New Features and Migration

For TargetLink 1.2

March 2001
How to Contact dSPACE

Mail
dSPACE GmbH
Technologiepark 25
D-33100 Paderborn
Germany

Phone
++49-5251-1638-0

Fax
++49-5251-66529

E-mail
info@dspace.de

Web
http://www.dspace.de

Subscription to e-mail newsletter

Technical support
support@dspace.de

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 2000-2001 by:
dSPACE GmbH
Technologiepark 25
D-33100 Paderborn
Germany

This publication and the contents hereof are subject to change without notice. dSPACE GmbH makes no warranty of any kind with regard to this publication, including but not limited to the implied warranties of merchantability or fitness for a particular purpose.
dSPACE GmbH shall not be liable for errors contained herein or direct, indirect, special, incidental, or consequential damages in connection with the furnishing, performance, or use of this publication.

Brand names or product names are trademarks or registered trademarks of their respective companies or organizations.
Contents

About This Guide 5

Key Features of TargetLink 1.2 7
Floating Network Licenses ................................................................. 9
  How to Obtain Permanent TargetLink Floating Network Licenses ... 9
Supported MATLAB Platforms ......................................................... 12
Supported Targets ........................................................................... 13
Code Generator .............................................................................. 15
Stateflow Code Generation ............................................................. 18
External Function Call Connections ................................................. 20
Block Library .................................................................................... 21
Message Browser ............................................................................. 23
Property Manager ........................................................................... 24
TargetLink API ................................................................................. 26
ASAP2 File Generation .................................................................... 27
TargetLink Documentation Tool ....................................................... 28
Model Conversion Simulink <-> TargetLink ..................................... 29

Migrating from TargetLink 1.1 to 1.2 31
Upgrading the Project Configuration File ......................................... 32
Upgrading TargetLink Models .......................................................... 33
  How to Upgrade a Model Automatically ....................................... 34
  How to Upgrade a Library Manually ........................................... 36
  How to Upgrade a Model Non-Interactively ................................. 37
Upgrading Custom Code Blocks ...................................................... 37
Changes to Existing Features .......................................................... 39
  Changes to TargetLink API Properties ....................................... 41
  Changed Code Generator Options .............................................. 44

Last-Minute Information 47
  TargetLink Messages ................................................................... 47
  TargetLink Limitations ............................................................... 50
About This Guide

This document provides you with a brief overview of the major features coming with TargetLink 1.2.

New features and enhancements
For more detailed descriptions of the features and a summary of the major enhancements made since TargetLink 1.1, refer to:
- Key Features of TargetLink 1.2 on page 7

Migration
You will also find information on the changes you have to perform when you migrate from TargetLink 1.1 to TargetLink 1.2. For more information, refer to:
- Migrating from TargetLink 1.1 to 1.2 on page 31

Last-minute information
For information on last-minute changes of TargetLink version 1.2, refer to:
- Last-Minute Information on page 47
Legend

The following symbols are used in this document.

⚠️ Warnings provide indispensable information to avoid severe damage to your system and/or your work.

💬 Notes provide important information that should be kept in mind.

💡 Tips show alternative and/or easier work methods.

🔍 Examples illustrate work methods and basic concepts, or provide ready-to-use templates.
Key Features of TargetLink 1.2

TargetLink 1.2 comes with the following new features, enhancements and changes. This list shows TargetLink 1.2’s new key features.

- **New Installation Features**
  - Floating network license (see Floating Network Licenses on page 9 for details)

- **New TargetLink Features**
  - Support for MATLAB 6.0 (R12) and MATLAB 5.3.x (R11). See Supported MATLAB Platforms on page 12 for details.
  - Additional supported targets. See Supported Targets on page 13 for details.
  - Optimization attributes for variable classes. See Code Generator on page 15 for details.
  - Support for Stateflow 3.0 and 4.0 (and 2.0). See Stateflow Code Generation on page 18 for details.
  - Support of external function call connections to a TargetLink subsystem. See External Function Call Connections on page 20 for details.
Key Features of TargetLink 1.2

- Additional and enhanced TargetLink blocks. See Block Library on page 21 for details.
- Graphical user interface for TargetLink messages. See Message Browser on page 23.
- Enhanced Property Manager. See Property Manager on page 24 for details.
- Additional TargetLink API features. See TargetLink API on page 26 for details.
- Enhanced ASAP2 file generation. See ASAP2 File Generation on page 27 for details.
- Enhanced Documentation Tool. See TargetLink Documentation Tool on page 28 for details.
- Enhanced model conversion Simulink <-> TargetLink. See Model Conversion Simulink <-> TargetLink on page 29 for details.
Floating Network Licenses

TargetLink 1.2 comes with a new licensing mechanism that is especially suitable for large development teams.

The various TargetLink modules are license-protected. Up to TargetLink 1.1, only single-user licenses were available. For Targetlink 1.2, you can also purchase floating network licenses that can be used on networked PCs on which TargetLink is installed (dSPACE License Clients). Via TCP/IP, the clients are connected to a PC that manages the corresponding dSPACE licenses (dSPACE License Server).

To provide suitable license management for floating network licenses, dSPACE uses the program FLEXlm, a product of GLOBE trotter Software, Inc. This allows you to easily install and configure the dSPACE License Server, which provides the available dSPACE licenses for the connected dSPACE License Clients. Since FLEXlm is also used by MATLAB and other products, it is often known and available in many development departments.

See Installing TargetLink in the TargetLink Production Code Generation Guide for instructions on the installation process.

For further information on the requirements and the handling of floating network licenses, see Managing Floating Network Licenses in the TargetLink Production Code Generation Guide for details.

How to Obtain Permanent TargetLink Floating Network Licenses

When you purchase floating network licenses for TargetLink 1.2, the initial licenses expire after 4 weeks. To obtain permanent licenses, you need to provide dSPACE with

- the dSPACE system ID, and
- the MAC address of the network adapter installed in the dSPACE License Server. This address serves as the system’s hardware ID.
To find the dSPACE system ID
1 On a dSPACE License Client PC with TargetLink installed, select Programs – dSPACE Tools – dSPACE License Manager from the Start menu to start the dSPACE License Manager.
2 Click the Show Licenses from license file button. The Show Licenses dialog opens, displaying all the available dSPACE licenses.
3 In the Product list, search for products with the comment Network license. The Dongle# column displays the corresponding dSPACE system ID.

You can also find the dSPACE system ID on the disk label of the Key-Disk.

