How to Contact dSPACE

Mail: dSPACE GmbH
Technologiepark 25
33100 Paderborn
Germany
Tel.: +49 5251 1638-0
Fax: +49 5251 66529
E-mail: info@dspace.de
Web: http://www.dspace.com
http://www.caldesk.com

How to Contact dSPACE Support

There are different ways to contact dSPACE Support:
- Visit our Web site at http://www.dspace.com/goto/support
- Send an e-mail or phone:
  - General Technical Support:
    support@dspace.de
    +49 5251 1638-941
  - SystemDesk Support:
    support.systemdesk@dspace.de
    +49 5251 1638-996
  - CalDesk Support:
    support.caldesk@dspace.de
    +49 5251 1638-363
  - TargetLink Support:
    support.tl@dspace.de
    +49 5251 1638-700
- Use the dSPACE Support Wizard:
  - On your dSPACE DVD at \Diag\Tools\dSPACESupportWizard.exe
  - Via Start – Programs – dSPACE Tools (after installation of the dSPACE software)
  - At http://www.dspace.com/goto/supportwizard

You can always find the latest version of the dSPACE Support Wizard here.

dSPACE recommends that you use the dSPACE Support Wizard to contact dSPACE Support.

Important Notice

This document contains proprietary information that is protected by copyright. All rights are reserved. Neither the documentation nor software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of dSPACE GmbH.

© Copyright 2008 by:
dSPACE GmbH
Technologiepark 25
33100 Paderborn
Germany

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

AutomationDesk, CalDesk, ConfigurationDesk, ControlDesk, SystemDesk and TargetLink are registered trademarks of dSPACE GmbH in the United States or other countries, or both. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>About This Document</strong></td>
<td>7</td>
</tr>
<tr>
<td><strong>Overview of dSPACE Release 6.2</strong></td>
<td>9</td>
</tr>
<tr>
<td>General Enhancements and Changes</td>
<td>9</td>
</tr>
<tr>
<td>Product Version Overview</td>
<td>10</td>
</tr>
<tr>
<td>New Product Key Features</td>
<td>11</td>
</tr>
<tr>
<td>Migrating to dSPACE Release 6.2</td>
<td>14</td>
</tr>
<tr>
<td><strong>AutomationDesk</strong></td>
<td>15</td>
</tr>
<tr>
<td>New Features of AutomationDesk 2.1</td>
<td>15</td>
</tr>
<tr>
<td><strong>Automotive Simulation Models (ASM)</strong></td>
<td>17</td>
</tr>
<tr>
<td>ASM Brake Hydraulics Blockset</td>
<td>18</td>
</tr>
<tr>
<td>New Features of ASM Brake Hydraulics Blockset 1.1</td>
<td>18</td>
</tr>
<tr>
<td>Migrating to ASM Brake Hydraulics Blockset 1.1</td>
<td>19</td>
</tr>
<tr>
<td>ASM Drivetrain Basic Blockset</td>
<td>20</td>
</tr>
<tr>
<td>New Features of ASM Drivetrain Basic Blockset 1.2</td>
<td>20</td>
</tr>
<tr>
<td>Migrating to ASM Drivetrain Basic Blockset 1.2</td>
<td>20</td>
</tr>
<tr>
<td>ASM Electric Components Blockset</td>
<td>21</td>
</tr>
<tr>
<td>New Blockset ASM Electric Components 1.0</td>
<td>21</td>
</tr>
<tr>
<td>ASM Engine Diesel Blockset</td>
<td>22</td>
</tr>
<tr>
<td>New Features of ASM Engine Diesel Blockset 1.3</td>
<td>22</td>
</tr>
<tr>
<td>Migration to ASM Engine Diesel Blockset 1.3</td>
<td>22</td>
</tr>
<tr>
<td>ASM Engine Gasoline Basic Blockset</td>
<td>24</td>
</tr>
<tr>
<td>Migrating to ASM Engine Gasoline Basic Blockset 1.3</td>
<td>24</td>
</tr>
<tr>
<td>ASM Engine Gasoline Blockset</td>
<td>25</td>
</tr>
<tr>
<td>Migrating to ASM Engine Gasoline Blockset 2.1</td>
<td>25</td>
</tr>
<tr>
<td>ASM Environment Blockset</td>
<td>26</td>
</tr>
<tr>
<td>New Features of ASM Environment Blockset 1.3</td>
<td>26</td>
</tr>
<tr>
<td>Migrating to ASM Environment Blockset 1.3</td>
<td>27</td>
</tr>
<tr>
<td>ASM Utilities</td>
<td>30</td>
</tr>
<tr>
<td>Migrating to ASM Utilities 1.5.1</td>
<td>30</td>
</tr>
<tr>
<td>ASM Vehicle Dynamics Blockset</td>
<td>31</td>
</tr>
<tr>
<td>New Features of ASM Vehicle Dynamics Blockset 1.2</td>
<td>31</td>
</tr>
</tbody>
</table>
## Contents

- **Migrating to ASM Vehicle Dynamics Blockset 1.2** .................. 33

### ControlDesk
- **New Features of ControlDesk 3.2.2** ........................................ 39

### dSPACE Data Dictionary
- **Migrating to dSPACE Data Dictionary 1.5.1** .......................... 42
- **How to Upgrade a Data Dictionary Without Include Files** ...... 45
- **How to Upgrade a Data Dictionary With Include Files** ......... 46

### dSPACE FlexRay Configuration Package
- **New Features of dSPACE FlexRay Configuration Package 1.10** 49

### Model Compare
- **New Features of Model Compare 2.0** ..................................... 51
- **Migration to Model Compare 2.0** ......................................... 52

### ModelDesk
- **New Features of ModelDesk 2.1** .......................................... 55
- **Migration to ModelDesk 2.1** ................................................. 55

### RTI
- **New Features of RTI 6.1** .................................................... 57

### RTI Bypass Blockset
- **New Features of the RTI Bypass Blockset 2.4.1** ..................... 59

### RTI CAN MultiMessage Blockset
- **New Features of the RTI CAN MultiMessage Blockset 2.3** ...... 61

### SystemDesk
- **New Features of SystemDesk 1.1** ....................................... 63
- **Migration to SystemDesk 1.1** ............................................. 65

### TargetLink
- **New Features of TargetLink 3.0** ......................................... 68
- **New Blockset Features** ..................................................... 68
- **New Blockset Features** ..................................................... 68
- **Signal Width Inheritance** .................................................. 76
Contents

Model Referencing ................................................................. 77
New Features of Model Referencing ........................................ 77
Miscellaneous Features .......................................................... 79
Miscellaneous Features ......................................................... 79
Migrating Existing Models and Data Dictionary Files ................. 81
Model Upgrade ....................................................................... 82
How to Upgrade Models and Libraries .................................... 83
Migration Aspects for Former SL2TL and TL2SL Conversion
(System Preparation and System Clearance) ......................... 87
TargetLink Port Blocks Discontinued ...................................... 93
Modified TargetLink Blocks ................................................... 94
Scalar Expansion and Block Output Width ............................. 95
Replacement for TargetLink Port Blocks Used for Rescaling ..... 97
Logging Behavior Changes ..................................................... 98
Simulation Behavior Changes ............................................... 106
Code-Relevant Stateflow Enhancements and Changes ........... 108
Changes in TargetLink API Functions ................................. 112
Changes on TargetLink Block Properties ............................. 113
Various Migration Aspects .................................................... 114

Compatibility Information .................................................... 119
Supported MATLAB Releases .............................................. 119
Supported Operating Systems .............................................. 120

Index .................................................................................... 123
About This Document

This document informs you about the new features of all the dSPACE software products in dSPACE Release 6.2. It also gives you an overview of software products with no or minor changes. There are instructions on migrating from older dSPACE releases, especially from older product versions, if required.
Overview of dSPACE Release 6.2

Objective
Gives you an overview of the new key features in dSPACE Release 6.2, and also information about unchanged products and general instructions on migrating.

Where to go from here
Information in this section

<table>
<thead>
<tr>
<th>General Enhancements and Changes</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Version Overview</td>
<td>10</td>
</tr>
<tr>
<td>New Product Key Features</td>
<td>11</td>
</tr>
<tr>
<td>Migrating to dSPACE Release 6.2</td>
<td>14</td>
</tr>
</tbody>
</table>

General Enhancements and Changes

Objective
The following new features concern several dSPACE products.

New documentation features
The New Features and Migration document that you are reading contains information about all the dSPACE software products. There are no more separate documents for RCP&HIL software, TargetLink, and CalDesk.

Because CalDesk has no new features in dSPACE Release 6.2, you must look at the separate CalDesk New Features and Migration document for further information about CalDesk 2.0.
Release update

The printed user documentation is not delivered with dSPACE Release 6.2 if you receive the release as an update for your existing dSPACE release. Use the current online help, for example, dSPACE HelpDesk, to obtain information about new features, enhancements, and the current safety precautions regarding your products.

Product Version Overview

Objective

The following table shows the product versions of the current release and of three older releases as an extract of the version history of a certain product.

<table>
<thead>
<tr>
<th>Product</th>
<th>dSPACE Release</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.4</td>
</tr>
<tr>
<td>Automotive Simulation Models</td>
<td>1.3.1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>AutomationDesk</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>CalDesk</td>
<td>1.4.1</td>
</tr>
<tr>
<td>ConfigurationDesk</td>
<td>1.2.1</td>
</tr>
<tr>
<td>ControlDesk</td>
<td>3.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>dSPACE Data Dictionary</td>
<td>1.4</td>
</tr>
<tr>
<td>dSPACE FlexRay Configuration Package</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Model Compare</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>ModelDesk</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>MotionDesk</td>
<td>2.1.1</td>
</tr>
<tr>
<td>RTI</td>
<td>5.5</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview of dSPACE Release 6.2

New Features and Migration

If you have not updated regularly, you can look at the New Features and Migration documents for the dSPACE releases listed above for information about the new features and necessary migration steps.

For the new features of CalDesk 2.0 since CalDesk 1.4.0 and the necessary migration steps, refer to dSPACE Calibration System New Features and Migration.

### New Product Key Features

**Objective**

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

**AutomationDesk**

The new key features of AutomationDesk are:

- Enhancements to the Remote Diagnostics (COM) library, for example, ECU Variant support and ECU connection handling.
- The Failure Simulation Access library now supports the DS1450 Bus FIU Board.

<table>
<thead>
<tr>
<th>Product</th>
<th>dSPACE Release</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5.4</td>
</tr>
<tr>
<td>RTI-MP</td>
<td>5.2</td>
</tr>
<tr>
<td>RTIBypass Blockset</td>
<td>2.2.3</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>RTI CAN Blockset</td>
<td>2.6.3</td>
</tr>
<tr>
<td>RTI CAN MultiMessage Blockset</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>RTI LIN MultiMessage Blockset</td>
<td>1.3</td>
</tr>
<tr>
<td>RTI RapidPro Control Unit Blockset</td>
<td>1.5</td>
</tr>
<tr>
<td>SystemDesk</td>
<td>(1.0)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TargetLink</td>
<td>2.2.1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable Editor</td>
<td>1.0</td>
</tr>
</tbody>
</table>

If you have not updated regularly, you can look at the New Features and Migration documents for the dSPACE releases listed above for information about the new features and necessary migration steps.

For the new features of CalDesk 2.0 since CalDesk 1.4.0 and the necessary migration steps, refer to dSPACE Calibration System New Features and Migration.
Real-Time Testing supports multiprocessor systems. You can access variables of real-time applications running on a different CPU than the CPU where the RTT sequence is running.

In Real-Time Testing you can stimulate variables with data from MAT files.

For details on the new features, refer to New Features of AutomationDesk 2.1 on page 15.

Automotive Simulation Models (ASM)

For details on the new features, refer to Automotive Simulation Models (ASM) on page 17.

ControlDesk

The new key features of ControlDesk are:

- The Failure Simulation component supports the DS1450 Bus FIU Board.
- ControlDesk supports the DS2302 Direct Digital Synthesis Board with a revision of DS2302-04.

For details on the new features, refer to New Features of ControlDesk 3.2.2 on page 39.

dSPACE Data Dictionary

For details on migration aspects, refer to dSPACE Data Dictionary on page 41.

dSPACE FlexRay Configuration Package

The new key feature of the dSPACE FlexRay Configuration Tool is:

- Improved use of checksum algorithms
- Extended TRC file generation

The new key feature of dSPACE FlexRay Configuration Blockset is:

- You can manipulate update bits of frames and the payload length.

For details on the new features, refer to New Features of dSPACE FlexRay Configuration Package 1.10 on page 49.

Model Compare

The new key features of Model Compare are:

- You can eliminate differences between the reference and comparison models by merging the two models.
- You can create PDF/HTML reports according to your own layout rules.

For details on the new features, refer to New Features of Model Compare 2.0 on page 51.
ModelDesk

The new key feature of ModelDesk is:

- You can now migrate projects and experiments created with ModelDesk 2.0 or earlier.

For details on the new features, refer to Migration to ModelDesk 2.1 on page 55.

RTI

The new key feature of RTI is:

- The implementation software for DS2211 supports the DS2211 with board revision DS2211-04. The board can simulate a reverse crank sensor and has smooth camshaft phase updating.

For details on the new features, refer to New Features of RTI 6.1 on page 57.

RTI Bypass Blockset

The new key features of the RTI Bypass Blockset are:

- Deleting old, unused data dictionaries of models built with the RTI Bypass Blockset
- New configuration options for the CCP and XCP on UDP/IP interfaces

For details on the new features, refer to New Features of the RTI Bypass Blockset 2.4.1 on page 59.

RTI CAN MultiMessage Blockset

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

- New counter modes
- ECU as hierarchy levels in the TRC file and RX output port bus structure

For details on the new features, refer to New Features of the RTI CAN MultiMessage Blockset 2.3 on page 61.

SystemDesk

The new key features of SystemDesk are:

- Importing and exporting AUTOSAR 2.1 files
- Importing LDF files
- Linking SystemDesk to third-party tools
- More flexible element names
- Exporting data on connections between application software components and AUTOSAR services

For details on the new features, refer to New Features of SystemDesk 1.1 on page 63.
Overview of dSPACE Release 6.2

| TargetLink | For details on the new features and migration aspects, refer to TargetLink on page 67. |

Migrating to dSPACE Release 6.2

<table>
<thead>
<tr>
<th>Objective</th>
<th>After you install dSPACE Release 6.2, some additional steps may be necessary.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Migration from dSPACE Release 6.1</th>
<th>There are no general migration steps to be done. For product-specific migration steps, refer to the product section.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Migrating from dSPACE Release 6.0 or earlier</th>
<th>To migrate from dSPACE Release 6.0 or earlier to dSPACE Release 6.2, you also have to perform the migration steps of the intervening dSPACE Releases. All of the required migration steps can be done with dSPACE Release 6.2 installed.</th>
</tr>
</thead>
</table>

| Example | For example, if you want to migrate from dSPACE Release 5.4 to dSPACE Release 6.2, you have to perform the migration steps described in:  
1. New Features and Migration of dSPACE Release 5.4  
2. New Features and Migration of dSPACE Release 6.0  
3. New Features and Migration of dSPACE Release 6.1  
4. Finally, the migration steps described above. |
|--------|-------------------------------------------------------------------------------------------------------------------|

| Previous release documents | The New Features and Migration documents for previous releases are available via Internet and on the dSPACE DVD:  
- Read them from the dSPACE DVD (see the \Doc folder). The PDF files are called NewFeaturesAndMigrationxx.pdf, where xx stands for the release number. |
|--------------------------|-------------------------------------------------------------------------------------------------------------------|

Until this release, the new features and migration steps for RCP & HIL software, CalDesk and TargetLink were described in separate documents.
# AutomationDesk

## New Features of AutomationDesk 2.1

<table>
<thead>
<tr>
<th>AutomationDesk 2.1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enhancements to the Remote Diagnostics (COM) library</strong></td>
</tr>
<tr>
<td>- ECU variants that are defined in the ODX database can be used in AutomationDesk.</td>
</tr>
<tr>
<td>- There is a new block to set the parameters of a service dynamically during the execution of a test.</td>
</tr>
<tr>
<td>- There are new blocks for managing the ECU communication. You can open, close, and reset a logical link.</td>
</tr>
<tr>
<td>- The library now supports DTS 7.63.020 and DTS 7.65.031.</td>
</tr>
</tbody>
</table>

**Enhancement to the Failure Simulation Access library**

AutomationDesk now supports the new DS1450 Bus FIU Board.

