**Linux operating system on host PC**

The dSPACE products of dSPACE Release 2021-B with Linux compatibility support the following operating systems:

- Ubuntu 18.04 LTS with the General Availability Kernel in the Desktop, Server, and Cloud version

Some limitations apply when you use dSPACE software in conjunction with features of Linux. Refer to Limitations for Using Linux Features on page 212.

**Support of Linux Docker** You can also run the dSPACE products with Linux compatibility in a Linux Docker container. For more details, refer to https://www.dspace.com/go/ProductsForLinux.

**Using MicroAutoBox Embedded PC as host PC**

ControlDesk can also be installed on:

- MicroAutoBox Embedded PC 6th Gen. Intel® Core$^{TM}$ i7-6822EQ Processor, running on Microsoft® Windows® 10 IOT Enterprise, LTSB 2016, 64-bit version
- MicroAutoBox III Embedded PC, running on Microsoft® Windows® 10 IoT Enterprise LTSC 2019, 64-bit version

**Operating system on SYNECT server**

The SYNECT server supports the following operating systems:

- The same operating systems as listed above for all dSPACE products of dSPACE Release 2021-B.
- Windows Server 2012, Windows Server 2012 R2

**Operating system on server for floating network licenses**

If you purchased floating network licenses, you have to specify one of the network PCs as a license server. Any PC with CodeMeter Runtime software can be used as a license server.

**Valid for servers without dSPACE software** dSPACE tests license servers only with Microsoft Windows operating systems in combination with protected dSPACE software.

> **Note**
>
> Non-Windows operating systems, such as Ubuntu Linux, are not tested. You can use them at your own risk. dSPACE does not provide support in this case.

**Valid for servers with dSPACE Installation Manager** dSPACE Installation Manager supports the same operating systems as the other dSPACE software products described above.

**Allowing communication**

**Installing of additional firewall rules** Additional Windows firewall rules are installed during the installation of various dSPACE software products. For example, one rule allows communication with a dSPACE expansion box, such as AutoBox. Another rule allows MotionDesk to receive motion data from a network channel. These example rules are created by the following commands:

-
  ```
  netsh advfirewall firewall add rule name="dSPACE Net Service"
  service=any dir=in action=allow profile=any
  protocol=icmpv4:0, any description="Allow the dSPACE Net
  Service to connect to a dSPACE expansion box via network."
  ```

-
  ```
  netsh advfirewall firewall add rule name="dSPACE MotionDesk"
  program=<main installation path>\dSPACE MotionDesk 2021-
  B\MotionDesk\Bin\MotionDesk.exe"
  dir=in action=allow profile=any description="Allow dSPACE
  MotionDesk to receive motion data via network."
  ```

**Required open TCP/IP network ports** If you are using third-party firewall software on your host PC, ensure that the TCP/IP communication of dSPACE software is not blocked:

- VEOS requires the following open TCP/IP network ports:
  - For Windows: 111 (TCP and UDP), 3702 (UDP), 7214 (TCP and TCP6), 7215 (TCP and UDP), 7216 (TCP), 8090 (TCP), 9923 (UDP), 49152 ... 65535 (TCP, TCP6 and UDP)
  - For Linux: 111 (TCP and UDP), 7215 (TCP and UDP), 7216 (TCP), 8090 (TCP), 9923 (UDP), 32768 ... 60999 (TCP, TCP6 and UDP)
- MotionDesk requires the following open TCP/IP network port: 15000 (UDP)
- dSPACE Installation Manager and CodeMeter licensing software require the following open TCP/IP network ports:
  - 22350 (TCP and UDP) for communication in a LAN network (if not changed from the default setting).
  - 22352 (TCP and UDP): To access CodeMeter WebAdmin via http.
  - 22353 (TCP and UDP): To access CodeMeter WebAdmin via https.
- dSPACE Help requires an open TCP/IP network port for interprocess communication between its components. The default port number is 11000. If this port number is already being used, another free port is used automatically. The related processes can be identified via the following prefixes: `HelpAbsLayer<xxx>`, `HelpInstaller<xxx>`.

## Using dSPACE Software on Virtual Machines (VMs)

**Introduction**

You can operate several dSPACE products installed on virtual machines. However, some dSPACE products support VMs only with limitations, and other dSPACE products cannot be operated on VMs at all.