Refer to the following instructions to get the dSPACE License Server's MAC address, and send it to dSPACE with the dSPACE system ID.

To find the dSPACE License Server's MAC address
1 On the dSPACE License Server, select Programs – dSPACE License Server – dSPACE Floating Network Utility from the Start menu to start the dSPACE Floating Network Licenses Utility. The Network Information dialog displays a list of MAC addresses for the available network adapters.
2 From the list, select the MAC address that corresponds to the network adapter used for communication with the dSPACE License Clients.

You can now send the dSPACE system ID and the MAC address to dSPACE.

To send the required information to dSPACE
1 In the Network Information dialog, click the Send e-mail button. The installed e-mail program appears, and an e-mail containing the MAC address opens.
2 Add the dSPACE system ID to the e-mail.
3 Send the e-mail to fnl@dspace.de.

Both the MAC address and the dSPACE system ID are required to provide permanent TargetLink licenses.

Press the Save to file button to save the MAC address to a text file for later use.
Supported MATLAB Platforms

TargetLink 1.2 supports the following MATLAB platforms:
- MATLAB 5.3.x (R11) with Simulink 3.0 and Stateflow 2.0
- MATLAB 5.3.x (R11) with Simulink 3.0 and Stateflow 3.0
- MATLAB 6.0 (R12) with Simulink 4.0 and Stateflow 4.0

MATLAB 5.3.x has not been released for Windows 2000. Therefore, you cannot install TargetLink for MATLAB 5.3.x under Windows 2000. However, MATLAB 5.3.x is fully supported for Windows NT/98/95 PCs.
Supported Targets

In addition to the Target Simulation Modules (TSMs) and Target Optimization Modules (TOMs) previously supported, the following modules are supported by TargetLink 1.2:

New target simulation modules
- Texas Instruments TMS470
- Mitsubishi M32R

New target optimization module
A new Target Optimization Module (TOM) is available for:
- Motorola HC12
- Mitsubishi M32R

The table below shows an overview of the targets’ characteristics:

<table>
<thead>
<tr>
<th>Processor</th>
<th>ANSI C Code Support</th>
<th>Target Optimization Module</th>
<th>Target Simulation Module</th>
<th>Refer to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motorola HC12</td>
<td>yes</td>
<td>for Cosmic Compiler</td>
<td>for Cosmic Compiler</td>
<td>Motorola HC12 in the TargetLink Reference</td>
</tr>
<tr>
<td>Mitsubishi M32R</td>
<td>yes</td>
<td>for GAIO Compiler</td>
<td>for GAIO Compiler</td>
<td>Mitsubishi Electric M32R in the TargetLink Reference</td>
</tr>
<tr>
<td>Texas Instruments TMS 470</td>
<td>yes</td>
<td>-/-</td>
<td>for Texas Instruments Compiler</td>
<td>Texas Instruments TMS470R1x in the TargetLink Reference</td>
</tr>
</tbody>
</table>

The links in this table work only for the targets that you installed.

Production code target simulation
The communication protocol between host PC and target evaluation board (EVB) was modified. Enhanced transmission error recognition and error handling improve communication, especially at high transfer rates. The simulation frame files have been restructured to simplify adaptation to customer-specific targets.

Target optimization module TriCore
Optimized Assembly functions for arithmetic operations with 64-bit intermediate results. This reduces the code size and execution time for applications with many 32-bit integer datatypes.
Key Features of TargetLink 1.2

New compiler for evaluation board

TargetLink 1.2 now also supports the following compilers:
- Green Hills PowerPC C Compiler V1.89
- Green Hills PowerPC C Compiler V3.0

See Motorola MPC5xx in the TargetLink Reference for further information.
Code Generator

This section shows the new features, enhancements and changes in TargetLink’s code generation.

New features

**External function calls** Function call trigger event lines may now pass the border of a TargetLink subsystem. By this means, subsystems residing inside a TargetLink subsystem can be configured as callable from external code (see Implementing Externally Function-Call Triggered Subsystems in the TargetLink Production Code Generation Guide for details).

**BaseType shortcuts** Shortcuts for the basic datatypes can be defined and used in variable name specifications by the $T$ macro (see Renaming Basic Datatypes in the TargetLink Production Code Generation Guide for details).

**User datatype Bool** In addition to the predefined ‘bool’ datatype, you may now specify your own Bool datatypes that are alternatively selectable for Boolean variables (see Defining Usertypes in the TargetLink Production Code Generation Guide for details).

**Variable class property ‘optimization’** The new variable class property 'optimization' allows you to influence the handling of variables in the optimization phase of the code generation (see optimization Field in the TargetLink Reference for details).

**Name macros** A couple of new naming macros have been introduced to provide even more flexibility in the specification of variable and function names: $C$, $E$, $F$, $R$, $T$. (see Configuring Names in the TargetLink Production Code Generation Guide for details).

Enhancements and changes

**Initialization and restart functions** The initialization of state variables of enabled subsystems and the initialization of variables for a system restart are now properly separated (see Specifying User-Defined Functions in the TargetLink Production Code Generation Guide for details).
New Info file entries    The info file contains some additional entries that reflect the new features of restart functions and external function calls (see FunctionHierarchy in the TargetLink Reference for details).

Range propagation and scaling    The calculation of ranges for variables is no longer limited to 64-bit integer arithmetic. The automatic scaling for intermediate results of operations in Stateflow has also been further improved (see Optimizing the Code by Range Propagation and Scaling Expressions in the TargetLink Production Code Generation Guide for details).

Special handling of value ‘inf’    If the code generator detects a relational operation (RelationalOperator, Min/Max, etc.) with one operand being +/- inf, the operation is simplified to a fixed Boolean constant (true/false), which can be the starting point for additional optimization. This allows modeling of variant coding by means of constants controlling, for example, conditionally executed model parts. Keep in mind that infinity is not handled within the range propagation algorithms (see Optimizing the Code by Range Propagation in the TargetLink Production Code Generation Guide for details).

Additional optimizations    The code generator’s capabilities for eliminating superfluous variables and optimizing the control flow has been improved. In particular, sequences of cascaded Switch blocks are now mapped to very efficient nested if statements (see Optimizing the Production Code in the TargetLink Production Code Generation Guide for details).

New advanced production code options    A couple of new options are selectable on the Advanced Page of the TargetLink Main Dialog (see Advanced Page in the TargetLink Reference for details):  

- Use global bitfields for Booleans  
  Global Boolean variables are implemented as bitfield components. This may significantly reduce RAM consumption.