<table>
<thead>
<tr>
<th>Real-Time Testing 1.4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Datastreaming</strong></td>
</tr>
<tr>
<td>You can stimulate variables in an RTT sequence with data from a MAT file. Data is copied in parts from the host PC to the real-time system while the RTT sequence is running. By this, even large MAT files located on the host PC can be replayed by Real-Time Testing. For details, refer to Basics of Data Replay (<a href="#">Real-Time Testing Guide</a>).</td>
</tr>
</tbody>
</table>

**MP support** |
When using Real-Time Testing on a CPU of a multiprocessor system, you can transparently access data of all connected CPUs. For details, refer to Real-Time Testing in a Multiprocessor System ([Real-Time Testing Guide](#)).
**General improvement**  The amount of data required for the Python interpreter on the real-time system was reduced. Python modules which are not necessary are no longer loaded to the real-time system. This speeds up the first download of an RTT sequence and decreases the memory required by the Python interpreter.
# Automotive Simulation Models (ASM)

## Where to go from here

<table>
<thead>
<tr>
<th>Information in this section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM Brake Hydraulics Blockset</td>
<td>18</td>
</tr>
<tr>
<td>ASM Drivetrain Basic Blockset</td>
<td>20</td>
</tr>
<tr>
<td>ASM Electric Components Blockset</td>
<td>21</td>
</tr>
<tr>
<td>ASM Engine Diesel Blockset</td>
<td>22</td>
</tr>
<tr>
<td>ASM Engine Gasoline Basic Blockset</td>
<td>24</td>
</tr>
<tr>
<td>ASM Engine Gasoline Blockset</td>
<td>25</td>
</tr>
<tr>
<td>ASM Environment Blockset</td>
<td>26</td>
</tr>
<tr>
<td>ASM Utilities</td>
<td>30</td>
</tr>
<tr>
<td>ASM Vehicle Dynamics Blockset</td>
<td>31</td>
</tr>
</tbody>
</table>

## Information in other sections

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

Provides general information on the migration process of ASM models.
ASM Brake Hydraulics Blockset

New Features of ASM Brake Hydraulics Blockset 1.1

New parameter for valve state and flow direction

New Sw_nom_state mask parameter to switch between the nominal valve open states "normally closed" or "normally open". New Sw_FlowDirection mask parameter to switch the valve flow directions between one-way and two-way. The mask parameters are new in the following blocks:

- PRE_CHARGE_VALVE_1, PRE_CHARGE_VALVE_2
- CHANGE_OVER_VALVE_1, CHANGE_OVER_VALVE_2
- INLET_VALVE_FL, INLET_VALVE_FR, INLET_VALVE_RL, INLET_VALVE_RR
- OUTLET_VALVE_FL, OUTLET_VALVE_FR, OUTLET_VALVE_RL, OUTLET_VALVE_RR

Operator version

Now this blockset is available as operator version. This is a model variant specifically designed for offline simulation with Simulink®.

The model offers the same functionality, simulation quality and parameterization options as the standard simulation package. The operator version is compatible with the standard model (developer version) and can be parameterized in ModelDesk, the parameterization software.

The fundamental difference with this model is the way the library components are implemented: They are encapsulated in separate systems to ensure good performance during Simulink simulation. The systems are accessible in the model so that their input/output behaviour can be studied.
Migrating to ASM Brake Hydraulics Blockset 1.1

**General changes**

**Added SIM.Stepsize parameter**  The simulation step size has been added to the masks of the following blocks. Now all the parameters of a library block are parameterized via the mask:
- BRAKE_CYLINDER
- DAMPER_CHAMBER
- CONNECTION_CHAMBER
- RESERVOIR
- BRAKING_CIRCUIT_BASIC

**Added MDL.BrkHyd.UpSampleFaktor parameter**  The upsample factor for oversampling the brake hydraulics has been added to the masks of the following blocks. Now all the parameters of a library block are parameterized via the mask:
- BRAKE_CYLINDER
- DAMPER_CHAMBER
- CONNECTION_CHAMBER
- RESERVOIR

The simulation step size (SIM.StepSize parameter) and the upsample factor (MDL.BrkHyd.UpSampleFaktor parameter) are not accessible from ModelDesk. To change these parameters, edit the `sim_ini.m` function in `%PROJECT_ROOT\%Simulation.current\IniFiles`.

**Brake hydraulics**

**MASTER_BRAKE_CYLINDER block**  A new p_Ambient input port was added for the ambient pressure. The p_Ambient mask parameter was removed.
New Features of ASM Drivetrain Basic Blockset 1.2

Redesigned friction model
The friction model has been enhanced with additional saturation of the static torque. The following blocks are affected:
- CLUTCH
- LOCKUP_CLUTCH
- GEARBOX_AT
- GEARBOX_MT

Migrating to ASM Drivetrain Basic Blockset 1.2

General changes
Replaced Mux blocks with BusCreator In the following blocks, the Mux for creating a bus have been replaced by BusCreator blocks:
- SIGNAL_SELECTION
- COMMON_DRIVETRAIN_PARAMETERS
- STARTER

Drivetrain
STARTER block The MDL.DrivetrainBasic.Starter.Const_n_Starter_Max mask parameters is now only used once in the model. The n_Engine[rad/s] input value is converted to unit [rpm] internally.
ASM Electric Components Blockset

New Blockset ASM Electric Components 1.0

The ASM Electric Component Library includes modules such as machines, loads and power electronics to simulate the electric circuits of an automotive system. It is divided into two basic parts: one subsystem with model components to simulate the Automotive Electrical System and one subsystem with models for Electric Components Closed Loop applications such as testing a real machine controller in an HIL system.

For details, refer to Electric Components Overview (ASM Electric Components Reference).
ASM Engine Diesel Blockset

Where to go from here

Information in this section

New Features of ASM Engine Diesel Blockset 1.3

34 cylinders

The maximum number of cylinders has been increased to 34 (from 20). Some modifications were made to the following blocks:
- SOFT_ECU_DIESEL
- INJECTOR
- UNIT_INJECTOR

Migration to ASM Engine Diesel Blockset 1.3

General changes

**Added SIM.Stepsize and SIM.ItNum parameters**
The simulation step size and the iteration number of For Iterator subsystems have been added to the masks of the following blocks. Now all the parameters of a library block are parameterized via the mask:
- EGR_VALVE
- THROTTLE_VALVE
- INTAKE_MANIFOLD
- EXHAUST_MANIFOLD
- SOFT_ECU_DIESEL
- COMMON_DIESEL_PARAMETERS

**COMMON_DIESEL_PARAMETERS**
The Mux block has been replaced by the BusCreator block for creating bus signals.

**COOLER block**
The unit of cooler parameters in the mask of the block was changed from “degC” to “K”.

**DIESEL_OXIDATION_CATALYST block**
The Const.lambda.Uplim parameter was deleted, because it is not used inside the block.
Fuel system

**INJECTOR block**  The `Const_phi_Inj_TrqEffLimit[bTDC]` constant block was moved one level up to avoid the parameter being used twice in the block. This was necessary so that the parameter can be changed via ControlDesk or ModelDesk.

**UNIT_INJECTOR**  The `Const_phi_Inj_TrqEffLimit[bTDC]` constant block was moved one level up to avoid the parameter being used twice in the block. This is necessary so that the parameter can be changed via ControlDesk or ModelDesk.

Soft ECU

**SOFT_ECU_DIESEL block**  The `MDL.SoftECU.SoftECUDiesel.Map_q_Inj_Smoke_Limit` mask parameter is only used once in the model. This is necessary so that the parameter can be changed via ControlDesk or ModelDesk. If the number of injection pulses was set to one, the soft ECU calculated a wrong value for the injection quantity set point. This has been fixed in this version. Now the friction torque is included in the calculation of the driver's torque set point. An additional parameter containing the friction torque map has been added to the block mask for this.
ASM Engine Gasoline Basic Blockset

Migrating to ASM Engine Gasoline Basic Blockset 1.3

General changes

<table>
<thead>
<tr>
<th>Added SIM.Stepsize and SIM.ItNum parameter</th>
<th>The simulation step size and the iteration number of For Iterator subsystems have been added to the masks of the following blocks. Now all the parameters of a library block are parameterized via the mask:</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ SOFTWARE_ECU_GASOLINEBASIC</td>
<td>The Mux block has been replaced by the BusCreator block for creating bus signals.</td>
</tr>
<tr>
<td>■ SOFTWARE_ECU_GASOLINE</td>
<td></td>
</tr>
<tr>
<td>■ MAPS_TC</td>
<td></td>
</tr>
</tbody>
</table>
# ASM Engine Gasoline Blockset

## Migrating to ASM Engine Gasoline Blockset 2.1

<table>
<thead>
<tr>
<th>General changes</th>
<th>Added SIM.Stepsize and SIM.ItNum parameter</th>
<th>The simulation step size and the iteration number of For Iterator subsystems have been added to the masks of the following blocks. Now all the parameters of a library block are parameterized via the mask:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• SOFT_ECU_GASOLINE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• DIRECTINJECTOR</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>COMMON_GASOLINE_PARAMETERS</strong></td>
<td>The Mux block has been replaced by the BusCreator block for creating bus signals.</td>
</tr>
</tbody>
</table>

### Related topics

- Basics
  - Migrating ASM Models ([ASM User Guide](#))
# ASM Environment Blockset

## New Features of ASM Environment Blockset 1.3

### Driver gets vehicle mass and inertia from vehicle model

The driver now detects information on additional loads. This is especially helpful for truck simulation. The driver parameterization no longer needs adapting when the vehicle mass changes. The required information is transmitted to the driver via signal lines. The old `Const_m_Vehicle`, `Const_Inertia_Tensor_Vehicle`, and `Const_PosVec_CoG_Vehicle` driver parameters were removed. This affects the following blocks:

- LATERAL_CONTROL1
- CONTROLLER
- COMMON_DRIVER_PARAMETERS

### Enhanced BASIC_ROADS block

The slope, lateral slope, friction coefficients and tire parameters have been moved from internal constant values to block inputs. They can therefore be used in the model without breaking the link to the library block.

- BASIC_ROADS

### Enhanced driver startup parameterization

The GEAR_SHIFTER block now contains a new parameter `Const_Pos_AccPedal_Startup_FF`. This defines the accelerator pedal position during the startup process.

- GEAR_SHIFTER

### Handling up to 25 gears

The driver parameterization was adapted to handle up to 25 gears.

- GEAR_SHIFTER
### Operator version

Now this blockset is available as operator version. This is a model variant specifically designed for offline simulation with Simulink®.

The model offers the same functionality, simulation quality and parameterization options as the standard simulation package. The operator version is compatible with the standard model (developer version) and can be parameterized in ModelDesk, the parameterization software.

The fundamental difference with this model is the way the library components are implemented: They are encapsulated in separate systems to ensure good performance during Simulink simulation. The systems are accessible in the model so that their input/output behaviour can be studied.

### Migrating to ASM Environment Blockset 1.3

#### Road

**BASIC_ROADS block**  The slope, lateral slope, friction coefficients and tire parameters have been moved from internal constant blocks to block inputs. During migration from previous versions, the subsystem `migrate60_newinports` is inserted before the `BASIC_ROADS` block. This subsystem contains default values for all new input ports. All inputs are connected to this subsystem.

#### Driver

**COMMON_DRIVER_PARAMETERS**  The Const_m_Vehicle parameter was removed. The necessary information is now provided to the relevant blocks via signal lines.

**CONTROLLER**  The vehicle mass information was previously provided by a from-goto connection from the `COMMON_DRIVER_PARAMETERS` block. It must now be provided as an input signal via a signal line. The mass signal is connected automatically during the migration process. If problems occur with automatic connection, refer to the ASM_VehicleDynamics default model or to the following description:

- The `VehicleMovement.VehicleMassAndAdditionalLoads.m_TotalVehicle[kg]` bus signal must be selected using a Bus Selector block from the `VehicleDynamics_Signals` bus. This selected signal must be connected to the `m_Vehicle_Total[kg]` block input.
GEAR_SHIFTER The dimension of the Map_omega_UpShift and Map_omega_DownShift shift tables was increased to handle up to 25 gears. The ASM SHIFTTABLE_LOOKUP block from the ASM Utilities library is used to evaluate the shift tables. This avoids multiple use of local mask variables. The GEAR_SHIFTER block now contains the new Const_Pos_AccPedal_Startup_FF parameter. During startup it is assumed that an accelerator pedal position defined by Const_Pos_AccPedal_Startup_FF (default: 5%) is sufficient to reach the engine speed defined by Const_n_Engine_StartUp_Set (default: 1100 rpm). If this is not the case, engine startup will fail. Most likely the engine parameterization must be modified in this case. As a workaround, the Const_Pos_AccPedal_Startup_FF parameter can be increased.

LATERAL_CONTROL1 The Const_Inertia_Tensor_Vehicle and Const_PosVec_CoG_Vehicle parameters were removed. The necessary information is now provided via signal lines. The vehicle mass information was previously provided by a from-goto connection from the COMMON_DRIVER_PARAMETERS block. It must now be provided as an input signal via a signal line.

The new block inputs must be connected to signals from a bus selector which is connected to the VehicleDynamics_Signals bus. The signals are connected automatically during the migration process. If problems occur with automatic connection, refer to the ASM_VehicleDynamics default model or to the following description:

- The m_Vehicle_Total[kg] block input must be connected to the VehicleMovement.VehicleMassAndAdditionalLoads.m_TotalVehicle[kg] bus signal.
- The Pos_x_CoG_Vehicle_Total[m] block input must be connected to the first element of the bus signal VehicleMovement.VehicleMassAndAdditional Loads.PosVec_CoG_TotalVehicle[x;y;z][m].