**Usage restrictions**

**Note**

The dSPACE End User License Agreement (EULA) prohibits:

- Using a virtual machine for circumventing license protection mechanisms, for multiple use of an acquired license or for use outside the use determined by the license type.
- Accessing dSPACE software via Internet or network applications (e.g., Citrix, Microsoft Remote Desktop or other terminal/device servers) or to grant such access to third parties.

If you have any questions or encounter any problems, contact dSPACE Support (www.dspace.com/go/supportrequest).

**Recommended virtual machine software**

dSPACE tests the functionality of dSPACE software products with current VMware products and VM hardware compatibility version 10 and version 13.

You can use Windows, Linux, or macOS® as the host operating system.

### Support of dSPACE software on virtual machines

**Note**

The following table shows the compatibility for all dSPACE products. For products that support VMs with limitations, the known limitations are listed. For these products, further limitations might apply depending on the use case.

| Product | Full Support | Support with Known Limitations | | No Support |
|---|---|---|---|---|
| ASM | ✓ | — | | — |
| AutomationDesk | — | ✓ | Known limitations:<br>▪ Access to DS1006 modular systems via dSPACE link boards is not possible.<br>▪ Access to DS1006 modular systems via Ethernet connection and slot CPU: Communication and therefore performance is very low.<br>▪ Access to DS1104 R&D Controller Boards is not possible. | — |
| Bus Manager | ✓ | — | | — |
| ConfigurationDesk for RapidPro | ✓ | — | | — |
| ConfigurationDesk | ✓ | — | | — |
| Container Manager | ✓ | — | | — |
| ControlDesk | — | ✓ | Known limitations:<br>▪ Access to DS1006 modular systems via dSPACE link boards is not possible.<br>▪ Access to DS1006 modular systems via Ethernet connection and slot CPU: Communication and therefore performance is very low.<br>▪ Access to DS1104 R&D Controller Boards is not possible. | — |
| Data Dictionary Manager | ✓ | — | | — |
| DCI-GSI Configuration Package | ✓ | — | | — |
| dSPACE AUTOSAR Compare | ✓ | — | | — |
| dSPACE Installation Manager | ✓ | — | | — |
| ECU Flash Programming Tool | ✓ | — | | — |
| ECU Interface Base Package | ✓ | — | | — |
| ECU bypassing target compiler | ✓ | — | | — |
| Firmware Archives | — | ✓ | Known limitations:<br>▪ Access to DS1006 modular systems via dSPACE link boards is not possible.<br>▪ Access to DS1006 modular systems via Ethernet connection and slot CPU: Communication and therefore performance is very low.<br>▪ Access to DS1104 R&D Controller Boards is not possible. | — |
| Firmware Manager | | | | |
| FlexRay Configuration Tool | ✓ | — | | — |
| Model Compare | ✓ | — | | — |

| Product | Full Support | Support with Known Limitations | No Support |
|---|---|---|---|
| ModelDesk | — | ✓ Known limitations:<br>▪ The Traffic Object Manager cannot show custom sensor points in the preview.<br>▪ Plotting occasionally does not start if a start trigger is used. | — |
| Model Interface Package for Simulink | ✓ | — | — |
| MotionDesk | — | — | ✓[1] |
| Platform API Package | — | ✓ Known limitations:<br>▪ Access to DS1006 modular systems via dSPACE link boards is not possible.<br>▪ Access to DS1006 modular systems via Ethernet connection and slot CPU: Communication and therefore performance is very low.<br>▪ Access to DS1104 R&D Controller Boards is not possible. | — |
| Real-Time Testing | — | ✓ Known limitations:<br>▪ Access to DS1006 modular systems via dSPACE link boards is not possible.<br>▪ Access to DS1006 modular systems via Ethernet connection and slot CPU: Communication and therefore performance is very low.<br>▪ Access to DS1104 R&D Controller Boards is not possible. | — |
| RTI Blocksets (Real-Time Interface) | — | ✓ Known limitations:<br>▪ Access to DS1006 modular systems via dSPACE link boards is not possible.<br>▪ Access to DS1006 modular systems via Ethernet connection and slot CPU: Communication and therefore the performance is very low.<br>▪ Access to DS1104 R&D Controller Boards is not possible. | — |
| Sensor Simulation | — | — | ✓[1] |
| SYNECT | ✓ | — | — |
| SYNECT Server | ✓ | — | — |
| SYNECT License Server | ✓ | — | — |
| SystemDesk | ✓ | — | — |
| TargetLink | ✓ | — | — |
| VEOS | ✓[2] | — | — |

[1] VMs do not fulfill the requirements for graphics adapters.