- Share functions between TargetLink subsystems  
  Auxiliary functions are named without system-specific identifiers. For example, look-up functions generated for one TargetLink subsystem can be reused within another, which reduces ROM size.
Key Features of TargetLink 1.2

- **Omit initializations to zero in restart functions**
  Variables with an initial value of zero are not initialized in the RESTART function. Thus you can reduce the code size if there is already another function which clears the ECU RAM on startup.

- **Line break limit**
  The maximum number of characters per line in the output file is specified by this option.

- **Inlining threshold**
  The inlining function compares the cost of inlining a function against this value to decide whether to inline the function or not.

- **Functions for 64-bit operations**
  Functions are used to implement 64-bit operations rather than macros. This option further reduces the code size.
Stateflow Code Generation

This section shows the new features, enhancements and changes that were made in TargetLink's Stateflow code generation.

New features

Stateflow modeling

Stateflow 2.0, 3.0, and 4.0 supported TargetLink 1.2 fully supports the new modeling features introduced in Stateflow 3.0/4.0: subcharts, supertransitions, graphical functions, and temporal logic (see Stateflow Object Properties in the TargetLink Reference for details).

User-written functions TargetLink 1.2 supports calls to user-written functions in the action language. This Stateflow modeling feature was not supported in TargetLink 1.1. The interface description for the user-written functions is provided by a scripting mechanism similar to the user-lookup scripts (see User-Written Functions in the TargetLink Production Code Generation Guide for details).

TargetLink production code

Separate functions for subcharts In TargetLink 1.2, each subcharted state can become a separate function. You can specify the function name and the code file it is written to. In addition, you can specify function classes that contain information about the storage class, declaration prefix, linker section, etc. The function classes can be specified individually for the initialization, the state entry actions, the state during actions, the state exit actions, and for auxiliary functions (see Separate Functions for Subcharts in the TargetLink Production Code Generation Guide for details).

TargetLink properties for graphical functions In TargetLink 1.2, you can specify the function name and the code file the graphical function will be written to. In addition, you can specify function classes that contain information about the storage class, declaration prefix, linker section, etc. (see Graphical Functions in the TargetLink Production Code Generation Guide for details).
Key Features of TargetLink 1.2

Code Generator option ‘Multivalued state variables’  In TargetLink 1.2, the use of single-bit state representation and multivalued state variables can be customized via the new option Multivalued state variables on the Advanced page of the TargetLink Main Dialog. With multivalued state variables, TargetLink decodes the active states by SWITCH/CASE statements rather than by using IF/ELSE (see Optimize the Code for a State Machine in the TargetLink Production Code Generation Guide details).

Stateflow modeling

C-like bit operations  For Stateflow 3.0/4.0, models TargetLink 1.2 evaluates the native Stateflow option Enable C-like bit operations that is available individually for each chart. For these models the option on the Advanced page of the TargetLink Main Dialog is ignored. For Stateflow 2.0 models, the option on the Advanced page of the TargetLink Main Dialog is still evaluated (see Advanced Page in the TargetLink Reference for details).

TargetLink production code

Stateflow subsystem  TargetLink 1.2 introduces the new properties InitFunctionClass, StepFunctionClass, and AuxiliaryFunctionClass for charts. The properties StepFunctionClass and AuxiliaryFunctionClass replace the old FunctionClass property (see Separate Function for Charts in the TargetLink Production Code Generation Guide for details).

Code efficiency

Directed event broadcasts:  TargetLink 1.2 automatically generates separate functions for the receiver states of directed event broadcasts. This optimization leads to smaller and faster code. Directed event broadcasting is now the recommended means to model event logic and preferred to global event broadcasting (see Useful Tips in the TargetLink Production Code Generation Guide for details).
External Function Call Connections

TargetLink now supports external function call connections to a TargetLink subsystem. For example, you can model a special initialization function within the TargetLink subsystem and invoke it from outside of the system. This mechanism can also be used to implement step functions for different sample rates within one TargetLink subsystem. This feature is fully supported for all three of TargetLink's simulation modes (see Implementing Externally Function-Call Triggered Subsystems in the TargetLink Production Code Generation Guide for details).
Key Features of TargetLink 1.2

Block Library

This section shows the new features, enhancements and changes that were made in TargetLink’s block library.

New features

Discrete-Time Integrator block  The new Discrete-Time Integrator block allows you to use a discrete-time numerical integration algorithm in a TargetLink subsystem (see Discrete-Time Integrator Block in the TargetLink Reference for details).

Trigonometric Function  This block now supports all trigonometric functions for floating-point code generation (see Trigonometric Function Block in the TargetLink Reference for details).

Flip-Flop blocks  All four blocks of the Simulink Extras Library are now supported: D Latch, D Flip-Flop, S-R Flip-Flop, and J-K Flip-Flop (see TargetLink Simulation Blocks in the TargetLink Reference for details).

Function Call Generator block  You can use the Simulink Function Call Generator block in a TargetLink subsystem to invoke functions in a controlled sequence or to generate Events for Stateflow.

Custom Code block  There are several new features of the Custom Code block:

- The name of Custom Code variables in the generated code can be specified in the dialog.
- New code section to be placed in the generated production code header file („_HEADER_“).
- New code section to be placed in the RESTART function of the generated production code („_RESTART_“).
### Key Features of TargetLink 1.2

- New keywords to control the placement of custom code.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>top</td>
<td>at the beginning of a code section, for example the beginning of the step function</td>
</tr>
<tr>
<td>bottom</td>
<td>at the end of a code section</td>
</tr>
<tr>
<td>common</td>
<td>generate the code only once for all instances of a Custom Code block in the model</td>
</tr>
</tbody>
</table>

See *Inserting Custom Code with a Custom Code Block* in the *TargetLink Production Code Generation Guide* for details.

#### Enhancements and changes

**Merge block**
- There is a new dialog for the Merge block. Now you can specify all the attributes of the Merge variable, including name, class, and scaling factors. The code generation for the Merge block is significantly improved by eliminating auxiliary variables (see *Merge Block* in the *TargetLink Production Code Generation Guide* for details).

**Logical Operator and Relational Operator**
- The datatype of the block output variable can now be selected. Every Boolean datatype is available.