LATERAL_CONTROL2 The controller parameters in the default parameterization have been modified to ensure better stability on straight roads. The following table shows the old and new parameter values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Old Value</th>
<th>New Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Const_LatCtrl_kLat</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
These parameter changes are not automatically transferred to the parameterization. You have to do this manually if necessary.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Old Value</th>
<th>New Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Const_LatCtrl_kLat_i</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Const_LatCtrl_kLat_Preview</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>Const_LatCtrl_kYaw</td>
<td>35</td>
<td>30</td>
</tr>
</tbody>
</table>
ASM Utilities

Migrating to ASM Utilities 1.5.1

<table>
<thead>
<tr>
<th>General changes</th>
<th>Added SIM.Stepsize parameter</th>
<th>The simulation step size has been added to the masks of the following blocks. Now all the parameters of a library block are parameterized via the mask:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FirstOrderDynamics</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>SecondOrderDynamics</td>
<td>-</td>
</tr>
</tbody>
</table>
ASM Vehicle Dynamics Blockset

Where to go from here

Information in this section

New Features of ASM Vehicle Dynamics Blockset 1.2 31

Migrating to ASM Vehicle Dynamics Blockset 1.2 33

New Features of ASM Vehicle Dynamics Blockset 1.2

Improved idle speed control

The idle speed controller has improved transient behavior between the idle speed controlled accelerator pedal position and the forwarded driver accelerator pedal position. This affects the following block:

- THROTTLE

SoftECU handles up to 25 gears

The dimension of the shift tables was increased to handle up to 25 gears. This affects the following block:

- SOFT_ECU_TRANSMISSION

Improved friction model

The friction model has been improved with additional saturation of the static torque. This affects the following blocks:

- CLUTCH
- LOCKUP_CLUTCH
- WHEEL_SPEED
- CENTRAL_DIFFERENTIAL
- FRONT_DIFFERENTIAL
- REAR_DIFFERENTIAL

Enhanced parameterization of gearbox

The inertia parameterization has been adapted to a more general implementation. The transmission inertia reduced to the output shaft is now parameterized with a look-up table. The inertia of the input shaft is now only needed for simulating neutral gear. The transmission efficiency parameter has been replaced by a forward and a reverse efficiency look-up table which are functions of the gear. The transient behavior between neutral and first gear has been improved. This affects the following blocks:

- GEARBOX_MT
New Features and Migration July 2008

- **GEARBOX_AT**

**Improved differential locks**
The differential lock has been redesigned. You can select the parts to be connected by the lock clutch with the new Sw_DifferentialLock_Mode parameter. It is now possible to simulate a part-time all-wheel drive with appropriate parameters. Refer to Differentials ([ASM Vehicle Dynamics Reference](#)). This affects the following blocks:
- CENTRAL_DIFFERENTIAL
- FRONT_DIFFERENTIAL
- REAR_DIFFERENTIAL

**Consideration of ambient pressure**
A new p_Ambient input port for ambient pressure was added. This solves issues due to mixed use of absolute and relative pressures in the brake system. This affects the following block:
- BRAKE_DISC

**Enhanced parameterization of STEERING block**
The transient friction coefficient has been put in the block's mask and is now accessible from ModelDesk. Before, the parameter was set to constant below the mask. A new Const_m_steering_rod parameter has been added to the summation of the wheel masses to take the mass of the steering rod itself into account.
- STEERING

**Considering external forces and torques**
This VEHICLE_MOTION_CAR subsystem was given four new inports: F_External_CoorSys_V[x;y;z][N], Trq_External_CoorSys_V[x;y;z][Nm], Stabilization_F_External[3x6] and Stabilization_Trq_External[3x6]. Thus, external forces and torques acting on the vehicle's reference system can be taken into account in the generalized forces and torques calculation and in the mass matrix of the system. This affects the following subsystem:
- VEHICLE_MOTION_CAR

**Redesigned aerodynamics model**
The aerodynamics forces and torques are now functions of the aerodynamics coefficients (which are functions of the angle of incidence), the longitudinal shadow area of the vehicle, and the characteristic length of the vehicle (mostly the vehicle wheelbase). Thus, several new parameters have been added to the block's mask. The main difference in the new aerodynamics implementation compared to the former one is the way the aerodynamics coefficients are included. These are now functions of the angle of incidence. These
functions can be gathered from wind tunnel tests. In addition the pressure point has been replaced by the aerodynamics reference point. If you want to change back to the former aerodynamics implementation, you can find the former block in the `FormerVersions` subsystem in the aerodynamics library. Just drag and drop the former block to your Simulink model. See also Migrating to ASM Vehicle Dynamics Blockset 1.2 on page 33. This affects the following block:

- AERODYNAMICS

**Operator version**

Now this blockset is available as operator version. This is a model variant specifically designed for offline simulation with Simulink®. The model offers the same functionality, simulation quality and parameterization options as the standard simulation package. The operator version is compatible with the standard model (developer version) and can be parameterized in ModelDesk, the parameterization software.

The fundamental difference with this model is the way the library components are implemented: They are encapsulated in separate systems to ensure good performance during Simulink simulation. The systems are accessible in the model so that their input/output behaviour can be studied.

**Migrating to ASM Vehicle Dynamics Blockset 1.2**

**General changes**

*Added SIM.Stepsize parameter*  The simulation step size has been added to the masks of the following blocks. Now all the parameters of a library block are parameterized via the mask:

- ESP_TORQUE_INTERVENTION_SLOW
- ENGINE_BASIC
- ESP_TORQUE_INTERVENTION_FAST
- STABILIZATION
- CRANK_SHAFT
- CLUTCH
- GEARBOX_MT
- LOCKUP_CLUTCH
- GEARBOX_AT
- CENTRAL_DIFFERENTIAL, FRONT_DIFFERENTIAL, REAR_DIFFERENTIAL
- SHAFT_CR, SHAFT_CF, SHAFT_FL, SHAFT_FR, SHAFT_RL, SHAFT_RR
- STEERING
- TIRE_MODEL_TMEASY_FL, TIRE_MODEL_TMEASY_FR, TIRE_MODEL_TMEASY_RL, TIRE_MODEL_TMEASY_RR
- TIRE_MODEL_MF_FL, TIRE_MODEL_MF_FR, TIRE_MODEL_MF_RL, TIRE_MODEL_MF_RR
- COORDINATE_TRANSFORMATION
- VEHICLE_MOVEMENT_INFO_1_0
- VEHICLE_MOTION_1_0
- VEHICLE_MOTION_CAR
- VEHICLE_MOVEMENT_INFO_CAR
- WHEEL_SPEED
- SUSPENSION_FORCE_KINEMATICS_FRONT, SUSPENSION_FORCE_KINEMATICS_REAR
- SUSPENSION_COMPLIANCE_FRONT, SUSPENSION_COMPLIANCE_REAR
- BASIC_ROADS
- LATERAL_CONTROL1, LATERAL_CONTROL2

The simulation step size (SIM.StepSize parameter) is not accessible from ModelDesk. To change this parameter, edit the sim_ini.m function in %PROJECT_ROOT\Simulation.current\IniFiles.

**Replaced Mux blocks with BusCreator**  In the following blocks the Mux block has been replaced by the BusCreator block for creating bus signals.

- STABILIZATION
- CRANK_SHAFT
- CENTRAL_DIFFERENTIAL
- FRONT_DIFFERENTIAL
- REAR_DIFFERENTIAL
- SHAFT_CR, SHAFT_CF, SHAFT_FL, SHAFT_FR, SHAFT_RL, SHAFT_RR
- TIRE_MODEL_TMEASY_FL, TIRE_MODEL_TMEASY_FR, TIRE_MODEL_TMEASY_RL, TIRE_MODEL_TMEASY_RR
- TIRE_MODEL_MF_FL, TIRE_MODEL_MF_FR, TIRE_MODEL_MF_RL, TIRE_MODEL_MF_RR
Correction of unit of damping parameter  The unit of the damping parameter has been corrected from [Nm rad/s] to [Nm(rad/s)] in the following blocks:

- CLUTCH
- GEARBOX_MT
- GEARBOX_AT
- LOCKUP_CLUTCH
- TORQUE_CONVERTER
- TRANSMISSION_TO_STABILIZATION
- STABILIZATION
- SHAFT_CR, SHAFT_CF, SHAFT_FL, SHAFT_FR, SHAFT_RL, SHAFT_RR
- CENTRAL_DIFFERENTIAL, FRONT_DIFFERENTIAL, REAR_DIFFERENTIAL

Engine Basic

THROTTLE block  The AngleThrottle[deg] variable has been replaced by Pos_AccPedal[%]. The related input and output ports, labels and have been changed. During the migration process AngleThrottle[deg] is replaced by Pos_AccPedal_out[%] in all bus selector blocks. The Const_Max_ThrottleAngel[deg] parameter is now unnecessary and has been removed.

ESP_TORQUE_INTERVENTION_SLOW block  The AngleThrottle[deg] variable has been replaced by Pos_AccPedal[%]. The related input and output ports, labels and parameter maps have been changed. During the migration process AngleThrottle_ESPInfluenced[deg] is replaced by Pos_AccPedal_ESPInfluenced [%] in all bus selector blocks. The Map_Throttle parameter has been replaced with Map_Trq_Engine_Inv.

ENGINE_BASIC block  The AngleThrottle[deg] variable has been replaced by Pos_AccPedal[%]. The related input port and parameter maps have been changed. The new variable is more suitable for the simulation of diesel and gasoline engines. The FristOrderDynamics block has been renamed FirstOrderDynamics.

SoftECU

SOFTWARE_TRANSMISSION block  The shift tables were renamed from Map_AccPedal_downshiftFactor to Map_omega_DownShift and from Map_AccPedal_upshiftFactor to Map_omega_UpShift. In addition the dimension of the shift tables was increased to handle up to 25 gears. The ASM_SHIFTTABLE_LOOKUP block from the ASM Utilities library is used to evaluate the shift tables. This solves the problems that occur with the multiple use of local mask variables.
The Map_Stiffness parameter has been renamed to Map_Stiffness.

The new inertia map (Map_Inertia_Out) is initialized with default values. To achieve the original simulation result it has to be changed to:

\[
\text{Map\_Inertia\_Out}(\text{Gear}) = \text{Const\_Inertia\_Output}\text{Gear} + \text{Map\_Gear\_Ratio}(\text{Gear})^2 \times \text{Const\_Inertia\_Output}\text{Gear}
\]

The new default parameterization already fulfills this equation with the old default parameterization.

The new transmission efficiencies (Map_Forward_Efficiency, Map_Reverse_Efficiency) are initialized with default values. To achieve the original simulation result they have to be changed to:

\[
\text{Map\_Forward\_Efficiency}(\text{Gear}) = \text{Const\_Transmission\_Efficiency};
\]

\[
\text{Map\_Reverse\_Efficiency}(\text{Gear}) = \text{Const\_Transmission\_Efficiency};
\]

The change of friction torque according to speed difference in the lock clutch has been put in the block’s mask as Const_Coef_DynamicClutchTrq. The default value is equal to the old model value. The inner gear ratio parameter has been put in the block’s mask as Const_i_InnerGear. The default value is set to match the old implementation. The default value of Sw_DifferentialLock_Mode is set so that the two output shafts are connected. This matches the old implementation.

As the new implementation of the vehicle’s aerodynamics changed a lot and the parameters cannot be migrated automatically, the link to the aerodynamics library is changed to the former implementation (‘FormerVersion/AERODYNAMICS_1_0’ subsystem) during the migration of older ASM models. Thus, the simulation behavior is not changed. If you want to use the new aerodynamics calculation, just drag and drop the AERODYNAMICS block from the ASM_VehicleDynamics_lib Simulink library to your model and adapt the new parameters (mainly the aerodynamics coefficients as functions of the angle of incidence) to your needs. (See also notes on the aerodynamics subsystem in New Features of ASM Vehicle Dynamics Blockset 1.2 on page 31)

The tags at the from-blocks for the inertias of all wheels were corrected to FL, FR, RL and RR. Before, all inertias had been taken from the front left wheel.
STEERING block  The Const_Angel_Max_SteeringWheel parameter has been renamed Const_Angle_Max_SteeringWheel. (See also notes on STEERING in New Features of ASM Vehicle Dynamics Blockset 1.2 on page 31)

SUSPENSION_FORCES_FRONT block  The outport names have been changed from F_FL/FR_Damper_dDispl_FL/FR_Damper[Nl(mls)] to F_FL/FR_Damper_dDispl_FL/FR_Damper_dt[Nl(mls)]. The related signal labels have been changed accordingly at ASMSignalBus.

SUSPENSION_FORCES_REAR block  The outport names have been changed from F_RL/RR_Damper_dDispl_RL/RR_Damper[Nl(mls)] to F_RL/RR_Damper_dDispl_RL/RR_Damper_dt[Nl(mls)]. The related signal labels have been changed accordingly at ASMSignalBus.

SUSPENSION_FORCE_KINEMATICS_FRONT block  The inport names have been changed from F_FL/FR_Damper_dDispl_FL/FR_Damper[Nl(mls)] to F_FL/FR_Damper_dDispl_FL/FR_Damper_dt[Nl(mls)].

SUSPENSION_FORCE_KINEMATICS_REAR block  The inport names have been changed from F_RL/RR_Damper_dDispl_RL/RR_Damper[Nl(mls)] to F_RL/RR_Damper_dDispl_RL/RR_Damper_dt[Nl(mls)].

WHEEL_SPEED block  The calculation of the wheel speeds is now implemented as a vector signal.

VEHICLE_MOTION_CAR block  During migration, the new external forces and torques inports are connected automatically to constant zero vectors and matrices.
## ControlDesk

### New Features of ControlDesk 3.2.2

<table>
<thead>
<tr>
<th>ControlDesk Failure Simulation 1.5</th>
<th>ControlDesk’s Failure Simulation component supports the DS1450 Bus FIU Board. The board provides a failure insertion unit (FIU) for differential bus systems such as CAN and FlexRay. It has been designed to insert electrical errors on the physical layer of the bus system. The DS1450 supports four differential bus channels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ControlDesk DS2302-3 support</td>
<td>ControlDesk supports the DS2302 Direct Digital Synthesis Board with a revision of DS2302-4.</td>
</tr>
</tbody>
</table>
# dSPACE Data Dictionary

Where to go from here

<table>
<thead>
<tr>
<th>Information in this section</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Migrating to dSPACE Data Dictionary 1.5.1</strong></td>
<td>42</td>
</tr>
<tr>
<td><strong>How to Upgrade a Data Dictionary Without Include Files</strong></td>
<td>45</td>
</tr>
<tr>
<td><strong>How to Upgrade a Data Dictionary With Include Files</strong></td>
<td>46</td>
</tr>
</tbody>
</table>
Migrating to dSPACE Data Dictionary 1.5.1

Upgrading Data Dictionary files

When you upgrade existing TargetLink models to TargetLink Version 3.0, you also need to upgrade the associated DD files to dSPACE Data Dictionary Version 1.5.1 (this belongs to TargetLink 3.0).

The upgrade routine in TargetLink 3.0 and dSPACE Data Dictionary 1.5.1 upgrades only models and Data Dictionary files created under a TargetLink 2.x version (and the associated Data Dictionary versions). Projects that were created under TargetLink 1.3 or even older versions cannot be upgraded directly to TargetLink 3.0 and dSPACE Data Dictionary 1.5.1. You must first perform an upgrade to a TargetLink 2.x version (including the associated Data Dictionary version) before you can upgrade to TargetLink 3.0 and dSPACE Data Dictionary 1.5.1.

For last minute information regarding TargetLink 3.0 as well as potential difficulties regarding model upgrades, you are recommended to visit the TargetLink 3.0 website at http://www.dspace.de/goto?tl30.

Method to upgrade Data Dictionary files

dSPACE Data Dictionary 1.5.1 provides an upgrade routine that automatically upgrades older DD files to Version 1.5.1.

The upgrade routine can be called in three ways:

- Automatically opening an old TargetLink model.

When you open a TargetLink model with an old (not upgraded) DD file, a model callback function first automatically runs the dSPACE Data Dictionary’s upgrade routine.
Via the Tools menu in the Data Dictionary Manager
Manually via Tools -- Upgrade Current DD in the DD Manager.

Data Dictionary API command
To call the upgrade routine via the Data Dictionary's API, type \texttt{dsdd('Upgrade')} in the MATLAB Command Window.