[2] If you want to simulate adaptive AUTOSAR V-ECUs, refer to Hypervisor Configuration for Adaptive V-ECU Simulation on Windows (VEOS Manual 🕮).

### Required knowledge for setting up a virtual machine

To set up a virtual machine, you must have knowledge about the technology of VMs.

In virtual environments, significantly higher latencies and lower network performance (network throughput) must be expected compared to physical PCs. dSPACE has no influence on this.

**Using virtual machines in parallel** If you use multiple VMs simultaneously on one PC, sharing of host resources such as CPU, network, and disk I/O bandwidth can cause timing issues. dSPACE recommends using a physical PC if high performance is required by an application.

## Using a virtual machine on the host PC

**System requirements** PCs that host virtual machines with dSPACE software, must meet at least the requirements listed in Appendix: System Requirements (Installing dSPACE Software 📖). You are recommended to use a PC with more resources so that the software runs smoothly on a VM, because the VM software itself uses up some of the resources:

- The CPU speed and RAM size must be sufficient to run the operating system and the software on the host PC as well as the guest operating system and the application software on the VM.
- You also require sufficient free disk space to install the VM software and the software you want to run, just as you would if you were installing it directly on your PC.

**Connecting dongle-based devices** If you use dongle-based single-user licenses to use dSPACE software, you first have to connect the CmDongle to the host PC. Then you have to connect the WIBU-Systems CodeMeter-Stick device to the virtual machine on the host PC. For instructions, refer to the documentation of the VM software you use.

**Using floating network licenses** If you use floating network licences, the virtual machine requires access to the dSPACE License Server. For further instructions, refer to How to Set up a Connection Between Client and Server (Working with CodeMeter Licensing Technology 📖).

**Optimal display of dSPACE Help** For an optimal display of the content in dSPACE Help, you have to activate the ClearType setting in the VM (= default setting).

You can access this setting via the Windows Start menu (Start – Control Panel – Appearance and Personalization – Display – Adjust ClearType text).

## Using the 'Revert to snapshot' feature

***NOTICE***

**Using the 'Revert to snapshot' feature causes licenses to become invalid.**

If you use the 'Revert to snapshot' feature in a VM, all software-based CmContainers on your host PC (dSPACE Activation Container and/or dSPACE Borrow Container) become invalid and the contained licenses are lost.

- Do not use the 'Revert to snapshot' feature for VMs that contain software-based CmContainers with activated licenses.
- Store the license information on CmDongles. There, the CmContainers do not become invalid after use the 'Revert to snapshot' feature.

**Moving the virtual machine to a host PC with a different hardware configuration**

***NOTICE***

**Moving the virtual machine to a PC with a different hardware configuration causes licenses to become invalid.**

If you move your virtual machine to a host PC with a different hardware configuration, all software-based CmContainers on your host PC (dSPACE Activation Container and/or dSPACE Borrow Container) become invalid and the contained licenses are lost. This happens, for example, if the CPU type of the physical PC changes.

- Do not move a virtual machine that contains software-based CmContainers with activated licenses to a host PC with a different hardware configuration.
- Store the license information on CmDongles. There, the CmContainers do not become invalid after the virtual machine is moved.
- However, if moving the virtual machine is absolutely necessary, contact dSPACE Support (www.dspace.com/go/supportrequest) beforehand to find solutions that can avoid major downtimes.

## 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 2021-B**

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

Observe the following points:

- Limitations regarding run-time compatibility in the dSPACE tool chain might occur if products from different dSPACE Releases are used together.

  If dSPACE products interact directly (through automation interfaces) or indirectly (through common file types like A2L), limitations might apply. For minor limitations, refer to the relevant product documentation. The major limitations are described in the following.

  In rare cases, an additional patch must be installed for a product to achieve run-time compatibility. For more information on the patch and whether a patch is required, refer to https://www.dspace.com/go/CompPatch.
- RCP and HIL software products (of Release 2021-B) cannot be used in combination with RCP and HIL software products from earlier dSPACE Releases.

**Major limitation for working with a SCALEXIO system and with MicroAutoBox III** The products for working with a SCALEXIO system and with MicroAutoBox III must be compatible. This is guaranteed only for products delivered with the same dSPACE Release. Contact dSPACE for more information.