**Look-Up Table blocks**
- There are two new features here:
  - Look-up functions can be reused in other TargetLink subsystems if the option *Share functions between TL subsystems* is enabled on the Advanced page of the TargetLink Main Dialog. This option may reduce the code size significantly.
  - The number of generated look-up functions is reduced. The same look-up function is used when the table data consists of `const` and `volatile const` variables. TargetLink 1.1 generated two functions in that case.

See *Look-Up Table Block* and *Look-Up Table 2D Block* in the *TargetLink Reference* for details.

**Math block**
- The functions `sqrt`, `rem` and `mod` are now supported for fixed-point datatypes (see *Math Block* in the *TargetLink Reference* for details).
Message Browser

During code generation, model conversion, and autoscaling, you may receive a large number of notes, warnings, and error messages. The Message Browser was introduced as a graphical user interface to handle TargetLink messages. Using the new Message Browser, you can:

- view message texts conveniently
- open the dialog of the block which caused the problem
- show the affected block in the Simulink model
- exclude selected messages from the displayed list
- go straight to the online-help to get troubleshooting information
- create an error log file

See *The TargetLink Message Browser* in the *TargetLink Production Code Generation Guide* for details.)
Property Manager

There are several enhancements of the TargetLink Property Manager, which provides convenient graphical interface for handling large models.

**New features**

- **Hierarchy Filter**  The level currently selected in the Hierarchy Filter drop-down list is stored and applied when Property Manager is reopened.
- **New display mode**  Depending on the selected hierarchy level, TargetLink blocks and Stateflow objects are displayed or hidden. The current level is displayed above the Block Explorer.
- **Property visualization**  Invalid and read-only properties are visualized.
- **Inline editing**  Property values, block and Stateflow object labels can be edited by left-clicking on the current value or label.
- **Progress notification bar**  For lengthy operations such as Property Manager initializations or model loading operations, a progress notification bar and an Abort button are displayed.
- **Context menus**  Via the context menus of selected blocks, Stateflow objects or Block Explorer columns, you have access to various commands and functions.
- **Tooltips**  Tooltips are available for each Property Manager pane.
- **Autoscaling**  Autoscaling can now be performed subsystem- and block-wise.
Considerable speed improvements Opening a model in the Property Manager can take several minutes, depending on the size of the model (number of blocks, Stateflow objects, and properties to be displayed in the Block Explorer). However, with TargetLink 1.2, block properties are retrieved in their entirety and not property by property as with TargetLink 1.1. This avoids superfluous and time-consuming API checks.

See Using the TargetLink Property Manager in the TargetLink Production Code Generation Guide and TargetLink Property Manager in the TargetLink Reference for details).
TargetLink API

This section shows the new features that were added to TargetLink's API (application programming interface).

**New features**

**Stand-alone S-functions for TargetLink subsystems** With the utility `tl_build_stand_alone` you can create a stand-alone S-function for a TargetLink subsystem. This is useful to execute the production code generated by TargetLink in a pure Simulink model, independent of the TargetLink installation. If you place the stand-alone S-function in an RTI model, you can execute it on a dSPACE Rapid Prototyping system (see `tl_build_standalone` in the TargetLink Reference for details).

**Stand-alone S-functions for Custom Code blocks** When converting a TargetLink model back to Simulink, you can create stand-alone S-functions for your TargetLink Custom Code blocks with the command `tl_compile_custom_sfcn`. Thus you can use TargetLink's convenient Custom Code Interface to create your Simulink S-functions (see `tl_compile_custom_sfcn` in the TargetLink Reference for details).

**Search path for configuration files** You may define a special search path for your own project-specific hook functions (see `tl_get_config_path` in the TargetLink Reference for details).

**Multiple hook functions** In TargetLink 1.2, you can define hook functions with your own project-specific names. You can use hook functions to customize production code generation. For example, TargetLink searches for all hook functions which match the file name specification `*pre_codegen_hook*.m` and executes them before code generation (see Code Generation Related Utilities in the TargetLink Reference for details).

**Error log file** The new utility `tl_error_log` can be used to create a log file containing all error messages, warnings, and notes (see `tl_error_log` in the TargetLink Reference for details).
ASAP2 File Generation

This section shows the enhancements and changes that were made in TargetLink’s ASAP2 file generation (see Using the ASAP2 File Generator in the TargetLink Production Code Generation Guide for details).

Customization of ASAP1b interfaces  The entries made for ASAP1b interface specific data (IF_DATA) can now be fully customized. To create your own ASAP1b interfaces, you can provide a file if_data_asap1b_<name>.m in your working directory or on the search path for configuration files.

ASAP1b ETK interface  There is a new predefined ASAP1b interface for the ETK interface.

Mirror memory  In the ASAP2 template file, you can use the new function $tlasap2('GenerateDefinedPages') to create a DEFINED_PAGES entry in the ASAP2 file. The DEFINED_PAGES entry describes memory areas containing calibratable or measurable variables. They are used to define mirror memory on the host side of the calibration system, making ECU memory access more effective.

Common axis description  An entry for a common axis is now generated in the ASAP2 File, if the row or column axis data is not contained within the table structure when using user-written look-up functions. The following new keywords for axis data structures have been introduced:

- AXIS_X_STRUCT
- AXIS_Y_STRUCT.

New predefined variables  With the new pre-defined variables defaultAccuracy, defaultResolution, defaultFormatFloat32, and defaultFormatFloat64, you can control the resolution and display format of MEASUREMENTs and CHARACTERISTICs.
TargetLink Documentation Tool

This section shows the enhancements and changes that were made in the TargetLink Documentation Tool.

Enhancements and changes

Atomic subsystems  If a Simulink subsystem is declared as an Atomic Subsystem, it is documented like an inlined function, including an interface description and the list of variables belonging to the Atomic Subsystem (see Mapping of Subsystems to Functions or Code Blocks in the TargetLink Production Code Generation Guide and TargetLink Subsystems Command in the TargetLink Reference for details).

Stateflow book  During automatic document generation, the Stateflow Report Generator is invoked to create a Stateflow book, which describes the whole Stateflow machine. This feature is only available if the state machine does not contain any subcharts.

Hyperlinks  You can use hyperlinks within description strings or block comments. For example, if you enter 'Refer to $file:other.doc' in the description string of a block output, a hyperlink to <other.doc> will appear in the generated documentation. See the demo example fuelsys.