Preconditions when upgrading Data Dictionary files
To ensure the DD file is upgraded correctly, the following preconditions must be met:

- There must be write permission for the DD file and the file must not be write-protected. If Data Dictionary include files are used, there must be write access to all the included files.
If Data Dictionary Include files are used, the included files must be saved after updating as well as the main DD file. This ensures that the Data Dictionary partitions into several files. To save the include files correctly, you must first make the appropriate settings for them.

Related topics

Basics
- How to Upgrade a Data Dictionary With Include Files on page 46
- How to Upgrade a Data Dictionary Without Include Files on page 45
- Model Upgrade on page 82
How to Upgrade a Data Dictionary Without Include Files

Objective
If you open a TargetLink model with an old, nonupgraded Data Dictionary file, you have to upgrade the Data Dictionary file.

Method
To upgrade a Data Dictionary without include files

1. Open the model and therefore the referenced dSPACE Data Dictionary, or type `dsdd('Open',<DDFile>)` in the MATLAB Command Window.

   The Data Dictionary needs upgrading dialog automatically opens if an older DD version is involved.

   ![Data Dictionary needs upgrading dialog](image.png)

   2. Click Yes if no Include files are used in the Data Dictionary.

      If Include files are used, refer to *How to Upgrade a Data Dictionary With Include Files* on page 46.

3. Save the data dictionary.

   When you have saved the Data Dictionary (assuming you have write privileges to the relevant DD file), the upgrade of the DD file is completed.

Result
The next time you open the DD file, the upgrade dialog will not open because the DD file is up-to-date.
How to Upgrade a Data Dictionary With Include Files

Objective

If you open a TargetLink model with an old, nonupgraded Data Dictionary file, you have to upgrade the Data Dictionary file.

Method

To upgrade a Data Dictionary with include files

1. Open the model and therefore the referenced dSPACE Data Dictionary, or type `dsdd('Open',<DDFile>)` in the MATLAB Command Window.

   The Data Dictionary needs upgrading dialog automatically opens if an older DD version is involved.

   ![Data Dictionary needs upgrading](image)

   - The DataModel version number of your Data Dictionary specifies an older DataModel version (132).
   - You are therefore recommended to upgrade your DataDictionary.
   - Note that after an upgrade, the DD might not work with previous versions of your software. For example, you will not be able to generate code with previous TargetLink versions.

   Do you wish to have your DD upgraded?

   - Yes
   - No

2. Select No in the upgrade dialog.

3. Under /Config/DDIncludeFiles, set the AutoLoad and AutoSave properties for each Include file as shown in the illustration below.
This ensures that after the Data Dictionary and the Include files have been upgraded, the upgraded include files are saved when the Data Dictionary is saved. You can set these properties for a large number of Include files via the Object Explorer.

4 Start the DD upgrade (including the included files) via Tools - Upgrade Current DD in the DD Manager, or enter `dsdd('Upgrade')` in the MATLAB Command Window.

5 Save the data dictionary. When you have saved the Data Dictionary (assuming you have write privileges to the relevant DD file), the upgrade of the DD file itself as well as the included DD files is completed.

Result

The next time you open the DD file, the upgrade dialog will not open because the DD file is up-to-date and so are the included Data Dictionary files. After the files have been properly upgraded, you might want to restore the old settings for the Data Dictionary include files.
dSPACE FlexRay Configuration Package

New Features of dSPACE FlexRay Configuration Package 1.10

<table>
<thead>
<tr>
<th>FlexRay Configuration Tool</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC template file</td>
<td>The FlexRay Configuration Tool can create a template in which you can implement your checksum algorithms. For details, refer to [Implementing Checksum Algorithms](FlexRay Configuration Tool Guide).</td>
</tr>
<tr>
<td>CRC settings import/export</td>
<td>The settings of the CRC Configuration dialog can be exported to a file and imported in another configuration project to use the same checksum algorithms in several configuration projects. For details, refer to [How to Use the Same Checksums Algorithm in Several Configuration Projects](FlexRay Configuration Tool Guide).</td>
</tr>
<tr>
<td>Path of FIBEX file</td>
<td>You can save the path of the FIBEX file as a complete, absolute path or as a path relative to the working folder. For details, refer to [Save Relative FIBEX Path](FlexRay Configuration Tool Reference).</td>
</tr>
<tr>
<td>Extended TRC file</td>
<td>The generated TRC file contains all the signals and raw data which are monitored, received or sent. To visualize them, you can build a data connection with ControlDesk instruments. For details, refer to [Using the Generated TRC File in ControlDesk](FlexRay Configuration Features).</td>
</tr>
</tbody>
</table>
Payload length manipulation  To manipulate the payload length, you can read and write the value in the frames. For details, refer to Manipulating or Reading the Payload Length of a Frame (FlexRay Configuration Features).

Application task blocks  For application task blocks, function-called triggered subsystems are created in the configuration process.

Update bit manipulation  To manipulate the update bit, you can set or get the update bit of FlexRay frames. For details, refer to How to Manipulate Update Bits (FlexRay Configuration Features).
New Features of Model Compare 2.0

Merging models
You can eliminate differences between the reference and comparison models by merging the two models as follows:
- Copy model parts or property values from one model to the other.
- Delete model parts that exist only in one model.

Reloading models
If a model has been changed, for example, due to merge operation, it is not consistent with the related dump file. You can redump the model by reloading it, hence a new comparison is launched.
Model Compare 2.0 offers enhanced options for filtering differences:

**Ignore meta data changes**  This is a new pre-defined filter option, refer to Basic Filters Page (Model Compare Reference). This filter hides changes in meta data, for example, LastModifiedDate or ModelVersion.

**Property filter**  This is a helpful complement to the pre-defined filter options. If you are not interested in a specific kind of differences, for example, ones relating to specific TargetLink properties, you can build a filter that filters out the irrelevant properties.

You can create PDF/HTML reports according to your own layout rules. Model Compare lets you generate XML reports which you can transform into PDF or HTML by applying XLS style sheets. You must have installed an appropriate converter program (XML →PDF/HTML).

A list of the four most recently used sessions is displayed in the File menu. You can click a session to open it.

Model Compare 2.0 is fully compatible with TargetLink 3.0.

### Migration to Model Compare 2.0

**Session and dump files**  Session files as well as dump files created by Model Compare 1.0 can be processed by Model Compare 2.0.

**Comparison results**  Model Compare 2.0 comes with changes in the display of properties and the comparison algorithm, hence comparisons performed with Model Compare 2.0 may lead to slightly different results in comparison to Model Compare 1.0.
### User preferences and comparison settings

User preferences and comparison settings are stored separately from the tool. Thus, Model Compare 2.0 does not make use of the user preferences and comparison settings previously specified with Model Compare 1.0, but uses the defaults. However, user preferences and comparison settings that have been exported with Model Compare 1.0 can be imported by Model Compare 2.0.

Model Compare 2.0 lets you specify more user preferences and comparison settings as this is possible with Model Compare 1.0.
ModelDesk

Where to go from here

<table>
<thead>
<tr>
<th>Information in this section</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>New Features of ModelDesk 2.1</strong></td>
</tr>
<tr>
<td><strong>Migration to ModelDesk 2.1</strong></td>
</tr>
</tbody>
</table>

New Features of ModelDesk 2.1

Initial road height and slope angle

In the Road Generator, you can define the initial height and slope angle of the road to easily create complex road profiles.

Migration to ModelDesk 2.1

Migrating projects and experiments created with ModelDesk 2.0 or earlier

Projects created with ModelDesk 2.0 or earlier cannot be used with ModelDesk 2.1. You must therefore migrate these projects and their project data and update the model information. The migration is described in *Migrating a Project and Its Experiments* ([ModelDesk Guide](#)).
RTI

New Features of RTI 6.1

The DS2211 RTI blockset and the DS2211 RTLib functions were improved:

Reverse crank sensor signal  If you need to simulate a crankshaft sensor which detects the rotation direction, for example when simulating the start-stop mechanism of a motor vehicle, you can do so by selecting a special wave table and simulating a reverse crank sensor signal. Refer to Reverse Crankshaft Rotation (DS2211 Features).

Smooth camshaft phase update  You can enable smooth camshaft phase updating to suppress undesired spikes in phase offset updates.

Compressed wave tables  Crankshaft and camshaft wave tables are now stored and loaded in compressed form to reduce their file sizes and enhance system performance.
RTI Bypass Blockset

New Features of the RTI Bypass Blockset 2.4.1

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deleting old data dictionaries of models built with RTI Bypass Blockset</td>
<td>The RTI Bypass Blockset now provides several options to delete old, unused data dictionaries of models built with the RTI Bypass Blockset from the file system:</td>
</tr>
<tr>
<td>Deleting old data dictionaries automatically when a modified RTI Bypass model is saved</td>
<td>In the Setup block, you can specify to delete automatically old data dictionary files of a model. With this option enabled, the RTI Bypass Blockset removes the obsolete data dictionaries of the current model directory when a moved/renamed model is saved.</td>
</tr>
<tr>
<td>Deleting old data dictionaries immediately</td>
<td>The RTI Bypass Blockset lets you delete old data dictionaries immediately. Immediate deletion must be started manually in the Setup block. After you start deletion, all the models located in the current working folder are opened sequentially to identify the unused data dictionaries. When an obsolete data dictionary file is found, it is immediately removed from the file system.</td>
</tr>
<tr>
<td>New configuration option for CCP interfaces</td>
<td>CCP command timeout</td>
</tr>
<tr>
<td>New configuration option for XCP on UDP/IP interfaces</td>
<td>Send one ODT per Ethernet frame</td>
</tr>
</tbody>
</table>
Working with models from RTI Bypass Blockset versions 2.0, 2.1 and 2.2

dSPACE Release 6.2 comes with RTI Bypass Blockset 2.4.1, which is compatible with earlier blockset versions 2.x. However, the format of the data dictionary was changed in comparison to RTI Bypass Blockset Versions 2.0, 2.1 and 2.2. The data dictionaries of Simulink models built with blockset Versions 2.0, 2.1 and 2.2 are automatically converted to the format used by RTI Bypass Blockset 2.4.1.

Limitation when using the Variable Editor

You cannot open the Variable Editor via the RTIBYPASS_SETUP block of the RTI Bypass Blockset if you have installed CalDesk 2.0. You can open the Variable Editor via the RTIBYPASS_SETUP block only if you have installed CalDesk 1.4.1 or earlier.

If you have a model that was saved with RTI Bypass Blockset 2.4.1 and want to use it with an earlier 2.x version of the RTI Bypass Blockset, you must first delete the model's data dictionary (the name and path of the data dictionary can be found in the Info block) and import the ASAM-MCD 2MC (A2L) file again. The RTI Bypass Blockset then automatically creates a data dictionary according to the appropriate format.
New Features of the RTI CAN MultiMessage Blockset 2.3

New counter modes

The RTI CAN MultiMessage Blockset now supports different counter modes. You can choose three different counter modes for signals which are not transmitted with their counter value but with a constant or parity signal value, for example:

- You can let the counter stop counting.
- You can let the counter continue counting internally.
- When the signal is transmitted with its counter value again, you can let the counter start counting with the current signal value.

If you switch from a gateway signal value to a counter signal value, the counter always starts counting with the current gateway signal value. For details, refer to Counter Page (RTICANMM MainBlock) (RTI CAN MultiMessage Reference).

Modified TRC file and output port bus structure

You can now specify an ECU hierarchy for the TRC file and the output port bus structure. In this case, ECU names are the topmost elements in the structures. For details, refer to TRC Options Page (RTICANMM MainBlock) (RTI CAN MultiMessage Reference) and Peripheral Options Page (RTICANMM MainBlock) (RTI CAN MultiMessage Reference), respectively.
New Features and Migration July 2008

SystemDesk

Where to go from here

Information in this section

New Features of SystemDesk 1.1  63
Migration to SystemDesk 1.1  65

New Features of SystemDesk 1.1

Import and export of AUTOSAR 2.1 files

You can now import AUTOSAR XML files according to AUTOSAR Version 2.1. No additional AUTOSAR elements are supported compared with the last version of SystemDesk. During import, AUTOSAR elements that have been introduced since the release of AUTOSAR Version 2.0 are ignored.

You can now export AUTOSAR XML files according to AUTOSAR version 2.1. When exporting AUTOSAR files, you can choose between AUTOSAR Versions 2.0 and 2.1. You can specify AUTOSAR version 2.1 export options for base type references that allow the seamless integration of dSPACE’s TargetLink and third-party tools with SystemDesk.
### Importing LDF files

To specify network communication, you can now import an existing LIN description (LDF) file containing network nodes, messages and signals into a communication matrix. Refer to *How to Add and Specify a Communication Matrix* ([SystemDesk Guide](#)).

To map the network communication defined in an LDF file to a hardware topology, SystemDesk also lets you specify Local Interconnect Network (LIN) buses and LIN ECU communication ports. Refer to *How to Add and Specify a Bus* ([SystemDesk Guide](#)) and *How to Add and Specify an ECU* ([SystemDesk Guide](#)).

### Linking SystemDesk to third-party tools

Each SystemDesk element is now accessible via a uniform resource locator (URL). You can copy an element’s URL in SystemDesk to the Clipboard and paste it to other applications like Telelogic's DOORS or Microsoft's Internet Explorer. You can then focus a SystemDesk element from within the other application by selecting its URL. SystemDesk is opened if necessary.

### More flexible element names

You now have greater flexibility in specifying some of SystemDesk’s element names. When you generate RTE code, the names of SystemDesk elements such as the port element are among the generated RTE API function names. You can now enter 128 character names instead of 32 character names. However, if your compiler has different limitations for function names, for example, a length limit of 31 characters, you must comply with those limitations.

### Exporting data on connections between application software components and AUTOSAR services

Modeling AUTOSAR EcuSwComposition elements according to AUTOSAR Version 2.1 is now supported. You can use SystemDesk to model these elements as software architectures. In these software architectures, you can specify AUTOSAR service components with service ports and connect these ports with application software component ports. If you export the modeled AUTOSAR service components in a package together with the corresponding application software components, the connections between services and applications are written to the exported AUTOSAR file(s).
Migration to SystemDesk 1.1

Automating SystemDesk if you have installed SystemDesk 1.0 and 1.1

If you have two SystemDesk versions installed, you can initially automate only the version you installed last. If you want to automate both of them, you must first re-register (in the Windows registry) the automation feature of the SystemDesk version that you installed first.

For example, if you first installed SystemDesk Version 1.0 and then installed Version 1.1, you have to register for automating SystemDesk Version 1.0.

To register for automating SystemDesk, open a command window, navigate to the \./bin folder of your SystemDesk installation and type:

```
SystemDesk -register
```

SystemDesk registers for automating SystemDesk 1.0. You can automate SystemDesk 1.0 by calling `Dispatch("SystemDesk.Application")` in your automation script. If you want to automate SystemDesk 1.1 you have to call `Dispatch("SystemDesk.Application.1.1")` in your script for automating SystemDesk 1.1.