**Compatibility of real-time applications loaded to a DS1006, DS1104 or MicroAutoBox II platform** If a real-time application is loaded to one of these platforms with a software product of dSPACE Release 2016-B or later, software products of dSPACE Release 2016-A (and earlier) do not detect that the loaded real-time application is the same as the real-time application stored on your host PC. In this case, you cannot work with the related software product without restrictions.

This also applies if you load a real-time application with a software product of dSPACE Release 2016-A or earlier and use software products of dSPACE Release 2016-B or later, for example, for experimenting.

**Combining dSPACE products from earlier Releases**

For more information and notes on the combined use of different products from and with earlier Releases, refer to https://www.dspace.com/go/ds_sw_combi.

## Limitations for Using Windows Features

**Motivation**

Some limitations apply to using dSPACE software in conjunction with features of Windows.

**Installing and running dSPACE software within the Windows service accounts**

Non-service-based dSPACE software is not designed to be installed or run in the context of any predefined Windows service account (LocalService, NetworkService, LocalSystem).

**Fast user switching not supported**

dSPACE software does not support the fast user switching feature of Windows.

**Closing dSPACE software before PC shutdown**

The shutdown process of Windows operating systems might cause some required processes to be aborted although they are still being used by dSPACE software. To avoid a loss of data, it is recommended to close the dSPACE software manually before shutting down the PC.

**User Account Control**

It is recommended to disable the Windows User Account Control (UAC) during the installation of dSPACE software. If you cannot disable UAC, note the following Windows behavior: If UAC is enabled, the setup programs use the administrator account instead of the user account. Therefore, it is important that the administrator account has access to the required drives, particularly the required network drives.

**USB devices**

If you connect dSPACE USB devices that use cables with optoisolation to the PC for the first time, there might be a message that the device driver software was not installed successfully. However, the dSPACE device will work properly later on.

**Using 4K monitors**

The following dSPACE software products have limitations for working with 4K monitors:

- SYNECT: If you use 4K monitors, the SYNECT graphical user interface might not be displayed properly in some cases, but this does not cause functional limitations.
- Real-Time Testing: The Real-Time Test Manager, the user interface for handling RTT sequences, does not support working with 4K monitors.
- FlexRay Configuration Package: The FlexRay Configuration Tool does not support working with 4K monitors.

**FIPS support**

dSPACE software was not developed for or tested against the FIPS PUB 140-2 U.S. government computer security standard (Security Requirements for Cryptographic Modules). For more information on FIPS, refer to https://docs.microsoft.com/en-us/windows/security/threat-protection/fips-140-validation.

**Long paths**

dSPACE software does not support the long path syntax of the Windows API. If a path that exceeds 260 characters is used directly or indirectly, the behavior of the dSPACE software is not defined.

**Enabling Windows 8dot3name creation option**

> **Note**
>
> It is strongly recommended that the Windows **8dot3name creation** option is enabled for all drives (drives used for installation and drives used for work) before you install third-party software, such as MATLAB®/Simulink®, and the dSPACE software.

If the option is disabled during software installation, serious errors can occur when you run the dSPACE software. For example, the build process might be aborted. To repair an installation that was installed while the **8dot3name creation** option was disabled, you have to install dSPACE software and required third-party software again.

For instructions on checking the setting and enabling the option, refer to https://www.dspace.com/faq?346 or to the Microsoft Windows documentation.

**Settings in Windows for user locale and system locale must match**

MATLAB reads the user locale and system locale settings that are specified in Windows operating systems. The user locale and the system locale must match. If these settings are not the same, the system might not behave as expected when working with MATLAB and dSPACE software.

For instructions on checking and changing the settings, refer to https://www.mathworks.com/help/matlab/matlab_env/setting-locale-on-windows-platforms.html?s_tid=gn_loc_drop.

This affects all MATLAB versions and all Windows operating systems, that are supported by dSPACE.

## Limitations for Using Linux Features

**FIPS support**

dSPACE software was not developed for or tested against the FIPS PUB 140-2 U.S. government computer security standard (Security Requirements for Cryptographic Modules). For more information on FIPS, refer to https://ubuntu.com/blog/fips-certification-ubuntu-18-04-lts.

**Long paths**

dSPACE software does not support the available path length of 4,096 characters. If a path that exceeds 260 characters is used directly or indirectly, the behavior of the dSPACE software is not defined.

**Locale**

dSPACE software was tested only on a system with a US English locale.