Besides $file:, the following strings are supported:

- $http:
- $ftp:
- $mailto:
- $wais:

Inserting files  By using the keyword $insert:file in a description string or block comment, you can instruct TargetLink to copy the contents of the given file to the generated document. Text files, image files, and MAT files are supported. See demo example fuelsys (see Automatic Document Generation in the TargetLink Production Code Generation Guide for details).
Model Conversion Simulink <-> TargetLink

This section shows the enhancements and changes that were made in TargetLink’s model conversion from Simulink <-> TargetLink.

- Problem detection has been considerably improved.
- Conversion is embedded in TargetLink’s message system, and messages during conversion are shown in the Message Browser, which provides direct access to blocks and online help.
- Automatic conversion of TargetLink Custom Code blocks to Simulink S-functions has been introduced.

See *Creating a Model by Conversion* in the *TargetLink Production Code Generation Guide* for details.

**Multiple Libmap files** In TargetLink 1.1 the conversion rules were described in the file `tl_get_libmap.m`. In TargetLink 1.2 you can create your own libmap files with project-specific names, for example, `my_utils_libmap.m` (see *Libmap File* in the *TargetLink Reference* for details).

**Mask evaluation** TargetLink 1.2 now supports the evaluation of mask parameters during model conversion, for example, in callback functions.
Migrating from TargetLink 1.1 to 1.2

After you install TargetLink 1.2 it is necessary to carry out the following steps:

- Upgrading the Project Configuration File on page 32
- Upgrading TargetLink Models on page 33

In addition, you should be aware of the following changes since TargetLink 1.1:

- Changes to Existing Features on page 39

Please see Last-Minute Information on page 47 for important last-minute information.
Upgrading the Project Configuration File

The project configuration file of TargetLink 1.2 (by default default_cfg.m located in \DSpace\ROOT\matlab\tl\config) contains some new settings, for example, the new optimization field of the variable classes and the shortcuts for basic datatypes.

Project configuration files created with TargetLink 1.1 can still be used in TargetLink 1.2 without changes. TargetLink assumes the following default settings for the missing entries:

```matlab
baseTypeName.Int8Cut = 'S8';
baseTypeName.UInt8Cut = 'U8';
...
class(i).optimization = {'movable'};
```

To take full advantage of the new TargetLink 1.2 features, it is recommended to upgrade the project configuration files to the new format.

Upgrading project configuration files is optional from TargetLink 1.1 and mandatory from version 1.0.

**tl_upgrade_projectfile**

To upgrade your project configuration file, you can use the upgrade tool `tl_upgrade_projectfile`.

**To convert the project configuration file from version 1.x to 1.2**

Type `tl_upgrade_projectfile('OldFile', <file_Vs1_x>, ...
'NewFile', <file_Vs1_2>)`

Where `<file_Vs1_x>` is the name of the version 1.0 or 1.1 project configuration file and `<file_Vs1_2>` the name of the version 1.2 project configuration file.

The names of the files can be specified relative to the working directory or with an absolute path name. If you invoke the tool with identical file names, a backup copy `<file_Vs1_x>.bak` is generated.
Suppose your old project configuration file is `myproj_cfg.m` and you want to convert it to the new file `myproj_1_2_cfg`:

```matlab
tl_upgrade_projectfile('OldFile','myproj_cfg',
                      'NewFile','myproj_1_2_cfg')
```

processes `myproj_cfg.m` (which must reside on the MATLAB search path or in the current working directory) and generates a new version 1.2 project configuration file `myproj_1_2_cfg.m` in your current working directory.

If you need to specify a subdirectory because the project configuration file is not on the MATLAB search path or in the current working directory, you can invoke the utility as shown below:

```matlab
tl_upgrade_projectfile(...
                      'OldFile', '.\myconfigs\myproj_cfg',
                      'NewFile', 'f:\proj\configs\myproj_1_2_cfg')
```

processes `myproj_cfg.m` in the `myconfigs` subdirectory, and generates a new version 1.2 project configuration file `myproj_1_2_cfg.m` in `f:\proj\configs`.

You can find out the current project name with the utility `tl_get_project` and set a new project name with `tl_set_project` (see Configuring the Project Name in the TargetLink Production Code Generation Guide and `tl_get_project` and `tl_set_project` in the TargetLink Reference for details).

### Upgrading TargetLink Models

Some new block properties have been introduced in TargetLink 1.2. In addition, new Stateflow objects types are supported. Therefore, all Simulink models that contain TargetLink blocks or Stateflow objects with TargetLink data and that were created with TargetLink 1.1 or earlier need to be upgraded if they are to be processed with TargetLink 1.2.
Migrating from TargetLink 1.1 to 1.2

If you skip upgrading, you will encounter numerous errors when you try to work with your non-upgraded model. This will happen, for example, if you try to open a TargetLink block dialog, start code generation, or invoke any of the TargetLink tools. Therefore, never continue to work with a model which is not upgraded properly.

You need a valid TargetLink 1.2 or 1.1-compatible project configuration file before any upgrading is performed (see Upgrading the Project Configuration File on page 32 for details).

How to Upgrade a Model Automatically

TargetLink 1.2 provides the upgrading tool \texttt{tl\_upgrade} to upgrade your model automatically. The tool is started when you open a model whose internal revision number does not match the current TargetLink revision number.

To upgrade a TargetLink model

1. Open the model that was created with TargetLink 1.0 or 1.1. This automatically starts \texttt{tl\_upgrade}.
2. Press the Upgrade now button to start the upgrade.
   This is strongly recommended because you will not be able to process the model with TargetLink 1.2 if it is not upgraded. During the upgrade, messages about which block is currently being processed are pasted to the MATLAB Command Window.
If you click the **Skip upgrading** button, you will be told how to invoke the upgrading tool manually at a later time.

3 For TargetLink **Custom Code** blocks, special upgrading rules apply. You are prompted to rebuild the associated Custom Code S-function.

Click the **Build now** button to generate and compile the Custom Code S-function.

If you click the **Do not build** button, the S-function will not be built. However, in this case you will not be able to simulate the Custom Code block in floating-point simulation mode.

See **Upgrading Custom Code Blocks on page 37** for details if your custom code file contains a non-empty section for initialization code (\_\_\_INIT\_\_\_).
How to Upgrade a Library Manually

If your model contains TargetLink blocks with library links, the blocks cannot be upgraded automatically because the link keeps the tool from writing data to the block. This applies if you designed your own libraries with blocks that contain TargetLink blocks. In this case, the Message Browser informs you that the libraries still need to be upgraded.