Automating SystemDesk after you have deinstalled another version of SystemDesk

If you uninstall one SystemDesk version and want to automate another SystemDesk version that is still installed, you have to re-register the installed version in the Windows registry. For example, if you have installed SystemDesk Versions 1.0 and 1.1 and deinstalled Version 1.0, you have to re-register the automation feature of SystemDesk Version 1.1. To register for automating SystemDesk open a command window, navigate to the \./bin folder of your SystemDesk installation and type:

```
SystemDesk -register
```

SystemDesk registers for automating SystemDesk 1.1.
# TargetLink

<table>
<thead>
<tr>
<th>Where to go from here</th>
<th>Information in this section</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>New Features of TargetLink 3.0</strong></td>
<td>68</td>
</tr>
<tr>
<td><strong>Migrating Existing Models and Data Dictionary Files</strong></td>
<td>81</td>
</tr>
</tbody>
</table>
New Features of TargetLink 3.0

<table>
<thead>
<tr>
<th>Where to go from here</th>
<th>Information in this section</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Blockset Features</td>
<td>68</td>
</tr>
<tr>
<td>Model Referencing</td>
<td>77</td>
</tr>
<tr>
<td>Miscellaneous Features</td>
<td>79</td>
</tr>
</tbody>
</table>

New Blockset Features

Objective
Information on new blockset features of TargetLink 3.0 is provided below.

<table>
<thead>
<tr>
<th>Where to go from here</th>
<th>Information in this section</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Blockset Features</td>
<td>68</td>
</tr>
<tr>
<td>Signal Width Inheritance</td>
<td>76</td>
</tr>
</tbody>
</table>

New Blockset Features

Introduction to the new blockset design
TargetLink 3.0 has a redesigned blockset that provides closer integration with Simulink. TargetLink-specific S-functions used in previous TargetLink versions have been replaced with masked Simulink blocks. The blocks still provide special TargetLink functionality such as dedicated TargetLink block dialogs and integrated logging and overflow checking. The modified block implementation provides
additional benefits, although TargetLink models mostly look identical to previous versions. Using masked Simulink blocks ensures equivalent simulation behavior in all model-in-the-loop (MIL) simulation modes, i.e., Simulink MIL, TargetLink full-featured MIL and TargetLink stand-alone MIL.

In addition, TargetLink ports have been merged with Simulink inports/outports to replace the TargetLink InPorts/OutPorts used in earlier TargetLink versions (see illustration below).

Since TargetLink blocks are now masked Simulink blocks, they inherit all the properties of the underlying Simulink blocks. Simulink properties can be accessed via Simulink block dialogs. Dedicated TargetLink block properties such as variable classes are still available through TargetLink block dialogs and are stored in a mask parameter.
The TargetLink blockset was redesigned for closer integration into the MATLAB/Simulink world, at the same time preserving the blockset's special features such as convenient user interfaces for specifying code generation settings and overflow checks. TargetLink 3.0 has the following advantages:

- Numerous third-party tools that are available for Simulink® models can also be used with TargetLink models. This applies to both the TargetLink full-featured mode and the TargetLink stand-alone mode. The tools concerned include toolboxes and extensions from The MathWorks®, and third-party tools relating to Simulink such as test vector generation tools. The integration of TargetLink models into dSPACE’s own products such as RTI and ControlDesk has also been improved. Prototyping code can be generated with the Real-Time Workshop® in TargetLink 3.0’s full-featured mode and stand-alone mode.

- It is easier to transfer Simulink models to the TargetLink world. Simulink-to-TargetLink conversion has been replaced by a preparation process, which takes only a fraction of the time that was needed by conversion. You also have numerous new options for specifying how to map/synchronize Simulink properties such as Simulink data types to/with the corresponding TargetLink properties. Simulink blocks are no longer replaced by their TargetLink counterparts. To prepare Simulink blocks for use with TargetLink, you can enhance them with a mask and a TargetLink block dialog.

- The simulation behavior of TargetLink and Simulink blocks is identical. Occasional differences in behavior no longer exist, for example, previous differences between data types used in MIL simulation for Simulink models, TargetLink full-featured models, and TargetLink stand-alone models. Simulink blocks and corresponding TargetLink blocks always behave identically. The TargetLink stand-alone mode is still available. It differs from the full-featured mode with regard to license, i.e., the stand-alone mode does not provide the entire TargetLink functionality.

- With regard to performance aspects, working with TargetLink is faster in several ways:
  - Model loading times have been reduced.
  - Model initialization times have been reduced.
  - MIL simulations generally run faster (refer to Logging speed in MIL simulation mode in Logging Behavior Changes on page 98).
System preparation

System preparation (replacing the former \texttt{sl2tl} conversion) is used to prepare Simulink models, mixed models consisting of TargetLink and Simulink blocks, user libraries or referenced models for code generation with TargetLink. System preparation in general is much faster than the former conversion process and provides a lot more flexibility for synchronizing Simulink and TargetLink properties like data types and others. The system preparation dialog shown below provides several checkboxes to specify precisely which properties are to be synchronized between Simulink and TargetLink. In addition, user-specific M scripts can be used to specify how Simulink and TargetLink properties are synchronized. This mechanism is easier to use than the hook functions of the former conversion. Synchronization can also be performed independently of system preparation at any time using the system synchronization dialog. Hence, with TargetLink 3.0, you now have full control of how system preparation is performed. System preparation can be invoked via dialog or via the \texttt{tl\_prepare\_system} API command. For details on system preparation, refer to Basics on System Preparation (\textit{TargetLink Production Code Generation Guide}).
After system preparation has finished, you can generate production code and use TargetLink’s logging and plotting capabilities. Since Simulink blocks are only prepared and not completely replaced as was previously the case, all Simulink properties are preserved.

### Clearing TargetLink data from system

System clearance (replacing the former \texttt{tl2sl} reconversion) removes TargetLink data from a TargetLink model. System clearance always includes the synchronization of TargetLink and Simulink properties. As with system preparation, you can now specify precisely which properties are to be synchronized from TargetLink to Simulink and how this is carried out. Specifications are made via checkboxes in the Clear System from TargetLink dialog as well as in user-defined M scripts. This mechanism is easier to use than the hook functions of the former reconversion. Hence, with TargetLink 3.0, there is a lot more flexibility than with the former \texttt{tl2sl} conversion. Synchronization can also be performed independently of system clearance at any time. System clearance can be invoked via dialog or via the \texttt{tl_clear_system} API command. For details on system clearance, refer to \textit{Reversion of System Preparation} (\textit{TargetLink Production Code Generation Guide}) and to \textit{How to Clear All TargetLink Data} (\textit{TargetLink Production Code Generation Guide}).
TargetLink 3.0 provides the opportunity to synchronize Simulink and TargetLink properties not just during system preparation and system clearance but at any time. This is useful because some properties in Simulink on one hand and TargetLink on the other hand have similar but not identical semantics (for example, data types in Simulink for MIL simulation, data types in TargetLink for code generation). System synchronization provides the means to synchronize such properties in both directions. Note that this does not apply to properties such as the gain value of a Gain block, which only exists as a Simulink property but can be accessed from TargetLink block dialogs and the TargetLink API.

The System Synchronization dialog is used for data synchronization, i.e., to select the properties to be synchronized and the direction of synchronization (Simulink to TargetLink or TargetLink to Simulink). User-specific M scripts are used to specify precisely how synchronization, e.g., between Simulink and TargetLink data types, is carried out. Whereas with previous TargetLink versions, synchronization was called implicitly during sl2tl and tl2sl conversion, it can now be carried out at any time in a very flexible manner. Synchronization can be invoked via block dialog or the tl_sync_system API command. For details on system synchronization, refer to Synchronization of Simulink and TargetLink Data (TargetLink Production Code Generation Guide).
Removing the TargetLink subsystem

For better integration with other tools, TargetLink now provides the means to remove the TargetLink Subsystem block. It is sometimes useful to do this to preserve identical hierarchies and blockpaths in both the original Simulink model and the TargetLink model. Removal of the TargetLink Subsystem block is invoked via the API function `tl_clear_system` or the Clear System from TargetLink dialog. The TargetLink Subsystem block is of course mandatory for SIL/PIL simulations since it contains the TargetLink simulation frame, which
enables you to switch between the MIL and SIL/PIL simulation modes. The TargetLink Subsystem block is reinserted upon system preparation. For details on removing the TargetLink simulation frame, refer to How to Remove TargetLink Simulation Frames (TargetLink Production Code Generation Guide).

Using the Simulink API

With the TargetLink blocks implemented as masked Simulink blocks you can now also set and read the pure simulation properties (such as the Gain value of a TargetLink Gain block) via the Simulink API. You can now use the Simulink API command `set_param(gcah, 'Gain', '5')` to set the Gain value of a TargetLink block. For reasons of downward compatibility, you can also set this property via the appropriate TargetLink API command, `tl_set(gcah, 'gain.value', '5')`.

Related topics

Basics
- Migration Aspects for Former SL2TL and TL2SL Conversion (System Preparation and System Clearance) on page 87
- Preparing Simulink Systems for TargetLink
Signal Width Inheritance

Scalar expansion

It is now no longer necessary to specify a signal width in the TargetLink block dialogs explicitly. Instead, Scalar Expansion (see illustration below) can be used to automatically inherit the width information from Simulink.

This not only reduces the amount of specification work, but also lets you use width-independent library blocks. The precondition for using Scalar expansion is that there are no differences between the scaling, saturation, and min/max values of the individual components in a vector. If the checkbox is selected, the values specified for LSB, Offset, Min/Max: and Saturate are applied to all the components of a vector.

Related topics

Basics
- Scalar Expansion and Block Output Width on page 95
Model Referencing

New Features of Model Referencing

Support of model referencing

Simulink model referencing is a mechanism that lets you include models in other models by using Model blocks. Each Model block represents a reference to another model and displays the inputs and outputs of the referenced model. The referenced model is not loaded until it is needed, which makes this feature particularly suitable for handling very large models. TargetLink uses the original Simulink block for model referencing. There is no corresponding TargetLink Model block. The following example shows a Model block that represents a reference to a model with one input and one output port respectively.

For details, refer to Basics of Model Referencing (TargetLink Production Code Generation Guide).

Code generation for referenced models

The Code Generator in TargetLink handles referenced models in a similar way to subsystems configured for incremental code generation. The difference is that the referenced models must only be open when production code is generated for them. They do not need to be open while production code is generated for the surrounding system, since the information on the interface for the referenced models is not read directly from the model, but from the data dictionary output created for them during production code generation. This ensures that the only system loaded is the one that the production code is generated for.

Logging of signals in referenced models

There are additional options for logging signals in referenced models. You can:
- Enable logging in all referenced models
- Enable logging for selected referenced models
- Disable logging in all referenced models
No signals are logged for a referenced model in MIL simulation unless logging is explicitly enabled for that model, regardless of the local logging options set for the single signal and the global logging option set in the TargetLink Main Dialog. These additional logging options do not apply to SIL/PIL simulation mode.

To log signals in referenced models during MIL simulation mode, you must start the simulation by a TargetLink-specific API command or via the Start simulation button of the model Referencing Control Center. For details, refer to Logging Behavior Changes on page 98.

### Enabling/Disabling the model reference

You can work either with subsystems configured for incremental code generation or with referenced models, depending on the development phase. TargetLink offers tools to disable the model reference (by converting the referenced model into a subsystem configured for incremental code generation) or enable the model reference (by converting the subsystem configured for incremental code generation into a referenced model).
Miscellaneous Features

Revert button replaced by a toolbar icon

The **Revert** button in the TargetLink dialogs has been replaced by a **Refresh** icon in the toolbar. When you click the **Refresh** icon, the block data is reread and displayed in the appropriate fields of the TargetLink block dialog. The data from the block dialog is not written to the actual block until you click the **Apply** or **Close** button. Before you do that, you can always restore the original block data in the TargetLink dialog by clicking the **Refresh** icon. Changes made to DD variables in the dSPACE Data Dictionary are also refreshed when you click the **Refresh** icon.
Logging bus signals via Sink block

With TargetLink 3.0, you can use the TargetLink Sink block to log whole buses. The TargetLink Sink block is bus-capable, i.e., it can be addressed directly by a bus in order to log bus signals. If the bus is nested, all the signals in the bus are logged and designated by their signal names.

Overflow checking for virtual blocks

TargetLink 3.0 supports overflow checks on virtual blocks, for example, for Simulink ports that were prepared for TargetLink. This was not the case with TargetLink 2.x, where this depended on how the block was configured.
Migrating Existing Models and Data Dictionary Files

Upgrade routine

To update existing TargetLink models and libraries to TargetLink Version 3.0, TargetLink provides an update routine. In addition to models and libraries, the associated DD files must also be updated. This is done by the DD update routine.

For last minute information regarding TargetLink 3.0 as well as potential difficulties regarding model upgrades, you are recommend to visit the TargetLink 3.0 website at http://www.dspace.de/goto?tl30.

The update routine in TargetLink 3.0 and dSPACE Data Dictionary 1.5.1 updates only models and Data Dictionary files created under a TargetLink 2.x version (and the associated Data Dictionary versions). Projects that were created under TargetLink 1.3 or even older versions cannot be updated directly to TargetLink 3.0 and dSPACE Data Dictionary 1.5.1. You must first perform an update to a TargetLink 2.x version (including the associated Data Dictionary version) before you can update to TargetLink 3.0 and dSPACE Data Dictionary 1.5.1.

Where to go from here

Information in this section

| Model Upgrade                  | 82 |
| How to Upgrade Models and Libraries | 83 |
| Migration Aspects for Former SL2TL and TL2SL Conversion (System Preparation and System Clearance) | 87 |
| TargetLink Port Blocks Discontinued | 93 |
| Modified TargetLink Blocks | 94 |
| Scalar Expansion and Block Output Width | 95 |
| Replacement for TargetLink Port Blocks Used for Rescaling | 97 |
| Logging Behavior Changes | 98 |
| Simulation Behavior Changes | 106 |
| Code-Relevant Stateflow Enhancements and Changes | 108 |
Model Upgrade

Upgrading TargetLink models

When you upgrade existing TargetLink models to TargetLink Version 3.0, you also need to upgrade the associated DD files to dSPACE Data Dictionary Version 1.5.1 (this belongs to TargetLink 3.0). The upgrade routine for TargetLink 3.0 automatically removes the contained TargetLink ports, prepares the corresponding Simulink ports, and writes the data of the original TargetLink ports to the Simulink ports. There is also an option to replace the name of the Simulink port by the name of the deleted TargetLink port during the model upgrade. This is particularly helpful if you built custom library blocks and data was written to the TargetLink ports via mask initialization code data, accessing the TargetLink ports by their names.

Adapting custom libraries and scripts

Because there are no more TargetLink ports as independent blocks, it may be necessary to modify the custom libraries and scripts in a tool chain. It can happen that TargetLink ports that are accessed by name no longer exist (depending on the upgrade option). The removal of TargetLink ports can occasionally cause problems with label...
propagation. If custom scripts in a tool chain insert TargetLink ports automatically, modifications also have to be made there, since when the ports are taken from the tllib, prepared Simulink ports are now inserted. For details, refer to How to Upgrade Models and Libraries on page 83.

### Unset dirty flag after upgrade
The first time a TargetLink model is opened with Version 3.0, it is modified in nearly all cases. However, this is not indicated by the dirty flag being set.

### Related topics
- Basics
- How to Upgrade Models and Libraries on page 83
- Migrating to dSPACE Data Dictionary 1.5.1 on page 42

## How to Upgrade Models and Libraries

### Objective
If you open an older model, the upgrade routine is called automatically (by a model callback function, though first the data dictionary upgrade routine runs). However, if you open a library, it is not automatically upgraded, as the MDL file does not contain callbacks for this. You therefore have to initiate the library upgrade explicitly using the API command `tl_upgrade`. To display the options for the command, enter the API command `help tl_upgrade` in the MATLAB Command Window.