To upgrade libraries manually

1. Open the library.
2. Unlock the library by selecting Unlock library in the library window’s Edit menu.
3. Start the upgrading tool by invoking:
   ```markdown
tl_upgrade <library>
   ```
   in the MATLAB Command Window, where `<library>` is the library’s name. The library is now upgraded.
4. Save and close the library after the tool has finished.

The TargetLink data of all blocks in the library now comply with TargetLink 1.2.
How to Upgrade a Model Non-Interactively

Sometimes it is inconvenient to upgrade models interactively. This may be the case if you want to upgrade numerous models simultaneously by using a script that opens, upgrades and saves Simulink models automatically.

To upgrade a model non-interactively

1. Set TargetLink to the batch processing mode by typing:
   ```matlab
   tl_error_set batchmode on
   ```
   in the MATLAB Command Window.

2. Start the upgrading tool by invoking:
   ```matlab
   tl_upgrade <mymodel>
   ```
   in the MATLAB Command Window,

3. Disable the batch processing mode with the command:
   ```matlab
   tl_error_set batchmode off
   ```
   in the MATLAB Command Window.

In the batch processing mode, `tl_upgrade` runs through the upgrading process without having you answer message boxes.

Upgrading Custom Code Blocks

The custom code DLLs built with TargetLink 1.1 or earlier are not compatible with TargetLink 1.2. During model upgrade, you will be asked to recompile them. If you skip this during model upgrade, you must recompile the custom code DLLs manually from the Custom Code Block Dialog.
TargetLink 1.2 has a new \texttt{..._RESTART...} code section. This is not contained in old custom code files. If your block contains some initialization code, you will typically wish to execute the same initialization code in the \texttt{..._RESTART...} code section. Therefore, if TargetLink detects an old custom code file, it asks you if you wish to copy the \texttt{..._INIT...} code section to the new \texttt{..._RESTART...} code section.

In the production code, the \texttt{..._INIT...} code is executed only when the states of a subsystem must be reinitialized, for example, when an enabled subsystem becomes active.
Changes to Existing Features

This chapter describes changes you should be aware of when migrating from TargetLink 1.1 to TargetLink 1.2:

**Merging variables**
The identification of variables, i.e., the mapping of symbols from the block diagram to a single variable, has been simplified by the new variable class attribute `optimization = {'mergeable'}`. This specification corresponds to the `GLOBAL/EXTERN_GLOBAL` mechanism used in TargetLink 1.1. With the new capabilities, the check for identity is more restrictive. Now not only the name and type are checked, but the initial value, constrained range specifications and even the unit and description specifications are also checked for mismatch.

**Limit variable name length**
The algorithm to restrict identifiers to 31 characters has been modified. Variable names that are longer than 31 characters are mapped to different identifiers in TargetLink 1.2 than in TargetLink 1.1.

**Implementation of constants**
If constant values are compared against variables, the user is now warned if the constant cannot be represented exactly in the scaling of the variable. This helps to identify potential problems due to rounding or truncation effects.

**Merging Static and Extern variables**
TargetLink 1.2 now detects if variables with `static` and `extern` scope are merged, for example, if the `STATIC_CAL` and `EXTERN_CAL` variable classes are used for the two instances of the same variable. Make sure that you do not merge static and extern variables.

**$L name macro**
The `$L` macro (signal label name) is no longer supported for block parameters, for example, a Gain value. It can be used for block output variables only (see Configuring Names in the TargetLink Production Code Generation Guide for details).

**Multiport Switch**
The control input is no longer rounded, but truncated. Control input values less than 1 and greater than the number of inputs are not allowed.
The output of the Multiport-Switch block has an undefined value if the control output is outside the range \([1 .. \text{number of inputs}]\) (see Multiport Switch Block in the TargetLink Reference for details).

**Data logging of default variables**
Logging of variables with the 'default' variable class prevented the variable from being moved to conditional statements in TargetLink 1.1. In TargetLink 1.2, default variables are 'movable', i.e., the calculation of their values, and the logging of calculation, may be performed conditionally. Therefore, the simulation results may be different. Select the 'NOPT_GLOBAL' variable class to prevent the variable from being moved to a conditionally executed statement.

**Enabled TargetLink system**
With TargetLink 1.2, the TargetLink system itself may be an enabled or triggered system. While in TargetLink 1.1, the Enable block at the topmost level of a TargetLink system was replaced by a pseudo Enable block, this is no longer necessary. During model upgrade to TargetLink 1.2 the old representative is replaced by a standard Simulink Enable block.

**Look-Up Tables**
The Code Generation process for Look-Up Tables has been improved for TargetLink 1.2.

- In TargetLink versions 1.0 and 1.1, look-up tables with parameter variable classes that have different const and volatile attributes lead to different look-up table functions. In TargetLink 1.2, the same functions can be used and thus code size is reduced.

- RAM consumption is saved by reducing the datatype of the number-of-table-entries variable in the generated look-up table map from UInt16 to UInt8 when possible. That saves RAM size.

- The **Add boundary points** option is now disabled by default when you copy a Look-Up Table block (1D) from the TargetLink block library tllib.
Constrained limit disables saturation

If a constrained limit is given, the saturation flag is disabled. As a result of this, saturation can no longer be selected for the block output. No code is generated to saturate the output’s value to the given limit. In TargetLink versions 1.0 and 1.1, code was generated that saturated the value of the output to the limits given by the output variable’s scaling. To avoid overflow problems make sure that the constrained limits you specify are valid.

ASAP2 file generation

**Constant Blocks** TargetLink 1.2 now generates CHARACTERISTIC entries for Constant blocks in the ASAP2 file. Previous versions have generated MEASUREMENT entries for constants.

**Variable names** TargetLink 1.2 uses a different naming scheme for Stateflow variables (\$C macro). The algorithm which shortens the variable names to less than 31 characters was changed. This results in different variable names in the generated C code and the ASAP2 file compared to TargetLink 1.1.

**Changes to TargetLink API Properties**

This chapter highlights the changes made to the API properties since TargetLink 1.1. For a list of new API features, see TargetLink API on page 26.

Model properties

Model-specific TargetLink properties are kept in TargetLink Main Dialog blocks.

**codeopt.inline** This property is obsolete and replaced by the new property **codeopt.inliningthreshold**.

During upgrading, the new property is set as follows:

<table>
<thead>
<tr>
<th>TargetLink 1.1</th>
<th>TargetLink 1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>codeopt.inline</td>
<td>codeopt.inliningthreshold</td>
</tr>
<tr>
<td>0</td>
<td>inf</td>
</tr>
<tr>
<td>1</td>
<td>6 (= default value)</td>
</tr>
</tbody>
</table>

The new property **codeopt.inliningthreshold** specifies the threshold for inlining a cost function. If the inlining cost function exceeds this value, the function is not inlined.
The cost function is the number of statements multiplied by the number of instances of a function.