The following options apply to TargetLink 3.0. It is always advisable to use the default settings.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>Name of the model or library to be upgraded.</td>
</tr>
<tr>
<td>Upgradelibs</td>
<td>Ensures that the libraries referenced by a model are automatically included in the upgrade. However, it is basically recommended to upgrade all libraries manually first, i.e., not to use this option.</td>
</tr>
<tr>
<td>Rebuildsfcn</td>
<td>Rebuilds all the custom code blocks in a model (including, for example, the Bitwise Logical Operator block). This option is automatically enabled in TargetLink 3.0 and should be used here.</td>
</tr>
<tr>
<td>useTL2importnames</td>
<td>In TargetLink 3.0, Simulink and TargetLink ports are merged. This option allows you to use the original name of the TargetLink port (which is removed) as a new name for the connected Simulink port. The original name of the Simulink port then no longer exists. This option is useful in cases such as user-built library blocks with mask initialization code that write data to the TargetLink port. In the future, this data has to be written to the Simulink port, as the TargetLink port and the Simulink port were merged.</td>
</tr>
<tr>
<td>Loadmodelonly</td>
<td>The model is not opened explicitly.</td>
</tr>
</tbody>
</table>
**Precondition**

If you have made use of restricted read/write permissions for individual subsystems, you first have to set read/write permission for all the subsystems to be upgraded to ReadWrite. This ensures that the upgrade routine is allowed to carry out the necessary modifications during the upgrade process.
Method

To upgrade models and libraries

1 If you have used restricted read/write permissions for individual subsystems, set the **Read/Write permissions** for all subsystems to be upgraded to "ReadWrite". This makes sure that subsystem modifications during the upgrade process are permitted.

![Function Block Parameters: Subsystem](image1)

For models this must be done before a model is opened with TargetLink 3.0, otherwise the upgrade routine appears. If you have to set multiple properties, you can do this using the Model Explorer.

![Model Explorer](image2)

2 Type `tl_upgrade(<options>)` in the MATLAB Command Window, to upgrade all the Simulink libraries that are used, beginning with libraries that do not themselves reference any other libraries (so the blocks they contain have no links to other libraries).
You do not have to open the file first. The recommended upgrade settings are as follows:
- `upgradelibs= "off"`
- `rebuildsfcn= "on"`

The libraries themselves do not need to be opened exclusively. You have to decide whether you want to use the `TL2xportname` option. This depends on your modeling style. When the upgrade/update has finished, you must save the libraries (under the same names). You can now continue with the libraries that reference only upgraded libraries.

3. After having upgraded all the referenced user libraries, continue with the TargetLink models and upgrade them.

- The data types for MIL simulation or small, individual code fragments might change after the models are updated automatically to TargetLink 3.0. In some cases this can initially cause problems with model initialization, or code that looks slightly different can be generated. You should therefore check the model initialization/model simulation and the code generation performed with the updated models. If a situation arises where a model has to be made initializable again due to changed Simulink data types after model upgrade, there are no generally valid rules for doing so. In most cases, such initialization errors can be avoided by adjusting the initial data types.

For last minute information regarding TargetLink 3.0 as well as potential difficulties regarding model upgrades, you are recommended to visit the TargetLink 3.0 website at http://www.dspace.de/goto?tl30.

Related topics

Basics
- Model Upgrade on page 82
Migration Aspects for Former SL2TL and TL2SL Conversion
(System Preparation and System Clearance)

**System preparation**

System preparation (replacing the former `sl2t1` conversion) is used to prepare Simulink models, mixed models consisting of TargetLink and Simulink blocks, user libraries or referenced models for code generation with TargetLink. System preparation `tl_prepare_system` always includes synchronization of Simulink and TargetLink properties (`tl_sync_system`) but can also be performed independently of system preparation. A simulation frame is inserted for convenient switching between TargetLink MIL/SIL/PIL simulation modes. For details on system preparation, refer to *Basics on System Preparation* (TargetLink Production Code Generation Guide).

Preparation must be performed not only for models but also for libraries; the TargetLink libmap mechanism is still supported. However, there is no longer any separate library conversion. The former `slib2tllib` command no longer exists, and TargetLink no longer generates the libmap files automatically.
You can start system preparation via the **Prepare System** button on the **Tools** page of the **TargetLink Main Dialog**. The API command for calling the dialog is **tl_prepare_system**. To display the individual options, type **help tl_prepare_system** in the MATLAB Command Window. The old API command, **sl2tl** is no longer available.

Some **TargetLink** block mask variables used in previous **TargetLink** versions are no longer available. Where **TargetLink** mask variables were previously accessed directly (for example, if the **TargetLink 2.x gain mask variable** was accessed in the block annotations), adaptations are now necessary, i.e., the corresponding **Simulink** parameters have to be accessed instead.

Most of the **TargetLink** blocks no longer have a library link (the library link is automatically broken) and are given additional block callback functions. This does not apply for the **TargetLink Sink** block and the **TargetLink Unit Delay Reset Enabled** block.

The **TargetLink** properties of a block can still be displayed with the **tl_get(chBlock)** API command. You can use this to look up what properties a block has.
After system preparation has finished, you can generate production code and use TargetLink’s logging and plotting capabilities. Since Simulink blocks are only prepared and not completely replaced as was previously the case, all Simulink properties such as the **Require all Inputs to have same data type** property are preserved. If necessary, you can customize system preparation using hook functions. For details, refer to Customizing system preparation, system clearance and system synchronization.

### Clearing TargetLink data from system

System clearance (replacing the former `tl2sl` reconversion) removes TargetLink data from a TargetLink model. System clearance always includes the synchronization of TargetLink and Simulink properties. As with system preparation, you can now specify precisely which properties are to be synchronized from TargetLink to Simulink and how this is carried out. For details on system clearance, refer to Reversion of System Preparation (TargetLink Production Code Generation Guide) and to How to Clear All TargetLink Data (TargetLink Production Code Generation Guide).

You can start the system clearance via the **Clear System** button on the **Tools** page of the TargetLink **Main Dialog**. The API command for calling the dialog is `tl_clear_system`. To display the individual options, type `help tl_clear_system` in the MATLAB Command Window. The old API command, `tl2sl`, is no longer available.
If necessary, you can customize system clearance using hook functions. For details, refer to Customizing system preparation, system clearance and system synchronization.

**System synchronization**

TargetLink 3.0 provides the opportunity to synchronize Simulink and TargetLink properties not just during system preparation and system clearance but at any time. This is useful because some properties in Simulink on one hand and TargetLink on the other hand have similar but not identical semantics (for example, data types in Simulink for MIL simulation, data types in TargetLink for code generation). System synchronization provides the means to synchronize such properties in both directions. Note that this does not apply to properties such as the gain value of a **Gain** block, which only exists as a Simulink property but can be accessed from TargetLink block dialogs and the TargetLink API. For details on system synchronization, refer to *Synchronization of Simulink and TargetLink Data* ([TargetLink Production Code Generation Guide](#)).

Synchronizing MinMax values as well as saturation flags between Simulink and TargetLink has to be thought through very carefully, since the semantics in Simulink and TargetLink are not identical. Both settings however have an important impact on the generated code.

The **System Synchronization** dialog is used for data synchronization, i.e., to select the properties to be synchronized, and the direction of synchronization (**Simulink to TargetLink** or **TargetLink to Simulink**), and to start synchronization.
You can start system synchronization via the **Sync System** button on the **Tools** page of the **TargetLink Main Dialog**. The API command for calling the dialog is `tl_sync_system`. To display the individual options, type `help tl_sync_system` in the MATLAB Command Window.

If necessary, you can customize system synchronization using hook functions. For details, refer to Customizing system preparation, system clearance and system synchronization.

### Customizing system preparation, system clearance and system synchronization

Previous TargetLink versions let you customize the conversion and reconversion process using pre- and post conversion hook functions. TargetLink still provides a set of hook functions in the form of M scripts that let you influence TargetLink’s default settings for system preparation, system clearance and system synchronization individually. This is why the names and functionalities of the former hook functions have changed.

These are now called
- `tl_pre_preparation_hook.m`
- `tl_pre_clear_hook.m`
- `tl_post_clear_hook.m`
- `tl_post_preparation_hook.m`
The commands `pre_conversion_hook`, `tl_pre_reconversion_hook`, `post_conversion_hook` and `tl_post_reconversion_hook` are no longer available.

There are various hook function template implementations under `<TL_Root>\Matlab\Tl\config\blocklib`. TargetLink searches the working directory and all directories registered in the global configuration directory (`tl_get_config_path`) for hook functions whose file names contain the naming fragment `*_hook.m`. As a parameter, each hook function always has the handle to the subsystem that is currently to be prepared. The scripts do not take effect unless modified. TargetLink defaults are used instead. The templates supplied with TargetLink contain some commented out M code lines that explain the procedure.

Because there are separate scripts for user-specific synchronization of Simulink and TargetLink properties, which is always a part of preparation, it is also recommended to transfer some of the functionality of the former hook functions and to the synchronization scripts.

### Separation of simulation-relevant and code-relevant data

The new blockset design distinguishes between Simulink data and TargetLink data, and stores them separately. Simulation-relevant data is stored in the Simulink blocks under the TargetLink mask, while code-relevant data is stored in the `data` mask variable as a cell array of strings with property/value pairs.

**Example** If you use the TargetLink block dialog to adjust the value of `Gain`, the modified value is passed directly to the Simulink `Gain` value under the TargetLink mask. The `Gain` variable is present only at the actual Simulink block. The TargetLink API commands have therefore been modified so that using `tl_set(blockh, 'Gain.value', '10')` calls the Simulink API with `set_param`. In contrast, a property that is relevant to code generation, such as the specification of the `DISP` variable class for the output of a block, is stored in the TargetLink mask variable `data`.

### Related topics

- **Basics**
  - `New Blockset Features` on page 68
  - `Preparing Simulink Systems for TargetLink`
TargetLink Port Blocks Discontinued

The previous TargetLink ports (with the exception of AUTOSAR) no longer exist in the new blockset design. Instead, TargetLink ports have been merged with Simulink ports (such as ref, pos and uPi shown below) which are available in any subsystem anyway.

Simulink ports already present in a model or inserted in it later can be converted into TargetLink ports manually. To do so, first select the block(s), and then select Enhance block via the Tools - TargetLink menu or call it via the API command tl_enhance_block.
If you use scripts that insert ports automatically, you must not take ports from the tlib. If your script formerly inserted both Simulink and TargetLink ports, you can simply insert the enhanced TargetLink port. If your script inserted only a TargetLink port directly before or after a Simulink port, you must now enhance the port.

TargetLink ports that were previously used for rescaling within a subsystem can now be replaced by the Rescaler block or by a virtual subsystem with a prepared import.

Modified TargetLink Blocks

TargetLink supports several MATLAB-dependent variants of the Prelookup block and of the Interpolation Using Prelookup block. There are different versions for the combination of Prelookup index search/Interpolation block.

- There will still be S-functions according to the old blockset design for the particular TargetLink variant, with common variables for index and fraction. If you selected the option for variable sharing in this block, the variant will be preserved in the model upgrade.

- If you work under R2006a, there is a variant with directly masked Simulink S-functions. However, the directly masked Simulink S-functions are out-of-date in terms of Simulink and should no longer be used in Simulink 6.5 or higher.

- If you work under R2006b or higher, TargetLink supports the new variant of Pre-lookup/Interpolation blocks which operates with separate signals for index and fraction.
Scalar Expansion and Block Output Width

Width property

The **Width** is no longer a separate TargetLink block data item, i.e., it is not a constituent of the actual TargetLink data. The width of the vector in the TargetLink dialog does not have to be specified explicitly, but results automatically from the width of the Simulink signal.

As you know, the TargetLink dialogs always display the simulated ranges after simulation has been performed. Since it is not possible to iterate through the individual components of the vector if the **Scalar expansion** checkbox is set, it is always the maximum or the minimum value across all the components of a vector that is displayed for the simulated ranges. This is particularly useful because when all components have the same scaling, this value is definitive for simulation-based scaling.

If the individual components of a vector have different scalings or saturations, you must clear the **Scalar expansion** checkbox and specify the appropriate width. Then you must set the **LSB**, **Offset**, **Saturate** checkboxes, and **Min** and **Max** values for each component individually.

However, the width can still be read and written via the TargetLink API commands `tl_get(gca, 'output.width')` and `tl_set(gca, 'output.width', '2')` respectively, which is particularly useful when the **Scalar expansion** checkbox is cleared.
The \texttt{tl_set(gcb, 'output.width','-1')} API command specifies that the port dimension at the Simulink port under the TargetLink mask is set to -1, which means inherit.

### Defining the Relay block output width for scalar inputs

The Relay block can output a vectorized scalar output signal even if the input is a pure scalar. In TargetLink 2.x, this was done by explicitly specifying the output width in the TargetLink dialog to provide the Relay block output width for scalar input. In TargetLink 3.0, the output width no longer exists as a separate data item. Instead, the Relay block output width for scalar input is defined by the dimension of the OnValue and OffValue variables.

If the input is scalar and the OnValue and OffValue variables are vectorized, the Relay block outputs a vectorized output signal with the width of the OnValue and OffValue variables.

### No scalar expansion for Direct Look-Up Table block

It is no longer possible to model a Direct Look-Up Table block capable of scalar expansion at the block output. The block output always has the same width as the index input, as it does in Simulink. As a workaround, you can use a Gain block to expand the signal again.

### Related topics

- Basics
  - Signal Width Inheritance on page 76
Replacement for TargetLink Port Blocks Used for Rescaling

Wherever a TargetLink port block was used for rescaling, measurement, logging, etc., within a subsystem in previous TargetLink versions, it is obviously not possible to use a Simulink port. The new, redesigned blockset therefore has a Rescaler block that is based on a masked Simulink Data Type Conversion block and has to be used instead. The TargetLink-specific data (for rescaling information) is stored in the data mask variable.

If a Simulink model contains a Simulink Data Type Conversion block, this block is not converted into a TargetLink Rescaler block by the system preparation process and is completely ignored by the Code Generator during code generation. This behavior is correct, as a Simulink Data Type Conversion block has different semantics for the Code Generator than a TargetLink Rescaler block. A TargetLink Rescaler block contains TargetLink-specific data and causes the Code Generator to rescale the signal. The TargetLink Rescaler block and the associated TargetLink block dialog are not bus-capable, i.e., the block cannot be used for rescaling the individual signals of a bus as was previously possible with the TargetLink Bus ports. As a workaround in such cases, you must insert a (virtual) subsystem, and rescaling must be performed in the prepared Simulink port.
Logging Behavior Changes

MIL Handler block
Whenever signals are to be logged or SIL/PIL simulations are to be performed with TargetLink 3.0, it is mandatory to have one instance of the new introduced MIL Handler block in your TargetLink model. One (and only one) MIL Handler block must be located at the root level of a TargetLink model and is automatically inserted during the model upgrade if logging is set in the model.

Logging speed in MIL simulation mode
With the new logging concept, the simulation speed depends on:
- The number of logged signals (fewer signals mean greater speed)
- Most particularly, the specified plot interval (defining the time intervals at which the plot display is updated while simulation is running)

To speed up simulation, it is therefore recommended to select a large plot interval for updating the display of logged signals online. The default for the plot interval was changed from 0.1 sec to Inf for TargetLink 3.0. Modifying the plot interval changes the refresh rate for the display, but does not affect the simulation results or the number of logged data points.
### No logging of states of the Discrete State-Space block

Logging of states of a **Discrete State-Space** block is not supported in TargetLink 3.0. This was possible in previous TargetLink versions.

![Discrete State-Space block](image)

### State logging at the FIR filter is not supported

With TargetLink 3.0, it is not possible to log states, i.e., states with simulated ranges cannot be scaled as part of autoscaling. There are therefore no simulated ranges at the delay line of the FIR Filter block.