### Custom Code block properties

**name**  The meaning of this property has changed in TargetLink 1.2. As for other TargetLink blocks, name now describes the identifier of the variable in the generated code. The logical name within the custom code (former name property) is now contained in the new property varname. The new property has been introduced for each variable, i.e. input, output, state, param or work variables. The property name now specifies the identifier which should be generated for the associated variable in production code. According to TargetLink conventions, valid C identifiers which may contain common Targetlink name macros can be assigned to this property.

<table>
<thead>
<tr>
<th>TargetLink 1.1</th>
<th>TargetLink 1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>output(1).name</td>
<td>output.name</td>
</tr>
<tr>
<td>&lt;identifier&gt;</td>
<td>['$S_$<em>B</em>-' &lt;identifier&gt;]</td>
</tr>
<tr>
<td>e.g., 'y1'</td>
<td>e.g. '$S_$_B_y1'v</td>
</tr>
<tr>
<td></td>
<td>output.varname</td>
</tr>
<tr>
<td></td>
<td>&lt;identifier&gt;</td>
</tr>
<tr>
<td></td>
<td>e.g, 'y1'</td>
</tr>
</tbody>
</table>

The Custom Code block can have an arbitrary number of input, output, state, param, and work variables. In TargetLink 1.1, adding or removing a variable could only be performed via the block’s dialog. In TargetLink 1.2, you can also use the API:

**To add a variable by setting a property value**

- Type `tl_set(b, 'output(2).type', 'Int16')` to set a 2nd output.

This will set the datatype of the 2nd output of the Custom Code block denoted by `b` to 'Int16'. If there is only one output, a new one will be added.

**To remove an output variable**

- Type `tl_set(b, 'output(2).varname', '')`
By setting the `varname` property to an empty string, you remove the 2nd output.

**Look-up tables**

The default settings of the following properties have been changed:

<table>
<thead>
<tr>
<th>TargetLink 1.1</th>
<th>TargetLink 1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>inheritclass 1</td>
<td>inheritclass 0</td>
</tr>
<tr>
<td>extrapolate 'on' (1)</td>
<td>extrapolate 'off' (0)</td>
</tr>
</tbody>
</table>

During Simulink → TargetLink conversion, the `extrapolate` property is now left at the library default `'off'`. This can result in deviant simulation behavior after conversion, since Simulink look-up tables always extrapolate beyond table boundaries.

**Properties of Stateflow objects**

The descriptions of all Stateflow objects can now also be set with the TargetLink API.

- **functionclass** For Statecharts, this property is obsolete and replaced by the new property `stepfunctionclass`:

<table>
<thead>
<tr>
<th>TargetLink 1.1</th>
<th>TargetLink 1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>functionclass</td>
<td>stepfunctionclass</td>
</tr>
</tbody>
</table>
Migrating from TargetLink 1.1 to 1.2

Function block properties

<table>
<thead>
<tr>
<th>Function block properties</th>
<th>functionname</th>
<th>This property is obsolete and replaced by the new property stepfunctionname:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>TargetLink 1.1</th>
<th>TargetLink 1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>functionname</td>
<td>stepfunctionname</td>
</tr>
</tbody>
</table>

### Changed Code Generator Options

It is recommended to set the Code Generator options on the Code Generation Page or the Advanced page of the TargetLink Main Dialog. If you have set any options directly in the `tl_pre_codegen_hook` hook function, you have to adapt them for use with TargetLink 1.2. The following table describes the changes to Code Generator options.

<table>
<thead>
<tr>
<th>Code Generator Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OptimizeTables</td>
<td>This option is obsolete. The Code Generator does not reduce the number of table points automatically. Table data can be optimized prior to code generation using TargetLink’s Table Tool.</td>
</tr>
<tr>
<td>ControlFlowOptimization</td>
<td>This option is obsolete. In TargetLink 1.2 control flow optimization is performed only if the variable class attribute <code>class(i).optimization = {'movable'}</code> is set. This means that you can adjust the control flow optimization for each block output separately.</td>
</tr>
<tr>
<td>EnablePragma</td>
<td>This option is obsolete. Replaced by <code>EnableSections</code> option, which enables support for section names, #pragmas, interrupt functions, inline functions and user attributes.</td>
</tr>
<tr>
<td>UserLookupFunction</td>
<td>This option is obsolete. Support for user-written look-up functions is always enabled.</td>
</tr>
<tr>
<td>UseStateflowAutoscale</td>
<td>This option is obsolete. Stateflow autoscaling is always enabled.</td>
</tr>
</tbody>
</table>
UseStateActivityEncoding  This option is obsolete. It is replaced by the new option StateActivityEncodingLimit (default: 5). If the number of states in a chart is greater than this value, TargetLink decodes the active state by using SWITCH/CASE rather than IF/ELSE statements. StateActivityEncodingLimit = 1 corresponds to UseStateActivityEncoding = 'off'. StateActivityEncodingLimit = inf corresponds to UseStateActivityEncoding = 'on'.

OmitZeroInitializationsInRestartFunction  This is now a public option of the Code Generator (Default: 'off')

UseGlobalBitfields  This is now a public option of the Code Generator (Default: 'off')

StateflowUseBitfields  This is now a public option of the Code Generator (Default: 'on')

MaximumCostOfInlining InliningMode  This two options are now obsolete. Replaced by the new option InliningThreshold (Default: 6)

StateflowShortNames  This option is obsolete. Short names in Stateflow can be achieved with the new naming macros $S$, $R$

LineBreakLimit  This is now a public option of the Code Generator (Default: 100)
Last-Minute Information

This chapter provides information on changes and enhancements that occurred after finishing TargetLink Production Code Generation Guide and TargetLink Reference.

TargetLink Messages

This chapter contains information on the TargetLink messages that are not included in TargetLink Messages in the TargetLink Reference.

W03018 Discrete Integrator block <block> has a level-triggered external reset input, which is not supported by TargetLink. The reset input will be made either-edge triggered.