### Loggable bus port signals

For bus port blocks, you cannot specify the maximum number of values to be logged for each bus element, but only for the whole signal. A warning is currently issued if the bus port elements have different settings for `MaxNumValues` and the lowest value for all bus port elements is evaluated.

### Logging signals from Bus Selector blocks

The capturing of Simulink log data fails if one of the signals to be logged comes from a Bus Selector block and the bus signal was generated by a Mux block. Simulink suppresses this action and issues the following error message:

> Cannot find signal `<signalname>` in input signal of Bus Selector block `<block>`. This may indicate that a Mux block in this model is used to create a bus signal. To identify Mux blocks used to create bus signals, set the 'Mux blocks used to create bus signals' option to 'warning' or 'error' in the Connectivity pane of the Diagnostics page in the Configuration Parameters Dialog.
Try to replace the identified Mux blocks with Bus Creator blocks manually or use the `slreplace_mux` command to eliminate this error. As a solution, you must adapt your model accordingly, for example, by replacing the Bus Selector block by a Demux block.

**Logging of signals with identical names**

The logging of signals that have the same name can fail in some cases if the label is specified directly at the source block. It can happen that TargetLink blocks interrupt this behavior. This situation cannot be clearly detected by querying the propagated signals. A different label that is assigned at a later point (for example, outside the same library blocks) is ignored. Inserting Simulink gains or logical blocks also does not change the behavior. To make simulation possible, you can use a TargetLink LogicalOperator block. Simply evaluating the propagated signals does not indicate clearly whether Simulink can log the signal or not.

**No direct logging of inputs to a Merge block**

In TargetLink 3.0, the direct input signals of a Merge block cannot be logged, as was possible in previous versions. This is a limitation in Simulink that now also applies to TargetLink. If logging is activated for the input signals of a Merge block, TargetLink simply ignores this and outputs an error message. In rare cases, if a signal cannot be traced back sufficiently without model initialization, it can happen that the simulation aborts if logging is activated. You can deactivate input signal logging for a Merge block manually in these cases. This limitation also applies if global logging for all signals in TargetLink subsystems is activated.

**Example**

Suppose your model provides two conditionally executed, enabled subsystems addressing a Merge block.
You cannot log the $u_{\text{high}}$ output variable of the Sum block located in subsystem $\text{high\_gain}$ directly. To log the Sum block and thus indirectly the input signals of a Merge block, you must activate logging for it and then add a further, nonvirtual block without simulation behavior (such as a Data Type Conversion block) downstream.

**Initial values of Merge blocks**

The initial values of Merge blocks can change if they are not specified `[ ]`. This can affect signal behaviors.

**Logging identical muxed signals**

If you log a muxed signal in which two signals are identical (because they are connected) and logging is activated at a succeeding outport (a virtual block), then TargetLink issues a warning during simulation and does not log the signal. To avoid this situation, you can place a **Data Type Conversion** block between the **Mux** block and the succeeding virtual block.
No more overflow checks for input signals

During MIL simulation in TargetLink 2.x, the input signals of look-up table blocks (such as the Look-up Table1D block) and the FIR block were checked for overflows and underflows, i.e., a check was made whether the values of the input signals fitted the value range (framed in red) of the data point vector. Overflows and underflows can occur, for example, if the input signal of a look-up table block exceeds the representable range of a data point vector resulting from its specified scaling. This overflow check on input signals is no longer performed in TargetLink 3.0.

In TargetLink 3.0, the Code Generator only issues a warning if the value range implemented for the output signal of a preceding block is greater than the value range implemented for the data point vector. This is the same as with TargetLink 2.x. TargetLink 3.0 also checks output signals for overflows, as was done under TargetLink 2.x.

Changed step size determination for variable step solvers

If variable step solvers are used in the simulation of TargetLink models, the step size of the variable step solver used in Simulink can differ between TargetLink 2.x and TargetLink 3.0. This means that the logging behavior in TargetLink differs from that in TargetLink 2.x for the following solver settings:

- Variable-step
- ode45 solver
The following model is an example.

If the solver type is ‘Variable-step’ and the solver is ‘ode45’, only one value is logged per pulse and the plot looks like this:

If fixed step and discrete (with fixed step size 0.1) are set instead, one value is logged every 0.1 s. The plot looks the same as in TargetLink 2.x.
Loggable Stateflow objects

In TargetLink 3.0, the following Stateflow objects can be logged:
- Stateflow objects with output scope
- Stateflow objects with local scope

It is no longer possible to log constants or variables with scope Imported or Exported.

Logging Stateflow objects with local scope

When Stateflow objects are logged with “local” scope, in some circumstances it can happen that changes to the logging settings are not updated until after the first simulation run. As a result, the logging data of Stateflow objects with “local” scope is not yet available when logging was just activated. You can remedy this by restarting or repeating the simulation. The problem no longer occurs if a local object was already logged once, even if logging was deactivated for the local object in the meantime.

The following use cases are affected:
- A new model with Stateflow local objects to be logged is simulated for the first time.
- A new local object to be logged is generated in an existing model.
- Local objects are logged for the first time in an existing model (e.g. by switching global logging to “Log signal histories”).

In these cases, you are given a warning informing you of the problem and its solution.
| No more Stateflow logger blocks | In TargetLink Blockset Version 2.2.1 and earlier versions, the following two blocks were available specifically for logging in Stateflow:  
- Stateflow Logger  
- Stateflow FcnCallLogger  
These blocks no longer exist in the new blockset, as their functionality is covered by the modified logging behavior. |
|---|---|
| Modified logging behavior at the Stateflow-Simulink interface | Multiple write access to one and the same (chart) output during a single simulation time step can be implemented by Stateflow means, for example, by multiple activation of a Stateflow chart. If this output was logged by a downstream TargetLink port in TargetLink 2.x, but the logger S-function was executed at a point in time before the last write access to the variable, "only" an intermediate result was captured.  
With the new blockset design, it is always the last value that is recorded. |
| Logging Stateflow objects with local scope | When Stateflow objects are logged with local scope, in some circumstances it can happen that changes to the logging settings are not updated until after the first simulation run. As a result, the logging data of Stateflow objects with "local" scope is not yet available when logging was just activated. You can remedy this by restarting or repeating the simulation. The problem no longer occurs if a local object was already logged once, even if logging was deactivated for the local object in the meantime.  
The following use cases are affected:  
- A new model with Stateflow local objects to be logged is simulated for the first time.  
- A new local object to be logged is generated in an existing model.  
- Local objects are logged for the first time in an existing model (e.g. by switching global logging to "Log signal histories").  
In these cases, you are given a warning informing you of the problem and its solution. |
Logging referenced models

Referenced models can be logged in MIL only if the simulation start is implemented via TargetLink, for example, via the **Start simulation** button of the Model Referencing Control Center. This is necessary to ensure correct logging/overflow checks in referenced models, as the S-functions for the referenced models (which have test points for logging) must be built before the simulation starts. It is too late to set the test points during initialization, as they would not take effect until the next simulation.

![TargetLink Model Referencing: mdref_1.mdl.png](image)

**Related topics**
- **Basics**
- *Code-Relevant Stateflow Enhancements and Changes* on page 108

## Simulation Behavior Changes

### No Saturation in MIL simulation mode

The special TargetLink feature "Saturation in MIL simulation mode" is no longer available in TargetLink 3.0. In earlier TargetLink versions, the block output signal in TargetLink full-featured MIL simulation mode was saturated to the maximum value range if one of the Saturate checkboxes was selected. This behavior was not and is not present in Simulink MIL mode and in the TargetLink Blockset (stand-alone), and is no longer available in TargetLink 3.0 full-featured mode either.

Simulink MIL simulation, TargetLink Blockset and TargetLink Blockset (stand-alone) MIL simulations always have completely identical simulation behavior in TargetLink 3.0.
<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slight change in MIL arithmetic</td>
<td>Because the implementation of TargetLink blocks has changed, an MIL simulation with the TargetLink Base Suite blockset can differ very slightly from MIL simulation with the TargetLink Base Suite blockset in previous versions. These differences arise in the area of calculation imprecision in floating-point arithmetic. TargetLink 3.0 in full-featured mode as well as stand-alone mode now exactly reflects the Simulink behavior.</td>
</tr>
<tr>
<td>Different input signals at Merge blocks</td>
<td>If several active signals arrive at a Merge block, it is now not always the same signal that gets through in MIL simulation mode. In TargetLink 3.0, you can change the names of the driving blocks to change the behavior. It was not possible to produce different behavior in TargetLink 2.x.</td>
</tr>
</tbody>
</table>
| MATLAB dependent MIL simulation             | Under MATLAB R2006a(+), when MIL simulation is started by `sim('ModelName')`, no `logsout` data structure is written to the workspace, so TargetLink's logging mechanism cannot save data to the TargetLink Data Server (TLDS). From MATLAB R2006b, this is no longer a problem. As a workaround under MATLAB R2006a, you can call the `sim` command together with the simulation end time and a simulation data structure. **Example for MyModel.mdl**  
  ```matlab
  sim('MyModel', [],
  simset('DstWorkspace', 'base'))
  ```  
  If you omit `tEnd`, the model settings are used. It is sufficient to define the destination workspace. |
TargetLink

Code-Relevant Stateflow Enhancements and Changes

Different semantics for data types and scalings in Stateflow/TargetLink

The interpretation of TargetLink data types of Stateflow objects has changed in TargetLink 3.0. The Stateflow data item EGO has the int16 Stateflow simulation type (left), while the TargetLink data type is set to Uint16 (right). Up to now, the Stateflow data type was also used for code generation provided the Stateflow simulation data type was not set to double. With TargetLink 3.0, if a TargetLink data type has been set, it is always used for code generation. If no TargetLink data type is set, float64 is used as the default. The Stateflow data type in no longer used.

List of Stateflow properties that can be used by TargetLink directly, i.e., no separate TargetLink data exists.

<table>
<thead>
<tr>
<th>Stateflow Property</th>
<th>Name for access via TargetLink</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>contains the TargetLink block structure (only data unequal default) as a string</td>
</tr>
<tr>
<td>Name</td>
<td>sfname</td>
</tr>
<tr>
<td>Minimum</td>
<td>min</td>
</tr>
<tr>
<td>Maximum</td>
<td>max</td>
</tr>
<tr>
<td>Units</td>
<td>unit</td>
</tr>
<tr>
<td>Initial value</td>
<td>value</td>
</tr>
<tr>
<td>Size</td>
<td>width</td>
</tr>
</tbody>
</table>
Semantic change for the inherited data type

Setting the Inherited data type for Stateflow variables at the interface between Simulink and Stateflow is limited, both in the new blockset and in earlier TargetLink versions.

![Data type](image)

However, the Code Generator does not return an error in such cases. Up to now, the inherited Simulink data types were used for code generation provided that inheritance did not result in a double data type and that no TargetLink data type had been specified for code generation. With TargetLink 3.0, the default TargetLink type for Stateflow data (float64) is set if no other TargetLink type is set. If you want to change this, you must set the TargetLink data type explicitly.

Generating interface variables at the Simulink/Stateflow interface

In TargetLink 3.0, Stateflow input variables, i.e. Stateflow data with scope = input, have an additional TargetLink property called createinputvariable. This property determines whether a variable for the Stateflow input data is generated in the code (createinputvariable = 1) or the specifications of the preceding Simulink block are applied (createinputvariable = 0).

In TargetLink 2.x, the specifications made for additional TargetLink properties determined whether an extra variable was generated for the Stateflow input or not. In contrast, under TargetLink 3.0 this behavior is determined exclusively by the createinputvariable property. TargetLink 3.0’s upgrade routine automatically sets the createinputvariable property according to the old conventions, so that the behavior after the upgrade is identical to that under TargetLink 2.x.
If a Stateflow variable is generated because the `createinputvariable` is set to 1, the usual TargetLink optimization mechanisms are applied to the generated variable, i.e., it can be optimized out if the variable class states it to be erasable.
Modified code patterns for Stateflow integer data types

The Code Generator in TargetLink 2.x distinguishes between a pure integer (Stateflow data type integer, for example sf.sftype = int16) and a scaled integer (Stateflow data type single or double and TargetLink data type, for example sf.sftype = double, sf.type = Int16, sf.lab = 2^0, sf.offset = 0).

The generated code is different, as the following example shows:

For the pure integer and scaled integer variants, TargetLink 2.x code was generated which was semantically identical but different. As described above, TargetLink 3.0 always uses separate data types for MIL simulation on the one hand and for the SIL simulation with generated code on the other. This applies to both Simulink and Stateflow. TargetLink 3.0 therefore does not have a pure integer type, and always generates code as an unscaled integer. Thus, the code generated by TargetLink 2.x and TargetLink 3.0 can look different, though it is semantically identical, if integer data types were used in the Stateflow parts of TargetLink 2.x models.
Support of Stateflow start index

TargetLink supports the Stateflow start index and the resultant positive and negative start indices of the Stateflow plot channel for the TargetLink API (\texttt{tl\_set/tl\_get}) and the Property Manager. During TargetLink plotting, the indices \([\text{startindex} \ldots \text{startindex} + \text{size} - 1]\) are mapped to \([1 \ldots n]\). This is the same behavior as in TargetLink 2.x, except that you can now also use the Property Manager and the API to specify plot channels for Stateflow that use the Stateflow start index (include negative indices). The value -1 is ambiguous. TargetLink 3.0's new blockset interprets it as "plot all", although in Stateflow it could also mean "plot only channel -1". This produces a limitation so that plot channel -1 of a multi-element vector cannot be plotted exclusively.

Generating Switch-Case instructions

With TargetLink 3.0, Stateflow generates Switch-Case instructions instead of If-Then-Else cascades, as was the case with TargetLink 2.3. This is equivalent in terms of semantics but can lead to numerous code differences. The old behavior (If-Then-Else) cannot be produced with the new blockset.

Related topics

- Basics
- \texttt{Logging\ Behavior\ Changes} on page 98

Changes in TargetLink API Functions

Comparison of API changes

In TargetLink 3.0, some TargetLink API commands have different names and properties. The changes are listed in the table below.

<table>
<thead>
<tr>
<th>TargetLink 2.x</th>
<th>TargetLink 3.0</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{sl2tl}</td>
<td>\texttt{tl_prepare_system}</td>
<td>The command \texttt{sl2tl} no longer exists in TargetLink 3.0 and has been replaced by \texttt{tl_prepare_system}.</td>
</tr>
<tr>
<td>\texttt{tl_pre_conversion_hook}</td>
<td>\texttt{tl_pre_preparation_hook}</td>
<td>Actions before a model is transferred to the TargetLink sphere</td>
</tr>
<tr>
<td>\texttt{tl_post_conversion_hook}</td>
<td>\texttt{tl_post_preparation_hook}</td>
<td>These hook functions let you adapt the rules for defining TargetLink data type settings that are derived from Simulink settings during model preparation. These rule adaptations are also used when data type settings are equalized between TargetLink and Simulink.</td>
</tr>
<tr>
<td>\texttt{slib2tllib}</td>
<td>\texttt{tl_prepare_system}</td>
<td>The library conversion command \texttt{slib2tllib} no longer exists in TargetLink 3.0. Instead, the \texttt{tl_prepare_system} command is used for models as well as for libraries.</td>
</tr>
<tr>
<td>\texttt{tl2sl}</td>
<td>\texttt{tl_clear_system}</td>
<td>The former \texttt{tl2sl} command has been replaced by the \texttt{tl_clear_system} command with slightly different options.</td>
</tr>
<tr>
<td>\texttt{tl_pre_reconversion_hook}</td>
<td>\texttt{tl_pre_clear_hook}</td>
<td></td>
</tr>
</tbody>
</table>
Changes on TargetLink Block Properties

In TargetLink 3.0, some TargetLink block properties have different semantics and names. Accessing the old TargetLink 2.x names results in an error. To get a compact list of the TargetLink properties available for the current block (gcb), type `tl_get(gcb)` in the MATLAB Command Window. The changes are listed in the table below.