During Simulink to TargetLink conversion, a Discrete Integrator block was found with a level-triggered reset input. Because this is not supported by TargetLink, the reset input was made either-edged triggered during conversion.
N03432  **Integer wordwidth of parameter <variable> reduced**  
(<<previous_datatype>> -> <<new_datatype>>) and optimum LSB value(s) evaluated.

The integer wordwidth of parameter <variable> has been reduced, and LSBs have been evaluated according to its current value or adjustable limits.

**E12029**  
**<block> Currently, the 'bus selection mode' of block 'demux' is not supported.**

This mode is currently not supported.

➢ Do not use this mode.

**E12030**  
**The Simulink Merge block is no longer supported in this version of TargetLink. Use the TargetLink Merge block instead.**

TargetLink has its own Merge block.

➢ Replace the Simulink Merge block by the TargetLink Merge block.

**E21001**  
**<Block1> Name ambiguity of identifier <identifier>.**

<Variable/Function/Function parameter/Macro> was first declared for block <block2>. **<Explanation>.**

The identifier is not unique, which means there is a variable, function, function parameter or macro with the same identifier. This means the production code cannot be compiled.

➢ If you wish to have one code variable generated for multiple variables in the model, the following requirements must be met in addition to identical identifiers:

- the datatypes must be identical
- for parameters, the variable values must be identical
- the dimensions must be identical
- storage classifiers (const, volatile, etc.) must be identical
- all but one variable must be declared as extern, or the associated variable class must have the mergable attribute.

See **Variable Classes** in the **TargetLink Production Code Generation Guide** for details.
If you wish to have one function generated to hold code from several sources, the following requirements must be met in addition to identical identifiers:

- the function classes must be identical
- the function types must be identical

For function identifiers, this error might be caused by the initFunction attribute of a variable class that uses the specified name. See TargetLink Subsystems and Function Classes in the TargetLink Production Code Generation Guide for details.

**E23043** `<block>` Not yet implemented: `<associated parameters>` with different variable classes.

There are different variable classes for associated block parameters. This message can occur for the:

- onswitch, offswitch parameters of the TargetLink Relay block
- onoutput, offoutput parameters of the TargetLink Relay block
- coefficients of the TargetLink Discrete Filter block
- coefficients of the TargetLink Discrete Transfer Fcn block
- the matrices of the TargetLink Discrete State-Space block

➤ Select the same variable class for all associated parameters.

**W26009** `<block>` The scaling range of the input for the `<x-axis/row/column>` vector is wider than the scaling range of the vector. This might cause an overflow in the look-up function’s call.

The scaling range of the block’s input for the specified vector is wider than the scaling range of the vector. This might cause an overflow in the look-up function’s call.

➤ Choose the same scaling for the input and the vector, or specify a constrained range at the input.

**E28037** `<data>` The property `<property>` is not allowed for Stateflow data with scope `<sf-scope>`.

The TargetLink data tag embedded in the data’s description contains a property which is invalid for Stateflow data with this specific scope.

➤ Remove this property from the description string.
F31014  **Operation canceled by the user.**
The user pressed **CTRL+C**

### TargetLink Limitations

This chapter contain information on the TargetLink limitations that are not included in TargetLink Limitations in the TargetLink Production Code Generation Guide.

**Function calls**

In floating-point mode, outputs in function call triggered subsystems are initialized with 0 if no initial output value is explicitly given in the subsystem. The simulation could therefore differ from Simulink and production code.

**Watcom Version 11 MEX compiler**

Exception handling is not supported for the Watcom Ver. 11 MEX compiler. For example, an integer division by zero will result in an application fault when the production code S-function is simulated. Exceptions may also occur during simulations if you use fixed variable addresses. Earlier versions of the Watcom compiler allow the generation of exception handlers.

**Frame generation**

A simulation frame for functions generated for externally function-called subsystems can only be generated if input and output arguments are globals which are also inputs and outputs of the main root function. If this constraint is not met, a frame generation error is issued.

**Simulation frame and enable ports**

In floating-point simulation mode, enabled blocks which reside on the topmost level of a TargetLink subsystem and which show their output port do not emit the enable signal which has been passed to the simulation frame, but the signal is always equal to 1. In production code simulation mode, the signal is set to a correct value. This is a limitation inherent in TargetLink’s simulation frame, which enables switching between simulation modes.
Custom Code variables with class ‘default’ cannot be used in code sections placed in different code modules. The code for the common section of TargetLink Custom Code blocks is always placed in the file containing the function for the TargetLink root system. If Custom Code is placed in another file and the same Custom Code template uses variables that have the ‘default’ variable class or are specified as ‘local’, these variables cannot be accessed from the Custom Code’s common section.

Bit operations with scaled operand (Lsb != 1.0 or Offset != 0.0) are not supported by TargetLink.

Discrete-Time Integrator block
- A Discrete-Time Integrator block cannot reside in an enabled or triggered subsystem if you want to generate code for it.
- The TargetLink Discrete-Time Integrator block does not support the ‘show state port’ property from the Simulink Discrete-Time Integrator.
- The TargetLink Discrete-Time Integrator block does not support the new ‘level’ property for ‘external reset’ of the Simulink Discrete-Time Integrator from MATLAB 6.0 (R12).
- Using vectorized signals with the TargetLink Discrete-Time Integrator block is not supported yet.

D Flip-Flop block
The behavior of the TargetLink D Flip-Flop block is different from the corresponding Simulink block if it resides in an enabled subsystem with the enable port setting StatesWhenEnabling set to ‘held’. In this case, the outputs of the TargetLink D Flip-Flop is not reset when the block is disabled, while the output of Simulink’s Flip-Flop block is reset.

J-K Flip-Flop block
The behavior of the TargetLink J-K Flip-Flop block is different from the corresponding Simulink block if it resides in an enabled subsystem which is disabled when the simulation is being started. In this case, the Q output of the Simulink block is not Boolean, but equal to the initial condition until the subsystem is enabled. This is a Simulink bug because the block’s initial output should be 0 if the initial condition is 0, and 1 if it is not 0.

Merge block
The output signal widths of blocks that precede TargetLink Merge blocks must be identical to allow production code generation.
The Simulink Merge block property **Allow unequal port widths** is currently not supported by TargetLink’s Merge block.

**Enable block**

For all Simulink outports in direct nested conditional subsystems, the **Output when disabled** drop-down list must be set to “reset” for all outports or to “hold” for all outports. If the settings are mixed, production code simulation differs from floating point.

D Flip-Flop and Discrete-Time Integrator blocks, and triggered Stateflow systems, are also conditional systems with held outports.