<table>
<thead>
<tr>
<th>TargetLink 2.x</th>
<th>TargetLink 3.0</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>tl_remap_*</td>
<td>tl_post_reconversion_hook</td>
<td>These hook functions let you adapt the rules for defining Simulink data type settings that are derived from TargetLink settings during the model clear operation. These rule adaptations are also used when data type settings are equalized between Simulink and TargetLink.</td>
</tr>
<tr>
<td>-</td>
<td>tl_post_clear_hook</td>
<td>Separate initiation of synchronization between Simulink and TargetLink properties. This function no longer exists in TargetLink 3.0 and has not been replaced.</td>
</tr>
<tr>
<td>tl_set_variable_widths</td>
<td>tl_sync_system</td>
<td></td>
</tr>
</tbody>
</table>

### Changed block properties and names

<table>
<thead>
<tr>
<th>TargetLink 2.x</th>
<th>TargetLink 3.0</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>output.width</td>
<td>output.width</td>
<td>Changed semantics. The function now returns a value of -1 if the scaling settings are used for an arbitrary number of vector elements of a variable. If you scale elements individually, the width is a defined number greater than zero, as it was before.</td>
</tr>
<tr>
<td>inheritscaling</td>
<td>output.inheritscaling</td>
<td>Assigns a property directly to the output variables. The property is available only in blocks that support the inheriting of data type and scaling, such as the Unit Delay block.</td>
</tr>
<tr>
<td>output.autoscalingmode</td>
<td>autoscaling.mode</td>
<td>Separate autoscaling setting</td>
</tr>
<tr>
<td>output.lowerheadroom</td>
<td>autoscaling.lowerheadroom</td>
<td>Separate autoscaling setting</td>
</tr>
<tr>
<td>output.upperheadroom</td>
<td>autoscaling.upperheadroom</td>
<td>Separate autoscaling setting</td>
</tr>
<tr>
<td>output.headroomunit</td>
<td>autoscaling.headroomunit</td>
<td>Separate autoscaling setting</td>
</tr>
<tr>
<td>output.freeze</td>
<td>autoscaling.keepoutputoffset</td>
<td>Separate autoscaling setting</td>
</tr>
<tr>
<td>output.keepdatatypetype</td>
<td>autoscaling.keepoutputtype</td>
<td>Separate autoscaling setting</td>
</tr>
<tr>
<td>&lt;parameter&gt;.freeze</td>
<td>autoscaling.keepparamoffset</td>
<td>Separate autoscaling setting</td>
</tr>
<tr>
<td>&lt;parameter&gt;.keepdatatypetype</td>
<td>autoscaling.keepparamtype</td>
<td>Separate autoscaling setting</td>
</tr>
<tr>
<td>scalingvalid</td>
<td>autoscaling.scalingva</td>
<td>This property for the cast-output flag no longer exists. Accessing it leads to an error.</td>
</tr>
<tr>
<td>output.castoutput</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
Various Migration Aspects

No more Cast output to TargetLink type property

The former TargetLink property **Cast output to TargetLink type** no longer exists in the new blockset design. The property was used exclusively to influence data types during MIL simulation. With TargetLink 3.0, however, these settings are made directly in the relevant Simulink dialogs or via the Simulink API. In some cases, it may be necessary to use a Simulink **Data Type Conversion** block because the former TargetLink block had a **Cast output to TargetLink type** checkbox but the corresponding Simulink block does not provide settings for the data type (this also depends on the Simulink version used). Write and read accesses to this property via the TargetLink API, for example, `tl_get(gob, 'output.castoutput')`, generate error messages.

The automatic model upgrade to TargetLink 3.0 can cause the MIL simulation data types to change. This may in some cases result in model initialization errors, if model initialization was not possible after reconversion under TargetLink 2.2. If a situation arises where a model has to be made initializable again, there are no generally valid rules for doing so. In most cases, such initialization errors can be avoided by adjusting the initial data types.
Simulink data types for the TargetLink D Flip-Flop block

Up to now, the TargetLink D Flip-Flop block accepted all Simulink data types at the D-Input port except fixed-point data types. At the same time, the Simulink data type double always appeared at both outputs of the block.

With TargetLink 3.0, however, the TargetLink D Flip-Flop behaves in exactly the same way as the corresponding Simulink D Flip-Flop. As a result, the block accepts only the Boolean and double data types at the D-Inport. The **Implement logic signals as boolean data (vs. double)** checkbox on the Configuration Parameters dialog defines the behavior of the TargetLink and Simulink D Flip-Flop blocks.

If the checkbox is selected, the TargetLink D Flip-Flop accepts only the Boolean data type at the D-Input and also outputs a Boolean data type at both outputs. If the checkbox is cleared, only double signals are accepted and output at the D-Input. As a result, after model upgrade to the new blockset design, it can happen that model initialization fails if the **Implement logic signals as boolean data (vs. double)** checkbox is selected and the D Flip-Flop is addressed by a data type other than double. The TargetLink (2.x) and the Simulink D Flip-Flop block can both handle arbitrary input signals, though the Simulink block can do so only if **Implement logic signals as boolean data (vs. double)** checkbox is cleared in the optimization settings. This behavior now also applies to the TargetLink block. Thus, it can happen that a model cannot be initialized after upgrading, because the signal pending at the data input is not a Boolean signal.

Code changes when using $L and $B at TargetLink ports

If the $B name macro is used in a TargetLink port under TargetLink 2.x and the upgrade option `useTL2xportnames` is not set, the block variables are usually renamed in the code when the Simulink and TargetLink ports are merged. This is because the $B name macro designates the current block name, which the upgrade process
changed when the TargetLink port was removed and the data was transferred to the Simulink port. If you want to prevent this happening, you can set the upgrade option `TL2xportnames`, which causes the TargetLink port name to be transferred to the corresponding Simulink port.

If a TargetLink port is linked to an RTOS message in multirate models, TargetLink uses the port name to create the message name. If the port name changes in the merge between Simulink and TargetLink ports, however, the message variable in the code is automatically given a different name under TargetLink 3.0.

In addition, the behavior for resolving the `$L` name macro for the block output variable of an outport block has also changed in TargetLink 3.0, due to the merging of TargetLink and Simulink ports. TargetLink 2.x always used the signal label after a block for resolving `$L`, but for a TargetLink outport block in TargetLink 3.0, the signal label pending at its input is used.

- `$L` name macro resolution has not changed for any other blocks.
- These changes can result in the names of the generated variables being changed in the code. In an extreme case, it could theoretically happen that code generation aborts because a variable name is used twice.

### $R$ macro has no effect on code file names

When two function blocks are specified with the same C code file name with a `$R` macro at the end of the name, TargetLink does not produce two code files for the two functions, but only one. The `$R` macro therefore has no effect on the code file name.

### Network lists not migrated in autoscaling

If the results of a worst-case value range calculation were saved as part of autoscaling under TargetLink 2.x, these values cannot be loaded under TargetLink 3.0. This is because the network list changes when Simulink and TargetLink ports are merged. If you try to load a network list file that was saved under TargetLink 2.x, an error message is output.

### Using PostLoadFcn for model modification

It is no longer possible to execute code that changes the model before the TargetLink PostLoad function. The reason is that PostLoadFcn checks whether an upgrade can be performed on the model. If this is the case, the upgrade is performed. Any changes made before the upgrade are either lost or they cannot be performed at all.
No mixing of Demux/Mux and Bus Creator/Bus Selector blocks  

In older TargetLink (and Simulink) versions, it was possible to combine vectors and buses fairly freely (by using Mux/Demux blocks and Bus Creator/Bus Selector blocks respectively), although The MathWorks expressly advised against this modeling style. There is a Simulink diagnostics option called **Mux blocks used to create bus signals** with three settings, "None", "Warning", and "Error", for defining whether this behavior is permitted.

This has now changed: At the start of simulation, TargetLink 3.0 checks whether the option **Mux blocks used to create bus signals** is set to "Error". If it is not, a warning is displayed and logging, overflow detection, and min/max logging for MIL (and TargetLink blocks outside TargetLink subsystems) are deactivated. The model setting is correctly automatically, for example, by adding a TargetLink Main dialog to the model or performing a model upgrade. If buses and vectors have been mixed in the model, an error message is issued when the simulation starts. As a remedy, Simulink provides an auxiliary function called **slreplace_mux** for replacing Mux blocks with buses. This is the recommended procedure.

Libmap format modified  
The format of libmaps has been modified, see *Replacement of Simulink Blocks* for details. To have TL2.x libmap files upgraded, type `tl_upgrade_libmapfile` in the MATLAB Command Window. All TL2.x libmap files are upgraded. Libmap files are searched for in directories returned by `tl_get_config_path`.
## Compatibility Information

<table>
<thead>
<tr>
<th>Where to go from here</th>
<th>Information in this section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported MATLAB Releases</td>
<td>119</td>
</tr>
<tr>
<td>Supported Operating Systems</td>
<td>120</td>
</tr>
</tbody>
</table>

### Supported MATLAB Releases

The RCP and HIL software, TargetLink, and ModelCompare in dSPACE Release 6.2 offer full compatibility with the following releases of MATLAB from The MathWorks:
- MATLAB R2008a
- MATLAB R2007b+
- MATLAB R2007a+
- MATLAB R2006b
- MATLAB R2006a+

Supported Operating Systems

The following table shows which software items in dSPACE Release 6.2 support which operating system:

<table>
<thead>
<tr>
<th>Operating System…</th>
<th>Is Supported By…</th>
<th>RCP &amp; HIL Software</th>
<th>TargetLink 3.0</th>
<th>CalDesk 2.0</th>
<th>Model Compare 2.0</th>
<th>SystemDesk 1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 2000 with Service Pack 4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Windows XP (32-bit version) with Service Pack 2 ¹</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Windows Vista (32-bit version) with Service Pack 1</td>
<td>X</td>
<td>X</td>
<td>–</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

¹ It is currently not recommended to use Service Pack 3. For the latest information on Windows XP with Service Pack 3, refer to http://www.dspace.com/goto?winxpsp3.

Windows XP Professional x64 Edition and Windows Vista (64-bit version) are not supported by dSPACE Release 6.2.

For an overview of operating systems supported by dSPACE Releases and products, refer to www.dspace.com/goto?os_compatibility.

Limitations when working with Windows Vista (32-bit)

- **MATLAB support** Under Windows Vista, the dSPACE software supports only MATLAB versions since MATLAB R2007a+.

- **Sleep mode not supported** The dSPACE software does not support Windows Vista’s sleep mode for power saving. When restarting the PC from the sleep mode, you must reboot it to restore communication with the dSPACE hardware.

To avoid the automatic sleep mode, you must disable it. Refer to How to Disable Windows Vista’s Sleep Mode (Software Installation and Management Guide).

- **Fast user switching not supported** The dSPACE software does not support the fast user switching feature of Windows Vista.
Closing dSPACE software before PC shutdown  
The modified shutdown procedure of Windows Vista causes some required processes to be aborted although they are still being used by dSPACE software. To avoid data loss, you must terminate the dSPACE software manually before performing a PC shutdown.

Allowing communication via additional firewall rules  
During installation of the dSPACE software, two additional Windows Vista firewall rules are automatically installed. The first rule allows communication with a dSPACE expansion box, for example, AutoBox. The second rule allows MotionDesk to receive motion data from a network channel.

The rules are created by the following commands:

```bash
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."
advfirewall firewall add rule name="dSPACE MotionDesk" program="%dspace_root%\MotionDesk\Bin\MotionDesk.exe" dir=in action=allow profile=any description="Allow dSPACE MotionDesk to receive motion data via network."
```

Connecting/disconnecting DCI-GSI1 via USB causes a bluescreen  
If you use a DCI-GSI1 in connection with the RTI Bypass Blockset, repeatedly connecting and disconnecting the DCI-GSI1 via USB to the host PC can cause a bluescreen on the host PC.
A
adapting
  TargetLink custom libraries 82
  TargetLink scripts 82
ASM Brake Hydraulics Blockset
  migrating 19
  new features 18
ASM Drivetrain Basic Blockset
  migrating 20
  new features 20
ASM Electric Components Blockset
  new features 21
ASM Engine Diesel Blockset
  migrating 22
  new features 22
ASM Engine Gasoline Basic Blockset
  migrating 24
ASM Engine Gasoline Blockset
  migrating 25
ASM Environment Blockset
  migrating 27
  new features 26
ASM Utilities
  migrating 30
ASM Vehicle Dynamics Blockset
  migrating 33
  new features 31
AutomationDesk
  new features 15
autoscaling
  network lists not migrated 116

B
benefits
  TargetLink blockset 70

C
Cast output to TargetLink type 114
  changes
    TargetLink API 112
  clearing
    TargetLink data from system 72
  code changes
    using name macros 115
  code generation
    for referenced models 77
  ControlDesk
    new features 39
  customizing
    system clearance 91
    system preparation 91
    system synchronization 91

D
DD project file
  upgrading 82
  defining
    Relay block output width 96
  Direct Lookup Table block
    no scalar expansion 96
  disabling
    model reference 78
  discontinued
    TargetLink port blocks 93
dSPACE Data Dictionary
  migration 42
dSPACE FlexRay Configuration Package
  new features 49

E
enabling
  model reference 78

G
general enhancements and changes 9

I
inheritance
  signal width 76
Interpolation Using Prelookup
  MATLAB 94

K
key features 11

L
libmap
  modified format 117
logging
  signals in referenced models 77
  TargetLink Sink block 80
  logging speed
    MIL simulation mode 98

M
MATLAB 94
  Merge block
  no direct logging of inputs 100
  migrating
    ASM Brake Hydraulics Blockset 19
    ASM Drivetrain Basic Blockset 20
    ASM Engine Diesel Blockset 22
    ASM Engine Gasoline Basic Blockset 24
    ASM Engine Gasoline Blockset 25
    ASM Environment Blockset 27
migrating
  ASM Utilities 30
  ASM Vehicle Dynamics Blockset 33
migration
dSPACE Data Dictionary 42
  Model Compare 52
  ModelDesk 55
  SystemDesk 65
Model Compare
  migration 52
  new features 51
  model modification
    PostLoadFcn 116
Model referencing
  TargetLink 77
  ModelDesk
    migration 55
    new features 55

N
name macros
  code changes 115
  network list
    not migrated 116
new features
  ASM Brake Hydraulics Blockset 18
  ASM Drivetrain Basic Blockset 20
  ASM Electric Components Blockset 21
  ASM Engine Diesel Blockset 22
  ASM Environment Blockset 26
  ASM Vehicle Dynamics Blockset 31
  AutomationDesk 15
  ControlDesk 39
dSPACE FlexRay Configuration Package 49
  Model Compare 51
  ModelDesk 55
  Real-Time Testing 15
  RTI 57
  RTI Bypass Blockset 59
  RTI CAN MultiMessage Blockset 61
  SystemDesk 63
  TargetLink 68

O
overflow checking
  for virtual blocks 80
overview
  TargetLink API changes 112
  TargetLink block property changes 113

P
PostLoadFcn 116